Jenzabar CX

Financial

Technical Manual

Filename: tmfinan
Distribution date: 11/12/2000

Contact us at www.jenzabar.com

JENZABAR, INC.
FINANCIAL TECHNICAL MANUAL

TABLE OF CONTENTS

SECTION 1 - USING THIS MANUAL

Overview

Purpose of This Manual

This manual provides technical information required to install, customize, and maintain the following Jenzabar products:

- Accounts Payable
- Cashier
- Check Writing
- Fixed Assets
- Financial Budgeting
- Requisitioning/Purchasing

Technical Information for Other Financial Products

Additional technical information for other Jenzabar financial products exists in the following manuals:

- *General Ledger Technical Manual*
- *RPA Technical Manual*

The *Jenzabar CX Technical Manual* also provides general information that relates to all Jenzabar products, including the financial products.

Intended Audience

This guide is for use by those individuals responsible for the installation, customization and maintenance of the CX.

How to Use This Manual

If you are not familiar with the processes and features of the financial products, read the manual for:

- Detailed reference information about how the products work
- Procedures for customizing and maintaining the products

If you are familiar with the processes and features of the financial products, and just need specific reference information or a procedure, look through the Table of Contents or Index and refer to the pages you need.

Product Differences

This manual contains information for using all the features in the financial products. Your institution may or may not have all the features documented in this manual.

Standard Product

This manual provides technical information about the standard CX product. If your institution changes CX to meet its specific needs, then your tables and records may differ from those shown in this manual. Your screens will also vary from the standard CX program screens if you use CX in character-based format.

Structure of This Manual

This manual contains both general reference information and procedures for installing and maintaining the financial products. The manual's organization follows:

Overview information:

Section 1 - Information about using this guide

Section 2 - Overview information about the products

Module reference information:

Section 3 - Tables used in the products

Section 4 - Macros and Includes

Section 5 - 29 - Financial program documentation

Section 30 - Menus, Screens, Scripts, and Reports

Module procedures:

Section 31- Procedures for installing and customizing your processes

Section 32 - Procedures for maintaining the products

Error reference/Recovery procedures:

Section 33 - A reference of fatal and serious errors and recovery procedures

Reference information:

Index

Related Documents and Help

The following resources are also available to assist you in installing, supporting, maintaining and using the financial products.

QuickMate online help:

- *Using QuickMate*
- *Getting Started User Guide*

UNIX-based help:

Help command (**Ctrl-w**) in screens and menus

User guides:

- *Using Cashier*
- *Using Financial Budgeting*
- *Using Fixed Assets*
- *Getting Started User Guide*
- *Using Checkwriting*

Conventions Used in This Manual

Introduction

Jenzabar, Inc. has established a set of conventions to help you use this manual. The list of conventions presented below is not exhaustive, but it includes the more frequently used styles and terms.

Style Conventions

Jenzabar CX technical manuals observe the following style conventions.

Boldface type
Represents text that you type into the system (e.g., Type **UNDG**) and command names or keys you use to execute a command or function (e.g., **Finish**).

Bulleted list
Show items not ranked or without a sequential performance.

CAUTION:
Indicates a caution or warning of a potential risk or condition.

<Enter>
Represents the Enter, Return, Line Feed or ↵ key on your keyboard.

Italic type
Is used in any of these ways:

- To represent a new or key term
- To add emphasis to a word
- To designate a program name (e.g. *identry*)
- To cross-reference a section of text
- To represent a variable for which you substitute another variable (e.g., substitute *filename* with an appropriate filename)

<Key name>
Represents a key that you must press.

Note:
Indicates a note, tip, hint or additional information.

Numbered lists
Show ranking of items or sequence of performance.

Parentheses
When used around a field name, indicate the field is unlabeled. The field description includes the location of the field.

Quotation marks
Represent information written in this guide exactly as it appears on the screen.

Example: The message, "Now Running..." appears.

Jenzabar-Specific Terms

Some terms used in this manual may be unfamiliar to you, either because they are terms you have not used before or because Jenzabar, Inc. has assigned a slightly different meaning to a familiar term. The following list identifies and explains the most common Jenzabar CX-specific terms:

Application
One or more software programs that enable you to perform a particular procedure, such as entering student information.

Data
Specific information you enter into fields on a particular data entry screen.

Enter
To type information on a keyboard and execute by any of the following actions:

- Pressing the <Enter> key
- Clicking on the OK button
- Selecting Finish

F key
Any of the function keys located on your keyboard (e.g., <F1>).

Hot key
The capitalized and underlined (or highlighted) letter of a command on a menu.

ID
The number assigned to each student or organization associated with your institution (e.g., 12345).

Parameter
A variable in the system that is given a constant value for a specific application (e.g., a date can be a parameter for producing a report).

Select
To execute a command by any of the following actions:

- Performing the keystrokes
- Pressing the hot key
- Highlighting the command or option and pressing the <Enter> key
- Clicking with the mouse

System
The Jenzabar CX product.

Keystrokes

When you see two keys separated by a dash (e.g., <Ctrl-c>), hold down the first key (Ctrl) while pressing the second (c).

SECTION 2 - FINANCIAL PROCESSES

Overview

Introduction

This section provides information on the purpose and process flow of the CX financial processes.

Purposes of Products

The primary purposes of the financial products are to enable institutions to do the following:

- Process purchases and accounts payable outside of the Requisitioning, Purchasing and Accounts Payable products
- Perform cashiering functions
- Write checks
- Process fixed assets
- Process payrolls
- Create and use budgets
- Post charges and financial aid to student accounts
- Print statements for accounts receivable

Background Knowledge

The following list describes the necessary background information that you should know to implement and support the financial products.

UNIX

Know the following about the UNIX operating system:

- Csh (C-Shell) environment and commands
- Editor commands (e.g., vi)

INFORMIX-SQL

Know about the following INFORMIX tools:

- SQL database
- PERFORM screens
- ACE reports

Jenzabar CX database tools and utilities

Know how to use the following database tools:

- MAKE processor
- Schemas
- Macros
- Includes
- Program screens
- The product update process

Jenzabar CX

Know the following about the CX standard product:

- The CX directory structure
- The menu processor
- The CX database engine

QuickMate features

Know the following about the CX Graphical Server:

- Client/Server processing
- Telnet settings
- Keyboard settings
- Mouse settings
- GUI mode commands

C Programming

If you want to modify any CX programs to meet unique needs at your institution, you must attend the CX Platform class and know how to use the C programming language.

Financial policies and procedures

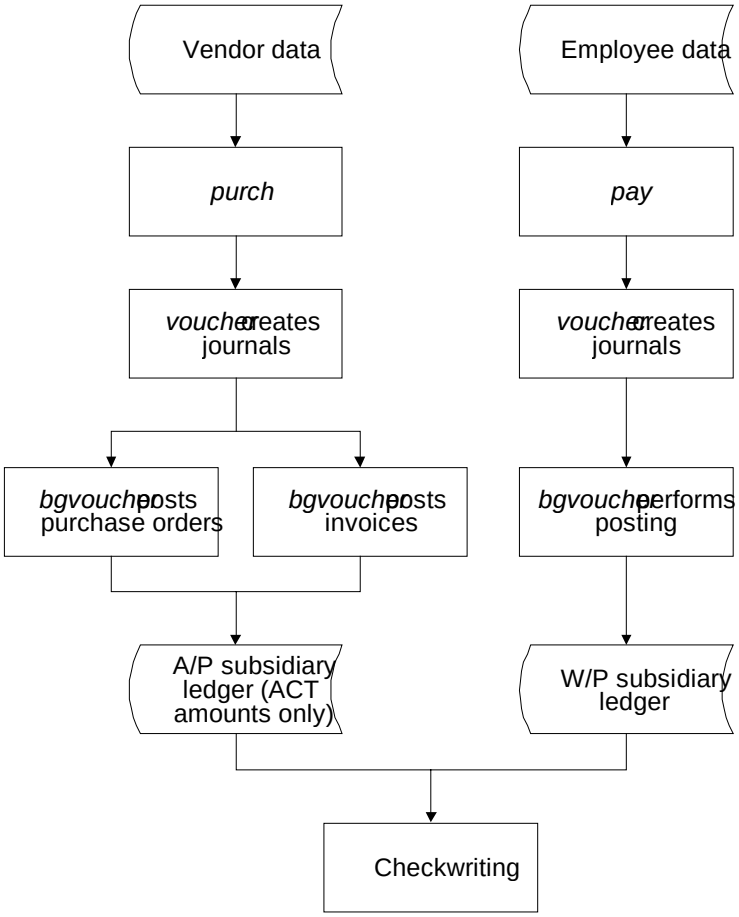
Know answers to the following questions:

- What CX product does the institution use for processing purchases and accounts payable?
- How does the institution approve Requisitions/POs/Invoices?
- Who is authorized to create PO's?
- How does the institution prepare budgets?
- What departments can process checks?
- How does the institution prepare 1099 forms?
- How does the institution account for fixed assets?

Accounts Payable Flow

Diagram

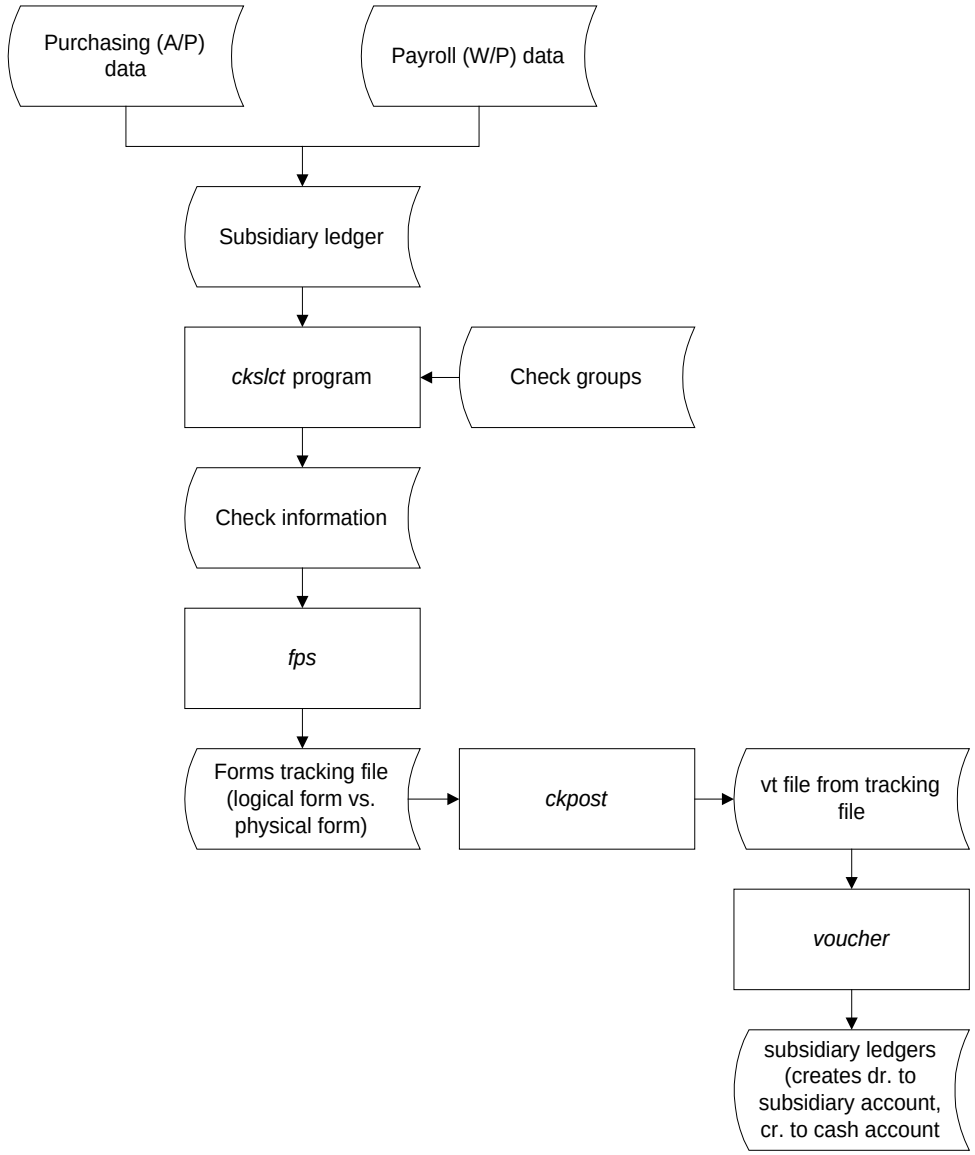
The following diagram shows the process in the Accounts Payable product and compares it to the Payroll product.



Checkwriting Process Flow

Diagram

The following diagram shows the process for writing checks.



Process Description for Checkwriting

The following processes relate to checkwriting.

1. The *ckslct* program uses information from Purchasing and Payroll subsidiary ledgers to select and create checks.
2. The *fps* program prints checks.
3. The *ckpost* program posts the transactions crediting cash and debiting the appropriate subsidiary ledgers.

Checkwriting Process for Payroll Checks

The following table shows the menu options and programs to execute when processing payroll checks. The table also lists the names of CX records/tables that are impacted by the process, the names of the forms used in the process, the status of records at each point in the process, and a brief description of the purpose of each step in the process.

Menu option	Program	Table	Forms	Status	Description
Select Checks	ckslct	ckreq_rec	prcheck.i#	N/A	new ckreq_rec will be added for the group (ckgrp_rec)
		subb_rec	prcheck.i#	L	subb_rec will be locked and the subb_rec.req_no will be populated with the number from the ckreq_rec
		vch_rec	prcheck.i#	N/A	new PR journal will be created for the liabilities
		ckgrp_rec	prcheck.f#	W	ckgrp_rec status will be updated to Written, and the extension on the prcheck form file will change from I# to f# when ckslct has completed

					Criteria for the success of check selection:
					subb_rec.subs = ckgrp_rec.subs;
					subb_rec.bal_prd = ckgrp_rec.prd;
					subb_rec.amt_act != 0:

					subb_rec.amt_enc = 0;
					subb_rec.due_date <= chgrp_rec.due_date;
					subb_rec.stat = "O";
					subb_rec.hold_pay = "N";
					subb_rec.ck_req_no – 0

Print Checks	fps	doc_table	prcheck.t#		a new form file will be added in spool/forms starting with "t" and ending with the same # number.

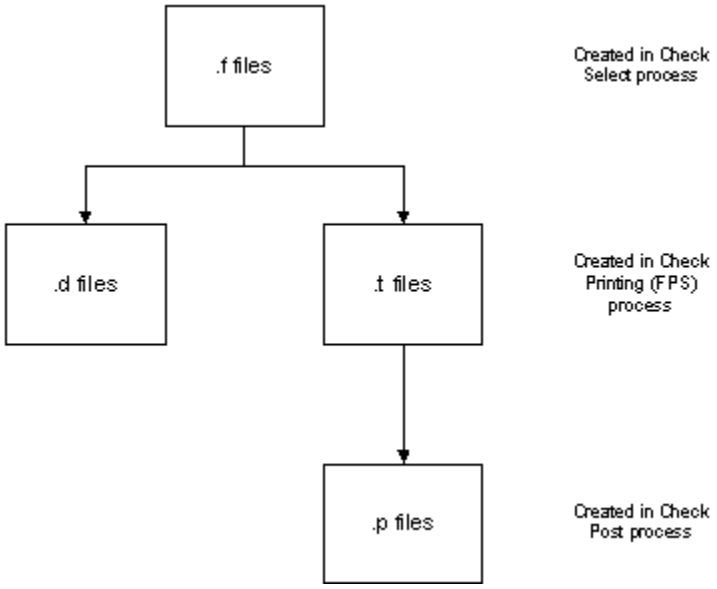
Menu option	Program	Table	Forms	Status	Description
					The check number will be retrieved from the doc_table. last_issued_number, incremented by 1.
Post Checks	ckpost	subb_rec	prcheck.t#	C	the subb_rec status will change to Completed
		doc_table	prcheck.t#	N/A	the doc_table. last_issued_number will change to match the last check number printed
		vch_rec	prcheck.t#	N/A	a new PD journal will be created for check posting
		ckgrp_rec	prcheck.p#	P	the ckgrp_rec.jrnl_ref and ckgrp_rec.jrnl_no will be updated with journal information, and the ckgrp_rec.stat will be updated to Posted when ckpost completes
Termi-nate Check Journal	termpost	vch_rec	prcheck.p#	V	the vch_rec (found in the ckgrp_rec) is terminated with a Void status
		subb_rec	prcheck.p#	L	the subb_rec.stat is changed from Completed to Locked
		doc_table	prcheck.p#	N/A	the doc_table. last_issued_number is reset back to the first check number prior to this check group
		ckgrp_rec	prcheck.t#	W	the ckgrp_rec.stat is updated from Posted to Written. The spool/forms file name will change from prcheck.p# to prcheck.t#
					Note: the option to "Terminate Journal" should never be used since it will not update the subb_rec and ckgrp_rec, causing the links between records to be lost
					Used when voiding the check group (opposite of "Post Checks"

Menu option	Program	Table	Forms	Status	Description
Inter-rupted Posting	intpost				Similar to above with the exception that this option is used if "Post Checks" completes unsuccessfully
Prepare for FPS Restart	restform	N/A	prcheck.t# prcheck.f#	N/A	the prcheck.d# is moved to prcheck.f#
Prepare for Check Abort	remtrk	N/A	N/A	N/A	the prcheck.t# is removed
					First step to backing out the selection of the checks (opposite of "Check Select")
Check Abort	ckabort	subb_rec	N/A	O	the subb_rec.stat is updated from Locked to Open, and the subb_rec.ckreq_no is updated to zero (0).
		ckgrp_rec	N/A	S	the ckgrp_rec.stat is updated from Written to Started
					the prcheck.f# is removed.
					second step to backing out the selection of the checks (opposite of "Check Select")

* For each form, the # symbol represents a system-defined serial number that uniquely defines the form file stored in the spool/forms directory.

Form Extensions in Checkwriting

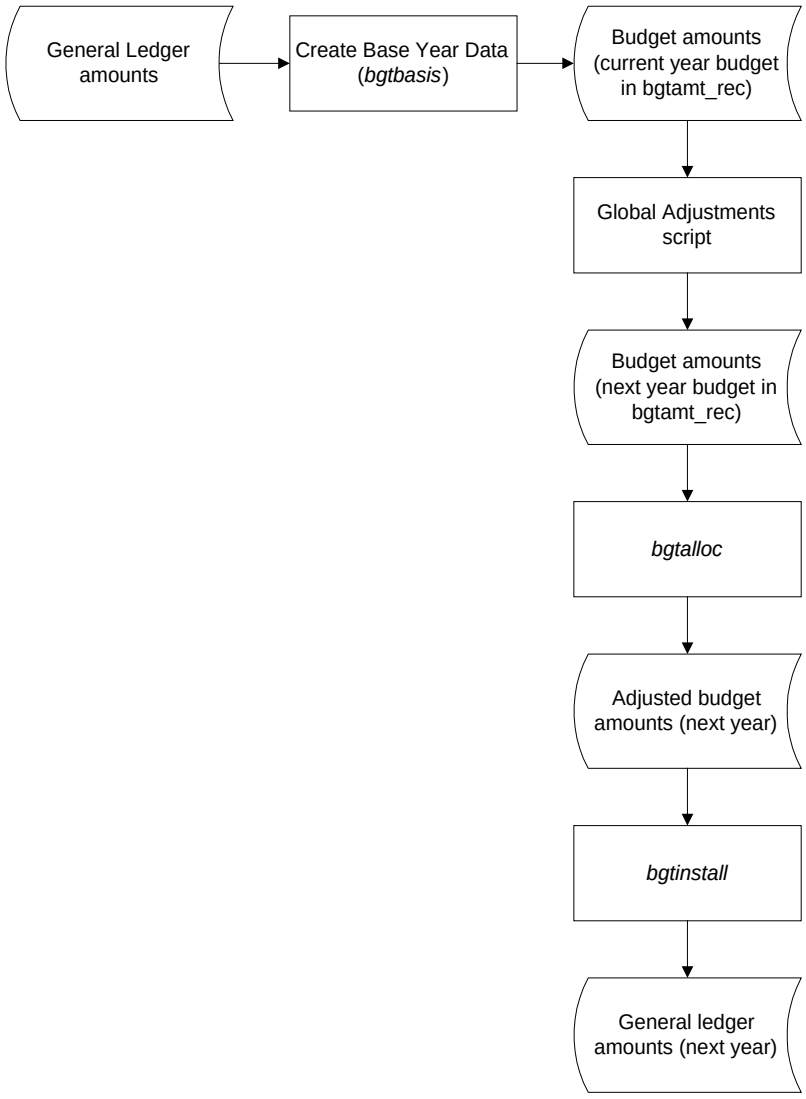
The forms files located in spool/forms have extensions that reflect their progress through the checkwriting process. The following diagram shows the steps of the process, along with the file extensions. Note that each file extension is alphabetic, but in practice, a numeric sequence number follows the alphabetic character to enable you to identify the order in which files have been created.



Budgeting Process Flow

Diagram

The following diagram shows the process for creating a budget.



Process Description for Budget

The following processes create a budget.

1. The user verifies that the Budget Calendar record (bgtcal_rec) contains valid records for the budget years and amount types.
2. The *bgtbasis* program copies information from the general ledger as a starting point for creating the new budget.
3. The Global Adjustments script *addbgtamt* makes a working copy of the information copied in step 2 above, setting the amount type to the code used for budget (e.g., REQ, APP or BGT). It also adjusts amounts based on the information from the Budget Distribution table (bgtdist_table).
4. The user executes the *bgtalloc* program to adjust the budgeted figures.
5. The user executes the *bgtinstall* program to post the approved budget (generally APP) to the general ledger.

Note: Typical processing at some institutions would include the following steps:

- The business office creates a REQuested budget.
- Based on requests from the various departments, the business office updates the REQ, using *bgtalloc.*
- The business office runs the *addbgtamt* process to copy REQ to APProved.
- Using *bgtalloc*, the business office updates the APP with final adds, cuts and adjustments.
- The business office uses *bgtinstall* to post the APP to BGT in the general ledger.

Module Relationships

Related Jenzabar CX Modules

The financial products interact with several other CX product areas. The following list describes the interrelationships.

General Ledger

General Ledger impacts the financial products and processes in a variety of ways, including the following:

- CX posts the accounting information you enter or create in the financial products using the Background Voucher (*bgvoucher*) program. The following programs call *bgvoucher*:
  - Journal Entry (*voucher*)
  - Check Posting (*ckpost*)
  - Budget Install (*bgtinstall*)
  - Student Billing (*billing*)
- You can use Accounting Query (*acquery*) and Subsidiary Account Query (*saquery*) to view the accounting effects of using the financial products.
- If you experience a system failure when posting a journal, you use the Voucher Recovery (*vchrecover*) program to clear up the results of the system failure.
- You can archive obsolete records created with the financial products using the Subsidiary Archive (*sarc*) program.
- The Budget Review (*bgtreview*) program enables you to compare the actual posted amounts to the amounts created in Financial Budgeting.

Payroll

The Payroll product uses the financial function for check writing to process payroll checks.

SECTION 3 - FINANCIAL TABLES AND RECORDS

Overview

Introduction

This section provides you with reference information about the records and tables associated with the financial products.

Some financial records and tables relate most directly to the General Ledger, Purchasing and Accounts Payable, Requisitioning, or Payroll components of the financial products, and are included in documentation for those components. This section includes reference information for the following records and tables, which relate most directly to the CX as documented within this manual.

Budgeting

- bgtacct_rec
- bgtamt_rec
- bgtcal_rec
- bgtconstr_rec
- bgt_detail_rec
- bgtdist_table
- bgtresp_rec
- bgtsum_rec
- pbgt_rec

Cashiering

- cashent_table
- chrecon_rec

Checkwriting

- ckalloc_rec
- ckgrp_rec
- ckreq_rec
- f1099_rec
- f1099_table
- grphist_rec
- grpreq_rec
- precon_rec
- r1099_rec
- recon_rec

Fixed Assets

- depas_rec
- fix_rec
- fix_table
- fixhist_rec
- fixmaint_rec
- fixskel_rec

Purchasing and Accounts Payable

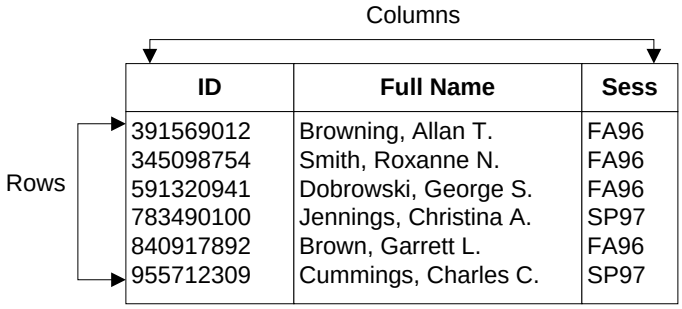
- appid_rec
- appr_table
- commod_table
- goods_table
- payfrm_table
- payterm_table
- po_rec
- pobody_rec
- podtl_rec
- splr_rec
- venqual_table
- ventype_table
- vnd_blob
- vnd_rec

Miscellaneous Financial

- f990rpt_rec
- glecred_rec
- glename_rec
- typ_table

What Is an SQL Table?

In a relational SQL database, a table is an organized set of any kind of data, regardless of its purpose for validation or information maintenance. The basic unit of organization of a table is a column, a category of data. A table can have multiple columns, and columns typically contain multiple rows of data.



What Is a CX Table?

Jenzabar, Inc. makes name distinctions in the usage of database tables. A *table* in CX contains information that remains static and is denoted with the *_table* extension. For example, the State table, named *st_table,* contains the list of the United States of America. On the CX menu, you can access most tables in Table Maintenance menus using PERFORM screens.

What Is a CX Record?

Jenzabar, Inc. makes name distinctions in the usage of database tables. A *record* in CX is a table that contains information that changes on a regular basis and is denoted with the *_rec* extension. For example, the Alternate Address record, named *aa_rec*, contains any other addresses at which students can be contacted, such as a summer address. You access records in CX program screens, detail windows, and PERFORM screens.

Common Tables and Records

Programs in the financial products use several tables and records that appear throughout the CX. The tables and records are:

- Alternate Address record (aa_rec)
- Alternate Address table (aa_table)
- Address table (adr_table)
- Alternate Name record (addree_rec)
- Country table (ctry_table)
- State table (st_table)
- ID record (id_rec)
- Profile record (profile_rec)
- Title table (title_table)
- User ID table (userid_table)

General Ledger Tables and Records

Programs in the financial products use several tables and records that appear or originate in other product areas. The following tables and records are most commonly used in the General Ledger product. For more information about these tables and records, see *General Ledger Technical Manual*.

- Amount Type table (atype_table)
- Defined Account record (gld_rec)
- Document table (doc_table)
- Entry Type table (ent_table)
- Financial Statement table (fs_table)
- Fiscal Calendar record (fscl_cal_rec)
- Function table (func_table)
- Fund table (fund_table)
- General Ledger Account record (gla_rec)
- General Ledger Amount record (gl_amt_rec)
- General Ledger Entry record (gle_rec)
- General Ledger Transaction record (gltr_rec)
- Journal record (vch_rec)
- Journal table (vch_table)
- Object table (obj_table)
- Subfund table (subfund_table)
- Subsidiary Account record (suba_rec)
- Subsidiary Association table (subas_table)
- Subsidiary Balance record (subb_rec)
- Subsidiary Balance table (subb_table)
- Subsidiary Entry record (sube_rec)
- Subsidiary table (subs_table)
- Subsidiary Total record (subt_rec)
- Subsidiary Total table (subt_table)
- Subsidiary Transaction record (subtr_rec)

Required Tables and Records

The following records are required to run the features of the financial products.

- Amount Type table (atype_table)
- Document table (doc_table)
- Entry Type table (ent_table)
- Fiscal Calendar record (fsclcal_rec)
- Fund table (fund_table)
- Function table (func_table)
- General Ledger Definition record (gld_rec)
- Object table (obj_table)
- Subfund table (subfund_table)
- Subsidiary table (subs_table)
- Voucher table (vch_table)
- Financial Schemas

Introduction

Schema files define the structure of database files and associated fields in the CX data dictionary. You can access schema files associated with the financial products in the following directory path: $CARSPATH/schema/financial

File Naming Conventions

Jenzabar makes name distinctions in the naming of schemas. For schema files containing definitions of CX tables, the UNIX file name begins with the letter *t* followed by characters describing the table's English name (e.g., tst for the State table). For schema files containing definitions of CX records, the UNIX file name describes the record's English name (e.g., id for ID record).

The first line in a schema file, after revision information, specifies the INFORMIX database table that the schema defines. For example, *st_table* (State table) is specified in the *tst* schema file.

Field Descriptions

Schema files contain descriptions of each field defined in a table or record. You can view descriptions of fields in financial tables and records by accessing the schema files.

Financial Tables and Records

Introduction

The following list contains the tables and records that originate from the financial products. The list indicates each table/record's purpose, location and access information, and association with programs and other tables and records.

Note: The *Program interrelationships* in the list are included in the financial products.

The *Module/application interrelationships* in the list are not included in the financial products.

The tables and records appear in this section in alphabetical order by Informix filename.

ID Approval record

UNIX filename: appid

Informix filename: appid_rec

Schema location: $CARSPATH/schema/financial

Purpose: Serves as a temporary holding file for ID numbers of individuals who must approve a purchase order.

Program interrelationships: approve, *purch*

Approval table

UNIX filename: tappr

Informix filename: appr_table

Schema location: $CARSPATH/schema/financial

Purpose: Defines the approval hierarchy for approving purchase orders.

Program interrelationships: approve, purch

Budget Account record

UNIX filename: bgtacct

Informix filename: bgtacct_rec

Schema location: $CARSPATH/schema/financial

Purpose: Defines how to modify an existing budget (for a year and/or account) to create a new budget

Program interrelationships: bgtalloc, bgtbasis, bgtinstall

Budget Amount record

UNIX filename: bgtamt

Informix filename: bgtamt_rec

Schema location: $CARSPATH/schema/financial

Purpose: Contains budgeted amounts by general ledger account number.

Program interrelationships: bgtalloc, bgtbasis, bgtinstall

Budget Calendar record

UNIX filename: bgtcal

Informix filename: bgtcal_rec

Schema location: $CARSPATH/schema/financial

Purpose: Contains fiscal year and amount type combinations that can be accessed or updated for financial budgeting.

Program interrelationships: bgtalloc, bgtbasis

Budget Distribution table

UNIX filename: tbgtdist

Informix filename: bgtdist_table

Schema location: $CARSPATH/schema/financial

Purpose: Determines how to distribute yearly budget amounts into each fiscal period.

Program interrelationships: bgtalloc

Budget Summary record

UNIX filename: bgtsum

Informix filename: bgtsum_rec

Schema location: $CARSPATH/schema/financial

Purpose: Contains summarized budget information by blocks, groups and schedules.

Program interrelationships: bgtalloc

Cashier Entry table

UNIX filename: tcashent

Informix filename: cashent_table

Schema location: $CARSPATH/schema/financial

Purpose: Defines valid cashier entry types and associated defaults.

Program interrelationships: cashier

Cashier Reconciliation record

UNIX filename: chrecon

Informix filename: chrecon_rec

Schema location: $CARSPATH/schema/financial

Purpose: Tracks the reconciliation of individual cash journals.

Program interrelationships: cashier

Check Allocation record

UNIX filename: ckalloc

Informix filename: ckalloc_rec

Schema location: $CARSPATH/schema/financial

Purpose: Contains information used to determine the allocation of funds between accounts for a direct deposit check.

Program interrelationships: ckslct, dirdep

Check Group record

UNIX filename: ckgrp

Informix filename: ckgrp_rec

Schema location: $CARSPATH/schema/financial

Purpose: Contains information used to control the selection and printing of checks.

Program interrelationships: ckabort, ckpost, dirdep

Check Request record

UNIX filename: ckreq

Informix filename: ckreq_rec

Schema location: $CARSPATH/schema/financial

Purpose: Stores information about checks to be printed.

Program interrelationships: ckabort, ckpost, dirdep

Depreciation Association record

UNIX filename: depas

Informix filename: depas_rec

Schema location: $CARSPATH/schema/financial

Purpose: Defines accumulated depreciation accounts and associates these accounts with depreciation expense accounts

Program interrelationships: fixpost

1099 record

UNIX filename: f1099

Informix filename: f1099_rec

Schema location: $CARSPATH/schema/financial

Purpose: Records information about 1099 reporting.

Program interrelationships: f1099bld, f1099form, f1099tape, r1099bld, r1099tape

1099 table

UNIX filename: t1099

Informix filename: f1099_table

Schema location: $CARSPATH/schema/financial

Purpose: Defines valid 1099 form types.

Program interrelationships: f1099bld, f1099form, f1099tape, r1099bld, r1099tape

990 Report Sort record

UNIX filename: f990rpt

Informix filename: f990rpt_rec

Schema location: $CARSPATH/schema/financial

Purpose: Stores the ID number and sort key for some reports.

Fixed Asset record

UNIX filename: fix

Informix filename: fix_rec

Schema location: $CARSPATH/schema/financial

Purpose: Contains information about fixed assets, including location, depreciation details and authorization.

Program interrelationships: fixpost

Fixed Asset table

UNIX filename: tfix

Informix filename: fix_table

Schema location: $CARSPATH/schema/financial

Purpose: Defines the different types of fixed assets.

Program interrelationships: fixpost

Table/record interrelationships: fix_rec

Fixed Asset History record

UNIX filename: fixhist

Informix filename: fixhist_rec

Schema location: $CARSPATH/schema/financial

Purpose: Tracks maintenance and other historical information about fixed assets.

Program interrelationships: fixpost

Table/record interrelationships: fix_rec

Fixed Asset Maintenance record

UNIX filename: fixmaint

Informix filename: fixmaint_rec

Schema location: $CARSPATH/schema/financial

Purpose: Tracks maintenance information about fixed assets.

Program interrelationships: fixpost

Table/record interrelationships: fix_rec

Fixed Asset Skeleton record

UNIX filename: fixskel

Informix filename: fixskel_rec

Schema location: $CARSPATH/schema/financial

Purpose: Automatically records fixed assets acquired through Accounts Payable.

Program interrelationships: acctspay, fixpost

Table/record interrelationships: fix_rec, fix_table

Credit Card Entry record

UNIX filename: glecred

Informix filename: glecred_rec

Schema location: $CARSPATH/schema/financial

Purpose: Maintains information associated with the credit card payment form.

Program interrelationships: cashier

Group History record

UNIX filename: grphist

Informix filename: grphist_rec

Schema location: $CARSPATH/schema/financial

Purpose: Contains history information about grouping sheet runs.

Program interrelationships: grpselect

Group Request record

UNIX filename: grpreq

Informix filename: grpreq_rec

Schema location: $CARSPATH/schema/financial

Purpose: Stores information about groups of checks to be printed.

Program interrelationships: grpselect

Payment Form table

UNIX filename: payfrm

Informix filename: payfrm_table

Schema location: $CARSPATH/schema/financial

Purpose: Defines valid forms of payment codes.

Program interrelationships: cashier

Payment Terms table

UNIX filename: payterm

Informix filename: payterm_table

Schema location: $CARSPATH/schema/financial

Purpose: Defines valid payment terms and gives valid discounts for each term.

Program interrelationships: cashier, ckabort, ckpost, ckslct, purch

Product interrelationships: Purchasing and Accounts Payable, Student Billing

Budget Parameter record

UNIX filename: pbgt

Informix filename: pbgt_rec

Schema location: $CARSPATH/schema/financial

Purpose: Contains parameters used by the Budget Allocation program.

Program interrelationships: bgtalloc

Purchase Order record

UNIX filename: po

Informix filename: po_rec

Schema location: $CARSPATH/schema/financial

Purpose: Contains information relating to purchase orders, including amounts, dates and vendors.

Program interrelationships: approve, ckslct, purch, purchaudit

Product interrelationships: Purchasing and Accounts Payable

Purchase Order Body record

UNIX filename: pobody

Informix filename: pobody_rec

Schema location: $CARSPATH/schema/financial

Purpose: Contains the information that is in the body of the purchase order.

Program interrelationships: approve, purch

Statement Parameter record

UNIX filename: precon

Informix filename: precon_rec

Schema location: $CARSPATH/schema/financial

Purpose: Provides processing parameters for the check reconciliation process.

Program interrelationships: ckrecon

1099R record

UNIX filename: r1099

Informix filename: r1099_rec

Schema location: $CARSPATH/schema/financial

Purpose: Contains the fields necessary to process 1099R earnings statements.

Program interrelationships: r1099bld, r1099form

Reconciliation record

UNIX filename: recon

Informix filename: recon_rec

Schema location: $CARSPATH/schema/financial

Purpose: Contains record of the data in the accounting system at the time of reconciliation, preventing normal system activity from affecting the reconciliation process.

Program interrelationships: ckrecon

Subsidiary Entry Join record

UNIX filename: sube2

Informix filename: sube2_rec

Schema location: $CARSPATH/schema/financial

Purpose: Enables scripts to look up selected financial records by joining between a sube2 key and the corresponding sube key.

Type table

UNIX filename: ttyp

Informix filename: typ_table

Schema location: $CARSPATH/schema/financial

Program interrelationships: grpselect

Vendor Quality table

UNIX filename: tvenqual

Informix filename: venqual_table

Schema location: $CARSPATH/schema/financial

Purpose: Defines specialties or particular characteristics of vendors.

Program interrelationships: purch, *vndentry*

Vendor Type table

UNIX filename: tventype

Informix filename: ventype_table

Schema location: $CARSPATH/schema/financial

Purpose: Defines the contents of the social security number field for vendor records (e.g., EIN or SSN).

Product interrelationships: purch, *vndentry*

Vendor File Comments record

UNIX filename: bvnd

Informix filename: vnd_blob

Schema location: $CARSPATH/schema/financial

Purpose: Provides the free form text comments for the vendor file.

Program interrelationships: f1099tape, grpselect, r1099tape, vndentry

Vendor record

UNIX filename: vnd

Informix filename: vnd_rec

Schema location: $CARSPATH/schema/financial

Purpose: Contains information about vendors, including ID, contact name, and delivery terms.

Program interrelationships: f1099tape, grpselect, r1099tape, vndentry

SECTION 4 - MACROS, INCLUDES, AND CONFIGURATION TABLE ENTRIES

Overview

Introduction

This section provides you with reference information about macros, includes, and Configuration table entries used to set up the CX financial components.

The Relationship among Macros, Includes, Configuration Table Entries and C Programs

An m4 macro cannot be used directly in a C program since the system does not process C program code through the m4 processor. Therefore, CX uses includes so that a C program can communicate and process a macro. An include statement in an include file contains the information for defining a macro using syntax that a C program understands. C programs read and understand include files.

Some CX programs use Configuration table entries in place of macros. Configuration table entries can enable features, set up options, and define valid and default values. Unlike macros, you can update or add Configuration table entries through normal table maintenance.

General Installation Procedures

See *Jenzabar CX Technical Manual* for general procedures on setting and installing changes to macros and includes.

For More Information

Most of the macros that affect processing in the financial products also affect the General Ledger application. Information about the macros that relate specifically to financial products appears in this section. For more information about the other macros, see *General Ledger Technical Manual*.

Steps for Modifying Macros

When you begin your implementation of the financial products, use the following process to change macro values.

1. Review the macro file (macros/custom/financial), searching for the macros described in this section. Identify the macros for which you want to change the values.
2. Check out the macro file, and make the desired changes.
3. Use the tinstall command to temporarily install the macro file.

   Note: Check the file in when you are certain that the changes you have made are correct.
4. Access the directory in which the include file (include/custom) resides, and reinstall the include file.
5. Access the directory in which the source (src) programs reside, and reinstall all the program files.
6. Set up the tables that impact the product.
7. Test the setup by entering test information and verifying the output.
8. If the results are correct, check in the macro file. If the results are incorrect, use the make uncheckout command to cause the system to ignore your previous modifications to the macro file, then repeat phases 2-7.

Financial Macros

Introduction

CX uses macros to define specific values used throughout the financial products. The macros and includes enable you to change the available options and functionality of the products without having to modify C code. By modifying macros, you can customize your implementation of the products, making them easier to maintain.

Definition and Function

A macro is an instruction that causes the execution of a pre-defined sequence of instructions in the same source language. A macro consists of uppercase letters and underscores, and is used in place of a text string within source files. CX expands the macro to the longer text during the installation process for a file. CX uses the following kinds of macros:

- Enables - allows you to enable a feature of the CX
- DBS_COMMON - allows you to define database values in screens
- Periodic - allows you to make changes on a periodic basis

Macros can perform one of the following functions:

- Define defaults on a screen (_DEF)
- Define valid values in a field (_VALID or _INCL)
- Enable system modules (ENABLE_MOD)
- Enable system features (ENABLE_FEAT)
- Establish a valid value for an include

Applocate Program

You can also locate macros using the *applocate* program. *Applocate* checks the descriptions of macro files for the product area you specify, and lists each file that it locates in a file.

Note: To locate the macros used in financial products, using *applocate*, you must specify the module's name.

The following procedure lists the steps to run the *applocate* program.

1. Select Utilities from the CX System menu. The Utilities: Main menu appears.
2. Select File Options. The Utilities: File Options menu appears.
3. Select Locate Macro Values. The Locate Macro Values screen appears.
4. Select **Table Lookup** in the Category field. A list of module names appears in a table lookup box.
5. Select a module name (e.g., FINANCIAL). The table lookup box disappears.
6. Select **Finish**. The Output Parameters window appears.
7. Do the following:
   - In the Time field, enter **NOW**.
   - In the Background field, enter **Y**.
   - Select **Finish**.

   The system places the file, *applocate.out,* in your home directory.

Macro File Locations

Macros that affect the processing of the financial products appear in the following files:

- $CARSPATH/macros/custom/financial
- $CARSPATH/macros/custom/periodic
- $CARSPATH/macros/custom/table

Common macros that also can influence the processing of the financial products appear in $CARSPATH/macros/custom/common.

For More Information

For more information about *General Ledger Technical Manual* contains detailed information about the macros that influence processing for all financial products.

Enable Macros

The following lists the financial enable macros, located in the *financial* macro file. The macros appear in this list in the order in which they appear in the macro file.

m4_define(`ENABLE_FEAT_FWS',`N')
Defines whether or not to implement Federal Work Study processing. If the institution defines the macro value to Y, the Federal Work Study processes general mail reporting for those students who are near or have exceeded their work study awards. The system posts amounts that exceed the award to an overage account. The macro's default setting is N.

m4_define('ENABLE_GRNT_RPT','N')
Defines whether or not the grant reports and tables appear on the menu. The macro's default setting is N.

m4_define('ENABLE_MULTI_AP_SUBS','Y')
Defines whether or not the Vendor Entry menu options display two payables subsidiaries. If the institution accepts the default setting of Y, then the programs prompt the user to enter a subsidiary.

