BILLING OPERATIONS GUIDE

KBP-11.5-BOG-2003-08-31

KENAN®/BP

Note: Kenan/BP and Kenan/OM are the products formerly known as Arbor/BP and Arbor/OM.

Contents

Tables

Figures

Notational Conventions

Note Important notes appear in this format.

Caution Indicates possible danger to data, software, or hardware.

Warning! Indicates serious risk of damage to data, software, or hardware.

Table 1 Notational Conventions

Notation	Explanation of Convention
References to printed documents	*Helvetica italic* **Example:** See *Database Reference Volume 2*.
<KEYS>	UPPERCASE HELVETICA, in angle brackets **Example:** Press <CTRL><Q><SHIFT><P> to create an em dash.
Buttons and *icon names*	*Helvetica italic* **Example:** Click *OK* to apply the chosen action.
`User-entered text`	**`Courier bold`** **Example:** Enter **`Total Charges`** in the field.
`Placeholders for user-determined text`	*`Courier italic`*, in angle brackets **Example:** Enter your *`<password>`*.
`Code samples, TABLE_NAMES, field_names, file and directory names, file contents, user names, passwords, UNIX ENVIRONMENT_VARIABLES`	`Courier`
Placeholders for system-generated text	*Helvetica italic* **Example:** Messages appear in this form: *timestamp messageID* >> *text.*
Menu items	**Helvetica bold** **Example:** Choose **Reports** from the main menu.

Additional Documentation

The Kenan/BP documentation set for release 11.5 consists of the following manuals:

- *ARCH Guide*
  ARCH-1.2-AG-2003-08-31

  Instructions for using the Archiver module to archive, restore, count, and delete data from Kenan/BP database tables.
- *Database Reference (Administration Tables, Vol. 1)*
  KBP-11.5-DRA1-2003-08-31

  Database Reference (Administration Tables, Vol. 2)
  KBP-11.5-DRA2-2003-08-31

  Detailed description of fields and tables in the Admin database.
- *Database Reference (Catalog and Customer Tables)*
  KBP-11.5-DRCC-2003-08-31

  Detailed description of fields and tables in the Catalog and Customer databases.
- *Kenan/BP Highlights*
  KBP-11.5-Highlights-2003-08-31

  Overview of Kenan/BP functionality
- *Order Services Guide*

  KBP-OSG-1.0-2003-08-31

  Instructions for using the Orders Services component for the generation, management, and fulfillment of orders and sub-orders.
- *Software Documentation Update*
  KBP-11.5-SDU-2003-08-31

  Detailed description of the enhancements in the current release of Kenan/BP.
- *System Administrator Guide*
  KBP-11.5-SAG-2003-08-31

  Installation, configuration, and maintenance instructions for the Kenan/BP system.
- *Technical Reference*
  KBP-11.5-TR-2003-08-31

  Detailed descriptions of inputs, outputs, and processing flow of all modules.
- *Billing Operations Guide*
  KBP-11.5-BOG-2003-08-31

  Operation and maintenance instructions for modules.

The Kenan/BP documentation set for release 11.0 consists of the following manuals:

- *Advanced Collections Guide*
  KBP-11.0-ACG-2002-10-30

  Instructions on using the Advanced Collections module to identify and aid in the collection of delinquent debts.
- *Common Configuration Tasks*
  KBP-11.0-CCT-2002-10-30

  Detailed instructions for configuring data model entities.
- *Documentation Overview*
  KBP-11.0-DO-2002-10-30

  Describes the documentation for Kenan/BP.

- *Guide to Products, Rates, and Discounts*
  KBP-11.0-GPRD-2002-10-30

  Overview of data model entities (such as products and rate schedules).
- *Journals Guide*
  KBP-11.0-JG-2002-10-30

  Detailed description of Journals module functionality.
- *Module Configuration Guide*
  KBP-11.0-MCG-2002-10-30

  Detailed instructions for configuring modules.
- *Reports and File Layouts*
  KBP-11.0-RFL-2002-10-30

  Detailed descriptions of control reports, summary and detail reports, and input and output file formats.
- *Software Documentation Update*

  KBP-11.0-SDU-2002-10-30

  Details changes in this release.

Additionally, *Release Notes* may be available for your deployment, containing additional information not covered in other documents.

You may also need to refer to documentation for other products, such as operating systems, database management systems, tax packages, and third-party software integrated with Kenan/BP.

1 Introduction

Welcome to the CSG Kenan/BP *Billing Operations Guide*. This document explains how to schedule, run, and monitor the modules that make up the CSG Kenan/BP billing system.

This document is intended primarily for Kenan/BP *billing operators* — that is, for users responsible for scheduling and running the Kenan/BP modules that retrieve usage, calculate balances, generate invoices, execute collections events, process payments and perform other "back-end" billing tasks. (These users may have a different designation in your organization.)

This document may also be useful to anyone interested in how the various modules that make up Kenan/BP work together to perform billing tasks as well as to system administrators who monitor system activity and ensure that the network and databases are functioning properly. (System administrator tasks are covered primarily in the *System Administrator Guide*, however.)

This document assumes familiarity with certain concepts and tasks common to billing operations. See "What You Should Know" on page 2 for more information.

Note Your deployment may or may not include all or some of the select modules listed in this document. Therefore, you can disregard sections of this document that do not pertain to your deployment.

What This Document Includes

This document explains how to schedule and run Kenan/BP billing operations modules — that is, the modules that retrieve usage, calculate balances, generate invoices, execute collections events, process payments, and perform other billing tasks.

More specifically, this document includes the following types of information for billing operations modules:

- general purpose and function of each module
- how to schedule tasks to fit your organization's billing needs
- expected outputs for each module
- how to investigate and repair failed transactions

This document assumes that Kenan/BP has already been installed and that Kenan/BP modules have already been properly configured for use in your deployment. Consequently, this document does not include the following information:

- installation and system configuration instructions — see the *System Administrator Guide*
- initial setup instructions for Kenan/BP modules — see the *Guide to Products, Rates, and Discounts* and the *Module Configuration Guide Module Configuration Guide*
- instructions for building your own modules — see the *API TS Guide*

- detailed descriptions of module process flows — see the *Technical Reference*
- instructions for using Event Time Unit Credits — see the *System Administrator Guide*

Finally, if your deployment includes custom modules for specialized tasks these modules are not described in this document — see your site-specific documentation.

What You Should Know

Before you start scheduling and running Kenan/BP modules you should become familiar with the following information:

- Basic UNIX commands

  Sometimes, you can run modules directly from the UNIX shell prompt or through Operations Center using UNIX syntax. First you should understand the basic tools for navigating a directory structure (**`ls`**, **`cd`**, and so forth) and running UNIX executables.

  The precise syntax for these operations can vary depending on your system configuration. Contact your local UNIX system administrator for this information.
- Basic SQL commands

  Sometimes, you may have to access the Kenan/BP database directly using **`isql`** or a similar SQL tool or through Operations Center using SQL syntax. First you should understand the basic concepts underlying inserting and selecting rows in relational database tables and the associated SQL syntax for your relational database system. Contact your local database administrator for this information.
- Overall Kenan/BP structure
  What each module does and how the modules relate to one another.
  A general overview of this structure appears in chapter 2, "Billing Operations Overview," and is available in other documents and through Kenan/BP training.
- Operation of Operations Center
  The interface through which you schedule and configure tasks for most Kenan/BP modules. This interface is referred to in chapter 3, "Operating Kenan/BP," and described in detail in the *Operations Center Guide*.
- Specific task modes, configuration options, and scheduling concerns related to the modules you will run
  If you are responsible for running only a single set of modules refer directly to the corresponding chapters (see "How This Document Is Organized," on page 2) and study them in depth.

  If you are responsible for running most or all of the Kenan/BP modules, read this document once all the way through and refer to specific sections in more depth as necessary.

What's New in This Manual

If you are familiar with the Billing Operations Guide for Kenan/BP release 11.0, you should be aware of the following changes. This is not a complete list.

- Customer Center replaces the Customer Information (CustInfo) interface. Customer Center now offers support for Ordering and Billing in the same interface.
- API Transaction Set replaces Kenan/BP C APIs.

- Multiview Data Model. In previous versions of Kenan/BP, products and services were represented at a single moment in time. The new data model for version 11.5 enables not only storage of current information about the status of a product or service, but also of information on its past status and pending changes.
- Restructuring of Large Call Data Record and Billing tables. The `CDR_DATA` and `BILL_EQUIP_DETAIL` tables are heavily restructured and broken down into a family of smaller tables, to:
  - » eliminate duplication of information between the two large tables
  - » make database maintenance easier and improve performance
  - » enable the use of partition keys for database maintenance in Oracle installations

Other changes are listed by chapter.

Chapter 3

Command Center has been replaced by the Operations Center. The Operations Center is an enterprise-wide solution to ensure the availability, performance, and recovery of mission-critical applications running in a geographically distributed environment. All Command Center material has been removed. See the *Operations Center Guide* for more information on scheduling and monitoring processes with Operations Center.

Chapter 4

- There is a new MPS module, the Long Term Persistence (LTP) module. CAP no longer inserts guided and rated usage records into the database directly, but writes them to a flat file. LTP writes them to the database from this file.
- CAP now calculates some taxes on usage charges during rating. CAP does not calculate any binned taxes. CAP uses the same tax packages as BIP does, and BIP includes the CAP-calculated taxes normally on invoices and handles adjustments.

Chapter 5

Kenan/BP now supports embedded bitmapped graphics in BIF PCL files.

Chapter 6

- All payments modules now insert payments or clearinghouse responses into the database in batches from payment or response files instead of one at a time to improve performance.
- LBX has similar functionality to CPM and EFT in processing payments that exceed an account's credit balance.

Chapter 7

AMP has new recovery functionality that allows it to roll back partially moved accounts if it is interrupted during processing. Other improvements have also been made to AMP to allow it to move tables more efficiently if they contain large amounts of repeated data.

2 Billing Operations Overview

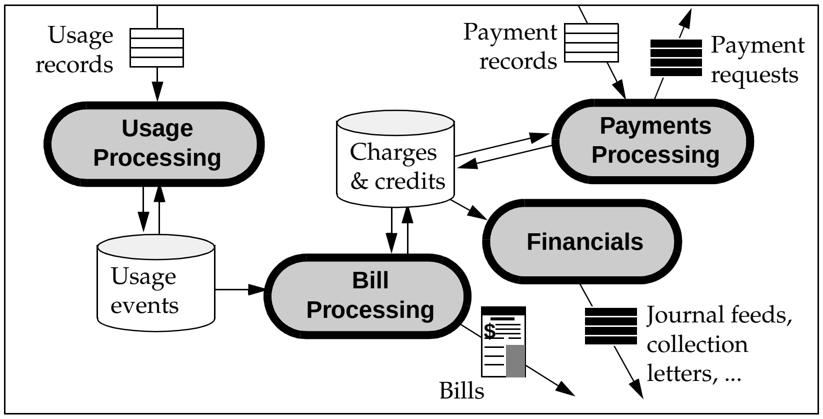
This chapter provides an overview of billing operations tasks: the types of tasks each Kenan/BP module performs and the issues associated with scheduling modules to work together.

See chapter 3 for instructions on running Kenan/BP modules.

Overview

Kenan/BP modules perform billing operations tasks in several major categories as figure 1 illustrates.

Figure 1 Billing Operations Subsystems



Usage processing tasks (see “Usage Processing Modules” on page 7) include:

- Retrieve records of usage events such as phone calls, pay-per-view movies, or e-mail from network mediation layers.
- Calculate usage charges for each usage event.
- Route usage records for external clients to external clearinghouses.
- Store usage records for local customers in the local database.

Payments processing tasks (see “Payment Processing Modules” on page 9) include:

- Initiate payments of customer balances from financial clearinghouses (for example, credit card clearinghouses).
- Retrieve records of payments from banks and financial clearinghouses.
- Associate payment amounts with the specific charges that were paid.

Bill processing tasks (see “Bill Processing Modules” on page 11) include:

- Calculate customer balances based on charges and credits (for example: service charges, monthly fees, usage charges, taxes, late fees, adjustments, discounts, and payments).
- Arrange charges and credits on formatted customer bills.
- Dispatch customer bills.

Financials tasks (see “Financials Modules” on page 12) include:

- Perform collections activities against delinquent accounts or bills.
- Book financial transactions to General Ledger systems.

Several additional modules allow billing operators to perform a variety of administrative tasks. See “Administration Modules” on page 13 for more information.

Table 2 on page 15 summarizes the modules available in each category. Billing operators configure and schedule these modules through the Operations Center. See the *Operations Center Guide* for more information.

Usage Processing Modules

The usage processing modules (sometimes called the Message Processing System or *MPS*) receive usage records from a network mediation layer, associate them with accounts or external systems, calculate usage charges, and (if desired) export corresponding records to external systems such as resellers.

MPS supports both real-time and batch processing. These options are described separately in the following sections.

Real-Time Usage Processing

Real-time usage processing requires the following modules:

- A custom *usage record generator* places records of usage events on an interprocess message queue.
- The *Usage Router* process, called *MCAP*, retrieves those usage records, rates them, and writes rated usage records to the Kenan/BP database.

If necessary use the following modules to correct failed usage:

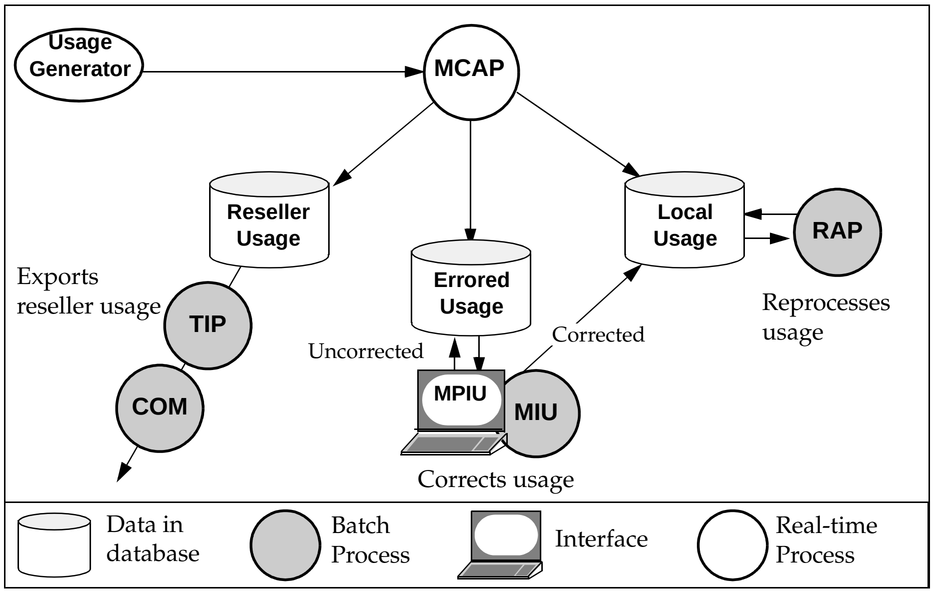
- The *Message Processing Investigation Unit* (*MPIU*) interface reviews, corrects, and reprocesses failed records.
- The *Batch Message Investigation Unit* (*MIU*) reprocesses batches of corrected records.
- The *Usage Reprocessor* (*RAP*) reprocesses records to reflect configuration changes (for example, new usage rates).

If desired use the following modules to export usage to external systems:

- The *File Translator* (*TIP*) module creates exportable files of usage records in appropriate file formats.
- The *Communications* (*COM*) module transfers those files to external systems.

Figure 2 illustrates how these modules work together to process usage.

Figure 2 Usage Processing (Real-Time)



See chapter 7, “Administration,” for more detailed information about COM, and see chapter 4, “Usage Processing,” for more detailed information about the other real-time MPS modules.

Batch Usage Processing

Batch usage processing requires the following modules:

- The Communications (COM) process retrieves files of usage records from a network mediation layer.
- The Usage Router (MCAP) process routes usage records to the proper server.
- The *Usage Guider/Rater* (*CAP*) process rates usage records and associates them with accounts or roaming clearinghouses.
- The *Long Term Persistence* (LTP) module commits rated usage data to the Kenan/BP database.

If necessary use the File Translator (TIP) process to convert usage files into a format MCAP and CAP can understand.

If necessary use the following modules to correct failed usage:

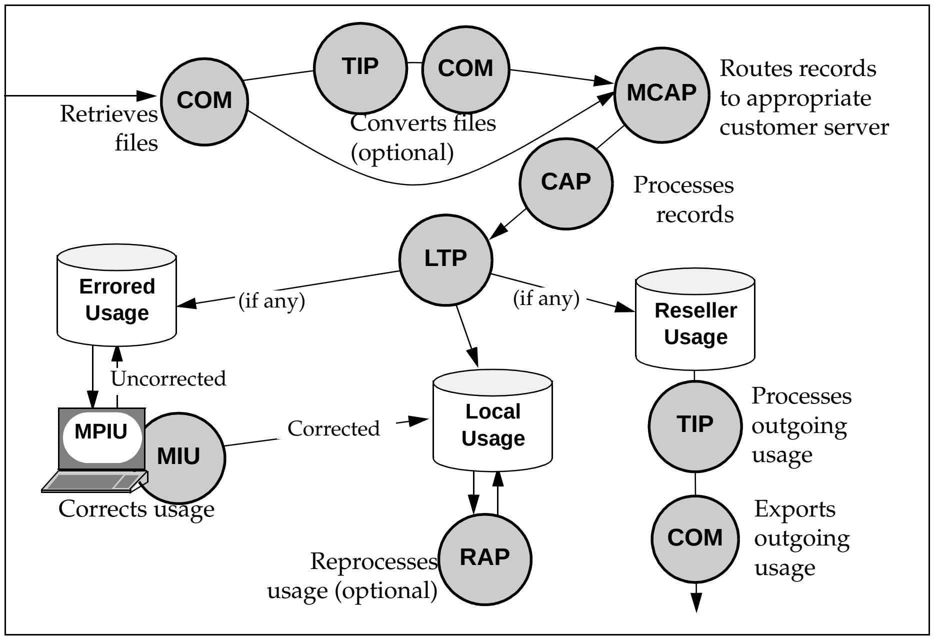
- The Message Processing Investigation Unit (MPIU) interface reviews, corrects, and reprocesses failed records.
- The Batch Message Investigation Unit (MIU) reprocesses batches of corrected records.
- The Usage Reprocessor (RAP) reprocesses records to reflect configuration changes (for example, new usage rates).

If desired use the following modules to export usage to external systems:

- The File Translator (TIP) module creates exportable files of usage records in appropriate file formats.
- The Communications (COM) process transfers those files to external systems.

Figure 3 illustrates how these modules work together to process usage.

Figure 3 Usage Processing (Batch)



Payment Processing Modules

The payments processing subsystem consists of the following modules:

- *Communications* (COM) — exchanges payment files and payment request files with external systems
- *File Translator* (TIP) — converts payment files from one format to another
- *Lockbox Payments (LBX*) — processes check payments from bank lockboxes
- *Credit Card Payments* (*CPM*) — creates payment request files to be handled by credit card clearinghouses, processes acknowledgment files returned by those clearinghouses, and in some cases automatically posts payments as received
- *Electronic Funds Transfer* (*EFT*) — creates payment request files to be handled by direct-debit clearinghouses, processes acknowledgment files returned by those clearinghouses, and in some cases automatically posts payments as received

- *LBX, CPM, and EFT Investigation Units* (*LIU*, *CCIU*, and *EIU*, respectively) — review and (sometimes) correct failed payment transactions
- *Real-Time Credit Card (RTCC) server* — allows custom applications to validate and authorize transactions with credit card clearinghouses in real time

  RTCC can verify accounts and earmark funds but cannot capture or transfer those funds.

See chapter 6, “Payments Processing,” for more detailed information about LBX, LIU, CPM, CCIU, EFT, EIU, and RTCC. See chapter 7, “Administration,” for more detailed information about COM and TIP.

In addition, the Payments interface allows customer service representatives to enter payment amounts manually. Because manual payment entry is primarily a customer care rather than a billing operations task it is not described in this document; see the *User Guide* for more information.

Figure 4 illustrates the interactions between payments modules, banks, clearinghouses, and information stored in the Kenan/BP database. (TIP and RTCC are not included in figure 4.)

Figure 4 Payments Processing

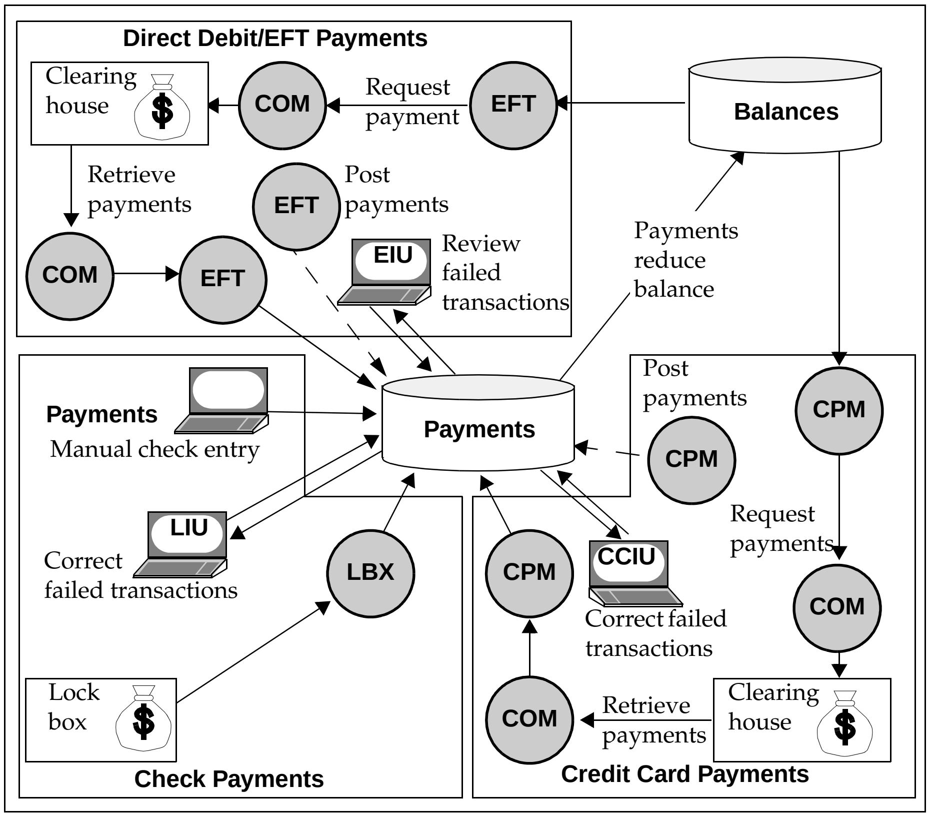


Figure 4 illustrates several important facts about payments:

- Kenan/BP supports three distinct payment methods (check, credit card, and direct debit/EFT).

  Your deployment may support some or all of these methods. For example, if all payments enter Kenan/BP manually through the Payments interface you do not have to schedule or run any of these modules.

- CPM and EFT perform two separate tasks — requesting payments based on customer balances and subsequently either processing or posting those payments — and depend on COM to exchange files with clearinghouses. LBX only processes payments.

The relationship among payment requests, retrieved payments, and calculated balances depends on the specific needs of your deployment and affects the number of tasks you must configure and the order in which you should schedule them. See “Scheduling Batch Processes,” starting on page 18 for more information.

Bill Processing Modules

The bill processing modules calculate charges and credits, arrange them on formatted bills, and dispatch those bills to customers. These modules support both real-time and batch processing. The real-time method is recommended in most cases.

Real-time bill processing involves the following four steps:

1. A customer service representative (CSR) requests a bill for a specific account through the Customer Care interface.
2. The *Bill Preparer* (*BIP*) collects and calculates charges and credits for that account. In some cases BIP uses third-party tax packages to calculate tax charges.
3. The *Bill Formatter* (*BIF*) or *Invoice Generator* arranges those charges and credits on a formatted bill.
4. The Customer Care interface notifies the CSR that the bill is available.

This process occurs in seconds without human intervention.

Batch bill processing involves the following four steps:

1. BIP collects and calculates charges and credits for all accounts scheduled for billing. In some cases BIP uses third-party tax packages to calculate tax charges.
2. The *Historic Discount Processor* (*HDP*) calculates discounts and paybacks under historic contracts based on contributions collected by BIP.

   Note HDP runs in batch mode only.

3. Either BIF or the *Invoice Generator* arranges those charges and credits on a formatted bill.

   (The Invoice Designer interface defines the formats used by the invoice generator. See the Invoice Designer *Guide to Invoice Design* for more information.)
4. The *Bill Dispatcher* (*BID*) dispatches bills to a printer or other distribution device.

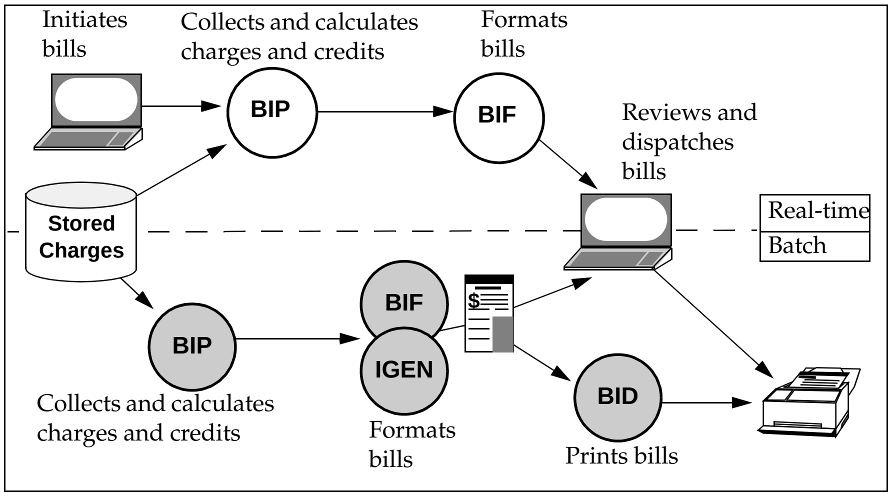
   In some cases (for example, e-mail bills) formatted bills are dispatched directly to customers, so BID is not used.

The BIP, BIF or Invoice Generator, and BID modules must be scheduled to run at specific times to perform batch bill processing.

Real-time and batch bill processing can occur side-by-side on the same system producing regular, scheduled bills as well as on-demand interim bills. Kenan/BP tracks the billed status of each credit and charge so it appears only once.

Figure 5 illustrates how the bill processing modules work together. See chapter 5, "Bill Processing"for more information about these modules.

Figure 5 Bill Processing


Initiates bills
Collects and calculates charges and credits
BIP
Formats bills
BIF
Reviews and dispatches bills
Stored Charges
Real-time
Batch
BIF
IGEN
BIP
BID
Collects and calculates charges and credits
Formats bills
Prints bills

Financials Modules

The financials subsystem consists of two independent modules.

- *Journals* (*JNL*) — books charges, adjustments, payments, and other financial transactions for accounting purposes

  JNL has four distinct modes or *run types.* These largely perform the following tasks:

  » Run type 1 books transactions that have been billed.

  » Run type 2 books transactions not yet billed.

  » Run type 3 summarizes booked transactions.

  » Run type 4 produces journal files for use by General Ledger systems.

  See chapter 8 for more information about JNL.

- *Collections* — handles collections activity for delinquent accounts and invoices. Collections is documented in the *Collections Guide*.

  Collections performs eight distinct tasks:

  » *Treat* moves items into collections if they meet certain criteria. For example, Collections (Treat) might move an account into collections if the unpaid balance exceeds a certain amount.

  » *Event Treat* moves items into collections in response to certain events. For example, Collections (Event Treat) might move an account into collections if a customer's credit card payment is rejected by the clearinghouse.

  » *Queue* schedules activity for collectables in collections. For example, Collections (Queue) might instruct CSRs to call customers or write off outstanding debts.

  » *Shuffle* checks to see if collectables are no longer in the correct collections scenario as a result of changes in collectable attributes and moves them to a configurable point in a new scenario. For example, Collections (Shuffle) might move a collectable into a new scenario if the account status changed from VIP to normal.

  » *Promise Queue* updates the status of promises to pay.

» *Cure* moves collectables out of collections if they meet certain criteria. For example, Collections (Cure) might move an account out of collections if the unpaid balance drops below a certain amount.

» *Assertive Cure* forces the removal of a collectable from collections regardless of its current progress in collections or outstanding balance.

» *Re-Treat* removes a collectable from collections and runs Treat on it. Treat may move the item back into collections in a different scenario.

Administration Modules

Billing operators commonly run the following modules regularly to perform administrative tasks:

- Utility (*UTL*) — executes any given UNIX command

  UTL is most commonly used to schedule utilities provided with Kenan/BP, but you can also use it to schedule your own scripts, standard UNIX shell commands, or other executables.
- Account Mover (*AMP*) — moves accounts from one server to another
- Archiver (*ARCH*) — copies records from key database tables to archive files, or restores records from those files
- Manual Replication (*RepStat*) — replicates tables from one server to another
- Usage Point Loader (*LOAD_USG_PTS*) — populates `USAGE_POINTS` and related tables required for location-based usage rating
- Refresh Jurisdiction Utility (*RJU*) — updates product jurisdictions for jurisdiction-based recurring charge rating

You can also think of COM and TIP as administration modules that perform the general-purpose functions of exchanging and translating files. COM and TIP are discussed as part of the usage (see page 7) and payments (see page 9) processing systems.

See chapter 7, “Administration,” for more information about these modules.

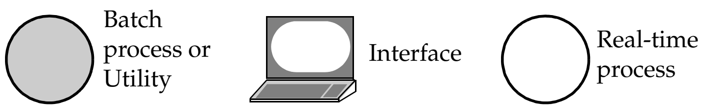
Summary of Modules

Table 2 on page 15 and table 3 on page 17 summarize the modules described in the previous sections and discussed elsewhere in this document. These modules fall into the following general categories:

- *Real-time processes* — run continuously, processing whatever input is available at a given moment
- *Batch processes* — start running at a predetermined time and stop running when no further input is available
- *Utilities* — similar to batch processes but scheduled indirectly through the Utility process (UTL)

  In general, utilities perform optional or peripheral tasks rather than primary tasks.
- *Graphical User Interfaces* (GUIs), also called *interfaces* — Kenan/BP users run these modules as needed to perform specific tasks.

This document uses the following icons to represent these types of modules:



Not all modules are used by every installation of Kenan/BP, and some installations may have custom modules available. See your site-specific documentation for more information.

Table 2 Billing Operations Subsystems

Usage Processing (real-time) *(see chapter 4)*	Collect, guide, and rate usage events. Real-time processes: <varies> Generate usage records MCAP Guide and rate usage records Utilities: RAP Reprocess usage events TIP Postprocess files for external systems MIU Release corrected transactions Interfaces: MPIU Review and correct failed transactions
Usage Processing (batch) *(see chapter 7 for COM and TIP, chapter 4 for others)*	Collect, guide, and rate usage events. Batch processes: COM Exchange usage files MCAP Route usage records to CAP CAP Guide and rate usage records LTP Commit rated usage records to database Utilities: TIP Convert usage file formats RAP Reprocess usage events MIU Release corrected transactions Interfaces: MPIU Review and correct failed transactions
Bill Processing (real-time) *(see chapter 5)*	Generate bills including charges and credits. Real-time processes: BIP Collect and calculate charges BIF Format BIP output onto bills Interfaces: Customer Care Initiate and review bills Invoice Designer Define Invoice Generator bill formats
Bill Processing (batch) *(see chapter 5)*	Generate bills including charges and credits. Batch processes: BIP Collect and calculate charges HDP Calculate historic discounts and rebates BIF or Invoice Generator Format BIP output onto bills BID Dispatch bills to output devices Interfaces: Customer Care Review bills Invoice Designer Define Invoice Generator bill formats

Table 2 Billing Operations Subsystems (Continued)

Payments Processing *(see chapter 7 for COM and TIP; User Guide for Payments and Refunds; chapter 6 for remaining modules)*	Process customer payments. Real-time processes: RTCC Validate/authorize credit card payments Batch processes: COM Exchange payment files LBX Process lockbox bank payments CPM Initiate/process credit card payments EFT Initiate/process EFT payments Utilities: TIP Translate payment files Interfaces: LIU Review failed LBX transactions CCIU Review failed CPM transactions EIU Review failed EFT transactions Payments Manual payment entry Refunds Refunds and deposits
Financials *(see Collections Guide and the User Guide for Collections, and chapter 8 and the Journals Guide for JNL)*	Accounting and collections. Batch processes: JNL Books financial transactions Collections Schedules collections activities Interfaces: Collections Handles collections events
Administration *(see chapter 7)*	Billing operations-related system maintenance. Batch processes: ARCH Archive and restore database tables COM Transfer files UTL Execute UNIX commands Utilities: RepStat Replicate file status across servers TIP Translate files AMP Move accounts among servers LOAD_USG_PTS Populate usage point tables RJU Update product jurisdictions
Operations Center *(see the Operations Center Guide)*	Configuring, launching, and monitoring processes. Real-time processes: MA Process launcher for Operations Center CC Process launcher for Operations Center Interfaces: Operations Center GUI

Modules and Servers

Kenan/BP modules exchange information with database tables in a relational database management system (RDBMS) such as Oracle or Sybase. The set of database tables is referred to throughout this document as the *Kenan/BP database* or simply as “the database.” Because almost every Kenan/BP module task involves exchanging information with the database this document refers to the database only when particularly relevant. This document uses the following icon to represent database information:

Data stored in database

The Kenan/BP database contains three distinct types of information maintained on three different types of *servers* in a multiserver architecture (*MSA*). Every server has a unique server ID in the `SERVER_DEFINITION` and `LOCAL_SERVER_ID` tables.

An MSA deployment has servers of the following types:

- An *Admin* server stores process configurations, business model representations, and similar system-wide data.

  System administrators and product managers maintain this information when configuring the database as described in the *Configurator Guide*. For example, information they enter about available products, promotions, and rate plans controls how the bill processing system produces charges.
- One or more *Customer* servers store account-specific information for some or all Kenan/BP customers.

  Customer service representatives maintain this information using the Customer Care interface, as described in the *User Guide*.
- A *Catalog* server keeps track of which accounts are stored on which Customer server.

  The Customer Care interface maintains this information automatically while updating the Customer server(s).

In an MSA deployment you run some modules against the Admin server and others against the Customer server(s). For example, if your deployment uses three Customer servers you must schedule and run three separate BIP instances — one for each Customer server.

Not all deployments use MSA. In a *single-server* deployment the Customer and Admin servers are combined into a single server. (You can still refer to functions performed on the Customer, Admin, and Catalog servers, however, for consistency's sake.)

See the *Technical Reference* for more detailed information about these servers.

Table 3 lists billing operations modules and the servers against which they run.

Table 3 Billing Operations Processes

Module	Long Name	Type	Subsystem	Server(s)
AMP	Account Mover	Utility	Administration	special
ARCH	Archiver	Batch	Administration	Customer
BID	Bill Dispatcher	Batch	Bill Processing	Customer
BIF	Bill Formatter	Real-time/ Batch	Bill Processing	Customer
BIP	Bill Preparer	Real-time/ Batch	Bill Processing	Customer
CAP	Usage Guider and Rater	Batch	Usage Processing	Customer
Collections	Collections	Batch	Financials	Customer
COM	Communications	Batch	Several	Admin
CPM	Credit Card Payments	Batch	Payments	Customer
EFT	EFT Payments	Batch	Payments	Customer
HDP	Historic Discount Processor	Batch	Bill Processing	Customer
IGEN	Invoice Generator	Batch	Bill Processing	Customer
JNL	Journals	Batch	Financials	Customer

Table 3 Billing Operations Processes (Continued)

Module	Long Name	Type	Subsystem	Server(s)
LBX	Lockbox Payments	Batch	Payments	Admin
LOAD_ USG_PTS	Usage Point Loader	Utility	Administration	Admin
LTP	Long Term Persistence	Real-time/ Batch	Usage Processing	Customer
MCAP	Usage Router	Real-time/ Batch	Usage Processing	Admin
RAP	Usage Reprocessor	Utility	Usage Processing	Customer
RepStat	Manual Replication	Utility	Configuration	special
RJU	Refresh Jurisdiction Utility	Utility	Administration	Admin
RTCC	Real-Time Credit Card Server	Real-time	Payments	special
TIP	File Translator	Utility	Several	Admin
UTL	Utilities	Batch	Administration	Admin, Customer

Scheduling Batch Processes

Before a Kenan/BP process can perform a billing operations task you must configure it to perform that task at a particular time. Therefore, before Kenan/BP can process usage, bills, payments, and so forth you must define a *process schedule* for the relevant modules. In other words you must decide what tasks each process should perform and when it should start performing those tasks and then configure that task scheduling information through Operations Center.

A Simple Example

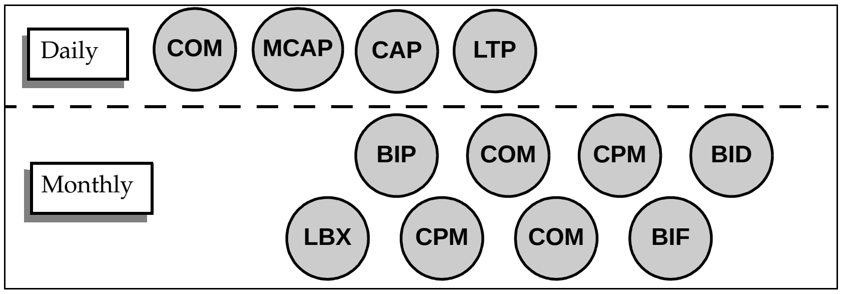
Consider the following five tasks in this order:

1. Batch-process usage once/day (COM → MCAP → CAP → LTP)
2. Process lockbox payments once/month (LBX)
3. Calculate charges and credits once/month (BIP)
4. Process credit card payments once/month (CPM → COM → clearinghouse → COM → CPM)
5. Format and dispatch bills once/month (BIF → BID)

This example is simplified for illustration. See “Sample Process Schedule” on page 24 for a more comprehensive example.

Figure 6 illustrates this process schedule with module position indicating in general terms the order of module, left to right.

Figure 6 Simple Process Schedule Schematic



You might implement this process schedule as shown in table 4.

Table 4 Simple Process Schedule

Interval	Start time	Module	Task
once	5 A.M. Jan 25	LTP	Commit rated usage events to database
daily	6 A.M. Jan 25	COM	Receive usage files.
daily	8 A.M. Jan 25	MCAP	Route files to CAP.
daily	6 P.M. Jan 25	CAP	Rate usage events.
monthly	11:30 P.M. Jan 31	COM	Receive files from credit card clearing house.
monthly	9 A.M. Feb 1	LBX	Retrieve payments from local directory.
monthly	10 A.M. Feb 1	LBX	Process payments into database.
monthly	11 A.M. Feb 1	BIP	Calculate/collect charges and credits.
monthly	8:30 P.M. Feb 1	CPM	Create payment request files (Create).
monthly	8:45 P.M. Feb 1	COM	Send payment requests to credit card clearinghouse.
monthly	10:45 P.M. Feb 1	COM	Receive files from credit card clearinghouse.
monthly	11 P.M. Feb 1	CPM	Process credit card files into database (Process).
monthly	2 A.M. Feb 2	BIF	Format bills.
monthly	1 P.M. Feb 2	BID	Dispatch bills.

Note that this is just one possible order and probably does not reflect the actual order of modules used in your deployment. However, the order is not arbitrary. For example:

- COM, MCAP, and CAP always run in that order because each module depends on the output of the earlier one.
- LTP processes CAP output, but is a continually running server process and does not need to be scheduled once started.
- LBX usually runs before BIP to ensure that recent payments through bank lockboxes appear on the customer's bill.
- CPM usually runs after BIP to ensure that credit card payment requests include the most recently calculated customer balance.
- CPM runs before BIF in this example to allow the payment status to appear on the customer's bill.

  However, in some deployments CPM runs after BID to ensure the customer knows the billed amount before the credit card clearinghouse pays it.

Interactions between daily and monthly modules can sometimes be misleading. For example, CAP must run before BIP because CAP must process usage before BIP can include it on a bill. However, in this example BIP runs on Feb 1 at 11 A.M. and the daily CAP process runs at 6 P.M.

— seven hours later! This may be an error, but it is not because CAP also runs at 6 p.m. on Jan 31, 17 hours before BIP.

Note This example is oversimplified for illustrative purposes; see “Run Times and Processing Times” on page 23.

Optimal Process Schedules

The optimal process schedule for your organization depends heavily on details of your business model, network, and customer base. For example:

- If your customers generate a lot of usage you might process usage several times a day; otherwise, you might process usage only once a day.
- If you have a large number of customers you might produce bills for several subsets of your customer base on different days, issuing monthly customer bills while distributing processing loads over the course of a month.
- If your network mediation layer produces usage records in a standard Kenan/BP format you don’t have to run TIP to convert them to another format; otherwise you do.

These are just a few examples; there are many, many more. For this reason there is no one optimal process schedule — you can run Kenan/BP modules in many different sequences depending on your deployment and customer base.

Subsequent sections of this document — for example, “Scheduling CAP” on page 51 — discuss module-specific scheduling issues for optimizing module schedules.

In addition, note the following general optimization issues:

- You can define multiple *instances* of a single process, each of which runs on a separate schedule.

  For example, table 4 on page 19 includes multiple instances of COM, LBX, and CPM, each performing a different task.

  See “Instances and Tasks” on page 21 for more information.
- If one module depends on the output of another be sure that the output is available.

  For example, BIF formats bills based on the output of calculations performed by BIP. If there is no BIP output to format, BIF will run on schedule, but though it will not generate any errors it will also not produce any formatted bills. Therefore, be sure to run BIF after BIP, not before.

  Always allow time in your schedule to review activity logs and control reports for key modules. That way, if a module fails to produce important output (for example, due to a network or server failure) you discover this as quickly as possible.

Note Examples in this section of module output dependencies are simplified for illustrative purposes; see “Run Times and Processing Times” on page 23.

- When dependencies among modules are complicated it is often simpler to "overschedule" those modules.

  For example, consider the following situation:

  » Collections depends on payments processed by EFT.

  » EFT cannot process a payment until after the customer receives the corresponding bill.

  » BIF depends on collection status updated by Collections.

  » The customer cannot receive a bill until BIF runs.

  This set of demands seems paradoxical (BIF runs after Collections, which runs after EFT, which runs after BIF), but you can resolve it by running EFT twice, once before Collections and once after BIF.

  You can avoid this problem by scheduling EFT to run every day. See "Recurring Schedules" on page 21 for more information.

Instances and Tasks

When you schedule a process you define a single instance of the process that runs on that schedule. You can define multiple instances of each process, and each instance runs independently.

For example, table 4 on page 19 includes four separate COM instances, each performing a separate task:

- One receives usage files every day at 6 A.M.
- One receives payment files monthly starting Jan 31 at 11:30 P.M.
- One sends payment requests monthly starting Feb 1 at 8:45 P.M.
- One receives payment files monthly starting Feb 1 at 10:45 P.M.

You can also define multiple instances to perform the same task on separate schedules — for example, if you want COM to receive usage files at 1 A.M., 3 A.M., 1 P.M., and 3 P.M. every day. One way to do this is to define two COM instances, `com01` and `com02`. Both perform the same task at 12-hour intervals, but one starts at 1 A.M. and the other at 3 A.M.

You can schedule multiple instances of a single process to run simultaneously. As a general rule this is not recommended, with one exception — you may wish to do this if there is too much data for a single process instance to handle efficiently. (Of course, instances running against different databases can run simultaneously without difficulty — see "Modules and Servers" on page 16 for more information.)

If you schedule multiple simultaneous instances against a single database be sure to configure them so they do not interfere with each other. For example, use the SQL Query field to run each instance against a different subset of the database as described in "Choosing an SQL Query Value" on page 32. You must experiment with your local configuration to determine the optimal number of instances. See later chapters for module-specific information.

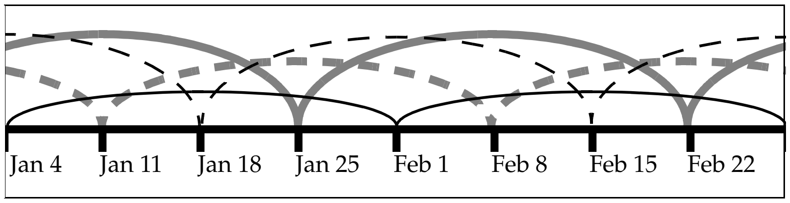
Recurring Schedules

Whenever possible, set tasks to run on a recurring schedule making them easier to maintain and modify.

Generally, usage processing tasks occur one or more times a day and other billing operations tasks occur once every *bill cycle* or *bill period*. (The major exception to this rule is JNL, for which type 2, 3, and 4 runs happen once per journal period; see chapter 8, "Running Journals, Troubleshooting, and Audit Trail," for more information.)

Scheduling modules to run in a bill cycle can be confusing if different customers have different bill cycles, which is common. For example, suppose all customers are billed every 28 days split evenly over four separate bill cycles (for example, suppose 25% are billed on Jan 4, 25% on Jan 11, and so forth, as figure 7 illustrates).

Figure 7 Overlapping Bill Cycles



Suppose you process bills, credit card payments, and journals every bill cycle, as follows:

Table 5 Sample Billing, Payment, Journaling Process Schedule

Day	Time	Module	Task
1	11 A.M.	BIP	Calculate/collect charges and credits.
2	2 A.M.	BIF	Format bills.
2	1 P.M.	BID	Dispatch bills.
9	8:30 P.M.	CPM	Create payment request files (Create).
9	8:45 P.M.	COM	Send payment requests to credit card clearinghouse.
12	6 P.M.	COM	Receive files from credit card clearinghouse.
12	8 P.M.	CPM	Process credit card files into database (Process).
15	6 P.M.	JNL	Journal billed transactions (type 1 run).

This results in the following process schedule for Jan 25 - Feb 8:

Timestamp	Module	Cycle
11:00 Jan 25 1999	BIP	A
18:00 Jan 25 1999	JNL (type 1)	C
02:00 Jan 26 1999	BIF	A
13:00 Jan 26 1999	BID	A
20:30 Jan 26 1999	CPM (Create)	D
20:45 Jan 26 1999	COM (Send)	D
18:00 Jan 29 1999	COM (Receive)	D
20:00 Jan 29 1999	CPM (Process)	D
11:00 Feb 01 1999	BIP	B
18:00 Feb 01 1999	JNL (type 1)	D
02:00 Feb 02 1999	BIF	B
13:00 Feb 02 1999	BID	B
20:30 Feb 02 1999	CPM (Create)	A
20:45 Feb 02 1999	COM (Send)	A
18:00 Feb 05 1999	COM (Receive)	A
20:00 Feb 05 1999	CPM (Process)	A
11:00 Feb 08 1999	BIP	C
18:00 Feb 08 1999	JNL (type 1)	A

So on Jan 25 BIP runs for customers on bill cycle A and, seven hours later, JNL runs for customers on a bill cycle C. On Jan 26, CPM and COM run for customers on bill cycle D after BIF and BID run for cycle A again.

As this example illustrates, overlapping bill cycles can be confusing. At first glance it is natural to assume CPM is processing payments for bills dispatched earlier the same afternoon. In fact it is processing payments for bills dispatched a week earlier. This can be even more complicated if bill cycles aren't constant — for example, if some customers are billed quarterly, others monthly, and others on an irregular schedule.

Notice that this schedule includes a week's delay between dispatching bills and initiating payments, unlike the sample schedule in table 4 on page 19 which initiates payments before formatting bills. Both schedules are valid; which one you use, if either, depends on your organization's policies.

In many cases, you can simplify this kind of overlapping bill cycle schedule by running a module several times a month or even several times a week against the entire database (instead of running against just the customers on a particular bill cycle as in the example above). For example, schedule one BIF instance to format all available bills every Tuesday morning at 2 A.M. — the effect is similar to the example above, but the schedule is easier to maintain because you needn't worry about which BIF is formatting which BIP's bills. You can schedule CPM similarly, configuring it to initiate payments no sooner than seven days after bill dispatching no matter how often it runs.

Running modules against the entire database sometimes incurs performance penalties, depending on your data model configuration. Test both options and compare the results before using either in a production setting.

This document consistently discusses module scheduling in terms of a single bill cycle (as in table 5 on page 22) to emphasize dependencies between modules. Nevertheless, you can configure those modules on a fixed schedule as with BIF above.

Run Times and Processing Times

For illustration, the examples in this section assume that if one module processes the output of another, it processes all of that module's output. In actual practice, that is not always true.

For example:

- If BIP runs after CAP these examples assume that BIP bills all the usage CAP processed.

  In fact, BIP does not bill accounts until a configurable preparation delay has passed from the bill cycle cutoff date and does not include charges or credits on a bill if their effective date is later than the cutoff date.
- If Collections runs after BIP these examples assume that Collections uses the current balance calculated by BIP to determine collections status.

  In fact, Collections waits until the payment due date for the current balance (and a configurable grace period afterwards) before moving accounts into collections based on it.
- If EFT retrieves payments after sending payment requests these examples assume that EFT has retrieved the payment amounts previously requested. In other words, these examples assume that funds transfer is immediate.

  In fact, payments often occur based on payment dates in the request files themselves independent of the EFT run date.

See the module descriptions in later chapters for more detailed scheduling information about each module.

Also, the relationships between BIP, Collections, and payment modules depend heavily on initial module configuration described in the *Module Configuration Guide* Before scheduling these processes be sure you understand how they are configured in your deployment and the order in which they should run.

Sample Process Schedule

Table 6 displays a sample billing operations process schedule.

Note The following is an example only. The listed modules can be executed in another order or eliminated. Module sets vary by deployment.

Table 6 Sample Process Schedule, Summary

Schedule	Time	Module	Task
once	5 A.M.	LTP	Commit rated usage to database (continually running server process).
Daily	5 A.M.	Collections	Schedule collections events (Queue).
	6 A.M	COM	Receive usage files.
	6:30 A.M	TIP	Translate usage files.
	7:30 A.M	COM	Present translated files to MCAP.
	8 A.M	MCAP	Route files to Customer servers.
	11 A.M	CAP	Rate usage events.
Weekly (Monday)	9 P.M.	TIP	Create reseller usage export files.
	11:30 P.M.	COM	Send reseller usage files to external clearinghouse.
Monthly (1st)	1:15 P.M	COM	Receive usage files.
	1:30 P.M	TIP	Translate usage files.
	2 P.M	COM	Present translated files to MCAP.
	2:15 P.M	MCAP	Route files to Customer servers.
	3 P.M	CAP	Process usage into database.
	4:15 P.M	LBX	Retrieve payments from local directory.
	4:30 P.M	LBX	Process payments into database.
	6 P.M	BIP	Calculate/collect charges and credits.
	8 P.M	Collections	Move accounts/invoices into collections (Treat).
	8:30 P.M	CPM	Create payment request files (Create).
	8:45 P.M	COM	Send payment requests to credit card clearinghouse.
	10:45 P.M	COM	Receive acknowledgment files from CC clearinghouse.
	11 P.M	CPM	Process acknowledgment files into database (Process).

Table 6 Sample Process Schedule, Summary (Continued)

Schedule	Time	Module	Task
Monthly (2nd)	1 A.M.	Collections	Move accounts/invoices out of collections (Cure).
	2 A.M.	BIF or Invoice Generator	Format bills.
	1 P.M	BID	Dispatch bills.
	6 P.M	JNL	Book billed transactions (type 1 run).
Monthly (8th)	8:30 P.M	EFT	Create payment request files (Create).
	8:45 P.M	COM	Send payment requests to EFT clearinghouse.
	10:45 P.M	COM	Receive acknowledgments from EFT clearinghouse.
	11 P.M	EFT	Process acknowledgment files into database (Process).
Monthly (15th)	6 P.M	JNL	Book unbilled transactions (type 2 run — this could optionally be scheduled simultaneously with type 3 and type 4 runs).
Monthly (16th)	6 P.M	JNL	Summarize booked transactions (type 3 run this could optionally be scheduled simultaneously with type 2 and type 4 runs).
Monthly (17th)	6 P.M	JNL	Output booked transactions to G/L (type 4 run this could optionally be scheduled simultaneously with type 2 and type 3 runs).

Some things to notice about this schedule:

- COM, TIP, MCAP, and CAP run every morning to process usage and run again in the early afternoon on the first of the month.

  This special usage run is intended to pick up “stragglers” — usage events that take longer than normal to reach your server. For example, if a usage event occurs at 6 P.M. on the thirtieth but is not available to COM until noon on the first the daily 11 A.M. CAP run does not process it, but the special 3 P.M. CAP run does.

  Note that the delays between modules are shorter than during the daily run because less usage is likely to be available.
- LTP processes CAP output, but is started first because it is a continually running process that does not need to be scheduled daily.
- Bill and payment processes run on a monthly cycle starting the first of each month. This implies that all customers share a single bill cycle.
- The daily, weekly, and monthly schedules are designed so that no two processes ever start at the same time. (See table 7 on page 26 for a demonstration.)

  Of course, there is no way to be sure how long a given task takes to complete, so processes may overlap and cause problems. Your process schedule must take into account the estimated running time of each module. For example, this sample schedule assumes that retrieving each day’s usage takes less than 30 minutes and that translating it through TIP takes less than an hour.
- The process schedule does not explicitly include time spent investigating and correcting errors although this is an important part of billing operations.

  In this example, billing operators use LIU before 6 P.M. on the first of each month to ensure that LBX has successfully retrieved and processed all bank payments before BIP runs and use various interfaces to spot-check formatted bills on the morning of the second of each month before BID dispatches them. Similarly, they use MPIU (or MIU) every afternoon after LTP runs, CCIU after CPM runs, and so forth.

Note If the first of a month falls on a weekend or a holiday, consider rescheduling the billing modules for the month to reflect changes in available staff.

- CPM in this example runs shortly after BIP, but EFT waits a week. This means that customers' credit cards are charged for the balance immediately, but electronic funds transfers do not occur until after the customer receives a bill.

  Also, note the two-hour delay between COM's sending payment requests and receiving acknowledgments. This delay reflects estimated time for clearinghouses to process and respond to requests.

Note This example is an oversimplification — funds transfer is not necessarily immediate. See "Run Times and Processing Times" on page 23 for more information.

- BIP, BIF, Collections, and CPM schedules are interrelated and control when customers enter and exit collections. This is important because bills for customers in collections often include collections messages.

  For example, with this schedule a customer who has transferred appropriate funds to a direct debit account can still receive a collections message on a bill because EFT does not check the direct debit account before bills are dispatched.

  Collections (Treat) runs after BIP to move delinquent accounts or invoices into collections as soon as the most recent balance is available. Collections (Cure) then runs before BIF ensuring that paid-up customers move out of collections before the bill is formatted.

  Both LBX and CPM run before Collections (Cure) ensuring that recent checks and this cycle's credit card payments are honored when determining collections status.

Note This example is an oversimplification; see "Run Times and Processing Times" on page 23 for more information.

Table 7 displays this schedule in detail for Jan 25, 2004 through Feb 17, 2004.

Table 7 Sample Process Schedule, Detail

Date	Time	Module	Task
Jan 25	4:55 A.M	LTP	Commit rated usage to database.
	5 A.M.	Collections	Schedule collections events (Queue).
	6 A.M.	COM	Receive usage files.
	6:30 A.M.	TIP	Translate usage files.
	7:30 A.M.	COM	Present translated files to MCAP.
	8 A.M.	MCAP	Route files to Customer servers.
	11 A.M.	CAP	Rate usage events.
	9 P.M.	TIP	Create reseller usage export files.
	11:30 P.M.	COM	Send reseller usage files to external clearinghouse.

Table 7 Sample Process Schedule, Detail (Continued)

Date	Time	Module	Task
Jan 26	5 A.M.	Collections	Schedule collections events (Queue).
	6 A.M.	COM	Receive usage files.
	6:30 A.M.	TIP	Translate usage files.
	7:30 A.M.	COM	Present translated files to MCAP.
	8:00 A.M.	MCAP	Route files to Customer servers.
	2 P.M.	CAP	Process usage into database.
Jan 27 - Jan 30	*(Not included in this example)*		
Jan 31	5 A.M.	Collections	Schedule collections events (Queue).
	6 A.M.	COM	Receive usage files.
	6:30 A.M.	TIP	Translate usage files.
	7:30 A.M.	COM	Present translated files to MCAP.
	8 A.M.	MCAP	Route files to Customer servers.
	11 A.M.	CAP	Rate usage events.
Feb 1	5 A.M.	Collections	Schedule collections events (Queue).
	6 A.M.	COM	Receive usage files.
	6:30 A.M.	TIP	Translate usage files.
	7:30 A.M.	COM	Present translated files to MCAP.
	8 A.M.	MCAP	Route files to Customer servers.
	11 A.M.	CAP	Rate usage events.
Feb 1 (cont.)	1:15 P.M	COM	Receive usage files.
	1:30 P.M	TIP	Translate usage files.
	2 P.M	COM	Present translated files to MCAP.
	2:15 P.M	MCAP	Route files to Customer servers.
	3 P.M	CAP	Rate usage events.
	4:15 P.M	LBX	Retrieve payments from local directory.
	4:30 P.M	LBX	Process payments into database.
	6 P.M	BIP	Calculate/collect charges and credits.
	8 P.M	Collections	Move accounts/invoices into collections (Treat).
	8:30 P.M	CPM	Create payment request files (Create).
	8:45 P.M	COM	Send payment requests to credit card clearinghouse.
	9 P.M	TIP	Create reseller usage export files.
	10:45 P.M	COM	Receive acknowledgment files from CC clearinghouse.
	11 P.M	CPM	Process acknowledgment files into database (Process).
	11:30 P.M.	COM	Send reseller usage files to external clearinghouse.

Table 7 Sample Process Schedule, Detail (Continued)

Date	Time	Module	Task
Feb 2	1 A.M.	Collections	Move accounts/invoices out of collections (Cure).
	2 A.M	BIF or Invoice Generator	Format bills.
	5 A.M	Collections	Schedule collections events (Queue).
	6 A.M	COM	Receive usage files.
	6:30 A.M	TIP	Translate usage files.
	7:30 A.M	COM	Present translated files to MCAP.
	8 A.M	MCAP	Route files to Customer servers.
	11 A.M	CAP	Rate usage events.
	1 P.M.	BID	Dispatch bills.
	6 P.M.	JNL	Book billed transactions (type 1 run).
Feb 3-Feb 7	*(Not included in this example)*		
Feb 8	5 A.M	Collections	Schedule collections events (Queue).
	6 A.M	COM	Receive usage files.
	6:30 A.M	TIP	Translate usage files.
	7:30 A.M	COM	Present translated files to MCAP.
	8 A.M	MCAP	Route files to Customer servers.
	11 A.M	CAP	Rate usage events.
	8:30 P.M.	EFT	Create payment request files (Create).
	8:45 P.M.	COM	Send payment requests to EFT clearinghouse.
	9 P.M.	TIP	Create reseller usage export files.
	10:45 P.M.	COM	Receive acknowledgments from EFT clearinghouse.
	11 P.M.	EFT	Process acknowledgment files into database (Process).
	11:30 P.M.	COM	Send reseller usage files to external clearinghouse.
Feb 9-Feb 14	*(Not included in this example)*		
Feb 15	*(Other modules on this day not included in this example)*		
	6 P.M.	JNL	Book unbilled transactions (type 2 run — this could optionally be scheduled simultaneously with type 3 and type 4 runs).
Feb 16	*(Other modules on this day not included in this example)*		
	6 P.M.	JNL	Summarize booked transactions (type 3 run — this could optionally be scheduled simultaneously with type 2 and type 4 runs) .
Feb 17	*(Other modules on this day not included in this example)*		
	6 P.M.	JNL	Output booked transactions to G/L (type 4 run — this could optionally be scheduled simultaneously with type 2 and type 3 runs).

3 Operating Kenan/BP

This chapter describes how to define, schedule, and launch tasks for Kenan/BP processes using Kenan/BP's Operations Center. It also describes how to review control reports and activity logs to determine the status of each Kenan/BP process.

See chapter 2, "Billing Operations Overview," for a summary of Kenan/BP billing operations processes and functions, and the issues associated with scheduling them. See later chapters in this document for detailed, process-specific operation information.

Using Operations Center

With Kenan/BP's Operations Center you can configure, schedule, launch, and monitor standard Kenan/BP processes. You can access Operations Center with any application that displays HTML graphically and runs Java applets, such as Internet Explorer.

Detailed information on starting Operations Center is in chapter 4 and 5 of the *Operations Center Guide*. A general overview of Operations Center is in chapter 2 of the *Operations Center Guide*. This chapter concentrates on configuring Kenan/BP processes and on non–Operations Center activities.

Configuring Processes

Each Kenan/BP process performs a certain type of task, as described in chapter 2. However, whenever you schedule a specific instance of a process, you must define the specific task type the process instance to perform. For example, COM instances can receive usage files from your network mediation layer, send payment files to a credit card clearinghouse, and receive records of check payments from bank lockboxes; each step is a separate task.

Schedule process instances and define the specific tasks they perform either through the Operations Center's Schedule Process panel as described below or through scripts run directly from the UNIX command line (see "Configuring Processes without Operations Center" on page 34).

To schedule a process instance through Schedule Process, follow these five steps, from the *Operations Center Guide*:

1. On the main Navigation Bar, click **Processes**.
   Result: The Processes page appears.
2. In the Filter window click **Schedule Process.**
   Result: The Schedule Process panel appears.
3. Select the **Module Group**:
   - » Message
   - » Invoice
   - » Payment Processing
   - » Miscellaneous Processing
   - » Various Utilities
4. Enter/select/click the appropriate information for each **Module Group** as described in table 8, "Schedule Batch Processes."
5. Click Submit.

.

Table 8 Schedule Batch Processes

Module Group	**Enter/Select/Click**
Message Processing, Invoice Processing, and Payment Processing	Select the **Module Name, Node, Task Mode** and **Database** Enter **Comments** and **SQL query** if desired. Enter the **Process Name**, and **Task Name**. Select the **Schedule Type**. Click **Submit**.
Various Utilities —	**Note:** UNIX processes require only the Process Type and Filename. Select the **Module Name** and **Node**. Select the **Process Type** and **Database**. Enter the **Filename, Process Name, Task Name, Task Mode, Task Priority**. Enter information in these optional fields as necessary: **Name, Comment, SQL query, Parameters** Enter **Advanced Feature** file information if required. Enter variable=value in the **Environment** field if required. Select the **Schedule Type**. Click **Submit**.
Miscellaneous Processing — **Generic**	Select **Module Name** (Generic) and **Node**. Select the **Process Type** (Kenan BP, Kenan OM, UNIX) and **Database**. Enter **Filename, Process Name, Task Name, Task Mode, Task Priority**. Enter information in these optional fields as necessary: **Name, Comment, SQL query, Parameters** Enter **Advanced Feature** file information if required. Enter variable=value in the **Environment** field. Select the **Schedule Type**. Click **Submit**.
Miscellaneous Processing — **Arch/UTL**	Select the **Module Name** (**Arch** or **UTL**) and **Node**. Enter the **Process Name** and **Task Name**. Select the **Database**, and **Task Mode**. Enter information in these optional fields as necessary: **SQL/UNIX Command, Comment**. Select the **Schedule Type**. Click **Submit**.

For more information, and for descriptions of the difference between batch and continuous processes, see chapter 12 of the *Operations Center Guide*, "Managing Continuous and Batch Processes."

Notes:

- Module Name is CAP, BIP, and so on.
- A node is a hostname where Operations Center runs processes.
- Process Name is generally the process plus two numbers (cap01, bip03). Process names should be unique.
- Task Name. Some Kenan/BP modules accept any task name of eight or fewer characters. Others require specific task names, as described in table 9, "Task Name Constraints."
- Database is the database the process runs against.

- Task Mode is optional, and only used for some modules, such as BIP (for example, backout or pro forma task modes).
- SQL/UNIX Command is an SQL query. These are detailed in “Choosing an SQL Query Value” on page 32.
- Schedule Type is either batch or continuous.

Table 9 Task Name Constraints

Process	Task Name	Comments
Collections	TREAT, QUEUE, CURE, SHUFFLE, PROMISEQ	TREAT moves accounts/invoices into collections, CURE moves them out, QUEUE schedules collections events, SHUFFLE moves accounts/invoices to other collections scenarios, PROMISEQ updates the status of a promise to pay
ARCH	Bill Invoice Detail Bill Image	Bill Invoice Detail archives old bill records; Bill Image archives old bill images.
BID	{Print, Tape}{Hold, No-Hold, Error}[Re-Run]	These are concatenated — for example, Print No-Hold. Print and Tape send to either a printer or a tape drive; Hold prints held bills, No-Hold unheld bills, and Error errored bills; Re-Run reprints successfully dispatched bills. Re-Run is available only for Print No-Hold and Print Error.
CPM	CREATE, PROCESS, or POST	CREATE creates payment request files; PROCESS processes response files from clearinghouses. POST is like CREATE but does not expect a response file.
EFT	CREATE, PROCESS, or POST	
LBX	RCV, GET, or PRO	RCV and GET retrieve lockbox files (use only one of these); PRO processes them.

Choosing an SQL Query Value

For many processes, the `SQL Query` field controls which database records the process selects when performing a task. More specifically, `SQL Query` is the criterion for an SQL `where` clause used against the table(s) the process selects from (listed in table 10 on page 32). (Do not include the word “where” in the `SQL Query`.)

For example, configuring MCAP with the `SQL Query` value **`file_name like "Gate%"`** causes MCAP to add the constraint `where file_name like “Gate%”` to the normal SQL command selecting `FILE_STATUS` rows to process. In other words, MCAP processes only a subset of the files — those whose names begin with `Gate`.

See your SQL documentation for more information about `where` clause syntax. See the *Database Reference* for more information about database fields for the tables listed in table 10.

Table 10 Tables Used for SQL Query

Process	Mode(s)	Table(s) Selected
ARCH	Count, Archive, Delete, and Invoice Restore	Master table for the archive type, such as `BILL_INVOICE` for type 1 or `BILL_IMAGE` for type 2.
	Restore Archive	Not applicable; selects specified archive.
BID	All	`BILL_BATCHES`

Table 10 Tables Used for SQL Query (Continued)

Process	Mode(s)	Table(s) Selected
BIF	Production	`BILL_INVOICE`
	Flash	Not applicable; selects specified accounts.
Invoice Generator	All	`BILL_INVOICE`
BIP	Production	`CMF`, `CMF_INTERIM_BILLS`, `BILL_CYCLE`
	Pro forma	`CMF_INTERIM_BILLS`
	Disconnect-only	`PRODUCT`
	Flash	Not applicable; selects specified accounts.
	HQ and Pro forma HQ	`CONTRACT_ASSIGNMENTS_HQ` (CAH)
	Backout	`BILL_INVOICE` (BI), `CMF_BALANCE` (CB)
CAP	Production	`FILE_STATUS`, `EXT_CONTACTS` and `EXT_SOURCE_ID_REF`
	Consolidate	Not applicable; selects all available files.
COM	All	`EXT_CONTACTS` and `EXT_CONTACTS_STATUS`
MCAP	Production	`FILE_STATUS`
	Flash	Not applicable; processes usage off XIPC queue.
UTL	All	No tables; use a UNIX command line entry as the SQL Query value.

Note As table 10 indicates, the UTL module uses the SQL Query field to specify a UNIX command to execute. See "Running Other Processes (UTL)" on page 126 for more information.

Some additional examples:

- (COM) **`is_send = "1"`**

  Selects contacts with `is_send` = 1 in `EXT_CONTACTS`, indicating an output (send) contact.
- (COM) **`retry_count > 5`**

  Selects contacts with a `retry_count` value of 6 or higher in `EXT_CONTACTS_STATUS`, indicating a contact that has errored out during access more than five times in a row.
- (CAP) **`file_name like "Gate%"`**

  Selects files whose `FILE_STATUS.file_name` values begin with the letters "`Gate`."
- (BID) **`batch_id like "01211130040000"`**

  Selects files with the given `BILL_BATCHES.batch_id` value.

In each example, note that the SQL Query value specifies a field (`is_send`, `retry_count`, and so forth) but not a table name, because the specified field exists in only one selected table — for example, `is_send` exists in `EXT_CONTACTS` but not `EXT_CONTACTS_STATUS` — which Kenan/BP uses automatically.

If a field exists in multiple selected tables, you must identify the specific table. For example, **`ext_contact_id = "1"`** fails for COM, because `ext_contact_id` exists in both `EXT_CONTACTS` and `EXT_CONTACTS_STATUS`. Specify the table name as follows:

`EXT_CONTACTS.ext_contact_id = "1"`.

Note Some processes take abbreviated forms of table names, as table 10 indicates.

Configuring Processes without Operations Center

In addition to scheduling process instances through the Operations Center, you can schedule processes by running a script from the UNIX command line to insert a line into the `PROCESS_SCHED` table.

Warning! Be certain you mean to do this. An incorrectly written script can damage important files. If you are uncertain of any part of this procedure, consider scheduling through the Operations Center interface.

The script differs slightly for Oracle and Sybase systems but must contain values for most of the `PROCESS_SCHED` fields. Fields that require specific values are further explained below; others should be 0 or NULL.

Open a text file and write a script as follows:

Oracle:

```
#!/bin/sh
setenv OAM_ENV_CONN_MA FALSE
DB=<oracle sid>
USER=<user>
PASS=<password>
sqlplus $USER/#PASS@$DB<<THEEND
delete PROCESS_SCHED where process_name='<process name>'
insert into PROCESS_SCHED
(process_name, task_name, task_cycle, task_mode, sched_start, task_
   intrvl, task_status,  task_priority, slide_time, db_name, sql_
   query, debug_level, plat_id, usg_crt_hour,  usg_plat_id, usg_
   version)
values
('<process name>', '<task name>', '<task cycle>', <task mode>, <date
   / time>, <task  interval>, 0, <task priority>, 0, <database name>,
   <sql query>, 0, NULL, NULL, NULL, NULL)
exit
THEEND
```

Sybase:

```
#!/bin/sh
setenv OAM_ENV_CONN_MA FALSE
account=<user>
PASS='cat $ARBORDIR/.arborpw'
isql -U$ARBORSU -P$PASS<<THEEND
```

```
use $DS_DATABASE
go
delete PROCESS_SCHED where process_name='<process name>'
go
insert PROCESS_SCHED
(process_name, task_name, task_cycle, task_mode, sched_start, task_
   intrvl, task_status,  task_priority, slide_time, db_name, sql_
   query, debug_level, plat_id, usg_crt_hour,  usg_plat_id, usg_
   version)
values
('<process name>', '<task name>', '<task cycle>', <task mode>, <date
   / time>, <task  interval>, 0, <task priority>, 0, <database name>,
   <sql query>, 0, NULL, NULL, NULL, NULL)
go
quit
THEEND
```

where

- *`process name`* — The process instance identifier.

  Typically this is the process name in lowercase type followed by a two-digit number from 01 to 10 (for example, `cap01` or `bip03`). This is NOT the process name (that is, not CAP or BIP). Because it is unique on a given server the script must delete any existing tasks with this process name before scheduling a new one.
- *`task name`* — Some processes accept any task name up to 8 letters long, such as the process ID. Other processes require a special valid task name. See table 9, "Task Name Constraints," for a list of task name constraints.
- *`task cycle`* — One of two values, 'R' for recurring or 'N' for non-recurring.
- *`task mode`* — Mode of operation:
  - » 0 = production
  - » 1 = test
  - » 2 = flash
  - » 3 = pro-forma (BIP)
  - » 4 = redo
  - » 5 = disconnect credits only (BIP)
  - » 6 = backout (BIP)
  - » 7 = HQ discout (BIP)
- *`date / time`* — System date and time when the process will start. For the process to start immediately, this is `getdate()` in Sybase and `sysdate` in Oracle.
- *`task interval`* — For recurring tasks (task cycle = 'R') the number of seconds between the start of one scheduled task and the next. To run the task every 24 hours, this is 86400.
- *`task priority`* — Execution priority of the task. Tasks with lower task priority numbers are executed first.
- *`database name`* — Name of the database the process instance runs against.
- *`sql query`* — See "Choosing an SQL Query Value" on page 32.

These fields are all documented in the `PROCESS_SCHED` table in the *Database Reference Vol. 1 (Admin Tables).*

After saving the script, run it at the UNIX command line:

```
<process name> <task name> <server identifier>
```

For example, to run a BIP process named `bip02` against the server with `SERVER_DEFINITION.server_id` = 3, after creating the appropriate script type:

```
BIP bip02 3
```

Reviewing Process Activity

A running process generates a *control report* summarizing its activity and an activity *log file* with detailed information about each transaction. Some processes generate additional status files. You can view all of these files through the Logs window in Operations Center. See chapter 14 of the *Operations Center Guide* for more details.

Each process also logs its results — start and end times, items processed, status — in rows of database tables. This simplifies determining the results of a set of processes, for example, in the case of a BIP run where twenty BIP processes ran against a series of invoices across multiple servers. You can view process status and results multiple ways using Operations Center:

- Process status: use the Processes page (see the *Operations Center Guide*, chapter 11)
- Logs: use the Logs page (see the *Operations Center Guide*, chapter 14)
- Alarm configuration: see the *Operations Center Guide*, chapter 19
- Process status definitions and process group status definitions: see the *Operations Center Guide*, chapters 20, 21, and 22

Control reports and log files are ASCII text files, so you can save them, print them, include them in other documents, or otherwise manipulate them as desired. You also can view them using any text file viewer (for example, the UNIX **`more`** command). They typically reside in the `$ARBORLOG` and `$ARBORCTRLRPT` directories, with filenames of the form `Pname`-*YYMMDD-HHMMSS-serverID-Pid*.*type*, where:

- *`Pname`* -- the Kenan/BP process name (bip01, cap05, and so forth)
- *`YYMMDD`* and *`HHMMSS`* -- the process start time. All *`YY`* combinations are interpreted correctly for the century they fall in. Thus the 99 two-digit combination for *`YY`* is 1999 for log files generated in 1999, and 2099 for log files generated in 2099. The 00 two-digit combination for *`YY`* is interpreted as the year 2000.
- *`serverID`* -- the server against which the process ran
- *`PID`* -- the UNIX process ID issued to the process
- *`type`* -- log for log files and rpt for control reports

See the *Technical Reference* for more information about control report and log file locations, and log file contents. See *Reports and File Layouts* for more information about control report contents.

You can check log files for errors when troubleshooting problems with process execution. However, not all errors indicate a problem with the process. For example, processes exit in error if no tasks are scheduled and when a process completes a non-recurring task it posts an error indicating that it is unable to read the next scheduled task. These errors do not necessarily indicate a problem, but only indicate that no new task information is available.