Note: This macro relates to the following two macros:
- SUBS_AP_DEF
- SUBS_AP_TWO_DEF

m4_define('ENABLE_MULTI_AP_BALS','Y')
Defines whether or not the accounts payable programs use a single balance code.. If the institution accepts the default setting of Y, then the programs prompt the user to enter a balance code.

Note: This macro relates to the macro SUBS_PRD_AP_VALID.

m4_define('ENABLE_GRP_SHEETS','N')
Defines whether or not the grouping sheets and schedules required of state institutions appear on the menu. The default setting is N.

m4_define('ENABLE_MULTI_FORM_PO','Y')
Defines whether or not the accounts payable programs use a single purchase order form. If the institution accepts the default setting of Y, then the programs prompt the user to enter a purchase order form type.

Note: This macro relates to the macro FORM_PO_VALID.

m4_define('ENABLE_MULTI_DOC_PO','Y')
Defines whether or not the accounts payable programs use a single document code for purchase orders. If the institution accepts the default setting of Y, then the programs prompt the user to enter a purchase order document code.

Note: This macro relates to the macro DOC_PO_VALID.

m4_define('ENABLE_FEAT_PO_APPROVAL','Y')
Defines whether or not purchase order approval options appear on the menu.

m4_define('ENABLE_FEAT_GAIN_LOSS','Y')
Defines whether or not the institution uses gain/loss accounts when disposing of fixed assets. If the institution accepts the default setting of Y, then the *fixpost* program computes and posts gains or losses on disposal to the gain/loss account. If the institution sets the value to N, *fixpost* will not compute gains or losses, and the fixed asset source account (i.e., the account you credited when you acquired the asset) will be used for disposal entries. The system locates the source account in the Fixed Asset record (fix_rec).

Note: This macro, if enabled, relates to the macro FIX_GAIN_DEFINE.

m4_define('ENABLE_FEAT_HALF_MONTH','Y')
Defines how the *fixpost* program computes depreciation in the first month of ownership of a fixed asset. If the institution accepts the default setting of Y, fixpost computes and posts one-half month's depreciation in the first month of ownership. If the institution sets the value to N, fixpost computes and posts a full month's depreciation in the first month of ownership.

Note: This macro applies to the straight-line method of depreciation only.

m4_define('ENABLE_SEPARATE_ACCTS_PAYABLE','N')
Defines whether or not the accounts payable and purchasing functions appear on separate menus. If the institution accepts the default of N, users will be able to access both accounts payable and purchasing functions from the same menus.

Definition Macros

The definition macros define some commonly used valid values. The following lists the definition macros in the $CARSPATH/macros/custom/financial macro file.

m4_define(`CH_RECON_ENTTYPE_DEF',`RECN')
m4_define(`CH_CLOSE_ENTTYPE_DEF',`CLOS')
m4_define(`CH_RECON_PAYFRM_DEF',`DC')
m4_define(`CH_CLOSE_PAYFRM_DEF',`DC')
m4_define('CH_PAYFRM_DEF','CA')
Defines the valid codes for *cashier* reconciliation and closing entry types and payment forms. The values defined for these macros must also be valid in the Entry and Payment Form tables.

```
m4_define('CH_OS_DEFINE','#define CH_OS_ACT_DEFINE(a)    Glacct a = { \
        "10",       \
        "  ",       \
        "2020",     \
        "  ",       \
        };')
```

Defines the account number that *cashier* uses when a cash drawer is over or short.

m4_define('BGT_AXIS_DEF','FUNC')
m4_define('BGT_AXIS_VALID','OBJ,FUNC')
m4_define('BGT_AXIS_INCL','include=(BGT_AXIS_VALID),upshift')
m4_define(BGT_AXIS_EX', ' (OBJ) or (FUNC) .')
Defines the financial statement axis types used in *bgtalloc*.

m4_define(‘BGT_TYPE1_DEF’, ‘ACT’)
m4_define(‘BGT_YR1_DEF’, FS_YR_PREV)
m4_define(‘BGT_TYPE2_DEF’, ATYPE_BGT_DEF)
m4_define(‘BGT_YR2_DEF’, FS_YR_CUR)
m4_define(‘BGT_TYPE3_DEF’, ‘ACT’)
m4_define(‘BGT_YR3_DEF’, FS_YR_CUR)
m4_define(‘BGT_TYPE4_DEF’, ‘REQ’)
m4_define(‘BGT_YR4_DEF’, FS_YR_NEXT)

Define reporting defaults for Financial Budgeting.

m4_define(‘DOC_APCK_DEF’,’AP’)
m4_define(‘DOC_APCK_VALID’,’AP’)
m4_define(‘DOC_APCK_INCL’,’include=(DOC_APCK_VALID),upshift’)
m4_define(‘DOC_APCK_EG’, ‘, eg: DOC_APCK_DEF.’)
m4_define(‘DOC_GRP_DEF’,’GP’)

Define document codes for accounts payable check writing.

```
m4_define('AP_DISC_ACCT_DEFINE','#define REV_DEFINE(a)    Glacct a = { \
        "10",       \
        "  ",       \
        "5207",     \
        "  ",       \
        };')
```

Defines the account number to which accounts payable charges discounts.

m4_define(‘DOC_CR_DEF’,’CR’)
m4_define(‘DOC_CR_VALID’,’CR’)
m4_define(‘DOC_CR_INCL’,’include=(DOC_CR_VALID) ,upshift’)
m4_define(‘DOC_CR_EG’,’, eg: DOC_CR_DEF.’)

Defines the document codes for cash receipts.

m4_define(‘/DOC_PO_DEF’,’PO’)
m4_define(‘DOC_PO_VALID’,’PO’)
m4_define(‘DOC_PO_INCL’,’include=(DOC_PO_VALID) ,upshift’)
m4_define(‘DOC_PO_EG’,’, eg: DOC_PO_DEF.’)

Defines the document codes for purchase orders.

m4_define(‘DOC_RFCK_DEF’,’RF’)
m4_define(‘DOC_RFCK_VALID’,’RF’)
m4_define(‘DOC_RFCK_INCL’,’include=(DOC_RFCK_VALID) ,upshift’)
m4_define(‘DOC_RFCK_EG’,’, eg: DOC_RFCK_DEF.’)

Defines the document codes for refunds issued.

m4_define(‘CKGRP_FORM_DEF’,’apcheck’)

Defines the valid form for check groups.

m4_define(‘FORM_APCK_DEF’,’apcheck’)
m4_define(‘FORM_APCK_VALID’,’apcheck’)
m4_define(‘FORM_APCK_INCL’,’include=(FORM_APCK_VALID) ,downshift’)
m4_define(‘FORM_APCK_EX’,’ ‘(FORM_APCK_DEF).’)

Defines the document codes for accounts payable checks.

m4_define('FORM_RFCK_DEF','rfcheck')
m4_define('FORM_RFCK_VALID','rfcheck')
m4_define('FORM_RFCK_INCL','include=(FORM_RFCK_VALID) ,downshift')
m4_define('FORM_RFCK_EX',' '(FORM_RFCK_DEF).')

Defines the document codes for refund checks.

m4_define('FORM_STMT_DEF','stmt2')
m4_define('FORM_STMT_VALID','stmt1, stmt2, stmt3, stmt4')
m4_define('FORM_STMT_INCL','include=(FORM_STMT_VALID) ,downshift')
m4_define('FORM_STMT_EX',' '(FORM_STMT_DEF).')
m4_define('FORM_STMT_EG', ', eg: FORM_STMT_DEF.')

Defines the form type for accounts receivable statements.

m4_define('CKGRP_STAT_DEF','S')
m4_define('CKGRP_FRM_DEF','A')

Defines the valid check group status and form type.

m4_define('FORM_PO_DEF','rfcheck')
m4_define('FORM_PO_VALID','rfcheck')
m4_define('FORM_PO_INCL','include=(FORM_PO_VALID) ')
m4_define('FORM_PO_EX',' '(FORM_PO_DEF).')

Defines the document codes for refund checks.

m4_define('FIX_DEP_METH_DEF','SL')
m4_define('FIX_DEP_METH_VALID','SL, D200, D150, D125, AC3, AC5, AC10, AC15, AC18, AC19, MA3, MA5, MA7, MA10, MA15, MA20, MANP, MARP')
m4_define('FIX_DEP_METH_INCL','include=(FIX_DEP_METH_VALID),upshift')

Defines the valid types of depreciation codes.

```
m4_define('FIX_GAIN_DEFINE','#define GAIN_DEFINE(a)    Glacct a = { \
        "66",       \
        "  ",       \
        "4400",     \
        "  ",       \
        };')
```

Defines the account number to which *fixpost* will charge gains on disposal of fixed assets.

```
m4_define('FIX_SOURCE_DEFINE','#define SOURCE_DEFINE(a)    Glacct a = { \
        "66",       \
        "  ",       \
        "3901",     \
        "  ",       \
        };')
```

Defines the account number to which *fixpost* will charge purchases of fixed assets.

m4_define('FIX_POST_FILE','fixpost.out')

Defines the name of the fixed asset posting file.

m4_define('FIX_TYPE_EG', ', eg: FIX_TYPE_DEF.')

Defines the valid types of fixed assets.

m4_define('FIN_STAT_SITE_CODE', '12345')

Defines the code used to represent the site code for state reporting.

m4_define('SUBS_PRD_EG', ', eg: SUBS_PRD_DEF.')

Defines the valid processing periods for subsidiaries, in conjunction with the values in the file $CARSPATH/macros/custom/periodic.

m4_define('SUBS_PRD_AP_DEF','INV')
m4_define('SUBS_PRD_AP_VALID','INV')
m4_define('SUBS_PRD_AP_INCL','include=(SUBS-PRD_AP_VALID),upshift')
m4_define('SUBS_PRD_AP_EG', ', eg: SUBS_PRD_AP_DEF.')

Defines the valid types of depreciation codes.

m4_define('PAY_MODE_INCL','include=(B,C,D,null), upshift')
m4_define('PAY_MODE_EX', '(B)ill, (C)heck, (D)irect deposit.')

Defines the valid payment modes.

m4_define('RCPT_RUNCODE_DEF', 'CASHIER')
m4_define('RCPT_FORM_DEF', 'cash_rcpt')

Defines the default receipt codes to be used within *cashier*.

m4_define('SUBS_AP_DEF','A/P')
m4_define('SUBS_AP_TWO_DEF', 'PIP')
m4_define('SUBS_AP_VALID','"A/P", "PIP", "PIPC"')
m4_define('SUBS_AP_INCL','include=(SUBS_AP_VALID), upshift')
m4_define('SUBS_AP_EG', ', eg: SUBS_AP_DEF.')

Defines the valid subsidiary codes used in accounts payable. The primary code is SUBS_AP_DEF, and the secondary code is SUBS_AP_TWO_DEF.

m4_define('SUBS_AR_DEF','F/S')
m4_define('SUBS_AR_VALID','"ADV", "F/S", "O/F", "R/D","R/H"')
m4_define('SUBS_AR_INCL','include=(SUBS_AR_VALID), upshift')
m4_define('SUBS_AR_EG', ', eg: SUBS_AR_DEF.')

Defines the valid subsidiary codes used in accounts receivable.

m4_define('VCH_AP_DEF','AP')
m4_define('VCH_APCK_DEF','CK')
m4_define('VCH_AP_VALID', 'AC,AP,CK,QC')
m4_define('VCH_AP_INCL','include=(VCH_AP_VALID), upshift')
m4_define('VCH_AP_EG', ', eg: AP, CK, QC.')

Defines the valid journals for invoicing and writing non-payroll checks.

m4_define('VCH_AR_DEF','AR')
m4_define('VCH_AR_VALID', 'AR,SB')
m4_define('VCH_AR_INCL','include=(VCH_AR_VALID), upshift')
m4_define('VCH_AR_EG', ', eg: VCH_AR_DEF.')

Defines the valid journals for non-student receivables.

m4_define('VCH_PC_DEF','PC')
m4_define('VCH_PC_VALID', 'AC,AP,PC')
m4_define('VCH_PC_INCL','include=(VCH_PC_VALID), upshift')
m4_define('VCH_PC_EG', ', eg: VCH_PC_DEF.')

Defines the valid journals for purchasing without check writing.

m4_define('VCH_SA_DEF','PR')
m4_define('VCH_SA_VALID', 'SA,SB,AC')
m4_define('VCH_SA_INCL','include=(VCH_SA_VALID), upshift')
m4_define('VCH_SA_EG', ', eg: SA, SB.')

Defines the valid journals for student accounts.

m4_define('CASHIER_HOLD_ACT','CASHIER')

Defines the hold action in *cashier*.

m4_define(‘DIRDEP_INST_BANK_ACCT_NO,’ ’)
m4_define(‘DIRDEP_BANK_DEST_CODE,’ ’)
m4_define(‘DIRDEP_ORG_BANK_CODE,’ ’)
m4_define(‘DIRDEP_REF_CODE,’ ’)

Define the values that *ddtp* inserts into the tape you send to your bank for direct deposits. The *ddtp* process uses the values you define for these macros as parameters during processing.

Fixed Assets Macros

The macros that customize *fixpost* are as follows:

CAUTION: After you modify the macros that pertain to Fixed Assets, you must reinstall the following files:

- $CARSPATH/include/custom/fixpost
- $CARSPATH/src/fixassets

ENABLE_FEAT_GAIN_LOSS

Controls the use of gain/loss accounts when the institution disposes of fixed assets.

- If you set this macro to Y, the *fixpost* program will compute and post gains and losses on disposal of assets to the gain/loss account.
- If you set this macro to N, the *fixpost* program will not compute gains or losses, and the fixed asset source account (i.e., the account you credited when you acquired the asset) will be used for disposal entries. The system locates the source account in the fix_rec.

ENABLE_FEAT_HALF_MONTH

Controls the use of half-month depreciation.

- If you set this macro to Y, the *fixpost* program will compute and post one-half month’s depreciation in the first month of ownership.
- If you set this macro to N, the *fixpost* program will compute and post a full month’s depreciation in the first month of ownership.

Note: This macro applies to the straight-line method of depreciation only.

FIX_DEP_METH_VALID

Defines the valid depreciation methods for the Fixed Asset product. The codes are as follows:

- STRAIGHT_LINSL
- DECLINE_20D200
- DECLINE_15D150
- DECLINE_12D125
- ACRS_ AC3
- ACRS_ AC5
- ACRS_1 AC10
- ACRS_1 AC15
- ACRS_1 AC18
- ACRS_1 AC19
- MACRS_MA3
- MACRS_MA5
- MACRS_MA7
- MACRS_1MA10
- MACRS_1MA15
- MACRS_2MA20
- MACRS_REAMANP
- MACRS_RENTAMARP

CAUTION: These values must appear in the fix_table exactly as they appear here (and in the macro file) in order for *fixpost* program to compute depreciation correctly. You do not need to change any of the values in this file.

FIX_GAIN_DEFINE
Defines the fund/function/object/subfund that you want to use for gains or losses on disposal of fixed assets.

FIX_SOURCE_DEFINE
Defines the fund/function/object/subfund that you want to use as the default source account, if you do not define a source account in the Fixed Asset record.

FIX_POST_FILE
Defines the name of the file to which you want to route the results of verifying and posting Fixed Assets transactions. The program sends the file to the user's home directory, with the default file name of *fixpost.out*.

Cashier Macros

The following list describes the setting of m4 macros used by *cashier*. Update the values of these macros to meet the requirements of your institution.

CAUTION: How you change the following macros depends on the current setting of the macro in the standard CX product.

BILL_PARAM_VALID
The macro is located in the following directory path: $CARSPATH/macros/custom/periodic.

CH_CLOSE_ENTTYPE_DEF
The value of the macro must exist as an entry type in the Entry Type table. The macro is located in the $CARSPATH/macros/custom/financial directory path.

CH_CLOSE_PAYFRM_DEF
The value of this macro must exist as a payment form type in the Payment Form table. The macro is located in the $CARSPATH/macros/custom/financial directory path.

CH_RECON_ENTTYPE_DEF
The value of this macro must exist as an entry type in the Entry Type table. The macro is located in the $CARSPATH/macros/custom/financial directory path.

CH_RECON_PAYFRM_DEF
The value of this macro must exist as a payment form type in the Payment Form table. The macro is located in the $CARSPATH/macros/custom/financial directory path.

DOC_CR_DEF
The value of this macro defines the default Cashier document code. The macro is located in the $CARSPATH/macros/custom/financial directory path.

ENABLE_FEAT_PAY_PLAN
Do you want payment plan information to be calculated for statements?

- If yes, define this macro to **Y** to calculate plan information for statements.
- If no, define this macro to **N** to not calculate plan information for statements.

The macro is located in the $CARSPATH/macros/custom/financial directory path.

ENABLE_MIN_PAY_LOGIC

Do you want minimum payment information to be calculated for statements?

- If yes, define this macro to **Y** to calculate minimum payment information for statements.
- If no, define this macro to **N** to not calculate minimum payment information for statements.

The macro is located in the $CARSPATH/macros/custom/financial directory path.

ENABLE_STMT_DISP_PEND_AID

Do you want pending financial aid to appear on statements?

- If yes, define this macro to **Y** to display pending aid on statements
- If no, define this macro to **N** to not allow pending financial aid to appear on statements.

The macro is located in the $CARSPATH/macros/custom/financial directory path.

FORM_STMT_DEF

The value of this macro is the default statement form (stmt2) used on the statement parameter perform screen. The macro is located in the $CARSPATH/macros/custom/financial directory path.

FORM_STMT_VALID

Statement form names included in this macro must exist in the $CARSPATH/modules/accounting/forms/stmt directory for the ability to print statements. The macro is located in the $CARSPATH/macros/custom/financial directory path.

1099 Macros

Two macro files, $CARSPATH/macros/custom/f1099 and $CARSPATH/macros/custom/r1099, are required for 1099 processing. Because maintenance of these macros is a routine annual task, information about these macros resides in the section of this document titled *Financial Maintenance Procedures*.

Before Setting C Program Macros

Before you start to set C program macros, you must install all macro files that you have modified. You must install the macros to successfully set the C program macros.

C Program Macros

The C program macros used by *cashier* are defined in the following include files:

- $CARSPATH/include/custom/cashier
- $CARSPATH/include/custom/libbill

Customize the value of the macros to meet the institution's specific requirements.

The following list describes the C program macros:

BEGIN_ENTRY_ FIELD

This macro defines the first field in which *cashier* prompts for information. You must define this macro with the name of an attribute from the $CARSPATH/src/accounting/cashier/SCR/entry screen definition field. Possible values for this macro are the following:

- 'ctyp' for cash transaction type
- 'idno' for ID number

Syntax:

```
#define BEGIN_ENTRY_FIELD   "ctyp"
```

-- or --

```
#define BEGIN_ENTRY_FIELD   "idno"
```

CC_LINE1, CC_LINE2, CC_LINE3, DS_LINE1, DS_LINE2, DS_LINE3, RC_LINE1, RC_LINE2, RC_LINE2, TR_LINE1, TR_LINE2, TR_LINE3

These macros (four sets of three macros) define the field descriptions that appear next to the three amounts on the main *cashier* program entry screen. Each set of macros corresponds to one of the four classifications of cash transactions, including the following:

- CC_LINE* for Check Cashing transactions
- DS_LINE* for Disbursement transactions
- RC_LINE* for Receipt transactions
- TR_LINE* for Transfer transactions

CAUTION: You must define these macros. Their values cannot exceed 14 characters in length.

Syntax:

```
#define CC_LINE1    "Check Amount"
#define CC_LINE2    "              "
#define CC_LINE3    "              "

#define DS_LINE1    "Disburse"
#define DS_LINE2    "              "
#define DS_LINE3    "              "

#define RC_LINE1    "Remittance"
#define RC_LINE2    "Total Applied"
#define RC_LINE3    "Change"

#define TR_LINE1    "Transfer"
#define TR_LINE2    "              "
#define TR_LINE3    "              "
```

CH_CLOSE_ ENTDESC, CH_RECON_ ENTDESC

These macros define the entry description used for the reconciliation and closing entries.

CAUTION: You must define these macros. Their values cannot exceed 24 characters in length.

Syntax:

```
#define CH_RECON_ENTDESC  "Drawer Reconciliation"
#define CH_CLOSE_ENTDESC  "Drawer Closing"
```

CH_CLOSE_ ENTTYPE, CH_RECON_ ENTTYPE

These macros define the entry code used for the reconciliation and closing entries.

CAUTION: You must define these macros, and you should not change their values.

Note: The values of these macros are equal to the values of the corresponding m4 macros, CH_RECON_ENTTYPE_DEF and CH_CLOSE_ENTTYPE_DEF, that you defined previously.

Syntax:

```
#define CH_RECON_ENTTYPE "CH_RECON_ENTTYPE_DEF"
#define CH_CLOSE_ENTTYPE "CH_CLOSE_ENTTYPE_DEF"
```

CH_CLOSE_PAYFRM, CH_RECON_ PAYFRM

These macros define the payment form code used for the reconciliation and closing entries.

CAUTION: You must define these macros, and you should not change their values.

Note: The values of these macros are equal to the values of the corresponding 4 macros, CH_RECON_ PAYFRM and CH_CLOSE_PAYFRM, that you defined previously.

Syntax:

```
#define CH_RECON_PAYFRM  "CH_RECON_PAYFRM_DEF"
#define CH_CLOSE_PAYFRM  "CH_CLOSE_PAYFRM_DEF"
```

CH_OS_FUND, CH_OS_CNTR, CH_OS_ACCT, CH_OS_PROJ

These macros define the General Ledger account used as the overage/shortage account for overages/shortages that occur as a result of reconciliation.

CAUTION: You must define these macros with values that define a valid general ledger account.

Syntax:

```
#define CH_OS_FUND   "10"
#define CH_OS_CNTR   "    "
#define CH_OS_ACCT   "2020"
#define CH_OS_PROJ   "    "
```

DEF_PREV_TEXT

This macro describes the previous balance due on the student statement. The description appears next to the previous statement balance amount on the student's statement.

CAUTION: You must define this macro. Its value cannot exceed 24 characters in length.

Syntax:

```
#define DEF_PREV_TEXT  "Previous Statement Balance"
```

DEF_RECPT_ RUNCODE

This macro specifies the default ADR run code for formatting addresses printed on cash receipts.

CAUTION: You must define this macro. Its value cannot exceed 8 characters in length.

Syntax:

```
#define DEF_RECPT_RUNCODE  "CASHIER"
```

-- or --

```
#define DEF_RECPT_RUNCODE  "SINGLEI"
```

DSPL_CLOSE_OPT

This macro determines whether or not the Close option displays on the drawer menu.
Do you want the Close option to appear on the drawer menu?

- If yes, define this macro.
- If no, do not define this macro.

Syntax:

```
Enabled:  #define DSPL_CLOSE_OPT
Disabled: /* #define DSPL_CLOSE_OPT */
```

DSPL_NONPOST_ AID_PEND

If the expected financial aid exceeds the actual financial aid received, this macro determines whether or not the difference appears on the student's statement.

Do you want the difference in expected and actual financial aid to appear on the student's statement?

- If yes, define this macro and the difference between expected and actual financial aid received appears on the student's statement as Pending Financial Aid.
- If no, do not define this macro and the difference between expected and actual financial aid does not appear on the student's statement.

Note: This macro only affects types of financial aid not posted directly to the student's account by the financial aid office (i.e., taid_disb_to_bill = N).

Syntax:

```
Enabled:  #define DSPL_NONPOST_AID_PEND
Disabled: /* #define DSPL_NONPOST_AID_PEND */
```

PRINT_CLOSE_ RECPT, PRINT_RECON_ RECPT

These macros determine whether or not the Cashier program increments the cash receipt number for closing and reconciliation entries, respectively.

Do you want the Cashier program to increment the cash receipt number for closing and reconciliation entries?

- If yes, define these macros; you can then print a cash receipt for closing and reconciliation entries and increment the cash receipt number.
- If no, do not define these macros and you cannot print a cash receipt for closing and reconciliation entries, and the cashier program does not increment the cash receipt number.

Note: If desired, you can define one of these macros and comment out the other.

Syntax:

```
Enabled:  #define PRINT_CLOSE_RECPT
          #define PRINT_RECON_RECPT
Disabled: /* #define PRINT_CLOSE_RECPT
          #define PRINT_RECON_RECPT */
```

TOTAL_DUE_DESC

This macro describes the total balance due. The description appears next to the total balance due amount on the main Cashier entry screen.

CAUTION: You must define this macro. Its value cannot exceed 14 characters in length.

Syntax:

```
#define TOTAL_DUE_DESC "Balance Due"
-- or --
#define TOTAL_DUE_DESC "Total Balance"
```

VER_RECPT_BAL

This macro determines whether or not the cash receipt operation allows you to control the printing of account balance(s) on the cash receipt.

Do you want to control the printing of account balance(s) on the cash receipt?

- If yes, define this macro; you then have the option to specify at run-time whether or not the student's account balance is printed on the cash receipt.
- If no, do not define this macro, and the cash receipt prints as it appears in the form definition file.

Syntax:

```
Enabled:   #define VER_RECPT_BAL
Disabled:  /* #define VER_RECPT_BAL */
```

Budgeting Macros

The following list describes the setting of m4 macros used by the CX budgeting programs. Update the values of these macros to meet the requirements of your institution.

ENABLE_MANUAL_BGT
This macro determines whether you want to enable users to manually enter budget information using a BG journal. If you set this macro to N, the option to enter budget information manually will not appear on the menu, and users will be required to enter budget information through the Budget module only.

ATYPE_BGT_VALID
ATYPE_BGT_INCL
ATYPE_BGT_DEF
ATYPE_BGT_EG
These macros define the amount types that are valid within the budgeting programs.

BGT_AXIS_VALID
BGT_AXIS_INCL
BGT_AXIS_DEF
BGT_AXIS_EG
These macros define the financial statement axis codes that are valid for use within *bgtalloc*.

BGT_TYPE1_DEF
BGT_YR1_DEF
BGT_TYPE2_DEF
BGT_YR2_DEF
BGT_TYPE3_DEF
BGT_YR3_DEF
BGT_TYPE4_DEF
BGT_YR4_DEF
These macros define the columns that display on Budget reports.

ENABLE_TRM_BGT
TRM1_NAME
TRM2_NAME
TRM3_NAME
TRM1_END
TRM2_END
TRM3_END
TRM_BGT_PRDS
These macros enable the use of trimester budgeting and define the names and periods included in each trimester.

Financial Includes

Introduction

The financial products use includes that determine the features that are enabled. An include can either be a compile option that enables or disables a feature, or a default value include that defines a default value for a feature.

To enable a feature in the financial products, you must define an include in $CARSPATH/include/common. To disable an include, comment out the include in the same file. See *Jenzabar CX Technical Manual* for more information on enabling and disabling includes. By modifying includes, you can customize your implementation of the financial products, and make the module easier to maintain.

Purpose

An include allows you to activate or deactivate features in C programs without changing the C code.

Macro Dependency

Includes have a dependency on macros. Normally, you do not directly modify includes for the module. You must modify a corresponding macro value and then reinstall the include.

Includes in the cashier Program

The *cashier* program uses include files to control some processing features. The includes are located in the $CARSPATH/include/custom/cashier file.

Refer to comments in the include file to customize these features:

- Overriding the printing of account balances on receipts.
- Using separate closing and reconciliation options.
- Using numbered cash receipts during closing and reconciliation.
- Setting the initial field on the main *cashier* entry screen.
- Defining the description for the grand total of all open balance periods.
- Defining field descriptions for the three amounts on the main *cashier* entry screen.
- Setting field descriptions for various transactions (e.g., receipts, check cashing, and payouts).
- Specifying the alternate address runcode for formatting and printing addresses on receipts.
- Defining entry types used in *cashier*.
- Establishing the over/short account.
- Defining the hold action used in *cashier*.

Configuration Table Entries

Introduction

The Configuration table maintains program processing information for some CX programs. Programs that use information in the Configuration table do not need to be recompiled when Configuration table values change.

Configuration Table Entries for Financial Products

In the financial products, the Configuration table values affect processes and features:

AUTH_CHG_BY_CRS_FOR_PURGE
When set to Y, the student billing process verifies the existence of subtcw_recs (or creates the records), then evaluates subtcw_recs for purging.

ENABLE_DBCC_CHK
When set to Y, enables special check formatting in the Check Select process.

ENABLE_DEPT_BGT_CHECK
When set to Y, checks the budget at the department level and also for the entire account. If set to N (or if no config_table entry exists for this ENABLE), it checks only the entire account. The Requisition, Purchase Order Entry, and Invoice Entry programs refer to this Configuration table entry.

ENABLE_MULTI_STMT_SUBS
When set to Y, causes the CX library functions to display and print data related to multiple subsidiaries. To use this feature, you must define the subsidiaries to consolidate in the MULTI_STMT_SUBS Configuration table entry. The Statement program refers to this Configuration table entry.

ENABLE_NET_CHECKS
When set to Y, the Student Account to Student Refund (*sa2sr*) process calculates the net amount owed by the student across multiple subsidiary values and across multiple sessions. The process nets amounts due to and from a student using the NET_CHARGE_SUBS Configuration table value to compute the refund amount.

F1099_PHONE_NO
Defines the telephone number of the institution, as you want it to appear on 1099 forms. Both the 1099 Tape and 1099 Form programs use this table value. If you do not use the 1099 application for the generation of 1099s (either forms or tapes), you do not need to define this table entry.

MULTI_STMT_SUBS
Defines the subsidiaries to include on student billing statements when more than one subsidiary is to be consolidated on bills. To consolidate subsidiaries, you must define the Configuration table entry ENABLE_MULTI_STMT_SUBS value as Y. The Statement program refers to this Configuration table entry.

NET_CHARGE_SUBS
Defines all the subsidiaries to include in calculating the balance in a student's accounts. The *sa2sr* program uses this entry if the value for ENABLE_NET_CHECKS is Y. For example, if you maintain tuition charges in a S/A (student accounts) subsidiary and housing charges in a H/B (housing and board) subsidiary and want to take them both into consideration when refunding financial aid, list both S/A and H/B as the values for this Configuration table entry.

REV_EXP_ONLY_ENABLED
Enables you to display only revenue and expense accounts on Budget Review screens. The Budget Review program refers to this Configuration table entry.

SECTION 5 - JENZABAR CX PROGRAM FILES

Overview

Introduction

This section provides reference information about the files that relate to most CX programs. By understanding the file structure and the contents of the files, you can locate most of the information you need about any program.

Program Files Detailed

This section contains details about the following files:

Note: All other files for each CX program are standard C programming files with standard components and structure.

def.c

The def.c file contains the declaration of external variables (including structures) that must be available to all source files in the program. These variables can also be initialized in this file. As with other C source files, the files also contain comments. The **makedec** command uses the def.c file to create the dec.h file.

mac.h

The mac.h file contains preprocessor include and define statements, typedef statements, and structure template definition statements. The file also contains macro substitution defines and declarations of structures. This file is included in all source files during compilation through use of the dec.h file.

Definition File

Every program uses a definition (def.c) file, located in the following path: $CARSPATH/src/<*product area*>/<*program name*>.

The *def.c* file for a screen-oriented program can contain the following information:

- Includes for a mac.h file
- Declaration of global variables and structures used throughout the program
- Structure and non-structure screen binds (i.e., program buffer to screen buffer binds)
- Ring menu definitions
- Prompt line information
- Program parameters
- Declarations of dynamic memory (dmms, dmls, and dmlts) in relation to functionality within libdmm (the dynamic memory management package)
- Screen pointers

The def.c file for a non-screen-oriented program can contain the following information:

- Includes for a mac.h file
- Global program variables
- Includes for schema files def.c files
- Form pointers that provide the location for forms
- Sqlda pointers that bind the file structure to the form
- dmm, dml, and dmlt definitions
- Program parameters
- Declarations of functions so the compiler can handle a call of that function

Example of a def.c File

The following is an edited excerpt from the def.c file for the financial program *purchaudit*. It illustrates the common components of a standard CX def.c file.

```
#include "mac.h"

#include <schema/financial/podef.c>
#include <schema/financial/gledef.c>
#include <schema/financial/gltrdef.c>

struct      gle_type gle_rec;
struct      gltr_type        gltr_rec;
struct  po_type              po_rec;

long prog_date;                         /* Today's Date  */
int prog_group;                         /* Group ID */
int prog_user;                                  /* User Id */

char *getcars();                        /* database utility */

char        statbuf[200];                       /* Buffer for Status messages */

double      enctotal;                           /* Total encumbrance for po */
double      acttotal;                           /* Total actual for po */

char        po_code[sizeof(po_rec.doc_code)]; /* Po code argument */
int         allow_update = FALSE;               /* Should updating be done */
int prog_flow;                          /* Denotes the program flow */
long        begin_pono;                         /* Beginning po number for range */
long        end_pono;                           /* Ending po number for range */
int report = FALSE;                     /* Is the program to report on ALL pos
                                                it audits or just errors */
DMM_DEF(pono_dmm, long);                /* PO numbers to be audited */

char        filename[100];                      /* File name for messages */
FILE        *fileptr;                           /* Variable for writing to */
FILE        *fopen();                           /* Variable for writing to */

struct tm *localtime(), *ptr;           /* Time gathering variables */
time_t time();
long progtime;
char today[11];
long atol();
void exit();
```

mac.h Files

Every program uses a macro header (mac.h) file, located in the following path: $CARSPATH/src/purchasing/purchaudit.

The *mac.h* file for a screen-oriented program can contain the following information:

- Includes related to system header files
- Includes related to CX library and other application processes
- Includes for schema files mac.h files
- Program constant definitions (i.e., *#define* statements)
- Structure definitions

Example of a mac.h File

The following is an edited excerpt from the mac.h file for *purchaudit*. It illustrates the common components of a standard CX mac.h file.

```
#include <util/cars.h>
#include <sys/types.h>
#include <string.h>
#includeutil/msg.h>
#include <util/dmm.h>
#include <custom/acct.h>
#include <schema/financial/pomac.h>

#include <schema/financial/glemac.h>
#include <schema/financial/gltrmac.h>
#include <schema/common/idmac.h>


#define     AUDIT_PATH                "/audit/purchasing/purchaudit/"

#define     FATAL            (-2)
#define     LOAD_ERR (-3)
#define     SUCCESSFUL       (0)

#define INV_DOC_REF          "IV"             /* Document ref for inv */
#define ACT_ADD_INV_TYPE     "AINV"
#define ACT_UPD_INV_TYPE     "UINV"
#define ACT_CR_MEMO_TYPE     "CMEM"

#define PURCH_VCH            "PC"
#define AP_VCH                        "AP"

/*  Status code for po with archived supporting data   */
#define ARCHIVED             'A'

#define PARAMETERS   1                        /* Parameters passed */
#define STATUS_GRP   2                        /* Status group for msg */
#define LOAD_ERR_GRP 3                        /* PO's not loaded -error */
#define FATAL_GRP    4                        /* FATAL ERR Group for msg */
#define PO_HEADER_GRP        5                        /* Header for Purchase Order
                                                  Information */
#define PO_GRP               6                /* Purchase Order Information */

#define RANGE                1                        /* Range of IDS  */
#define SPECIFIC     2                        /* Specific IDS  */
#define ALL          3                        /* ALL IDS       */
#define NOT_DETERMINED       4                /* Prog Flow not determined */
```

SECTION 6 - ACCOUNTS PAYABLE/PAYROLL: CHECK ABORT

Overview

Introduction

This section provides you with reference information about the Check Abort (*ckabort*) program. The Accounts Payable and Payroll products use *ckabort* to restore the system to the state in which it existed before checks were selected. It should be run if the system fails while checks are being selected or if the *ckslct* program unexpectedly causes a fatal error.

Program Features Detailed

This section contains details about the following features of the *ckabort* program:

- Process flow
- Parameters

Program Screens

Because *ckabort* is a background maintenance process, it does not use program screens.

Records and Tables Used

The *ckabort* program uses the following records:

(ckgrp_rec)
The Check Group record that stores information about selecting and printing checks.

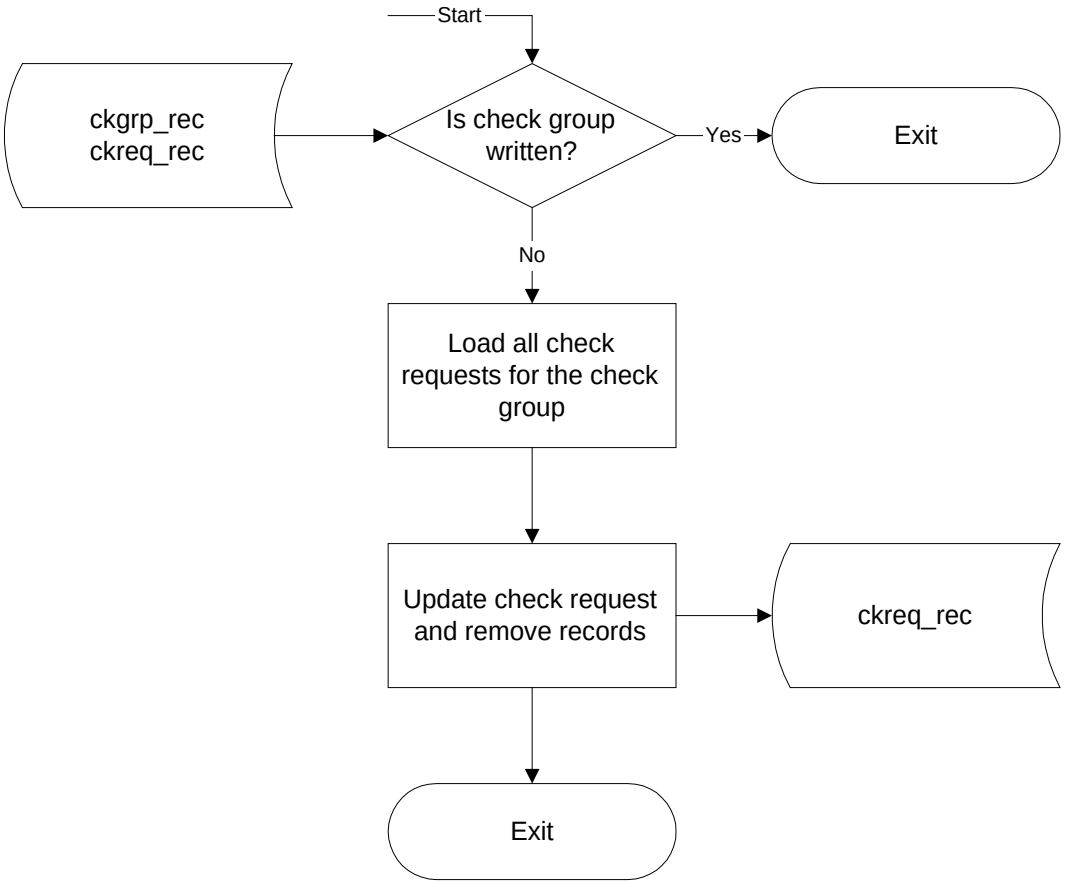
ckreq_rec
The Check Request record that stores information about checks to be printed.

subb_rec
The Subsidiary Balance record that stores information about the amount of the invoice and the ckreq_rec linking the check with the actual invoice paid.

Process Flow

Diagram

The following diagram shows the flow of data in the *ckabort* program.



Data Flow Description

The following describes the data flow in the *ckabort* program.

1. The *ckabort* program verifies that the check group has not been written. If it has not, the program continues processing the abort of the check writing process.
2. The *ckabort* program loads all information for the check group and uses that information to update each of the check requests. It zeroes the ckreq_no found in the subb_rec, then removes the ckreq_rec.
3. The *ckabort* program updates the Check group record to show that the processing of the group has been aborted.

Program Relationships

Since an institution only uses the program under unusual circumstances, and since it is a maintenance program, *ckabort* does not interact with any other CX program.

Check Abort Parameters

Introduction

CX contains parameters and compilation values for executing the *ckabort* program. You can specify parameters to compile *ckabort* in a specified manner at the time of execution.

Note: You can also specify compilation values with the includes for the financial products that affect the *ckabort* program.

Parameter Syntax

You can display *ckabort* parameters by entering the following: **ckabort -,**

The following is the correct usage for running the *ckabort* program from the UNIX shell:

Usage: ckabort -g group_no

Parameters that appear in brackets are optional. Parameters that do not appear in brackets are required.

Parameters

The following lists the parameters for running *ckabort*.

-g group number
Required - Specifies the group number for check selection.

SECTION 7 - ACCOUNTS PAYABLE/PAYROLL: CHECK POST

Overview

Introduction

This section provides you with reference information about the Check Post (*ckpost*) program. The Accounts Payable and Payroll products use *ckpost* to post check transactions to the general ledger. Users run *ckpost* after *ckslct* and *fps*.

Program Features Detailed

This section contains details about the following features of *ckpost*:

- Process flow
- Parameters

It also contains instructions on how to troubleshoot problems in the checkwriting process.

Program Screens

Because *ckpost* is a background process, it does not use program screens.

Records and Tables Used

The *ckpost* program uses the following records:

ckgrp_rec
The Check Group record that stores information about selecting and printing checks.

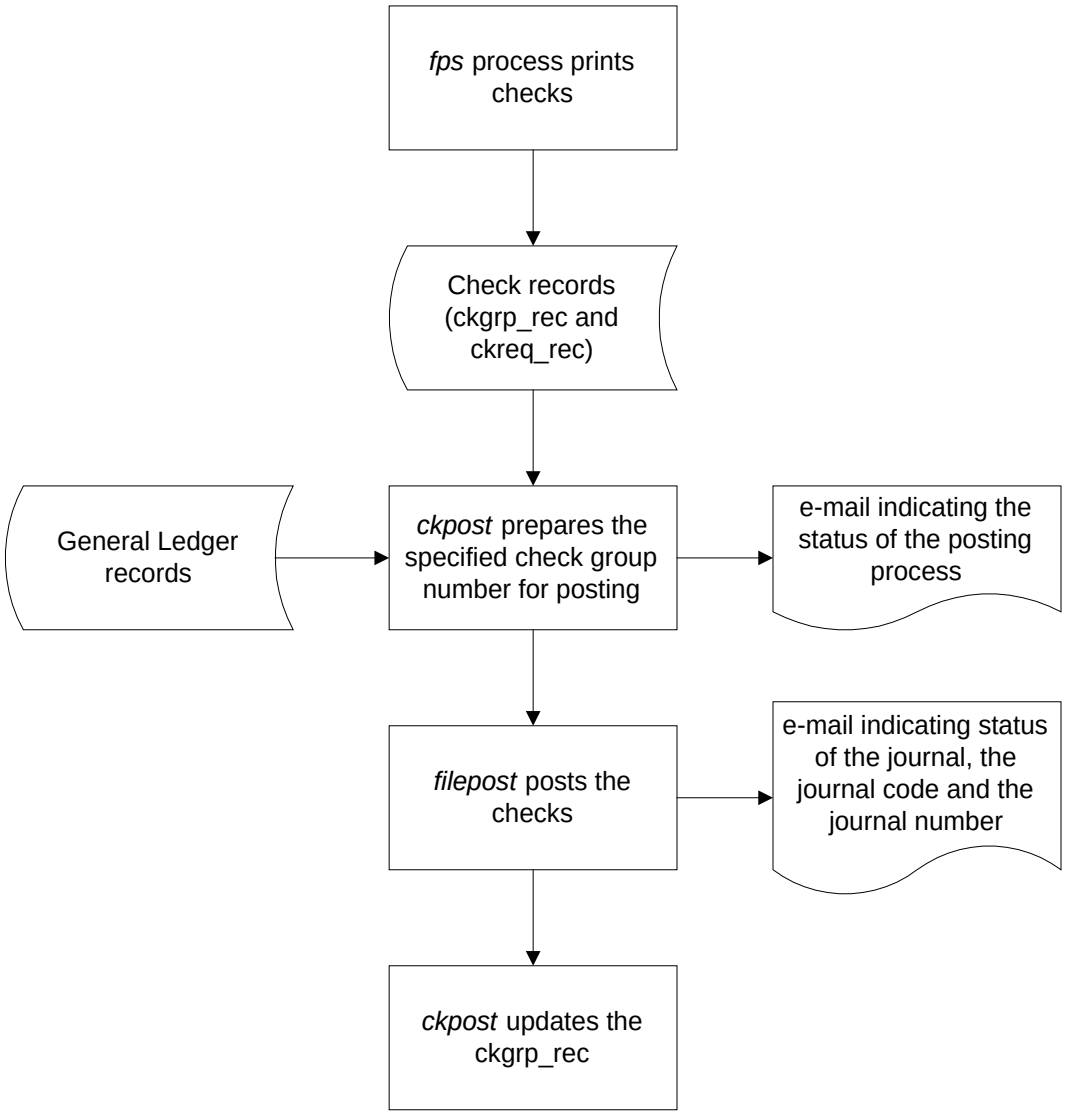
ckreq_rec
The Check Request record that stores information about checks to be printed.

subb_rec
The Subsidiary Balance record that stores the amount of the invoice that is being paid on this check. After the check is printed the *ckpost* process will change the status of the subb_rec to “C”losed so that it will not be selected again.