Reviewing File Status

Processes that read or write to files depend on entries in the `FILE_STATUS` table. When troubleshooting these processes, you can check the status of problem files in `FILE_STATUS`.

Some things to be aware of when reviewing `FILE_STATUS`:

- Processes often handle only files with appropriate `FILE_STATUS` entries.

  Even if the file exists in an appropriate directory, some processes do not read it unless a `FILE_STATUS` entry exists with the appropriate file type. See the *Technical Reference* for more information about how processes generate lists of available files.
- Processes generally create/update `FILE_STATUS` entries as they handle files.

  New files have `file_status` = 1.

  When COM brings a file into Kenan/BP, it populates `FILE_STATUS` with information configured for the external contact.
- When a process starts to handle a file, it locks the `FILE_STATUS` entry (sets `process_name` to its own ID).

Warning! If a process fails while handling a file (for example, if the server crashes), the file may remain locked. DO NOT unlock the file! Instead, rerun the process instance that locked it. That process handles the file properly.

- Every file has a unique identifier (the combination of `file_id` and `file_id_serv`) with which you can find it.

  If you do not know the file ID, select `FILE_STATUS` entries by file name (`file_name`), external contact (`ext_contact_id`), arrival date (`entry_create_dt`), or any combination. These are not guaranteed to be unique.
- The file source type (which controls file processing) is not stored directly in `FILE_STATUS`. You can find the source type in `EXT_SOURCE_ID_REF.source_type` for the `EXT_CONTACTS.source_id` for the `FILE_STATUS.ext_contact_id` for that file.
- `FILE_STATUS` associates each file with a server (`file_id_serv`), even if the file is in a directory (like `$ARBORDATA`) NFS-mounted to several servers.

Note Sometimes you can resolve a process problem by editing `FILE_STATUS` (for example, inserting a required row) and running the process again. However, this is not recommended except in time-critical situations. It is not a long-term solution because the original problem is likely to recur.

See “Replicating File Status (RepStat)” on page 126 for more information about the `FILE_STATUS` table.

4 Usage Processing

This chapter describes the Message Processing System, a Kenan/BP subsystem responsible for extracting usage event information from external usage files, calculating charges for each usage event, and associating those events and charges with the appropriate customers.

For information on Event Time Unit Credits (ETUC), see the *System Administrator Guide*.

MPS Overview

The Message Processing System (MPS) calculates usage charges for Kenan/BP customers (and sometimes for foreign accounts). More specifically, MPS processes perform the following tasks:

- process records of phone calls, pay-per-view movies, Internet resource usage, and similar usage events

  A typical Kenan/BP deployment handles a huge volume of usage charges every bill cycle, often coming from several usage sources on different schedules.
- guide each usage event to an appropriate account

  For example, wireline telephone calls from A to B belong to A’s account, ensuring that A is properly billed for them. Similarly, MPS guides collect calls from B to A to A’s account.

  MPS guides usage events automatically based on the usage type and service IDs on the event record.
- calculate a usage charge for each event

  For example, a long-distance phone call on Saturday might cost 10 cents/minute; a document download from a premium Web site might cost $10/KB. MPS calculates these charges automatically based on predefined usage rates and key fields on the event record.
- (optional) calculate unit credits for an event

  CAP can calculate some unit credits while rating usage and pass them on to BIP. See the *System Administrator Guide* for more information.
- (optional) calculate taxes during rating

  CAP can calculate some taxes while rating usage, and passes these on to BIP.
- (optional) export rated usage events to foreign carriers and other external systems

  If you receive *reseller usage* initiated by your customers’ customers MPS exports those records to the appropriate resellers.

Because many customer-care activities depend on the availability of guided, rated usage events it is important to process usage as rapidly as possible. For example, if a customer inquires about a recent usage event CSRs must be able to review that usage.

MPS supports real-time usage processing (*Flash* mode), guiding and rating each usage record as it arrives. Although MPS can perform batch processing (*Production* mode), real-time processing is recommended. MPS can process usage in real-time using an XIPC queue or over a TCP/IP socket connection. See “Processing Real-Time Usage (MCAP)” below and “Scheduling MPS in Production Mode” on page 43 for more information.

Processing Real-Time Usage (MCAP)

Note This section discusses MCAP in Flash mode. See “Routing Usage (MCAP)” on page 45 for Production mode.

MCAP in Flash mode processes usage events continually as they arrive from a real-time usage generator (called real-time usage processing or *hot billing*). MCAP routes each usage event to the appropriate Customer server for the corresponding account (see “Modules and Servers” on page 16) and then guides and rates the event as described for CAP in "Guiding and Rating Usage (CAP)" on page 49.

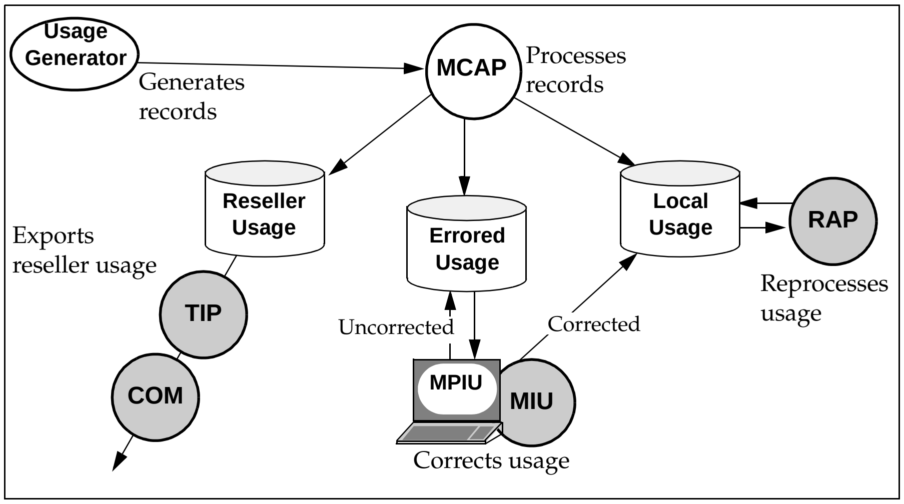
MCAP has two Flash modes, one using XIPC and the other using a TCP/IP socket connection. To use MPS in either, your deployment must have a real-time usage record generator. If this is not the case do not use MPS in Flash mode. Use Production mode instead (see "Scheduling MPS in Production Mode" on page 43).

In addition, run the following MPS modules as needed:

- Message Investigation Units allow you to review, correct and reprocess usage records that CAP cannot process successfully. See "Correcting Errors (MPIU and MIU)" on page 56.
- The Usage Reprocessor (RAP) reflects configuration changes. For example, if usage rates change after CAP has rated a usage event, run RAP to calculate a new charge. See "Reprocessing Usage (RAP)" on page 61.
- The File Translator (TIP) process extracts reseller usage from the database and creates formatted files for export. See "Translating Files (TIP)" on page 122.
- The Communications (COM) process transfers those files to foreign carriers and other external systems. See "Transferring Files (COM)" on page 118.

Figure 8 illustrates how these modules work together to process usage.

Figure 8 Usage Processing (Real-Time)



MCAP in TCP/IP Socket Flash Mode

Note Only MCAP can be run in TCP/IP socket Flash mode. XIPC is still necessary to run BIP in Flash mode.

In Flash mode using TCP/IP sockets MPS operates as follows:

1. The usage record generator connects to the Usage Router (MCAP).
2. MCAP accepts the connection and performs a "handshake," checking the sequence number of the last message it received from that client in `MCAP_FLASH_SEQUENCE_NUMBERS` to avoid processing duplicate messages.
3. The usage record generator sends a usage record to MCAP.

4. MCAP guides and rates the usage record. MCAP stores free usage, rated usage, foreign usage, errors, and other types of usage in different database tables.

   See “Processing Real-Time Usage (MCAP)” on page 40 for more information.
5. MCAP repeats the handshake and receives a new usage record.

When launched, MCAP runs in the background and processes usage without additional scheduling.

Before you run MCAP in TCP/IP Socket Flash Mode

Before running MCAP be sure that a real-time usage generator exists and that the port number is set correctly in Kenan/BP and the usage generator. See the *System Administrator Guide* and *Technical Reference* for more information on setting the port number.

Also, be sure all products, usage rates, usage types, roaming clearinghouses, and other items related to usage processing are configured correctly in the database as described for CAP in “Before You Run CAP” on page 50.

Defining MCAP Tasks in TCP/IP Socket Flash Mode

In MSA deployments, run MCAP against the Admin server (see “Modules and Servers” on page 16). Select RTMP (real-time message processing) mode and schedule MCAP to run continuously.

Once launched, MCAP runs indefinitely, processing usage events as they arrive. However, if MCAP shuts down you can restart it.

For general task definition instructions see “Configuring Processes” on page 30.

Reviewing MCAP Output

You can review MCAP control reports and activity logs as described in “Reviewing Process Activity” on page 36.

MCAP in XIPC Flash Mode

In Flash mode using XIPC MPS operates as follows:

1. The usage record generator places records on an XIPC queue.
2. The Usage Router (MCAP) guides and rates records from the XIPC queue. MCAP stores free usage, rated usage, foreign usage, errors, and other types of usage in different database tables.

   See “Processing Real-Time Usage (MCAP)” on page 40 for more information.

Once launched, MCAP runs in the background and processes usage without additional scheduling.

Before You Run MCAP in XIPC Flash Mode

Before you run MCAP be sure that XIPC is installed on the real-time server and all participating client machines and that a real-time usage record generator exists that can place usage records on the XIPC queue MCAP reads. See “Processing Real-Time Usage (MCAP)” on page 40 for more information.

Also, be sure all products, usage rates, usage types, roaming clearinghouses, and other items related to usage processing are configured correctly in the database as described for CAP in “Before You Run CAP” on page 50.

Defining MCAP Tasks in XIPC Flash Mode

In MSA deployments, run MCAP against the Admin server (see “Modules and Servers” on page 16). Select Flash mode and schedule MCAP to run continuously.

Once launched, MCAP runs indefinitely, processing usage events as they arrive. However, if MCAP shuts down, you can restart it as long as XIPC is still running.

For general task definition instructions see “Configuring Processes” on page 30.

Reviewing MCAP Output

You can review MCAP control reports and activity logs as described in “Reviewing Process Activity” on page 36. Also see the *System Administrator Guide* for more information about reviewing and troubleshooting XIPC.

Scheduling MPS in Production Mode

The network mediation layer for your deployment must monitor usage and create usage files with a record for each event. For example, a telephony switch generates files with one record per phone call.

To process these usage files run the following six MPS modules in the following order:

1. The Communications (COM) process retrieves usage files from the network. See “Transferring Files (COM)” on page 118.
2. (Optional) The File Translator (TIP) process converts files into standard formats. See “Translating Files (TIP)” on page 122.

   This step is unnecessary if the file already is in a standard format (see *Reports and File Layouts* for standard formats).
3. (Optional) COM retrieves the translated files.

   COM marks these files available for processing but does not move them because they are already on the proper server.
4. The Usage Router (MCAP) process routes usage records to the appropriate Customer server. See “Routing Usage (MCAP)” on page 45.
5. The Usage Guider/Rater (CAP) process guides and rates usage events and, in some cases, calculates unit credits and taxes. CAP writes rated usage event details to an output file. See “Guiding and Rating Usage (CAP)” on page 49.

6. The Long Term Persistence (LTP) process inserts records from the CAP output file to the Kenan/BP database. LTP stores free usage, rated usage, errors, and other types of usage in different database tables. See “Persisting Usage (LTP)” on page 55.
7. The Manual Replication process (RepStat) replicates changes to the `FILE_STATUS` table across Customer servers, making the usage events available to other processes. See “Replicating File Status (RepStat)” on page 126.

In addition, run the following MPS modules as needed:

- Message Investigation Units allow you to review, correct and reprocess usage records that CAP cannot process successfully. See “Correcting Errors (MPIU and MIU)” on page 56.
- The Event Processing Replicator (EPR) and Rating Platform Admin Component (RPAC) inteface between the Kenan/BP and Datablitz databases to copy information on event-time unit credits calculated by CAP. Run these modules only if your installation uses event time unit credits. See the *System Administration Guide* for more details.
- The Usage Reprocessor (RAP) reflects configuration changes. For example, if usage rates change after CAP has rated a usage event, run RAP to calculate a new charge. See “Reprocessing Usage (RAP)” on page 61.
- The RPU (Reprocessing Unit) produces a new file of CDRs for CAP to re-rate if event time unit credits are active. See the *System Administration Guide* for more details.
- The File Translator (TIP) process extracts reseller usage from the database and creates formatted files for export. See “Translating Files (TIP)” on page 122.
- The Communications (COM) process transfers those files to foreign carriers and other external systems. See “Transferring Files (COM)” on page 118.

To summarize, schedule batch MPS modules as follows:

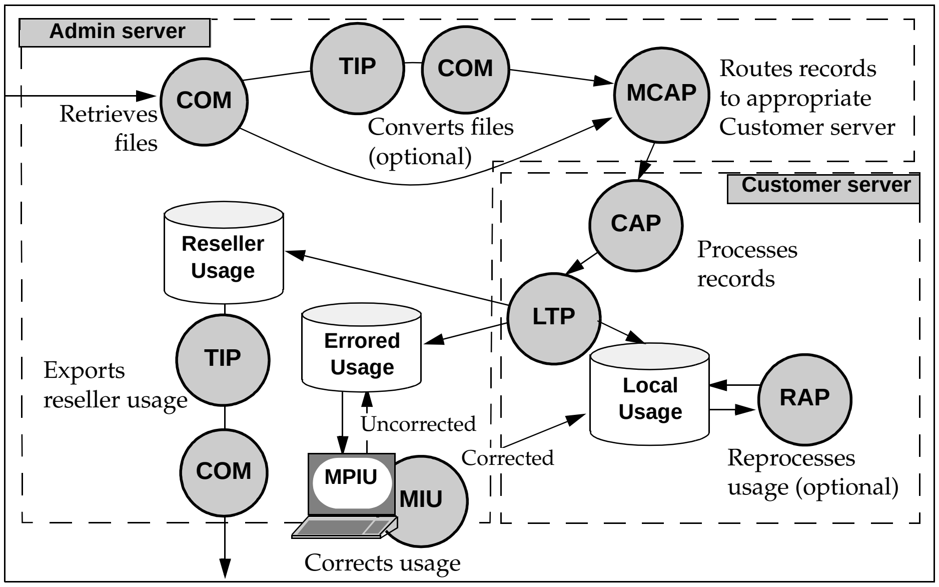
- Schedule the following processes in this order (items in square brackets are optional):

  COM, [TIP, COM], MCAP, CAP, LTP, RepStat, [RAP], [TIP, COM]

  Leave enough time for COM to retrieve files before running MCAP, enough time for MCAP to route records before running CAP, and so forth. How much time these steps take depends on the amount of usage and the usage throughput in your deployment.
- Use the MPIU interface as needed to review and correct errors. Be sure to schedule enough time to correct errors in a usage batch before BIP processes that usage.

Figure 9 illustrates how these modules work together. Figure 9 does not have event time unit crediting active.

Figure 9 Usage Processing (Batch)



Schedule batch MPS modules as often as you want to process usage. If they are configured properly you can run them effectively continuously (for example, every minute). See the *Module Configuration Guide* for more information.

If MPS runs infrequently (a few times a day or less) schedule a final MPS run as close as possible to the start of the BIP run for the current bill cycle to allow for possible lag time for usage events traveling through the network. If the final MPS cycle runs too early it can miss usage events that occurred during the bill period but arrived after COM runs.

Routing Usage (MCAP)

Note This section discusses MCAP in Production mode only. See "Processing Real-Time Usage (MCAP)" on page 40 for Flash mode.

MCAP identifies the account and subscription associated with each usage record by looking up key fields from that usage record in the Kenan/BP database. MCAP routes each record to the appropriate Customer server for the corresponding account, creating one usage file per Customer server. See "Modules and Servers" on page 16 for more information.

You must run MCAP even in a single-server Kenan/BP deployment.

Before You Run MCAP

Be sure all usage types and usage field mappings are properly configured for the usage record format. See the *Module Configuration Guide* for more information on configuring these items.

Also, be sure that all usage files are in the directory in which COM placed them during file transfer.

Defining MCAP Tasks

In MSA deployments run MCAP against the Admin server (see “Modules and Servers” on page 16). Select Production mode.

By default, MCAP processes all new usage files for the Admin server (see “Reviewing File Status” on page 37). You can further restrict the file list by entering an SQL `where` clause for `FILE_STATUS` in the `SQL Query` field for the task. For example:

- Set `SQL Query` to **`file_name like “P1%”`**

  MCAP processes all (and only) new usage files whose filenames begin with `P1`.
- Set `SQL Query` to **`ext_contact_id = “5”`**

  MCAP processes all (and only) new usage files retrieved from the external contact with ID 5.

For general task definition instructions see “Configuring Processes” on page 30.

Scheduling MCAP

Run MCAP immediately after COM to process new usage files. See “Scheduling MPS in Production Mode” on page 43 for more information. For general scheduling instructions see “Configuring Processes” on page 30.

Reviewing MCAP Output

You can review MCAP control reports and activity logs as described in “Reviewing Process Activity” on page 36. MCAP also produces an unresolved files control report showing the names of files not completely processed with the extension `unresolv`.

MCAP also generates a usage file for each Customer server containing usage records for accounts on that server. These files are named *filename*`.`*server*`.`*file* where *filename* is the original filename, *server* is the Customer server ID, and *file* is the file ID from `FILE_STATUS` for the original file (see “Reviewing File Status” on page 37).

For example, suppose an incoming usage file (`usg120197`, file ID 100) contains 1100 usage records as follows:

- 50 invalid records on “server 0”
- 500 valid records for accounts on Customer server 1
- 300 valid records for accounts on Customer server 2
- 250 valid records for accounts on Customer server 5

In this case MCAP produces the following usage files:

`usg120197.1.100` (500 records)
`usg120197.2.100` (300 records)
`usg120197.5.100` (250 records)
`usg120197.0.100` (50 records)

MCAP places these files on the appropriate Customer server in a directory with the same name as the directory the original usage file appears in. This is usually the same directory mounted on multiple servers.

MCAP updates `FILE_STATUS` for each file it processes or creates.

Troubleshooting MCAP

The following sections describe possible reasons and solutions for problems during batch usage routing.

Usage Routed to the Wrong Server

If MCAP routes usage for a local customer account to the wrong Customer server do not change the usage files directly. Instead run CAP and LTP and then use MPIU or MIU to repair the failed transactions. If the misrouting occurs repeatedly repair the cause of the problem.

One possible cause is that the `EXTERNAL_ID_ACCT_MAP` and `EXTERNAL_ID_EQUIP_MAP` tables in the Catalog database are populated incorrectly. MCAP uses these tables to select a Customer server and internal service instance ID based on the *external ID* that appears on the usage record. Each external ID is mapped to one or more internal IDs in these tables which are populated automatically by the customer-care interfaces.

Check these tables to ensure that the correct internal ID (`account_no` or `subscr_no/subscr_no_resets`) and server (`server_id`) appear for the `external_id`s on the incorrectly routed usage. If there are problems with these tables:

- Customer data recently transferredfrom one server to another (see “Moving Accounts between Servers (AMP)” on page 127) may be the cause of the problem. Double-check your AMP configuration and run it again.
- An external ID assigned to the wrong account or service instance may have caused an error during provisioning. Ensure that your organization’s methods and procedures for provisioning IDs are correct and are being followed. See the *User Guide* for more information about customer provisioning.

MCAP Rejects File

If MCAP rejects a file it attempts to process, one of the problems in table 11 may exist:

Table 11 MCAP Rejects File

Problem	Description	Solution
Corrupted or inaccessible file	MCAP cannot read the file	Confirm that you can read the file normally while logged in to the UNIX server as the user `arbor`. (See the *System Administrator Guide* for more information about users and file locations.) Otherwise, confirm that you can read the original file retrieved by COM. If so, the COM task that retrieved the file may be configured incorrectly (see “Troubleshooting COM” on page 121). If not, your usage mediation layer provided a corrupted file and must reproduce it.

Table 11 MCAP Rejects File

Problem	Description	Solution
Excessive number of incorrect records	The `SOFT_LIMIT`, `HARD_LIMIT`, and `ERR_LIMIT` system parameters set the percentages of failed records allowed in a file. If these values are exceeded the entire file is rejected.	Usually a file that exceeds these limits has a fundamental problem such as corruption or incorrect format, and you should correct that problem (see below) and rerun MCAP. If you wish you can override these settings by setting the `OVERRIDE` system parameter to 1. Of course, this does not fix errored records in the file; it simply causes MCAP to route the small number of valid records to the proper Customer servers and store the rest as failed records. In some cases this is not a problem — for example, if a usage file has a large number of outcollects for non-Kenan/BP accounts.
Wrong usage format	If all the records of a particular type fail processing the usage record format may be incorrect.	If you translate usage files using TIP or some other mechanism the problem may lie in the translation process. Confirm your TIP (or other) configuration. Otherwise the record format may be incorrectly configured in `RAW_USAGE_FIELD_MAPPING`. Check the file and the mapping configuration against the file layout specification for your network mediation layer. See the *Module Configuration Guide* for more information.
Record count mismatches in the file	Some usage file formats include summary records, and MCAP may reject files where the number of detail records indicated in a summary is not equal to the number of detail records found in the file.	Check the number of records listed in the control report and check the original file. You can force MCAP to ignore record count mismatches by setting the `OVERRIDE` system parameter to 1.
Missing control file	If a file format requires a control file and there is none available, the problem may be that the Control File Expected flag was not set for the external contact when COM was run. Conversely if this flag is set when none is required MCAP does not process the file.	In either case, reconfigure the external contact as described in the *Module Configuration Guide* (you may need to rerun COM as well); then rerun MCAP.

File Remains Unprocessed with No Error

If MCAP runs but leaves a usage file untouched, confirm that the file exists as a usage file in `FILE_STATUS` (see “Reviewing File Status” on page 37). One of the problems in table 12 may exist:

Table 12 File Unprocessed by MCAP with No Error

Problem	Description	Solution
No entry exists for the file	If no entry exists for the file either the file was not brought into the system by COM, or the file name or location was changed after it came in.	Confirm that COM ran correctly (see "Troubleshooting COM" on page 121) when retrieving the file. If not, rerun COM to retrieve the file; then rerun MCAP. If COM ran correctly the file was probably moved or renamed after retrieval, perhaps during usage preprocessing. If this is a one-time problem rename the file to the file name given in `FILE_STATUS` (if any) or insert a row in `FILE_STATUS` representing the file; then rerun MCAP. If this is a recurring problem, run a COM task to move each file of this type from its current location to its current location (use the current location as the file's destination). The file does not move, but COM makes an entry in `FILE_STATUS`. Then rerun MCAP. Schedule this COM task to run before MCAP on a regular basis. (You may have to configure new external contacts for COM; see the *Module Configuration Guide* for more information.)
File is not a usage file	If the file is not a usage file the COM external contact is probably misconfigured.	Reconfigure and rerun COM; then run MCAP again.
File has failed processing too many times	If the number of failed processing attempts (`num_failed_ attempts`) for this file exceeds the value of the `MAX_RETRIES` system parameter MCAP does not attempt to reprocess the file.	Reset `num_failed_attempts` and rerun MCAP. Prevent further failures by resolving the cause of the original failure.
New usage file	If the file is new and was available when MCAP ran, the SQL query may not have included it.	check that the SQL Query in the MCAP task did not exclude this file. If it did, run a new MCAP task that includes the file.

Guiding and Rating Usage (CAP)

Note Unless otherwise stated this document refers to CAP in Production mode. See "Consolidated Usage Detail Files" on page 52 for more information about Consolidate mode.

For each record representing a valid usage event in each usage file CAP does the following:

- guides the event — that is, determines the customer account or external clearinghouse associated with the event

- rates the event — that is, calculates the usage charge associated with the event
- (optional) calculates unit credits — that is, calculates unit credits applied against the charge associated with the event
- (optional) taxes the event — that is, calculates the tax on the charge associated with the event (non-binned taxes only)
- writes the guided, rated usage event to an output file0

Note Event time unit credits are not covered in this book. See the *System Administrator Guide*, chapter 7, for more information on event time unit credits.

Kenan/BP supports a wide variety of guiding, rating, taxing, and storage options for different usage types. For example, one usage type might be *free* (not rated), another *split* (stored in two or more database records), and a third billed to the target instead of the origin point. See the *Guide to Products, Rates, and Discounts* for more information.

Before You Run CAP

Be sure all products, usage rates, usage types, roaming clearinghouses, real-time forward taxes, and other items related to usage processing are correctly configured in the database. See the *Guide to Products, Rates, and Discounts* for more information.

In particular, be sure the usage location tables are up to date for deployments that use location-based rating. You may have to run LOAD_USG_PTS to update usage locations (see "Loading Usage Points (LOAD_USG_PTS)" on page 130).

Be sure the MPS system parameter `LTP_EXT_CONTACT_ID` is set. This system parameter specifies the `EXT_CONTACTS.ext_contact_id` where CAP writes the output file. This system parameter has no default value; CAP fails if it is not set. Generally, it is equal to `$ARBOR_ DATA/usage/ltp`.

To use rate-time taxation, make sure the system parameter `TAX_POST_RATE_USAGE` is set to 1. Otherwise, set it to 0.

Also, be sure that all usage files are in the directory MCAP placed them in during usage routing.

Defining CAP Tasks

In MSA deployments, run CAP separately against each Customer server (see "Modules and Servers" on page 16). By default, CAP processes all new usage files for that server (see "Reviewing File Status" on page 37) and also processes "server 0" soft error files if any are available (see "Reviewing MCAP Output" on page 46).

You can further restrict the file list by entering an SQL `where` clause for the `FILE_STATUS`, `EXT_ CONTACTS` and `EXT_SOURCE_ID_REF` tables in the `SQL Query` field for the task. For example, enter an `SQL Query` value of **`file_name like "Gate%"`** for CAP to process all (and only) new usage files on the server whose names start with `Gate`.

To use the `change_dt`, `change_who`, `ext_contact_id`, or `is_send` fields in `SQL Query` you also must specify the table name because these fields appear in more than one table. For example, to restrict CAP to files from external contact 17 enter one of the following `SQL Query` values:

```
EXT_CONTACTS.ext_contact_id = "17"
FILE_STATUS.ext_contact_id = "17"
```

For general task definition instructions, see “Configuring Processes” on page 30.

In Consolidate mode, CAP processes all new usage detail files (see “Consolidated Usage Detail Files” on page 52 for more information about these files).

Scheduling CAP

Run CAP in Production mode immediately after MCAP to process new usage files.

Run CAP in Consolidate mode only if your deployment uses usage consolidation. In this case, run CAP (Consolidate) just before running BIP; otherwise usage charges might not appear on the customer bill. You can run CAP (Consolidate) more often; however, the more frequently it runs, the less effective it is in saving database space and improving performance. See “Consolidated Usage Detail Files” on page 52 for more information.

For general scheduling instructions see “Configuring Processes” on page 30.

Real-Time Tax Calculation

CAP can calculate taxes at rate time and save and return tax details. CAP writes a taxation record (type TXD) to its output file. This record type carries real-time taxation details. The KP1 record also carries a field called `num_tax_details` which indicates the number of TXD rows that follow immediately in the file. CAP reads the TXD information and records it in `CDR_DATA_STRUCT` as a linked list.

Data from TXD records is stored in `CDR_DATA_TAX` whether the system parameter TAX_POST_RATE_USAGE (MPS) is set to 0 or 1. When BIP runs, if TAX_POST_RATE_USAGE = 0, BIP aggregates pre-taxed usage separately from usage that has yet to be taxed. If TAX_POST_RATE_USAGE = 1, all usage has been taxed at rating time, whether it was pre-taxed or not.

Real-Time Tax Calculation

Taxes are calculated in real time by the same BILL_LIB functions used by BIP. When TAX_POST_RATE_USAGE = 1, CAP calls the appropriate tax packages and writes tax information to `CDR_DATA_TAX` after rating a usage event in real-time mode. The number of rows written to `CDR_DATA_TAX` for a single usage event is recorded in `CDR_DATA.tax_pkg_count`. When BIP runs, it retrieves tax data from `CDR_DATA_TAX` and writes it to `BILL_INVOICE_DETAIL`.

Binned Taxes Not Eligible for Real-Time Taxation

A binned tax applied at rate time on multiple charges (each record processed individually) produces a different result than the same tax applied by BIP (records processed in bulk). To correct this discrepancy, Kenan/BP 11.5 does not calculate taxes at rate time for CDRs that include a binned tax. All such tax calculation is now deferred to BIP.

Reviewing CAP Output

CAP creates an output file in `LTP_EXT_CONTACT_ID/arbor_ready` . The filename is of the form

<ext_contact_id>.*<file_name>*.*<server_id>*.*<file_id>*

where

- *<ext_contact_id>* is the external contact ID
- *<file_name>* is the base filename, from `FILE_STATUS.file_name`
- *<server_id>* is the server ID
- <file_id> is a file ID generated by CAP

You can review CAP control reports and activity logs as described in "Reviewing Process Activity" on page 36. CAP also produces an error log file showing error codes and descriptions for *soft errors* (readable records it cannot guide and rate) with the standard CAP logfile name except for an `ERR` extension.

CAP updates `FILE_STATUS` for each usage file and stores readable usage records in an output file. CAP stores the output file in the `arbor_ready` directory of the server whose `EXT_CONTACTS.ext_contact_id` is specified in the system parameter `LTP_EXT_CONTACT_ID`.

In some cases CAP stores consolidated records in the database and creates detail files with the original usage records. In others it creates no database records but does create detail files. See "Consolidated Usage Detail Files" below for a description of expected CAP output in these cases.

Consolidated Usage Detail Files

Depending on the account, the usage type, and global MPS configuration CAP does one of the following:

- stores each usage record as a separate database record as described above
- summarizes one or more records in each input usage file into a single database record and writes the original records to a separate usage detail file for each input usage file

CAP consolidation behavior depends on two system parameters: `CONSOLIDATE_USAGE` (MPS) and `DETAIL_FILE_BY_INPUT_FILE` (MPS).

If the system parameter `CONSOLIDATE_USAGE` (MPS) is set to 0, CAP does not consolidate usage.

If the system parameter `CONSOLIDATE_USAGE` (MPS) is set to 1 or 2, CAP consolidates usage while rating CDRs. CAP writes consolidated usage detail records to its export file in `EXT_CONTACTS.ext_contact_id/arbor_ready`, and LTP commits the consolidated records to the database.

If the system parameter CONSOLIDATE_USAGE (MPS) is set to 3, CAP consolidates usage in two steps:

1. First, CAP rates usage and creates consolidation detail files. CAP creates a normal output file for unconsolidated usage, and also creates usage detail files in the directory defined in `EXT_CONTACTS.arbor_detail` for the external contact with the filename *original_filename*`.detail.CAP`. These consolidation detail files summarize one or more

records in each input usage file into a single database record and write the original records to a separate usage detail file for each account

In this case, CAP creates one or more detail files for each account once per bill cycle named *ID*_*dates*`.detail`.*counter* where:

ID is the account internal ID (`CMF.account_no`)

dates is the bill cycle date range, in the form `yyyymmdd_yyyymmdd`

counter is 1 for the first file, 2 for the second, and so forth (CAP starts a new file whenever a file becomes too large)

For example, `1_19980101_19980201.detail.1`. These files appear in the `$ARBORDATA/usage/detail/ready` directory.

a. If the system parameter DETAIL_FILE_BY_INPUT_FILE is set to 0, CAP creates consolidation detail files for each account/bill cycle as above but does not create records in the main output file for consolidated usage.

b. If the system parameter DETAIL_FILE_BY_INPUT_FILE is set to 1, CAP creates detail files for each account/bill cycle as above, summarizes one or more records in each input usage file into a single database record in the main output file, and writes the original records to a separate usage detail file for each input file.

2. Second step of consolidation, you must run CAP in Consolidate mode (task 13) to consolidate the usage and create a file for LTP to insert into the database.

See the *Technical Reference* for a complete description of how CAP consolidates usage.

Troubleshooting CAP

The following sections describe possible reasons and solutions for problems during batch usage routing.

CAP Fails on Startup

If the system parameter `LTP_EXT_CONTACT_ID` (MPS) is not set, CAP fails because it cannot create a file to write output to. Make sure this system parameter is set to a valid `EXT_ CONTACTS.ext_contact_id` value that has a `ready` directory.

CAP Rejects File

If CAP rejects an entire usage file the probable cause is that too many individual records failed processing (see below).

If this is not the problem, troubleshoot the file as described in “Troubleshooting MCAP” on page 47. Because MCAP already has routed the records in this file successfully it is unlikely that the file is corrupted, inaccessible, of the wrong format, or incorrectly stored in `FILE_STATUS`.

If these conditions do occur, the probable cause is another process that altered the file before CAP processed it.

CAP may still reject individual records in a file that MCAP has already routed. This happens because MCAP reads only enough of each record to identify the associated service instance, while CAP reads the entire record.

Few Failed Records

If CAP processing fails on a few records you can correct the immediate problem using MPIU (see "Correcting Errors (MPIU and MIU)" on page 56). Do not rerun CAP; MPIU guides and rates the corrected records.

Caution Never modify `CDR_DATA_WORK` entries manually; use MPIU instead.

If this is a recurring problem, the cause is probably one of the following:

- errors in the records themselves

  Compare the failed records to other records of the same type. If they are inconsistent contact the individuals maintaining your usage mediation layer to identify and resolve the problem.
- errors in customer account information

  Probably caused by errors during provisioning. For example, if a customer's service instance is provisioned with two products that share a usage type CAP has no way of identifying which product is associated with a usage of that type.

  Ensure that your organization's methods and procedures for provisioning are correct and are being followed. See the *User Guide* for more information about customer provisioning.

Numerous Failed Records

If CAP processing fails on a large number of records the probably cause is incorrect database configuration. (This is especially likely if most of the failed records share a single usage type.) In particular, if CAP fails to rate a large number of records the probable cause is missing rows in `RATE_USAGE` or related tables.

To fix this problem confirm and repair CAP configuration as described in the *Module Configuration Guide* and correct the records as described in "Correcting Errors (MPIU and MIU)" on page 56.

Incorrect Charge Amounts

If CAP runs without error but incorrect usage charges appear in the CAP control report or the customer invoice the probable cause is incorrect usage rate configuration. Confirm that `RATE_USAGE` and related tables are configured properly as described in the *Configurator Guide;* then use RAP to reprocess the usage as described in "Reprocessing Usage (RAP)" on page 61.

In particular, if `USAGE_TYPES.use_default_rate_type` is 4 the problem may be that the correct rate is missing from the database and CAP is using a default rate instead.

SQL Buffer Overflow

If CAP fails with the message

```
(E 24140) DATABASE_ERR2 ERROR: Arbor data layer SQL buffer overflow
```

the BATCH_SIZE system parameter may be set too high. Lower the parameter value (20 is a common value) and run CAP again. See the *Module Configuration Guide* for more information about setting system parameters.

Itemized Usage Is Missing

If CAP runs without error but itemized usage does not appear in CDR_DATA, CAP may be configured to consolidate usage and produce usage detail files. If USAGE_TYPES.consolidate_usage for the usage event's type_id_usg is 1, it is the most likely reason. See "Consolidated Usage Detail Files" on page 52 for more information.

If you need itemized usage for later processing (for example, if the customer bill format requires itemized usage) you can either:

- use the detail files as itemized usage records
- reprovision accounts with a different account category to eliminate usage consolidation for those accounts
- reconfigure relevant account categories, usage types, or both to eliminate usage consolidation altogether

  See the *Module Configuration Guide* for more information on configuring CAP to consolidate usage.

Persisting Usage (LTP)

LTP commits rated usage records from CAP output files to the Kenan/BP database. LTP writes rated usage records to the following tables:

- CDR_DATA (Customer server) for guided and rated records for local accounts
- CDR_FREE (Customer server) for free usage for local accounts
- CDR_OUTCOLLECTS (Admin server) for guided and rated records for roaming clearinghouses
- CDR_DATA_TAX (Customer server) for taxes on usage events calculated by CAP
- CDR_DATA_WORK (Admin server) for soft errors

LTP also writes any *hard errors* — that is, records CAP could not read well enough to process or records in files that it rejected — that CAP could parse to CDR_DATA_WORK.

LTP also inserts rows in CDR_UNBILLED pointing to new rows inserted in CDR_DATA.

Before You Run LTP

Be sure all LTP and MPS system parameters are set. In particular, be sure that the MPS system parameter LTP_EXT_CONTACT_ID is set. This is where LTP looks for the file of rated usage records created by CAP.

Be sure the CAP output file is in the ready directory specified by LTP_EXT_CONTACT_ID.

Defining LTP Tasks

LTP is a server application. That is, one LTP process per Customer server needs to run. It runs continuously until it is shut off. By default, LTP processes all new rated usage files for that server as CAP generates them (see “Reviewing File Status” on page 37).

For general scheduling instructions see “Configuring Processes” on page 30.

Reviewing LTP Output

You can review LTP control reports and activity logs as described in “Reviewing Process Activity” on page 36. LTP also produces an error log file showing error codes and descriptions for *soft errors* (readable records it cannot guide and rate) with the standard LTP logfile name except for an `ERR` extension.

Correcting Errors (MPIU and MIU)

The Message Processing Investigation Unit (`miuexe`, generally called MPIU) is a Windows NT graphical interface for correcting soft errors during usage guiding.

Note The MIU process described in “MIU Batch Processing” on page 61 is related to but different from the MPIU interface.

Specifically, MPIU reviews and corrects `CDR_DATA_WORK` entries with incomplete status (see “Reviewing CAP Output” on page 52). To review these entries:

1. Launch MPIU.

   Choose **Message Processing Investigation Unit** from the Windows NT Start menu or double-click the corresponding icon in a program group window.

   If it is unavailable or does not launch see the *Module Configuration Guide* for further instructions.
2. Enter your Kenan/BP username and password.

   Contact your system administrator for an appropriate username and password if you do not have one.

   The `Message Investigation` window (see figure 10) appears.
3. Enter filter settings and click *Filter*.

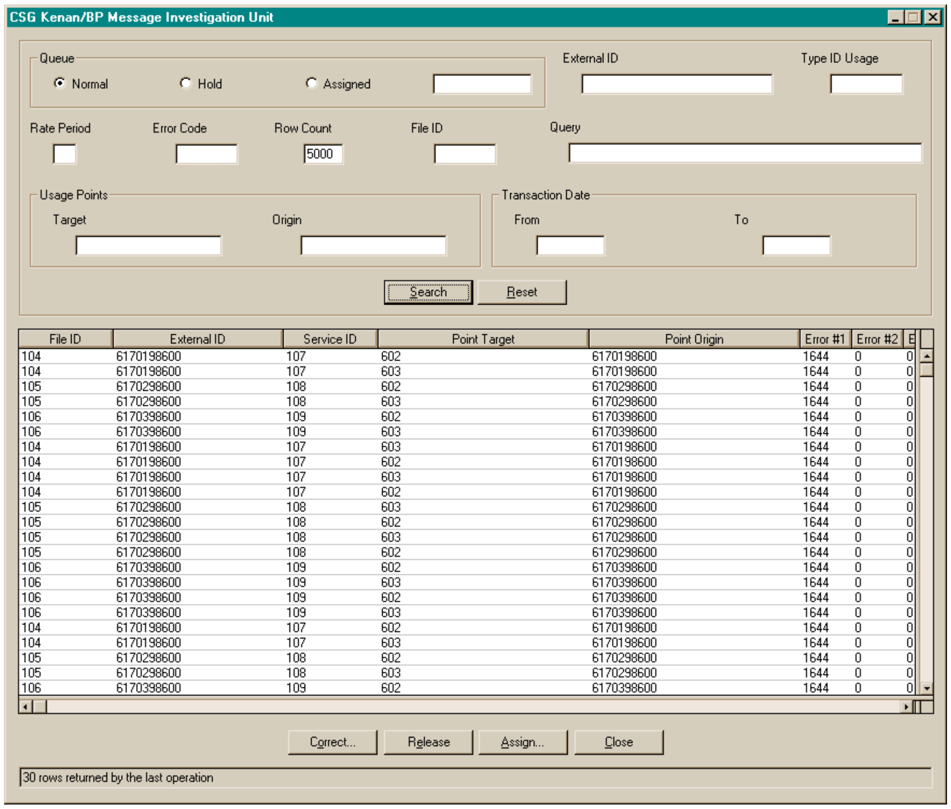
   MPIU lists the entries that match your filter settings up to the maximum number of entries.

   See “Filtering Messages” on page 57 for more information.

Note MPIU does not display any entries until you click *Filter*.

Once entries appear you can review, correct, reprocess, mark them as uncorrectable, or reassign them (see “Process Records” on page 59).

Figure 10 Message Investigation Window



Queue: [✓ Normal] Hold Assigned

Row Count: 5000

File ID	External ID	Service ID	Point Target	Point Origin	Error #1	Error #2	E
104	6170198600	107	602	6170198600	1644	0	0
104	6170198600	107	603	6170198600	1644	0	0
105	6170298600	108	602	6170298600	1644	0	0
105	6170298600	108	603	6170298600	1644	0	0
106	6170398600	109	602	6170398600	1644	0	0
106	6170398600	109	603	6170398600	1644	0	0
104	6170198600	107	603	6170198600	1644	0	0
104	6170198600	107	602	6170198600	1644	0	0
104	6170198600	107	603	6170198600	1644	0	0
104	6170198600	107	602	6170198600	1644	0	0
105	6170298600	108	603	6170298600	1644	0	0
105	6170298600	108	602	6170298600	1644	0	0
105	6170298600	108	603	6170298600	1644	0	0
105	6170298600	108	602	6170298600	1644	0	0
106	6170398600	109	602	6170398600	1644	0	0
106	6170398600	109	603	6170398600	1644	0	0
106	6170398600	109	602	6170398600	1644	0	0
106	6170398600	109	603	6170398600	1644	0	0
104	6170198600	107	602	6170198600	1644	0	0
104	6170198600	107	603	6170198600	1644	0	0
105	6170298600	108	602	6170298600	1644	0	0
105	6170298600	108	603	6170298600	1644	0	0
106	6170398600	109	602	6170398600	1644	0	0

30 rows returned by the last operation

The `Message Investigation` window includes up to three error codes attached to each failed transaction. Descriptions of these errors appear in the `Message Correction` window (see figure 13 on page 59). Error codes also appear in the CAP error report as described in "Reviewing CAP Output" on page 52.

Filtering Messages

As figure 10 shows the `Message Investigation` window includes numerous filter settings such as `Type Id Usg` (usage type) and `External Id` (phone number, e-mail address, or similar identifier). Use these fields to define the subset of failed transactions that appear in the list box or leave these fields blank to list all failed transactions.

For example, use the following fields to filter errors:

- *Queue* radio buttons — Use *Normal*, *Hold*, or *Assigned* to restrict the list to normal messages, messages on hold for special attention, or messages assigned to particular users, respectively.
- `External Id` — Lists failed transactions with a specific external ID.
- `Type Id Usg` — Lists failed transactions of a given usage type. If many failed transactions share the same usage type you may have to reconfigure CAP or TIP to avoid similar problems in the future (see "Troubleshooting CAP" on page 53).
- `File Id` — Lists failed transactions for usage records from the same file. All usage records from a single usage file share a file ID that COM stores in the `FILE_STATUS` table. (See "Reviewing File Status" on page 37 for more information about the file ID.)

If many failed transactions share the same file ID you may have to reconfigure CAP or TIP to avoid similar file-level problems in the future. See "Troubleshooting CAP" on page 53 for more information.

- `Error code` — Filter on this field to simultaneously view several messages with the same error.

  Error codes appear on the CAP error report as described in "Reviewing CAP Output" on page 52.
- `Rate period` — Lists failed transactions from a given rate period.
- `Query` — Use to filter by SQL query.
- `Usage Points` — Lists failed transactions for usage with the same origin or target point.
- `Trans Date` — Lists failed transactions that fall between given start and end dates.

You must click *Filter* before MPIU lists any entries. The status bar indicates the number of listed entries.

By default no more than 5000 entries appear at once. You can enter a new maximum in the `Row Count` field.

Sorting Messages

As figure 10 on page 57 shows, the `Message Investigation` window includes a number of sort settings to help you prioritize messages based on file ID, usage event target/origin, and similar fields.

Use these fields to control how MPIU sorts displayed records as follows:

- Check a field to sort output by that field.
- Check several fields to sort output by more than one sort key.
- Check *Reverse* next to a checked field to sort output by that field in descending order instead of ascending order.

When all fields are set properly click *Sort* to reorder the list.

For example, suppose your list includes 10 entries as follows:

Figure 11 Unsorted Message List

File ID	Point Target	Other fields
23	6171239123	not listed here...
24	6158391287	not listed here...
23	6170294923	not listed here...
25	7029132212	not listed here...
23	3139234962	not listed here...
23	6170938100	not listed here...
25	7023239121	not listed here...
25	7029593234	not listed here...
25	3134123401	not listed here...
23	6173421912	not listed here...

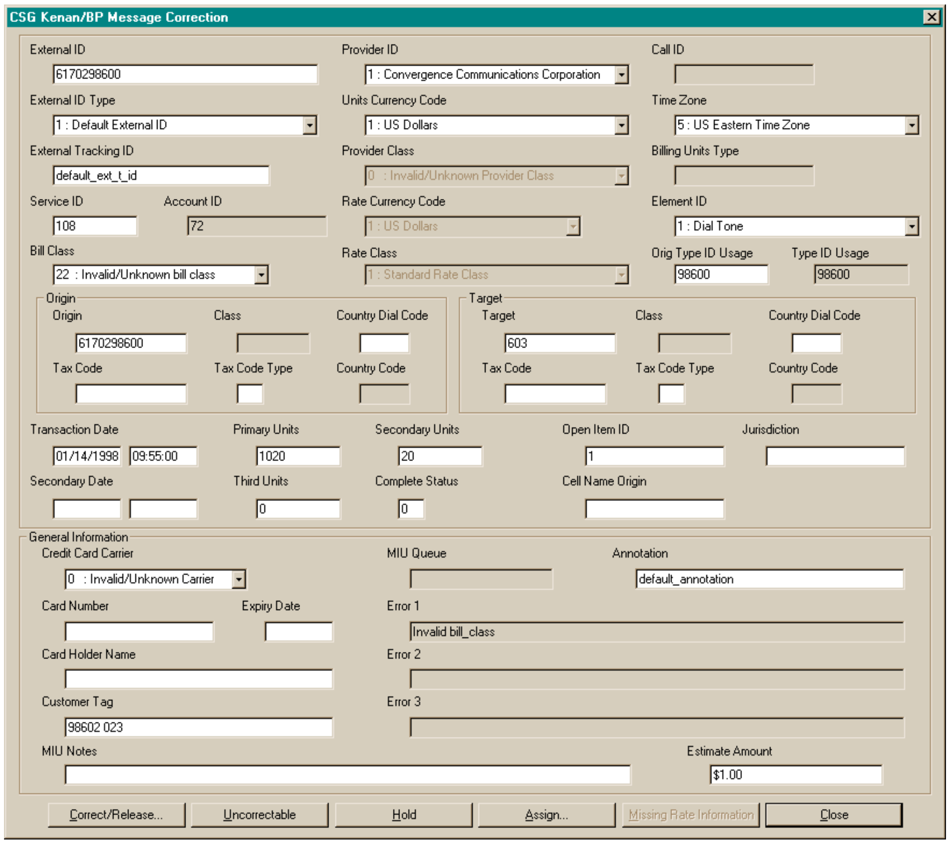
Check *File ID* and *Target* and click *Sort* to reorder them as follows:

Figure 12 Sorted Message List

File ID	Point Target	Other fields
23	3139234962	not listed here...
23	6170294923	not listed here...
23	6170938100	not listed here...
23	6171239123	not listed here...
23	6173421912	not listed here...
24	6158391287	not listed here...
25	3134123401	not listed here...
25	7023239121	not listed here...
25	7029132212	not listed here...
25	7029593234	not listed here...

Process Records

Figure 13 Message Correction Window


CSG Kenan/BP Message Correction

External ID: 6170298600
Provider ID: 1 : Convergence Communications Corporation
Call ID:
External ID Type: 1 : Default External ID
Units Currency Code: 1 : US Dollars
Time Zone: 5 : US Eastern Time Zone
External Tracking ID: default_ext_t_id
Provider Class: 0 : Invalid/Unknown Provider Class
Billing Units Type:
Service ID: 108
Account ID: 72
Rate Currency Code: 1 : US Dollars
Element ID: 1 : Dial Tone
Bill Class: 22 : Invalid/Unknown bill class
Rate Class: 1 : Standard Rate Class
Orig Type ID Usage: 98600
Type ID Usage: 98600

Origin
Origin: 6170298600
Class:
Country Dial Code:
Tax Code:
Tax Code Type:
Country Code:

Target
Target: 603
Class:
Country Dial Code:
Tax Code:
Tax Code Type:
Country Code:

Transaction Date: 01/14/1998 09:55:00
Primary Units: 1020
Secondary Units: 20
Open Item ID: 1
Jurisdiction:
Secondary Date:
Third Units: 0
Complete Status: 0
Cell Name Origin:

General Information
Credit Card Carrier: 0 : Invalid/Unknown Carrier
MIU Queue:
Annotation: default_annotation
Card Number:
Expiry Date:
Error 1: Invalid bill_class
Card Holder Name:
Error 2:
Customer Tag: 98602 023
Error 3:
MIU Notes:
Estimate Amount: $1.00

To correct an individual entry in the `Message Investigation` window list, follow these four steps:

1. Double-click the entry or select it and click *Correct...*

   The `Message Correction` window (figure 13) appears, displaying detailed information for the selected record.

2. Modify editable fields (those with white backgrounds) as necessary.

   Noneditable fields are calculated by Kenan/BP; you cannot modify them.

3. Once you correct the entry click *Correct/Release* to reprocess it. Otherwise the record is not available for billing.

   Once reprocessed the record is no longer available through MPIU.
4. If you cannot correct the entry do one of the following:
   - » Click *Close* to leave the entry queued normally
   - » Click *Hold* to hold the record for later processing

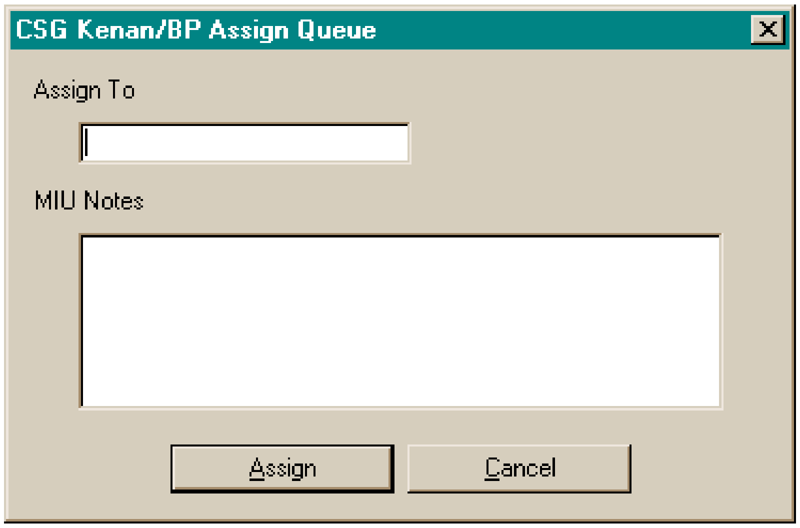
     MPIU filters on-hold records separately (see "Filtering Messages" on page 57) so that when you return to the main window, you must filter for on-hold records to see them again.
   - » Click *Assign* to assign the record to an MPIU user queue.

     The `Assign Queue` window appears (see figure 14).

     Enter the work queue name in the `Assign To` field and enter any notes in the `Miu Notes` field.
   - » Click *Uncorrectable* to mark the record uncorrectable.

     Uncorrectable records are not billed and do not appear in MPIU although they remain in `CDR_DATA_WORK` until explicitly dealt with. Be sure some process exists in your organization for handling uncorrectable records.

Figure 14 Assign Queue



You can select a block of records and click *Correct...* to correct them all at once using the `Message Global Correction` window (figure 13 on page 59). All fields in this window are blank; enter values in one or more to change those fields in all selected records. For example, enter **`123`** in the `Provider ID` box to set the Provider ID of all selected records to **`123`**. This does not change any other fields in the selected records, nor does it affect other records.

Similarly, you can assign multiple records at one time, mark them uncorrectable, or click *Close* to close the `Message Global Correction` window without applying corrections.

Once you have corrected a set of records (either through the `Message Global Correction` window or by making changes to the underlying MPS configuration, such as adding missing rates) you can release those records all at once — select them and click *Correct/Release*. MPIU reprocesses the selected records and reports how many were successfully reprocessed and how many failed reprocessing.

You can also use the MIU module to release large numbers of corrected records as described in the following section.

MIU Batch Processing

MIU is a noninteractive module that releases corrected `CDR_DATA_WORK` rows. For example, you can use MIU to process and release a global correction to numerous failed transactions.

Define a UTL process to run MIU as described in “Running Other Processes (UTL)” on page 126. Use an SQL Query value of the following form:

`MIU miu##` *`server`* [**`output`**] [*`SQLquery`*]

where:

- *`##`* is a two-digit number (**`01`** to **`99`**)

  This defines the MIU process ID, which is unique on a given server. If you use an existing ID you are scheduling a second task for that instance.
- `server` specifies the server ID against which MIU runs

  In MSA deployments run MIU against the Admin server (see “Modules and Servers” on page 16)
- the optional output argument causes MIU to generate detail records for consolidated usage (see below)
- the optional *`SQLquery`* constrains the records against which MIU runs

For example, **`MIU miu09 2 'FILE_STATUS.file_id = 5'`** runs MIU against failed usage records from a particular usage file.

MIU creates log files and (sometimes) detail files.

Correcting Consolidated Usage

If a usage event that normally is consolidated failed processing MIU generates detail records as CAP does (in a *filename*`.detail.MIU` file or a *ID_dates*`.detail.`*counter* file; see “Consolidated Usage Detail Files” on page 52 for more information), but MPIU does not. You can generate detail records for consolidated events corrected by MPIU by running MIU in a special “output” task mode as follows:

`MIU miu## 2 output` *`sql_query`*

Reprocessing Usage (RAP)

RAP reprocesses successfully rated usage in cases where rates or other usage configuration was defined incorrectly in the database, and usage was guided and rated based on those rates.

If event time unit credits are active, RAP does not reprocess usage, but creates a list of CDRs for CAP to re-rate. See the *System Administrator Guide* for more information.

Before You Run RAP

Before you run RAP to reprocess usage be sure the following conditions apply to that usage:

- It has been successfully guided and rated.

  If usage failed guiding or rating use MPIU to correct the errors as described in "Correcting Errors (MPIU and MIU)" on page 56.
- Correct usage configuration now exists in the database.

  See "Before You Run CAP" on page 50 for more information.
- It is unbilled.

  If incorrectly processed usage has been billed already you must back out the relevant bills (see "Backing Out Bills" on page 78) before you run RAP to reprocess the usage. You then can regenerate the bills.

  Do not run RAP if you already have dispatched the bills with incorrect rates. Instead issue adjustments or non-recurring charges against the account to correct the balance.
- Original record information is available.

  If usage was consolidated by CAP, if the B-number was truncated due to privacy levels, or if usage record information was deleted after guiding and rating, RAP cannot reprocess it.

  Instead issue adjustments or non-recurring charges against the account to correct the balance, or back out the bill and reprocess the original file.

See the Kenan/BP *User Guide* for more information about issuing adjustments and charges.

Defining RAP Tasks

RAP is a utility, so define a UTL process to run it as described in "Running Other Processes (UTL)" on page 126. Use an SQL Query value of the following form:

`RAP rap``##` *`server`* [**`reguide`**] *`SQLquery`*

where:

- `##` is a two-digit number

  This defines the RAP process ID, which is unique on a given server. If you use an existing ID you are scheduling a second task for that instance.
- *`server`* specifies the server ID against which RAP runs

  In MSA deployments run RAP separately against each Customer server (see "Modules and Servers" on page 16).
- the **`reguide`** argument indicates that RAP should guide and rate specified unbilled usage

  If this argument is not supplied RAP rerates but does not reguide usage.
- *`SQLquery`* constrains the records against which RAP runs

  This is a constraint for the `CDR_DATA` and `CMF` tables.

  The `no_bill` and `rev_rcv_cost_ctr` fields appear in more than one table, so specify the table name for them (for example, **`CDR_DATA.no_bill = "1"`**).

  `SQLquery` is required. To run RAP unconstrained use an *`SQLquery`* that is always true, such as **`where 3 > 1`**.

For example, to rerate records from external contact 17 enter
`RAP rap01 1 where ext_contact_id like "17"` in the `SQL Query` field.

Scheduling RAP

To schedule RAP schedule the UTL instance that runs it (see “Running Other Processes (UTL)” on page 126 for more information).

Run RAP as needed. Do not configure it to run on a regular schedule. For general scheduling instructions see “Configuring Processes” on page 30.

Reviewing RAP Output

You can review RAP control reports and activity logs as described in “Reviewing Process Activity” on page 36. The control report shows usage records that failed guiding and rating process, and the number of records successfully and unsuccessfully processed.

You also can check records in `CDR_DATA` to ensure they have been correctly rated (see “Troubleshooting CAP” on page 53 for more information).

5 Bill Processing

This chapter discusses the bill processing system, which calculates, formats, and dispatches all customer bills.

Bill Processing Overview

Bill processing modules perform the following tasks:

- The Bill Preparer (*BIP*) collects and calculates charges and credits for each account due to be billed as described in "Calculating Bills (BIP)" on page 67.

  In some cases BIP uses third-party tax packages to calculate tax charges; in other cases BIP calculates tax charges directly. BIP also collects taxes calculated at rating time by CAP.

  BIP creates one or more *bill records* in the database for each billed amount as well as summary records for each invoice.
- The Historic Discount Processor (*HDP*) calculates discounts and paybacks under historic contracts based on past contributions.

  This module must be run in coordination with BIP. BIP accumulates historic contract contributions each bill cycle. HDP calculates discounts and paybacks based on past contributions, and BIP uses these to apply discounts to target charges of historic contracts and to apply prepayments.
- The Bill Formatter (*BIF*) and invoice generator format bill records based on each customer's preferred language, currency, bill format, and other provisioned information. See "Formatting Bills (BIF)" on page 81 and "Formatting Bills (Invoice Generator)" on page 86 for more information.

  Your deployment uses BIF, or the invoice generator, or a custom bill-formatting solution.
- The Bill Dispatcher (*BID*) dispatches formatted bills to a printer or other distribution device as described in "Dispatching Bills (BID)" on page 88.

  Some dispatch methods (such as e-mail and HTML) cannot use BID.
- The Customer Care interface allows customer service representatives (CSRs) to schedule interim bills in response to customer requests.
- Various interfaces allow CSRs to review charges and bills as described in "Viewing Bills" on page 90.
- BIP cancels or reverses already calculated bills as necessary as described in "Backing Out Bills" on page 78.

The BIP, BIF, and Invoice Generator modules support real-time billing (also called *hot billing* or *billing on demand*). CSRs using the Customer Care interface can initiate an interim bill that the BIP and BIF modules calculate and format immediately.

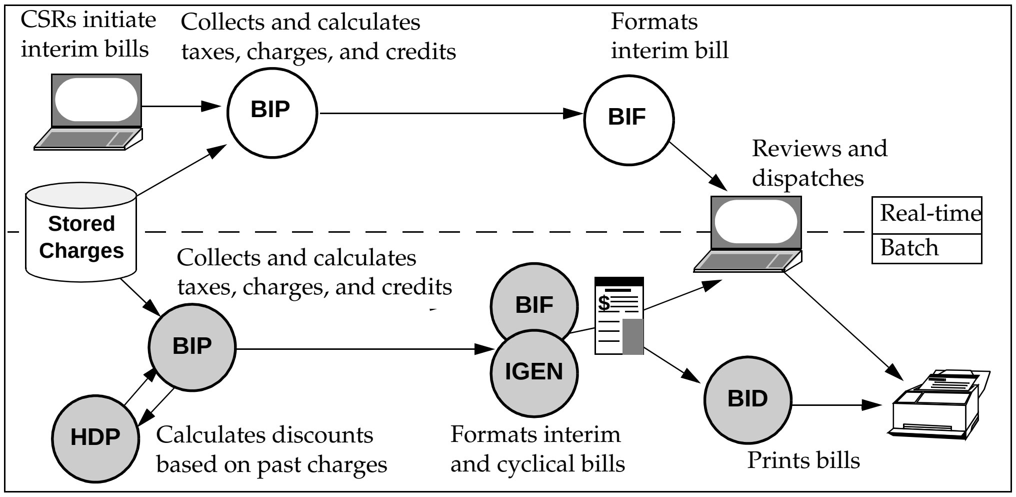
To set up real-time billing run BIF (or Invoice Generator) and BIP (in that order) in a special Flash mode in which they run in the background and respond to messages over an XIPC message queue. BIP calculates the charges on the bill before it is formatted, but BIF or Invoice Generator must be running first to be ready to format bills as soon as they are prepared. IF either quits running, you must restart it. See "Defining BIP Tasks" on page 69 and "Defining BIF Tasks" on page 82 for more information on Flash mode.

Even if your deployment uses real-time billing you must schedule batch runs to generate cyclical bills.

Note Note that Kenan/BP tracks the billed status of each credit and charge so it appears only once no matter how many interim and cyclical bills are produced for a given customer.

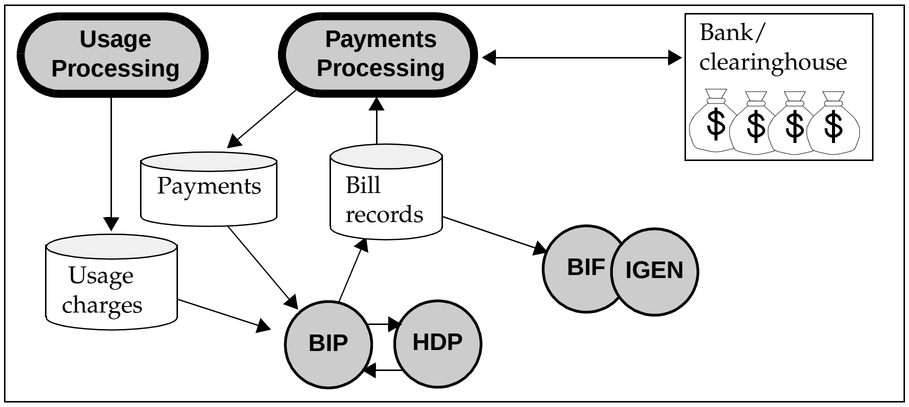
Figure 15 illustrates how the bill processing modules work together.

Figure 15 Bill Processing


CSRs initiate interim bills
Collects and calculates taxes, charges, and credits
BIP
Formats interim bill
BIF
Reviews and dispatches
Real-time
Batch
Stored Charges
Collects and calculates taxes, charges, and credits
BIF
$
BIP
IGEN
BID
HDP
Calculates discounts based on past charges
Formats interim and cyclical bills
Prints bills

The BIP and BIF modules process items such as payments and usage charges which enter the database through other modules. In addition, the payments modules sometimes depend on BIP output. Figure 16 illustrates these interactions.

Figure 16 Bills, Usage, and Payments



Calculating Bills (BIP)

In general the Bill Preparer (BIP) selects a set of accounts to bill, collects and calculates billed amounts for each account, and generates unformatted *bill records* for those amounts. Specific BIP behavior depends on the BIP task mode, however. For example:

- accounts

  BIP selects different sets of accounts in *Production*, *HQ*, and other modes.
- billed amounts

  In *Disconnect-Only* mode BIP produces disconnect credits only.

  In other modes BIP collects/calculates usage charges, non-recurring charges, recurring charges, taxes, discounts, adjustments, late fees, and other billable amounts.

- updates

  In most cases BIP updates account balances for use by other modules (for example, payment modules) and marks charges and credits so they are not billed twice.

  In *Pro Forma* mode BIP does not update balances or mark billed amounts.

Production, HQ, Disconnect-Only, Pro Forma, and other modes are described in more detail in “Defining BIP Tasks” on page 69.

Note BIP in Backout mode performs a completely different set of tasks. See “Backing Out Bills” on page 78 for more information.

Before You Run BIP

BIP preparation depends on the mode you define for each task (see “Defining BIP Tasks” on page 69 for a list of modes). In particular:

- Before you run BIP in Flash mode be sure BIF is already running in Flash mode (see “Bill Processing Overview” on page 66).
- Before you run BIP in Production and HQ modes make sure required charge-related information is available and current in the Kenan/BP database. In particular be sure that:
  - » Corrected usage charges exist in the database.

    Run MPS before BIP to ensure that the maximum amount of billable usage is collected as described in chapter 4.
  - » Products and other data model elements are configured properly as described in the *Guide to Products, Rates, and Discounts* and the *Configurator Guide*, and customer provisioning information is stored properly and up-to-date in the Customer database(s).

    Run BIP in Pro Forma mode (see below) against at least one account with each possible bill period to ensure that valid bill cycle dates exist for each bill period and review the BIP output to verify BIP configuration.

Note You need not run Pro Forma BIP against every Customer server in an MSA environment although you may wish to if they are very different from one another.

  Also, be sure all scheduled point-of-service bills for selected accounts have already been processed because BIP in Production mode collects charges regardless of their point-of-service status. This is seldom an issue because point-of-service bills are usually processed as hot bills immediately upon scheduling.
- Before you run BIP in Disconnect-Only mode confirm product configuration and provisioning.
- No special preparations are necessary before running BIP in Pro Forma mode — it makes no permanent changes and therefore runs safely regardless of the database state.

Note BIP in Backout mode requires additional preparation not described here. See “Backing Out Bills” on page 78 for more information.

Defining BIP Tasks

For general task definition instructions see “Configuring Processes” on page 30.

BIP’s default behavior and the effects of the `SQL Query` field depend on the task mode:

- **Production** —Required; generates bill records for non-HQ accounts due to be billed. See “Production Mode” on page 69.

  In MSA deployments run BIP in Production mode separately against each Customer server (see “Modules and Servers” on page 16).
- **Flash** — Supports real-time billing from the Customer Care interface. See “Bill Processing Overview” on page 66.
- **Pro Forma** — Optional; primarily for testing BIP configuration. Similar to Production mode except the customer billing state remains unchanged. See “Pro Forma Mode” on page 70.
- **Disconnect-Only** — Optional; primarily for improving BIP performance. Generates disconnect credits; does nothing else. See “Disconnect-Only Mode” on page 71.
- **Backout** — Reverses transactions performed when generating one or more bills. Use as needed to undo flawed bills. See “Backing Out Bills” on page 78.
- **HQ Discount** — Required if HQ discounts exist in your database; generates bill records for HQ accounts due to be billed. See “HQ Mode” on page 70.

  If necessary run BIP in HQ and Production mode against each Customer server in MSA deployments. Otherwise do not use HQ mode.
- **Pro Forma HQ** — Optional; used primarily for testing BIP configuration. Similar to HQ mode except the customer billing state remains unchanged. See “Pro Forma Mode” on page 70.

Production Mode

In general, BIP in Production mode generates bill records for non-HQ accounts (see “HQ Mode” on page 70) where the cutoff date for amounts appearing on the next bill is more than a given number of days ago (called *prep delay*).

For example, suppose the prep delay for an account is two days, the cutoff date is January 15, and BIP runs only on even-numbered days. BIP does not produce a bill for that account until January 18.

This rule has several exceptions. For example, BIP does not produce bills for accounts with no new charges or credits and does produce interim bills based on an interim bill date instead of a cyclical cutoff date. For a complete description of how BIP selects accounts see the *Technical Reference*.

You can further restrict the account list by entering an SQL `where` clause for the `CMF`, `CMF_INTERIM_BILLS`, and `BILL_CYCLE` tables in the `SQL Query` field for the task. For example, enter an `SQL Query` value of **`pay_method = "1"`** for BIP to process only accounts paying by check. (See “Choosing an SQL Query Value” on page 32 for more information.)

To use the `account_no`, `bill_period`, or `billing_frequency` fields in `SQL Query` you must specify the table name because these fields appear in multiple tables. For example, to restrict BIP to accounts with account numbers less than 1000 use **`CMF.account_no < 1000`**.

HQ Mode

If your deployment does not use headquarters (HQ) discounts do not run BIP in HQ mode at all. If you are uncertain run BIP in HQ mode and see if it produces any bills. See the *Guide to Products, Rates, and Discounts* for more information about HQ discounts.

BIP in HQ mode generates bill records in much the same way as BIP in Production mode except it selects only HQ accounts. You can further restrict the account list by entering an SQL `where` clause for the `CMF` and `CONTRACT_ASSIGNMENTS_HQ` (abbreviated `CAH`) tables in the `SQL Query` field for the task. For example, enter an `SQL Query` value of **`pay_method = "1"`** for BIP to process only HQ accounts paying by check. (See “Choosing an SQL Query Value” on page 32 for more information.)

You cannot use the `account_no` field in an HQ SQL query. The most commonly used fields are `tracking_id` and `tracking_id_serv` in `CAH`.

Pro Forma Mode

BIP in Pro Forma mode generates bill records for all non-HQ accounts (see “HQ Mode” on page 70) whether they are scheduled to be billed or not. BIP can even include normally unbillable accounts in a Pro Forma run depending on the `IGNORE_PROFORMA_SUPPRESS` system parameter (see the *Module Configuration Guide* for more information).

Similarly, BIP in Pro Forma HQ mode generates bill records for all HQ accounts (see “HQ Mode” on page 70) whether they are scheduled to be billed or not.

You can further restrict the account list by entering an SQL `where` clause for the `CMF` and `CMF_INTERIM_BILLS` tables (in Pro Forma mode) or the `CMF` and `CONTRACT_ASSIGNMENTS_HQ` tables (in Pro Forma HQ mode) in the `SQL Query` field for the task. For example, enter an `SQL Query` value of **`pay_method = "1"`** for BIP to process only accounts paying by check. (See “Choosing an SQL Query Value” on page 32 for more information.)

To use the `account_no` field in `SQL Query` you must specify the table name because it appears in multiple tables. For example, to restrict BIP to accounts with account numbers less than 1000 use **`CMF.account_no < 1000`**.

The primary difference between Pro Forma and Production modes is that Pro Forma bills do not change the status of affected accounts. In effect, BIP in Pro Forma mode generates facsimile bill records for you to review. For example, suppose BIP in Pro Forma mode creates a bill for a particular account. You can format the bill using BIF, view it through various interfaces, and so forth. However, because the bill is only pro forma:

- BIP does not mark amounts as billed. The next time BIP runs it picks up the same amounts again.
- Payments modules like CPM do not generate payment requests for the billed amount.
- Even if the account is scheduled to be disconnected after this bill BIP does not disconnect the account.
- Even if a disconnect credit is due for a product BIP does not disconnect the product or create the credit.

You must later schedule BIP in Production mode to create an actual bill including those amounts.

Disconnect-Only Mode

BIP in Disconnect-Only mode does not generate bill records, calculate billed amounts, or update existing charge records. All BIP does in this mode is generate disconnect credits for recently disconnected products. See the *Guide to Products, Rates, and Discounts* for more information about product disconnect credits.

BIP generates disconnect credits in Production mode also, so Disconnect-Only mode is not necessary. You can run BIP in Disconnect-Only mode regularly to improve the performance of BIP in Production mode.

BIP in Disconnect-Only mode selects the same set of accounts as in Production mode (except those with no disconnect credits). You can further restrict the account by entering an SQL `where` clause for the `CMF` or `PRODUCT` tables in the `SQL Query` field for the task. For example, enter an `SQL Query` value of **`pay_method = "1"`** for BIP to process only accounts paying by check. (See "Choosing an SQL Query Value" on page 32 for more information.)

To use the following fields in `SQL Query` you must also specify the table name because they appear in more than one table:

`account_no`	`arch_flag`	`bill_period`
`component_id`	`contract_tracking_id`	`contract_tracking_id_serv`
`converted`	`date_created`	`date_rc_begin`
`date_rc_billed_through`	`disconnect_reason`	`element_id`
`in_arrears_override`	`no_bill`	`open_item_id`
`order_number`	`product_start`	`product_status`
`product_stop`	`sales_channel_id`	`serial_number`
`tracking_id`	`tracking_id_serv`	`usg_amt_limit`
`usg_units_limit`		

For example, to restrict BIP to accounts with account numbers less than 1000 use **`CMF.account_no < 1000`**.

Scheduling BIP

How often you run a BIP task depends on the mode you define for the task (see "Defining BIP Tasks" on page 69 for a list of modes). In particular:

- Production — Schedule BIP in production mode often enough to generate all cyclical bills. If CSRs in your organization frequently schedule interim bills for customers you can schedule BIP in production mode more often. See "What Days to Run BIP" below for more information.
- HQ — Schedule BIP in HQ mode often enough to generate all cyclical bills for HQ accounts (if any). See "What Days to Run BIP" on page 72 for more information.
- Disconnect-Only — Schedule BIP in Disconnect-Only mode as often as desired to improve BIP performance in Production and HQ modes.

- Pro Forma, Pro Forma HQ, Backout — Schedule BIP in these modes for one-time runs as needed.
- Flash — Run BIP in Flash mode once after running BIF in Flash mode. BIP continues to run in the background once started.

Do not run BIP in conflict with any other modules. See “What Times to Run BIP” on page 72 for more information.

What Days to Run BIP

Run BIP once for every bill cycle shortly after the cutoff date and prep delay for that cycle. For example, if all customers share a single bill cycle with a cutoff date of the first of each month and a prep delay of three days run BIP on the fourth of each month to bill all your accounts.

As another example, suppose your customers are distributed over four separate bill cycles, each with cutoff dates on Mondays 28 days apart and a three-day prep delay. In other words, in 2004 one cycle cuts off on Jan 5, a second on Jan 12, a third on Jan 19, and so forth. Run BIP every Thursday (that is, on Jan 8, Jan 15, Jan 22, and so forth in 2004) to bill a different set of accounts each week. No action is necessary to constrain BIP to those accounts because BIP bills only those accounts due to be billed.

See “Recurring Schedules” on page 21 for more information about bill cycles.