Process Flow

Diagram

The following diagram shows the flow of data in the *ckpost* program.



Data Flow Description

The following describes the data flow in the *ckpost* program.

Check creation includes the following steps.

Note: In each file extension, the *x* represents a system-generated sequence number.

1. The user creates the Check Group record (ckgrp_rec).
2. The user runs the *ckslct* process.

3. The *ckslct* process creates a file (apcheck.fx or prcheck.fx), in $CARSPATH/spool/forms, creating the ckreq_recs and updating the subb_rec with the ckreq_rec number.

   Note: The user should select the same form type used for printing the checks (e.g., Accounts payable checks with a formtype of *apcheck*). When the checks post to the general ledger, the check date serves as the posting date.

4. *The fps* process first reads the Document table to find the last check number printed and allows the user to verify that the correct check is in the printer. It then reads the apcheck.fx, prints the check and creates a tracking file apcheck.tx. This tracking file contains the check number associated with each ckreq_rec. When complete the apcheck.fx file is renamed to apcheck.dx.
5. The c*kpost* process then reads the apcheck.tx file and posts the transactions to the general ledger, updating the subb_rec with a status of “C”losed status. At the end of the process, the Document table is updated to reflect the last check number produced by the process and the apcheck.tx is renamed apcheck.px.
6. The *ckpost* program updates the appropriate fields in the ckgrp_rec and the subb_rec.
7. When *ckpost* completes processing, it sends electronic mail as follows:
   - Two messages from *filepost*. The first contains the status of the journal that was posted, and second contains the journal code and journal number of the journal that was posted.
   - A message from *ckpost*, containing a report of the status of the posting process.

   CAUTION: The user should always post checks and review the printed output before distributing them.

Impact of Processing on File Extensions

The check selection process and its impact on file extensions can be summarized as follows:

1. The *ckslct* process creates apcheck.fx.
2. The *fps* process reads apcheck.fx and creates apcheck.tx; when complete, it renames apcheck.fx to apcheck.dx.
3. The *ckpost* process reads apcheck.tx and posts to G/L; when complete, it renames apcheck.tx to apcheck.px and updates the Document table with the number of the last check posted.

Program Relationships

The *ckpost* program receives input from the tracking file that created *ckslct* and *fps*, and uses *filepost* to post transactions to the general ledger.

Check Post Parameters

Introduction

CX contains parameters and compilation values for executing the *ckpost* program. You can specify parameters to compile *ckpost* in a specified manner at the time of execution.

Parameter Syntax

You can display *ckpost* parameters by entering the following: **ckpost -,**

The following is the correct usage for running the *ckpost* program from the UNIX shell:

Usage: ckpost -g groupnum [-d po_doc_ref]

Parameters that appear in brackets are optional. Parameters that do not appear in brackets are required.

Parameters

The following lists the parameters for running *ckpost*.

-g group number
Required - Specifies the check group number.

-d po_doc_ref
Optional - Specifies the purchase order document reference code (e.g., PO).

Troubleshooting the Check Production Process

Introduction

When the check production process does not complete as expected, some intervention in the CX process is required. This section outlines the steps to follow in the case of a printer jam, or if unwanted checks are created.

Process in Case of Printer Jam

In case a printer jam occurs during a check run, the user should perform the following steps:

1. Interrupt *fps*.
2. Call the Computer Center support personnel and have them delete the current apcheck.tx file located in $CARSPATH/spool/forms.
3. Rerun *fps*, starting with the original check number for setup. When the process prompts for the beginning check number, enter the current check number. The system will automatically void all checks in the range between the two numbers; the user does not need to run *ckabort*.

Process in Case Unwanted Checks are Created

Occasionally, you may create a check run that includes unwanted or unneeded checks. In this case, the user should perform the following steps:

1. Call computer center and have them:
   - Delete the apcheck.tx file located in $CARSPATH/spool/forms.
   - Rename apcheck.dx file to apcheck.fx.
2. Run *ckabort*.

SECTION 8 - ACCOUNTS PAYABLE/PAYROLL: CHECK SELECT

Overview

Introduction

This section provides you with reference information about the Check Select (*ckslct*) program. The Accounts Payable and Payroll products use *ckslct* to retrieve the desired check records and invoices for processing and payment.

Program Features Detailed

This section contains details about the following features of the *ckslct* program:

- Process flow
- Parameters
- Program screens

Records and Tables Used

The *ckslct* program uses the following records:

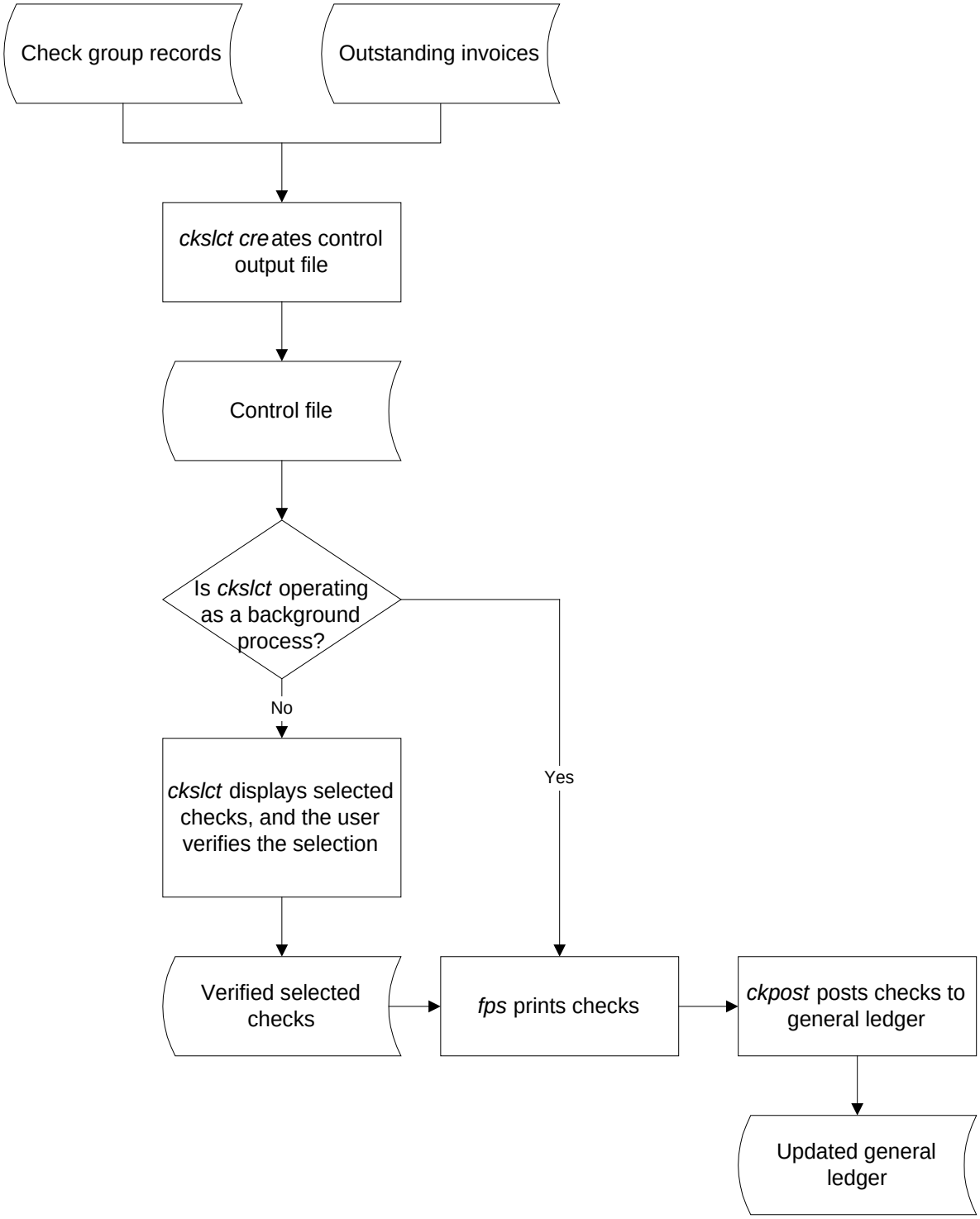
ckgrp_rec
The Check Group record that stores information about selecting and printing checks.

ckreq_rec
The Check Request record that stores information about checks to be printed.

Process Flow

Diagram

The following diagram shows the flow of data in the *ckslct* program.



Process Flow Description

The following describes the process flow in the *ckslct* program.

1. The *ckslct* process selects checks based on the following rules of selection. The rules relate to the contents of the subb_rec.
   - The subsidiary code must match the subsidiary code in the Check Group record (ckgrp_rec).
   - If the Check Group period has a non-blank value, the subsidiary balance period must match the Check Group period. If it does not, *ckslct* will ignore the subsidiary balance period criterion. The process normally uses the period for payroll checks to restrict the check run to one specific payroll.
   - The subsidiary balance actual (invoice) amount must have a debit or a credit value. If it is zero, it will not be selected unless your institution uses the Select zero bal checks option. The payroll process uses the Select zero bal checks option to print a check stub for employees who have wages and deductions that have a net amount of zero.
   - The subsidiary balance encumbered amount must be zero. If the encumbered amount is non-zero, then *ckslct* assumes that the record relates to a purchase order and is not an invoice. A Subsidiary Balance record with both an actual and an encumbered amount should not occur and is probably the result of an incorrectly posted manual check or adjusting entry. In any case, *ckslct* will not select the invoice.
   - The due date in the subb_rec must be less than or equal to the due date given in the ckgrp_rec.
   - The subsidiary balance status must be O (Open or Outstanding).
   - The Hold Payment flag in the invoice records must be N. If it is Y (signifying that the payment is being held) or A (meaning that the invoice needs approval before it can be paid) the invoice will not be selected.
   - The check request number in the subb_rec must be zero. When *ckslct* selects an invoice in the check run, it assigns a check request number. If an invoice has a check request number, *ckslct* assumes it has already been selected and will not select it again.
   - The payment terms in the subb_rec must match the selection payment terms. (Normally, users do not select special payment terms, so this rule does not apply).

   Note: The Check Group record allows for other limitations during the check selection process. For example, the user can limit the number of checks to be selected by updating the maximum number of checks to create.

   Users can limit the dollar amount of checks to select in the following two ways. When you reach the limit, *ckslct* will stop selecting outstanding invoices.
   - The dollar amount of a single check.
   - The total dollars available for all checks.

2. The *ckslct* program creates one Check Request record (ckreq_rec) for each check selected. The amount of the ckreq_rec is the amount of the check to be created, which is usually the total of the amounts in the selected subb_recs.

   Note: if *ckslct* selects a discounted invoice, the check request amount is less than the total dollar amount of the invoices to be paid with the check.

   One check usually pays all invoices selected for a given vendor. You can, however, set the selection process to select one check for each invoice in either of the following two ways.
   - Set the Single Invoice Per Check field in the Payment Terms table (payterm_table) to Y.
   - Set the One Ck field of the subb_rec to Y when entering the invoice into the system through use of the *purch* program.

3. If an invoice qualifies for a vendor discount, *ckslct* computes the discount and creates a ckreq_rec with the discounted amount.

   Note: To qualify for a discount, an invoice must meet the following three conditions:
   - The invoice must meet the selection criteria listed in step 1.
   - The invoice must use discounted payment terms. This is an entry in the payment terms table with a non-zero discount percentage.
   - The invoice date plus the number of discount days (as specified in the payment terms table) must be less than or equal to the check date.

4. The *ckpost* program refunds discount amounts to either the function(s) that were charged for the invoice, or to a specific General Ledger account. Your institution controls this process by setting definitions for the macro AP_DISC_ACCT_DEFINE in $CARSPATH/include/custom/ckslct. If you define the REV_FUND, REV_FUNC, REV_OBJ, and REV_SUBFUND macros, they will identify the G/L account for the discounts. Otherwise, the *ckpost* program refunds the amounts back to the functions(s) from which they came.

5. If the Display field in the ckgrp_rec contains Y, *ckslct* selects the checks and displays them on a selection screen. The selection screen enables users to preview the checks that *ckslct* selected to print. From the selection screen, users can eliminate any checks that *ckslct* selected in error.

   CAUTION: If the Display field contains Y, *do not* run *ckslct* in background. This process will lock the display. If this happens, users will need to call the Computer Center support personnel and have them kill the *ckslct* process. Users should then be able to restart the process without running *ckabort*. To eliminate this problem, users must be sure to run it in foreground or change the Display flag to N.

   Note: If there are specific invoices that you do not want to pay, you can also use this screen to remove them before printing the check for the rest of the invoices. Any checks or invoices removed will be available for payment on the next selection run.

6. The checks sort by designated office for the check distribution (as set in the suba_rec), and then by the name of the payee. For most institutions, the check distribution office is blank for accounts payable vendors; therefore, accounts payable checks print in alphabetical order by payee name. Most institutions use the distribution office for payroll check selections to specify check distribution points. Within each office, the checks sort alphabetically by the recipient's last name. The dollar amount of the check appears in the Check Amount column.

   Note: If the user erases the check on the main screen, *ckslct* will not allow the erasure of any of the individual invoices since erasing the check would have already erased all the invoices. Likewise, if all invoices for a check have been erased, *ckslct* will not allow the check to be erased. It will display the message, “Cannot erase a check that already has erased invoices.” To reset all checks back to their original statuses, use the **Set Back to Initial Status** command option on the Main Selection screen.

7. Upon the user’s selection of the **Execute** command, *ckslct* creates the form file that will be used in printing the checks.

   Note: The **Abort** command stops the selection process and interrupts the check run. The **Abort** command also resets all the invoices to their original statuses, making them available for another check run.

8. If the Display field in the ckgrp_rec contains N, *ckslct* operates under the assumption that all invoices eligible for selection are to be paid. You can run *ckslct* in background if you have entered **N** for no display. The *ckslct* program selects the same invoices that the Display mode selects, but it does not display any information to the screen and does not allow the user to change any of the invoices that are selected. Most institutions use this type of processing for payroll. In addition, some institutions select accounts payable checks for processing using this background method, depending on the procedures established for the check writing process.
9. The *ckslct* program sends error messages and status reports to the user through electronic mail.
10. The *ckslct* program updates the ckgrp_rec values for the following fields:
    - Amount Paid
    - Amount Owed
    - Number of Check Requests
    - Number of Payees Owed
11. The output file from *ckslct* serves as input to the *fps* program, which prints forms and checks.

    Note: Most institutions use form types of *apcheck* (accounts payable checks), *prcheck* (payroll checks) or *dirdep* (non-negotiable direct deposit slips) to print checks.
12. Check processing concludes when *ckpost* posts the checks to the general ledger. For more information about *ckpost*, see *Accounts Payable/Payroll: Check Post* in this manual.

Program Relationships

The following programs interrelate with *ckslct*.

fps

The *fps* program uses output from *ckslct* to print checks.

Check Select Parameters

Introduction

The CX contains parameters and compilation values for executing the *ckslct* program. You can specify parameters to compile *ckslct* in a specified manner at the time of execution.

Note: You can also specify compilation values with the includes for the financial products that affect the *ckslct* program.

Parameter Syntax

You can display *ckslct* parameters by entering the following: **ckslct -,**

The following is the correct usage for running the *ckslct* program from the UNIX shell:

Usage: ckslct -s subsidiary -g group_no [-a alt_addr] [-p] [-d po_docref]

[-e pay_excl] [-t pay_req] [-z]

Parameters that appear in brackets are optional. Parameters that do not appear in brackets are required.

Parameters

The following lists the parameters for running *ckslct*.

-s subsidiary
Required - Specifies the subsidiary type for check selection.

-g grp_no
Required - Specifies the group number for check selection.

-a alt_addr
Optional - Specifies the alternate address run code.

-p
Optional - Indicates the user wants to process the checks for direct deposit

-d po_docref
Optional - Specifies the purchase order document reference code

-e pay_excl
Optional - Indicates the payment terms, if any, to exclude from the selection process.

-t pay_req
Optional - Indicates the payment terms, if any, to specifically select during the selection process.

-z
Optional - Indicates that the user does not want to exclude zero balance checks.

Program Screens

Introduction

The *ckslct* program uses two screens to display information about the checks and their balances.

Access

The screen files are located in the following directory path:
$CARSPATH/src/acctspay/ckslct/SCR

Screen Files and Table/Record Usage

The *ckslct* screens appear in the following files and use the indicated tables and records:

ckdspl
Contains the Checks to Be Printed screen.

Tables/Records: none

subbdspl
Contains the Balances to Appear on Check screen.

Tables/Records: none

SECTION 9 - ACCOUNTS PAYABLE/PAYROLL: DIRECT DEPOSIT TAPE

Overview

Introduction

This section provides you with reference information about the Direct Deposit Tape (*ddtp*) program. The Accounts Payable/Payroll products use *ddtp* to process tapes for direct deposit payrolls and payments on accounts payable.

Program Features Detailed

This section contains details about the following features of the *ddtp* program:

- Process flow
- Parameters

Program Screens

Because *ddtp* is a background process, it does not use program screens.

Records and Tables Used

The *ddtp* program does not use any CX records or tables. It reads the tracking dirdep.fx file created by *fps*, where x is a system-generated sequence number.

Process Flow

Description

The *ddtp* program reads the output files txxxxxxx.dat and txxxxxxx.lab for direct deposit information and creates dxxxxxxx.lab and dxxxxxxx.dat files in the $CARSPATH/spool/tape directory. These files are read by the *taperead* program to create the tape.

Note: If your institution determines a tape must be rerun after *ddtp* has concluded successfully, you can change the file names from dxxxxxx.lab and dxxxxxxx.dat to txxxxxxx.lab and txxxxxxx.dat respectively.

Program Relationships

The *dirdep* program provides input for the *ddtp* program.

Direct Deposit Tape Parameters

Introduction

The CX contains parameters and compilation values for executing the *ddtp* program. You can specify parameters to compile *ddtp* in a specified manner at the time of execution.

Note: You can also specify compilation values with the includes for the financial products that affect the *ddtp* program.

Parameter Syntax

You can display *ddtp* parameters by entering the following: **ddtp -,**

The following is the correct usage for running the *ddtp* program from the UNIX shell:

Usage: ddtp -g group_no

Parameters that appear in brackets are optional. Parameters that do not appear in brackets are required.

Parameters

The following lists the parameters for running *ddtp*.

-g
Required - Specifies the group number for check selection.

SECTION 10 - ACCOUNTS PAYABLE/PAYROLL: DIRECT DEPOSIT

Overview

Introduction

This section provides you with reference information about the Direct Deposit (*dirdep*) program. The Accounts Payable/Payroll products use the *dirdep* for direct deposit payrolls to any number of accounts.

Program Features Detailed

This section contains details about the following features of the *dirdep* program:

- Process flow
- Parameters

Records and Tables Used

For an employee to be eligible for direct deposit, he/she must have a suba_rec and a ckalloc_rec. In addition, the *dirdep* program uses the following records and tables:

ckalloc_rec
The Check Allocation record that contains information for determining the allocation of funds between accounts for a direct deposit check.

ckgrp_rec
The Check Group record that stores information about selecting and printing checks.

ckreq_rec
The Check Request record that stores information about checks to be printed.

doc_table
The Document table that contains information about document codes and stations.

id_rec
The Identification record that contains information about each individual or entity in the CX database.

suba_rec
The Subsidiary Account record that contains information about the subsidiary number.

Process Flow

Data Flow Description

The following describes the data flow in the *dirdep* program.

1. The user runs the *ckslct* program to create the direct deposit information, either for a payroll or for accounts payable.
2. The *dirdep* program creates the two output files (txxxxxxx.lab and txxxxxxx.dat) in the /usr/spool/tape directory.

Program Relationships

The output file from *dirdep* is used by *ddtp* to create the direct deposit tape.

Direct Deposit Parameters

Introduction

CX contains parameters and compilation values for executing the *dirdep* program. You can specify parameters to compile *dirdep* in a specified manner at the time of execution.

Note: You can also specify compilation values with the includes for the financial products that affect the *dirdep* program.

Parameter Syntax

You can display *dirdep* parameters by entering the following: **dirdep -,**

The following is the correct usage for running the *dirdep* program from the UNIX shell:

```
Usage: dirdep -[-a account_no] -c check_grp -d dest_ach -i co_id_num
          -m immed_code -n co_name -o origin_ach -r ref_code
```

Parameters that appear in brackets are optional. Parameters that do not appear in brackets are required.

Parameters

The following lists the parameters for running *dirdep*.

-c check_grp
Required - Specifies the check group number to which the direct deposit applies.

-d dest_ach
Required - Specifies the destination ACH.

-i co_id_num
Required - Specifies the company's federal tax identification number.

-m immed_code
Required - Specifies the immediate destination code for the direct deposit.

-n co_name
Required - Specifies the name of the company to which the direct deposit applies.

-o origin_ach
Required - Specifies the originating ACH.

-r ref_code
Required - Specifies the reference code.

Program Screens

Introduction

The *dirdep* program uses one PERFORM screen to capture information about employees who are eligible for direct deposit.

Access

The screen file is located in the following directory path: $CARSPATH/modules/payroll/screens

Screen Files and Table/Record Usage

The *dirdep* screen appears in the following file and uses the indicated tables and records:

subacct

Contains the Subsidiary Account Record/Check Allocation Record screen.

Tables/Records: ckalloc_rec, id_rec, suba_rec

SECTION 11 - ACCOUNTS PAYABLE: F1099 BUILD

Overview

Introduction

This section provides you with reference information about the f1099 Build (*f1099bld*) program. The Accounts Payable product uses *f1099bld* to create 1099 records. CX supports two types of 1099s, 1099-INT and 1099-MISC.

Program Features Detailed

This section contains details about the following features of the *f1099bld* program:

- Process flow
- Parameters

Program Screens

Because *f1099bld* is a background process, it does not use program screens.

Tables and Records Used in the Program

The *f1099bld* program uses the following tables and records:

1099_table
The Form 1099 table that defines valid 1099 form types.

f1099_rec
The f1099 record that contains information about 1099 reporting.

gle_rec
The General Ledger Entry record that contains information about each entry.

id_rec
The Identification record that contains information about each individual or entity in CX.

subb_rec
The Subsidiary Balance record that contains summary information per period for subsidiary accounts or invoices for accounts payable subsidiary accounts.

sube_rec
The Subsidiary Entry record that contains information about postings to the subsidiary accounts.

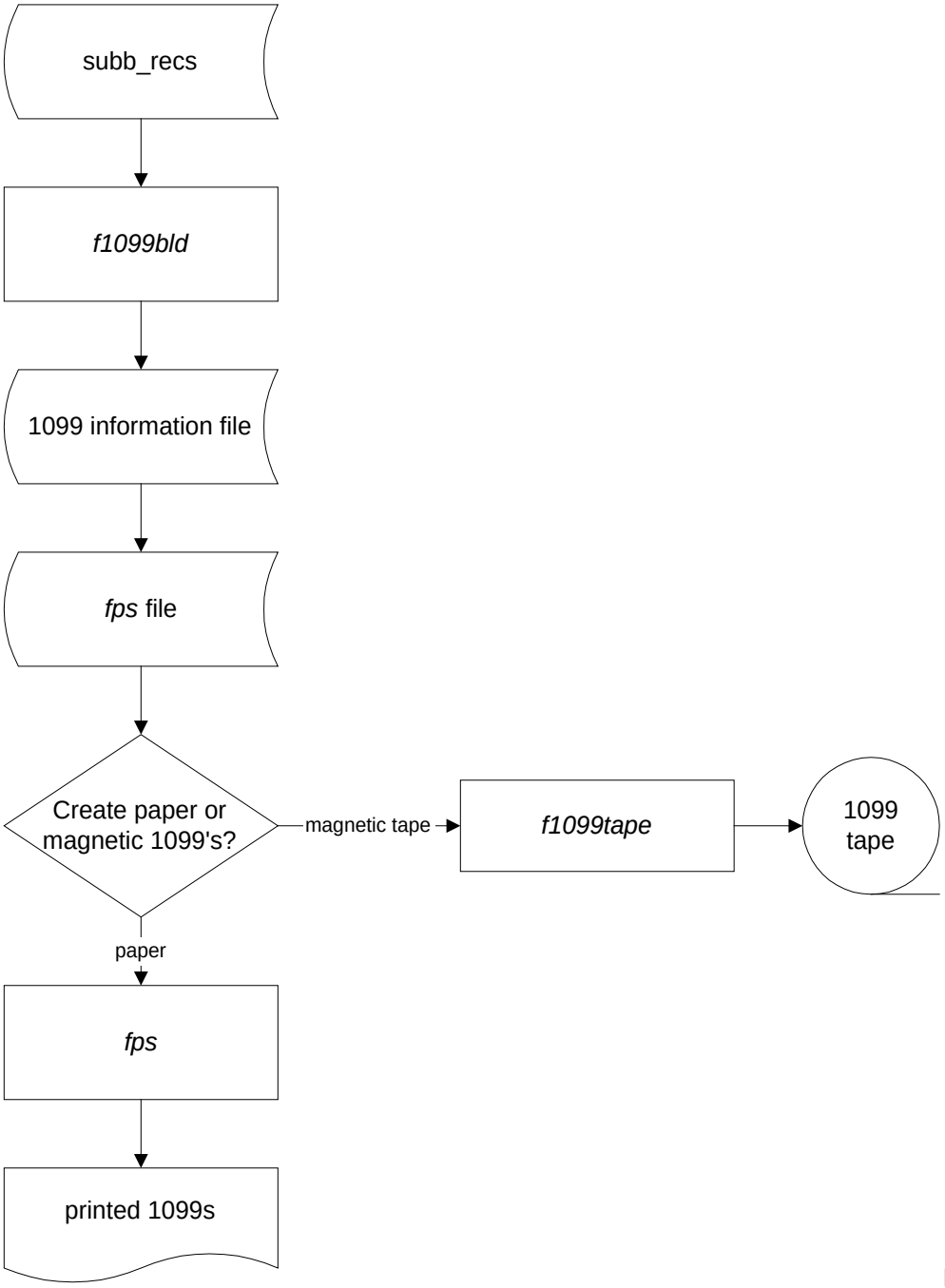
subtr_rec
The Subsidiary Transaction record that contains detailed transactions for subsidiary account posting.

vch_rec
The Journal record that contains information relating to groups of general ledger entries.

Process Flow

Diagram

The following diagram shows the flow of data in the *f1099bld* program.



Data Flow

Description

During the year, your institution enters invoices into the system using an Accounts Payable Voucher or the Purchasing process. The system stores the invoices in the subb_rec. At the end of the year, the subb_recs contain all the information required to determine the vendors who should receive 1099s. Then, for each invoice that indicates a 1099 should be printed, the system adds a 1099 record to the database, and uses these 1099 records for printing the 1099s.

The following describes the data flow in the *f1099bld* program.

1. The *f1099bld* program accesses the information in the subb_rec to select the invoices that will result in 1099s. The 1099 procedures use the f1099 code field to select the invoices that require 1099s.
2. The *f1099bld* program uses the invoices to build a file of 1099 records that contains all the information required to print 1099s. The system also uses values coded in the file $CARSPATH/macros/custom/f1099 to complete the 1099 records.
3. The system sends a status report using electronic mail.
4. The 1099 database file serves as the basis for the *fps* file which is input to *f1099form* and *fps* for printing the information on preprinted 1099 forms.

    Note: The 1099 file can also be used to produce magnetic tapes for the IRS. If you have filed 1099s on tape before, you do not need to submit an application or a test tape unless your equipment has changed drastically. The IRS assumes that institutions and enterprises file 1099s using the same programs and the same equipment each year.

Program Relationships

The following programs use *f1099bld*.

- *f1099form* (uses output from *f1099bld* to format 1099s for printing)
- *f1099tape* (uses output from *f1099bld* to format 1099s for tape)

Library Relationships

The *f1099bld* program uses the following library: Lib1099

f1099 Build Parameters

Introduction

CX contains parameters and compilation values for executing the *f1099bld* program. You can specify parameters to compile *f1099bld* in a specified manner at the time of execution.

Note: You can also specify compilation values with the includes for the financial products that affect the *f1099bld* program.

Parameter Syntax

You can display *f1099bld* parameters by entering the following: **f1099bld -,** and then viewing the mail message that the command generates.

The following is the correct usage for running the *f1099bld* program from the UNIX shell:

Usage: f1099bld -s subs -y year

Parameters that appear in brackets are optional. Parameters that do not appear in brackets are required.

Parameters

The following lists the parameters for running *f1099bld*.

-s subs
Required - Specifies the subsidiary (e.g., A/P) for which you want to create 1099s.

-y year
Required - Specifies the year (e.g., 1996) for which you want to create 1099s.

SECTION 12 - F1099FORM

Overview

Introduction

This section provides you with reference information about the f1099 Form (*f1099form*) program. The Accounts Payable product uses *f1099form* to create form information for 1099s.

Program Features Detailed

This section contains details about the following features of the *f1099form* program:

- Process flow
- Parameters

Tables and Records Used in the Program

The *f1099form* program uses the following tables and records:

1099_table
The Form 1099 table that defines valid 1099 form types

ctry_table
The Country table that defines valid country names and codes

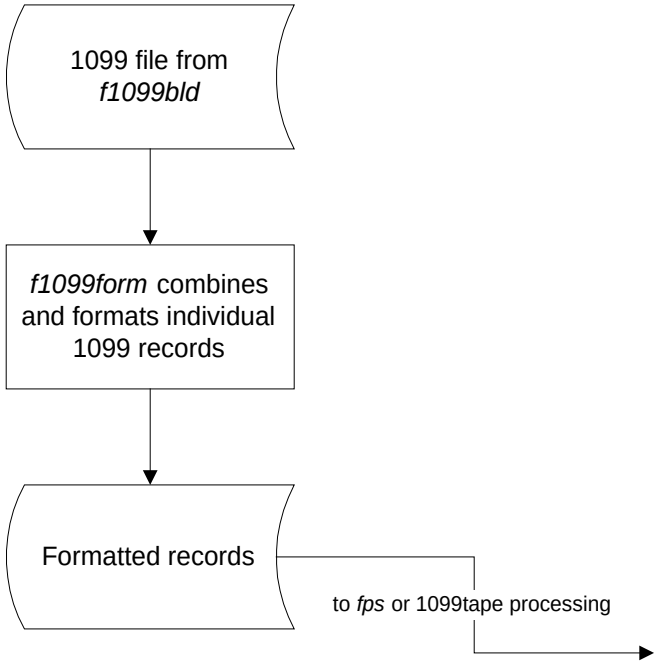
f1099_rec
The f1099 record that contains information about 1099 reporting

id_rec
The Identification record that contains information about each individual or entity in the CX database

Process Flow

Diagram

The following diagram shows the flow of data in the *f1099form* program.



Data Flow Description

The following describes the data flow in the *f1099form* program.

1. The *f1099form* program scans the 1099 file from the *f1099bld* process.
2. The *f1099form* program combines individual 1099 records and formats them for printing.
3. The *fps* program uses the formatted records to print the 1099s, or the *f1099tape* program uses the records to create a 1099 tape.

Note: CX offers two menu options for printing 1099s, as follows:

- Print 1099s for every 1099 record in the database
- Print a 1099 for a specific individual

Mail Messages from *f1099form*

When you complete processing in *f1099form*, the process generates a mail message, as in the following example:

```
The 1099 form information was produced successfully.
A total of 7 1099 forms was created.
F1099 form status report:
The following ID's have been processed.
Americana            (ID=  18704) has been processed.
Kern International I (ID=  19789) has been processed.
Outside Magazine     (ID=  19790) has been processed.
Total of 3 1099-I forms were processed. (2) w/o fed id numbers.

American             (ID=  19001) has been processed.
University Microfilm (ID=  18010) has been processed.
Wells, Norm          (ID=  19000) has been processed.
Westminster Press Th (ID=  18000) has been processed.
Total of 4 1099-M forms were processed. (3) w/o fed id numbers.
F1099 form run successfully completed Mon Feb  9 10:30:03 1997
```

The header of the message gives a status report (*The 1099 form information was produced successfully*.) and the total number of forms that the process formatted to print. The message then lists each individual or company that is scheduled to receive a 1099, grouped by form. The message also indicates the number of 1099s that did not have a federal identification number (either the Employer Identification Number or the Social Security Number).

Procedure for Corrected 1099 Forms

CX does not automatically generate corrected 1099s. If you are not filing on magnetic tape, you can print corrected 1099s by manually entering and updating the existing 1099 records in the database and by following these instructions:

1. If only the amounts were wrong, you can easily print corrected 1099s by updating the 1099 file with the correct amounts and printing the correct 1099s for the individuals affected using *f1099form*. You must manually enter X in the Correction box at the top of the forms.
2. If the wrong form was filed (e.g., you specified 1099-INT instead of 1099-MISC), make the correction in the following two steps.
    - Correct the incorrect form you supplied to the IRS. Make the correction by zeroing out or correcting all the amounts in the 1099 records that apply to the individuals affected, and printing the corrected forms (with zero amounts, or with the correct amounts) with the "1099 Forms by Id" option. Manually "X" the Corrected box at the top of the form. These corrected 1099s must have one corrected transmittal Form 1096, with an “X” in the appropriate Correction box.
    - Enter the correct information in the 1099_rec for the correct form, and print the corrected 1099s with *f1099form*. Prepare another 1099 for these corrected forms. Do *not* mark the Corrected box on these forms; instead, enter the words *Filed to Correct Document Type* in the blank space to the right of *For Official Use Only* at the bottom of the 1099.

Program Relationships

The *f1099form* program uses *f1099bld* output to format for printing.

Library Relationships

The *f1099form* program uses the following library: Lib1099

SECTION 13 - F1099 TAPE

Overview

Introduction

This section provides you with reference information about the f1099 Tape (*f1099tape*) program. The Accounts Payable product uses *f1099tape* to create tape information for 1099 forms.

Program Features Detailed

This section contains details about the following features of the *f1099tape* program:

- Process flow
- Parameters

Program Screens

Because *f1099tape* is a background process, it does not use program screens.

Records and Tables Used in the Program

The *f1099tape* program uses the following tables and records:

1099_table
The Form 1099 table that defines valid 1099 form types

ctry_table
The Country table that defines valid country names and codes

f1099_rec
The f1099 record that contains information about 1099 reporting

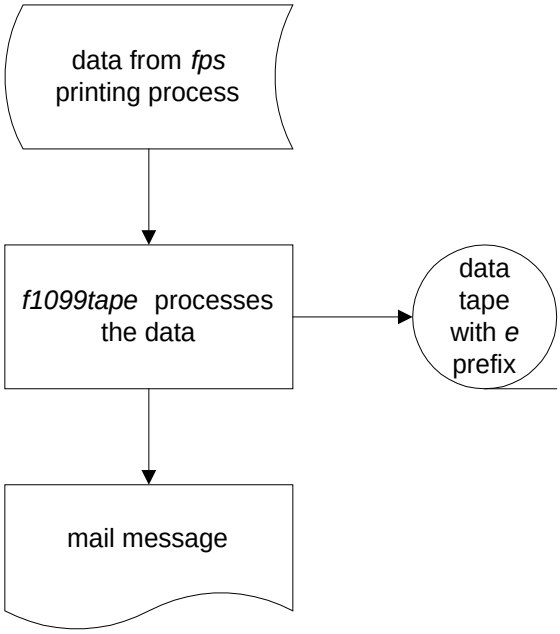
id_rec
The Identification record that contains information about each individual or entity in the CX database

vnd_rec
The Vendor record that contains information about the vendors that the institution uses to obtain goods and services

Process Flow

Diagram

The following diagram shows the flow of data in the *f1099tape* program.



Data Flow Description

The following describes the data flow in the *f1099tape* program.

1. The *fps* program prints the 1099s.
2. The *f1099tape* program creates an intermediate data file in /usr/$CARSV/spool/tape.
3. The *f1099tape* program takes the information from this file and puts it on tape.
4. The *f1099tape* program changes the prefix on the tape files from *f* to *e.*
5. The *f1099tape* program sends electronic mail to the operator, as in the following example:

```
Subject: F1099 tape information prepared
The 1099 tape information was produced successfully.
A total of 4 type 'B' records were created.
F1099 tape status report:
The following ID's have been processed for later use by tps.

Westminster Press Th (ID= 18000) has been processed.
Rose, Mildred H      (ID= 18008) has been processed.
University Microfilm (ID= 18010) has been processed.
Wells, Norm          (ID= 19000) has been processed.

F1099 tape run successfully completed Mon Feb  9 12:15:59 1997
```

Note: This message provides a list of all individuals and companies that appear on the magnetic tape. Use the *Total B records created* message to prepare the tape label of the magnetic tape for the IRS.

Process for Rerunning 1099 Tapes

As described in the previous section, when *f1099tape* writes the tape, it renames the tape files as follows:

Old name: "/usr/carsi/spool/tape/f######.dat" and "/usr/carsi/spool/tape/f######.lab"
New name: "/usr/carsi/spool/tape/e######.dat" and "/usr/carsi/spool/tape/e######.lab"

In this example, ###### is a six-digit number generated by the system.

Therefore, if you need to rewrite the 1099 tape, you must rename the tape file in /usr/carsi/spool/tape from "e######.dat" and "e######.lab" back to "f######.dat" and "f######.lab".

Process for Corrected 1099s in Tape Reporting

All tape corrections require manipulation of the /usr/carsi/spool/tape files produced by *f1099tape*. For more information, contact Jenzabar, Inc.

Program Relationships

The following programs use *f1099tape*.

- *f1099tp* uses output from *f1099tape* to create the magnetic tape for the IRS
- *f1099tape* uses output from *fps* to produce its updated files

SECTION 14 - F1099TP

Overview

Introduction

This section provides you with reference information about the f1099 Tp (*f1099tp*) program. The Accounts Payable product uses *f1099tp* to write 1099 tapes.

Program Features Detailed

This section contains details about the following features of the *f1099tp* program:

- Process flow
- Parameters

Program Screens

Because *f1099tp* is a background process, it does not use program screens.

Parameters

The *f1099tp* does not use processing parameters.

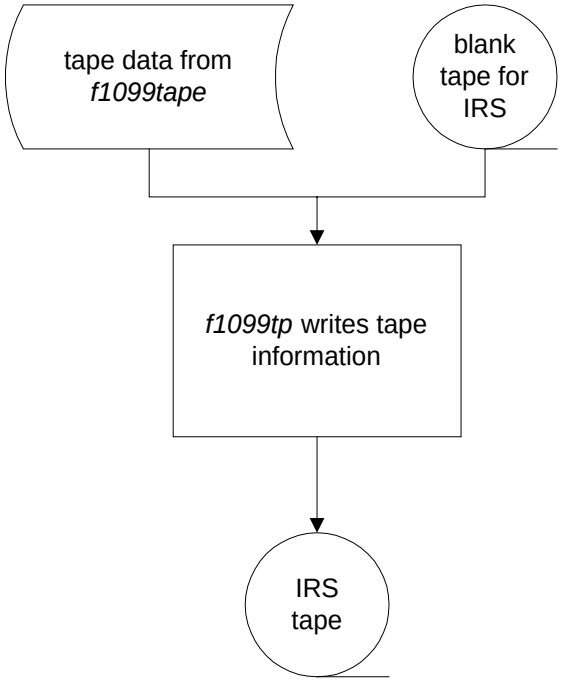
Records and Tables Used in the Program

The *f1099tp* program does not use any CX records or tables.

Process Flow

Diagram

The following diagram shows the flow of data in the *f1099tp* program.



Data Flow Description

The following describes the data flow in the *f1099tp* program.

1. The *f1099tape* program prepares tape information.
2. The user selects the Write Tape menu option that executes the *f1099tp* program.
3. The *f1099tp* program writes the information onto the magnetic tape that you send to the IRS, based on the user's responses to prompts as in the following example:

   Example: f010934.dat 02/12/97 12:15 - write this file (y or n)?
4. If the prompt displays the correct date and time, and the IRS tape is correctly mounted on the tape drive, the user enters **Y**.
5. The *f1099tp* program writes the information to the tape, and returns the user to the menu.

6. The user removes the tape from the tape drive and prepares the tape label for the IRS, using the following field descriptions.
    - Checkbox
      Always the *Unlabeled* box. In addition to this box, the user must also specify *Unlabeled processing* on form 4804.
    - Recording Code
      Always ASCII.
    - Tape Density
      Always 1600.
      Total number of 'B' records
      The number that *Create 1099 Tape File* reported in electronic mail.
    - Track
      Always 9.
    - Transmitter Control Code
      The five character code assigned by the IRS. If the user did not receive this code, the IRS can provide it.

Program Relationships

The following programs use *f1099tp*.
The *f1099tape* program provides input tape information for *f1099tp.*

SECTION 15 - GROUP SELECT

Overview

Introduction

This section provides you with reference information about the Group Select (*grpselect*) program. The Accounts Payable product uses *grpselect* to produce grouping sheets and schedules for state funded expenditures. The institution submits the reports produced by *grpselect* to the state comptroller's office.

Program Features Detailed

This section contains details about the following features of the *grpselect* program:

- Process flow
- Parameters
- Program screens
- Detail windows

Program Screens

Because *grpselect* is a background process, it does not use program screens.