If you are creating an interim bill, note that BIP generates only one interim bill per account in a given run. If more than one `CMF_INTERIM_BILLS` row exists for the account, BIP selects the earliest. If CSRs in your deployment frequently schedule interim bills for accounts you may wish to run BIP more often to pick up those scheduled bills. (If your deployment supports hot billing this is probably unnecessary because interim bills are handled by the Flash BIP.) You can only schedule interim bills that precede the next cyclical bill. When BIP generates the latest interim bill for an account, it also generates the next cyclical bill. (See the *User Guide* for more information about scheduling interim bills.)

What Times to Run BIP

Bill calculation is processor-intensive, so run BIP overnight or during off-peak periods to avoid conflict with other operations.

Caution	Do not run BIP simultaneously with any other billing operations modules or when the Manual Replication Utility is running.

You can run multiple BIP instances simultaneously on different subsets of billable accounts using the `SQL Query` field as described in “Defining BIP Tasks” on page 69. If you do, you must run a final BIP instance against the entire billable account list to ensure that all accounts are billed.

Run MPS before BIP to ensure that the maximum amount of billable usage is collected as described in chapter 4.

For general scheduling instructions see “Configuring Processes” on page 30.

Reviewing BIP Output

Perform the following three steps to review BIP output:

1. Check the Processes page in Operations Center shortly after BIP starts to run. Because any BIP errors usually occur immediately, you can resolve them and reschedule BIP with minimum loss of bill cycle time.
2. Check BIP control reports and activity logs for errors after BIP runs. See “Control Reports and Activity Logs” on page 73 for more information.

   Check for errors. BIP generates bill records even if an error occurs. The errored records are flagged, and BID does not dispatch them.
3. Run BIF against a small, representative set of bills (see “Formatting Bills (BIF)” on page 81) and review the formatted bills (see “Viewing Bills” on page 90) after each BIP run to ensure the proper billed amounts appear on the invoice.

   Check database bill records for a representative set of bills in the database. See “Database Tables” on page 74 for more information.

Control Reports and Activity Logs

Check BIP control reports and activity logs for errors as described in “Reviewing Process Activity” on page 36. In particular, check the following control report sections:

- TOTAL UNACCOUNTED FOR in the Results Summary
  MISSING column

  Should contain only zeros. Otherwise BIP has probably aborted. Check the activity log file and see “Errors during Billing” on page 76 for more information.
- TOTAL IN ERROR in the Results Summary
  ERROR column

  Should contain only zeros. Otherwise a bill failed processing. See “Errors during Billing” on page 76 for more information.
- SKIP column

  Non-zero amounts in this column indicate bills skipped because of no new activity. This is not necessarily an error; for example, a newly provisioned account may legitimately have no billed amounts. However, if many bills are being skipped this may indicate that new charge amounts are not being properly created. See “Incorrect Billed Amounts” on page 77 for more information.
- LOCKED column

  Non-zero amounts in this column indicate bills processed by another BIP instance. This is not an error but rescheduling BIP to avoid overlapping accounts should improve performance in the future. See “Scheduling BIP” on page 71 for more information.

The BIP activity log contains a detailed view of BIP activities for each account. If an account is not billed see “Unbilled Account” on page 76 for more information.

BIP log files include information on the number of usage records, recurring charges, and non-recurring charges processed. Set the `CHARGE_COUNT_LOG_THRESHOLD` (BIP) system parameter to a number greater than zero to log these totals. The parameter value is the threshold for process work logging. BIP monitors the number of CDRs for each account. When this count reaches the threshold number, BIP writes the CDR data to the log file. A line is written to the log file for:

- total recurring charges for each service instance
- total CDRs processed for each service instance

- cumulative recurring charges for the account after each service instance's total
- cumulative CDRs processed for the account after each service instance's total
- total NRCs processed for each account
- every <n> recurring charges processed for a service instance (<n> is the value of the `CHARGE_COUNT_LOG_THRESHOLD` system parameter)
- every <n> CDRs processed for a service instance (<n> is the value of the `CHARGE_COUNT_LOG_THRESHOLD` system parameter)
- every <n> NRCs processed for an account (<n> is the value of the `CHARGE_COUNT_LOG_THRESHOLD` system parameter)

Each message has severity of INFORM; BIP does not generate them if its log level is LOW.

Example: Account number 1 has three service instances:

- Service instance 1 has 17 CDRs and one recurring charge
- Service instance 2 has 9 CDRs
- Service instance 3 has 15 CDRs and one recurring charge

With the system parameter `CHARGE_COUNT_LOG_THRESHOLD` (BIP) set to 10, the following log messages are generated (message text only shown):

```
10 CDRs processed for service no 1,0
17 CDRs procesesd for service no 1, 0
1 RCs processed for service no 1, 0
17 CDRs processed for account no 1
1 RCs processed for account no 1
9 CDRs processed for service no 2,0
26 CDRs processed for account no 1
1 RCs processed for account no 1
10 CDRs processed for service no 3,0
15 CDRs processed for service no 3,0
1 RCs processed for service no 3,0
41 CDRs processed for account no 1
2 RCs processed for account no 1
1 NRCs processed for account no 1
```

Database Tables

`BILL_INVOICE` is the primary bill record table and contains one record for every bill. Each record contains general bill information such as cutoff date, dispatch method, and so forth.

- Check `BILL_INVOICE.prep_status` to determine bill status:
  - 0 — Incompletely processed; possible processing errors
  - 1 — Successful bill
  - 2 — Bill not produced due to no new activity
  - 3 — Bill "on hold" pending payment
  - 4 — Pro forma bill

5 — Bill not produced due to low balance

1003 — Backout

1004 — Suppress backout

- Check BILL_INVOICE.prep_error_code for the number of errors encountered by BIP.
- For cyclical bills check date fields listed in table 13 against the corresponding BILL_CYCLE fields. If these do not match adjust the BILL_CYCLE configuration as described in the *Guide to Products, Rates, and Discounts* and the *Configurator Guide*.

Table 13 BILL_INVOICE Date Fields

Field	Expected Value
from_date	Same as cutoff_date of previous BILL_CYCLE row
to_date	Same as cutoff_date of current BILL_CYCLE row
next_to_date	Same as cutoff_date of subsequent BILL_CYCLE row
statement_date	Same as statement_date of current BILL_CYCLE row
payment_due_date	Same as ppdd_date of current BILL_CYCLE row

Secondary bill record tables include:

- BILL_INVOICE_DETAIL — contains multiple records for every bill. Each record includes either detailed information for a single transaction or a summary of several transactions.
- CDR_BILLED — contains multiple records for each usage summary record in BILL_INVOICE_DETAIL. Each record includes detailed information for a billed usage event. Each row in CDR_BILLED contains a pointer to a row in CDR_DATA plus additional information calculated by BIP.
- BILL_INVOICE_TAX and BILL_INVOICE_DISCOUNT — contain multiple detail records for some, none, or all of the tax/discount rows in BILL_INVOICE_DETAIL

You can verify whether a CMF_BALANCE record exists for the bill. If not, then BIP did not successfully process the bill. (Do not alter the CMF_BALANCE table manually — other modules depend on it.)

You can check BIP output in many other database tables — BIP resets bill dates, marks items as billed, deletes unbilled usage records, and much more. See the *Technical Reference* for more specific information about BIP's database output.

Troubleshooting BIP

This section includes troubleshooting suggestions for the following problems:

- BIP logs errors. See "Errors during Billing" on page 76.
- BIP logs no errors but does not produce bill records for a particular account. See "Unbilled Account" on page 76.
- BIP logs no errors but incorrect amounts appear on an invoice. See "Incorrect Billed Amounts" on page 77.
- BIP cannot process an HQ group in HQ mode. See "Errors Processing HQ Discounts" on page 78.

Once you identify the cause of your BIP problem:

1. Correct the problem.
2. Back out the bill if one was created.

   See "Backing Out Bills" on page 78.
3. Rerun BIP to generate a bill (or to generate a new bill, if you backed one out).

Note Sometimes resolving a class of problems together is easier than correcting each problem individually. For example, you may need to reconfigure your database, retrain CSRs, or revise your internal methods and procedures.

Errors during Billing

BIP writes a record for each errored bill to `ERROR_BILL` identifying the bill, the account, the task during which the error occurred, and the nature of the error. The `error_desc` field contains a description of the error. For example, problems with `BILL_CYCLE` would result in one of the following values in `ERROR_BILL.error_desc`:

- No `BILL_CYCLE` row for current ppdd *date*, period *period*.
- No `BILL_CYCLE` row for previous ppdd *date*, period *period*.
- No `BILL_CYCLE` row subsequent to ppdd *date* period *period*.
- No `BILL_CYCLE` for RC *startdate* to *enddate*, period *period*.

where *date*, *startdate*, *enddate*, and *period* are variable items populated by BIP. These errors indicate that additional rows must be added to the `BILL_CYCLE` table.

If this description is not sufficiently clear a longer description may exist in the `description` field of the `SYSTEM_MESSAGES` entry whose `message_number` corresponds to `ERROR_BILL.manual_code`. The last four digits of `manual_code` are the same as the `message_number`, and the first digit of `manual_code` indicates the severity of the error.

For example, the first error listed above might have the following longer description:

```
The BILL_CYCLE table does not contain a bill_period and ppdd_date
that matches, respectively, CMF.bill_period and CMF.next_bill_date.
```

Sometimes an "error" is not a problem but a notice of an unusual situation. For example:

- Sometimes BIP errors a bill if it encounters a charge with an effective date earlier than the beginning of the current bill cycle.

  BIP automatically schedules an interim bill to pick up those charges. Rerun BIP.
- If an HQ discount amount allocated to an account exceeds the total new charges on that account BIP discards the extra discount and logs an error (BIP_HQ_ALLOC) to notify you of its action. Normally no correction is necessary.

Unbilled Account

An instance of BIP can fail to pick up an account for several reasons:

- no new account activity or insufficient account balance

  BIP does not generate bills; this is not an error.

However, if the account balance is implausibly high confirm the threshold provisioned for the account (check `CMF.charge_threshhold` or use the Customer Care interface as described in the *User Guide*). You can lower this threshold.

Total activity for HQ groups can be complicated. If the BIP log file indicates insufficient activity for an HQ account see “Errors Processing HQ Discounts” on page 78 for more information.

- The account is provisioned as unbilled.

  BIP does not generate bills; this is not an error.

  Check `CMF.no_bill` for the account or use the Customer Care interface as described in the *User Guide*. You can change this setting.
- A different BIP instance picks up the account.

  The BIP control report lists accounts in the LOCKED column. This means the account was billed by the other BIP instance; this is not an error. However, you can reschedule BIP in the future to avoid overlapping accounts for performance reasons. See “Scheduling BIP” on page 71 for more information.
- The account is part of an HQ group.

  This is an error only in HQ mode. See “Errors Processing HQ Discounts” on page 78 for more information. Otherwise run a BIP instance in HQ mode (see “HQ Mode” on page 70).
- The account’s bill date is in the future.

  BIP does not generate bills; this is not an error.

  Check `CMF.next_bill_date` for the account. If it is incorrect the customer probably has the wrong bill cycle. You can assign the customer a new bill cycle; see the *User Guide*.

  Remember that BIP updates `next_bill_date` after running successfully, so `CMF` records can appear to have `next_bill_date` set too late for the previous BIP run when in fact they have been set correctly for the next BIP run. Therefore, be sure that the accounts you are looking at have not been billed successfully.

See the *Technical Reference* for more information about how BIP selects accounts.

In most of these cases you do not need to back out bills because no bill was produced.

Incorrect Billed Amounts

If incorrect amounts appear on an invoice but BIP logs no errors, check the following as possible causes:

- incorrectly configured charge-related items

  If all incorrect amounts share a common product, discount, usage type, or similar configured item, double-check the item configuration as described in the *Guide to Products, Rates, and Discounts* and the *Configurator Guide*.
- incorrectly provisioned accounts

  If all incorrect amounts share a common account or service instance a CSR may have provisioned the wrong product, contract, or similar item to that account. Double-check the account configuration as described in the *User Guide* and your own methods and procedures documents.
- usage processing errors

  If all incorrect amounts are usage charges of a particular type double-check the MPS output for this bill cycle as described in chapter 4.

Errors Processing HQ Discounts

In HQ mode BIP attempts to synchronize and lock all the accounts in an HQ group before processing any of them. If this does not work (for example, if an account is already being processed by another BIP instance) the entire group is not processed.

If an HQ group fails processing because of a problem with individual accounts you can temporarily exclude those accounts from the HQ group. Set `CONTRACT_ASSIGNMENTS_HQ.exclude` to 1 for the account(s).

While this flag is set the account is processed neither as an HQ account nor as a non-HQ account. You must unset the flag before the account can be billed normally. In the meantime the remainder of the HQ group is processed excluding any amounts contributed by the excluded account.

For example, if an account's `CMF.next_bill_date` is different from the group (but has the same billing cycle) you can exclude all other accounts from the HQ group and run an HQ mode BIP on the group to synchronize the excluded account's `next_bill_date` with that of the rest of the group. When several accounts have different `next_bill_date` values you must do this several times.

An error in group setup occurs if the accounts have different bill cycles. Either change the account bill cycles to be identical or remove the accounts from the HQ group. The same is true for other group-membership criteria.

Note You must reset the `exclude` bit to 0 when you have finished.

See the *Technical Reference* for more detailed information about how BIP processes HQ and other accounts.

Backing Out Bills

In Backout mode, BIP backs out one or more bills — that is, marks the unformatted bill records obsolete, issues adjustments to compensate for existing billed amounts, and generally cancels or reverses the database changes made during Production mode.

Warning! Backout mode can cause reconciliation and journaling errors. Use it only as a last resort for resolving BIP problems.

For example, if the `RATE_RC` table includes incorrect monthly service rates for a product and BIP therefore calculates incorrect monthly charges for that product on a set of customer bills you can run BIP in Backout mode to reverse those charges (as well as all the other charges on those bills).

To run a bill backout, perform the following six steps:

1. Select the bills to be backed out (see "Marking Bills for Backout" on page 91).

   The bill must be explicitly selected for backout.
2. Define and run a BIP task in Backout mode (see "Defining BIF Tasks" on page 82).

   You can constrain the list of bills by entering an SQL `where` clause for the `CMF`, `BILL_INVOICE` (abbreviated `BI`) and `CMF_BALANCE` (abbreviated `CB`) tables in the `SQL Query` field for the task.

For example, use:
`BI.bill_ref_no > 1000 and BI.bill_ref_resets = 0`
to constrain backout to invoice numbers greater than 1000.

Many fields (`account_no`, `account_status`, `bill_disp_meth`, `bill_hold_code`, `bill_period`, `bill_ref_no`, `bill_ref_resets`, `bill_sequence_num`, and many more) are shared by multiple tables. The simplest solution is to include a specific table reference for every field in the SQL query.

3. Resolve whatever errors caused the original problem.
4. Set `CMF.no_bill` to 0 for the accounts whose bills were backed out either directly in the database or by checking "Suppress Billing" in the Billing Information window of the Customer Care interface.
5. Resynchronize HQ groups if necessary.

   Backing out bills for accounts in an HQ group can desynchronize the group. See "HQ Mode" on page 70 for more information.
6. Run a normal BIP to recreate the bills.

If multiple bills for a single account are marked for backout BIP processes the most recent ones first.

Note BIP maintains a full audit trail of each backout — it does not delete the original database records.

BIP tracks the charges, currency conversions, cancellations, and reversals created during initial invoicing, backout, and subsequent reinvoicing and reconciles them for payments, journals, and other downstream financial procedures.

Evaluating Historic Contracts (HDP)

The Historic Discount Processor (HDP) evaluates historic discount and historic payback contracts based on contributions written to `HISTORIC_CONTRIBUTION` by BIP. The next time BIP run, it applies the discount rates from HDP calculations to target charges and apply paybacks as prepayments.

HDP does not run in Flash mode.

Before You Run HDP

HDP's behavior depends on the mode you define for each task (see "Defining HDP Tasks" on page 80 for a list of modes).

In particular, before you run HDP to evaluate historic contracts in normal or Pro Forma mode, make sure that BIP has run and written all contributions for the current contract evaluation cycles to `HISTORIC_CONTRIBUTIONS`.

Defining HDP Tasks

For general task definition instructions see “Configuring Processes” on page 30.

HDP can run in three distinct task modes: normal contract evaluation mode (`task_mode` = 0), pro forma mode (`task_mode` = 4), and backout mode (`task_mode` = 6).

- **Normal contract evaluation mode —** Totals contributions made over the evaluation period and marks them as evaluated. For historic discount contracts, HDP then writes the contribution to `HISTORIC_THRESHOLD` as a discount threshold. For historic payback contracts, HDP then determines a discount rate, calculates a payback amount, and writes it to `PREPAYMENTS`.

  Constrain the set of historic contracts HDP evaluates by entering an SQL `where` clause for the `CUSTOMER_CONTRACTS` (abbreviated `CC`) and `CONTRACT_TYPES` (abbreviated `CT`) tables. By default, HDP evaluates all historic discount and payback contracts that have reached their `next_eval_date` + `prep_delay` as defined in `BILL_CYCLE` and read by HDP in `CUSTOMER_CONTRACTS`.

  Some possible `SQL Query` values are:

  - » **`CC.next_eval_date = ‘01/01/2001’`**

    Processes only historic contracts with a next evaluation date of Jan 01, 2001.
  - » **`CC.next_eval_date < ‘02/01/2001’`**

    Processes only historic contracts with the next evaluation date before Feb 01, 2001.
  - » **`CC.tracking_id = 105 and CC.tracking_id_serv = 3`**

    Processes only the specified historic contract.
  - » **`CC.contract_type in (36, 37, 38, 39)`**

    Processes only historic contracts of the specified type.
  - » **`CT.contract_category = 12`**

    Processes only historic discount contracts.
- **Pro Forma mode** — Optional; primarily for testing HDP configuration. HDP in Pro Forma mode evaluates all contracts, not only those scheduled for normal evaluation. The difference between Pro Forma and Normal modes is that BIP never uses the discount thresholds and paybacks calculated, and the contracts are not marked as evaluated.
- **Backout mode** — HDP reverses the results of the last normal contract evaluation mode HDP. It looks for any historic contract with a `prev_eval_date` specified and tries to reverse the effects of the last historic discount evaluation by marking any `HISTORIC_THRESHOLD` rows created as backed-out, cancelling any `PREPAYMENT` rows created, and marking any `HISTORIC_CONTRIBUTION` rows marked as evaluated as unevaluated.

Scheduling HDP

HDP is scheduled and run as a batch process. HDP can be scheduled through Operations Center and run in recurring or non-recurring mode.

Run HDP in coordination with BIP. If BIP is delayed or has not run, HDP cannot access all the required contract information and evaluates some contracts with incompletely recorded contributions. In that case the accounts associated with the incomplete contributions do not receive the full benefit of the historic contract.

Multiple HDP processes can run in tandem. Each process locks the contracts it is evaluating to prevent interference from other HDP processes. HDP processes can run in tandem with BIP, but it is important to ensure that HDP does not run on a contract for which BIP has not processed all appropriate contributions.

For general scheduling instructions, see “Configuring Processes” on page 30.

Reviewing HDP Output

You can review HDP control reports and activity logs as described in “Reviewing Process Activity” on page 36.

HDP outputs results to `HISTORIC_THRESHOLD` and `PREPAYMENTS`.

Troubleshooting HDP

Check the control report for errors. HDP may or may not log errors, but produce incorrect discount thresholds or paybacks, or fail to evaluate contracts.

Once you identify the cause of your HDP problem:

1. Resolve the problem.
2. Run HDP in Backout mode.
3. Rerun HDP to reevaluate all backed out contracts.

Incorrect discount thresholds or paybacks probably result from missing contributions caused by problems with a recent BIP run or scheduling HDP improperly. See “Troubleshooting BIP” on page 75 for BIP problems, or back out and rerun HDP if a scheduling problem occurred.

Note If a BIP run is backed out and rerun, any HDP runs that follow it must be backed out and rerun.

HDP failure to evaluate a contract type usually results from an error in contract configuration. Double-check the contract configuration as described in the *Guide to Products, Rates, and Discounts* and the *Configurator Guide*.

Formatting Bills (BIF)

The Bill Formatter (BIF) retrieves bill records BIP writes and produces formatted bills that include the billed amounts on those records. You can then dispatch the formatted bills to customers (see “Dispatching Bills (BID)” on page 88) or review them online (see “Viewing Bills” on page 90).

Your deployment uses BIF, or the invoice generator, or a custom bill-formatting solution.

Before You Run BIF

Before you run BIF to format bills for dispatch be sure the bill records are ready to dispatch — that is, be sure `BILL_INVOICE.prep_status` for those records is 1 (normal). See “Reviewing BIP Output” on page 73 for more information. You can format other bills for review, but you cannot dispatch them later.

If you changed BIF configuration or bill formats since its last run, run BIF on some pro forma bill records first. That is, run a Pro Forma BIP against a representative set of accounts and then run BIF against those bill records. See “Defining BIP Tasks” on page 69 for more information on Pro Forma BIP and see the *Module Configuration Guide* for more information on BIF configuration and bill formats.

Before you run BIF in Flash mode be sure XIPC is properly configured on the server and on all client machines (see “Bill Processing Overview” on page 66).

Defining BIF Tasks

In MSA deployments run BIF separately against each Customer server (see “Modules and Servers” on page 16).

In general BIF creates one formatted bill for each set of unformatted bill records BIP creates. You can constrain this list by entering an SQL `where` clause for the `BILL_INVOICE` table in the `SQL Query` field for the task. You can also enter a `where` clause that is a join on `BILL_INVOICE` and other tables. For example:

- Enter **`prep_status=4`** to format only pro forma bills.
- Enter **`pay_method=2`** to format only bills paid by credit card.
- Enter **`bill_ref_no in (25, 26, 27)`** to format only the three bills with those invoice IDs.

  (Strictly speaking, `bill_ref_no` does not uniquely identify an invoice, so a more accurate `SQL Query` value is
  `bill_ref_no in (25, 26, 27), bill_ref_resets=0`.
  However, probably only one set of unformatted bills with those IDs exists in the database.)
- Enter **`CMF.bill_city = “Cambridge” and CMF.account_no = BILL_ INVOICE.account_no`** to format only bills for customers with billing addresses in Cambridge.
- Enter **`CMF.bill_format_opt = 100 and CMF.account_no = BILL_ INVOICE.account_no`** to format only bills with bill format 100.

Depending on BIP, BIF, and payment module configuration BIF ignores certain bills until payment is complete. See “Processing Credit Card Payments (CPM)” on page 102 for more information.

If you are using hot billing, run BIF in Flash mode to support hot billing on demand from BIP (see “Bill Processing Overview” on page 66). Otherwise, schedule BIF in Production mode.

For general task definition instructions see “Configuring Processes” on page 30.

Embedded Graphics in PCL Files

BIF supports embedded bitmapped graphics in BIF PCL files. Follow these two steps:

1. Create a monochrome (1-bit) image in JPEG or BMP format and store it in the `$ARBORSITE/BIFResources/user` directory.
2. To position the image on the invoice, add a DEFINE SECTION statement with an HP Image Filename argument to the fields.def resource file. See chapter 2 of the *Module Configuration Guide* for more information on working with BIF resource files.

When fields.def is read, the graphics file referenced in the HP Image Filename argument is written into the .dat file. During initialization the graphics file is cached in memory, and at output time it is converted into PCL instructions.

Scheduling BIF

You can schedule BIF to start formatting bills as soon as BIP finishes calculating them. However, you can schedule time between BIP and BIF to review BIP output.

BIP, BIF, and payment modules configuration may necessitate scheduling other modules between BIP and BIF. For example, if a bill format includes a note such as “This invoice is for information only. The charges will be taken from your credit card,” you can run CPM between BIP and BIF to ensure a successful credit-card transaction before formatting the invoice. (This also depends on how BIP and CPM are configured.)

For general scheduling instructions, see “Configuring Processes” on page 30.

If you are running BIF in Flash mode you need not schedule it. BIF in this mode runs in the background and responds to real-time requests from BIP.

Reviewing BIF Output

You can review BIF control reports and activity logs as described in “Reviewing Process Activity” on page 36.

You can review online images of formatted bills through various interfaces as described in “Viewing Bills” on page 90.

You also can view bill images directly from the following sources:

- ASCII bill images for all bills in the `BILL_IMAGE` table, with `image_type` = 0
- ASCII bill image files for all bills in the `$ARBORDATA/disp/screen` directory
- HTML bill images for all HTML-dispatched bills as described below

  You cannot view HTML images of non-HTML-dispatched bills.

Bill images exist on a server identified by the `BILL_IMAGE_SERVER_ID` (DB) system parameter if defined or the local Customer server. Bill images in the database typically are compressed to save space.

You can review dispatch-ready formatted bills in the following locations depending on the dispatch method:

- Print, FTP, or HTML

  Dispatch files in `$ARBORDATA/disp/ready` described in "Dispatch Files" on page 84.
- HTML

  Bill images in the `BILL_IMAGE` table (one row per page) with `image_type` = 1. This is in addition to the ASCII images described above. These typically are compressed to save space.
- E-mail — Bill files in `$ARBORDATA/disp/email/ready`.

  Each such file includes the recipient's e-mail address.

Note Kenan/BP does not automatically dispatch HTML or e-mail bills; you must dispatch them yourself using local tools.

You can review entries in `BILL_FILES` for each file which contain the following information:

- Batch ID and filename
- Size in bytes, bills, and pages
- Offset in bytes of the first bill
- BIF process ID and creation date and time for the file
- Printer or device
- Formatting status

You can review the status of a bill or a batch of formatted bills through the following database fields:

- `bill_hold_code` (1 = temporary hold; 2 = permanent hold) and `format_status` (-1 = error, 0 = not started, 1 = in progress, 2 = complete) in `BILL_INVOICE` for an invoice
- `format_status` in `BILL_BATCHES` for a bill batch (as above)

  Every batch has an identifier of the form <*serverID*><*processNum*><*createDateTime*> — for example, `bif01` on December 31 at 4:32 A.M. on server 3 creates a batch with ID `00311231043200` and twelve hours later a batch with ID `00311231163200`.

Dispatch Files

For printer and FTP dispatching dispatch files are print queue files sorted by Zip or postal code. The files appear in the `$ARBORDATA/disp/ready` directory with the filename:

`P`<*pay_method*><*server*><*process*><*date*><*time*><*csc*><*type*><*hold*>-<*counter*>

where:

- *pay_method* is one of the following:
  - » `H` for bills paid by check
  - » `C` for bills paid by credit card
  - » `D` for bills paid by direct debit
- *server* is the local server ID (3 digits, left-padded with zeros).
- *process* is the 1-digit BIF process number (0-9).
- *date* is the batch creation date, in `MMDD` format.
- *time* is the batch creation time, in `hhmmss` format.
- *csc* is the 5-digit service-center ID associated with the print queue (for example, `00001`)

- *type* is one of the following:
  - » `I` for the initial bill for a customer
  - » `F` for the final bill for a customer
  - » `P` for pro forma bills
  - » `A` for all bills that don't meet any of the above requirements
- *hold* is one of the following:
  - » `H` for hold bill files
  - » `E` for errored bills
  - » `B` for boxed bills (bill size >= 65 pages)
  - » `M` for manual bills (bill size 11-64 pages)
  - » `L` for flat bills (bill size 6-10 pages)
  - » `F` for fold bills (bill size 1-5 pages)

  Page counts are controlled by the `FOLDED_BILL`, `FLAT_BILL`, `MANUAL_BILL`, and `BOXED_BILL` system parameters (see the *Module Configuration Guide*).
- *counter* is a three-digit number incremented when the dispatch size exceeds the `FILE_SIZE` system parameter, at which point the file is closed and a new file is opened. This prevents the file from becoming too large.

For example, the filename `PH00310417145147000001IF-000` could be interpreted as follows:

P	Batch file of bills...
H	...paid by check...
003	...processed on server #3...
1	...by bif01...
0417	...on April 17th...
145147	...at 2:51:47 p.m....
00001	...via customer service print center 1...
I	...for initial bills...
F	...between 1 and 5 pages long (depending on system parameter settings)...
-000	...this is the first file in this batch.

The filename `PH00310417145147000001IF-001` indicates the second file of the same batch.

HTML dispatch files are standard .htm files, one invoice per file. The files appear in the `$ARBORDATA/disp/ready` directory with the filename:

`fmt`*<bill_ref_no>*`.htm`

where *<bill_ref_no>* is the invoice's `BILL_INVOICE_REF.bill_ref_no`. You can view .htm output files with Microsoft Internet Explorer or Netscape browsers.

Troubleshooting BIF

Check the control report for errors and review a representative sample of bills (see "Viewing Bills" on page 90) for incorrect totals, missing charges, incorrect messages or account information, and other problems:

- Incorrect charges (totals, charge lists, amounts, and so forth) usually result from problems with a recent BIP run. See "Troubleshooting BIP" on page 75.

- Incorrect formats (items in the wrong place, incorrect text, and so forth) usually result from BIF configuration errors. See the *Module Configuration Guide* for more information.

Make sure all the bills you expect have in fact been created. If bills are missing:

- Bill records may not have been created for those bills due to BIP problems (see “Troubleshooting BIP” on page 75).
- BIF may be holding those bills for payment (see “Processing Credit Card Payments (CPM)” on page 102).
- You may be using an incorrect SQL Query (see “Defining BIF Tasks” on page 82).

Formatting Bills (Invoice Generator)

The invoice generator retrieves bill records written by BIP and produces formatted bills that include the billed amounts on those records. You then can dispatch the formatted bills to customers (see “Dispatching Bills (BID)” on page 88) or review them online (see “Viewing Bills” on page 90).

Before You Run the Invoice Generator

Before you run the invoice generator to format bills for dispatch be sure the bill records are ready to dispatch — that is, be sure `BILL_INVOICE.prep_status` for those records is set to 1 (normal). See “Reviewing BIP Output” on page 73 for more information. You can format other bills for review, but you cannot dispatch them later.

Be sure the `$IGENLAYOUTFILE` environment variable on the invoice generator host correctly identifies the bill layout file to use. Sometimes you must run the invoice generator separately against different sets of customers and set a different bill layout file for each set.

If you changed the invoice generator configuration or bill formats since its last run, run the invoice generator on some pro forma bill records first. That is, run a Pro Forma BIP against a representative set of accounts and then run the invoice generator against those bill records. See “Defining BIF Tasks” on page 82 for more information on Pro Forma BIP, and see the *Kenan Invoice Design Guide to Invoice Generation* for more information on invoice generator configuration and bill formats.

Before you run IGEN in Flash mode be sure XIPC is properly configured on the server and on all client machines (see “Bill Processing Overview” on page 66).

Defining Invoice Generator Tasks

In MSA deployments run the invoice generator separately against each Customer server (see “Modules and Servers” on page 16).

In general, the invoice generator creates one formatted bill for each set of unformatted bill records created by BIP. You can constrain this list by entering an SQL `where` clause in the `SQL Query` field for the task as described for BIF in “Defining BIF Tasks” on page 82.

SQL queries for invoice generator tasks should include a clause specifying bill format. For example, the query **`CMF.bill_format_opt = 100 and CMF.account_no = BILL_INVOICE.account_no`** causes invoice generator to create only bills with bill format 100. Limiting an instance of invoice generator to a particular bill format speeds up processing because the instance only reads one layout file.

Depending on BIP, invoice generator, and payment module configuration, the invoice generator ignores certain bills until payment is complete. See "Processing Credit Card Payments (CPM)" on page 102 for more information.

If you are using hot billing, run invoice generator in Flash mode to support hot billing on demand from BIP (see "Bill Processing Overview" on page 66). Otherwise, schedule invoice generator in Production mode.

For general task definition instructions see "Configuring Processes" on page 30. To configure the invoice generator process you must click the button labelled IGEN in step 2.

Scheduling the Invoice Generator

You can run the invoice generator to format a set of bill records as soon as BIP calculates them. However, you can schedule time between BIP and the invoice generator to review BIP output.

BIP, invoice generator, and payment module configuration may necessitate scheduling other modules between BIP and the invoice generator. For example, if your bill format includes a credit card payment status you can run CPM between BIP and the invoice generator to make that status available.

For general scheduling instructions see "Configuring Processes" on page 30.

If you are running invoice generator in Flash mode you need not schedule it. Invoice generator in this mode runs in the background and responds to real-time requests from BIP.

Reviewing Invoice Generator Output

You can review invoice generator control reports and activity logs as described in "Reviewing Process Activity" on page 36.

You can review online images of formatted bills through several different interfaces as described in "Viewing Bills" on page 90. You also can view them directly if you wish as described for BIF in "Reviewing BIF Output" on page 83.

Troubleshooting the Invoice Generator

Check the control report for errors and review a representative sample of bills (see "Viewing Bills" on page 90) for incorrect totals, missing charges, incorrect messages or account information, and other problems:

- Incorrect charges (totals, charge lists, amounts, and so forth) usually result from problems with a recent BIP run. See "Troubleshooting BIP" on page 75.

- Incorrect formats (items in the wrong place, incorrect text, and so forth) usually result from invoice generator configuration errors. See the *Guide to Invoice Design* for more information.

Make sure all the bills you expect have in fact been created. If bills are missing:

- Bill records may not have been created for those bills due to BIP problems (see "Troubleshooting BIP" on page 75).
- The invoice generator may be holding those bills for payment (see "Processing Credit Card Payments (CPM)" on page 102).
- You may be using an incorrect SQL Query (see "Defining BIF Tasks" on page 82).

Dispatching Bills (BID)

The Bill Dispatcher (BID) selects available bills and dispatches them via printer, FTP, or other methods.

Note Not all bills are dispatched using BID.
In particular, if your deployment uses e-mail or HTML dispatching you must use local tools to deliver bills to customers.

Before You Run BID

Before running BID confirm that all output devices are operational. For example:

- Print dispatch — make sure printers are online and loaded with the correct paper in the correct orientation.
- FTP dispatch — make sure network connections to the remote host are functioning.

Be sure entries in the `BILL_FILES` table exist for the bills to be dispatched (see "Reviewing BIF Output" on page 83).

Defining BID Tasks

In MSA deployments run BID separately against each Customer server (see "Modules and Servers" on page 16).

Each BID task dispatches a subset of the bills formatted by BIF depending on the task name. Operations Center lists available tasks — for example, in the most common BID task, Print No-Hold prints unheld bills to printers or sends them to remote hosts.

Some task names include "Re-Run." Use these tasks to reprint bills that were successfully dispatched (for example, if a bill is damaged or lost in transit and the customer requests a new copy).

You can constrain the list of bills dispatched by entering an SQL `where` clause for the `BILL_ BATCHES` tables in the `SQL Query` field for the task. For example, use **`batch_id like "01211130040000"`** to dispatch only a single batch of bills (see “Reviewing BIF Output” on page 83). To use the `arch_flag` or `format_status` fields you must specify a table name as well.

For general task definition instructions see “Configuring Processes” on page 30.

Scheduling BID

Run BID with a Print No-Hold task any time after BIF has finished running. This dispatches all bills scheduled for dispatching. Allow some time to review BIF output before dispatching bills.

If bills in your deployment often are held (for example, while awaiting payment from a clearinghouse) you may have to run BID several times to dispatch all bills. You can run a Print Hold task to print held bills for review but be careful not to dispatch those bills to customers.

For general scheduling instructions see “Configuring Processes” on page 30.

Reviewing BID Output

You can review BID control reports and activity logs as described in “Reviewing Process Activity” on page 36.

If errors occur during dispatching files remain in the original `ready` directory (see “Reviewing BIF Output” on page 83). If no errors occur BID moves the files to `$ARBORDATA/disp/done`.

Check the dispatch device itself — printed bills appear at specified printers, FTP-dispatched bills appear at the remote host, and so forth.

Troubleshooting BID

If a file is not successfully dispatched check entries in `BILL_FILES` and `BILL_INVOICE` for the batch:

- If no entry for the file exists the problem lies with BIF’s output. See “Troubleshooting BIF” on page 85 for more information.
- If `BILL_FILES.dispatch_status` = 0 (available for dispatching) run a new BID task against that file and be sure it is actually defined to pick up files of that type (see “Defining BID Tasks” on page 88).
- If `dispatch_status` = -1 (error) confirm the dispatch devices and your connection to them and rerun BID.
- If `dispatch_status` = 2 (dispatched):
  - » The file was dispatched to an unexpected device.

    For example, the file is at a printer or remote host other than the one you expected. Check the file’s `disp_method` and the BID configuration to confirm the location.

    The simplest solution is to reprint the file (see “Defining BID Tasks” on page 88), but be sure the customer does not receive both bills.
  - » The file was dispatched at an unexpected time.

    Check `BILL_INVOICE.dispatch_date`.

If a file is dispatched but contains logical errors (missing or incorrect information) the problem does not lie with BID. Confirm the BIF and BIP output for that file.

If a file is dispatched but contains formatting errors that do not appear in the screen image (see "Viewing Bills" on page 90), the problem may lie with the dispatch mechanism itself (for example, the printer or related print drivers).

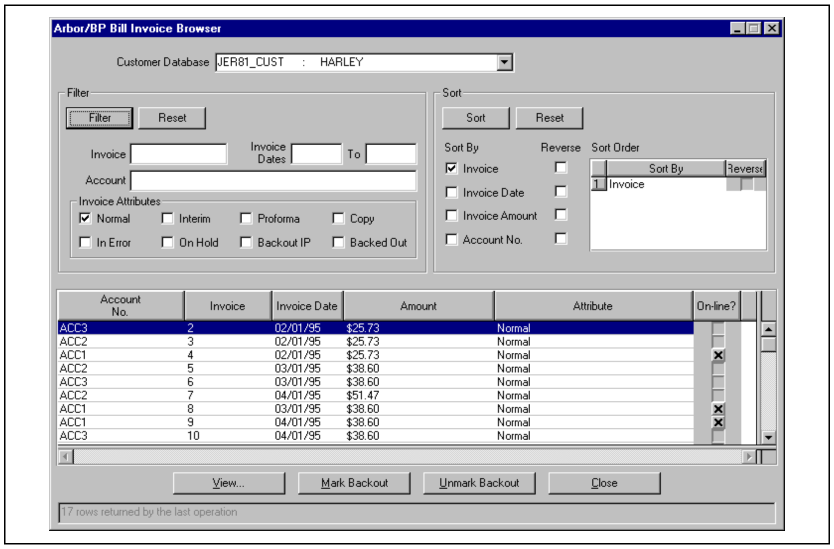
Viewing Bills

Use the `Bill Invoice Browser` interface (`bibexe`, frequently called BIB) to review all bills, including errored and backed-out bills, and to mark bills for backout if necessary.

To run BIB choose the **Bill Invoice Browser** menu item in the Kenan Customer Interfaces program group or click the corresponding icon. The BIB main window (see figure 17) appears.

To view a bill highlight it and click *View*. The `Invoice Review` window appears. This is identical to the `Bill Review` window in the Customer Care interface described in the Kenan/BP *User Guide*.

Figure 17 Bill Invoice Browser Main Window



Arbor/BP Bill Invoice Browser

Customer Database: JER81_CUST : HARLEY

Filter

Invoice:
Invoice Dates: To:
Account:

Invoice Attributes: [✓ Normal] Interim Proforma Copy In Error On Hold Backout IP Backed Out

Sort

Sort By: [✓ Invoice] Invoice Date Invoice Amount Account No.

Sort Order:

	Sort By	Reverse
1	Invoice	

Account No.	Invoice	Invoice Date	Amount	Attribute	On-line?
ACC3	2	02/01/95	$25.73	Normal	
ACC2	3	02/01/95	$25.73	Normal	
ACC1	4	02/01/95	$25.73	Normal	✗
ACC2	5	03/01/95	$38.60	Normal	
ACC3	6	03/01/95	$38.60	Normal	
ACC2	7	04/01/95	$51.47	Normal	
ACC1	8	03/01/95	$38.60	Normal	✗
ACC1	9	04/01/95	$38.60	Normal	✗
ACC3	10	04/01/95	$38.60	Normal	

17 rows returned by the last operation

Filtering and Sorting Listed Bills

To constrain the list of bills, use these three steps:

1. Enter one or more criteria in the fields provided.
   - » If an invoice number is specified all other criteria are ignored.
   - » The account number and dates are used in combination to select a particular bill if an invoice number is not specified.
2. Select one or more bill attributes from the check boxes provided. (Backout IP is an attribute indicating a bill backout is in progress.)
3. Click *Filter*.

To sort the list of bills, use these three steps:

1. Select one or more sort criteria using the *Sort By* check boxes.
2. Select the *Rev.* check box for each sort that should be applied in reverse order.
3. Click *Sort*.

Marking Bills for Backout

In the BIB main window (see figure 17) mark bills for backout by highlighting them and clicking *Mark for Backout*. The `Invoice Attribute` changes to "Backout IP" indicating that a backout is in progress.

The selected bill must be the most recent bill for the account that has not already been backed out (that is, it must have the latest Invoice Date value). Multiple invoices for one account can be selected and are backed out in sequence.

Only an original bill can be backed out. That is, bill copies, errored bills, and backed out bills cannot be backed out. Also, there can be no unbilled adjustments against the bill.

See "Backing Out Bills" on page 78 for more information.

To unmark a bill with backout in progress select the bill and click *Unmark Backout*. The bill cannot be unmarked once backout is complete.

Other Bill Review Tools

As mentioned in "Reviewing BIF Output" on page 83, ASCII image files for all bills appear in the `$ARBORDATA/disp/screen` directory. You can review these files directly using any program that displays ASCII text. These files approximate the dispatched output as closely as possible although the dispatched output may contain special characters, fonts, and other format effects not available in ASCII.

Use any HTML browser to view bills dispatched by HTML. Again see "Reviewing BIF Output" on page 83 for more information.

6 Payments Processing

This chapter discusses the batch payment modules that process payment transactions from external financial systems, the corresponding investigation units that investigate and correct errors in payments processing, and the real-time credit card server.

Payments Overview

After a charge appears on an invoice (calculated by BIP) the customer typically pays it. The payment enters the Kenan/BP database and reduces the customer's balance.

Sometimes customer service representatives enter those payments manually through the Customer Care interface and no further action is required. In most cases, however, payments enter Kenan/BP in batches through one of the following modules:

- The Lockbox (LBX) payments module processes payments files created by banks. These files represent checks paid by customers directly to the bank.

  See "Processing Lockbox Payments (LBX)" on page 95 for more information.
- The Credit Card Payments Module (CPM) generates transaction records for amounts due to be paid through credit card clearinghouses and processes results files returned by those clearinghouses.

  See "Processing Credit Card Payments (CPM)" on page 102 for more information.
- The Electronic Funds Transfer (EFT) module generates transaction records for amounts due to be paid through direct debit/EFT clearinghouses and processes results files returned by those clearinghouses.

  See "Electronic Funds Transfer (EFT)" on page 109 for more information.

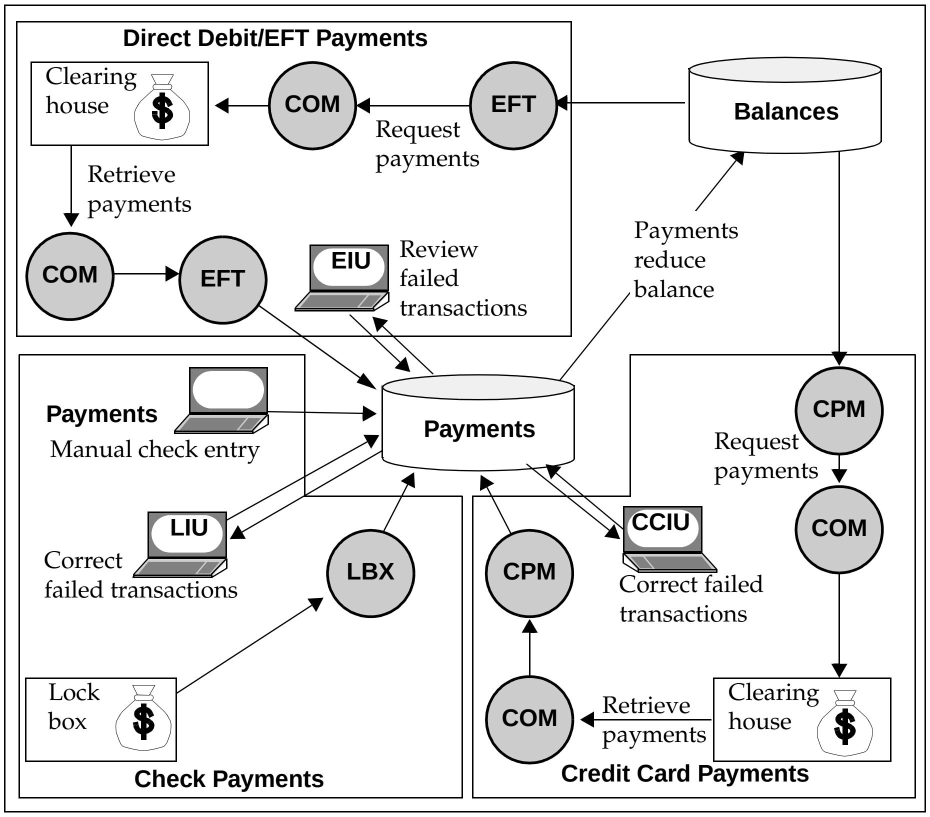
CPM and EFT depend on the COM module to exchange files with external systems. Figure 18 on page 95 illustrates the interactions between payments modules, banks, clearinghouses, and information stored in the Kenan/BP database.

Kenan/BP includes three separate interfaces for processing payment records that the batch modules cannot process:

- Use the LIU interface to correct failed LBX records (see "Lockbox Investigation with LIU" on page 100).
- Use the CCIU interface to correct failed CPM records (see "Credit Card Investigation with CCIU" on page 105).
- Use the EIU interface to correct failed EFT records (see "EFT Investigation with EIU" on page 112).

Your deployment does not necessarily support all three types of payments, in which case you need not schedule or run all of these modules. For example, if none of your customers pay by credit card you do not have to run CPM or CCIU.

Figure 18 Payments Processing



Kenan/BP also includes a centralized, high-performance Real-Time Credit Card (RTCC) server that allows custom applications to contact credit card clearinghouses to validate and authorize payments scheduled through a customer's credit card. See "Real-Time Credit Card Processing" on page 109 for more information.

Processing Lockbox Payments (LBX)

The Lockbox (LBX) payments module processes payments files created by banks. These files represent amounts paid by customers directly to the bank. If none of your customers pay by paying amounts to banks you do not have to run LBX.

LBX performs one of three tasks depending on the task mode in which you run it:

- **`RCV`** mode — retrieves files from the bank via modem using the CLEO communications package
- **`GET`** mode — retrieves files from a local directory
- **`PRO`** mode — processes retrieved files into the database

These task modes are described separately in the following sections.

Before You Run LBX

Note See the *Module Configuration Guide* for more information on configuring the items mentioned in this section.

Before you run LBX in RCV mode be sure that:

- All remote (type 1) LBX sources are properly configured.
- The modem is turned on and properly connected.
- CLEO is available to be run.

Before you run LBX in GET mode be sure that:

- All local (type 2) LBX sources are properly configured.
- Payments files exist in the appropriate directories for each source.

  For example, if the bank sends payment files via tape be sure all the files have been copied from the tape into the appropriate directories.
- Payments files have the proper naming conventions.

  Local filenames must be of the form `LBX`*`YYMMDDhhmm`* to be valid. Invalid files are left untouched.

Before you run LBX in PRO mode be sure that payment files have been retrieved successfully by LBX in GET or RCV modes.

Defining LBX Tasks

In MSA deployments run LBX against the Admin server (see “Modules and Servers” on page 16). Select either **`RCV`**, **`GET`**, or **`PRO`** mode depending on the task you are defining. LBX retrieves or processes all valid payment files depending on the task mode.

For general task definition instructions see “Configuring Processes” on page 30.

Scheduling LBX

You must run LBX twice, performing two separate tasks:

1. retrieve payment files (RCV or GET mode)

   Usually each deployment uses RCV or GET but not both.
2. process payment files (PRO mode)

If none of your customers pay by writing checks to banks you do not have to run LBX at all.

For RCV tasks schedule LBX to begin just before the expected transmission time. More specifically:

1. Determine the time(s) when the bank calls your deployment and initiates the file transfer. This generally occurs at the same time every day.
2. Determine the value of the `RCV_WAIT_TIME` (LBX) system parameter.

   This controls how long LBX waits before disconnecting.
3. Schedule LBX (RCV) to begin `RCV_WAIT_TIME` or fewer seconds earlier than the expected transmission time.

For example, if the bank initiates file transfers at 7 P.M. every day and `RCV_WAIT_TIME` is set to 60 schedule LBX (RCV) to begin at some time between 6:59:00 and 6:59:59, such as 6:59:30.

You cannot run more than one LBX (RCV) task at a time since CLEO and the modem are locked when in use by LBX.

For GET tasks schedule LBX to run any time after payments files are placed in the directory defined in the `LBX_SOURCE_ID_REF` table. If you retrieve these files using COM you must rename them into the form `LBX`*`YYMMDDhhmm`*, (manually or by running a shell script or other automated tool) before the GET task runs.

For PRO tasks schedule LBX to run any time after the GET or RCV task ends but before BIP (to ensure that received payments appear on the next customer bill).

Reviewing LBX Output

You can review LBX control reports and activity logs as described in “Reviewing Process Activity” on page 36.

In GET and RCV modes LBX creates an entry in `LBX_FILE_STATUS` for every file it receives and moves these files into the `$ARBORDATA/lockbox/ready` directory. LBX also renames the file to `LBX`*datetime*`.`*extension* where:

- *datetime* is the current date and time in RCV mode and the initial date and time in GET mode. The format is *YYMMDDhhmm* in both cases.
- *extension* is `LBX_SOURCE_ID_REF.naming_extension` for the lockbox source.

In PRO mode LBX creates a `BMF` record for each payment it receives and then moves valid files into the `$ARBORDATA/lockbox/done` directory and errored files into the `$ARBORDATA/lockbox/error` directory.

LBX processes payments in batches in PRO mode. Instead of processing each payment individually, LBX reads a batch of `PAYMENT_BATCH_SIZE` transactions from the response file and processes them all in a single batch. Ideal batch size varies by deployment, but one-fifth of the typical payment file is a good starting value. If a batch contains any invalid payment transactions, LBX inserts each record in the batch individually to find the invalid payment, so do not set `PAYMENT_BATCH_SIZE` too large.

Troubleshooting LBX

Errored payment records are stored in `LBX_ERROR` where they can be investigated by the LBX Investigation Unit (see “Lockbox Investigation with LIU” on page 100) and are indicated in the log file and control report.

Files Not Received

Lockbox files with invalid file names are not received by GET or RCV tasks and do not have `LBX_FILE_STATUS` entries. However, the file names are included in the log file.

Files Not Processed

Lockbox files that are received but cannot be processed into records are moved into the `lockbox/error` directory. In `LBX_FILE_STATUS` their `lbx_file_status` is set to -1 and `error_code` is set to a file-level error code value indicating the nature of the problem. See table 14 on page 98 for a complete list.

Contact the lockbox bank and request that the file be retransmitted.

Incompletely processed files are left in `$ARBORDATA/lockbox/ready`. You do not need to do anything with these files; LBX reprocesses them automatically when it next runs a PRO task.

Manual Intervention

Unusual cases such as errors caused by a power failure or by certain classes of exceptional database states require manual intervention before LBX reprocesses the file:

- Using a command-line SQL tool (such as **`isql`** or **`sqlplus`**) set `LBX_FILE_STATUS.lbx_file_status` for the file to 0.
- Move the file from the `done` or `error` directory back to the `ready` directory if necessary.

You must stop all LBX processes before you make any manual changes to the `LBX_FILE_STATUS` table.

Unapplicable Payments

When `CREATE_BMF_UNAPPLIED` (BIP) is set to 1, if LBX tries to process a refund or reversal in PRO mode and cannot match it with any charges because it exceeds the account's credit balance, LBX inserts a row in `BMF_UNAPPLIED`. If `CREATE_BMF_UNAPPLIED is` set to 0, the record is sent to LIU instead.

Check `BMF_UNAPPLIED` for new records with `applied_status` = 0. If the payment applies, create a payment for the appropriate amount using the Payments interface and set `applied_status` = 1. If the payment does not apply, set `applied_status` = 2.

Records in Error

Errored lockbox records (for example, records that refer to an invoice or account that does not exist) are stored in `LBX_ERROR` with an error code indicating the nature of the error as shown in table 14. Use LIU to fix record-level errors.

Table 14 Lockbox Error Codes

Code	Error Name	Description
1001	LBX_OPEN	Could not open the lockbox file for read.
1002	LBX_BAD_REC	Bad length or format in lockbox record.
1003	LBX_INV_REC	Invalid data field in lockbox record.
1004	LBX_SQL_ERROR	Failure executing SQL command.
1005	LBX_BAT_TRAIL	Batch trailer record did not exist.
1006	LBX_LBX_TRAIL	Lockbox trailer record did not exist.
1007	LBX_FILE_TRAIL	File trailer record did not exist.
1008	LBX_FILE_HEADER	File header record did not exist.

Table 14 Lockbox Error Codes (Continued)

Code	Error Name	Description
1009	LBX_SOURCE_ID	Invalid source_id.
1010	LBX_TRANS_FORM	Invalid file transfer date/time (form).
1011	LBX_TRANS_FTR	Invalid future transfer date/time.
1012	LBX_TRANS_CYCLE	Invalid transfer date/time (cycle).
1013	LBX_DUP_FILE	Duplicate file (duplicate header info).
1014	LBX_MAX_FILE	Maximum number of files per cycle reached.
1015	LBX_MV_FILE	Unable to move/rename file. This is usually a directory or file permissions problem.
1016	LBX_WIRE	Wire transfer not implemented.
1017	LBX_AMOUNT	Payment amount greater than check amount.
2001	LBX_BILL_REF_NO	`bill_ref_no` invalid or did not match account.
2002	LBX_ACCOUNT_ID	`account_no` invalid.
2003	LBX_POSTING	Unable to post payment. This is a general error message normally due to a conflict when applying the specific payment/reversal type to the account.
2004	LBX_ACCOUNT_ CLOSED	Account is closed or disconnected.
2005	LBX_CMF_ACCOUNT	Account was not found in Customer database.
2006	LBX_DUP_BILL_REF_NO	More than one bill with the same invoice tracking number (`bill_ref_no`) is associated with the account. This can occur because Kenan uses `bill_ref_no` and `bill_ref_resets` to uniquely identify an invoice, but only the `bill_ref_no` is passed in the LBX detail record.
2007	LBX_CURRENCY_CODE	LBX bank currency code does not match account's currency code.
2008	LBX_DUP_ACCOUNT_ID	The external ID provided maps to more than one `account_no`.
2009	LBX_TRANS_TYPE	Invalid payment type.
2010	LBX_ZERO_BILL_REF	`bill_ref_no` of 0 given and is not allowed.
2011	LBX_POST_DATED	This is a post-dated transaction (this code is used to guide post-dated transactions into the `LBX_ POST_DATED` table rather than as an error code).
2012	LBX_OPEN_ITEM_ID	Invalid open_item_id in record.
2014	LBX_CURRENCY_BASE	Base currencies of the account currency and external currency differ.
2015	LBX_CURRENCY_ CONVERSION	Could not perform the currency conversion.
2016	LBX_NO_ORIG_BMF	Could not find original BMF payment for this reversal.

Lockbox Investigation with LIU

Whenever LBX cannot match a check payment from an electronic bank statement to a Kenan/BP account or invoice the payment is set aside for investigation and correction by the LBX Investigation Unit (LIU). LIU allows administrators to make online corrections to erroneous payment records by correcting either the Kenan/BP account number or the invoice number (`bill_ref_no`).

Accessing the Lockbox Investigation Unit

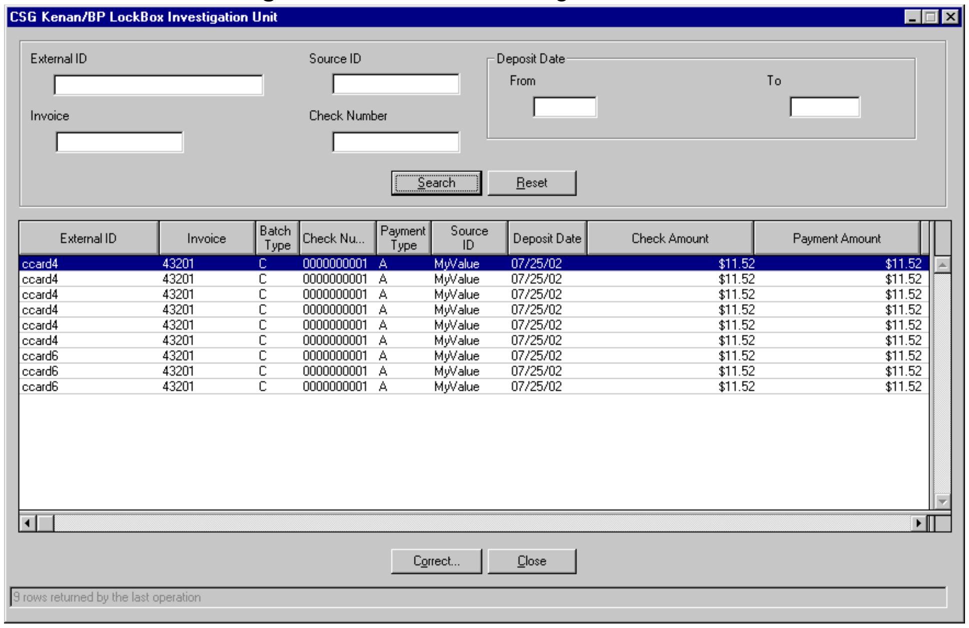
Access LIU through Arbor folder under **Programs** in the Windows NT Start menu. In the Arbor folder select **Lockbox Investigation Unit**. Only members of the `arboradm` or `arborsupercsr` user groups can use LIU.

Filtering Payments

The `LockBox Investigation` window shown in figure 19 displays errored payment records.

You must define filters before LIU displays any records. To define filters enter values in the filter fields and click *Filter*. Only those records that match the values you enter are displayed. You can leave all fields blank to display all records, but you must still click *Filter* to display them.

Figure 19 LockBox Investigation Window



External ID	Invoice	Batch Type	Check Nu...	Payment Type	Source ID	Deposit Date	Check Amount	Payment Amount
ccard4	43201	C	0000000001	A	MyValue	07/25/02	$11.52	$11.52
ccard4	43201	C	0000000001	A	MyValue	07/25/02	$11.52	$11.52
ccard4	43201	C	0000000001	A	MyValue	07/25/02	$11.52	$11.52
ccard4	43201	C	0000000001	A	MyValue	07/25/02	$11.52	$11.52
ccard4	43201	C	0000000001	A	MyValue	07/25/02	$11.52	$11.52
ccard4	43201	C	0000000001	A	MyValue	07/25/02	$11.52	$11.52
ccard6	43201	C	0000000001	A	MyValue	07/25/02	$11.52	$11.52
ccard6	43201	C	0000000001	A	MyValue	07/25/02	$11.52	$11.52
ccard6	43201	C	0000000001	A	MyValue	07/25/02	$11.52	$11.52

9 rows returned by the last operation

Sorting Payments

Use the *field* check boxes in the *sort* region of the LIU window to sort displayed records by selected fields such as account number, invoice number, batch type, and so on. To sort a particular field in descending rather than ascending order check its *Reverse* check box. When all fields are properly set click *Sort*.

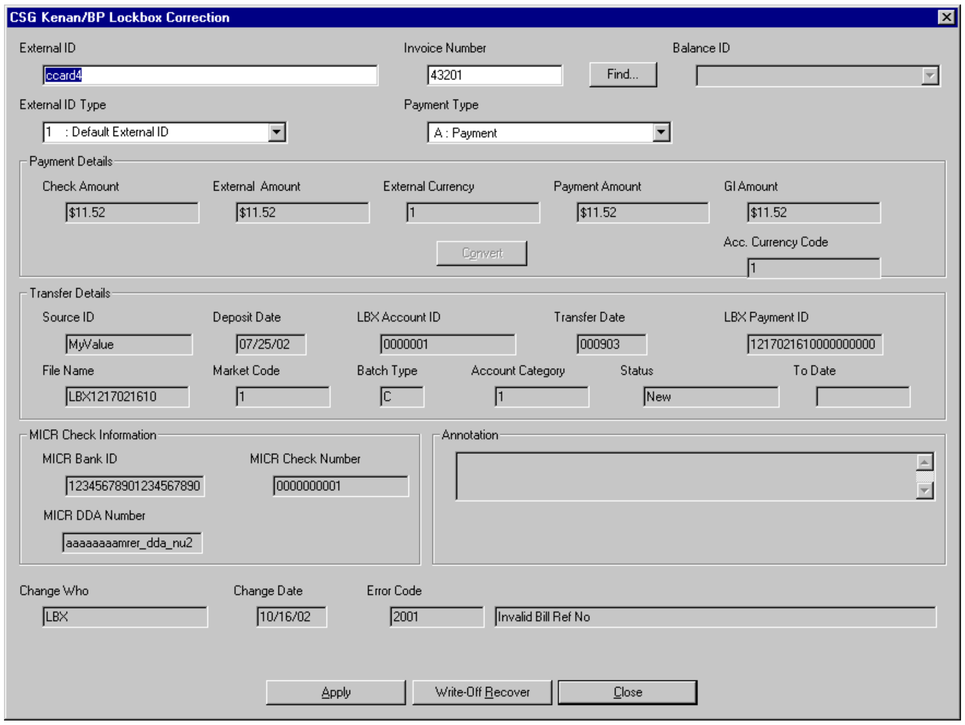
Note In current implementations the *Help* button on the LIU window is inactive. No online help is currently available for LIU functions.

Reviewing and Correcting a Payment

To review all data associated with an individual payment and to make corrections select the record in the bottom portion of the `Lockbox Investigation` window and click *Correct...*.

The `Lockbox Correction` window appears as shown in figure 20.

Figure 20 Lockbox Correction Window



CSG Kenan/BP Lockbox Correction

External ID	ccard4	
Invoice Number	43201	
Balance ID		
External ID Type	1 : Default External ID	
Payment Type	A : Payment	

Payment Details

Check Amount	$11.52
External Amount	$11.52
External Currency	1
Payment Amount	$11.52
Gl Amount	$11.52
Acc. Currency Code	1

Transfer Details

Source ID	MyValue
Deposit Date	07/25/02
LBX Account ID	0000001
Transfer Date	000903
LBX Payment ID	1217021610000000000
File Name	LBX1217021610
Market Code	1
Batch Type	C
Account Category	1
Status	New
To Date	

MICR Check Information

MICR Bank ID	12345678901234567890
MICR Check Number	0000000001
MICR DDA Number	aaaaaaaamrer_dda_nu2

Annotation

Change Who	LBX
Change Date	10/16/02
Error Code	2001 Invalid Bill Ref No

The `External ID, External ID Type, Payment Type, Open Item ID`, and `Invoice` fields can be changed and updated by entering new values in these fields and pressing *Apply*.

The *Convert* button becomes active when LBX is in multicurrency mode (specified in the version_code field in the payment file, see *Reports and File Layouts*). Use the *Convert* button to perform a conversion of the `External Amount` into the account currency.

When you successfully *Apply* a correction the payment record is no longer available for investigation by LIU. Otherwise the record remains in LIU until it has been assigned successfully to a Kenan/BP account and bill reference number or until a currency conversion is performed.

If the transaction is a payment to a written-off account press the *Write-Off Recover* button to apply the payment to the account's write-off amount. To have LBX automatically apply full and partial payments (but not overpayments) to a written-off disconnected account, the account must be in DISC_DONE state and the `AUTO_RECOVERY` system parameter must be set to 1.

If the payment fails because it exceeds the account's credit balance and the `CREATE_BMF_UNAPPLIED` (BIP) system parameter is 1, the payment is written to `BMF_UNAPPLIED` and must be applied manually. If `CREATE_BMF_UNAPPLIED` is set to 0, the payment is returned to LIU with the error “Could not distribute debit amount.”

To close the window without applying any changes click *Close*.

Processing Credit Card Payments (CPM)

The Credit Card Payments (CPM) module exchanges payments files with credit card clearinghouses, thus charging the amount owed by a given customer directly to his or her credit card account. Running CPM is unnecessary if no customers pay by credit card.

Before You Run CPM

Before you run CPM be sure all external contacts, clearinghouses, response codes, credit card types, and related items are configured properly. See the *Module Configuration Guide* for more information on configuring these items.

Before you run CPM in an MSA deployment run RepStat against the Admin server.

Before you run CPM in Create mode be sure BIP has updated the account balance for the accounts to be charged.

Before you run CPM in Process mode be sure response files are available in appropriate directories for each clearinghouse external contact.

Before you run CPM in Post mode for a clearinghouse be sure the clearinghouse is configured to support autoposting payments.

Defining CPM Tasks

In MSA deployments run CPM against each Customer server (see “Modules and Servers” on page 16). Select **`Create`**, **`Process`**, or **`Post`** mode, depending on the task you are defining:

- **Create** mode — generates transaction files to be sent to clearinghouses

  CPM includes all appropriate pending transactions from the `CCARD_TRANS` table. You do not have to specify which accounts are processed.
- **Process** mode — processes response files returned from clearinghouses and credits approved payments to customer accounts

  Response files indicate whether transactions were accepted, declined by the clearinghouse, or rejected by the customer. CPM can also handle payment reversal files from the clearinghouse.
- **Post** mode — automatically credits payments to customer accounts without having received a response from the clearinghouse

In general use Create mode to create transaction files and Process mode to process response files (see “Scheduling CPM” on page 103).

For general task definition instructions see “Configuring Processes” on page 30.

Scheduling CPM

Running CPM is unnecessary if no customers pay by credit card.

Otherwise you must schedule several CPM and COM instances to process credit card payments using the following five steps:

1. Run CPM in Create mode to generate transaction files.
2. Run COM to send the transaction files to the credit card clearinghouse.
3. Wait for response files to be available. The time required varies and depends on your clearinghouse.
4. Run COM to receive response files from the clearinghouse.
5. Run CPM in Process mode to process the response files.

If BIP has been configured to hold customers' invoices until payments are approved — that is, if the `PAYMENT_DATE_DELAY` (BIP) system parameter is set to 0 — you must also schedule BIF/Invoice Generator to format the invoice after CPM in Process mode runs.

In Process mode CPM processes payments in batches. Instead of processing payments individually, CPM reads multiple `PAYMENT_BATCH_SIZE` transactions from the response file and processes them together. Ideal batch size varies by deployment, but one-fifth of the typical payment file is a good starting value. If a batch contains any invalid payment transactions, CPM inserts each record in the batch individually to find the invalid payment, so do not set `PAYMENT_BATCH_SIZE` too large.

CPM includes only appropriate pending transactions in transaction files and ensures that transactions are never duplicated. Therefore, you can schedule steps 1 and 2 to run as often as you wish — for instance, you can run them daily to send a day's scheduled transactions to the clearinghouse. For example, if a clearinghouse is configured to delay sending transactions until 10 days after the bill is dispatched CPM in Create mode does not send transactions before that date no matter when you run it. Pending transactions simply stay in the `CCARD_TRANS` table until CPM deals with them.

Similarly, you can repeat steps 4 and 5 as often as you wish to process late-arriving response files. For example, if your clearinghouse's turnaround time varies significantly you can run steps 4 and 5 every day to collect and process files received that day. As another example, CPM in Process mode can process payment reversal files (also known as *return files* and *chargeback files*) indicating that a customer rejected a previously approved transaction. Since this can occur at any time you can run steps 4 and 5 every day to process payment reversal files as soon as they arrive.

Posting Payments

If a clearinghouse is configured to support auto-posting payments you can run CPM in Post mode to credit payments automatically to customer accounts without having received a response from that clearinghouse.

For instance, if a clearinghouse sends response files only for rejected records run steps 1 through 5 above to send transactions and process rejections and then:

6. Run CPM in Post mode to approve "leftover" transactions.

Note If CPM later attempts to process approvals or rejections for an auto-posted transaction the attempt fails and CPM reports an error.

Each clearinghouse that supports autoposting payments has a preconfigured autopost delay, and CPM (Post) only approves transactions that COM sent to the clearinghouse that many days earlier. For example, if a clearinghouse has an autopost delay of 5 days CPM (Post) running on 5/10/04 only approves payments sent to the clearinghouse on or before 5/5/04.

If you do not expect any response files from your clearinghouse running CPM in Process mode is unnecessary. Instead you can process payments using the following four steps:

1. Run CPM in Create mode to generate transaction files.
2. Run COM to send the transaction files to the credit card clearinghouse.
3. Wait for the auto-post delay period.
4. Run CPM in Post mode to approve sent transactions.

Reviewing CPM Output

You can review CPM control reports and activity logs as described in “Reviewing Process Activity” on page 36.

During the Process and Post tasks CPM updates `CMF_BALANCE`, `CMF_BALANCE_DETAIL` and `BMF` to reflect payments.

CPM also updates status fields in various tables:

- `CCARD_TRANS.cc_trans_status` — Transaction status
- `FILE_STATUS.file_status` — File processing status
- `CPM_CYCLE_STATUS.cycle_status` — CPM cycle status

  During the Create task, CPM creates a new cycle ID for the transaction file. The clearinghouse places the same cycle ID in corresponding return files. When CPM has processed a response for every transaction sent that cycle is complete.
- `BILL_INVOICE.prep_status` — Invoice hold status

  If an invoice has been held pending payment CPM removes that hold when the payment is approved. Subsequently BIF/IGEN can format the bill.
- `CCARD_LIST.avs_response_code` — Clearinghouse address verification status (only if AVS is being used)

Troubleshooting CPM

The clearinghouse attempts to authorize and deposit each transaction it receives and then returns one or more acknowledgment files containing a status record for each transaction sent. Errored transactions that are returned to CPM can be investigated and corrected using the Kenan/BP Credit Card Investigation Unit (CCIU). You can then rerun CPM to reprocess those transactions.

To verify the status of a transaction check the value of the `CCARD_TRANS.cc_trans_status` field. CPM sets this field to one of the following values for each transaction:

- 8 (completed)
- 6 (canceled)

- 5 (reversed)
- 4 (completed with errors)
- 2 (pending)
- 1 (hold for credit)

Also check the CPM control report for payments that were rejected because of credit card holds, invalid or expired credit cards, or errors in the payment record.

Check `BMF_UNAPPLIED` for new records with `applied_status` = 0. These are payments that were not processed because CPM could not match them with any charges. If the payment applies, create a payment for the appropriate amount using the Payments interface and set `applied_status` = 1. If the payment does not apply, set `applied_status` = 2.

If a customer's credit card is rejected you can change the payment method and reset the `BILL_INVOICE.bill_hold_code` and `prep_status` to 0, thereby allowing the invoice to be formatted (unnecessary if the PAYMENT_DATE_DELAY (BIP) system parameter is set to 1).

If CPM fails midway through a processing task, it picks up from the first unprocessed record when the file is reprocessed. If a record was partially processed (for example, a Create task that was interrupted after records were assigned to transaction files and `cc_trans_status` set to 2 (pending) but before the final payment send file was created), CPM does not complete it. Instead, before running CPM again run the script `cleanup_cpm.sh` from the command line to reset partially processed records to unprocessed. The script prompts for the following information:

- Customer database containing the errored files
- user
- password
- task mode that was interrupted (Create, Post, Process)

For more information, see the *Technical Reference*.

Credit Card Investigation with CCIU

If a credit card transaction does not clear — that is, if CPM cannot credit the account as paid — you must investigate and correct the transaction through the Credit Card Investigation interface (CCIU).

Note If BIP is configured to hold bills until payment is approved you must clear these transactions before the bill is formatted and dispatched.

Access CCIU through the Arbor folder under **Programs** in the Windows NT Start menu. In the Arbor folder, select **Credit Card Investigation Unit**. Only users in the `arboradm` user group can execute CCIU.

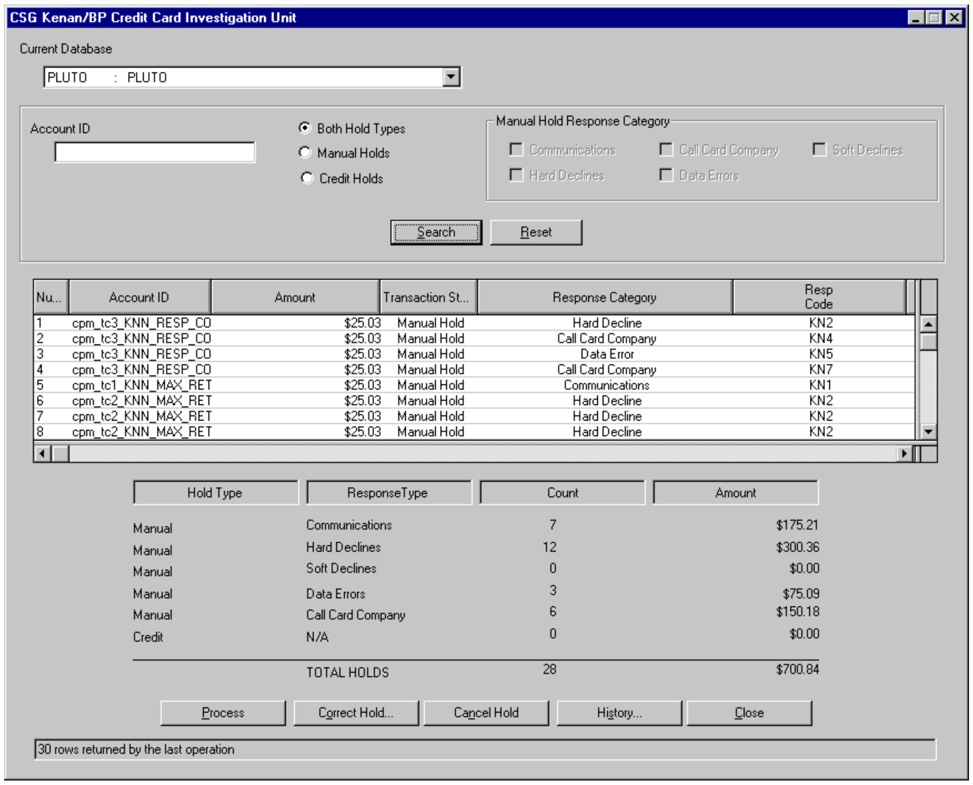
Select the transaction types to review and click *Filter* to list them in the `Credit Card Investigation` window (figure 21). CCIU can display the following types of transactions:

- **Credit holds** — amounts owed to a customer which an authorized user must verify before CPM can credit the amount to the customer account

- **Errors** — transactions CPM cannot process, categorized as *Communications*, *Hard Declines*, *Soft Declines*, *Data Errors*, or *Call Card Company* errors

Note The *Help* button on all CCIU windows is inactive. No online help is available for CCIU.