Records and Tables Used

The *grpselect* program uses the following tables and records:

doc_table
The Document table that contains information about document codes and stations

fs_table
The Financial Statement table that organizes blocks, groups and schedules of accounts for financial reporting purposes

gla_rec
The General Ledger Account record that contains the fund, function, object and subfund combinations that your institution has used

gltr_rec
The General Ledger Transaction record that contains the amount and account charged for each transaction in an entry

grphist_rec
The Grouping Sheet History record that contains history information of grouping sheet runs

grpreq_req
The Group Request record that stores information about groups to be printed

id_rec
The Identification record that contains information about each individual or entity in the CX database

invhead_rec
The Invoice Header record that contains general information about each invoice

obj_table
The Object table that defines valid object codes

pohead_rec
The Purchase Order Header record that contains general information about each purchase order

suba_rec
The Subsidiary Account record that contains information about the subsidiary number

subb_rec
The Subsidiary Balance record that contains summary information per period for subsidiary accounts or invoices for accounts payable subsidiary accounts

sube_rec
The Subsidiary Entry record that contains information about postings to the subsidiary accounts

subs_table
The Subsidiary table that defines valid subsidiary codes

subtr_rec
The Subsidiary Transaction record that contains detailed transactions for subsidiary account posting

vch_rec
The General Ledger Journal record that contains information about the journal

vnd_rec
The Vendor record that contains information about the vendors that the institution uses to obtain goods and services

Program Relationships

The *bgvoucher* program posts output from *grpselect*.

Library Relationships

The *grpselect* program uses the *libbgv* library.

Group Select Parameters

Introduction

CX contains parameters and compilation values for executing the *grpselect* program. You can specify parameters to compile *grpselect* in a specified manner at the time of execution.

Note: You can also specify compilation values with the includes for the financial products that affect *grpselect*.

Parameter Syntax

You can create a mail message that lists the *grpselect* parameters by entering the following: **grpselect -,**

The following is the correct usage for running the *grpselect* program from the UNIX shell:

Usage: grpselect -s subsidiary [-d run_date] [-a alt_addr] -c doc_code

Parameters that appear in brackets are optional. Parameters that do not appear in brackets are required.

Parameters

The following lists the parameters for running *grpselect*.

-s subsidiary
Required - Specifies the subsidiary type for grouping sheet selection

-d run_date
Optional - Specifies the run date of grouping sheets

-a alt_addr
Optional - Specifies the alternate address run code

-c doc_code
Optional - Specifies the document code for grouping sheets

SECTION 16 - ACCOUNTS PAYABLE: R1099 BUILD

Overview

Introduction

This section provides you with reference information about the r1099 Build (*r1099bld*) program. The Accounts Payable product uses *r1099bld* to populate the 1099r record with year-end or period-end information. The system uses values coded in the file $CARSPATH/macros/custom/r1099 to complete the 1099 records.

Program Features Detailed

This section contains details about the following features of the *r1099bld* program:

- Process flow
- Parameters

Program Screens

Because *r1099bld* is a background process, it does not use program screens.

Records and Tables Used

The *r1099bld* program uses the following records and tables:

1099_table
The Form 1099 table that defines valid 1099 form types

f1099_rec
The f1099 record that contains information about 1099 reporting

id_rec
The Identification record that contains information about each individual or entity in the CX database

r1099_rec
The 1099r record that contains information for processing 1099r earnings statements

w2_rec
The W-2 record that contains the information for processing W-2 forms and tapes

r1099 Build Parameters

Introduction

The CX contains parameters and compilation values for executing the *r1099bld* program. You can specify parameters to compile *r1099bld* in a specified manner at the time of execution.

Note: You can also specify compilation values with the includes for the financial products that affect the *r1099bld* program.

Parameter Syntax

You can display *r1099bld* parameters by entering the following: **r1099bld -,**

The following is the correct usage for running the *r1099bld* program from the UNIX shell:

Usage: r1099bld -y process year

Since the parameter does not appear in brackets, it is required input for the process.

Parameters

The following lists the parameters for running *r1099bld*.

-y process year
Required - Specifies the calendar year for which you want to process information

SECTION 17 - ACCOUNTS PAYABLE: R1099 FORM

Overview

Introduction

This section provides you with reference information about the r1099 Form (*r1099form*) program. The *r1099form* program uses initialized 1099R data to format 1099R forms for printing. If no initialized 1099R data exists when *r1099form* executes, the program informs the user through email.

Program Features Detailed

This section contains details about the following features of the *r1099form* program:

- Process flow
- Parameters

Records and Tables Used

The *r1099form* program uses the following records and tables:

id_rec
The Identification record that contains information about each individual or entity in the CX database

r1099_rec
The 1099r record that contains information for processing 1099r earnings statements

r1099 Form Parameters

Introduction

The CX contains parameters and compilation values for executing the *r1099form* program. You can specify parameters to compile *r1099form* in a specified manner at the time of execution.

Parameter Syntax

You can display *r1099form* parameters by entering the following: **r1099form -,**

The following is the correct usage for running the *r1099form* program from the UNIX shell:

Usage: r1099form -y year [-a run code] [-i id]

Parameters that appear in brackets are optional. Parameters that do not appear in brackets are required.

Parameters

The following lists the parameters for running *r1099form*.

-y year
Required - Specifies the calendar year for which the user wants to create forms

-a run code
Optional - Specifies the *adr* run code

-i id
Optional - Specifies the ID number for which the user wants to create forms

SECTION 18 - R1099 TAPE

Overview

Introduction

This section provides you with reference information about the r1099 Tape (*r1099tape*) program. The Accounts Payable product uses *r1099tape* to create tape information for 1099r forms.

Program Features Detailed

This section contains details about the following features of the *r1099tape* program:

- Process flow
- Parameters

Program Screens

Because *r1099tape* is a background process, it does not use program screens.

Records and Tables Used in the Program

The *r1099tape* program uses the following tapes and records:

1099_table
The Form 1099 table that defines valid 1099 form types

ctry_table
The Country table that defines valid country names and codes

id_rec
The Identification record that contains information about each individual or entity in the CX database

r1099_rec
The 1099r record that contains information for processing 1099r earnings statements

vnd_rec
The Vendor record that contains information about the vendors that the institution uses to obtain goods and services

SECTION 19 - CASHIER

Overview

Introduction

This section provides you with reference information about the Cashier (*cashier*) program. Financial system users use *cashier* to:

- Receive and disburse cash
- Perform simple banking functions (e.g., cashing checks and making change)
- Print cash or gift receipts
- View student account information (by accessing *bursar*)
- View or print a student's billing history on a statement
- Confirm(i.e., financially clear) students on those campuses that do a confirmation process
- Add or update holds on a student's records
- Close or reconcile the cash drawer
- Transfer cash into or out of the cash drawer

Program Features Detailed

This section contains details about the following features of the *cashier* program:

- Process flow
- Parameters
- Program screens
- Detail windows

Tables and Records Used in the Program

The *cashier* program uses the following tables and records:

adr_rec
The Addressing record that defines the type of addressing to use for the run code, alternate address, and individual specified

cashent_table
The Cash Entry table that defines valid cashier entry types and associated defaults

chrecon_rec
The Cashier Reconciliation record that contains information about the reconciliation status of General Ledger transactions

cw_rec
The Coursework record that contains information about a course completed, in progress, or scheduled to be taken by a student

doc_table
The Document table that contains information about document codes and stations

ent_table
The Entry table that contains information about valid general ledger entry types

fscl_cal_rec
The Fiscal Calendar record that defines fiscal calendar years and periods

gla_rec
The General Ledger Account record that contains the fund, function, object and subfund combinations that your institution has used

gle_rec
The General Ledger Journal Entry record that contains information about each entry

gltr_rec
The General Ledger Transaction record that contains the amount and account charged for each transaction in an entry

id_rec
The Identification record that contains information about each individual or entity in the CX database

payfrm_table
The Form of Payment table that defines valid forms of payment codes

payterm_table
The Payment Terms table that defines valid payment terms and the valid discounts for each term

profile_rec
The Profile record that contains personal information for individuals for whom the institution maintains an id_rec

progenr_rec
The Program Enrollment record that contains individual student academic program information

sbcust_rec
The Student Billing Custom record that contains client-specific field customizations for automated student billing

sdsform_table
The SDS Output Form table that defines what forms to include on SDS (Student Data Sheet) output

stu_acad_rec
The Student Academic record that contains individual student academic information for each session for which the student is enrolled

stufa_rec
The Student Financial Aid record that contains general student financial aid information within an award year

stuserv_rec
The Student Services record that contains individual student information for each academic session

suba_rec
The Subsidiary Account record that contains information about the subsidiary number

subas_table
The Subsidiary Association table that contains information about the relationships between balance and total codes

subb_rec
The Subsidiary Balance record that contains information about a subsidiary balance transaction

subb_table
The Subsidiary Balance table that defines valid balance codes

sube_rec
The Subsidiary Entry record that contains information about each entry that impacts a subsidiary

subs_table
The Subsidiary table that contains information about subsidiaries

subt_rec
The Subsidiary Total record that contains one type of summary information for a subsidiary

subt_table
The Subsidiary Total table that contains information about subsidiary tot codes

vch_rec
The General Ledger Journal record that contain information about the journal

vch_table
The Journal table that contains information about the valid journal types

Process Flow

Data Flow Description

The following describes the data flow in the *cashier* program.

1. The program loads all table information from the various tables.
2. To post an entry, the program verifies that the accounts being used are valid accounts according to the gld_rec. It the account is not a subsidiary, the program will do the normal posting to the GL records. If it is a subsidiary then additional information will be requested, such as the period and tot code to which to post.
3. After transactions are posted, the program prints a receipt and updates all balances.
4. To query on a student, *cashier* will switch to the Bursar Query program. The user can display current billing information, past billing information, do confirmations and print a statement.
5. At the end of the processing session (typically a workday), the program reads the ckrecon_recs that have been created while posting was done and allows the user to reconcile the cash drawer. When the cash drawer is reconciled the system automatically posts the deposit amount to the bank account and removes it from the cash drawer.

Program Relationships

The following programs interrelate with *cashier*:

Bursar Query
The *cashier* program calls the Bursar Query (*bursar)* program and allows the user to display past information on the student, print a historic statement and do confirmation (financial clearance).

Billing
The cashier program calls the Student Billing (*billing*) program in -b (bursar) mode.

Background Voucher
The *cashier* program uses the Background Voucher (*bgvoucher*) program to post cash transactions.

Library Relationships

The *cashier* program uses the following libraries:

- libacct
- libbgv
- libbill
- libfee
- libgl
- Librecon

Cashier Parameters

Introduction

The CX contains parameters and compilation values for executing *cashier*. You can specify parameters to compile *cashier* in a specified manner at the time of execution.

Setting Macros and Parameters

To initiate the features of *cashier*, you must set up the following macros and parameters:

1. m4 macros
2. C program macros
3. C program parameters

CAUTION: You *must* set the macros and parameters in the order listed above.

Parameter Syntax

You can display *cashier* parameters by entering the following: **cashier -,**

The following is the correct usage for running the *cashier* program from the UNIX shell:

Usage: cashier -[-a adr code] [-b compute billing] [-c] -d document code [-f sds form code]
[-g program code] -j journal code [-l] -n station number [-o statement device]
[-p program date] [-r receipt device] [-s balance period] [-v] [-V] [-z]

Parameters that appear in brackets are optional. Parameters that do not appear in brackets are required.

Parameters

The following lists the parameters for running *cashier*.

-a adr code

Optional - Specifies the alternate address code for printing address information on cash receipts and on statements. Unless overridden by this parameter, *cashier* uses the value of the C program macro DEF_RECPT_RUNCODE as the default alternate address code.

-b compute billing

Optional - Displays billing balances (e.g., include non-posted amounts). Enter the value **Y** to reflect both posted and non-posted amounts on the student subsidiary account balance. Enter the value **N** to reflect only posted amounts on the student subsidiary account balance.

-c

Optional - Specifies whether or not financial clearance can be set within the cashier program. Omit this parameter if you do not want to allow setting of financial clearance.

-d document code

Required - Specifies the document code for cash receipts. The value of the m4 Macro DOC_CR_DEF is the default document code.

CAUTION: The Cashier program requires this parameter to load properly.

-j sds form code

Optional - Specifies the journal code. The value of CH is hard-coded as the default journal code.

CAUTION: The Cashier program requires this parameter to load properly. You can override the hard-coded value by using this code when executing *cashier*.

-l

Optional - Specifies for *cashier* to post multiple entries without re-selecting the cash transaction type.

Do you want to set *cashier* in looping mode?

- If yes, enter this parameter, and you do not have to re-select cash transaction type for each entry.
- If no, omit this parameter to set *cashier* in non-looping mode, which requires you to re-select cash transaction type for each entry.

-n station number

Required - Specifies and reserves the document station number.

CAUTION: The *cashier* program requires this parameter to load properly.

-o statement device

Optional - Specifies the printer to use when printing statements. Unless overridden by this parameter, the default value is the environment variable CARSPRINTER.

-p program date

Optional - Specifies the default posting date for *cashier* transactions (e.g., default journal posting date). Unless overridden by this parameter, the default value is the current date.

-r receipt device

Optional - This parameter specifies the printer to use when printing cash receipts. Unless overridden by this parameter, the default value is the value of the environment variable CARSPRINTER.

-s balance period

Optional - Specifies the balance period (session) code to override normal defaulting. To specify the balance period (session) code, enter the parameter as in the following example:

Example: Enter SP97 to force remittance amounts to default against the SP97 balance. If an SP97 balance does not exist, the system adds one. The amounts default to SP97balance even if the current balance period is FA96.

Note: If you do not enter this parameter, you invoke the standard table-driven method of providing the default.

-v

Optional - Specifies whether or not verification should take place before posting. Enter this parameter to require a **Y** (yes) response to the following prompt (before posting occurs): "Are you sure you want to post (Y, N)?" When you enter this parameter, *cashier* also checks your list of incomplete journals before automatically starting a journal. If the list contains a journal that can be continued in the current station, before starting a journal, *cashier* displays the following prompt: "Do you want to Continue or Start a journal (C, N)?"

-V

Optional - Specifies verbose mode (used for debugging purposes only).

Note: When you invoke this parameter, diagnostic information appears on the screen when you load *cashier*. Also, *cashier* sends a progress report of program execution to you though electronic mail.

-z

Optional - Specifies debug mode (used for debugging purposes only). When you invoke this parameter, transaction information appears before posting occurs.

Program Screens and Windows

Introduction

The *cashier* program uses 36 screens or windows that provide help or enable users to enter and view parameters or data.

Access

The screen files appear in both of the following directory paths:

- $CARSPATH/modules/accounting/progscr/cashier
- $CARSPATH/src/accounting/cashier/SCR

Screen Files and Table/Record Usage

The *cashier* screens appear in the following locations and use the indicated tables and records:

adjbal
Contains the Balance window

Access: $CARSPATH/src/accounting/cashier/SCR/adjbal

Tables/Records: none

adjtot
Contains the Total window

Access: $CARSPATH/src/accounting/cashier/SCR/adjtot

Tables/Records: none

adjust
Contains the G/L Adjustments window

Access: $CARSPATH/src/accounting/cashier/SCR/adjust

Tables/Records: gla_rec, suba_rec

balinfo
Contains the untitled window that displays the period, code, subsidiary and balance

Access: $CARSPATH/src/accounting/cashier/SCR/balinfo

Tables/Records: subb_rec

balprd
Contains the Balance Periods window

Access: $CARSPATH/src/accounting/cashier/SCR/balprd

Tables/Records: none

ccmd
Contains the Cash Option help screen

Access: $CARSPATH/modules/accounting/progscr/cashier/ccmd

Tables/Records: none

clos
Contains the Closing Function Command help screen

Access: $CARSPATH/modules/accounting/progscr/cashier/clos

Tables/Records: none

close
Contains the Cash Drawer Closing screen

Access: $CARSPATH/src/accounting/cashier/SCR/close

Tables/Records: gla_rec, gltr_rec

cmds
Contains the Function Overview help screen

Access: $CARSPATH/modules/accounting/progscr/cashier/cmds

Tables/Records: none

continue
Contains the Incomplete Journals window

Access: $CARSPATH/src/accounting/cashier/SCR/continue

Tables/Records: vch_rec

dcmd
Contains the Drawer Option help screen

Access: $CARSPATH/modules/accounting/progscr/cashier/dcmd

Tables/Records: none

disc
Contains the Other Receivables Information window

Access: $CARSPATH/src/accounting/cashier/SCR/disc

Tables/Records: subb_rec

drawer
Contains the untitled window that displays the journal, journal number, document type and document number

Access: $CARSPATH/src/accounting/cashier/SCR/drawer

Tables/Records: none

entry
Contains the main Cashier entry screen

Access: $CARSPATH/src/accounting/cashier/SCR/entry

Tables/Records: cashent_table, id_rec, payfrm_table, suba_rec

glacct
Contains the G/L Information window

Access: $CARSPATH/src/accounting/cashier/SCR/glacct

Tables/Records: obj_table, func_table, fund_table, gltr_rec, subfund_table, suba_rec

gld
Contains the Short Hand Account Codes window

Access: $CARSPATH/src/accounting/cashier/SCR/gld

Tables/Records: gld_rec

help
Contains the main help screen for *cashier*

Access: $CARSPATH/src/accounting/cashier/SCR/help

Tables/Records: none

instfee
Contains the Institutional Fees screen

Access: $CARSPATH/src/accounting/cashier/SCR/instfee

Tables/Records: id_rec, payterm_table, sbcust_rec, subb_rec

jcmd
Contains the Journal Option help screen

Access: $CARSPATH/modules/accounting/progscr/cashier/jcmd

Tables/Records: none

ocmd
Contains the Other Option help screen

Access: $CARSPATH/modules/accounting/progscr/cashier/ocmd

Tables/Records: none

otbl
Contains the Other Option - Reload Table help screen

Access: $CARSPATH/modules/accounting/progscr/cashier/otbl

Tables/Records: none

param
Contains the Parameters window

Access: $CARSPATH/src/accounting/cashier/SCR/param

Tables/Records: doc_table, fscl_cal_rec, gla_rec, sdsform_table

parm
Contains the Parameter help screen

Access: $CARSPATH/modules/accounting/progscr/cashier/parm

Tables/Records: none

pcmd
Contains the Disburse Option help screen

Access: $CARSPATH/modules/accounting/progscr/cashier/pcmd

Tables/Records: none

qcmd
Contains the Query Option help screen

Access: $CARSPATH/modules/accounting/progscr/cashier/qcmd

Tables/Records: none

rcmd
Contains the Receipt Option help screen

Access: $CARSPATH/modules/accounting/progscr/cashier/rcmd

Tables/Records: none

recn
Contains the Reconciliation Option help screen

Access: $CARSPATH/modules/accounting/progscr/cashier/recn

Tables/Records: none

recondtl
Contains the untitled window that displays a list of items to be reconciled

Access: $CARSPATH/src/accounting/cashier/SCR/recondtl

Tables/Records: gle_rec, pay_frm_table

reconsum

Contains the untitled screen that contains form of payment, unreconciled, reconciled and combined amount information

Access: $CARSPATH/src/accounting/cashier/SCR/reconsum

Tables/Records: gla_rec, gltr_rec, pay_frm_table

recptfrm

Contains the Cash Receipt Forms window

Access: $CARSPATH/src/accounting/cashier/SCR/recptfrm

Tables/Records: none

recptprm

Contains the Cash Receipt Parameters window

Access: $CARSPATH/src/accounting/cashier/SCR/recptprm

Tables/Records: none

scmd

Contains the Statements Option help screen

Access: $CARSPATH/modules/accounting/progscr/cashier/scmd

Tables/Records: none

subsbal

Contains the Subsidiary Balance Information window

Access: $CARSPATH/src/accounting/cashier/SCR/subsbal

Tables/Records: stu_acad_rec, subb_rec

substot

Contains the Subsidiary Total Information window

Access: $CARSPATH/src/accounting/cashier/SCR/substot

Tables/Records: subt_rec

tcmd

Contains the Transfer Option help screen

Access: $CARSPATH/modules/accounting/progscr/cashier/tcmd

Tables/Records: none

void

Contains the Entries in Current Journal window

Access: $CARSPATH/src/accounting/cashier/SCR/void

Tables/Records: gle_rec, id_rec

SECTION 20 - CHECK RECONCILIATION

Overview

Introduction

This section provides you with reference information about the Check Reconciliation (*ckrecon*) program. The financial products use *ckrecon* to update outstanding general ledger accounting transactions to a status of *Reconciled*.

The Check Reconciliation program allows users to reconcile any account. Its primary function is bank statement reconciliation. The program will not allow the user to complete (finish) the reconciliation until the reconciliation amount is zero. The user can suspend the reconciliation process from one day to the next, and can use the Outstanding Items report in performing reconciliations. Only one Check Reconciliation record can be open (i.e., not finished) for an account at a time.

CX has the ability to allow the user to send the recon_rec information to the bank for automatic reconciliation and/or receive a reconciliation tape from the bank, and to have the system, with locally customized processes, do the reconciliation without user interaction.

Program Features Detailed

This section contains details about the following features of the *ckrecon* program:

- Process flow
- Parameters
- Program screens
- Detail windows

Records and Tables Used

The *ckrecon* program uses the following records and tables:

fscl_cal_rec
The Fiscal Calendar records that define fiscal calendar years and periods

gla_rec
The General Ledger Account record that contains the fund, function, object and subfund combinations that your institution has used

gl_amt_rec
The General Ledger Amount record that contains summarized amounts over fiscal periods

gle_rec
The General Ledger Journal Entry record that contains information about each entry

gltr_rec
The General Ledger Transaction record that contains the amount and account charged for each transaction in an entry

id_rec
The Identification record that contains information about each individual or entity in the CX database

precon_rec
The Statement Parameter record that must be present with an Incomplete status before *ckrecon* can be run

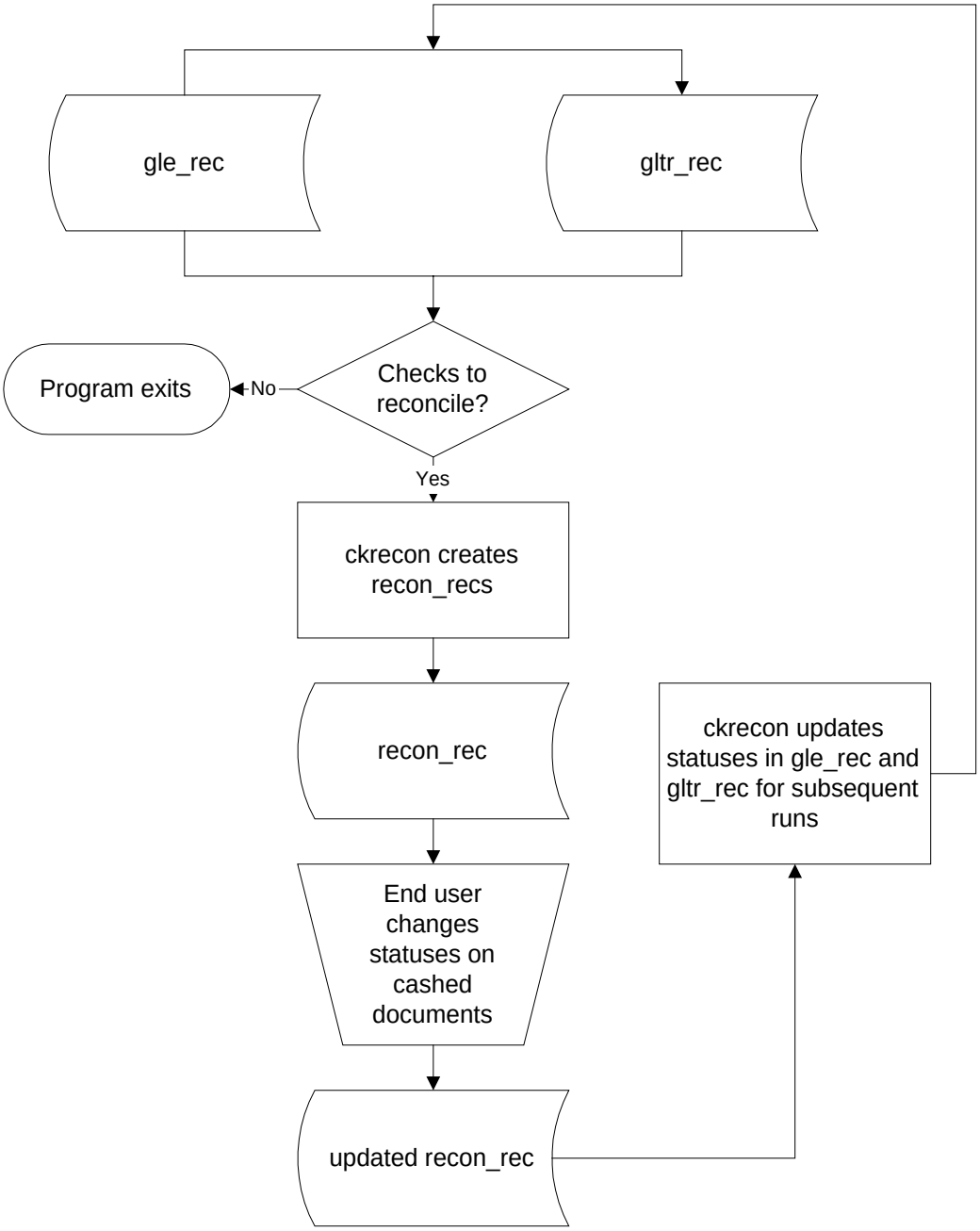
recon_rec
The Check Reconciliation record that contains the data in the accounting system to prevent normal system activity from affecting the reconciliation process

vch_rec
The General Ledger Journal record that contains information about the journal

Process Flow

Diagram

The following diagram shows the flow of data in the *ckrecon* program.



Data Flow Description

The following describes the data flow in the *ckrecon* program.

1. The *ckrecon* program reads the gle_rec and gltr_rec to create the recon_rec.
2. The user accesses the recon_recs and changes the status of the cashed documents from (O)utstanding to (R)econciled.
3. When the reconciliation is complete, the program updates the gle_rec and gltr_rec to a status of (R)econciled. All outstanding transactions will be selected by the program until they have been flagged as (R)econciled.

Program Relationships

There are no programs that call the *ckrecon* program.

Library Relationships

The *ckrecon* program uses the following library: *libgl*

Check Reconciliation Parameters

Introduction

The CX contains parameters and compilation values for executing the *ckrecon* program. You can specify parameters to compile *ckrecon* in a specified manner at the time of execution.

Note: You can also specify compilation values with the includes for the financial products that affect the *ckrecon* program.

Parameter Syntax

You can display *ckrecon* parameters by entering the following: **ckrecon -,**

The following is the correct usage for running the *ckrecon* program from the UNIX shell:

Usage: ckrecon -g group [-p load] [-v]

Parameters that appear in brackets are optional. Parameters that do not appear in brackets are required.

Parameters

The following lists the parameters for running *ckrecon*.

-g group
Required - Specifies the reconciliation group number

-p load
Optional - Specifies if the user wants to load the previous year's documents

-v
Optional - Specifies if the user wants to verify (i.e., initialize) the reconciliation process

Program Screens

Introduction

The *ckrecon* program uses two screens to capture and display data about reconciling transactions.

To use *ckrecon* the user must first setup the precon_rec using the PERFORM screen found in $CARSPATH/accounting/screens/precon.

Access

The screen files are located in the following directory path: $CARSPATH/src/accounting/ckrecon/SCR.

Screen Files and Table/Record Usage

The *ckrecon* screens appear in the following files and use the indicated tables and records:

ckrecon
Contains the Reconcile Documents screen

Tables/Records: gla_rec, precon_rec, recon_rec

precon
Contains the Reconciliation Parameter Record PERFORM screen

Tables/Records: gla_rec, precon_rec

SECTION 21 - DOCUMENT VOIDING

Overview

Introduction

This section provides you with reference information about the Document Voiding (*docvoid*) program. The financial products use *docvoid* to void checks and other documents (e.g., receipts) that have no accounting significance.

Note: The process does not reverse the payroll liability associated with voided payroll checks.

Program Features Detailed

This section contains details about the following features of the *docvoid* program:

- Process flow
- Parameters

Tables and Records Used in the Program

The *docvoid* program uses the following tables and records:

doc_table
The Document table that contains information about document codes and stations

gle_rec
The General Ledger Entry record that contains information about each entry

gltr_rec
The General Ledger Transaction record that contains the amount and account charged for each transaction in an account

id_rec
The Identification record that contains information about each individual or entity in the CX database

sube_rec
The Subsidiary Entry record that contains information about postings to the subsidiary accounts

subtr_rec
The Subsidiary Transaction record that contains detailed transactions for subsidiary account posting

vch_rec
The General Ledger Journal records that contain information about the journal

Program Screens

Introduction

The *docvoid* program uses two screens to capture and display data about reconciling transactions.

Access

The screen files are located in the following directory path: $CARSPATH/src/accounting/docvoid/SCR.

Screen Files and Table/Record Usage

The *docvoid* screens appear in the following files and use the indicated tables and records:

multdoc

Contains the Multiple Entries for Document screen

Tables/Records: gle_rec, id_rec, vch_rec

voidscr

Contains the main Document Voiding screen

Tables/Records: doc_table

SECTION 22 - FIXED ASSETS: FIXED ASSET POSTING

Overview

Introduction

This section provides you with reference information about the Fixed Asset Posting (*fixpost*) program. The financial products use *fixpost* to create acquisition, depreciation and disposal entries for the long-lived assets at the institution.

Program Features Detailed

This section contains details about the following features of the *fixpost* program:

- Process flow
- Parameters
- Screens

Records and Tables Used

The *fixpost* program uses the following records and tables:

fix_rec
The Fixed Asset record that contains information about fixed assets, including location, depreciation details and authorizations

fix_table
The Fixed Assets table that defines the different types of fixed assets that are valid at the institution

fscl_cal_rec
The Fiscal Calendar record that defines fiscal calendar years and periods

gle_rec
The General Ledger Entry record that contains information about each entry

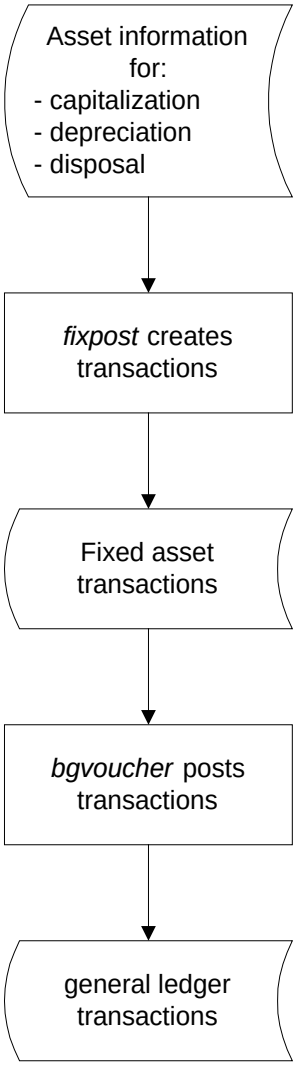
gltr_rec
The General Ledger Transaction record that contains the amount and account charged for each transaction in an account

vch_rec
The General Ledger Journal records that contain information about the journal

Process Flow

Diagram

The following diagram shows the flow of data in the *fixpost* program.



Data Flow Description

The following describes the data flow in the *fixpost* program.

1. The *fixpost* program selects all fix_recs. The fix_recs are grouped by asset type, and the fix_table validates the types.
2. If the Summary Status flag in fix_rec equals Y, *fixpost* treats all the assets of the type as one entity.
3. The *fixpost* program creates one general ledger entry and supporting general ledger transactions for each unsummarized asset, and for each group of summarized assets.

Fixed Asset Posting Parameters

Introduction

The CX contains parameters and compilation values for executing the *fixpost* program. You can specify parameters to compile *fixpost* in a specified manner at the time of execution.

Note: You can also specify compilation values with the includes for the financial products that affect the *fixpost* program.

Parameter Syntax

You can display *fixpost* parameters by entering the following: **fixpost -,**

The following is the correct usage for running the *fixpost* program from the UNIX shell:

Usage: fixpost -c calc date -d post date [-n asset num] [-p period] [-r]

Parameters that appear in brackets are optional. Parameters that do not appear in brackets are required.

Parameters

The following lists the parameters for running *fixpost*.

-c calc date
Required - Specifies the date (mm/dd/yyyy) that you want to use to compute depreciation.

-d post date
Required - Specifies the date (mm/dd/yyyy) that you want to use on the journal entry.

-n asset num
Optional - Specifies the system-assigned sequence number associated with the asset.

-p period
Optional - Specifies the fiscal posting period to which you want to post the transactions (e.g., 0196).

-r
Optional - Indicates that you want to verify information only, without posting any transactions.

Program Screens

Introduction

The *fixpost* program uses two screens that enable users to enter and view data.

Access

The screen files appear in the following directory path: $CARSPATH/modules/fixassets/screens

Screen Files and Table/Record Usage

The *fixpost* screens appear in the following locations and use the indicated tables and records:

fix

Contains the Asset Information screen, Asset Valuation screen, and the Authorization/Maintenance Information screen

Tables/Records: bldg_table, fix_rec, fix_table, fixmaint_rec, func_table, fund_table, id_rec, obj_table, subfund_table

tfix

Contains the Fixed Asset Type Table screen

Tables/Records: fix_table, func_table, fund_table, obj_table, subfund_table

SECTION 23 - FINANCIAL BUDGETING: BUDGET ALLOCATION

Overview

Introduction

This section provides you with reference information about the Budget Allocation (*bgtalloc*) program. The Budget product uses *bgtalloc* to allocate and summarize budget amounts. Users access the budget records in *bgtalloc* to work in a test mode until they finalize their budget figures and install the budget.

Program Features Detailed

This section contains details about the following features of the *bgtalloc* program:

- Process flow
- Parameters
- Program screens

Records and Tables Used

The *bgtalloc* program uses the following records and tables:

atype_table
The Amount Type table that contains information about the types of amounts that you maintain on your CX database (e.g., ACT, BGT, REQ, APP, PRJ)

bgtamt_rec
The Budget Amount record that contains budgeted and actual amounts by general ledger account number

bgtcal_rec
The Budget Calendar record that contains fiscal year and amount type combinations that users can access or update for financial budgeting

bgtsum_rec
The Budget Summarization record that contains summarized budget information by blocks, groups and schedules

fs_table
The Financial Statement table that organizes blocks, groups and schedules of accounts on financial statements

func_table
The Function table that defines the institution's valid function codes

fund_table
The Fund table that defines the institution's valid fund codes

gla_rec
The General Ledger Account record that contains the fund, function, object and subfund combinations that your institution has used

gld_rec
The General Ledger Definition record that defines rules used in validating new accounts; also contains codes that allow single-keystroke account entry (shortcut codes)

obj_table
The Object table that defines the valid object codes for your institution

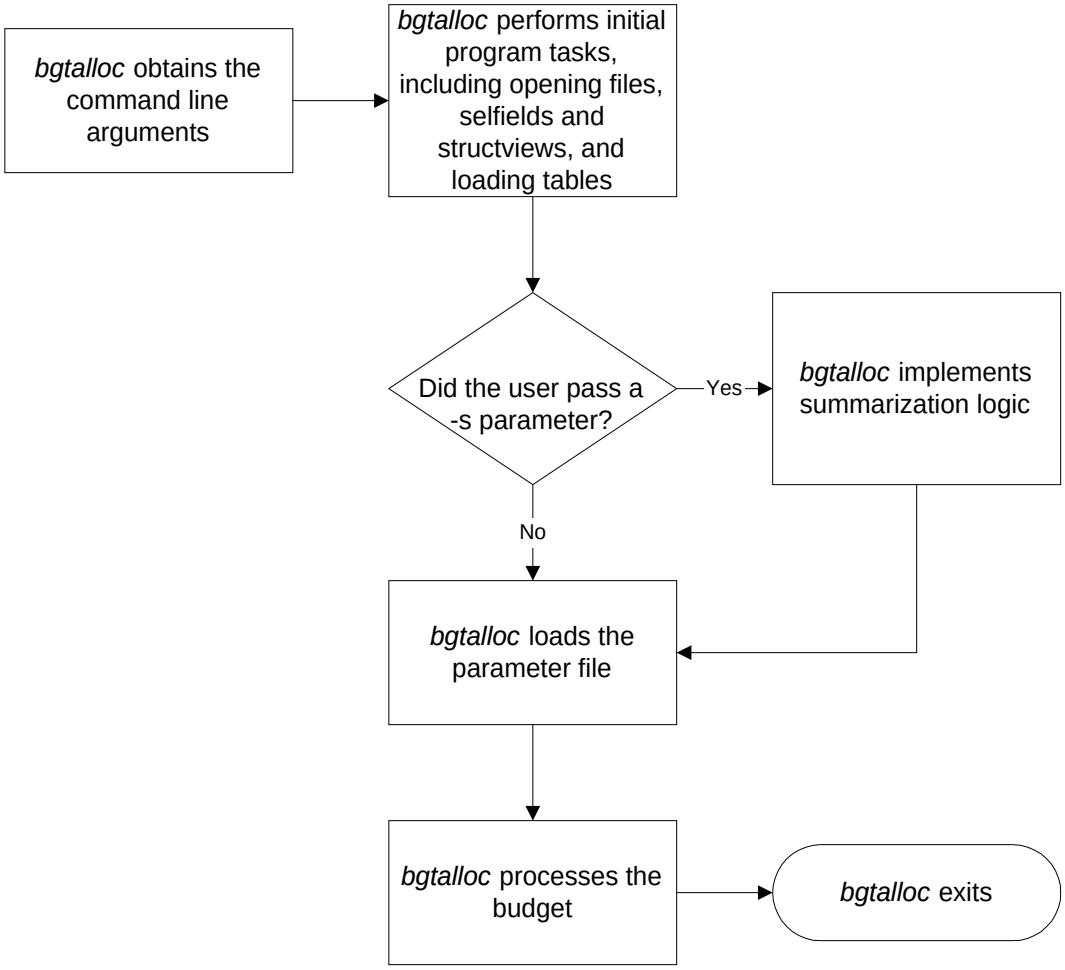
pbgt_rec
The Budget Parameter record that contains parameters used by *bgtalloc*

subfund_table
The Subfund table that defines the valid subfund codes for your institution

Process Flow

Diagram

The following diagram shows the flow of data in the *bgtalloc* program.



Data Flow Description

The following describes the data flow in the *bgtalloc* program.

1. The user launches *bgtalloc*.
2. The *bgtalloc* program performs initial program tasks.
3. If the user selected summarization, *bgtalloc* summarizes the information for display purposes. Generally for data entry, users do not select –s summarization; summarization is only done when the data is created in mass by *bgtbasis*.
4. For budget entry, the user selects the desired parameter record. The parameter record presents four columns of information. The four columns look at the bgtamt_rec information.
5. Based on permissions in the atype_table, the data is updated.
6. The user changes the data on the screen and writes the data to the database. At that point, the program updates the bgtamt_rec and the bgtsum_rec totals.

Program Relationships

The following programs interact with *bgtalloc*:

- The *bgtbasis* program provides historical financial information that is used by the *bgtalloc* program.
- The users use the *bgtalloc* program to create the final budget figures that serve as input for the *bgtinstall* program.

Library Relationships

The *bgtalloc* program uses the following libraries:

- *libgl*
- *libbgt*

Budget Allocation Parameters

Introduction

The CX contains parameters and compilation values for executing the *bgtalloc* program. You can specify parameters to compile *bgtalloc* in a specified manner at the time of execution.

Note: You can also specify compilation values with the includes for the financial products that affect the *bgtalloc* program.

Parameter Syntax

You can display *bgtalloc* parameters by entering the following: **bgtalloc -,**

The following is the correct usage for running the *bgtalloc* program from the UNIX shell:

Usage: bgtalloc [-c parameter code] [-s] [-f fund code] [-l]

Parameters that appear in brackets are optional. Parameters that do not appear in brackets are required.

Parameters

The following lists the parameters for running *bgtalloc*.

-c parameter code

Optional - Specifies the budget parameter code (e.g., 9596). *Bgtalloc* requires a pbgt_rec before you can use the program. The Budget parameter (pbgt_rec) record tells the program what to put in the four columns on the bgt screen. At least two of the columns must contain data from the bgtamt_rec. The other two columns can contain data from the database or can be calculated information (including percent (%) difference, and dollar ($) difference, from one column to the next.)

-f fund code

Optional - Indicates the general ledger fund for which you want to create summarized records.

-l

Optional - Indicates that *bgtalloc* should recalculate summarization records for revenue and expense accounts only.

-s

Optional - Indicates that *bgtalloc* should build summarization records.

Note: For summarization to occur, you must enter **Y** in the Summary field of the bgtcal_rec for the year. This is normally run only after *bgtbasis* has created new/updated information in the bgtamt_rec.

Program Screens

Introduction

The *bgtalloc* program uses four screens to enable users to view and enter budget information.

Access

The screen files are located in the following directory path:
$CARSPATH/src/budget/bgtalloc/SCR

Screen Files and Table/Record Usage

The *bgtalloc* screens appear in the following files and use the indicated tables and records:

addacct
Contains the Add General Ledger Account screen

Tables/Records: func_table, fund_table, gla_rec, obj_table, subfund_table

bgt
Contains the Budget Amount screen.

Tables/Records: bgtamt_rec, pbgt_rec

bgtdtl
Contains Monthly Detail screen

Tables/Records: bgtamt_rec, pbgt_rec

bgtparms
Contains the Budget Parameter screen

Tables/Records: pbgt_rec

SECTION 24 - FINANCIAL BUDGETING: BUDGET BASIS

Overview

Introduction

This section provides you with reference information about the Budget Basis (*bgtbasis*) program. The Budget product uses *bgtbasis* to copy budget information from historical data contained in the gl_amt_rec. The copied data, maintained in workspace in Budget Amount records (bgtamt_rec), serves as a starting point from which users can create the budget for the new year.

Program Features Detailed

This section contains details about the following features of the *bgtbasis* program:

- Process flow
- Parameters

Program Screens

Because *bgtbasis* is a background process, it does not use any program screens.