Figure 21 Credit Card Investigation



Current Database: PLUTO : PLUTO

Account ID:

(● Both Hold Types) Manual Holds / Credit Holds

Manual Hold Response Category: ☐ Communications ☐ Call Card Company ☐ Soft Declines ☐ Hard Declines ☐ Data Errors

Nu...	Account ID	Amount	Transaction St...	Response Category	Resp Code
1	cpm_tc3_KNN_RESP_CO	$25.03	Manual Hold	Hard Decline	KN2
2	cpm_tc3_KNN_RESP_CO	$25.03	Manual Hold	Call Card Company	KN4
3	cpm_tc3_KNN_RESP_CO	$25.03	Manual Hold	Data Error	KN5
4	cpm_tc3_KNN_RESP_CO	$25.03	Manual Hold	Call Card Company	KN7
5	cpm_tc1_KNN_MAX_RET	$25.03	Manual Hold	Communications	KN1
6	cpm_tc2_KNN_MAX_RET	$25.03	Manual Hold	Hard Decline	KN2
7	cpm_tc2_KNN_MAX_RET	$25.03	Manual Hold	Hard Decline	KN2
8	cpm_tc2_KNN_MAX_RET	$25.03	Manual Hold	Hard Decline	KN2

Hold Type	ResponseType	Count	Amount
Manual	Communications	7	$175.21
Manual	Hard Declines	12	$300.36
Manual	Soft Declines	0	$0.00
Manual	Data Errors	3	$75.09
Manual	Call Card Company	6	$150.18
Credit	N/A	0	$0.00
	TOTAL HOLDS	28	$700.84

30 rows returned by the last operation

Response Codes and Transaction Types

Figure 22 shows what a held transaction looks like in the CCIU interface. Note the `Resp Code` and `Trans Type` fields in the held transaction list. These fields are normally populated by values from the following tables.

Table 15 Manual Credit Card Hold Response Codes

Code	Description
0008	Detail Accepted for Deposit.
0010	Invalid Cardholder Account Number.
0011	Invalid Transaction Code.
0012	Invalid Transaction Amount.
0013	Invalid Transaction Date.
0014	Invalid Authorization Date.
0015	Invalid Card Expiration Date.
0020	Approved and Deposit Accepted.
0021	Approved but Deposit Failed.
0024	Referral.

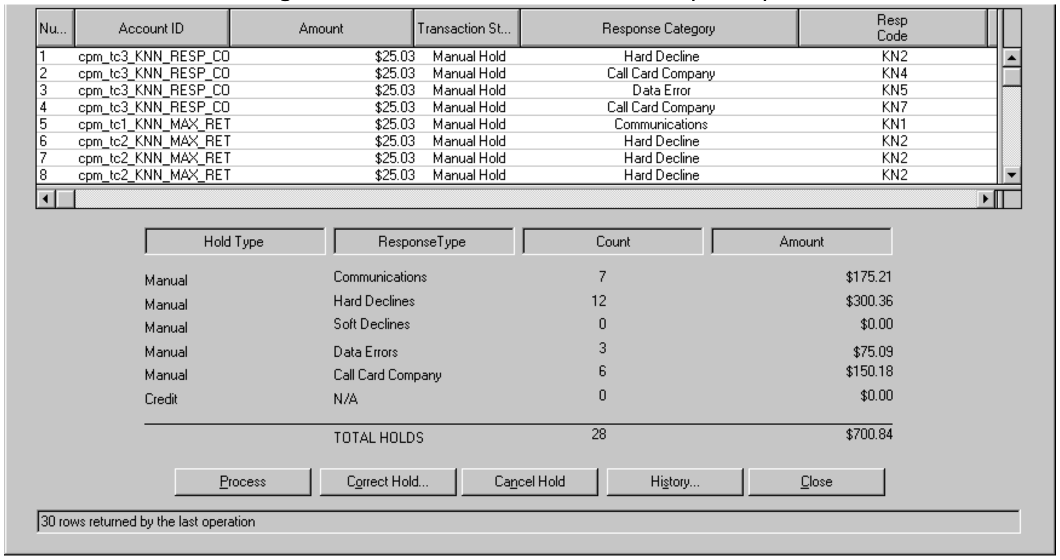
Table 16 Manual Credit Card Hold Transaction Types

Code	Description
8	Authorize/Deposit.
A	Authorize Only.
6	Credit Transaction.
5	Forced Deposit.

Resolving Holds

The field shown in figure 22 appears in the center of the CCIU interface. It displays all credit holds individually. For each, the choices are *Process* to resend, *Correct Hold* to edit the hold, *Cancel Hold* to cancel the transaction, and *History* to view the transaction history. Click the transaction to select it; then click a command button.

Figure 22 Credit Hold Review Field (Detail)



Nu...	Account ID	Amount	Transaction St...	Response Category	Resp Code
1	cpm_tc3_KNN_RESP_CO	$25.03	Manual Hold	Hard Decline	KN2
2	cpm_tc3_KNN_RESP_CO	$25.03	Manual Hold	Call Card Company	KN4
3	cpm_tc3_KNN_RESP_CO	$25.03	Manual Hold	Data Error	KN5
4	cpm_tc3_KNN_RESP_CO	$25.03	Manual Hold	Call Card Company	KN7
5	cpm_tc1_KNN_MAX_RET	$25.03	Manual Hold	Communications	KN1
6	cpm_tc2_KNN_MAX_RET	$25.03	Manual Hold	Hard Decline	KN2
7	cpm_tc2_KNN_MAX_RET	$25.03	Manual Hold	Hard Decline	KN2
8	cpm_tc2_KNN_MAX_RET	$25.03	Manual Hold	Hard Decline	KN2

Hold Type	ResponseType	Count	Amount
Manual	Communications	7	$175.21
Manual	Hard Declines	12	$300.36
Manual	Soft Declines	0	$0.00
Manual	Data Errors	3	$75.09
Manual	Call Card Company	6	$150.18
Credit	N/A	0	$0.00
	TOTAL HOLDS	28	$700.84

30 rows returned by the last operation

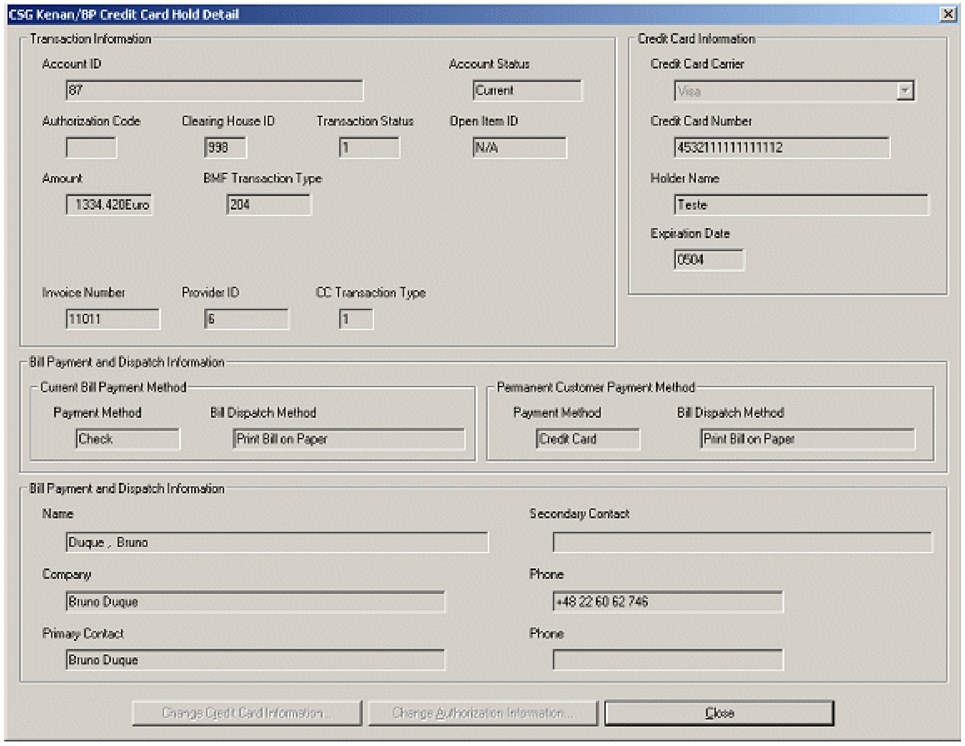
Processing Holds

To reprocess a hold select the desired transaction and press the *Process* button at the bottom of the CCIU interface. The transaction is reprocessed and removed from the holds list.

Correcting Holds

When you select a transaction and press the *Correct Hold* button the `Credit Card Hold Detail` window appears, as in figure 23.

Figure 23 Credit Card Hold Detail Window



CSG Kenan/BP Credit Card Hold Detail

Transaction Information

Account ID	87
Account Status	Current
Authorization Code	
Clearing House ID	998
Transaction Status	1
Open Item ID	N/A
Amount	1334.420Euro
BMF Transaction Type	204
Invoice Number	11011
Provider ID	6
CC Transaction Type	1

Credit Card Information

Credit Card Carrier	Visa
Credit Card Number	4532111111111112
Holder Name	Teste
Expiration Date	0504

Bill Payment and Dispatch Information

	Payment Method	Bill Dispatch Method
Current Bill Payment Method	Check	Print Bill on Paper
Permanent Customer Payment Method	Credit Card	Print Bill on Paper

Bill Payment and Dispatch Information

Name	Duque , Bruno
Secondary Contact	
Company	Bruno Duque
Phone	+48 22 60 62 746
Primary Contact	Bruno Duque
Phone	

Change Credit Card Information...
Change Authorization Information...
Close

In this window you can change credit card information on the transaction and the authorization code and clearinghouse ID. Once you have changed information press the *Change Credit Card Info* or *Change Authorization Code* button to update the transaction.

Note You need to telephone the credit card company to obtain the authorization code.

Canceling Holds

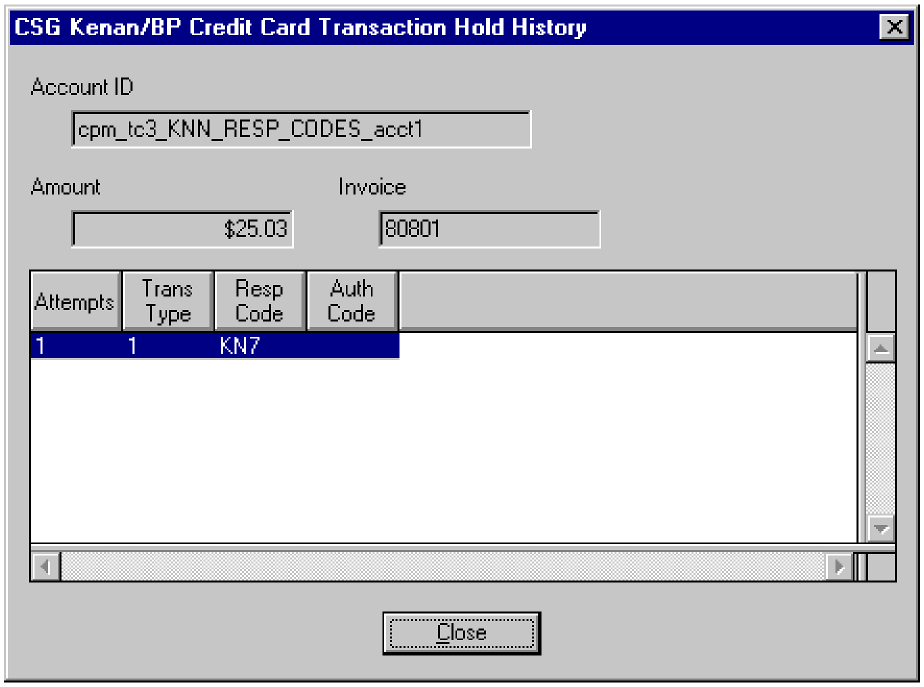
To cancel a hold select a transaction and press the *Cancel Hold* button. CCIU changes the payment method from credit card to check.

Viewing Transaction Histories

When you select a transaction and press the *History* button on the CCIU interface the `Hold History` window appears (as shown in figure 24).

This window can help you determine how to process a transaction that, for example, has failed several times.

Figure 24 Review Manual Hold History



CSG Kenan/BP Credit Card Transaction Hold History

Account ID: cpm_tc3_KNN_RESP_CODES_acct1

Amount: $25.03

Invoice: 80801

Attempts	Trans Type	Resp Code	Auth Code
1	1	KN7	

Real-Time Credit Card Processing

The Real-Time Credit Card (RTCC) server allows custom applications to contact credit card clearinghouses to validate and authorize payments scheduled through a customer's credit card. RTCC ensures that the card is valid and that the funds are available. It does not actually transfer funds.

Because RTCC is a server you need run it only once by entering the following command at the UNIX command line:

```
rtcc_serv >& logfile
```

RTCC writes status messages to the specified `logfile`. Unless you provide a log file name RTCC writes status messages to standard output.

To stop the RTCC server you must explicitly **`kill`** it. See your UNIX documentation for more information.

You may also have to configure, schedule, and troubleshoot custom applications developed by your deployment that use RTCC. See your site-specific documentation for more information.

Electronic Funds Transfer (EFT)

The Electronic Funds Transfer (EFT) module exchanges payments files with direct debit clearinghouses, charging the amount owed by a given customer directly to his or her direct debit account. Running EFT is unnecessary if none of your customers pay by direct debit.

Before running EFT be sure all configured and dynamic items are up-to-date as described for CPM in “Before You Run CPM” on page 102.

Defining EFT Tasks

In MSA deployments run EFT against each Customer server (see “Modules and Servers” on page 16). Select **`Create`**, **`Process`**, or **`Post`** mode depending on the task you are defining:

- **Create** mode — generates transaction files to be sent to clearinghouses

  EFT includes all appropriate pending transactions from the `EFT_TRANS` table. You do not have to specify which accounts are processed.
- **Process** mode — processes response files returned from clearinghouses and credits approved payments to customer accounts

  Response files indicate whether transactions were accepted, declined by the clearinghouse, or rejected by the customer. EFT can also handle payment reversal files from the clearinghouse.
- **Post** mode — automatically credits payments to customer accounts without having received a response from the clearinghouse

In most cases use Create mode to create transaction files and later use Process mode to process response files (see “Scheduling CPM” on page 103).

For general task definition instructions see “Configuring Processes” on page 30.

Scheduling EFT

Note Running EFT is unnecessary if none of your customers pay by direct debit .

Otherwise you must schedule several EFT and COM instances to process direct debit payments as described in the following five steps:

1. Run EFT in Create mode to generate transaction files.
2. Run COM to send the transaction files to the direct debit clearinghouse.
3. Wait for response files to be available. The time required varies and depends on your clearinghouse.
4. Run COM to receive response files from the clearinghouse.
5. Run EFT in Process mode to process the response files.

See “Scheduling CPM” on page 103 for additional CPM scheduling concerns that also apply to EFT.

Reviewing EFT Output

You can review EFT control reports and activity logs as described in “Reviewing Process Activity” on page 36.

During the Process and Post tasks EFT updates `CMF_BALANCE`, `CMF_BALANCE_DETAIL` and `BMF` to reflect payments.

In Process mode EFT processes payments in batches. Instead of processing payments individually, EFT reads multiple `PAYMENT_BATCH_SIZE` transactions from the response file and processes them together. Ideal batch size varies by deployment, but one-fifth of the typical payment file is a good starting value. If a batch contains any invalid payment transactions, EFT inserts each record in the batch individually to find the invalid payment, so do not set `PAYMENT_BATCH_SIZE` too large.

EFT also updates `EFT_TRANS.eft_trans_status`, `FILE_STATUS.file_status`, and `EFT_CYCLE_STATUS.cycle_status`. These fields are analogous to the `CCARD_TRANS.cc_trans_status`, `FILE_STATUS.file_status`, and `CPM_CYCLE_STATUS.cycle_status` fields described in "Reviewing CPM Output" on page 104.

Troubleshooting EFT

The clearinghouse attempts to authorize and deposit each transaction it receives and then returns one or more acknowledgment files containing a status record for each transaction sent. Errored transactions that are returned to EFT can be investigated and corrected using the Electronic Funds Transfer Investigation Unit (EIU). You can then rerun EFT to reprocess those transactions.

To verify the status of a transaction check the value of the `EFT_TRANS.eft_trans_status` field. EFT sets this field to one of the following values for each transaction:

- 8 (completed)
- 6 (canceled)
- 5 (reversed)
- 4 (rejected)
- 2 (pending)
- 1 (hold for credit)

In addition note that exception files (files containing EFT transactions rejected by the clearinghouse) must be manually investigated because they include error description text that is not stored in the database.

Check `BMF_UNAPPLIED` for new records with `applied_status` = 0. These are payments that were not processed because EFT could not match them with any charges. If the payment applies, create a payment for the appropriate amount using the Payments interface and set `applied_status` = 1. If the payment does not apply, set `applied_status` = 2.

If EFT fails midway through a processing task, it picks up from the first unprocessed record when the file is reprocessed. If a record was partially processed (for example, a Create task that was interrupted after records were assigned to transaction files and `eft_trans_status` set to 2 (pending) but before the final payment send file was created), EFT does not complete it. Instead, before running EFT again run the script `cleanup_eft.sh` from the command line to reset partially processed records to unprocessed. The script prompts for the following information:

- Customer database containing the errored files
- user
- password
- task mode that was interrupted (Create, Post, Process)

For more information, see the *Technical Reference*.

EFT Investigation with EIU

You can investigate direct debit transactions — that is, records in the `EFT_TRANS` table — through the EFT Investigation interface (EIU).

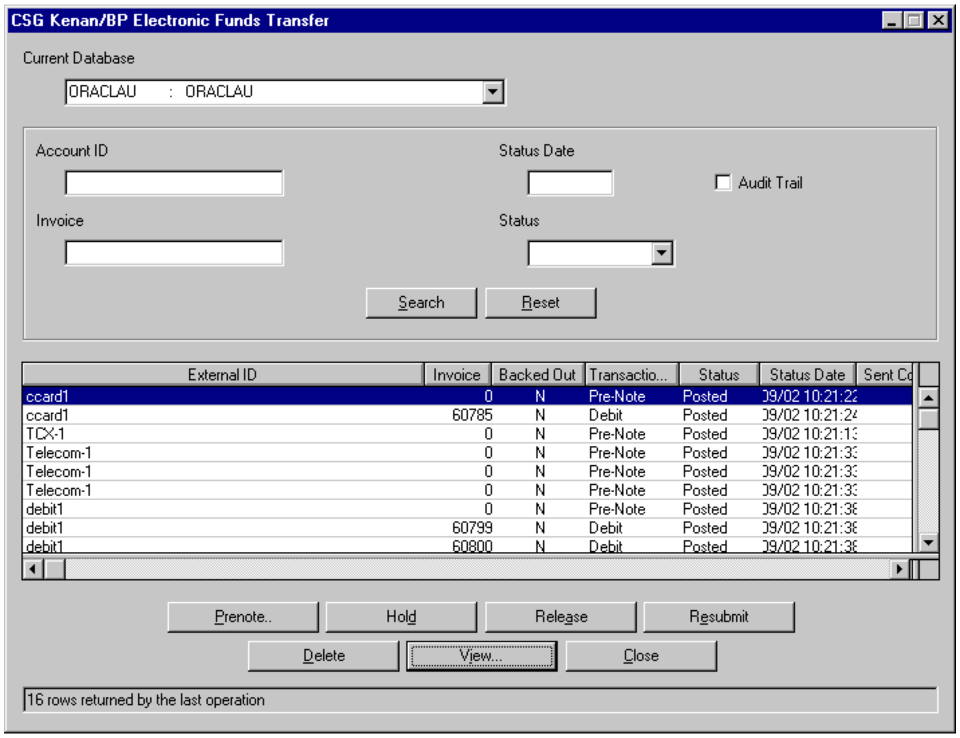
Access EIU through the Arbor folder under **Programs** in the Windows NT Start menu. In the Arbor folder select **Electronic Funds Investigation Unit**. Only by users in the `arboradm` user group can execute EIU.

Note The EIU interface looks very similar to the EFT interface available through Customer Care. However, only the EIU interface allows for the resubmission and deletion of transactions.

EIU Main Window

After a successful login the EIU main window appears as shown in figure 25.

Figure 25 EIU Main Window



ORACLAU : ORACLAU

External ID	Invoice	Backed Out	Transactio...	Status	Status Date	Sent Cc
ccard1	0	N	Pre-Note	Posted	09/02 10:21:22	
ccard1	60785	N	Debit	Posted	09/02 10:21:24	
TCX-1	0	N	Pre-Note	Posted	09/02 10:21:13	
Telecom-1	0	N	Pre-Note	Posted	09/02 10:21:33	
Telecom-1	0	N	Pre-Note	Posted	09/02 10:21:33	
Telecom-1	0	N	Pre-Note	Posted	09/02 10:21:33	
debit1	0	N	Pre-Note	Posted	09/02 10:21:38	
debit1	60799	N	Debit	Posted	09/02 10:21:38	
debit1	60800	N	Debit	Posted	09/02 10:21:38	

16 rows returned by the last operation

Click *Filter* to display EFT records. You can examine the queue of records, selecting and sorting the records by a number of different fields.

Each record has a status from the following list:

- Waiting — ready for transmission to the clearinghouse
- Sent — sent to the clearinghouse
- Accepted — accepted by clearinghouse
- Posted — accepted by Kenan/BP (written to the BMF table)
- Rejected — rejected; investigate further through View function

- Canceled — canceled by BIP (for example if the bill it applied to has been backed out)
- Deleted — deleted by an Kenan/BP user
- Reversed — reversed by clearinghouse after initial approval

EIU Prenote Generation

If no prenote has been sent to the EFT clearinghouse for a given account EIU can generate and send one. Press the Prenote button at the bottom left of the EIU interface, enter the account number when EIU requests it, and click *Save*.

Resubmitting Transactions

Transactions in EIU can be resubmitted. Select the desired transaction and click the *Resubmit* button at the bottom of the EIU interface.

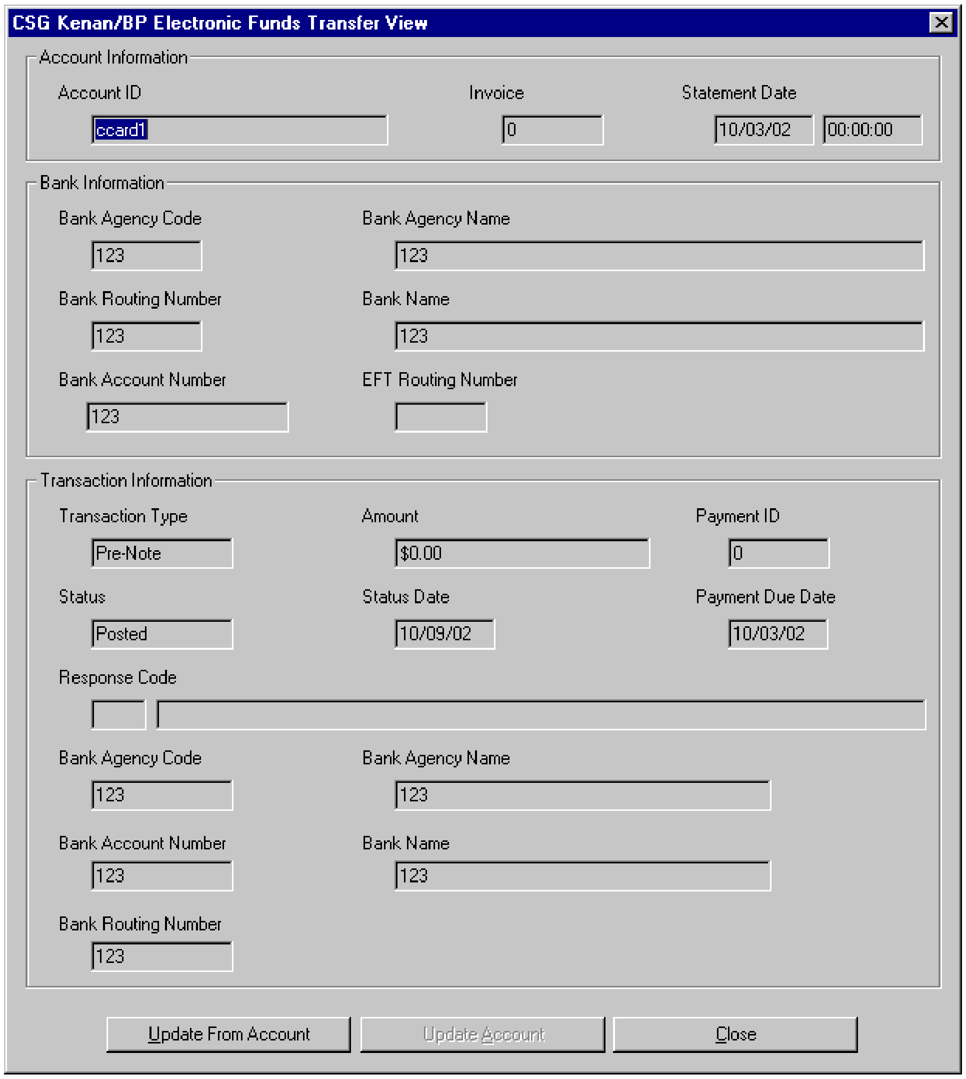
Deleting Transactions

To delete a transaction select it and click the *Delete* button at the bottom of the EIU interface. The selected transaction is removed from the list.

Viewing Records

To view a record, click the record once and then click the *View* button. The `EFT View` window appears as shown in figure 26.

Figure 26 EFT View Window



CSG Kenan/BP Electronic Funds Transfer View

Account Information

Account ID	Invoice	Statement Date
ccard1	0	10/03/02 00:00:00

Bank Information

Bank Agency Code: 123	Bank Agency Name: 123
Bank Routing Number: 123	Bank Name: 123
Bank Account Number: 123	EFT Routing Number:

Transaction Information

Transaction Type: Pre-Note	Amount: $0.00	Payment ID: 0
Status: Posted	Status Date: 10/09/02	Payment Due Date: 10/03/02
Response Code:		
Bank Agency Code: 123	Bank Agency Name: 123	
Bank Account Number: 123	Bank Name: 123	
Bank Routing Number: 123		

You can modify the `Bank Agency Code`, `Bank Agency Name`, `Bank Account No`, `Bank Name` and `Bank Routing No` fields. You can update the information on the transaction from the `CMF` record or update both the `CMF` record and the transaction data in the `EFT View` window:

- Click *Update from Account* to update the transaction details from the `CMF` record and save the information in `EFT_TRANS`.
- Click *Update Account* to update the transaction details from the bank details in the window and save the information to `EFT_TRANS` and `CMF`.
- Click *Cancel* to close the window without making any changes.

The response codes (taken from the `EFT_RESPONSE_CODE_VALUES` table) fall into one of the following categories and must be corrected by the following actions:

- **Account is closed**: Call the customer and arrange payment by an alternate method; then change that customer's payment method in the Customer Care interface. If the customer designates a new direct debit bank account resubmit the EFT. If another payment method is chosen, delete the EFT transaction.
- **No prenote**: Send a prenote.

- **Bank account information is incorrect**: Correct the direct debit bank account information for the customer in the Kenan/BP Customer Care interface and then resubmit the transaction.

Handling Reversals for Disconnected Accounts

If EFT processes a reversed payment transaction where the original transaction has an `eft_trans_status` of 8 (posted) and the account has an `account_status` of -2 (DISC_DONE) EFT copies the posted transaction and populates fields in the new transaction record as follows:

- `eft_trans_status` is 9 (ACCT_DISC)
- `file_id` is the file ID of the returns file
- `statement_date` and `payment_due_date` are the date from the record in the returns file
- `bmf_trans_type` is set to a value that maps to `EFT_RESPONSE_CODE_REF.reversal_trans_type`

  The value for `reversal_trans_type` is identical to the `bmf_trans_type` when the reversal is made to the BMF table. If `reversal_trans_type` is NULL EFT uses the default value of 11 (payment reversal) for the `bmf_trans_type` of the reversal.
- `no_bill` is 0 (this record is billable)

EFT then places a message in the report file indicating that the transaction status has changed to ACCT_DISC.

At this point you can use EIU to process the payment in one of two ways:

- Reactivate the account temporarily and resubmit the payment.
- Delete the payment and leave the account disconnected.

Select the payment transaction in EIU and click the *Resubmit* button or the *Delete* button.

If you click *Resubmit* EIU updates the `account_status` to 1 (DISC_REQ) and then reverses the posted payment and generates a chargeback transaction.

7 Administration

This chapter describes several modules typically scheduled by billing operators to perform administrative functions such as database archiving, replication, and population.

Transferring Files (COM)

The Communications process (COM) transfers files between Kenan/BP and *external sources* such as network usage recorders, credit card clearinghouses, banks, and so forth.

COM only transfers files; it does not process them. For example, COM does not parse usage records in any way when transferring usage files. Also, not all Kenan/BP input and output is mediated by COM. For example, LBX can transfer files using the CLEO communications package as described in "Processing Lockbox Payments (LBX)" on page 95.

Before You Run COM

Before you run COM, check that:

- external sources and *external contacts* for each source are configured properly
- access methods are configured properly

  For example, be sure the server on which COM runs is connected to the network and that all passwords, usernames, server IDs and so forth are current.
- local directories and remote system attributes match the configuration for each external contact

  For example, *ready, done,* and *error* directories must exist on the local system for each contact. They can have any name, although they often are called `ready`, `done` and `error`.

  Similarly, remote directories must exist for FTP and local directory transfers, user name and password for FTP transfers, modem phone numbers for Kermit access, and so forth.
- all outgoing files are in the local ready directory for the external contact and their names match the *name pattern* —that is, the prefix and postfix fields — for the external contact
- no subdirectories in the local ready directory match the name pattern for the external contact

  This causes an error when COM tries to move or delete the directory.

See the *Module Configuration Guide* for more information on configuring these items.

Defining COM Tasks

In MSA deployments run COM against the Admin server (see "Modules and Servers" on page 16).

By default, COM contacts every active external contact. For an outgoing contact COM sends all available outgoing files; for an incoming contact COM receives all available incoming files.

You can constrain the contact list by entering an SQL `where` clause for the `EXT_CONTACTS` and `EXT_CONTACTS_STATUS` tables in the `SQL Query` field for the task. For example, to configure a COM instance that contacts only external contact 1 use **`EXT_CONTACTS.ext_contact_id = "1"`** as the `SQL Query`. You can also constrain a COM instance to a set of external contacts with particular properties. Here are two examples:

- Suppose your database defines six external contacts each of which receives files from a separate source — five regional billing centers (source IDs 1–5) and one local mediation center (source ID 6).

Suppose sources 1–5 post files every two hours, and source 6 posts files once a day.

You can configure two COM processes to receive these files as follows:

- » `com01`, with a Recurring–Interval schedule recurring every 7200 seconds (two hours) and the following `SQL Query`: **`source_id >= 1 and source_id <= 5`**
- » `com02`, with a Recurring–Calendar schedule specifying the correct hour and minute and the following `SQL Query`: **`source_id = "6"`**

- Suppose the database contains two sources and four external contacts (one outgoing and one incoming for each source).

  Suppose both sources accept files between 4 and 6 A.M. and transmit files between 6 and 8 P.M.

  You can configure two COM processes to exchange files with these sources as follows:

  - » `com01`, with a Recurring–Calendar schedule specifying `Hour` (4, 5), `Minute` (00, 30), and the following `SQL Query`:
    `is_send = "1"`
  - » `com02`, with a Recurring–Calendar schedule specifying `Hour` (18, 19), `Minute` (10, 25, 40, 55), and the following `SQL Query`: **`is_send = "0"`**

  In this case, COM sends all outgoing files to all outgoing contacts at 4, 4:30, 5, and 5:30 P.M. and receives all incoming files every fifteen minutes between 6:10 and 7:55 P.M.

For more general instructions for configuring tasks see “Configuring Processes” on page 30.

Scheduling COM

Many other modules depend on COM to retrieve their input or transmit their output. For example, COM retrieves usage files for MCAP to route as described in chapter 4, “Usage Processing.” Therefore when receiving files be sure COM runs before the modules that process those files and when sending files be sure COM runs after the modules that create those files.

Otherwise, schedule COM whenever and as frequently as necessary. COM scheduling constraints usually result from the external contacts themselves — for example, some clearinghouses that are available only at certain hours.

COM often runs on different schedules for different external contacts — for example, retrieving usage files from a network mediation layer on one schedule and payment files from a credit card clearinghouse on a different schedule. See “Defining COM Tasks” on page 118 for an example.

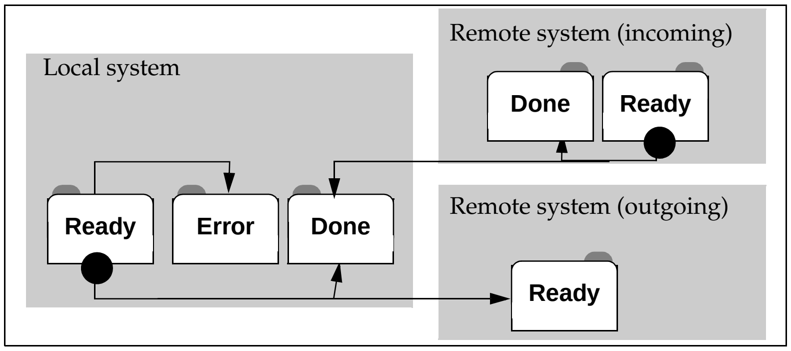
You must schedule two COM instances to retrieve IPDR documents. Schedule the first COM instance to retrieve IPDR documents and place them in the local Application Integrator directory. Schedule the second COM instance to run after Application Integrator has parsed the files to move them from its local output directory to the local directory MCAP retrieves files from. If these two directories are the same, the second COM instance is unnecessary. See the CSG *Application Integrator* User Guide for more information on scheduling Application Integrator. You must have Application Integrator installed to process IPDR documents.

Reviewing COM Output

You can review COM control reports and activity logs as described in “Reviewing Process Activity” on page 36. The COM control report shows you what files were moved, whether they were moved as data or control files, and whether COM encountered any errors.

You also can check the directory listings for each contact to verify successful transfers. Every time COM runs it moves and copies files (if any) between directories as figure 27 illustrates.

Figure 27 COM File Transfer



More specifically, COM does the following for each external contact:

- Outgoing contacts:
  - » Copies from local `ready` to remote `ready` directory

    COM appends the file ID to the original filename. For example, `990912.pmt` is renamed `990912.pmt.987` if the file ID is 987.
  - » Moves copied files to `done` directory
  - » Moves failed files to `error` directory or keeps them in the `ready` directory to retry depending on the contact and the number of failures

    If COM moves files to an `error` directory you must deal with them manually. See “Troubleshooting COM” on page 121 for more information.
- Incoming contacts:
  - » copies from remote `ready` to local `ready` directory

    COM prefaces the filename with the external contact ID and appends the server ID and file ID to it. For example, `990911.resp` is renamed `123.990911.resp.01.988` if the file ID is 988, the contact ID is 123, and the server ID is 01.
  - » for FTP and local contacts, moves copied files to remote `done` directory
  - » Failed files remain in the remote `ready` directory.

For example, suppose you have two contacts, one outgoing (`OUT`, with `ext_contact_id` 100) and one incoming (`IN`, with `ext_contact_id` 200) with directories as follows:

- Local `ready` directory for IN: `~arbor/data/IN/ready`
- Local `ready` directory for OUT: `~arbor/data/OUT/ready`
- Remote `ready` directory for IN: `~remote/data/IN/ready`
- Remote `ready` directory for OUT: `~remote/data/OUT/ready`
- Local `done` directory for IN: `~arbor/data/IN/done`
- Local `done` directory for OUT: `~arbor/data/OUT/done`
- Remote `done` directory for IN: `~remote/data/IN/done`
- Remote `done` directory for OUT: `~remote/data/OUT/done`

Suppose that all except the following files are empty (in other words, one file is in the local `ready` directory for OUT and one is in the remote `ready` directory for IN, but all other directories are empty):

```
~arbor/data/OUT/ready/990912.pmt
~remote/data/IN/ready/990911.resp
```

After you run COM against these contacts all directories are empty except for the following files:

```
~remote/data/OUT/ready/990912.pmt.987
~arbor/data/IN/ready/990911.resp.01.988
~arbor/data/OUT/done/990912.pmt
~remote/data/IN/done/990911.resp
```

You can conclude that COM successfully transferred both files.

Troubleshooting COM

If COM fails to transfer a file but leaves it in the `ready` directory the problem is probably temporary — for example, an intermittent network failure. Run COM again.

If COM transfers a file to the `error` directory the probable cause is that the file violates the external contact's parameters as configured in the database, such as a control file that was expected but not encountered. Check the external contact configuration as described in the *Module Configuration Guide* .

Otherwise check the most recent activity log and control report (see "Reviewing Process Activity" on page 36) for operation details and error messages.

Some common reasons for failure:

- failure of the external access method such as a network outage, missing executable, incorrect username, expired password, or incorrectly configured socket

  If the COM log file contains FTP error messages or similar system error information or if system-level error logs exist this is the most likely possibility. Try connecting to the external system manually using the same access method values configured in the database (described in the *Module Configuration Guide*) to verify the problem.

  To resolve this find the proper values to connect to the external system manually, reconfigure the external contact or access method using those values, and run COM again.