Records and Tables Used

The *bgtbasis* program uses the following records and tables:

atype_table
The Amount Type table that contains information about the types of amounts that you maintain on your CX database (e.g., ACT, BGT)

bgtamt_rec
The Budget Amount record that contains budgeted amounts by general ledger account number

bgtcal_rec
The Budget Calendar record that contains fiscal year and amount type combinations that users can access or update for financial budgeting

fs_table
The Financial Statement table that organizes blocks, groups and schedules of accounts on financial statements

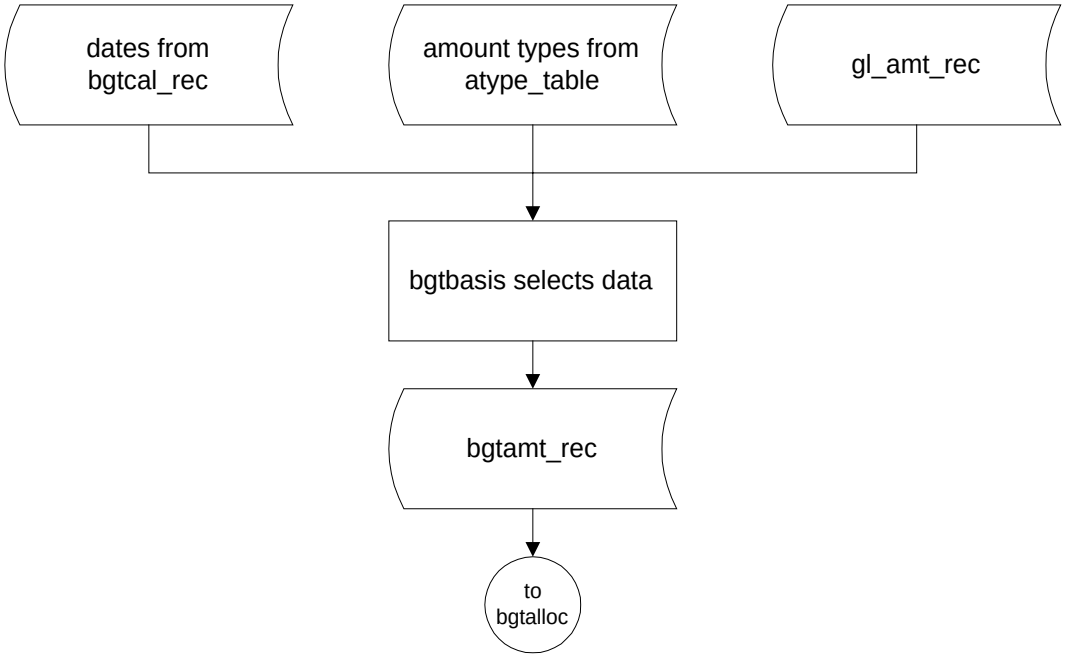
gl_amt_rec
The General Ledger Amount record that provides summarized totals for general ledger accounts over fiscal periods

gld_rec
The General Ledger Definition record that defines rules used in validating new accounts; also contains codes that allow single-keystroke account entry (shortcut codes)

Process Flow

Diagram

The following diagram shows the flow of data in the *bgtbasis* program.



Data Flow Description

The following describes the data flow in the *bgtbasis* program.

1. The *bgtbasis* program reads the bgtcal_rec to verify that the years the user creates are open. This step helps prevent the user from overlaying old data.
2. The *bgtbasis* program reads the atype_table to verify the amount type is active.
3. The *bgtbasis* program reads the gl_amt_rec for the year, fund, and object range requested to obtain the data to copy into the bgtamt_rec.
4. The *bgtbasis* program calls *bgtalloc* with the -s parameter to create the bgtsum_recs.

Program Relationships

The following programs interact with *bgtbasis*.

- The *bgtalloc* program accepts input from *bgtbasis*, providing users with a set of base data to update for the coming budget year.
- The *voucher* program creates the General Ledger Amount records (gl_amt_rec) that serve as input to *bgtbasis*.

Budget Basis Parameters

Introduction

The CX contains parameters and compilation values for executing the *bgtbasis* program. You can specify parameters to compile *bgtbasis* in a specified manner at the time of execution.

Parameter Syntax

The following is the correct usage for running the *bgtbasis* program from the UNIX shell. You can obtain this information from the electronic mail message the program sends you when you enter the command **bgtbasis -,** at the UNIX prompt.

Usage: bgtbasis limit_amts(Y or N) fscl_yr fund amt_types

SECTION 25 - FINANCIAL BUDGETING: BUDGET INSTALL

Overview

Introduction

This section provides you with reference information about the Budget Install (*bgtinstall*) program. The Budgeting product uses *bgtinstall* to post an approved budget into the general ledger. Users run *bgtinstall* when they have finalized their budgets and want to install them as the operating budget for the new accounting period.

Program Features Detailed

This section contains details about the following features of the *bgtinstall* program:

- Process flow
- Parameters

Program screens

Because *bgtinstall* is a background process, it does not use any program screens.

Records and Tables Used in the Program

The *bgtinstall* program uses the following records and tables:

atype_table
The Amount Type table that contains information about the types of amounts that you maintain on your CX database (e.g., ACT, BGT)

bgtamt_rec
The Budget Amount record that contains budgeted amounts by general ledger account number

ent_table
The Entry table that contains information about valid general ledger entry types

fscl_cal_rec
The Fiscal Calendar record that defines fiscal calendar years and periods

gl_amt_rec
The General Ledger Amount record that provides summarized totals for general ledger accounts over fiscal periods

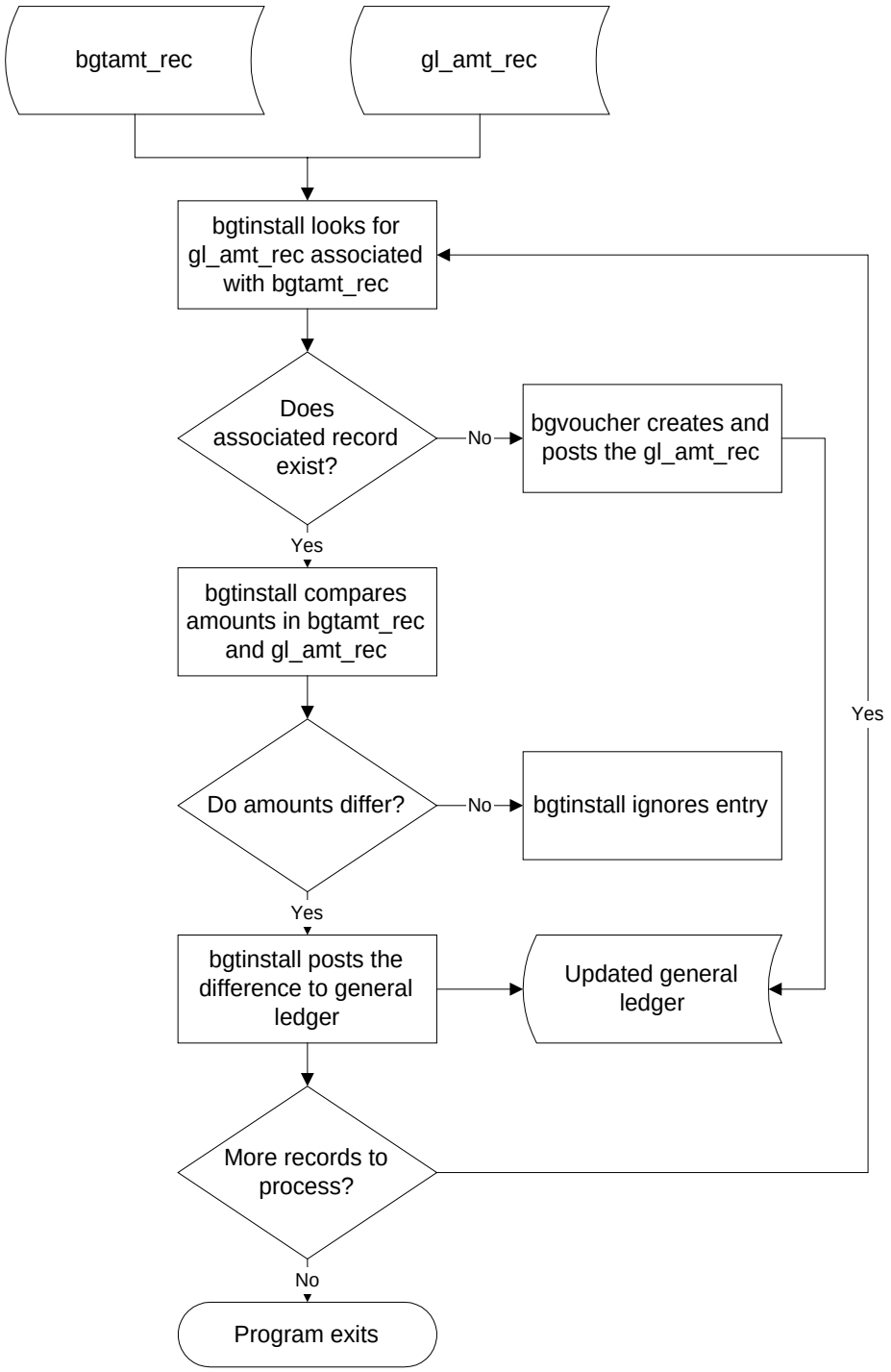
gld_rec
The General Ledger Definition record that defines rules used in validating new accounts; also contains codes that allow single-keystroke account entry (shortcut codes)

vch_rec
The General Ledger Journal record that contain information about the journal

Process Flow

Diagram

The following diagram shows the flow of data in the *bgtinstall* program.



Data Flow Description

The following describes the data flow in the *bgtinstall* program.

1. The *bgtinstall* program reads the bgtamt_rec and the gl_amt_rec.
2. If a gl_amt_rec does not exist for the period being tested, the transaction is passed to *bgvoucher* and posted to the general ledger. If a gl_amt_rec exists, it then compares the amount already posted to the amount in the bgtamt_rec. If the amounts are the same, the program proceeds to the next record. If there is a difference, it posts the difference to the period specified.
3. Once all bgtamt_rec have been processed, it closes the *bgvoucher* program.
4. The *bgtinstall* program will read the specified amount type but will only post to the BGT amount type in the general ledger.

Program Relationships

The following programs use *bgtinstall*.

- The *bgtalloc* program provides input to the *bgtinstall* process by creating the final budget.
- The *bgtinstall* program adds the budget to the general ledger so that the Budget Review (*bgtreview*) program can check budgeted figures against actual or proposed expenditures. For more information about *bgtreview*, see *General Ledger Technical Manual*.

Budget Install Parameters

Introduction

The CX contains parameters and compilation values for executing the *bgtinstall* program. You can specify parameters to compile *bgtinstall* in a specified manner at the time of execution.

Note: You can also specify compilation values with the includes for the financial products that affect the *bgtinstall* program.

Parameter Syntax

You can display *bgtinstall* parameters by entering the following: **bgtinstall -,**

The following is the correct usage for running the *bgtinstall* program from the UNIX shell:

Usage: bgtinstall -f fund -t amt_type -y fscl_yr

Parameters that appear in brackets are optional. Parameters that do not appear in brackets are required.

Parameters

The following lists the parameters for running *bgtinstall*.

-f fund
Required - Specifies the fund for which you want to install a budget

-t amt_type
Required - Specifies the amount type to install

-y fscl_yr
Required - Specifies the fiscal year for which you want to install a budget

SECTION 26 - PURCHASING: APPROVE

Overview

Introduction

This section provides you with reference information about the Approve (*approve*) program. The Purchasing product uses *approve* to track purchase orders that exceed the restrictions placed upon them. The *approve* program allows for subsequent approval or rejection of these purchase orders by budget managers.

Note: This program interacts with the CX programs *purch*, *invoice*, *purchaudit*, and *vndentry*. For information about other CX procurement products (*acctspay*, *assign*, *massinv*, *receive*, *requisition*, and *approval*), see the following documentation:

- *RPA Technical Manual*
- *Using Purchasing and Accounts Payable*
- *Using Requisitioning*

Program Features Detailed

This section contains details about the following features of the *approve* program:

- Process flow
- Parameters
- Program screens
- Detail windows

Records and Tables Used

The *approve* program uses the following records and tables:

appid_rec
The ID Approval record that serves as a temporary holding location for the IDs from which purchase order approvals are required

appr_table
The Approval table that defines the approval hierarchy for approving purchase orders

fscl_cal_rec
The Fiscal Calendar record that defines fiscal calendar years and periods

gla_rec
The General Ledger Account record that contains the fund, function, object and subfund combinations that your institution has used

gle_rec
The General Ledger Journal Entry record that contains information about each entry

gltr_rec
The General Ledger Transaction record that contains the amount and account charged for each transaction in an entry

id_rec
The Identification record that contains information about each individual or entity in the CX database

po_rec
The Purchase Order record that contains information relating to purchase orders, including amounts, dates and vendors

pobody_rec
The Purchase Order Detail record that contains information for the main body, or detail, portion of the purchase order

suba_rec
The Subsidiary Account record that contains information about the subsidiary number

subb_rec
The Subsidiary Balance record that contains summary information per period for subsidiary accounts or invoices for accounts payable subsidiary accounts

sube_rec
The Subsidiary Entry record that contains information about postings to the subsidiary accounts

subs_table
The Subsidiary table that defines valid subsidiary codes

subtr_rec
The Subsidiary Transaction record that contains detailed transactions for subsidiary account posting

userid_table
The User ID table that provides a link between a login user ID number and the database ID number

vch_rec
The General Ledger Journal records that contain information about the journal

Approve Parameters

Introduction

The CX contains parameters and compilation values for executing the *approve* program. You can specify parameters to compile *approve* in a specified manner at the time of execution.

Note: You can also specify compilation values with the includes for the financial products that affect the *approve* program.

Parameter Syntax

You can display *approve* parameters by entering the following: **approve -,**

The following is the correct usage for running the *approve* program from the UNIX shell:

Usage: approve [-d rundate] -o device [-m mode]

Parameters that appear in brackets are optional. Parameters that do not appear in brackets are required.

Parameters

The following lists the parameters for running *approve*.

-d rundate
Optional - Specifies the purchase order run date (mm/dd/yyyy) that you want to use.

-o device
Required - Specifies the name of the purchase order printing device.

-m mode
Optional - Indicates if you want to print purchase orders immediately (Y, N).

Program Screens

Introduction

The *approve* program uses four screens that users can access to view or enter approval information.

Access

The screen files are located in the following directory path:
$CARSPATH/src/purchasing/approve/SCR

Screen Files and Table/Record Usage

The *approve* screens appear in the following files and use the indicated tables and records:

glacct
Contains the General Ledger Account Distribution screen

Tables/Records: gla_rec, gltr_rec

invappr
Contains the Invoices Needing Approval screen

Tables/Records: id_rec, po_rec, subb_rec

poappr
Contains the Purchase Orders Needing Approval screen

Tables/Records: id_rec, po_rec

SECTION 27 - PURCHASING: PURCHASE

Overview

Introduction

This section provides you with reference information about the Purchase (*purch*) program. The Accounts Payable users uses *purch* to aid them in the entry and tracking of a purchase order in the accounting system.

Note: This program interacts with the CX programs *approval*, *purch*, *invoice*, *purchaudit*, and *vndentry*. For information about other Jenzabar procurement products (*acctspay*, *assign*, *massinv*, *receive*, *requisition*, and *approval*), see the following documentation:

- *RPA Technical Manual*
- *Using Purchasing and Accounts Payable*
- *Using Requisitioning*

Program Features Detailed

This section contains details about the following features of the *purch* program:

- Process flow
- Parameters
- Program screens

Records and Tables Used

The *purch* program uses the following tables and records:

doc_table
The Document table that contains information about document codes and stations

fscl_cal_rec
The Fiscal Calendar record that defines fiscal calendar years and periods

func_table
The Function table that defines the institution's valid function codes

gl_amt_rec
The General Ledger Amount record that provides summarized totals for general ledger accounts over fiscal periods

gla_rec
The General Ledger Account record that contains the fund, function, object and subfund combinations that your institution has used

gld_rec
The General Ledger Definition record that defines rules used in validating new accounts; also contains codes that allow single-keystroke account entry (shortcut codes)

gle_rec
The General Ledger Journal Entry record that contains information about each entry

gltr_rec
The General Ledger Transaction record that contains the amount and account charged for each transaction in an entry

id_rec
The Identification record that contains information about each individual or entity in the CX database

obj_table
The Object table that defines the valid object codes for your institution

po_rec
The Purchase Order record that contains information relating to purchase orders, including amounts, dates and vendors

pobody_rec
The Purchase Order Detail record that contains information for the main body, or detail, portion of the purchase order

suba_rec
The Subsidiary Account record that contains information about the subsidiary number

subb_rec
The Subsidiary Balance record that contains information about a subsidiary balance transaction

sube_rec
The Subsidiary Entry record that contains information about each entry that impacts a subsidiary

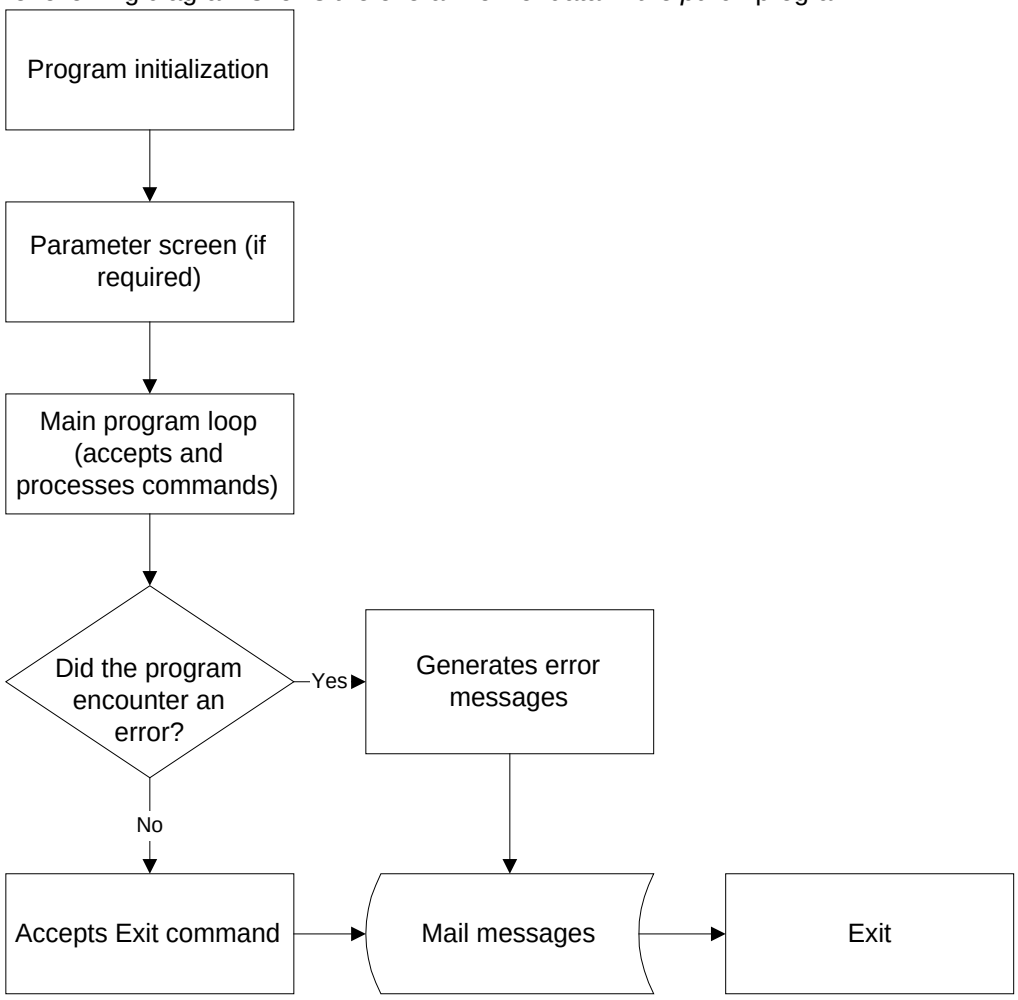
subs_table
The Subsidiary table that contains information about subsidiaries

subtr_rec
The Subsidiary Transaction record that contains detailed transactions for subsidiary account posting

Process Flows

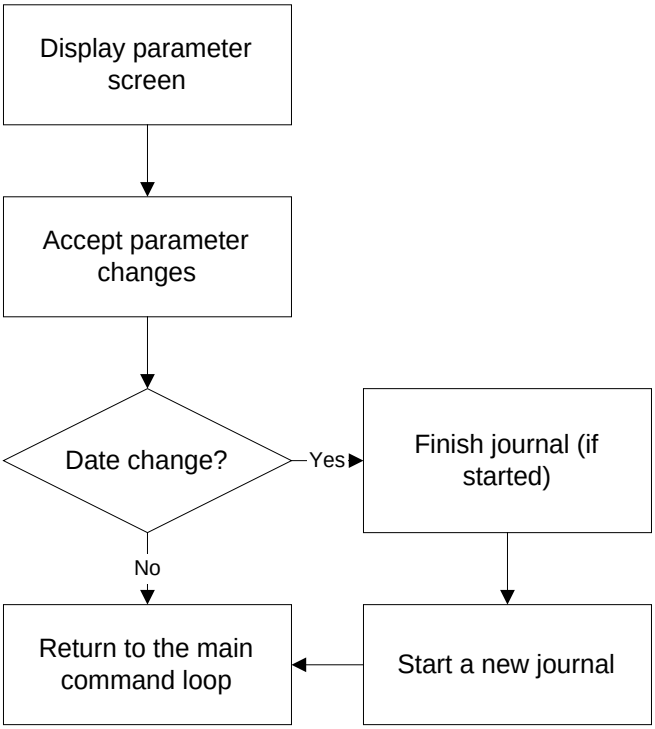
Overall Process Flow

The following diagram shows the overall flow of data in the *purch* program.



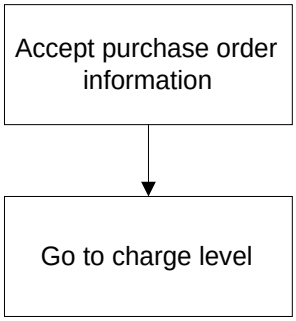
Process Flow for the Initialize Command

The following diagram shows the flow of data when the *purch* program encounters the Initialize command.



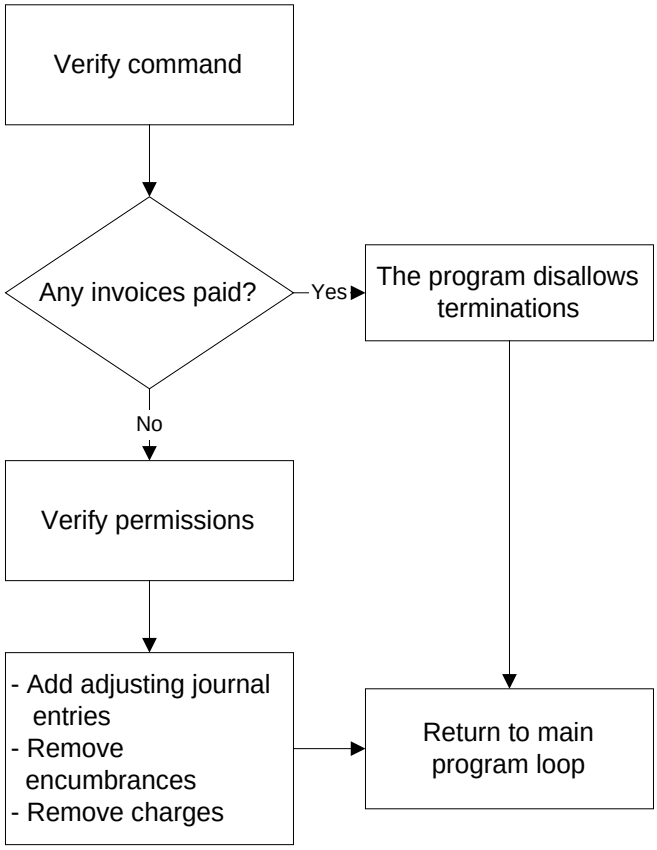
Process Flow for the Add Command

The following diagram shows the flow of data when the *purch* program encounters the Add command.



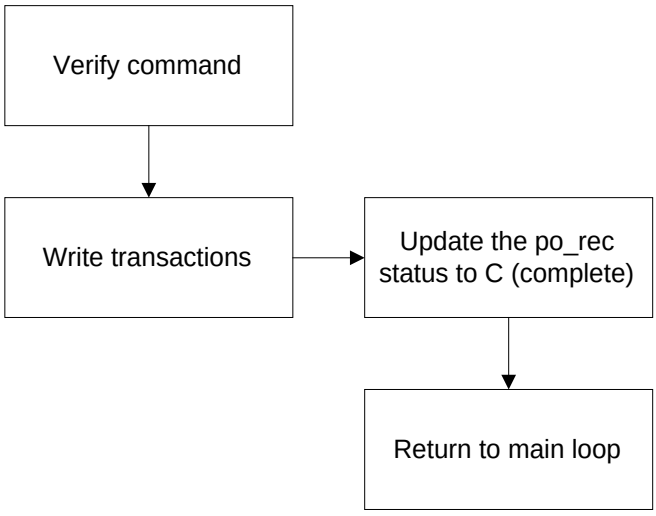
Process Flow for the Terminate Command

The following diagram shows the flow of data when the *purch* program encounters the Terminate command.



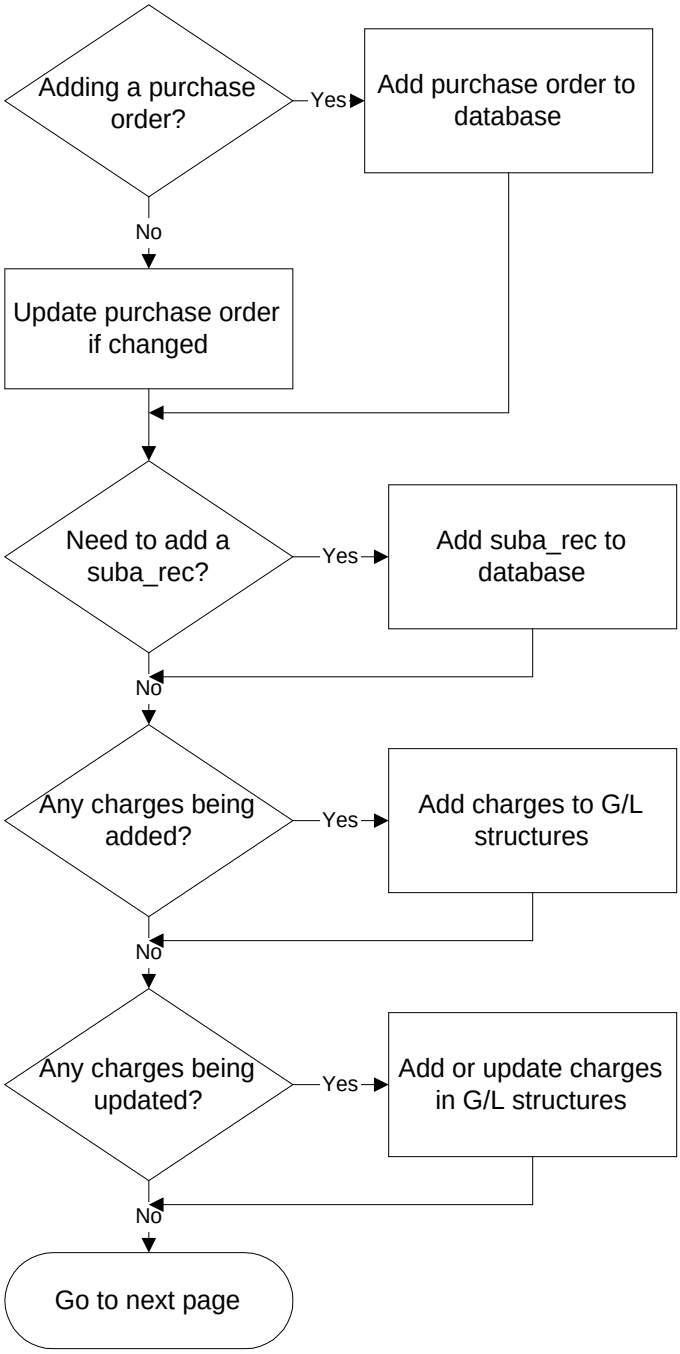
Process Flow for the Complete Command

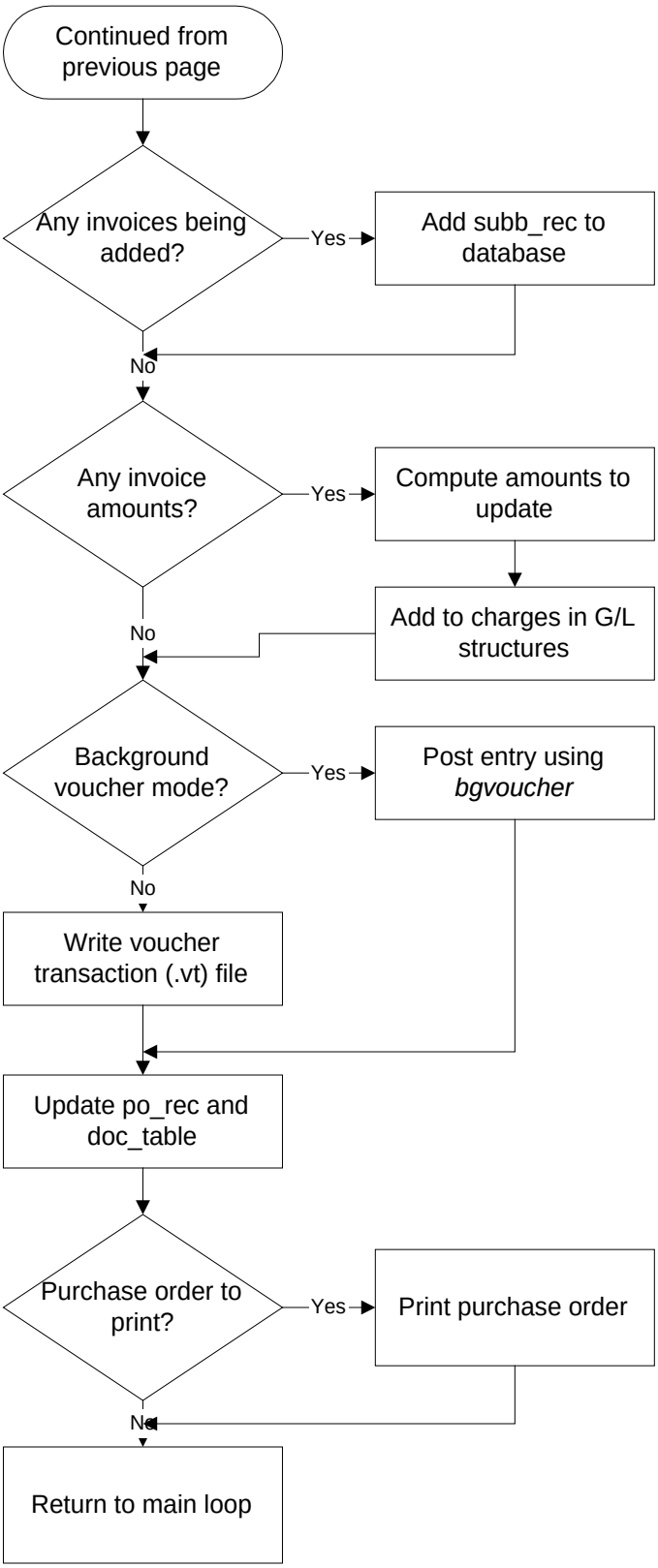
The following diagram shows the flow of data when the *purch* program encounters the Complete command.



Process Flow for the Write Command

The following diagram shows the flow of data when the *purch* program encounters the Write command.




Continued from previous page
Any invoices being added?
Yes
Add subb_rec to database
No
Any invoice amounts?
Yes
Compute amounts to update
Add to charges in G/L structures
No
Background voucher mode?
Yes
Post entry using *bgvoucher*
No
Write voucher transaction (.vt) file
Update po_rec and doc_table
Purchase order to print?
Yes
Print purchase order
No
Return to main loop

Data Flow Description

The following describes the data flow in the *purch* program.

1. The user launches the program and enters parameters as needed.
2. The *purch* program's main program loop activates and accepts commands.
3. The *purch* program's main program loop generates error messages and electronic mail messages as required, accepts the Exit command, and exits.

Command Flow Descriptions

The following lists describe the processing related to each *purch* command:

Initialize command

1. The *purch* program determines if the date has changed.
2. If the date has changed, *purch* finishes journals (if one has already been started), and begins a new journal.
3. The *purch* program returns to the main program loop.

Add command

1. The *purch* program accepts purchase order information, including vendor data and information about the goods and services ordered.
2. The *purch* program accesses the charge level where the user enters account information.
3. The *purch* program returns to the main program loop.

Terminate command

1. The *purch* program determines if any invoices for the purchase order have been paid. If yes, it disallows the command and returns to the main program loop.
2. If no invoices have been paid, *purch* determines if the user has permission to terminate a purchase order. If no, it disallows the command and returns to the main program loop.
3. If the user has permission, *purch* creates adjusting entries, removes encumbrances and removes charges, then returns to the main program loop.

Complete command

1. The *purch* program writes the required transactions for posting by *bgvoucher*.
2. The *purch* program updates the purchase order records then returns to the main program loop.

Write command

1. The *purch* program adds or updates the purchase order.
2. The *purch* program adds a suba_rec if required.
3. The *purch* program adds or updates charges against general ledger structures if required.
4. The *purch* program adds subb_recs if required.
5. The *purch* program process the invoice amounts, if any, by computing amounts and adding charges to the general ledger.
6. The *purch* program calls *bgvoucher* to perform posting.
7. The *bgvoucher* program posts transactions, creating a voucher transaction file.
8. The *purch* program updates the po_rec and the doc_table to reflect the new purchase order.
9. The *fps* program prints the purchase order if required.
10. The *purch* program returns to the main program loop.

Purchase Parameters

Introduction

The CX contains parameters and compilation values for executing the *purch* program. You can specify parameters to compile *purch* in a specified manner at the time of execution.

Note: You can also specify compilation values with the includes for the financial products that affect the *purch* program.

Parameter Syntax

You can display *purch* parameters by entering the following: **purch -,**

The following is the correct usage for running the *purch* program from the UNIX shell:

Usage: purch [-i command] [-d rundate] [-s subsidiary] [-c doc_code] -o device [-p permission] [-f form] [-b] [-m mode]

Parameters that appear in brackets are optional. Parameters that do not appear in brackets are required.

Parameters

The following lists the parameters for running *purch*.

-i command
Optional - Specifies the purchase order level code. Valid values are:
- Q (query)
- S (station number)

-d rundate
Optional - Specifies the run date for the purchase orders.

-s subsidiary
Optional - Specifies the subsidiary to which the purchase order applies.

-c doc_code
Optional - Specifies the document code from the doc_table for the purchase order.

-o device
Required - Specifies the purchase order printer.

-f form
Optional - Specifies the purchase order form type.

-b
Optional - Indicates that you want to use the immediate mode for *fps* printing.

-m mode
Optional - Specifies the *bgvoucher* mode that you want to use. Valid values are:
- V (Verify only)
- P (Verify and post)
- N (New *bgvoucher* transaction file)
- O (Old *voucher* transaction file)

Program Screens

Introduction

The *purch* program uses twelve screens that provide help or enable users to enter and view parameters or data.

Access

The screen files are located in the following directory path: $CARSPATH/src/purchasing/purch/SCR.

Screen Files and Table/Record Usage

The *purch* screens appear in the following files:

- chg
- chghelp
- gl
- inv
- invhelp
- param
- po
- pobody
- podspl
- pohelp
- qry
- suba

SECTION 28 - PURCHASING: PURCHASING AUDIT

Overview

Introduction

This section provides you with reference information about the Purchasing Audit (*purchaudit*) program. The Accounts Payable product uses *purchaudit* to update the amount fields on the po_recs to reflect the supporting data that is in the general ledger entries.

CAUTION: Never run *purchaudit* if any unposted voucher transaction files exist. The supporting detail for purchase orders does not exist until all posting is complete.

Note: This program interacts with the CX programs *approval*, *purch*, *invoice*, and *vndentry*. For information about other Jenzabar procurement products (*acctspay*, *assign*, *massinv*, *receive*, *requisition*, and *approval*), see the following documentation:

- *RPA Technical Manual*
- *Using Purchasing and Accounts Payable*
- *Using Requisitioning*

Program Features Detailed

This section contains details about the following features of the *purchaudit* program:

- Process flow
- Parameters

Program Screens

Because *purchaudit* is a background process, it does not use any program screens.

Records and Tables Used

The *purchaudit* program uses the following records and tables:

gle_rec
The General Ledger Journal Entry record that contains information about each entry

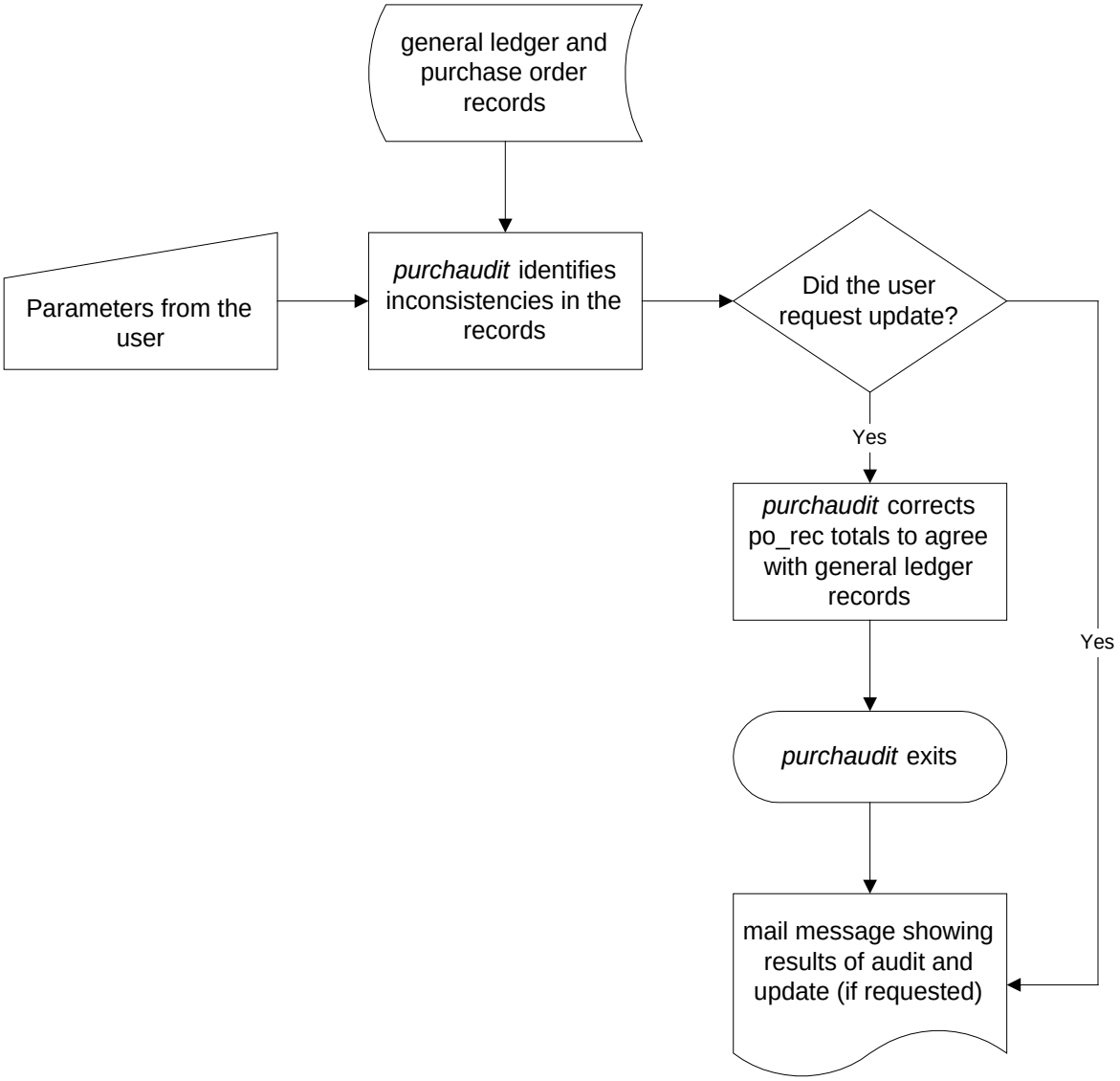
gltr_rec
The General Ledger Transaction record that contains the amount and account charged for each transaction in an entry

po_rec
The Purchase Order record that contains information relating to purchase orders, including amounts, dates and vendors

Process Flow

Diagram

The following diagram shows the flow of data in the *purchaudit* program.



Data Flow Description

The following describes the data flow in the *purchaudit* program.

1. The user launches *purchaudit*, specifying parameters as desired.
2. Based on the parameters, *purchaudit* accesses the necessary general ledger and purchase order records.
3. If specified, *purchaudit* updates the purchase order records to agree with the supporting general ledger information.
4. The program sends electronic mail with the results of the audit.

Purchasing Audit Parameters

Introduction

The CX contains parameters and compilation values for executing the *purchaudit* program. You can specify parameters to compile *purchaudit* in a specified manner at the time of execution.

Note: You can also specify compilation values with the includes for the financial products that affect the *purchaudit* program.

Parameter Syntax

The following is the correct usage for running the *purchaudit* program from the UNIX shell:

Usage: purchaudit [-r] [-u] [-c code] [-b beginning number] [-e ending number] [-n number]

Parameters that appear in brackets are optional. Parameters that do not appear in brackets are required.

Parameters

The following lists the parameters for running *purchaudit*.

-r
Optional - Indicates the user wants to receive a report of all audited purchase orders. If the user does not use this parameter, *purchaudit* produces a report only on the purchase orders that require updating.

-u
Optional - Causes the program to perform the updates if data in the po_recs does not match the supporting detail in the general ledger entries.

-c code
Optional - Designates the document code of the purchase orders to be audited (e.g., purchaudit -c PO).

-b beginning number
Optional - Designates the first purchase order number for which the user wants to perform the audit.

-e ending number
Optional - Designates the last purchase order number for which the user wants to perform the audit.

-n number
Optional - Designates the specific purchase order number(s) for which the user wants to perform the audit (e.g., purchaudit -n 233 234 to audit the purchase orders numbered 233 and 234).

Program Output

Introduction

The *purchaudit* program sends electronic mail to the user when it completes processing. The mail provides complete documentation about the *purchaudit* process.

Example Mail Message

The following example demonstrates the type of information available from the mail messages generated by *purchaudit*.

```
Parameters:
Document Code:'PO'
Numbers 3 4 56 54
Update Audited Purchase Orders
Report on All Purchase Orders

Program Successfully Completed

Could not find po record requested. 'PO'-'56'. Status:6010
Could not find po record requested. 'PO'-'54'. Status:6010

                          Encumbrances                    Actual
Code Number            Record      Computed          Record      Computed  Upd

PO-00000003           2200.70       2200.70         1220.50       2220.50    *
PO-00000004            625.00        625.00            0.00          0.00
```

How to Interpret the Example Mail Message

The first group of lines in the mail file state the parameters that the user passed to *purchaudit*. The next line states that *purchaudit* successfully completed. This message does not mean that all the records audited were without error; instead, it indicates the program did not abnormally end.

The next group of messages ("Could not find po ...") report on purchase orders that the user selected for auditing that did not have a purchase order record (po_rec).

The final group of messages reports on the audited purchase orders. The first two columns show the purchase order that was audited. The next two columns under the heading '"Encumbrances"' report the total encumbrance that has been charged against the purchase order. The first amount is the amount that existed in the po_rec. The next amount is the amount that *purchaudit* computed based on the general ledger entries and transactions for the purchase order.