  You can check the status of an external contact through the `EXT_CONTACTS_STATUS` table. For example, if `error_code` is set communication with the contact failed, and if `retry_count` is set COM has attempted to communicate with this contact several times.
- COM does not recognize the file — for example, the external contact has a naming pattern the file fails to match, or an outgoing file was improperly registered in the database by another module.

  COM updates `FILE_STATUS` for each file it processes, so if no `FILE_STATUS` entry exists for the file (see "Reviewing File Status" on page 37) COM did not recognize it. Identify the cause of the failure by rechecking the external contact configuration (see the *Module Configuration Guide*) or output configuration for earlier modules. Resolve the configuration problem and run COM again.

- Directories defined for the contact do not match the actual directory names on the local system.

  Check the `arbor_ready`, `arbor_done`, `arbor_error`, and `arbor_work` fields in `EXT_ CONTACTS` against the actual directory names for this contact. If they differ either rename the directories or reconfigure the contact and run COM again.
- COM does not have permission to work with the local directories.

  Make sure user `arbor` has read, write, and execute permissions for all directories for this contact. If not, set permissions properly and run COM again.

If COM transfers a file but the file itself is altered, truncated, or corrupted the problem probably lies in the transfer mechanism itself since COM does not manipulate the file contents. For example, FTP may be transferring binary files in ASCII mode. Confirm your network configuration and the external access method setup as described in the *Module Configuration Guide* .

Translating Files (TIP)

TIP translates files from one record layout to another, usually by reading an ASCII text file and creating a new ASCII text file containing the same information in a different format. Sometimes TIP also can extract Kenan/BP database records and save them as ASCII text files in a given format.

Before You Run TIP

Before you run TIP to translate a file be sure that all records, fields, and mappings are properly configured for the file's record format. See the *Module Configuration Guide* for more information on configuring these items.

Defining TIP Tasks

In MSA deployments run TIP against the Admin server (see "Modules and Servers" on page 16).

TIP is a utility, so define a UTL process to run it as described in "Running Other Processes (UTL)" on page 126. Use an `SQL Query` value of the following form:

```
TIP tip## server task <optional arguments>
```

where:

- *##* is a two-digit number (**01** to **99**)

  This defines the TIP process ID, which is unique on a given server. If you use an existing ID you are scheduling a second task for that instance.
- *server* is the server ID against which TIP should run
- *task* is either **in** or **out**

  For `in` tasks, TIP reads records from text files; for `out` tasks, TIP reads records from database tables. In either case TIP generates text files (see "Reviewing TIP Output" on page 124). TIP is most commonly used for `in` tasks. A standard use of TIP `out` tasks is creating reseller usage export files.

The following optional arguments are available for **`in`** tasks:

- **`-r`** *`record_type`*, where *`record_type`* is the record type ID (for example, **`A01`**)

  If you specify a *`record_type`* argument TIP parses input records into the fields defined for that record type.

  If you do not specify a *`record_type`* TIP parses input based on the record type ID in the first field of each record or in the header record for each file.

  If you do not specify a *`record_type`* and a record type ID does not appear in the record or header record TIP cannot translate the file.
- **`-s`** *`switch_id`*

  Provide a switch_id argument when several different versions of *`record_type`* are configured and you must select one for TIP to use. Do not provide a *`switch_id`* unless TIP has been configured with multiple versions of *`record_type`*.
- **`-d`** *`dir`*, where *`dir`* is the directory containing file(s) to be processed (defaults to `$ARBORDATA/tip/input/ready`)
- **`-f`** *`file`*, where *`file`* is a file in the *`dir`* directory
- **`-fp`** *`file`*, where *`file`* is the complete pathname for a file
- **`-pre`** *`prefix`*, where *`prefix`* is one or more characters with which every processed file name must begin
- **`-post`** *`postfix`*, where *`postfix`* is one or more characters with which every processed file name must end

The *`dir`*, *`file`*, *`prefix`* and *`postfix`* arguments control which files TIP processes:

- If you specify a *`file`*, TIP processes only that file.

  You can specify only one file at a time.
- If you specify a *`dir`*, TIP processes some or all files in that directory as follows:
  - » TIP does not process files beginning with ".".
  - » If you specify *`prefix`* and/or *`postfix`*, TIP processes only those files that match the prefix/postfix.
- If you specify neither *`file`* nor *`dir`*, TIP processes all new files in `FILE_STATUS` with source types of 7 or 8 (see "Reviewing File Status" on page 37).

  If you specify *`prefix`* and/or *`postfix`*, TIP processes only those files that match the prefix/postfix.

For example, suppose the `$ARBORDATA/test` directory contains the following files:

```
input1.txt
input2.txt
otherinput1.txt
otherinput2.txt
```

You can control which file(s) TIP processes using the following optional arguments:

- `-fp $ARBORDATA/test/input1.txt`

  TIP processes `input1.txt`.
- `-dir $ARBORDATA/test -f input1.txt`

  TIP processes `input1.txt`.
- `-dir $ARBORDATA/test`

  TIP processes all four files in the directory.

- `-dir $ARBORDATA/test -prefix in`
  TIP processes `input1.txt` and `input2.txt`.
- `-dir $ARBORDATA/test -postfix 2.txt`
  TIP processes `input2.txt` and `otherinput2.txt`.
- `-dir $ARBORDATA/test -prefix in -postfix 2.txt`
  TIP processes `input2.txt`.
- `-f input2.txt`
  TIP looks for `$ARBORDATA/tip/input/ready/input2.txt` and processes it if it exists.
- `-prefix in`
  TIP processes all new files in `FILE_STATUS` with source types of 7 or 8 and whose names begin with `in`.

Scheduling TIP

To schedule TIP schedule the UTL instance that runs it (see “Running Other Processes (UTL)” on page 126 for more information).

Typically, the TIP schedule depends on other process schedules — for example, when preprocessing usage files you must schedule TIP to run between two instances of COM (one to retrieve the original usage files from the external network, and one to register the translated files as available for MCAP processing) as shown in figure 9 on page 45.

Depending on the record formats, translating records can be time-consuming. Be sure to allow enough time in your process schedule. Run TIP against a sample file of known size to estimate TIP throughput.

Reviewing TIP Output

You can review TIP control reports and activity logs as described in “Reviewing Process Activity” on page 36. The amount of information in these log files depends on the `$TIPLOGLEVEL` environment variable.

TIP updates `FILE_STATUS` for processed files and can update other database tables depending on its configuration.

TIP places new files in the `$ARBORDATA/tip/output/ready` directory if successful with names of the following form:

`in.`*source_type*`.`*filename* (for `in` tasks)
`out.`*record_type*`.`*sort_key*`.`*datetime*`.data` (for `out` tasks)

where:

- *source_type* is the source type for the external contact for the original file
- *filename* is the original file name
- *record_type* is the original record type or table name
- *sort_key* is a preconfigured value from the database
- *datetime* is the date and time the file was created in `YYMMDDhhmmss` format

For `in` tasks TIP creates one file for each file it reads. For `out` tasks TIP creates one file for every *sort_key* value.

For example, if two files (`input102912.txt` and `input928213.txt`) exist in `$ARBORDATA/input1/ready` with source type 7 and you run **`TIP tip01 in -d $ARBORDATA/input1/ready`** TIP creates two files (`in.7.input102912.txt` and `in.7.input928213.txt`) in the `$ARBORDATA/tip/output/ready` directory.

If TIP encounters input records it cannot process it copies them into identically named files with an `error` extension in the `$ARBORDATA/tip/input/error` directory. So if the two files above each have 70 records, five of which are in error, TIP creates `in.7.input102912.txt` and `in.7.input928213.txt` as described above with 65 records each and two files (`in.7.input102912.txt.error` and `in.7.input928213.txt.error`) in the `error` directory with five records each.

Troubleshooting TIP

Check the most recent activity log and control report (see "Reviewing Process Activity" on page 36) for operation details and error messages.

If no files appear in `$ARBORDATA/tip/output/ready`:

- The probable reason is that TIP did not recognize the input files to be translated.

  If you listed files explicitly in the task definition (for example, using the **`-fp`** option) confirm those pathnames against the actual incoming file pathnames. If they do not match redefine the task and rerun TIP.

  Otherwise check the `FILE_STATUS` table for matching entries (see "Reviewing File Status" on page 37). If no such entry exists TIP did not recognize the file.
- If the input file format includes a trailer record the number of data records probably does not match the count specified in the trailer record. Check the input file.

If translated files exist but are missing records check the `$ARBORDATA/tip/input/error` directory for files with an `error` extension.

- If error files contain few records or appear intermittently the original records are probably the reason — for example, the network mediation layer leaving out information. Recheck the original files.
- If large error files start to appear for incoming record types that TIP previously handled successfully a changed file format used by the network mediation layer is probably the reason. Compare current records with earlier, successfully handled records.
- If large error files appear consistently flawed TIP configuration is probably the reason. Recheck the `RECORD_*` table configurations as described in the *Configurator Guide*.

If no error files exist but the translated files subsequently fail (for example, if MCAP cannot process usage files translated by TIP) TIP configuration is probably the reason — for example, input values mapped to the wrong locations on output records or incorrect defaults. Recheck the `RECORD_*` table configurations as described in the *Configurator Guide*.

Replicating File Status (RepStat)

RepStat replicates table changes across servers (see “Modules and Servers” on page 16). It is used commonly in MSA deployments to keep `FILE_STATUS` synchronized on all servers. See “Reviewing File Status” on page 37 for more information about the `FILE_STATUS` table. Running RepStat is unnecessary in a single-server deployment or a deployment with only one CustAdmin database.

RepStat is a utility, so define a UTL process to run it as described in “Running Other Processes (UTL)” on page 126 with one of the following SQL Query strings:

`RepStat ALL` (replace all records on target servers)

`RepStat UPDATE` (update only changed records)

Schedule **`RepStat UPDATE`** to run against the Admin server before and after each of the following modules run on a Customer server:

- COM
- MCAP
- CAP
- EFT
- CPM

Otherwise other processes on that server cannot access the unbilled usage CAP creates. You can schedule RepStat to run as frequently as you wish; no harm is done by running it more often than necessary.

Running Other Processes (UTL)

The Utility process (UTL) is a general-purpose scheduling module that can be configured to execute any UNIX command at a given time. It is primarily for scheduling Kenan/BP utilities such as RAP and TIP, but you can also use it to schedule any UNIX executable program to run in batch mode (that is, they cannot be interactive).

Before You Run UTL

Be sure the executable you wish to run is available and that you know the necessary command-line arguments.

Standard Kenan/BP utilities are described elsewhere in this document. For example, the Usage Preprocessor (TIP) is described in “Translating Files (TIP)” on page 122 and the Usage Reprocessor (RAP) is described in “Reprocessing Usage (RAP)” on page 61.

Custom Kenan/BP utilities may be available in your deployment. These are described in your site-specific documentation.

Some utilities track their status in the `UTL_TASK_STATUS` table. No more than one entry can exist in this table for a given task, so be sure to remove any existing `UTL_TASK_STATUS` entries before running UTL.

Defining UTL Tasks

The `SQL Query` field controls UTL's operation. Enter a valid UNIX command with the desired arguments in this field. This UNIX command runs as UNIX user `arbor`. For Kenan/BP utilities the appropriate `SQL Query` entry is documented along with the utility.

For general task definition instructions see "Configuring Processes" on page 30.

See the description of Kenan/BP utilities for module-specific scheduling issues.

Reviewing UTL Output

Some utilities track `status` (1 = ready, 2 = in progress, 3 = error) in the `UTL_TASK_STATUS` table.

Moving Accounts between Servers (AMP)

The Account Mover (AMP) moves accounts from one Customer server to another in a multiserver environment. Do not run AMP if you have only one Customer server.

Before You Run AMP

Use the ADD_MOVE_ACCT utility to schedule accounts for AMP to move, by running a UTL task with the following `SQL Query`:

> **`ADD_MOVE_ACCT`** *`<process_id> <account_external_id> <source> <target>`*

where:

- *<process_id>* is the AMP process that moves the accounts
- *<account_external_id>* is the account `external_id`. ADD_MOVE_ACCT adds all accounts in the same hierarchy as *`account`* to `ACCOUNT_MOVE_STATUS`, marking them for moving.
- *<source>* is the `server_id` (from `SERVER_DEFINITION`) of the Customer server on which the account currently is stored
- *<target>* is the `server_id` of the destination server

> **Note** You cannot designate the same server as source and target for an account move. AMP exits with an error.

For example, running UTL with the SQL Query

```
ADD_MOVE_ACCT amp01 100 3 4
```

and then running

```
AMP amp01 3
```

moves all accounts on server 3 in the same hierarchy as the account with `external_id` 100.

Configure and schedule ADD_MOVE_ACCT to run immediately, against the *<source>* Customer server (see "Configuring Processes" on page 30 for more information).

Note ADD_MOVE_ACCT does not move the accounts; it marks them for AMP to move later.

Defining AMP Tasks

AMP has three task modes:

- Full Move (task mode 0) — the default mode. Copies accounts from one server to another, updating the catalog database tables and deleting the originals.
- Move Only mode (task mode 1) — copies account records from the source server to the target server and updates catalog tables, but does not delete account records from the original server. Sets account `status` in `ACCOUNT_MOVE_STATUS` to 3 (catalog database updated).
- Delete Only mode (task mode 2) — deletes records of accounts moved by AMP in Move Only mode (`ACCOUNT_MOVE_STATUS.status` = 3) from the source server and updates the accounts' `status` in `ACCOUNT_MOVE_STATUS` to 4 (data removed from source).

AMP is a utility, so define a UTL process to run it as described in "Running Other Processes (UTL)" on page 126. Use an `SQL Query` value of the form

`AMP` *`<process_id> <source_server_id>`* [**`-mode`** *`<mode_id>`*]

where:

- *`<process_id>`* is the AMP process ID of the form **`AMP`***`##`*
- *`<source_server_id>`* is the `server_id` (from `SERVER_DEFINITION`) of the Customer server on which the account currently is stored
- *`<mode_id>`* is 0, 1, or 2

If you run a pair of Move Only and Delete Only AMP process instances, use the same *`process_id`* for each.

Run AMP separately against each Customer server (see "Modules and Servers" on page 16) to move accounts from the source server.

For general task definition instructions see "Configuring Processes" on page 30.

ACCOUNT_MOVE_TABLES.distinct_flag

Setting the field `ACCOUNT_MOVE_TABLES.distinct_flag` to minimize the amount of information AMP has to retrieve from a table while moving accounts improves performance.If this flag is set to 1 for a table, when AMP copies account information from that table to the table on the target server it retrieves only distinct values of data for the account being moved. If it is set to 0 for a table, AMP retrieves and copies all data for the account being moved from that table.

By default, `ACCOUNT_MOVE_TABLES.distinct_flag` = 1 for `BMF_BATCH, DISCOUNT_TRANS_MAP`, `HISTORIC_THRESHOLDS`, and `ADJ_TAX`. Performance can also be enhanced by setting this field for other tables depending on your deployment.

Scheduling AMP

To schedule AMP schedule the UTL instance that runs it (see “Running Other Processes (UTL)” on page 126 for more information).

Schedule AMP when no other modules are running on the server and do not schedule AMP between other cyclical modules. For example, do not schedule AMP between BIP and BID, or between SEND and GET tasks for CPM or EFT, or between MCAP and CAP, and so forth.

You can run multiple AMP processes at the same time in Oracle installations only. You can run AMP processes against the same source databases, target databases, both, or neither.

Note Do not run two AMP processes with the same `process_id` simultaneously.

If this is not possible (for example, if your process schedules are too complex to support “safe” windows) schedule several AMP runs at different times. AMP moves an account only if the following conditions apply to all accounts in the same hierarchy:

- no undispatched bills exist for the account
- no payments (EFT or CPM) exist that have not yet been accepted
- no usage awaits processing by CAP on the server
- no other module is processing the account

For general scheduling instructions see “Configuring Processes” on page 30.

AMP and ARCH

If you have previously archived information for an account on a given server (see “Running Other Processes (UTL)” on page 126) and then move the account to a different server using AMP you cannot later restore the archived records to the new server. If this is a concern restore all archived records for an account before moving the account.

Reviewing AMP Output

You can review AMP control reports and activity logs as described in “Reviewing Process Activity” on page 36. AMP logs include total successfully moved accounts, total errored accounts, and total accounts processed in the log file. AMP also records error messages for each account that failed processing.

Pay special attention to bill dispatching and journalization for moved accounts during the subsequent bill and journal period to make sure the chain of production is not interrupted.

Troubleshooting AMP

Failure of a complete account hierarchy to move usually results from one or more accounts in the hierarchy that fail to meet the conditions listed in “Scheduling AMP” on page 129. Schedule another AMP task at a time when those conditions are met.

Recovery

The system parameter RECOVERY_CONTINUE (AMP) toggles two recovery modes if AMP fails during processing. If this parameter is set to 1, AMP continues moving accounts from where it left off when it is restarted.

If RECOVERY_CONTINUE is set to 0, AMP rolls back account moves to their original state when it is restarted if their status in ACCOUNT_MOVE_STATUS.status = 0, 1, or 2 (scheduled to be moved, ready to be moved, and copied to the target server with a copy still on the original server and the Catalog database pointing to the original, respectively). Account information is deleted from the target server if status = 2, and in all cases status is reset to 0.

Accounts with status of 3 or 4 (account data on both servers and the Catalog database pointing to the target, and account data only on target server respectively) are not rolled back, since the account move was successful. When it is rerun, AMP ignores these accounts. You must manually delete the old account data on the original server for accounts with status = 3.

Loading Usage Points (LOAD_USG_PTS)

The Usage Point Loader (LOAD_USG_PTS) populates usage point database tables MPS uses during location-based usage processing. Among other things, MPS uses these tables to:

- display terminating city and state information for calls on a customer’s bill
- perform distance-based rating
- calculate taxes based on call origin or target location
- rate calls by jurisdiction

LOAD_USG_PTS reads files specifying geographical and network information for specific locations and loads that information into usage point tables.

Before You Run LOAD_USG_PTS

Before you run LOAD_USG_PTS be sure a LUP file containing the location information you want to load is available. A LUP file consists of several LUP records, each of which contains one field for each database field LOAD_USG_PTS populates.

One common strategy for creating LUP files is to configure and run TIP (see “Translating Files (TIP)” on page 122) to translate location source files into LUP files. For example, in North American telephony deployments location information is typically derived from the Local Exchange Routing Guide (LERG) and Terminating Point Master (TPM) files distributed monthly by Bellcore.

More specifically:

- LERG6.DAT files define the rate center, local access transport area (LATA), city, state, tariff, local operating company, and central office switch for each NPA/NXX in North America.
- TPM.DAT files define the city, state, and vertical and horizontal coordinates for each NPA/NXX.

If your deployment regularly receives and processes LERG6 and TPM files TIP probably already is configured to translate records from those files into records in a corresponding LUP file. Simply run TIP to perform the translation and then run LOAD_USG_PTS to process the translated file.

Defining LOAD_USG_PTS Tasks

LOAD_USG_PTS is a utility, so define a UTL process to run it as described in “Running Other Processes (UTL)” on page 126. Use an SQL Query value of the form:

```
LOAD_USG_PTS pID server mode LUPfile
```

where:

- *`pID`* is the LOAD_USG_PTS process ID (for example, **`lup01`**)
- *`server`* is the Admin server ID
- *`mode`* is one of the following:
  `FIRSTMONTH_FIRSTFILE` the first time you load these tables
  `FIRSTMONTH_ADDFILE` for additional files in the first month
  `FIRSTFILE` for the first file in subsequent months
  `ADDFILE` for additional files in subsequent months
- *`LUPfile`* is the LUP file path

Run LOAD_USG_PTS against the Admin server (see “Modules and Servers” on page 16).

For general task definition instructions see “Configuring Processes” on page 30.

Scheduling LOAD_USG_PTS

To schedule LOAD_USG_PTS schedule the UTL instance that runs it (see “Running Other Processes (UTL)” on page 126 for more information).

Run LOAD_USG_PTS to update the usage point tables as soon as the LUP file is available. Do not run LOAD_USG_PTS at the same time as any other billing modules.

A typical LUP file derived from `LERG6.DAT` and related files contains about 120,000 records usually takes four to six hours to initialize the tables. A typical monthly update contains about 4,000 records which usually takes about 10 minutes.

For general scheduling instructions see “Configuring Processes” on page 30.

Reviewing LOAD_USG_PTS Output

LOAD_USG_PTS creates and updates entries in the `USAGE_POINTS/_TEXT`, `ACCESS_REGION_REF/_VALUES`, `POINT_CLASS_REF/_VALUES`, and `POINT_REGION_REF/_VALUES` tables.

LOAD_USG_PTS also creates the following files:

- control reports in the `$ARBORCTRLRPT` directory with names of the form *process_name*`.`*datetime*

  summarizes errors, LUP records processed, and database records inserted, modified, and inactivated
- log files in the `$ARBORLOG` directory, with names of the form <*process_name*>*datetime*`.log`

  provides detailed information about state of execution and errors (if any). The amount of information in these log files depends on the `$LUPLOGLEVEL` environment variable.
- error files in the `$ARBORCTRLRPT` directory with names of the form `err`*process_name*`.`*datetime*

  contains LUP records that failed processing.

You can review LOAD_USG_PTS control reports and activity logs as described in “Reviewing Process Activity” on page 36.

Troubleshooting LOAD_USG_PTS

Parse errors indicate the LUP file is ill-formed, possibly caused by errors in TIP configuration or possibly caused by problems with the original data files from which the LUP files were generated. Review the original files to find the source of the problem. In the short term you can correct parse errors in error files manually using a text editor and then run LOAD_USG_PTS against the error file.

Database errors indicate violations to triggers or constraints on the usage point tables. The problem can be an ill-formed LUP file as above — for example, required fields left blank.

Another possible cause is your database configuration — for example, an LUP file that depends on a `country_code` that does not exist in the `COUNTRY_CODE_REF` table. In this case, add the new `country_code` and run LOAD_USG_PTS against the error file.

See the *Database Reference* for more detailed information about the usage point tables.

Refresh Jurisdiction Utility (RJU)

The Refresh Jurisdiction Utility (RJU) updates product jurisdictions in `PRODUCT_RATE_KEYS` BIP uses during jurisdiction-based RC rating. RJU reads files specifying product element, origin location, and target location combinations whose jurisdiction requires updating as of a given jurisdiction refresh date. It then ceases affected products and reprovides them with the new jurisdiction.

Before You Run RJU

Before you run RJU make sure that `PRODUCT_JURISDICTION` has been updated using ATM or Configurator with the new jurisdictions. RJU will read the new jurisdictions from it.

Also be certain that an input file containing the product element-origin location-target locations you want to update is available.

If you are running BIP in Flash mode, stop BIP before running RJU.

Defining RJU Tasks

RJU is a utility, so define a UTL process to run it as described in "Running Other Processes (UTL)" on page 126. Use an `SQL Query` value of the form:

```
RJU pID server product_level refresh_date filename
```

where:

- *`pID`* is the RJU process ID (for example, **`rju01`**)
- *`server`* is the Customer server ID against which RJU runs
- *`product_level`* is **`1`** (account level) or **`2`** (service instance level)
- *`refresh_date`* is the date to use as a lookup date for jurisdiction data in `PRODUCT_JURISDICTION` and as a cease-and-reprovide date in dynamic tables `CMF_PRODUCT_KEYS` and `EMF_PRODUCT_KEYS`
- *`filename`* is the name of an input file to RJU containing jurisdiction changes. It must contain lines of the following form:

  ```
  element origin target
  ```

  where *`element`* identifies a product's `element_id` value in `PRODUCT_JURISDICTION`, *`origin`* identifies an `origin_location_code` value, and *`target`* identifies a `target_location_code` value.

Run RJU against a Customer server (see "Modules and Servers" on page 16).

For general task definition instructions see "Configuring Processes" on page 30.

Scheduling RJU

Schedule RJU by scheduling the UTL instance that runs it (see "Running Other Processes (UTL)" on page 126 for more information).

Run RJU to update jurisdictions as soon as `PRODUCT_JURISDICTIONS` is updated and the input file is available. Do not run RJU at the same time as BIP. If BIP is running in Flash mode, stop it before running RJU.

For general scheduling instructions see "Configuring Processes" on page 30.

Reviewing RJU Output

RJU creates and updates entries in the `CMF_PRODUCT_KEYS` and `EMF_PRODUCT_KEYS` tables.

RJU also creates the following files:

- control reports in the `$ARBORCTRLRPT` directory with names of the form *process_ name*`.`*datetime*

  summarizes errors, files processed, and database records modified
- log files in the `$ARBORLOG` directory, with names of the form <*process_ name*>*datetime*`.log`

  provides detailed information about state of execution and errors (if any).
- error files in the `$ARBORCTRLRPT` directory with names of the form `err`*process_ name*`.`*datetime*

  contains files that failed processing.

You can review RJU control reports and activity logs as described in "Reviewing Process Activity" on page 36.

Troubleshooting RJU

Parse errors indicate the input file is ill-formed. Review the input file to find the source of the problem.

Errors may also occur due to `element_id`, `origin_location_code`, or `target_location_ code` in the input file not existing in `PRODUCT_JURISDICTION` or `CMF/EMF_PRODUCT_KEYS`.

See the *Database Reference* for more detailed information about the `PRODUCT_JURISDICTION`, `CMF_PRODUCT_KEYS`, and `EMF_PRODUCT_KEYS` tables.

8 Running Journals, Troubleshooting, and Audit Trail

This chapter describes running JNL, troubleshooting, and the audit trail. It also describes how to make changes and contains pointers to the setup documentation. For details on JNL runs see the *Journals Guide*.

Before You Run JNL

Be sure all JNL tables are properly populated. In particular be sure that:

- Journal codes exist for all transactions

  If new financial transaction types exist in the database since the last complete JNL run (for example, new adjustment types or service providers) be sure that `JNL_KEYS` and `JNL_KEYS_MASK` are properly populated.

  At least one complete active entry in `JNL_KEYS` must exist for each possible financial transaction.
- the Journal feed file and File Feed report formats are properly configured in the `JNL_FEED_LAYOUT`, `JNL_FEED_HEADER` and `JNL_FEED_TRAILER` tables
- `JNL_CYCLE` and `JNL_RUNS` include dates for at least this Journal period and the next one, if not more.

  Similarly, be sure that `JNL_SUBCYCLE` includes appropriate entries if subcycles are used.

See the *Journals Guide* for more information about JNL configuration.

Also, be sure that other processes (CAP and BIP in particular) have successfully updated financial data. Consider postponing a JNL run if there are too many errored transactions from other modules.

Defining JNL Tasks

In MSA deployments run JNL separately against each Customer server (see “Modules and Servers,” on page 16).

Most JNL task configuration exists in the `JNL_RUNS` table. See the *Journals Guide* for more information.

For general task definition instructions see “Configuring Processes,” on page 30.

Scheduling JNL

Schedule type 1 JNL runs as often as necessary, to spread the processor load over time. Schedule type 2, 3, and 4 runs in that order, once per journal period. (If you schedule all three run types for the same time, JNL automatically executes them in the correct order.) See the *Journals Guide* for more information about the journals cycle.

Be sure to schedule sufficient time to book all your transactions — JNL can take several hours to complete a cycle.

For general scheduling instructions see “Configuring Processes,” on page 30.

Reviewing JNL Output

You can review JNL control reports (see “JNL Control Reports” on page 137) and activity logs as described in “Reviewing Process Activity” on page 36. In addition, JNL produces the following output:

- Journal feed file for the third-party accounting system (Type 4 runs only).
- File Feed report containing information from the Journal feed file (Type 4 runs only).
- `JNL_RUNS_STATUS` entries on each Customer server, containing the current status of JNL runs on that server.

  Much of the information in this table is copied from the `JNL_RUNS` table when JNL starts.
- `JNL_DETAIL` entries for each journalable transaction (Type 1 and 2 runs only).
- `JNL_INVOICE_AUDIT` entries containing tracking information for each journaled invoice (Type 1 run only).
- `JNL_TRANS` entries that aggregate records from `JNL_DETAIL` (Type 3 runs only).
- `JNL_ERRORS` entries for errored transactions (see “Error Checking after a Journals Run” on page 139 for more information).

JNL also updates the following tables with tracking information:

- Type 1 and 2 runs update `BILL_INVOICE` and `JNL_DETAIL`.
- Type 3 runs aggregates information from `JNL_DETAIL` to `JNL_TRANS`.

See the *Journals Guide* for more information about JNL process flow.

Journal Feed File and Associated Report

The Journal feed file for input to the G/L system is produced by the Type 4 run. This is output to a subdirectory of the Kenan/BP home directory `/data/jnl/ready`. The naming convention for this file is:

<process ID>.*<date yymmdd>_<time hhmmss>_<jnl_ref_no><server - 2 digits>*

where the process ID is the JNL task number when run and jnl_ref_no is a database-resident tracking field for each JNL run.

The File Feed report is the associated human-readable report containing the fields specified in `JNL_FEED_HEADER`, `JNL_FEED_LAYOUT`, and `JNL_FEED_TRAILER` where `in_feed_report`=1. The headings for the fields are taken from `hdr_feed_description`. This can be found in a subdirectory of the Kenan/BP home directory `/data/reports/ctl`. The naming convention for this file is:

<process ID>.*<date yymmdd>_<time hhmmss>_<jnl_ref_no><server - 2 digits>*.`rpt`

where the process ID is the JNL task number when run, `jnl_ref_no` is a database-resident tracking field for each JNL run.

JNL Control Reports

JNL produces a control report for each run, containing status and summary statistics.

The first section of the report displays a unique journal reference number associated with the run, the run type, and the associated dates, times, and status.

If the run succeeds, the report displays the transactions by transaction type (`id_type`), currency code, the number of source records, the number of journal records, and the amount for each journaled record.

The Source Records column indicates how many transactions from the various Kenan/BP tables were extracted for journalizing. For backout runs, it indicates how many records are updated or deleted.

The other columns on the report differ depending on the type of Journals run:

- **Type 1 or 2 run**. The "JNL_DETAIL" column shows how many `JNL_DETAIL` entries were generated and always equals the Source Records column, except for usage and taxes (which have been aggregated based on grouping attributes) and recurring charges (which have been prorated into earned and unearned transactions).
- **Type 1 or 2 run**. lists transactions for which a `jnl_code_id` was not found in `JNL_KEYS` table
- **Type 3 run.** The JNL_DETAIL column shows the number of `JNL_DETAIL` rows that were summarized into aggregated `JNL_TRANS` entries. The Journal Trans column shows how many `JNL_TRANS` entries were generated.
- **Type 4 run.** The Journal Trans column shows the number of entries from `JNL_TRANS` that were joined with the `JNL_KEYS` table. The Feed Recs column shows the number of journal file feed records generated and always equals the JNL_DETAIL column from the corresponding Type 3 run Control Report.
- **Backout run**. displays number of deleted and updated records for each record type (`JNL_TRANS` or `JNL_DETAIL`), and the total amount backed out. The run type indicates the type of run that was backed out.

Figure 28 shows an example of the JNL control report.

Figure 28 JNL Control Report

```
ARBOR/BP                     JNL CONTROL REPORT
AUTHOR:  jnl01
PROGRAM: JNL                                                        PAGE:    1

----------------------------------------
JOURNAL TASK REFERENCE NUMBER 2
----------------------------------------

Journal Run Type             : 1-Journal Billcycle End
Currency Code                : UNKNOWN
Journal Period Start         : Jan  1 1998 12:00AM
Journal Period End           : Feb  1 1998 12:00AM
Journal Period Subcycle Start: N/A
Journal Period Subcycle End  : N/A
Journal Run Scheduled        : Feb  2 1998 12:00AM
Journal Run Started          : Jun  2 1998 3:35:42:760PM
Journal Run Status           : 1-SUCCEEDED

Record Type         Source Records JNL_DETAIL Recs Total Amount
------------------- -------------- --------------- ------------
01-Adjustment                    0               0            0
02-Payment                       0               0            0
03-Recurring Charge              0               0            0
04-Usage Charge                  0               0            0
05-Discount                      0               0            0
06-Non-Rec Charge                0               0            0
07-Tax                           0               0            0
08-Write-Off                   N/A             N/A          N/A
09-Unit Credit                   0               0            0
10-Errored Usage               N/A             N/A          N/A
11-Deposit Interest            N/A             N/A          N/A
------------------- -------------- --------------- ------------
                                 0               0            0
```

Troubleshooting JNL

Some notes to keep in mind when reviewing JNL output:

- Errored bills are not booked during a Type 1 run, and corrected bills are not booked during a Type 2 run. If errored bills are corrected after the normal billing Type 1 Journal cycle is complete, it may be necessary to schedule another Type 1 run to pick up the corrected bill.
- Type 1 or 2 runs search for valid transactions in `JNL_KEYS`. If a matching entry cannot be found for a particular transaction, the runs do not terminate, but the `jnl_code_id` field in the `JNL_DETAIL` table is not updated and details of these transactions are written to the Control Report. Check through these and create valid entries in `JNL_KEYS` before attempting the Type 3 run; otherwise, it fails.
- If a valid `jnl_code_id` cannot be found for transactions in a Type 3 run, the run fails after all the `JNL_DETAIL` rows have been processed. You must reverse the run. Use the information in the Control Report to identify and add the missing rows to `JNL_KEYS`, and rerun.
- The JNL tables `JNL_DETAIL` and `JNL_TRANS` do not contain complete information for a Journal period its end when all four run types are completed. Reason: some financial transactions, such as accruals, can only be generated at the Journal end period end (Type 2 run).
- Kenan/BP has no online function that enables a user to create bookable ad hoc financial transactions.

Error Checking after a Journals Run

At the end of any Journal run, the `run_status` in the `JNL_RUNS_STATUS` table and the control report shows whether the run was successful. Even if the `run_status` is “1,” indicating a successful completion, check the created data for accuracy.

Any error generates an entry in `JNL_ERRORS`. This table contains a free-form text description for each error condition and provides as much information as possible. Once the error is resolved, create a new entry in `JNL_RUNS` for the same JNL run type and Journal period that errored.

Caution For auditing purposes, do not change the existing entry in `JNL_RUNS`.

The following are causes of, and responses to, possible Journal errors:

- bad data in `JNL_KEYS`. The most common cause for errors during JNL runs is missing or inaccurate entries in `JNL_KEYS`. The free-form text in `JNL_ERRORS` lists the keys that make up the `JNL_DETAIL` or `JNL_TRANS` entry in error.

  If either of the following errors occurs, an error is written to the log file:

  » If multiple records are found in `JNL_KEYS`, JNL uses the first record found.

  » If no records are found, use the transaction keys provided in `JNL_ERRORS` to track and fix the `JNL_KEYS` entries.
- database/SQL server errors during read/update/insert. An unsuccessful journals run can create entries in `JNL_DETAIL` or `JNL_TRANS`. Because these records are indexed by a unique `jnl_ref_no`, which is linked to a failed run status in the `JNL_RUNS` table, they cannot be processed by a subsequent journals run. Run `cleanup_jnl.sh` script to return the database to a consistent state.

- hardware failures. Run the `cleanup_jnl.sh` script to return the database to a consistent state.
- bad data in `JNL_CYCLE` or `JNL_RUNS`. Any JNL run fails if the runs are scheduled incorrectly within the Journal period. Resolved this error by making the necessary adjustments to `JNL_CYCLE` or `JNL_RUNS` or both.
- SQL buffer overflow. If JNLs fails with a message in the log:

  (E 24140) DATABASE_ERR2 ERROR: Arbor data layer SQL buffer overflow

  then the setting for `BATCH_SIZE` is probably too big. A suggested setting to try is 20. If this fails, reduce it until it works.

Journal Backouts

If a run is aborted, the `cleanup_jnl.sh` script resets the database to a consistent state, undoing whatever journalizing occurred during the run. To reverse a complete run, perform a backout.

Backout JNL runs in the reverse order of their original run. For example, if you ran Types 1, 3, and 4, you would run in backout mode Types 4, 3, and 1. To setup a backout run in `JNL_RUNS`, the `jnl_backout` field must be set to 1 and the server must be identified. The other fields must be filled in as for a normal run of the required type. See the *Journals Guide* for more information.

When JNL runs in backout mode, it finds the last successful run for the given Journal period and undoes all of its effects within the database. You can schedule multiple backout runs to back out successive runs. After correcting the problem(s) that necessitated the backout, rerun the original JNL run(s).

The examples following describe two situations when backouts are run.

Example 1

In a given Journal cycle, Journals run Types 1, 2, 3, and 4 are all committed to the database when an error is found in `JNL_KEYS`. To correct this:

- schedule a backout to reverse the Type 4, 3, 2, and 1 runs in that order
- verify backout results through control file and log report after each run
- correct `JNL_KEYS` entries
- move the Journal feed file from the `/ready` directory
- reschedule and run another Type 1, 2, 3, and 4 run

Example 2

All JNL runs for a particular Journal cycle need to be backed out. In this case, the user schedules and runs each backout job separately, verifying each backout run before going on to the next one.

Note Backout control reports always show zero (0) destination records for each record type, because the backout process does not create output transactions.

When a backout is complete:

- `JNL_ERRORS` contains any errors that might have been generated from the original run
- the original run's `jnl_ref_no` entry remains in `JNL_RUNS`

Otherwise, all recorded transactions are completely deleted from the database.

In backing out a Type 4 run, the Kenan/BP system administrator must move the Journal feed file from the `/ready` directory to some other storage place if it is already but must be corrected and sent again.

Index