The next two columns under the heading “Actual” display the total invoices that have been issued against the purchase order. The first column, as with encumbrance, reports on the amount that existed in the po_rec. The next column is the amount that was computed for the purchase order based on the general ledger entries and transactions.

If an asterisk (*) appears in the final column, then purchaudit updated the purchase order in the database.

Note: If the user specified on the command line that the program was to audit all purchase orders, the reporting on specific purchase orders would not have been mailed to the user but would be stored on disk. The user would receive a message similar to the following, giving the location of the file:

```
Review file: '/usr/carsdev/audit/purchasing/purchaudit/960205.1517' for results
```

SECTION 29 - PURCHASING: VENDOR ENTRY

Overview

Introduction

This section provides you with reference information about the Vendor Entry (*vndentry*) program. The users of the Purchasing product use *vndentry* to enter the names and detail information about the vendors that supply goods and services to the institution. Most of the information that users enter through *vndentry* resides in the vnd_rec, although comments that users enter about each vendor reside in vnd_blob. Input to *vndentry* impacts the A/P subsidiary.

Users can access *vndentry* from within Purchasing and Accounts Payable or directly from the Purchasing/AP Main menu.

Note: This program interacts with all the CX purchasing and accounts payable programs. For information about other CX procurement products (*acctspay*, *assign*, *massinv*, *receive*, *requisition*, and *approval*), see the following documentation:

- *RPA Technical Manual*
- *Using Purchasing and Accounts Payable*
- *Using Requisitioning*

Program Features Detailed

This section contains details about the following features of the *vndentry* program:

- Process flow
- Parameters
- Program screens
- Detail windows

Records and Tables Used

The *vndentry* program uses the following records and tables:

aa_rec
The Alternate Address record that contains alternate address information for an individual

addree_rec
The Addressee record that defines the preferred salutation and/or name line for an individual

adr_rec
The Addressing record that describes the type of addressing to perform for the run code, alternate address, or individual specified

ctc_rec
The Contact record that records the sending or receipt of correspondence with an individual, institution, business or organization

emp_rec
The Employment record that identifies the company and position held by an individual

hold_rec
The Hold record that contains explanations for any interruption in regular processing of an individual's or organization's information; also ensures that the interruption in regular processing occurs

id_rec
The Identification record that contains information about each individual or entity in the CX database

po_rec
The Purchase Order record that contains information relating to purchase orders, including amounts, dates and vendors

profile_rec
The Profile record that contains demographic information about an individual

relation_rec
The Relationship record that identifies two individuals or entities and the relationship between them

suba_rec
The Subsidiary Account record that contains information about the subsidiary number

vnd_rec
The Vendor record that contains information about vendors, including the contact person and delivery terms

Process Flow

Data Flow Description

The following describes the data flow in the *vndentry* program. Users must run this process to create vendor information for every vendor, supplier, or other service provider for whom the institution wants to process requisitions and purchase orders.

1. The user launches vndentry, and selects the form to use for data entry.
2. Does the vendor exist in the CX database?
   - If yes, the user views or updates fields as needed.
   - If no, the user enters id_rec and vnd_rec information onto the form selected in step 1 above.
3. The user selects **Finish**, and vndentry creates id_recs and vnd_recs.
4. Does the user want to process other vendor information?
   - If yes, go to step 2.
   - If no, the user selects **Exit**.

Library Relationships

The *vndentry* program uses the *libentry* library.

Vendor Entry Parameters

Introduction

The CX contains parameters and compilation values for executing the *vndentry* program. You can specify parameters to compile *vndentry* in a specified manner at the time of execution.

Note: You can also specify compilation values with the includes for the financial products that affect the *vndentry* program.

Parameter Syntax

You can display *vndentry* parameters by entering the following: **vndentry -,**

The following is the correct usage for running the *vndentry* program from the UNIX shell:

Usage: vndentry -[-d] [-m menuname] [-F] [-o ofc_added_by] [-f form_selected] [-t today] [-P scr_path] [-a] [-M menu_title] [-q] [-D debug_level] [-S pause_level] [-p]

Parameters that appear in brackets are optional. Parameters that do not appear in brackets are required.

Parameters

The following lists the parameters for running *vndentry*.

-d
Optional - Specifies the ability to access *vndentry* in display mode only.

-m menuname
Optional - Specifies the name of the menu screen. The default is *vndmenu*.

-F
Optional - Requires the user to perform a query before entering Insert mode.

-o ofc_added_by
Optional - Specifies the name of the office for which the user is accessing the program. The default is NOFC.

-f form_selected
Optional - Specifies the name of the desired form data entry screen instead of using a menu.

-t today
Optional - Specifies the effective date for changes

-P scr_path

Optional - Specifies the path in which the screens are located. Pass this parameter if the screen location is not the default, $CARSPATH/modules/purchasing/progscr/vndentry.

-a

Optional - Causes the program to automatically enter Query mode.

-M menu_title

Optional - Changes the ring menu title.

-q

Optional - Causes additional restrictions on a name query.

-D debug_level

Optional - Specifies a higher level of information to be passed when maintaining the program. Valid values include 1, 3, 5, 7, and 9. The higher the number, the more messages the program passes. This parameter is used for debugging.

-S pause_level

Optional - Causes the program to pause more frequently. Valid levels are 1 - 9. The lower the number, the more pauses the program creates. This parameter is used for debugging.

-p

Optional - Specifies that a program-to-program (PTP) connection invokes the program. The requisition and purchase programs allow for direct connection (PTP) to *vndentry*.

Program Screens

Introduction

The *vndentry* program uses three screens and one screen menu for capturing and displaying vendor information.

Access

The screen files are located in the following directory path:
$CARSPATH/modules/purchasing/progscr/vndentry.

Screen Files and Table/Record Usage

The *vndentry* screens appear in the following files and use the indicated tables and records:

vnd_1
Contains the first page of the Vendor Entry for Subsidiary information. This screen is used more for corporate vendors.

Tables/Records: aa_table, ctry_table, id_rec, ofc_table, payterm_table, st_table, suba_rec, title_table, venqual_table, ventype_table, vnd_rec

vnd_2
Contains the second page of the Vendor Entry for Subsidiary information.

Tables/Records: id_rec, ofc_table, suba_rec, subs_table

vndmenu
Contains the Vendor Entry Menu screen

Tables/Records: None

vndoth_1
Contains the vendor information used for individuals, not corporations.

Tables/Records: aa_table, ctry_table, id_rec, ofc_table, payterm_table, st_table, suba_rec, subs_table, title_table, vnd_rec

SECTION 30 - MENUS, SCREENS, SCRIPTS AND REPORTS

Overview

Introduction

This section provides you reference information on the following features of the financial product:

- Menu source files
- Menu option files
- PERFORM screens
- SQL scripts
- Csh scripts
- ACE reports

Directory Locations

The features detailed in this section are located in the following directory paths:

Menu source files:

- $CARSPATH/menusrc/fiscal/acctspay
- $CARSPATH/menusrc/fiscal/acctsrecv
- $CARSPATH/menusrc/fiscal/approve
- $CARSPATH/menusrc/fiscal/appurch
- $CARSPATH/menusrc/fiscal/cashier
- $CARSPATH/menusrc/fiscal/finbgt
- $CARSPATH/menusrc/fiscal/fixassets
- $CARSPATH/menusrc/fiscal/purchasing

Menu option files:

- $CARSPATH/menuopt/accounting/informers
- $CARSPATH/menuopt/accounting/programs
- $CARSPATH/menuopt/accounting/screens
- $CARSPATH/menuopt/accounting/scripts
- $CARSPATH/menuopt/acctspay/others
- $CARSPATH/menuopt/acctspay/programs
- $CARSPATH/menuopt/acctspay/reports
- $CARSPATH/menuopt/acctspay/screens
- $CARSPATH/menuopt/acctspay/scripts
- $CARSPATH/menuopt/budget/others
- $CARSPATH/menuopt/budget/programs
- $CARSPATH/menuopt/budget/reports
- $CARSPATH/menuopt/budget/screens
- $CARSPATH/menuopt/budget/scripts
- $CARSPATH/menuopt/common/screens
- $CARSPATH/menuopt/fixassets/others
- $CARSPATH/menuopt/fixassets/programs
- $CARSPATH/menuopt/fixassets/reports
- $CARSPATH/menuopt/fixassets/screens
- $CARSPATH/menuopt/purchasing/others
- $CARSPATH/menuopt/purchasing/programs
- $CARSPATH/menuopt/purchasing/reports
- $CARSPATH/menuopt/purchasing/screens
- $CARSPATH/menuopt/utilities/programs

PERFORM screens:

- $CARSPATH/modules/accounting/screens
- $CARSPATH/modules/acctspay/screens
- $CARSPATH/modules/budget/screens
- $CARSPATH/modules/fixassets/screens
- $CARSPATH/modules/purchase/screens

Csh scripts:

- $CARSPATH/modules/acctspay/scripts
- $CARSPATH/modules/budget/scripts

ACE reports:

- $CARSPATH/modules/acctspay/reports
- $CARSPATH/modules/appurch/reports
- $CARSPATH/modules/budget/reports
- $CARSPATH/modules/fixassets/reports
- $CARSPATH/modules/purchase/reports
- $CARSPATH/modules/requisition/reports

Financial Menus

Introduction

The CX menu source (menusrc) directory path contains definitions of the CX menu structure. Specifically, the $CARSPATH/menusrc/fiscal directory path contains definitions for a variety of financial menus. The following directories corresponding to financial products appear in this path:

- $CARSPATH/menusrc/fiscal/acctspay
- $CARSPATH/menusrc/fiscal/acctspay/reports
- $CARSPATH/menusrc/fiscal/acctspay/rfckwrtg
- $CARSPATH/menusrc/fiscal/approve
- $CARSPATH/menusrc/fiscal/appurch
- $CARSPATH/menusrc/fiscal/appurch/ckwrtg
- $CARSPATH/menusrc/fiscal/appurch/ckwrtg/problems
- $CARSPATH/menusrc/fiscal/appurch/f1099
- $CARSPATH/menusrc/fiscal/appurch/purchaudit
- $CARSPATH/menusrc/fiscal/appurch/r1099
- $CARSPATH/menusrc/fiscal/appurch/reports1
- $CARSPATH/menusrc/fiscal/appurch/reports2
- $CARSPATH/menusrc/fiscal/cashier
- $CARSPATH/menusrc/fiscal/finbgt
- $CARSPATH/menusrc/fiscal/finbgt/bgtrpt
- $CARSPATH/menusrc/fiscal/finbgt/bgtrpt/account
- $CARSPATH/menusrc/fiscal/finbgt/bgtrpt/center
- $CARSPATH/menusrc/fiscal/finbgt/bgtrpt/project
- $CARSPATH/menusrc/fiscal/finbgt/tables
- $CARSPATH/menusrc/fiscal/finbgt/tables/othtables
- $CARSPATH/menusrc/fiscal/fixassets
- $CARSPATH/menusrc/fiscal/fixassets/reports
- $CARSPATH/menusrc/fiscal/purchasing
- $CARSPATH/menusrc/fiscal/purchasing/reports

Each directory above contains a *menudesc* file, which specifies what menu options appear in a menu. Specific menu options, however, are defined in the menu option (menuopt) directory path.

Menu Options

The following list associates each program, screen and script menu option and corresponding menuopt file and identifies the menuopt locations and what the menu option accesses.

Note:

- The following menus and options are listed in the order in which they appear on the standard CX menu structure. The order of display is determined by the menudesc file in the $CARSPATH/menusrc/fiscal directory, which defines the Main Fiscal menu.
- Some submenus on the Main Fiscal menu are documented in other Jenzabar, Inc. documents, as follows:
  - Accounting (the *finacctg* submenu) documentation - *General Ledger Technical Manual*.
  - Auditing (the *finaudit* submenu) documentation - *General Ledger Technical Manual*.
  - PO/Requisition/Approval (the *requisition* submenu) documentation – *RPA Technical Manual.*
  - Accounts Payable/Receiving (the *aprecv* submenu) documentation - *RPA Technical Manual.*
- Italicized parameters indicate those that a user can enter or change.

Fiscal Management: Accounts Payable Main menu

Enter Invoices

Accesses: Program: *purch*

Menuopt File: $CARSPATH/menuopt/purchasing/programs/purc.aBG

Parameters Passed:

- *Subsidiary (if ENABLE_MULTI_AP_SUBS = Y)*
- *Document code (if ENABLE_MULTI_DOC_PO = Y)*
- *Date*
- *Output device*
- *Purchase order form type (if ENABLE_MULTI_FORM_PO = Y)*

Fiscal Management: Accounts Payable Main menu

Enter Manual Invoices

Accesses: Program: *voucher*

Menuopt File: $CARSPATH/menuopt/accounting/programs/vch.AP

Parameters Passed:

- Journal type (AP)

Fiscal Management: Accounts Payable Main menu

Adjust Manual Invoices

Accesses: Program: *voucher*

Menuopt File: $CARSPATH/menuopt/accounting/programs/vch.ACinv

Parameters Passed:

- Journal type (AC)

Fiscal Management: Accounts Payable Main menu

G/L Journal Reports

Accesses: Csh script: jrnlgl

Menuopt File: $CARSPATH/menuopt/accounting/scripts/jrnlgl.AP

Parameters Passed:

- *Journal type*
- *Beginning journal number*
- *Ending journal number*
- *Output device*

Fiscal Management: Accounts Payable Main menu

S/L Journal Reports

Accesses: Csh script: jrnlsl

Menuopt File: $CARSPATH/menuopt/accounting/scripts/jrnlsl.AP

Parameters Passed:

- *Journal type*
- *Beginning journal number*
- *Ending journal number*
- *Output device*

Fiscal Management: Accounts Payable Main menu

Vendor Entry

Accesses: Program: *vndentry*

Menuopt File: $CARSPATH/menuopt/purchasing/programs/vnde

Parameters Passed:

- Automode (if ENABLE_FEAT_AUTOMODE = Y)
- Forced Query (if ENABLE_FEAT_FORCEQUERY = Y)
- Debugging (level 3)

Fiscal Management: AP Check Writing menu

Note: See *Purchasing/AP: Check Writing menu* later in this section.

Fiscal Management: Accounts Payable: Refund: Check Writing menu

Select Check Group

Accesses: PERFORM screen: ckgrpsr

Menuopt File: $CARSPATH/menuopt/acctspay/screens/ckgrp.sr

Parameters Passed:

- None

Fiscal Management: Accounts Payable: Refund: Check Writing menu

Select Checks

Accesses: Program: *ckslct*

Menuopt File: $CARSPATH/menuopt/acctspay/programs/cksl.sr

Parameters Passed:

- *-g (Check group)*
- *-s (Subsidiary)*

Fiscal Management: Accounts Payable: Refund: Check Writing menu

Print Checks / Print Checks and Grouping Sheets

Accesses: Utility program: *fps*

Menuopt File: $CARSPATH/menuopt/utilities/programs/fps.ck

Parameters Passed:

- None

Fiscal Management: Accounts Payable: Refund: Check Writing menu

Post Checks

Accesses: Program: *ckpost*

Menuopt File: $CARSPATH/menuopt/acctspay/programs/ckps.rf

Parameters Passed:

- *-g (Check group)*

Fiscal Management: Accounts Payable: Refund: Check Writing menu

Check Register

Accesses: ACE report: jrnldocreg

Menuopt File: $CARSPATH/menuopt/accounting/reports/jrnldoc.ap

Parameters Passed:

- *Journal type*
- *Beginning journal number*
- *Ending journal number*

Fiscal Management: Accounts Payable: Refund: Check Writing menu

G/L Journal Reports

Accesses: Csh script: jrnlgl

Menuopt File: $CARSPATH/menuopt/accounting/scripts/jrnlgl.AP

Parameters Passed:

- *Journal type*
- *Beginning journal number*
- *Ending journal number*
- *Output device*

Fiscal Management: Accounts Payable: Refund: Check Writing menu

S/L Journal Reports

Accesses: Csh script: jrnlsl

Menuopt File: $CARSPATH/menuopt/accounting/scripts/jrnlsl.AP

Parameters Passed:

- *Journal type*
- *Beginning journal number*
- *Ending journal number*
- *Output device*

Fiscal Management: Accounts Payable: Refund: Check Writing menu

Void Checks

Accesses: Program: *docvoid*

Menuopt File: $CARSPATH/menuopt/accounting/programs/docv.CK

Fiscal Management: Accounts Payable: AP Check Writing menu

Note: See *Purchasing/AP: Check Writing: Problem Solving menu* later in this section.

Fiscal Management: Accounts Payable: Reports menu

Check Register

Accesses: ACE report: jrnldocreg

Menuopt File: $CARSPATH/menuopt/accounting/reports/jrnldoc.ap

Parameters Passed:

- *Journal type*
- *Beginning journal number*
- *Ending journal number*

Fiscal Management: Accounts Payable: Reports menu

Invoices for Purchase Order

Accesses: ACE report: poinv

Menuopt File: $CARSPATH/menuopt/acctspay/reports/poinv

Parameters Passed:

- *Subsidiary (if ENABLE_MULTI_AP_SUBS = Y)*
- *Purchase order number*

Fiscal Management: Accounts Payable: Reports menu

Credit Memos

Accesses: ACE report: paydebit

Menuopt File: $CARSPATH/menuopt/acctspay/reports/paydebit

Menuopt File:

Parameters Passed:

- *Subsidiary (if ENABLE_MULTI_AP_SUBS = Y)*
- *Subsidiary balance period (if ENABLE_MULTI_AP_BALS = Y)*

Fiscal Management: Accounts Payable: Reports menu

Discounts Missed

Accesses: ACE report: discmissed

Menuopt File: $CARSPATH/menuopt/acctspay/reports/discmissed

Parameters Passed:

- *Subsidiary (if ENABLE_MULTI_AP_SUBS = Y)*
- *Beginning date*
- *Ending date*

Fiscal Management: Accounts Payable: Reports menu

Discounts Taken

Accesses: ACE report: disctaken

Menuopt File: $CARSPATH/menuopt/acctspay/reports/disctaken

Parameters Passed:

- *Subsidiary (if ENABLE_MULTI_AP_SUBS = Y)*
- *Beginning date*
- *Ending date*

Fiscal Management: Accounts Payable: Reports menu

Document Register

Accesses: ACE report: docreg

Menuopt File: $CARSPATH/menuopt/accounting/reports/docreg.ap

Parameters Passed:

- *Document type*
- *Beginning document number*
- *Ending document number*

Fiscal Management: Accounts Payable: Reports menu

Purchase Order Aging

Accesses: ACE report: poaging

Menuopt File: $CARSPATH/menuopt/acctspay/reports/poaging

Parameters Passed:

- *Subsidiary (if ENABLE_MULTI_AP_SUBS = Y)*
- *Date*
- *Ending day for first column (and beginning day for second column)*
- *Ending day for second column (and beginning day for third column)*
- *Ending day for third column*

Fiscal Management: Accounts Payable: Reports menu

Invoice Aging

Accesses: ACE report: invaging

Menuopt File: $CARSPATH/menuopt/acctspay/reports/invaging

Parameters Passed:

- *Subsidiary (if ENABLE_MULTI_AP_SUBS = Y)*
- *Date*
- *Ending day for first column (and beginning day for second column)*
- *Ending day for second column (and beginning day for third column)*
- *Ending day for third column*

Fiscal Management: Accounts Payable: Reports menu

Invoice Forecasting

Accesses: ACE report: payfore

Menuopt File: $CARSPATH/menuopt/acctspay/reports/payfore

Parameters Passed:

- *Subsidiary (if ENABLE_MULTI_AP_SUBS = Y)*
- *Date*
- *Ending day for first column (and beginning day for second column)*
- *Ending day for second column (and beginning day for third column)*
- *Ending day for third column*

Fiscal Management: Accounts Payable: Reports menu

S/L Account Balances

Accesses: ACE report: subbalance

Menuopt File: $CARSPATH/menuopt/accounting/reports/subbal.AP

Parameters Passed:

- *Subsidiary (if ENABLE_MULTI_AP_SUBS = Y)*
- *Date*
- *Debit balances*
- *Credit balances*
- *Zero balances*
- *Include Actual Amounts flag*
- *Include Encumbered Amounts flag*
- *Include Additional Information flag*
- *Subprogram (if enabled)*

Fiscal Management: Accounts Payable: Reports menu

S/L Cash Entries

Accesses: ACE report: subentcash

Menuopt File: $CARSPATH/menuopt/accounting/reports/subec.AP

Parameters Passed:

- *Subsidiary (if ENABLE_MULTI_AP_SUBS = Y)*
- *ID*
- *Beginning date*
- *Ending date*
- *Entry type*
- *Amount type*
- *Comparison amount*
- *Summary*

Fiscal Management: Accounts Payable: Reports menu

Subsidiary History

Accesses: ACE report: subtrhist

Menuopt File: $CARSPATH/menuopt/accounting/reports/subtrh.AP

Parameters Passed:

- *Subsidiary (if ENABLE_MULTI_AP_SUBS = Y)*
- *ID*
- *Beginning date*
- *Ending date*

Fiscal Management: Accounts Payable: Reports menu

Vendor Listing

Accesses: ACE report: vndlst

Menuopt File: $CARSPATH/menuopt/acctspay/reports/vndlst

Parameters Passed:

- None

Fiscal Management: Budgeting Main menu

Create Base Year Data

Accesses: Program: *bgtbasis*

Menuopt File: $CARSPATH/menuopt/budget/programs/bgtb

Parameters Passed:

- *Fiscal year*
- *Fund*
- *Limit accounts*
- *Amount type*

Fiscal Management: Budgeting Main menu

Global Adjustments

Accesses: Csh script: *addbgtamt.scp*

Menuopt File: $CARSPATH/menuopt/budget/scripts/addbgtamt

Parameters Passed:

- *First Fiscal Yr (e.g., 9798)*
- *Amount Type (e.g., BGT)*
- *Second Fiscal Yr (e.g., 9899)*
- *Amount Type (e.g., BGT)*
- *Fund Code (e.g., 10)*
- *Account Code Range (e.g., 4000 - 9999)*
- *Adjustments (e.g., Y)*

Fiscal Management: Budgeting Main menu

Allocate Budget

Accesses: Program: *bgtalloc*

Menuopt File: $CARSPATH/menuopt/budget/programs/bgta.c

Parameters Passed:

- *Budget parameter code*

Fiscal Management: Budgeting Main menu

Post Budget to G/L

Accesses: Program: *bgtinstall*

Menuopt File: $CARSPATH/menuopt/budget/programs/bgti

Parameters Passed:

- *Fiscal Year (e.g., 9798)*
- *Amount Type (e.g., BGT)*
- *Fund Code (e.g., 10)*

Fiscal Management: Budgeting Main menu

G/L Journal Reports

Accesses: SQL statement: *jrnlgl.scp*

Menuopt File: $CARSPATH/menuopt/accounting/scripts/jrnlgl.BG

Parameters Passed:

- -f (formtype; default is *Standard*)
- *COMMENT_TVCH_NO (print comments; default is No)*

Fiscal Management: Budgeting Main menu

Budget Review

Accesses: Program: *bgtreview*

Menuopt File: $CARSPATH/menuopt/accounting/programs/bgtr

Parameters Passed:

- *Fiscal year*
- *Output device*

Fiscal Management: Budgeting Main menu

Terminate G/L Accounts

Accesses: SQL script: *termacct*

Menuopt File: $CARSPATH/menuopt/accounting/informers/termacct

Parameters Passed:

- *Fiscal year (e.g., 9798)*
- *Fund*
- *Center*
- *Account*
- *Subfund*

Fiscal Management: Budgeting Main menu

Recompute Summary Records

Accesses: Program: *bgtalloc*

Menuopt File: $CARSPATH/menuopt/budget/programs/bgta.s

Parameters Passed:

- *Fund (e.g., 10)*

Fiscal Management: Budgeting: Reports: Objects

4 Column Detail

Accesses: ACE report bgtacctdtl

Menuopt File: $CARSPATH/menuopt/budget/others/acctd.2

Parameters Passed:

- *Sort Field*
- *Responsible Person*
- *First Fiscal Yr/Amount Type*
- *Second Fiscal Yr/Amount Type*
- *Third Fiscal Yr/Amount Type*
- *Fourth Fiscal Yr/Amount Type*
- *Fund Code Range*
- *Function Code Range*
- *Object Code Range*
- *Subfund Code Range*
- *Output Non-Display Object*
- *Subtotal by Schedule/Group*

Fiscal Management: Budgeting: Reports: Objects

1 Column Detail

Accesses: ACE report: bgtacctdtl

Menuopt File: $CARSPATH/menuopt/budget/others/acctd.1

Parameters Passed:

- *Sort Field*
- *Responsible Person*
- *Fiscal Yr*
- *Amount Type*
- *Fund Code Range*
- *Function Code Range*
- *Object Code Range*
- *Subfund Code Range*
- *Output Non-Display Object*
- *Subtotal by Schedule/Group*

Fiscal Management: Budgeting: Reports: Objects

Variance Detail

Accesses: ACE report: bgtacctdtl

Menuopt File: $CARSPATH/menuopt/budget/others/acctd.3

Parameters Passed:

- *Sort Field*
- *Responsible Person*
- *First Fiscal Yr/Amount Type*
- *Second Fiscal Yr/Amount Type*
- *Fund Code Range*
- *Function Code Range*
- *Object Code Range*
- *Subfund Code Range*
- *Output Non-Display Object*
- *Subtotal by Schedule/Group*

Fiscal Management: Budgeting: Reports: Objects

Detail by Month

Accesses: ACE report: bgtacctmon

Menuopt File: $CARSPATH/menuopt/budget/others/acctm.1

Parameters Passed:

- *Sort Field*
- *Responsible Person*
- *Output*
- *Fiscal Yr*
- *Amount Type*
- *Fund Code Range*
- *Function Code Range*
- *Object Code Range*

- *Subfund Code Range*
- *Output Non-Display Object*
- Subtotal by Schedule/Group

Fiscal Management: Budgeting: Reports: Objects

4 Column Summary

Accesses: ACE report: bgtacctsum

Menuopt File: $CARSPATH/menuopt/budget/others/accts.2

Parameters Passed:

- *Sort Field*
- *Responsible Person*
- *First Fiscal Yr/Amount Type*
- *Second Fiscal Yr/Amount Type*
- *Third Fiscal Yr/Amount Type*
- *Fourth Fiscal Yr/Amount Type*
- *Fund Code Range*
- *Function Code Range*
- *Object Code Range*
- *Subfund Code Range*
- *Output Non-Display Object*
- *Subtotal by Block/Group*
- *Subtotal by Schedule*
- *Summary*

Fiscal Management: Budgeting: Reports: Objects

1 Column Summary

Accesses: ACE report: bgtacctsum

Menuopt File: $CARSPATH/menuopt/budget/others/accts.1

Parameters Passed:

- *Sort Field*
- *Responsible Person*
- *Fiscal Yr*
- *Amount Type*
- *Fund Code Range*
- *Function Code Range*
- *Object Code Range*
- *Subfund Code Range*
- *Output Non-Display Object*
- *Subtotal by Block/Group*
- *Subtotal by Schedule*
- *Summary*

Fiscal Management: Budgeting: Reports: Objects

Variance Summary

Accesses: ACE report: bgtacctsum

Menuopt File: $CARSPATH/menuopt/budget/others/accts.3

Parameters Passed:

- *Sort Field*
- *Responsible Person*
- *First Fiscal Yr/Amount Type*
- *Second Fiscal Yr/Amount Type*
- *Fund Code Range*
- *Function Code Range*
- *Object Code Range*
- *Subfund Code Range*
- *Output Non-Display Object*
- *Subtotal by Block/Group*
- *Subtotal by Schedule*
- *Summary*

Fiscal Management: Budgeting: Reports: Functions

4 Column Detail

Accesses: ACE report: bgtcntrdtl

Menuopt File: $CARSPATH/menuopt/budget/others/cntrd.2

Parameters Passed:

- *Sort Field*
- *Responsible Person*
- *First Fiscal Yr/Amount Type*
- *Second Fiscal Yr/Amount Type*
- *Third Fiscal Yr/Amount Type*
- *Fourth Fiscal Yr/Amount Type*
- *Fund Code Range*
- *Function Code Range*
- *Object Code Range*
- *Subfund Code Range*
- *Output Non-Display Object*
- *Subtotal by Schedule/Group*

Fiscal Management: Budgeting: Reports: Functions

1 Column Detail

Accesses: ACE report: bgtcntrdtl

Menuopt File: $CARSPATH/menuopt/budget/others/cntrd.1

Parameters Passed:

- *Sort Field*
- *Responsible Person*
- *Fiscal Yr*
- *Amount Type*
- *Fund Code Range*
- *Function Code Range*
- *Object Code Range*
- *Subfund Code Range*
- *Output Non-Display Object*
- *Subtotal by Schedule/Group*

Fiscal Management: Budgeting: Reports: Functions

Variance Detail

Accesses: ACE report: bgtcntrdtl

Menuopt File: $CARSPATH/menuopt/budget/others/cntrd.3

Parameters Passed:

- *Sort Field*
- *Responsible Person*
- *First Fiscal Yr/Amount Type*
- *Second Fiscal Yr/Amount Type*
- *Fund Code Range*
- *Function Code Range*
- *Object Code Range*
- *Subfund Code Range*
- *Output Non-Display Object*
- *Subtotal by Schedule/Group*

Fiscal Management: Budgeting: Reports: Functions

Detail by Month

Accesses: ACE report: bgtcntrmon

Menuopt File: $CARSPATH/menuopt/budget/others/cntrm.1

Parameters Passed:

- *Sort Field*
- *Responsible Person*
- *Output*
- *Fiscal Yr*
- *Amount Type*
- *Fund Code Range*
- *Function Code Range*
- *Object Code Range*
- *Subfund Code Range*
- *Output Non-Display Object*
- *Summary*

Fiscal Management: Budgeting: Reports: Functions

4 Column Summary

Accesses: ACE report: bgtcntrsum

Menuopt File: $CARSPATH/menuopt/budget/others/cntrs.2

Parameters Passed:

- *Sort Field*
- *Responsible Person*
- *First Fiscal Yr/Amount Type*
- *Second Fiscal Yr/Amount Type*
- *Third Fiscal Yr/Amount Type*
- *Fourth Fiscal Yr/Amount Type*
- *Fund Code Range*
- *Function Code Range*
- *Object Code Range*
- *Subfund Code Range*
- *Output Non-Display Object*
- *Subtotal by Block/Group*
- *Subtotal by Schedule*
- *Print Responsible Person*

Fiscal Management: Budgeting: Reports: Functions

1 Column Summary

Accesses: ACE report: bgtcntrsum

Menuopt File: $CARSPATH/menuopt/budget/others/cntrs.1

Parameters Passed:

- *Sort Field*
- *Responsible Person*
- *Fiscal Yr*
- *Amount Type*
- *Fund Code Range*
- *Function Code Range*
- *Object Code Range*
- *Subfund Code Range*
- *Output Non-Display Object*
- *Subtotal by Block/Group*
- *Summary*

Fiscal Management: Budgeting: Reports: Functions

Variance Summary

Accesses: ACE report: bgtcntrsum

Menuopt File: $CARSPATH/menuopt/budget/others/cntrs.3

Parameters Passed:

- *Sort Field*
- *Responsible Person*
- *First Fiscal Yr/Amount Type*
- *Second Fiscal Yr/Amount Type*
- *Fund Code Range*
- *Function Code Range*
- *Object Code Range*
- *Subfund Code Range*
- *Output Non-Display Object*
- *Subtotal by Block/Group*
- *Summary*

Fiscal Management: Budgeting: Reports: Functions

Budget Profit Center

Accesses: ACE report: bgtcntrprf

Menuopt File: $CARSPATH/menuopt/budget/others/cntrp.2

Parameters Passed:

- *Sort Field*
- *Responsible Person*
- *First Fiscal Yr/Amount Type*
- *Second Fiscal Yr/Amount Type*
- *Third Fiscal Yr/Amount Type*
- *Fourth Fiscal Yr/Amount Type*
- *Fund Code Range*
- *Function Code Range*
- *Object Code Range*
- *Subfund Code Range*
- *Output Non-Display Object*
- *Subtotal by Block/Group*
- *Summary*

Fiscal Management: Budgeting: Reports: Functions

Budget Request Forms

Accesses: ACE report: bgtcntrdtl

Menuopt File: $CARSPATH/menuopt/budget/others/cntrdr.2

Parameters Passed:

- *Sort Field*
- *Responsible Person*
- *Output*
- *First Fiscal Yr/Amount Type*
- *Second Fiscal Yr/Amount Type*
- *Third Fiscal Yr/Amount Type*
- *Fourth Fiscal Yr/Amount Type*
- *Fifth Fiscal Yr/Amount Type*
- *Sixth Fiscal Yr/Amount Type*
- *Seventh Fiscal Yr/Amount Type*
- *Fund Code Range*
- *Function Code Range*
- *Object Code Range*
- *Subfund Code Range*
- *Output Non-Display Object*
- *Subtotal by Schedule/Group*

Fiscal Management: Budgeting: Reports: Subfunds

4 Column Summary

Accesses: ACE report: bgtprjsum

Menuopt File: $CARSPATH/menuopt/budget/others/prjsum.2

Parameters Passed:

- *Sort Field*
- *Responsible Person*
- *First Fiscal Yr/Amount Type*
- *Second Fiscal Yr/Amount Type*
- *Third Fiscal Yr/Amount Type*
- *Fourth Fiscal Yr/Amount Type*
- *Fund Code Range*
- *Function Code Range*
- *Object Code Range*
- *Subfund Code Range*
- *Output Non-Display Object*
- *Subtotal by Block/Group*
- *Summary*

Fiscal Management: Budgeting: Reports: Subfunds

1 Column Summary

Accesses: ACE report: bgtprjsum

Menuopt File: $CARSPATH/menuopt/budget/others/prjsum.1

Parameters Passed:

- *Sort Field*
- *Responsible Person*
- *Fiscal Yr*
- *Amount Type*
- *Fund Code Range*
- *Function Code Range*
- *Object Code Range*
- *Subfund Code Range*
- *Output Non-Display Object*
- *Subtotal by Block/Group*
- *Summary*

Fiscal Management: Budgeting: Reports: Subfunds

Variance Summary

Accesses: ACE report: bgtprjsum

Menuopt File: $CARSPATH/menuopt/budget/others/prjsum.3

Parameters Passed:

- *Sort Field*
- *Responsible Person*
- *First Fiscal Yr/Amount Type*
- *Second Fiscal Yr/Amount Type*
- *Fund Code Range*
- *Function Code Range*
- *Object Code Range*
- *Subfund Code Range*
- *Output Non-Display Object*
- *Subtotal by Block/Group*
- *Summary*

Fiscal Management: Budgeting: Reports: Subfunds

4 Column Object Detail

Accesses: ACE report: bgtprjadtl

Menuopt File: $CARSPATH/menuopt/budget/others/prjad.2

Parameters Passed:

- *Sort Field*
- *Responsible Person*
- *First Fiscal Yr/Amount Type*
- *Second Fiscal Yr/Amount Type*
- *Third Fiscal Yr/Amount Type*
- *Fourth Fiscal Yr/Amount Type*
- *Fund Code Range*
- *Function Code Range*
- *Object Code Range*
- *Subfund Code Range*
- *Output Non-Display Object*
- *Subtotal by Block/Group*

Fiscal Management: Budgeting: Reports: Subfunds

1 Column Object Detail

Accesses: ACE report: bgtprjadtl

Menuopt File: $CARSPATH/menuopt/budget/others/prjad.1

Parameters Passed:

- *Sort Field*
- *Responsible Person*
- *Fiscal Yr*
- *Amount Type*
- *Fund Code Range*
- *Function Code Range*
- *Object Code Range*
- *Subfund Code Range*
- *Output Non-Display Object*
- *Subtotal by Block/Group*

Fiscal Management: Budgeting: Reports: Subfunds

Variance Object Detail

Accesses: ACE report: bgtprjadtl

Menuopt File: $CARSPATH/menuopt/budget/others/prjad.3

Parameters Passed:

- *Sort Field*
- *Responsible Person*
- *First Fiscal Yr/Amount Type*
- *Second Fiscal Yr/Amount Type*
- *Fund Code Range*
- *Function Code Range*
- *Object Code Range*
- *Subfund Code Range*
- *Output Non-Display Object*
- *Subtotal by Block/Group*

Fiscal Management: Budgeting: Reports: Subfunds

Object Detail by Month

Accesses: ACE report: bgtprjamon

Menuopt File: $CARSPATH/menuopt/budget/others/prjam.1

Parameters Passed:

- *Sort Field*
- *Responsible Person*
- *Output*
- *Fiscal Yr*
- *Amount Type*
- *Fund Code Range*
- *Function Code Range*
- *Object Code Range*
- *Subfund Code Range*
- *Output Non-Display Object*
- *Summary*

Fiscal Management: Budgeting: Reports: Subfunds

4 Column Object Summary

Accesses: ACE report: bgtprjsum

Menuopt File: $CARSPATH/menuopt/budget/others/prjsum.2

Parameters Passed:

- *Sort Field*
- *Responsible Person*
- *First Fiscal Yr/Amount Type*
- *Second Fiscal Yr/Amount Type*
- *Third Fiscal Yr/Amount Type*
- *Fourth Fiscal Yr/Amount Type*
- *Fund Code Range*
- *Function Code Range*
- *Object Code Range*
- *Subfund Code Range*
- *Output Non-Display Object*
- *Subtotal by Block/Group*
- *Subtotal by Schedule*
- *Print Responsible Person*

Fiscal Management: Budgeting: Reports: Subfunds

1 Column Object Summary

Accesses: ACE report: bgtprjasum

Menuopt File: $CARSPATH/menuopt/budget/others/prjas.1

Parameters Passed:

- *Sort Field*
- *Responsible Person*
- *Fiscal Yr*
- *Amount Type*
- *Fund Code Range*
- *Function Code Range*
- *Object Code Range*
- *Subfund Code Range*
- *Output Non-Display Object*
- *Subtotal by Block/Group*
- *Summary by Schedule*
- *Summary*

Fiscal Management: Budgeting: Reports: Subfunds

Variance Object Summary

Accesses: ACE report: bgtprjasum

Menuopt File: $CARSPATH/menuopt/budget/others/prjas.3

Parameters Passed:

- *Sort Field*
- *Responsible Person*
- *First Fiscal Yr/Amount Type*
- *Second Fiscal Yr/Amount Type*
- *Fund Code Range*
- *Function Code Range*
- *Object Code Range*
- *Subfund Code Range*
- *Output Non-Display Object*
- *Subtotal by Block/Group*
- *Subtotal by Schedule*
- *Summary*

Fiscal Management: Budgeting: Reports: Subfunds

4 Column Function Detail

Accesses: ACE report: bgtprjcdtl

Menuopt File: $CARSPATH/menuopt/budget/others/prjcd.2

Parameters Passed:

- *Sort Field*
- *Responsible Person*
- *First Fiscal Yr/Amount Type*
- *Second Fiscal Yr/Amount Type*
- *Third Fiscal Yr/Amount Type*
- *Fourth Fiscal Yr/Amount Type*
- *Fund Code Range*
- *Function Code Range*
- *Object Code Range*
- *Subfund Code Range*
- *Output Non-Display Object*
- *Subtotal by Schedule/Group*

Fiscal Management: Budgeting: Reports: Subfunds

1 Column Function Detail

Accesses: ACE report: bgtprjcdtl

Menuopt File: $CARSPATH/menuopt/budget/others/prjcd.1

Parameters Passed:

- *Sort Field*
- *Responsible Person*
- *Fiscal Yr*
- *Amount Type*
- *Fund Code Range*
- *Function Code Range*
- *Object Code Range*
- *Subfund Code Range*
- *Output Non-Display Object*
- *Subtotal by Schedule/Group*

Fiscal Management: Budgeting: Reports: Subfunds

Variance Function Detail

Accesses: ACE report: bgtprjcdtl

Menuopt File: $CARSPATH/menuopt/budget/others/prjcd.3

Parameters Passed:

- *Sort Field*
- *Responsible Person*
- *First Fiscal Yr/Amount Type*
- *Second Fiscal Yr/Amount Type*
- *Fund Code Range*
- *Function Code Range*
- *Object Code Range*
- *Subfund Code Range*
- *Output Non-Display Object*
- *Subtotal by Schedule/Group*

Fiscal Management: Budgeting: Reports: Subfunds

Function Detail by Month

Accesses: ACE report: bgtprjcmon

Menuopt File: $CARSPATH/menuopt/budget/others/prjcm.1

Parameters Passed:

- *Sort Field*
- *Responsible Person*
- *Output*
- *Fiscal Yr*
- *Amount Type*
- *Fund Code Range*
- *Function Code Range*
- *Object Code Range*
- *Subfund Code Range*
- *Output Non-Display Object*
- *Summary*

Fiscal Management: Budgeting: Reports: Subfunds

4 Column Function Summary

Accesses: ACE report: bgtprjcsum

Menuopt File: $CARSPATH/menuopt/budget/others/prjcs.2

Parameters Passed:

- *Sort Field*
- *Responsible Person*
- *First Fiscal Yr/Amount Type*
- *Second Fiscal Yr/Amount Type*
- *Third Fiscal Yr/Amount Type*
- *Fourth Fiscal Yr/Amount Type*
- *Fund Code Range*
- *Function Code Range*
- *Object Code Range*
- *Subfund Code Range*
- *Output Non-Display Object*
- *Subtotal by Block/Group*
- *Summary*

Fiscal Management: Budgeting: Reports: Subfunds

1 Column Function Summary

Accesses: ACE report: bgtprjcsum

Menuopt File: $CARSPATH/menuopt/budget/others/prjcs.1

Parameters Passed:

- *Sort Field*
- *Responsible Person*
- *Fiscal Yr*
- *Amount Type*
- *Fund Code Range*
- *Function Code Range*
- *Object Code Range*
- *Subfund Code Range*
- *Output Non-Display Object*
- *Subtotal by Block/Group*
- *Summary*

Fiscal Management: Budgeting: Reports: Subfunds

Variance Function Summary

Accesses: ACE report: bgtprjcsum

Menuopt File: $CARSPATH/menuopt/budget/others/prjcs.3

Parameters Passed:

- *Sort Field*
- *Responsible Person*
- *First Fiscal Yr/Amount Type*
- *Second Fiscal Yr/Amount Type*
- *Fund Code Range*
- *Function Code Range*
- *Object Code Range*
- *Subfund Code Range*
- *Output Non-Display Object*
- *Subtotal by Block/Group*
- *Summary*

Fiscal Management: Budgeting: Reports: Subfunds

Budget Request Forms

Accesses: ACE report: bgtprjcdtl

Menuopt File: $CARSPATH/menuopt/budget/others/prjcdr.2

Parameters Passed:

- *Sort Field*
- *Responsible Person*
- *Final or Requested Output*
- *First Fiscal Yr/Amount Type*
- *Second Fiscal Yr/Amount Type*
- *Third Fiscal Yr/Amount Type*
- *Fourth Fiscal Yr/Amount Type*
- *Fifth Fiscal Yr/Amount Type*
- *Sixth Fiscal Yr/Amount Type*
- *Seventh Fiscal Yr/Amount Type*
- *Fund Code Range*
- *Function Code Range*
- *Object Code Range*
- *Subfund Code Range*
- *Output Non-Display Object*
- *Subtotal by Schedule/Group*

Summary Fiscal Management: Budgeting Table Maintenance: Budgeting (A-Z) menu

Budget Adjustment

Accesses: PERFORM screen: *bgtacct*

Menuopt File: $CARSPATH/menuopt/budget/screens/bgtacct

Parameters Passed: None

Fiscal Management: Budgeting Table Maintenance: Budgeting (A-Z) menu

Budget Calendar

Accesses: PERFORM screen: *bgtcal*

Menuopt File: $CARSPATH/menuopt/budget/screens/bgtcal

Parameters Passed: None

Fiscal Management: Budgeting Table Maintenance: Budgeting (A-Z) menu

Budget Distribution

Accesses: PERFORM screen: *bgtdist*

Menuopt File: $CARSPATH/menuopt/budget/screens/bgtdist

Parameters Passed: None

Fiscal Management: Budgeting Table Maintenance: Budgeting (A-Z) menu

Budget Parameter

Accesses: PERFORM screen: *pbgt*

Menuopt File: $CARSPATH/menuopt/budget/screens/pbgt

Parameters Passed: None

Fiscal Management: Budgeting Table Maintenance: Other Financial (A-Z) menu

Amount Type

Accesses: PERFORM screen: *tatype*

Menuopt File: $CARSPATH/menuopt/accounting/screens/tatype

Parameters Passed: None

Fiscal Management: Budgeting Table Maintenance: Other Financial (A-Z) menu

Defined Account

Accesses: PERFORM screen: *gldefine*

Menuopt File: $CARSPATH/menuopt/accounting/screens/gldefine

Parameters Passed: None

Fiscal Management: Budgeting Table Maintenance: Other Financial (A-Z) menu

Financial Statement

Accesses: PERFORM screen: tfs

Menuopt File: $CARSPATH/menuopt/accounting/screens/tfs

Parameters Passed: None

Fiscal Management: Budgeting Table Maintenance: Other Financial (A-Z) menu

Fiscal Calendar

Accesses: PERFORM screen: *fiscalcal*

Menuopt File: $CARSPATH/menuopt/accounting/screens/fiscalcal

Parameters Passed: None

Fiscal Management: Budgeting Table Maintenance: Other Financial (A-Z) menu

General Ledger

Accesses: PERFORM screen: *glfieldrpt*

Menuopt File: $CARSPATH/menuopt/accounting/screens/glfieldrpt

Parameters Passed: None

Fiscal Management: Cash Receipts menu

Cashier

Accesses: Program: *cashier*

Menuopt File: $CARSPATH/menuopt/accounting/programs/cash

Parameters Passed:

- *-j (journal type; default is CH)*
- *-d (debit/credit indicator; default is defined in the DOC_CR_DEF macro)*
- *-l (loop within entry class)*
- *-c (allow financial clearance)*
- *-v (verify before starting or posting a journal)*

Fiscal Management: Cash Receipts menu

Batch Fee Table Checking

Accesses: Csh script: feetable

Menuopt File: $CARSPATH/menuopt/accounting/scripts/feetable

Parameters Passed:

- None

Note: This option is available if the ENABLE_FEE_COLLECTION macro is set to Y.

Fiscal Management: Cash Receipts menu

Bursar Query

Accesses: Program: *bursar*

Menuopt File: $CARSPATH/menuopt/accounting/programs/bsr

Parameters Passed:

- None

Fiscal Management: Cash Receipts menu

Claim-on-Cash Update for Fee Collection

Accesses: ACE report: allocpymts

Menuopt File: $CARSPATH/menuopt/accounting/reports/allocpymts

Parameters Passed:

- *Subsidiary Tot Period*
- *Fiscal Year*
- *Beginning Fiscal Period*
- *Ending Fiscal Period*
- *Clearing Fund Object Payments*
- *Clearing Fund Objects Collections*
- *Other Funds Object Payments*
- *Other Funds Object Collections*

Note: This option is available if the CLAIM_ON_CASH_MENU macro is set to Y.

Fiscal Management: Cash Receipts menu

Post Claim-on-Cash Update for Fee Collection

Accesses: Csh script: allocpymts

Menuopt File: $CARSPATH/menuopt/accounting/scripts/allocpymts

Parameters Passed:

- *Subsidiary Tot Period*
- *Fiscal Year*
- *Beginning Fiscal Period*
- *Ending Fiscal Period*
- *Clearing Fund Object Payments*
- *Clearing Fund Objects Collections*
- *Other Funds Object Payments*
- *Other Funds Object Collections*

Note: This option is available if the CLAIM_ON_CASH_MENU macro is set to Y.

Fiscal Management: Cash Receipts menu

Accounting Query

Accesses: Program: *acquery*

Menuopt File: $CARSPATH/menuopt/accounting/programs/acqu.prWP

Parameters Passed:

- *-s (start on the subsidiary side using saquery)*
- *-r (subsidiary restrictions; default is defined in the SUBS_PR_DEF macro)*

Fiscal Management: Cash Receipts menu

Daily Station Reports

Accesses: Csh script: *daycash*

Menuopt File: $CARSPATH/menuopt/accounting/scripts/daycash

Parameters Passed:

- -f (formtype, default is *Standard*)

Fiscal Management: Cash Receipts menu

Document Register

Accesses: ACE report: jrnldocreg

Menuopt File: $CARSPATH/menuopt/accounting/reports/jrnldoc.ch

Parameters Passed:

- None

Fiscal Management: Cash Receipts menu

G/L Journal Reports

Accesses: Csh script: *jrnlgl*

Menuopt File: $CARSPATH/menuopt/accounting/scripts/jrnlgl.CH

Parameters Passed:

- -f (formtype; default is *Standard*)

Fiscal Management: Cash Receipts menu

S/L Journal Reports

Accesses: Csh script: jrnlsl

Menuopt File: $CARSPATH/menuopt/accounting/scripts/jrnlsl.CH

Parameters Passed:

- -f (formtype; default is *Standard*)

Fiscal Management: Cash Receipts menu

Pre-Reconciliation Report

Accesses: ACE report: prerecon

Menuopt File: $CARSPATH/menuopt/accounting/reports/prerecon

Parameters Passed:

- *Station number*

Fiscal Management: Cash Receipts menu

Reconciliation Report

Accesses: ACE report: jrnlrecon

Menuopt File: $CARSPATH/menuopt/accounting/reports/jrnlrecon

Parameters Passed:

- *Journal type (default is CH)*
- *Station number*
- *Report date (default is current day)*

Fiscal Management: Cash Receipts: Third-Party Billing menu

Produce Invoices

Accesses: Program: *invdef*

Menuopt File: $CARSPATH/menuopt/accounting/programs/invdef

Parameters Passed:

- *Agency*
- *Date*
- *Balance Period*
- *Station Number*

Note: This option is available if the ENABLE_FEE_COLLECTION macro is set to Y.

Fiscal Management: Cash Receipts: Third-Party Billing menu

Print Invoices

Accesses: Program: *fps*

Menuopt File: $CARSPATH/menuopt/utilities/programs/fps.inv

Parameters Passed:

- None

Note: This option is available if the ENABLE_FEE_COLLECTION macro is set to Y.

Fiscal Management: Cash Receipts: Third-Party Billing menu

Cashier

Accesses: Program: *cashier*

Menuopt File: $CARSPATH/menuopt/accounting/programs/cashier

Parameters Passed:

- Journal type (default is CH)
- -d (value of DOC_CR_DEF)
- *Balance Period*
- -b (reflect transactions that are not posted in student balances)
- -l (causes loop within entry class)
- -c (enables financial clearance)
- -v (verifies before starting or posting journal)

Note: This option is available if the ENABLE_FEE_COLLECTION macro is set to Y.

Fiscal Management: Cash Receipts: Third-Party Billing menu

Reconcile Agency Payment

Accesses: Program: *defrec*

Menuopt File: $CARSPATH/menuopt/accounting/programs/defrec

Parameters Passed:

- *None*

Note: This option is available if the ENABLE_FEE_COLLECTION macro is set to Y.

Fiscal Management: Cash Receipts: Third-Party Billing menu

Deferment Audit Report

Accesses: Program: *defaudit*

Menuopt File: $CARSPATH/menuopt/accounting/programs/defaudit

Parameters Passed:

- *Session/Year*
- *Audit Subsidiary*
- -m (Mails audit output to user)
- -u (Corrects audit output during processing)

Note: This option is available if the ENABLE_FEE_COLLECTION macro is set to Y.

Fiscal Management: Cash Receipts: Third-Party Billing menu

Outstanding Invoice Report

Accesses: ACE report: deferinv

Menuopt File: $CARSPATH/menuopt/accounting/reports/deferinv

Parameters Passed:

- None

Note: This option is available if the ENABLE_FEE_COLLECTION macro is set to Y.

Fiscal Management: Cash Receipts: Third-Party Billing menu

Non-Invoiced Recv Report

Accesses: ACE report: defernoinv

Menuopt File: $CARSPATH/menuopt/accounting/programs/defernoinv

Parameters Passed:

- None

Note: This option is available if the ENABLE_FEE_COLLECTION macro is set to Y.

Fiscal Management: Fixed Assets Main menu

Enter Assets

Accesses: PERFORM screen: *fix*

Menuopt File: $CARSPATH/menuopt/fixassets/screens/fix

Parameters Passed:

- None

Fiscal Management: Fixed Assets Main menu

Verify Asset Entries

Accesses: Program: *fixpost*

Menuopt File: $CARSPATH/menuopt/fixassets/programs/fixp.r

Parameters Passed:

- -c (Fixed Asset calculation date; default is *Current date*)
- -r (Verify with/without posting; default is *Verify without posting*)

Fiscal Management: Fixed Assets Main menu

Post Assets to G/L

Accesses: Program: *fixpost*

Menuopt File: $CARSPATH/menuopt/fixassets/programs/fixp

Parameters Passed: None

Fiscal Management: Fixed Assets Main menu

Verify Single Asset Entry

Accesses: Program: *fixpost*

Menuopt File: $CARSPATH/menuopt/fixassets/programs/fixp.rn

Parameters Passed:

- -r (Verify with/without posting; default is *Verify without posting*)

Fiscal Management: Fixed Assets Main menu

Post Single Asset to G/L

Accesses: Program: *fixpost*

Menuopt File: $CARSPATH/menuopt/fixassets/programs/fixp.n

Parameters Passed: None

Fiscal Management: Fixed Assets: Reports

Asset Acquisitions

Accesses: ACE report: acquire

Menuopt File: $CARSPATH/menuopt/fixassets/reports/acquire

Parameters Passed:

- Beginning Date
- Ending Date

Fiscal Management: Fixed Assets: Reports

Asset Disposals

Accesses: ACE report: dispose

Menuopt File: $CARSPATH/menuopt/fixassets/reports/dispose

Parameters Passed:

- Beginning Date
- Ending Date

Fiscal Management: Fixed Assets: Reports

Asset List by Building

Accesses: ACE report: fixloc

Menuopt File: $CARSPATH/menuopt/fixassets/reports/fixloc

Parameters Passed:

- *Building name*
- *Include totals flag*

Fiscal Management: Fixed Assets: Reports

Asset List by Person

Accesses: ACE report: fixpers

Menuopt File: $CARSPATH/menuopt/fixassets/reports/fixpers

Parameters Passed:

- *ID*

Fiscal Management: Fixed Assets: Reports

Asset List by Type

Accesses: ACE report: fixtype

Menuopt File: $CARSPATH/menuopt/fixassets/reports/fixtype

Parameters Passed:

- *Asset type*
- *Include total flag*

Fiscal Management: Fixed Assets: Reports

Asset List for Insurance

Accesses: ACE report: insure

Menuopt File: $CARSPATH/menuopt/fixassets/reports/insure

Parameters Passed:

- *Asset type*
- *Date*
- *Include total flag*

Fiscal Management: Fixed Assets: Reports

Asset Values/Non-Summary

Accesses: ACE report: assetacct

Menuopt File: $CARSPATH/menuopt/fixassets/reports/assetacct

Parameters Passed:

- *Date*

Fiscal Management: Fixed Assets: Reports

Asset Values/Summary

Accesses: ACE report: sumlist

Menuopt File: $CARSPATH/menuopt/fixassets/reports/sumlist

Parameters Passed:

- *Date*

Fiscal Management: Fixed Assets: Reports

Depreciation/Non-Summary

Accesses: ACE report: depreciation

Menuopt File: $CARSPATH/menuopt/fixassets/reports/depreciation

Parameters Passed:

- *Date*
- *Include G/L transaction detail flag*

Fiscal Management: Fixed Assets: Reports

Depreciation/Summary

Accesses: ACE report: sumdepr

Menuopt File: $CARSPATH/menuopt/fixassets/reports/sumdepr

Parameters Passed:

- *Date*

Fiscal Management: Fixed Assets: Reports

Specific Asset List

Accesses: ACE report: assetsrpt

Menuopt File: $CARSPATH/menuopt/fixassets/others/assetsrpt

Parameters Passed:

- *Limit field (for sorting)*
- *Limit value*

Fiscal Management: Fixed Assets: Reports

G/L Journal Reports

Accesses: Csh script: jrnlgl

Menuopt File: $CARSPATH/menuopt/accounting/scripts/jrnlgl.FX

Parameters Passed:

- *Beginning journal number*
- *Ending journal number*
- *Output device*

Fiscal Management: Other Receivables Main menu

Receivables Entry

Accesses: Program: *voucher*

Menuopt File: $CARSPATH/accounting/programs/vch.AR

Parameters Passed:

- None

Fiscal Management: Other Receivables Main menu

G/L Journal Reports

Accesses: Csh script: jrnlgl

Menuopt File: $CARSPATH/menuopt/accounting/scripts/jrnlgl.AR

Parameters Passed:

- *Journal reference*
- *Beginning journal number*
- *Ending journal number*
- *Output device*

Fiscal Management: Other Receivables Main menu

S/L Journal Reports

Accesses: Csh script: jrnlsl

Menuopt File: $CARSPATH/menuopt/accounting/scripts/jrnlsl.AR

Parameters Passed:

- *Journal reference*
- *Beginning journal number*
- *Ending journal number*
- *Output device*

Fiscal Management: Other Receivables Main menu

Enter ID/Subsidiary Data

Accesses: PERFORM screen: idsuba

Menuopt File: $CARSPATH/menuopt/acctsrecv/screens/idsuba

Parameters Passed:

- None

Fiscal Management: Other Receivables Main menu

S/L Account Query

Accesses: Program: *acquery*

Menuopt File: $CARSPATH/menuopt/accounting/programs/acqu.SA

Parameters Passed:

- *Output device*
- -l (Lock into one side of program)
- -s (Start on subsidiary side of program)
- -r (Subsidiary restrictions)

Fiscal Management: Other Receivables: Interest menu

Interest Parameters

Accesses: PERFORM screen: intbill

Menuopt File: $CARSPATH/menuopt/stubill/screens/intbill_AR

Parameters Passed:

- None

Fiscal Management: Other Receivables: Interest menu

Pre-Interest Proof Report

Accesses: Program: *billing*

Menuopt File: $CARSPATH/menuopt/stubill/programs/int.rvo

Parameters Passed:

- *Billing parameter run code*
- *Output device*
- *Sort order*
- -v (Print charge verification report)

Fiscal Management: Other Receivables: Interest menu

Test Interest Charges

Accesses: Program: *billing*

Menuopt File: $CARSPATH/menuopt/int.r

Parameters Passed:

- *Billing parameter run code*

Fiscal Management: Other Receivables: Interest menu

Post Interest Charges

Accesses: Program: *billing*

Menuopt File: $CARSPATH/menuopt/int.re

Parameters Passed:

- *Billing parameter run code*
- *Date*
- -e (Post charges)

Fiscal Management: Other Receivables: Letters menu

Select One Dunning Letter

Accesses: Screen: dunctc_AR

Menuopt File: $CARSPATH/menuopt/acctsrecv/screens/dunctc_AR

Parameters Passed:

- None

Fiscal Management: Other Receivables: Letters menu

Print Dunning Letters

Accesses: Utility program: *lps*

Menuopt File: $CARSPATH/menuopt/utilities/programs/lps.dun

Parameters Passed:

- None

Fiscal Management: Other Receivables: Reports menu

Receivables Aging

Accesses: ACE report: recvaging

Menuopt File: $CARSPATH/menuopt/acctsrecv/reports/recvaging

Parameters Passed:

- *Subsidiary*
- *As of date*

Fiscal Management: Other Receivables: Reports menu

Receivables Forecast

Accesses: ACE report: recvfore

Menuopt File: $CARSPATH/menuopt/acctsrecv/reports/recvfore

Parameters Passed:

- *Subsidiary*
- *Subsidiary period*
- *Beginning date*

Fiscal Management: Other Receivables: Reports menu

Receivables Late

Accesses: ACE report: recvlate

Menuopt File: $CARSPATH/menuopt/acctsrecv/reports/recvlate

Parameters Passed:

- *Subsidiary*
- *Subsidiary period*
- *Beginning date*

Fiscal Management: Other Receivables: Reports menu

Receivables One Period

Accesses: ACE report: recvprd

Menuopt File: $CARSPATH/menuopt/acctsrecv/reports/recvprd

Parameters Passed:

- *Subsidiary*
- *Subsidiary period*
- *Beginning date*

Fiscal Management: Other Receivables: Reports menu

Receivables All Periods

Accesses: ACE report: recvprds

Menuopt File: $CARSPATH/menuopt/acctsrecv/reports/recvprds

Parameters Passed:

- *Subsidiary*
- *Beginning date*

Fiscal Management: Other Receivables: Reports menu

S/L Account Balances

Accesses: ACE report: subbalance

Menuopt File: $CARSPATH/menuopt/accounting/reports/subbal.AR

Parameters Passed:

- *Subsidiary*
- *Report date*
- *Debit balances*
- *Credit balances*
- *Zero balances*
- *Include Actual Amounts flag*
- *Include Encumbered Amounts flag*
- *Include Additional Information flag*
- *Subprogram (if enabled)*

Fiscal Management: Other Receivables: Reports menu

S/L Entries by Date

Accesses: ACE report: subentdate

Menuopt File: $CARSPATH/menuopt/accounting/reports/subedt.AR

Parameters Passed:

- *Subsidiary*
- *ID*
- *Beginning date*
- *Ending date*
- *Subsidiary Balance code*
- *Entry type*
- *Amount type*
- *Summary Report flag*

Fiscal Management: Other Receivables: Reports menu

S/L Entries by G/L Period

Accesses: ACE report: subentprds

Menuopt File: $CARSPATH/menuopt/accounting/reports/subepds.AR

Parameters Passed:

- *Subsidiary*
- *ID*
- *Beginning period*
- *Ending period*
- *Fiscal year*
- *Subsidiary Balance code*
- *Entry type*
- *Amount type*
- *Summary Report flag*

Fiscal Management: Other Receivables: Reports menu

S/L Entries by S/L Period

Accesses: ACE report: subentprd

Menuopt File: $CARSPATH/menuopt/accounting/reports/subepd.AR

Parameters Passed:

- *Subsidiary*
- *ID*
- *Subsidiary period*
- *Fiscal year*
- *Subsidiary Balance code*
- *Entry type*
- *Summary Report flag*

Fiscal Management: Other Receivables: Reports menu

S/L Transactions by Date

Accesses: ACE report: subtrdate

Menuopt File: $CARSPATH/menuopt/accounting/reports/subtrd.AR

Parameters Passed:

- *Subsidiary*
- *ID*
- *Beginning date*
- *Ending date*
- *Subsidiary Balance code*
- *Subsidiary Total code*
- *Entry type*
- *Page Break flag*

Fiscal Management: Other Receivables: Reports menu

S/L Transactions by Entry

Accesses: ACE report: subtrent

Menuopt File: $CARSPATH/menuopt/accounting/reports/subtre.AR

Parameters Passed:

- *Subsidiary*
- *ID*
- *Subsidiary period*
- *Fiscal year*
- *Subsidiary Balance code*
- *Entry type*

Fiscal Management: Other Receivables: Reports menu

S/L Transactions by Total

Accesses: ACE report subtrtot

Parameters Passed: $CARSPATH/menuopt/accounting/reports/subtrt.AR

- *Subsidiary*
- *ID*
- *Subsidiary period*
- *Fiscal year*
- *Subsidiary Balance code*
- *Subsidiary Total code*

Fiscal Management: Other Receivables: Reports menu

Subsidiary History

Accesses: ACE report: subtrhist

Menuopt File: $CARSPATH/menuopt/accounting/reports/subtrh.AR

Parameters Passed:

- *Subsidiary*
- *ID*
- *Beginning date*
- *Ending date*

Fiscal Management: Other Receivables: Reports menu

Total Balances/Person

Accesses: ACE report: substot

Menuopt File: $CARSPATH/menuopt/accounting/reports/substot.AR

Parameters Passed:

- *Subsidiary*
- *Subsidiary period*
- *Fiscal year*
- *Subsidiary Total code*
- *ID*
- *Subsidiary Total Posting code A (financial aid)*
- *Subsidiary Total Posting code B (automated student billing)*
- *Subsidiary Total Posting code C (manual student billing)*
- *Subsidiary Total Posting code D (display manual student billing)*
- *Subsidiary Total Posting code I (interest)*
- *Subsidiary Total Posting code L (loans and other credits)*
- *Subsidiary Total Posting code M (manual charges and refunds)*
- *Subsidiary Total Posting code Y (payments)*

Fiscal Management: Other Receivables: Statements menu

Statement Parameters

Accesses: PERFORM screen: pstmt_AR

Menuopt File: $CARSPATH/menuopt/acctsrecv/screens/pstmt_AR

Parameters Passed:

- None

Fiscal Management: Other Receivables: Statements menu

Add Billing Recipients

Accesses: PERFORM screen: sbscr_AR

Menuopt File: $CARSPATH/menuopt/stubill/screens/sbscr_AR

Parameters Passed:

- None

Fiscal Management: Other Receivables: Statements menu

Select One Statement

Accesses: PERFORM screen: stmtctc _AR

Menuopt File: $CARSPATH/menuopt/acctsrecv/screens/stmtctc_AR

Parameters Passed:

- None

Fiscal Management: Other Receivables: Statements menu

Select Statements

Accesses: Csh script: stmtctc_AR

Menuopt File: $CARSPATH/menuopt/acctsrecv/scripts/stmtctc_AR

Parameters Passed:

- *Group number*
- *Subprogram*
- *Date*
- *Credit balances*
- *Debit balances*
- *Zero balances*

Fiscal Management: Other Receivables: Statements menu

Create Statements

Accesses: Program: *stmt*

Menuopt File: $CARSPATH/menuopt/acctsrecv/programs/stmt

Parameters Passed:

- *Group number*

Fiscal Management: Other Receivables: Statements menu

Print Statements

Accesses: Utility program: *fps*

Menuopt File: $CARSPATH/menuopt/utilities/programs/fps.stmt

Parameters Passed:

- None

Fiscal Management: Purchasing Main menu

Enter Purchase Orders

Accesses: Program: *purch*

Menuopt File: $CARSPATH/menuopt/purchasing/programs/purc.pBG

Parameters Passed:

- -p (Purchase order permission code; default is p)
- -f (Purchase order form type; default is f)
- -m (Purchase order *bgvoucher* mode; default is P)
- -i (Purchase order level code; default is S)
- *Subsidiary (if ENABLE_MULTI_AP_SUBS = Y)*
- *Document code (if ENABLE_MULTI_DOC_PO = Y)*
- *Date*
- *Output device*
- *Purchase order form (if ENABLE_FORM_PO = Y)*

Fiscal Management: Purchasing Main menu

Enter Manual Invoices

Accesses: Program: *voucher*

Menuopt File: $CARSPATH/menuopt/accounting/programs/vch.AP

Parameters Passed:

- -v (Journal type; default is *AP*)

Note: This option is available if ENABLE_MOD_ACCTS_PAYABLE = Y.

Fiscal Management: Purchasing Main menu

Adjust Manual Invoices

Accesses: Program: *voucher*

Menuopt File: $CARSPATH/menuopt/accounting/programs/vch.ACinv

Parameters Passed:

- -v (Journal type; default is *AP*)

Note: This option is available if ENABLE_MOD_ACCTS_PAYABLE = Y.

Fiscal Management: Purchasing Main menu

G/L Journal Reports

Accesses: Csh script: jrnlgl

Menuopt File: $CARSPATH/menuopt/accounting/scripts/jrnlg.APPC

Parameters Passed:

- *Journal reference*
- *Beginning journal number*
- *Ending journal number*
- *Output device*

Fiscal Management: Purchasing Main menu

S/L Journal Reports

Accesses: Csh script: jrnlsl

Menuopt File: $CARSPATH/menuopt/accounting/scripts/jrnls.APPC

Parameters Passed:

- *Journal reference*
- *Beginning journal number*
- *Ending journal number*
- *Output device*

Fiscal Management: Purchasing Main menu

Vendor Entry

Accesses: Program: *vndentry*

Menuopt File: $CARSPATH/menuopt/purchasing/programs/vnde

Parameters Passed:

- -f (Form selected (name of desired form); default is *vnd*)
- -a (Query or Auto-Insert mode; default is to *automatically enter Query mode*)
- -F (Forced query before permitting insert; default is to *require query before insertions*)
- -D (Debug level, where the higher the number (1-9), the more messages the program returns; default is *3*)

Fiscal Management: Purchasing: Reports menu

Open Encumbrances

Accesses: ACE report: openpoenc

Menuopt File: $CARSPATH/menuopt/purchasing/reports/openpoenc

Parameters Passed:

- None

Fiscal Management: Purchasing: Reports menu

Purchase Order History

Accesses: ACE report: pohist

Menuopt File: $CARSPATH/menuopt/purchasing/reports/pohist

Parameters Passed:

- *Document code (if ENABLE_MULTI_DOC_PO = Y)*
- *Beginning document number*
- *Ending document number*

Fiscal Management: Purchasing: Reports menu

S/L Account Balances

Accesses: ACE report: subbalance

Menuopt File: $CARSPATH/menuopt/accounting/reports/subbal.AP]

Parameters Passed:

- *Subsidiary (if ENABLE_MULTI_AP_SUBS = Y)*
- *Date*
- *Credit balances*
- *Debit balances*
- *Zero balances*
- *Include Actual Amounts flag*
- *Include Encumbered Amounts flag*
- *Subprogram (if ENABLE_FEAT_SUBPROG = Y)*

Fiscal Management: Purchasing: Reports menu

Accesses: ACE report: vndlst

Menuopt File: $CARSPATH/menuopt/acctspay/reports/vndlst

Parameters Passed:

- None

Fiscal Management: Purchasing/AP Main menu

Enter PO's and Invoices

Accesses: Program: *purch*

Menuopt File: $CARSPATH/menuopt/purchasing/programs/purc.apBG

Parameters Passed:

- *Subsidiary (if ENABLE_MULTI_AP_SUBS = Y)*
- *Document code (if ENABLE_MULTI_DOC_PO = Y)*
- *Date*
- *Output device*
- *Purchase order form type (if ENABLE_MULTI_FORM_PO = Y)*

Fiscal Management: Purchasing/AP Main menu

Enter Manual Invoices

Accesses: Program: *voucher*

Menuopt File: $CARSPATH/menuopt/accounting/programs/vch.AP

Parameters Passed:

- None

Fiscal Management: Purchasing/AP Main menu

Adjust Manual Invoices

Accesses: Program: *voucher*

Menuopt File: $CARSPATH/menuopt/accounting/programs/vch.ACinv

Parameters Passed:

- None

Fiscal Management: Purchasing/AP Main menu

G/L Journal Reports

Accesses: Csh script: jrnlgl

Menuopt File: $CARSPATH/menuopt/accounting/scripts/jrnlg.APPC

Parameters Passed:

- -f (Formtype; default is *standard*)

Fiscal Management: Purchasing/AP Main menu

S/L Journal Reports

Accesses: Csh script: jrnlsl

Menuopt File: $CARSPATH/menuopt/accounting/scripts/jrnls.APPC

Parameters Passed:

- -f (Formtype; default is *standard*)

Fiscal Management: Purchasing/AP Main menu

Vendor Entry

Accesses: Entry program: *vndentry*

Menuopt File: $CARSPATH/menuopt/purchasing/programs/vnde

Parameters Passed:

- -f (Form selected (name of desired form); default is *vnd*)
- -a (Query or Auto-Insert mode; default is to *automatically enter Query mode*)
- -F (Forced query before permitting insert; default is to *require query before insertions*)
- -D (Debug level, where the higher the number (1-9), the more messages the program returns; default is *3*)

Fiscal Management: Purchasing/AP Main menu

Approve Purchase Orders

Accesses: Program: *approve*

Menuopt File: $CARSPATH/menuopt/purchasing/programs/puap.cd

Parameters Passed:

- *Print flag*
- *Output device*
- *Date*

Note: This option is available if ENABLE_FEAT_PO_APPROVAL = Y.

Fiscal Management: Purchasing/AP Main menu

Accounting Query

Accesses: Program: *acquery*

Menuopt File: $CARSPATH/menuopt/accounting/programs/acqu.p

Parameters Passed:

- *Output device*

Note: This option is available if ENABLE_FEAT_PO_APPROVAL = Y.

Fiscal Management: Purchasing/AP Main menu

Budget Review

Accesses: Program: *bgtreview*

Menuopt File: $CARSPATH/menuopt/accounting/programs/bgtr

Parameters Passed:

- *Fiscal year*
- *Output device*

Note: This option is available if ENABLE_FEAT_PO_APPROVAL = Y.

Fiscal Management: Purchasing/AP: Check Writing menu

Select Check Group

Accesses: PERFORM screen: *ckgrp*

Menuopt File: $CARSPATH/menuopt/acctspay/screens/ckgrp

Parameters Passed:

- None

Fiscal Management: Purchasing/AP: Check Writing menu

Select Direct to State Grouping Sheets

Accesses: Program: *ckslct*

Menuopt File: $CARSPATH/menuopt/acctspay/programs/cksl.t

Parameters Passed:

- Check group
- Subsidiary (if ENABLE_MULTI_AP_SUBS = Y)
- Document type (default is PO)
- -t (if ENABLE_GRP_SHEETS = Y; required pay terms; default is STATE)

Fiscal Management: Purchasing/AP: Check Writing menu

Select Checks

Accesses: Program: *ckslct*

Menuopt File: $CARSPATH/menuopt/acctspay/programs/cksl.d

Parameters Passed:

- Document code (default is *PO*)

Fiscal Management: Purchasing/AP: Check Writing menu

Print Checks and Group Sheets

Accesses: Program: *fps*

Menuopt File: $CARSPATH/menuopt/utilities/programs/fps.ck

Parameters Passed: None

Fiscal Management: Purchasing/AP: Check Writing menu

Post Checks

Accesses: Program: *ckpost*

Menuopt File: $CARSPATH/menuopt/acctspay/programs/apckpost

Parameters Passed:

- -d (Purchase order document reference code; default is defined in the *macro DOC_PO_DEF*)

Fiscal Management: Purchasing/AP: Check Writing menu

Select Grouping Sheets

Accesses: Program: grpslct

Menuopt File: $CARSPATH/menuopt/acctspay/programs/grpsl

Parameters Passed: None

Note: Available if ENABLE_GRP_SHEETS = Y)

Fiscal Management: Purchasing/AP: Check Writing menu

Enter Manual Check

Accesses: Program: *voucher*

Menuopt File: $CARSPATH/menuopt/accounting/programs/vch.QC

Parameters Passed:

- -v (Journal code; default is *QC*)

Fiscal Management: Purchasing/AP: Check Writing menu

Check Register

Accesses: ACE report: jrnldocreg

Menuopt File: $CARSPATH/menuopt/accounting/reports/jrnldoc.ap

Parameters Passed:

- *Journal type*
- *Beginning journal number*
- *Ending journal number*

Fiscal Management: Purchasing/AP: Check Writing menu

Grouping Sheet Reconciliation

Accesses: ACE report: grprecon

Menuopt File: $CARSPATH/menuopt/accounting/reports/grprecon

Parameters Passed:

- *Beginning date*
- *Ending date*

Fiscal Management: Purchasing/AP: Check Writing menu

G/L Journal Reports

Accesses: Csh script: jrnlgl

Menuopt File: $CARSPATH/menuopt/accounting/scripts/jrnlgl.AP

Parameters Passed:

- -f (formtype; default is *standard*)

Fiscal Management: Purchasing/AP: Check Writing menu

S/L Journal Reports

Accesses: Csh script: jrnlsl

Menuopt File: $CARSPATH/menuopt/accounting/scripts/jrnlsl.AP

Parameters Passed:

- -f (formtype; default is *standard*)

Fiscal Management: Purchasing/AP: Check Writing menu

Void Checks

Accesses: Program: *docvoid*

Menuopt File: $CARSPATH/menuopt/accounting/programs/docv.CK

Parameters Passed:

- None

Fiscal Management: Purchasing/AP: Check Writing: Problem Solving menu

Document Table

Accesses: PERFORM screen: *tapdoc*

Menuopt File: $CARSPATH/menuopt/common/screens/tapdoc

Parameters Passed:

- None

Fiscal Management: Purchasing/AP: Check Writing: Problem Solving menu

Prepare for Check Abort

Accesses: Csh script: *remtrk.scp*

Menuopt File: $CARSPATH/menuopt/acctspay/scripts/remtrk

Parameters Passed:

- None

Fiscal Management: Purchasing/AP: Check Writing: Problem Solving menu

Check Abort

Accesses: Program: *ckabort*

Menuopt File: $CARSPATH/menuopt/acctspay/programs/ckab

Parameters Passed:

- None

Fiscal Management: Purchasing/AP: Check Writing: Problem Solving menu

Prepare for FPS Restart

Accesses: Csh script: restform

Menuopt File: $CARSPATH/menuopt/acctspay/scripts/restform

Parameters Passed:

- None

Fiscal Management: Purchasing/AP: Check Writing: Problem Solving menu

Interrupted Posting

Accesses: Csh script: intpost

Menuopt File: $CARSPATH/menuopt/acctspay/scripts/intpost

Parameters Passed:

- None

Fiscal Management: Purchasing/AP: Check Writing: Problem Solving menu

Terminate Check Journal

Accesses: Csh script: termpost

Menuopt File: $CARSPATH/menuopt/acctspay/scripts/termpost

Parameters Passed:

- None

Fiscal Management: Purchasing/AP: Check Writing: Problem Solving menu

Interrupted Grouping Sheet Posting

Accesses: Csh script: intgrp

Menuopt File: $CARSPATH/menuopt/acctspay/scripts/intgrp

Parameters Passed:

- *Journal type*
- *Journal number*

Note: Available if ENABLE_GRP_SHEETS = Y.

Fiscal Management: Purchasing/AP: Check Writing: Problem Solving menu

Terminate Grouping Sheet Journal

Accesses: Csh script: termgrp

Menuopt File: $CARSPATH/menuopt/acctspay/scripts/termgrp

Parameters Passed:

- *Grouping sheet history number*

Note: Available if ENABLE_GRP_SHEETS = Y.

Fiscal Management: Purchasing/AP: Check Writing: Problem Solving menu

Terminate Direct to State Grouping Sheets

Accesses: $CARSPATH/menuopt/acctspay/scripts/termpstg

Menuopt File: $CARSPATH/menuopt/acctspay/scripts/termgrp

Parameters Passed:

- *Grouping sheet history number*

Note: Available if ENABLE_GRP_SHEETS = Y.

Fiscal Management: Purchasing/AP: Check Writing: Problem Solving menu

Prepare for Grouping Sheet FPS Restart

Accesses: Csh script: restgrp

Menuopt File: $CARSPATH/menuopt/acctspay/scripts/restgrp

Parameters Passed:

- *Grouping sheet history number*

Note: Available if ENABLE_GRP_SHEETS = Y.

Fiscal Management: Purchasing/AP: Check Writing: Problem Solving menu

Grouping Sheet Tables

Accesses: PERFORM screen: grphist

Menuopt File: $CARSPATH/menuopt/acctspay/screens/grphist

Parameters Passed:

- None

Note: Available if ENABLE_GRP_SHEETS = Y.

Fiscal Management: Accounts Payable/Receiving Reports: Reports (A-M) menu

Check Register

Accesses: ACE report: jrnldocreg

Menuopt File: $CARSPATH/menuopt/accounting/reports/jrnldoc.ap

Parameters Passed:

- *Journal type*
- *Beginning journal number*
- *Ending journal number*

Fiscal Management: Accounts Payable/Receiving Reports: Reports (A-M) menu

Credit Memo Activity

Accesses: ACE report: credmem

Menuopt File: $CARSPATH/menuopt/purchase/reports/credmem

Parameters Passed:

- *Beginning date*
- *Ending date*

Fiscal Management: Accounts Payable/Receiving Reports: Reports (A-M) menu

Discounts Missed

Accesses: ACE report: discmissed

Menuopt File: $CARSPATH/menuopt/acctspay/reports/discmissed

Parameters Passed:

- *Subsidiary (if ENABLE_MULTI_AP_SUBS = Y)*
- *Beginning date*
- *Ending date*

Fiscal Management: Accounts Payable/Receiving Reports: Reports (A-M) menu

Discounts Taken

Accesses: ACE report: disctaken

Menuopt File: $CARSPATH/menuopt/acctspay/reports/disctaken

Parameters Passed:

- *Subsidiary (if ENABLE_MULTI_AP_SUBS = Y)*
- *Beginning date*
- *Ending date*

Fiscal Management: Accounts Payable/Receiving Reports: Reports (A-M) menu

Document Register

Accesses: ACE report: docreg

Menuopt File: $CARSPATH/menuopt/accounting/reports/docreg.ap

Parameters Passed:

- *Document type*
- *Beginning document number*
- *Ending document number*

Fiscal Management: Accounts Payable/Receiving Reports: Reports (A-M) menu

Invoice Activity

Accesses: ACE report: invactiv

Menuopt File: $CARSPATH/menuopt/purchase/reports/invactiv

Parameters Passed:

- *Beginning date*
- *Ending date*

Fiscal Management: Accounts Payable/Receiving Reports: Reports (A-M) menu

Invoice Aging

Accesses: ACE report: invaging

Menuopt File: $CARSPATH/menuopt/acctspay/reports/invaging

Parameters Passed:

- *Subsidiary (if ENABLE_MULTI_AP_SUBS = Y)*
- *Date*
- *Ending day for first column (and beginning day for second column)*
- *Ending day for second column (and beginning day for third column)*
- *Ending day for third column*

Fiscal Management: Accounts Payable/Receiving Reports: Reports (A-M) menu

Invoice Forecasting

Accesses: ACE report: payfore

Menuopt File: $CARSPATH/menuopt/acctspay/reports/payfore

Parameters Passed:

- *Subsidiary (if ENABLE_MULTI_AP_SUBS = Y)*
- *Date*
- *Ending day for first column (and beginning day for second column)*
- *Ending day for second column (and beginning day for third column)*
- *Ending day for third column*

Fiscal Management: Accounts Payable/Receiving Reports: Reports (A-M) menu

Invoices Unsubmitted

Accesses: ACE report: invunsub

Menuopt File: $CARSPATH/menuopt/purchase/reports/invunsub

Parameters Passed:

- None

Fiscal Management: Accounts Payable/Receiving Reports: Reports (A-M) menu

Invoices with No Checks

Accesses: ACE report: invnochk

Menuopt File: $CARSPATH/menuopt/purchase/reports/invnochk

Parameters Passed:

- *Subsidiary (if ENABLE_MULTI_AP_SUBS = Y)*

Fiscal Management: Accounts Payable/Receiving Reports: Reports (N-Z) menu

Open Encumbrances

Accesses: ACE report: openpoenc

Menuopt File: $CARSPATH/menuopt/purchasing/reports/openpoenc

Parameters Passed:

- None

Fiscal Management: Accounts Payable/Receiving Reports: Reports (N-Z) menu

Invoice Receipt Verification

Accesses: ACE report: invrpct

Menuopt File: $CARSPATH/menuopt/purchase/reports/invrcpt

Parameters Passed:

- *Beginning date*
- *Ending date*

Fiscal Management: Accounts Payable/Receiving Reports: Reports (N-Z) menu

PO to Receiving Report

Accesses: ACE report: potrc

Menuopt File: $CARSPATH/menuopt/requisition/reports/potrc

Parameters Passed:

- PO form
- PO number

Fiscal Management: Accounts Payable/Receiving Reports: Reports (N-Z) menu

Purchase Order Aging

Accesses: ACE report: poaging

Menuopt File: $CARSPATH/menuopt/acctspay/reports/poaging

Parameters Passed:

- *Subsidiary (if ENABLE_MULTI_AP_SUBS = Y)*
- *Date*
- *Ending day for first column (and beginning day for second column)*
- *Ending day for second column (and beginning day for third column)*
- *Ending day for third column*

Fiscal Management: Accounts Payable/Receiving Reports: Reports (N-Z) menu

Received Activity

Accesses: ACE report: recactiv

Menuopt File: $CARSPATH/menuopt/purchase/reports/recactiv

Parameters Passed:
- *Beginning date*
- *Ending date*

Fiscal Management: Accounts Payable/Receiving Reports: Reports (N-Z) menu

Received Single Item

Accesses: ACE report: recone

Menuopt File: $CARSPATH/menuopt/purchase/reports/recone

Parameters Passed:
- *PO form*
- *PO number*

Fiscal Management: Accounts Payable/Receiving Reports: Reports (N-Z) menu

Requisition to PO Report

Accesses: ACE report: rectpo

Menuopt File: $CARSPATH/menuopt/requisition/reports/reqtpo

Parameters Passed:
- *Requisition form*
- *Requisition number*

Fiscal Management: Accounts Payable/Receiving Reports: Reports (N-Z) menu

Returned Activity

Accesses: ACE report: retactiv

Menuopt File: $CARSPATH/menuopt/purchase/reports/retactiv

Parameters Passed:
- *Beginning date*
- *Ending date*

Fiscal Management: Accounts Payable/Receiving Reports: Reports (N-Z) menu

S/L Account Balances

Accesses: ACE report: subbalance

Menuopt File: $CARSPATH/menuopt/accounting/reports/subbal.AP

Parameters Passed:

- *Subsidiary (if ENABLE_MULTI_AP_SUBS = Y)*
- *Date*
- *Debit balances*
- *Credit balances*
- *Zero balances*
- *Include Actual Amounts flag*
- *Include Encumbered Amounts flag*
- *Include Additional Information flag*
- *Subprogram (if enabled)*

Fiscal Management: Accounts Payable/Receiving Reports: Reports (N-Z) menu

S/L Cash Entries

Accesses: ACE report: subentcash

Menuopt File: $CARSPATH/menuopt/accounting/reports/subec.AP

Parameters Passed:

- *Subsidiary (if ENABLE_MULTI_AP_SUBS = Y)*
- *ID*
- *Beginning date*
- *Ending date*
- *Entry type*
- *Amount type*
- *Comparison amount*
- *Summary*

Fiscal Management: Accounts Payable/Receiving Reports: Reports (N-Z) menu

Subsidiary History

Accesses: ACE report: subtrhist

Menuopt File: $CARSPATH/menuopt/accounting/reports/subtrh.AP

Parameters Passed:

- *Subsidiary (if ENABLE_MULTI_AP_SUBS = Y)*
- *ID*
- *Beginning date*
- *Ending date*

Fiscal Management: Accounts Payable/Receiving Reports: Reports (N-Z) menu

Vendor Listing

Accesses: ACE report: vndlst

Menuopt File: $CARSPATH/menuopt/acctspay/reports/vndlst

Parameters Passed:

- None

Fiscal Management: Accounts Payable/Receiving Reports: Reports (N-Z) menu

Vendor Spending Report

Accesses: ACE report: vndspend

Menuopt File: $CARSPATH/menuopt/purchase/reports/vndspend

Parameters Passed:

- *Beginning date*
- *Ending date*
- *ID*

Fiscal Management: Purchasing/AP: Purchase Order Audit menu

Audit All PO's

Accesses: Program: *purchaudit*

Menuopt File: $CARSPATH/menuopt/purchasing/programs/puau

Parameters Passed:

- None

Fiscal Management: Purchasing/AP: Purchase Order Audit menu

Audit PO Number(s)

Accesses: Program: *purchaudit*

Menuopt File: $CARSPATH/menuopt/purchasing/programs/puau.cbe

Parameters Passed:

- None

Fiscal Management: Purchasing/AP: Purchase Order Audit menu

Audit All PO's/Update

Accesses: Program: *purchaudit*

Menuopt File: $CARSPATH/menuopt/purchasing/programs/puau.u

Parameters Passed:

- -u (Update option; default is *update*)

Fiscal Management: Purchasing/AP: Purchase Order Audit menu

Audit PO Number(s)/Update

Accesses: Program: *purchaudit*

Menuopt File: $CARSPATH/menuopt/purchasing/programs/puau.cbeu

Parameters Passed:

- -u (Update option: default is *update*)

Fiscal Management: Purchasing/AP: 1099 menu

Initialize Data

Accesses: Program: *f1099bld*

Menuopt File: $CARSPATH/menuopt/acctspay/programs/99bd

Parameters Passed: None

Fiscal Management: Purchasing/AP: 1099 menu

Edit Data

Accesses: PERFORM screen: *f1099*

Menuopt File: $CARSPATH/menuopt/acctspay/screens/f1099

Parameters Passed: None

Fiscal Management: Purchasing/AP: 1099 menu

Update 1099 Field

Accesses: PERFORM screen: *upd1099*

Menuopt File: $CARSPATH/menuopt/acctspay/screens/upd1099

Parameters Passed: None

Fiscal Management: Purchasing/AP: 1099 menu

Prepare Forms

Accesses: Program: *1099form*

Menuopt File: $CARSPATH/menuopt/acctspay/programs/99fm

Parameters Passed: None

Fiscal Management: Purchasing/AP: 1099 menu

Prepare Individual Forms

Accesses: Program: *1099form*

Menuopt File: $CARSPATH/menuopt/acctspay/programs/99fm.i

Parameters Passed: None

Fiscal Management: Purchasing/AP: 1099 menu

Print Forms

Accesses: Program: *fps*

Menuopt File: $CARSPATH/menuopt/utilities/programs/fps.1099

Parameters Passed: None

Fiscal Management: Purchasing/AP: 1099 menu

Prepare Tape

Accesses: Program: *f1099tape*

Menuopt File: $CARSPATH/menuopt/acctspay/programs/99ta

Parameters Passed: None

Fiscal Management: Purchasing/AP: 1099 menu

Write Tape

Accesses: Program: *f1099tp*

Menuopt File: $CARSPATH/menuopt/acctspay/programs/99tp

Parameters Passed: None

Fiscal Management: Purchasing/AP: 1099 menu

Verify Tape

Accesses: Program: *taperead*

Menuopt File: $CARSPATH/menuopt/utilities/programs/tprd

Parameters Passed: None

Fiscal Management: Purchasing/AP: 1099 menu

Prepare Diskette File

Accesses: Program: *f1099tape*

Menuopt File: $CARSPATH/menuopt/acctspay/programs/99dsk

Parameters Passed:

- *Name*
- *Debugging level*

Fiscal Management: Purchasing/AP: 1099-R menu

Initialize Data

Accesses: Program: *r1099bld*

Menuopt File: $CARSPATH/menuopt/acctspay/programs/r1099bld

Parameters Passed:

- None

Fiscal Management: Purchasing/AP: 1099-R menu

Edit 1099-R Data

Accesses: PERFORM screen: *r1099*

Menuopt File: $CARSPATH/menuopt/acctspay/screens/r1099

Parameters Passed:

- None

Fiscal Management: Purchasing/AP: 1099-R menu

Update 1099 Field

Accesses: PERFORM screen: *upd1099*

Menuopt File: $CARSPATH/menuopt/acctspay/screens/upd1099

Parameters Passed:

- None

Fiscal Management: Purchasing/AP: 1099-R menu

Prepare 1099-R Forms

Accesses: Program: *r1099form*

Menuopt File: $CARSPATH/menuopt/acctspay/programs/r1099fm

Parameters Passed:

- None

Fiscal Management: Purchasing/AP: 1099-R menu

Prepare Individual Forms

Accesses: Program: *r1099form*

Menuopt File: $CARSPATH/menuopt/acctspay/programs/r1099fm.i

Parameters Passed:

- None

Fiscal Management: Purchasing/AP: 1099-R menu

Print 1099-R Forms

Accesses: Program: *fps*

Menuopt File: $CARSPATH/menuopt/utilities/programs/fps.1099r

Parameters Passed:

- None

Fiscal Management: Purchasing/AP: Refund: Check Writing menu

Note: See *Fiscal Management: Accounts Payable Refund: Check Writing menu* in this section.

Fiscal Management: Purchasing/AP: Refund: Check Writing: Problem Solving menu

Note: See *Purchasing/AP: Check Writing: Problem Solving menu* in this section.

Fiscal Management: Purchase Order Approval menu

Approve Purchase Orders

Accesses: Program: *approve*

Menuopt File: $CARSPATH/menuopt/purchasing/programs/puap.cd

Parameters Passed:

- *Print flag*
- *Output device*
- *Date*

Note: This option is available if ENABLE_FEAT_PO_APPROVAL = Y.

Fiscal Management: Purchase Order Approval menu

Accounting Query

Accesses: Program: *acquery*

Menuopt File: $CARSPATH/menuopt/accounting/programs/acqu.p

Parameters Passed:

- *Output device*

Note: This option is available if ENABLE_FEAT_PO_APPROVAL = Y.

Fiscal Management: Purchase Order Approval menu

Budget Review

Accesses: Program: *bgtreview*

Menuopt File: $CARSPATH/menuopt/accounting/programs/bgtr

Parameters Passed:

- *Fiscal year*
- *Output device*

Note: This option is available if ENABLE_FEAT_PO_APPROVAL = Y.

Financial PERFORM (Table Maintenance) Screens

Introduction

The financial products use PERFORM screens for displaying maintenance tables and some records. You can access the screen files in the following directory paths:

- $CARSPATH/modules/accounting/screens
- $CARSPATH/modules/acctspay/screens
- $CARSPATH/modules/budget/screens
- $CARSPATH/modules/fixassets/screens
- $CARSPATH/modules/purchasing/screens

PERFORM Screens

The following lists the PERFORM screens used in the financial products:

Note: In the following list, descriptions of PERFORM screens include:

- Purpose of the screen
- Tables used in the screen
- Master/detail relationships, if applicable

1099 Information screen

Purpose: Enables you to view, add, or update 1099 information for vendors who deal with your institution.

Menu Access: Fiscal Management: Purchasing/AP: 1099 Menu: Edit Data

Screen File: $CARSPATH/modules/acctspay/screens/f1099

*Tables/Records Used:*f1099_rec, f1099_table, id_rec

1099R Information screen

Purpose: Enables you to view, add, or update 1099R information that your institution maintains.

Menu Access: Fiscal Management: Purchasing/AP: 1099-R menu: Edit 1099-R Data

Screen File: $CARSPATH/modules/accounting/screens/r1099

*Tables/Records Used:*id_rec, r1099_rec

Amount Type Table screen

Purpose: Enables you to view, add and update the types of entries that your institution uses.

Menu Access: Fiscal Management: Budgeting: Table Maintenance: Other Financial (A-Z)

Screen File: $CARSPATH/modules/accounting/screens/tatype

*Tables/Records Used:*atype_table

Budget Adjustment Table screen

Purpose: Enables you to automatically add Budget Amount records.

Menu Access: Fiscal Management: Budgeting: Table Maintenance: Budgeting (A-Z)

Screen File: $CARSPATH/modules/budgeting/screens/bgtacct

*Tables/Records Used:*bgtacct_rec, func_table, fund_table, obj_table, subfund_table

Budget Calendar screen

Purpose: Enables you to view, add, or update the calendar that your institution uses for budgetary purposes.

Menu Access: Fiscal Management: Budgeting: Table Maintenance: Budgeting (A-Z)

Screen File: $CARSPATH/modules/accounting/screens/bgtcal

*Tables/Records Used:*bgtcal_rec

Budget Distribution Table screen

Purpose: Enables you to view, add, or update the percent of an annual budget that should be allocated to each month. The *bgtalloc* program uses these percentages to allocate the correct portion of a yearly budget to each month in the bgtamt_rec.

Menu Access: Fiscal Management: Budgeting: Table Maintenance: Budgeting (A-Z)

Screen File: $CARSPATH/modules/accounting/screens/bgtdist

*Tables/Records Used:*atype_table, bgtdist_table, func_table, fund_table, obj_table, subfund_table

Budget Parameter screen

Purpose: Enables you to view, add, or update parameters that your institution uses to process budgets through *bgtalloc* and *bgtbasis*.

Menu Access: Fiscal Management: Budgeting: Table Maintenance: Budgeting (A-Z)

Screen File: $CARSPATH/modules/accounting/screens/pbgt

*Tables/Records Used:*fund_table, pbgt_rec

Cash Transaction Type Table screen

Purpose: Enables you to view, add, and update the types of cash transactions that your institution uses.

Menu Access: Fiscal Management: Accounting: Table Maintenance: Financial (A-C)

Screen File: $CARSPATH/modules/accounting/screens/tcashent

*Tables/Records Used:*cashent_table, ent_table, func_table, fund_table, obj_table, subb_table, subfund_table, subs_table, subt_table

Check Group Record screen

Purpose: Enables you to view, add, or update information about the check groups that your institution must print.

Menu Access: Fiscal Management: Purchasing/AP: Check Writing menu: Select Check Group

Screen File: $CARSPATH/modules/acctspay/screens/ckgrp

*Tables/Records Used:*ckgrp_rec

Check Group Record screen

Purpose: Enables you to view, add, or update information about the check groups that your institution must print.

Menu Access: Fiscal Management: Purchasing/AP: Refund: Check Writing menu: Select Check Group

Screen File: $CARSPATH/modules/accounting/screens/ckgrpsr

*Tables/Records Used:*ckgrp_rec

Defined Account Record screen

Purpose: Enables you to view, add, and update the valid combinations of fund, function, object and subfund that your institution uses.

Menu Access: Fiscal Management: Budgeting: Table Maintenance: Other Financial (A-Z)

Screen File: $CARSPATH/modules/accounting/screens/gldefine

*Tables/Records Used:*func_table, fund_table, gld_rec, obj_table, subfund_table

Document Table screen

Purpose: Enables you to view, add, and update the information about the document types that your institution uses.

Menu Access: Either of the following:

- Fiscal Management: Purchasing/AP: Check Writing: Problem Solving: Document Table
- Fiscal Management: Purchasing/AP: Refund: Check Writing: Problem Solving: Document Table

Screen File: $CARSPATH/modules/common/screens/tapdoc

*Tables/Records Used:*doc_table

Enter Assets screen

Purpose: Enables you to view, add, and update information about your institution's fixed assets.

Menu Access: Fiscal Management: Fixed Assets Main menu: Enter Assets

Screen File: $CARSPATH/modules/fixassets/screens/fix

*Tables/Records Used:*fix_record

Financial Statement Table screen

Purpose: Enables you to view, add, or update the reporting structure that your institution uses.

Menu Access: Fiscal Management: Budgeting: Table Maintenance: Other Financial (A-Z)

Screen File: $CARSPATH/modules/accounting/screens/tfs

*Tables/Records Used:*fs_table

Fiscal Calendar Record screen

Purpose: Enables you to view, add, or update the valid dates for posting transactions at your institution.

Menu Access: Fiscal Management: Budgeting: Table Maintenance: Other Financial (A-Z)

Screen File: $CARSPATH/modules/accounting/screens/fiscalcal

*Tables/Records Used:*fscl_cal_rec, subs_table

General Ledger screen

Purpose: Enables you to view, add, or update the components of the account numbers that your institution uses.

Menu Access: Fiscal Management: Budgeting: Table Maintenance: Other Financial (A-Z)

Screen File: $CARSPATH/modules/accounting/screens/glfieldrpt

*Tables/Records Used:*func_table, fund_table, obj_table, subfund_table

Payment Plan Table screen

Purpose: Enables you to view, add, or update the types of payment plans that students can use at your institution.

Menu Access: Fiscal Management: Accounting: Table Maintenance: Financial (J-R)

Screen File: $CARSPATH/modules/accounting/screens/tpayplan

*Tables/Records Used:*payplan_table

Payment Terms Table screen

Purpose: Enables you to view, add, or update the types of payment terms that students can use at your institution.

Menu Access: Fiscal Management: Accounting: Table Maintenance: Financial (J-R)

Screen File: $CARSPATH/modules/accounting/screens/tpayterm

*Tables/Records Used:*defpay_table, pay_term_table

Subsidiary Balance Record screen

Purpose: Enables you to view, add, or update 1099 information that appears in Subsidiary Balance records.

Menu Access: Either of the following:

- Fiscal Management: Purchasing/AP: 1099 menu: Update 1099 Field
- Fiscal Management: Purchasing/AP: 1099-R menu: Update 1099 Field

Screen File: $CARSPATH/modules/acctspay/screens/upd1099

*Tables/Records Used:*id_rec, subb_rec, subs_table

Financial SQL Scripts

The *termacct* Script

The financial product Budgeting accesses an SQL script *termacct*. The script enables users to terminate accounts for an upcoming fiscal period, and is located in the following directory path:

$CARSPATH/modules/accounting/informers

The *termacct* script updates the General Ledger Account record (gla_rec).

Financial Csh Scripts

Introduction

The financial programs contain Csh scripts to automate the processing of information. Csh scripts are UNIX-based program statements that can execute a series of SQL scripts or reports. The financial Csh scripts are located in the financial directory paths under the *scripts* subdirectory.

Csh Scripts

The following list associates a financial menu option with the corresponding Csh script and provides a description of the script.

Note: In the following list, descriptions of Csh scripts include:

- Purpose of the script
- A list of SQL statements used, if applicable
- A list of ACE reports used, if applicable

Daily Station Reports (daycash)

Produces a daily cash deposit report and daily summary of cash by total code. These reports reside in the following file locations:

- $CARSPATH/install/arc/accounting/deposit
- $CARSPATH/install/arc/accounting/subtcash

Access: $CARSPATH/modules/accounting/scripts/daycash

G/L Journal Reports (jrnlgl)

Executes and prints the General Ledger Journal reports, using the following ACE reports:

- jrnlacct
- jrnlent

Access: $CARSPATH/modules/accounting/scripts/jrnlgl

Global Adjustments (addbgtamt)

Adds or updates Budget Amount records for global adjustments. Also copies budget information.

Access: $CARSPATH/modules/budget/scripts/addbgtamt

Interrupted Grouping Sheets (intgrp)

Corrects the check posting process for grouping sheets when a system failure occurs during the posting of a check journal.

Access: $CARSPATH/modules/acctspay/scripts/intgrp

Interrupted Posting (intpost)
Corrects the check posting process when a system failure occurs during the posting of a check journal.

Access: $CARSPATH/modules/acctspay/scripts/intpost

Last Accounts Payable Document (lastap)
Retrieves the last accounts payable document number used from the Document table and prints the next document number to be used.

Access: $CARSPATH/modules/acctspay/scripts/lastap

Last Refund Document (lastrf)
Retrieves the last refund document number used from the Document table and prints the next document number to be used.

Access: $CARSPATH/modules/acctspay/scripts/lastrf

Prepare for Check Abort (rmtrk)
Removes the tracking file for a specified check group. Use this option when you must abort a check run after printing has begun.

Access: $CARSPATH/modules/acctspay/scripts/rmtrk

Prepare for FPS Restart (restform)
Restores form files to an in-progress printing state.

Access: $CARSPATH/modules/acctspay/scripts/restform

Restore Grouping Sheet Printing (restgrp)
Restores form files to an in-progress printing state.

Access: $CARSPATH/modules/acctspay/scripts/restgrp

Restore Grouping Sheet Writing (termgrp)
Restores the grouping sheet writing process to a state in which grouping sheets can be reposted.

Access: $CARSPATH/modules/acctspay/scripts/termgrp

S/L Journal Reports (jrnlsl)
Executes and prints the Subsidiary Journal reports using the following ACE reports:

- jrnlslacct
- jrnlslent

Access: $CARSPATH/modules/accounting/scripts/jrnlsl

Terminate Check Journal (termpost)
Restores the check writing process to a state in which the checks can be reposted.

Access: $CARSPATH/modules/acctspay/scripts/termpost

Terminate Grouping Sheet Posting (termpstg)
Restores the state grouping sheet process to a state in which the grouping sheets can be reselected.

Access: $CARSPATH/modules/acctspay/scripts/termpstg

Financial ACE Reports

Introduction

The CX contains ACE reports for easy reporting of financial database information.

ACE Reports from the Fiscal Management: Accounting Main Menu

The following lists the ACE reports provided with the financial products and accessible from the Accounting Main menu.. For more information about other reports on the Accounting Main menu, see *General Ledger Technical Manual*.

1099 Type Table Report
Lists all the available form 1099 types.

Report File: $CARSPATH/modules/acctspay/reports/t1099

Payment Form Table Report
Displays the values in the payform_table.

Report File: $CARSPATH/modules/accounting/reports/tpayform

Payment Term Table Report
Displays the values in the payterm_table.

Report File: $CARSPATH/modules/accounting/reports/tpayterm

Vendor Quality Table Report
Lists the contents of the venqual_table.

Report File: $CARSPATH/modules/purchasing/reports/tvenqual

Vendor Type Table Report
Lists the contents of the ventype_table.

Report File: $CARSPATH/modules/purchasing/reports/tventype

ACE Reports from the Fiscal Management: Fixed Assets Main Menu

The following lists the ACE reports provided with the financial products and accessible from the Fixed Assets Main menu.

Asset Acquisitions Report
Prints information about assets acquired within the specified date range. The report sorts the information by control account number.

Report File: $CARSPATH/modules/fixassets/reports/acquire

Asset Disposals Report
Prints information about assets disposed within the specified date range.

Report File: $CARSPATH/modules/fixassets/reports/dispose

Asset List by Location
Prints list of all fixed assets by location.

Report File: $CARSPATH/modules/fixassets/reports/fixloc

Asset List by Responsible Person
Prints list of all fixed assets by ID number of the person responsible for the asset.

Report File: $CARSPATH/modules/fixassets/reports/fixpers

Asset List by Type
Prints list of all fixed assets by asset type.

Report File: $CARSPATH/modules/fixassets/reports/fixtype

Asset List for Insurance by Type
Prints list of all fixed assets for insurance purposes.

Report File: $CARSPATH/modules/fixassets/reports/insure

Asset Values: Summarized Assets
Prints a list of all summarized assets with associated itemized assets.

Report File: $CARSPATH/modules/fixassets/reports/sumlist

Book Value of Non-Summarized Assets
Prints dollar information about fixed assets sorted by capitalization account. The report does not include assets that have been disposed.

Report File: $CARSPATH/modules/fixassets/reports/assetacct

Depreciation: Non-Summarized Assets Only
Prints depreciation information about fixed assets.

Report File: $CARSPATH/modules/fixassets/reports/depreciate

Depreciation: Non-Summarized Assets Only
Prints depreciation information about fixed assets.

Report File: $CARSPATH/modules/fixassets/reports/deprfy

Depreciation: Summarized Assets Only
Prints depreciation for summarized assets through a specified date.

Report File: $CARSPATH/modules/fixassets/reports/sumdepr

Fixed Asset Type Table Report
Lists the attributes of fix_table.

Report File: $CARSPATH/modules/fixassets/reports/tfix

Specific Asset List
Prints selected fixed assets. Users pass selection and sorting criteria to the report through parameters. Selection criteria can be any column in the fix_rec.

Report File: $CARSPATH/modules/fixassets/others/assetsrpt

ACE Reports from the Fiscal Management: Budgeting Main Menu

The following lists the ACE reports provided with the financial products and accessible from the Budgeting Main menu.

Budget Account Detail Report
Prints the amounts from the bgtamt_rec by range of cost center for the fiscal periods entered as parameters (e.g., JULY or AUG). The report sorts by cost center.

Report File: $CARSPATH/modules/ budget/others/bgtcntrdtl

Budget Account Summary
Prints the amounts from the bgtamt_rec for the fiscal periods entered as parameters (e.g., JULY or AUG). The summary pages sort by account with one line for each account.

Report File: $CARSPATH/modules/budget/others/bgtacctsum

Budget Adjustment Table Report
Displays the values in the bgtacct_table.

Report File: $CARSPATH/modules/budget/reports/bgtacct

Budget Balance Sheet
Budget Subfund by Object Summary
Budget Trial Balance
Prints the summarized amounts by account.

Report File: $CARSPATH/modules/budget/others/bgtprjasum

Budget Calendar Report
Displays the values in the bgtcal_rec.

Report File: $CARSPATH/modules/budget/reports/btcal

Budget Cost Center Detail Report
Prints the budget amounts for expenses (optionally for revenues) for the fiscal period by range of cost center. The report sorts by cost center.

Report File: $CARSPATH/modules/budget/others/bgtcntrprj

Budget Cost Center Summary Report
Prints the budgets summarized by cost center.

Report File: $CARSPATH/modules/budget/others/bgtcntrsum

Budget Detail Report
Prints the amounts from the bgtamt_rec for the fiscal periods entered as parameters (e.g., JULY or AUG). The Detail Page report section sorts by object with a page break for each object, displaying a single line for each function or subfund.

Report File: $CARSPATH/modules/ budget/others/bgtacctdtl

Budget Distribution Table Report
Displays the values in the bgtdist_table.

Report File: $CARSPATH/modules/budget/reports/tbgtdist

Budget Parameter Report
Displays the values in the pbgt_rec.

Report File: $CARSPATH/modules/budget/reports/pbgt

Budget Profit Center Report
Prints profit and loss budgets for each cost centers, displaying revenues and expenses.

Report File: $CARSPATH/modules/budget/others/bgtcntrprf

Budget Subfunds by Function Detail
Prints the actual expenditures (and revenues if desired), for the specified fiscal period (e.g., JULY or AUG) by range of cost center. The report sorts by cost center.

Report File: $CARSPATH/modules/budget/others/bgtprjcdtl

Budget Subfunds by Object Detail
Prints the different column amounts for accounts.

Report File: $CARSPATH/modules/budget/others/bgtprjadtl

Budget Subfunds by Object Summary
Prints the actual expenditures for the fiscal period (e.g., JULY or AUG), summarized by cost center.

Report File: $CARSPATH/modules/budget/others/bgtprjcsum

Budget Subfund Summary
Prints the actual expenditures for the fiscal period (e.g., JULY or AUG) summarized by cost center.

Report File: $CARSPATH/modules/budget/others/bgtprjsum

Cash Flow Analysis by Cost Center Report
Displays the changes in cash for the specified cost center for the specified account and month.

Report File: $CARSPATH/modules/budget/others/bgtacctmon

Cash Flow Analysis Report by Cost Center
Displays the changes in cash for the specified cost center for the specified cost center and month.

Report File: $CARSPATH/modules/budget/others/bgtcntrmon

Cash Flow Analysis Report by Cost Center
Displays the changes in cash for the specified cost center for the project, account, and month.

Report File: $CARSPATH/modules/budget/others/bgtprjamon

Cash Flow Analysis Report by Cost Center
Displays the changes in cash for the specified cost center for the specified project, center, and month.

Report File: $CARSPATH/modules/budget/others/bgtprjcmon

ACE Reports from the Fiscal Management: Cash Receipts Menu

The following lists the ACE reports provided with the financial products and accessible from the Cash Receipts menu.

Cash Transaction Type Table Report
Displays the values in the cashent_table.

Report File: $CARSPATH/modules/accounting/reports/tcashent

ACE Reports from the Fiscal Management: Purchasing/AP Main Menu

The following lists the ACE reports provided with the financial products and accessible from the Purchasing/AP Main menu.

Credit Memos Report
Lists accounts payable that have debit balances.

Report File: $CARSPATH/modules/acctspay/reports/paydebit

Discounts Taken Report
Lists purchasing discounts that the institution has taken.

Report File: $CARSPATH/modules/acctspay/reports/disctaken

Invoice Actual Aging Report
Produces a report of payables by age.

Report File: $CARSPATH/modules/acctspay/reports/invaging

Invoice Forecast Report
Produces a report that forecasts payable activity.

Report File: $CARSPATH/modules/acctspay/reports/payfore

Invoices by Purchase Order Report
Produces a list of the invoices that relate to purchase orders.

Report File: $CARSPATH/modules/acctspay/reports/poinv

Missed Discounts Report
Lists purchasing discounts that the institution has missed.

Report File: $CARSPATH/modules/acctspay/reports/discmissed

Open Purchase Order Encumbrance Report
Lists open purchase orders with encumbrances and invoices posted.

Report File: $CARSPATH/modules/purchasing/reports/openpoenc

Purchase Order Approval Responsibility Report
Prints the purchase order approval restrictions for the list of IDs that the given ID is responsible for approving.

Report File: $CARSPATH/modules/purchasing/others/apprresp

Purchase Order Approval Restriction Report
Prints the purchase order approval restrictions for a give range of IDs. This report creates a hard copy of current approval table restrictions for a range of ID numbers.

Report File: $CARSPATH/modules/purchasing/others/restrict

Purchase Order Encumbrance Aging Report
Lists encumbrances by age.

Report File: $CARSPATH/modules/acctspay/reports/poaging

Purchase Order History Report
Prints the history of a range of purchase orders.

Report File: $CARSPATH/modules/purchasing/reports/pohist

Vendor Listing
Produces an alphabetical list of vendors that the institution uses.

Report File: $CARSPATH/modules/acctspay/reports/vndlst

SECTION 31 - CUSTOMIZING THE FINANCIAL PROCESSES

Overview

Introduction

This section provides you with procedures for setting and installing the features of the financial products. The following procedures are included:

- Assessing institution needs for module
- Reviewing data in tables and records
- Changing default field values in macros
- Changing table values

Basic Information

This section contains detailed procedures specific to the financial products. For information on performing basic procedures such as using the MAKE processor and reinstalling options, refer to the following resources:

- *Database Tools and Utilities* course notebook
- *Jenzabar CX Technical Manual*

Reviewing Data in Tables and Records

Introduction

After assessing features of the financial products and setting the appropriate enable macros, you must review the setup of CX tables and records.

Procedure

The following procedure provides the steps to review the values of CX tables and records.

1. For each table, review the codes supplied with the CX. Determine whether or not the codes meet the needs of your institution. Make updates as appropriate.
2. Review the institution's records converted from the previous financial system. Determine whether or not the records need to be updated to meet the needs of the CX reports. Make updates as appropriate.

Reviewing Data in Budgeting Tables and Records

Introduction

The application programs that are part of the financial budgeting product create data for the Budget Amount Record (bgtamt_rec) and the Budget Summarization Record (bgtsum_rec). You must set up the Budget Calendar record and Budget Parameter record before the budget programs, *bgtbasis* and *bgtalloc*, can be run. Some tables that impact other financial products can also require modification.

Implementation Order

When you implement the Financial Budgeting product, create or verify entries in the tables in the following order:

1. Amount Type Table (atype_table)
2. Budget Calendar record (bgtcal_rec)
3. Budget Adjustment Table (bgtacct_rec)
4. Budget Distribution Table (bgtdist_table)
5. Budget Parameter record (pbgt_rec)

Table Access

All users with appropriate permissions can access the budget tables from Budgeting: Table Maintenance. The Amount Type table, since it is primarily a General Ledger table, is accessible from the Accounting: Table Maintenance menu, and also from the Budgeting: Table Maintenance menu under the Other Financial Tables option.

Budget Calendar Record (bgtcal_rec)

Before you can run any of the budget programs, you must create entries for every combination of Fiscal Year and Amount Type that will be used by the Budgeting product. The bgtcal_rec captures this required information, and contains the following fields:

Fiscal Year
The fiscal year of the budget calendar (e.g., 9697).

Amount type
The type of general ledger amounts considered. Valid values include:

- ACT (actual)
- APP (approved)
- BGT (budgeted)
- PRJ (projected)
- REQ (requested)

Beginning date
The beginning date of the fiscal year.

Ending date
The ending date of the fiscal year.

Open date
The first date the user can make changes to the general ledger accounts linked to the fiscal year and amount type.

Close date
The last date the user can make changes to the general ledger accounts linked to the fiscal year and amount type. The use of the Open and Close dates permit users to prepare the budget before the start of the fiscal year.

Summaries
A Y/N flag indicating that you want to create bgtsum_recs while processing budgets. You must set this value to Y to enable rollups to occur in *bgtalloc*, but you can set the Summaries flag to N during data entry to cause the process to run faster.

Budget Parameter Record (pbgt_rec)

You must create an entry in this table before running *bgtalloc*. This record controls the contents of the columns of data that display when running *bgtalloc*. Set up one entry in this table for each column of data, up to a maximum of four. The pbgt_rec contains the following fields:

Code
The eight-character code designating the record and its related parameters. The same code must be passed to the Budget Allocation program when it is run.

Note: Many users enter the fiscal year and budget type as the code. For example, if the end user is working with the Fiscal Year 9697 Budget, the code used could be 9697BGT.

GL Fund
Fund with which this Budget Parameter record is associated.

Axis code
A flag indicating whether the default axis is on the function or object axis (i.e., will the display show all of the objects associated with a given department (function) or all of the departments (functions) that use the object).

Column Type
A code that controls the display for a column. Valid values include:

- $ (Displays the budget in whole dollars)
- + (Displays the dollar difference for two years)
- % (Displays the percentage difference for two years)
- (Causes no information to be displayed in the column)

Fiscal Year
The fiscal year for the data that appears in a column.

Amount Type
The type of budget to be displayed in a column. Valid values include:

- BGT (budgeted)
- ACT (actual)
- APP (approved)
- PRJ (projected
- REQ (requested)

Comparison Year
For column types of "+" or "%", the fiscal year to use for comparison.

Comparison Amount
For column types of "+" or "%", the amount type to use for comparison.

Note: You must designate a column for the fiscal year of the budget, the amount type of the budget, and "$" for the amount type. You can place this information in any column.

Budget Adjustment Table (bgtacct_rec)

This table works in conjunction with the Global Adjustments option. Together, they control how the original general ledger amounts change when they are moved to the new fiscal year and amount type for the account shown. If your institution intends to use this table, you must create an entry for each account that you want to adjust before you run the Global Adjustments option. It is not possible to mask accounts when creating entries for this table.

If the institution does not use this option, then the Global Adjustments option will not adjust the amounts from the General Ledger before *bgtalloc* modifies the values.

Note: The Budget Adjustment table screen contains a field labeled "Function". To ensure this table works correctly, do not enter data in this field.

Fiscal Year
The base fiscal year for this budget account record.

Fund
The base fund for this budget account record.

Account
The G/L account which will be updated when the program is run.

Percent
The percentage by which the indicated general ledger account should be changed. For example, an entry of 5.0 for a general ledger account means that the new budget will begin with the original amount from the general ledger plus 5%.

Flat Amount
The dollar amount by which the indicated account should be changed. As an additional help to the user, the screen displays what effect the proposed change would have on a base budget of $1000.00.

Note:

- To reduce the budgeted amount, enter negative numbers in the percentage field, the flat amount field, or both.
- You can use the percentage and flat amounts together. The program calculates the percentage change first, then adds or subtracts the flat amount.

Example: Assume the following values and table entries:

Budget 1000.00

Percentage 5 %

Flat Amount 100.00

The new budget figure is 1150.00 (1000 + (1000*.05) + 100)

Budget Distribution Table (bgtdist_table)

This table enables the user to enter the percentages that control how the yearly budget allocation is divided for monthly budgeting. If the institution uses this table, it should be set up before running *bgtalloc*. If the institution does not use this table, then the program divides the amount allocated over the entire fiscal year.

Budget type
The code for the budget amount type, which corresponds to the value in the bgtcal amt_type field. Valid values include:

- BGT (budgeted)
- ACT (actual)
- APP (approved)
- PRJ (projected
- REQ (requested)

Fund
The general ledger fund code for the G/L account.

Function
The general ledger function code for the G/L account.

Object
The general ledger object code for the G/L account.

Months
The individual fields that contain the percentage of the annual budget to allocate to the specified month. The total entered for all months should equal 100%.

Amount Type Table Setup for Financial Budgeting

The Amount Type table (atype_table) may require customization during General Ledger implementation. The *bgtalloc* and *bgtbasis* programs require the following entries in the atype_table. In each case, if the Allocation Allowed flag equals Y, *bgtalloc* enables users to add and/or update on particular amount type.

Amount type 1
ACT

Allocation Allowed flag 1
N

Description 1
Actual

Amount type 2
APP

Allocation Allowed flag 2
N

Description 2
Approved budget

Amount type3
BGT

Allocation Allowed flag 3
Y

Description 3
Budget

Amount type 4
REQ

Allocation Allowed flag 4
Y

Description 4
Requested budget

Amount type 5
PRJ

Allocation Allowed flag 5
Y

Description 5
Projected budget

Other Customization Issues in Financial Budgeting

Trimester Budgeting

Financial Budgeting offers the capability of trimester budget control. With this feature, you can subdivide the twelve-month year (excluding balance and adjustment “months”) into three separate periods. By default, each period is four months, but you can customize the period lengths as desired, as long as the three periods include the entire twelve-month year.

Also by default, the trimester names on reports are TRM1, TRM2, and TRM3, and TRM1 begins in the first period specified in your Fiscal Calendar record (fscl_cal_rec).

Macros for Trimester Budgeting

The following macros impact trimester budgeting:

ENABLE_TRM_BGT
: Implements the trimester budgeting feature. When enabled, this macro does not permit the user to override budgets for any period.

TRM1_NAME
TRM2_NAME
TRM3_NAME
: Define the names for each of the three trimester periods.

TRM1_END
TRM2_END
TRM3_END
: Define the last month in each trimester period, using values from the Fiscal Calendar record.

TRM_BGT_PRDS
: Defines the titles for budget reports that display trimester amounts.

Implementation Procedures for Trimester Budgeting

To use trimester budgeting:

1. Change the macros as required. Note that trimester budgeting is disabled within the standard CX product.
2. Reinstall the custom/financial macro file.
3. Reinstall the include/applic/acct file.
4. Reinstall all screens in src/Lib/libacct/SCR (or, if you prefer, you can reinstall only the *trans* and *review* screens).
5. Reinstall src/Lib/libacct.
6. Reinstall src/Lib/libgl.
7. Reinstall the following src directories in any order:
   - src/accounting/bgtreview
   - src/purchase/acctspay
   - src/purchase/massinv
   - src/purchase/purchase
   - src/purchasing/purch
   - src/requisition/requisition

Reviewing Data in Fixed Asset Tables and Records

Tables Used in Fixed Assets Processing

When you implement the Fixed Assets product, you must add entries to the following tables that already exist from your implementation of other CX modules.

- Chart of Accounts table
- Defined Account table
- Document table
- Entry Type table

These tables validate accounts, account code combinations, document types and entry types that your institution uses. Sample table values for the Entry Type table appear in this section.

In addition, you *must* complete the Fixed Asset Type table, a table you use only when you implement and use Fixed Assets.

Implementation Order

When you implement the Fixed Assets product, create or verify entries in the tables in the following order:

1. Chart of Accounts
2. Defined Account table
3. Document table
4. Entry Type table
5. Fixed Asset Type table

Updating the Entry Type Table for Fixed Assets Processing

The Entry Type table defines the valid entries that your institution uses. In particular, the table controls the number of transactions (debit lines and credit lines) each entry can have, what type of entries the institution can use, and the journals that can contain the entries. When you implement Fixed Assets, you must define entry types for the following types of transactions:

- Capitalization
- Depreciation
- Disposal of the assets

Entry Type Table Fields for Capitalization

Following is an example of the Entry Type Table PERFORM screen with the correct values for the capitalization entry type.

```
PERFORM:   Query  Next  Previous  View  Add  Update  Remove  Table  Screen  ...
Shows the next row in the Current List.       ** 1: ent_table table**
                              ENTRY TYPE TABLE

Entry Type Code..[CAPT] [Asset capitalization    ]  Amount Type...........[ACT]
Default Description.....[Asset Capitalization    ]  Itemize on Statement..[N]
Payment Form.....[  ]                               Maximum Transactions..[  2]

Ignore: Subsidiaries....[N]  Constraints.......[Y]
Debit or Credit to A/R..[ ]  A/P..............[ ]  Subsidiary Post.......[INV]

     Journal Allowed To Post Entry     |       Chart of Permission Codes
     ------------------------------    |       -------------------------
       AC - [Y]         JC - [N]       |            DR  CR       DR  CR
       AP - [N]         PC - [N]       |        B - 1   1    H - 0>  0>
       AR - [N]         PD - [N]       |        C - 0   1    I - 0   0>
       BG - [N]         PR - [N]       |        D - 1   0    J - 0>  0
       CH - [N]         PS - [N]       |        E - 1<  1<   K - 1>  1>
       CK - [N]         QC - [N]       |        F - 0   1<   L - 0   1>
       DA - [N]         SA - [N]       |        G - 1<  0    M - 1>  0
       FA - [N]         SB - [N]       |                     N - 0   0
                                       |        Cash..[E]    S/L...[Y]
```

Entry Type Table Fields for Depreciation

Following is an example of the Entry Type Table PERFORM screen with the correct values for the depreciation entry type.

```
PERFORM:   Query  Next  Previous  View  Add  Update  Remove  Table  Screen  ...
Shows the next row in the Current List.       ** 1: ent_table table**
                              ENTRY TYPE TABLE

Entry Type Code..[DEPR] [Depreciation            ]  Amount Type...........[ACT]
Default Description.....[Asset Depreciation      ]  Itemize on Statement..[N]
Payment Form.....[  ]                               Maximum Transactions..[ 10]

Ignore: Subsidiaries....[N]  Constraints.......[N]
Debit or Credit to A/R..[ ]  A/P..............[ ]  Subsidiary Post.......[INV]

     Journal Allowed To Post Entry     |       Chart of Permission Codes
     ------------------------------    |       -------------------------
       AC - [Y]         JC - [N]       |            DR  CR       DR  CR
       AP - [N]         PC - [Y]       |        B - 1   1    H - 0>  0>
       AR - [N]         PD - [N]       |        C - 0   1    I - 0   0>
       BG - [N]         PR - [N]       |        D - 1   0    J - 0>  0
       CH - [N]         PS - [N]       |        E - 1<  1<   K - 1>  1>
       CK - [N]         QC - [N]       |        F - 0   1<   L - 0   1>
       DA - [N]         SA - [N]       |        G - 1<  0    M - 1>  0
       FA - [N]         SB - [N]       |                     N - 0   0
                                       |        Cash..[E]    S/L...[Y]
```

Entry Type Table Fields for Disposals

Following is an example of the Entry Type Table PERFORM screen with the correct values for the disposal entry type.

```
PERFORM:   Query  Next  Previous  View  Add  Update  Remove  Table  Screen  ...
Shows the next row in the Current List.       ** 1: ent_table table**
                              ENTRY TYPE TABLE

Entry Type Code..[DSPL] [Asset Salvage/Disposal  ]  Amount Type...........[ACT]
Default Description.....[Asset Disposal          ]  Itemize on Statement..[N]
Payment Form.....[  ]                               Maximum Transactions..[ 10]

Ignore: Subsidiaries....[N]  Constraints.......[N]
Debit or Credit to A/R..[ ]  A/P...............[ ]  Subsidiary Post.......[INV]

     Journal Allowed To Post Entry     |       Chart of Permission Codes
     ------------------------------    |       -------------------------
       AC - [Y]         JC - [N]       |            DR  CR       DR  CR
       AP - [N]         PC - [Y]       |        B - 1   1    H - 0>  0>
       AR - [N]         PD - [N]       |        C - 0   1    I - 0   0>
       BG - [N]         PR - [N]       |        D - 1   0    J - 0>  0
       CH - [N]         PS - [N]       |        E - 1<  1<   K - 1>  1>
       CK - [N]         QC - [N]       |        F - 0   1<   L - 0   1>
       DA - [N]         SA - [N]       |        G - 1<  0    M - 1>  0
       FA - [N]         SB - [N]       |                     N - 0   0
                                       |        Cash..[E]    S/L...[Y]
```

Fields on the Entry Type Table Screen

The following fields on the Entry Type Table PERFORM screen contain codes or values required for setting up Fixed Assets.

Note: Fields that do not appear in this table are blank.

Amount Type
The type of amount that is used in the entry (e.g., ACT for Actual).

Cash
The number of cash transactions that you want to allow for the entry type. For Fixed Assets, enter **E** in this field.

Note: The E signifies that you want to allow for zero or one debit, and zero or one credit, to your institution's Cash account.

Default Description
The description you want to use for entries created for this entry type.

Entry Type Code
The four-character code that identifies the type of entry (e.g., DEPR for depreciation entries).

Note: The description of the code appears to the right of the four-character field.

Ignore: Constraints
A Y/N field indicating whether you want to permit the posting and creation of balance records during closed balance periods. Enter **Y** to allow record posting and creation during closed balance periods.

Ignore: Subsidiaries
A Y/N field indicating whether you want the entry type to post to subsidiary accounts. Enter **N** for fixed assets.

Itemize on Statement
A Y/N field indicating whether you want the system to itemize subsidiary information on student statements. Enter **N** for fixed assets.

Journal Allowed to Post Entry
The journals to which the entry type applies. For the capitalization type, only the AC journal contains Y; for depreciation and disposal types, both the AC and the PC journals contain Y.

Maximum Transactions
The maximum number of transactions (e.g., lines of an accounting entry) that you want to allow for the entry type.

S/L
A Y/N field indicating whether you want to allow subsidiary transaction types for the entry type. Enter **N** for fixed assets.

Subsidiary Post
A code indicating whether the payment or invoice side of the Subsidiary Total record is affected by the entry type. Enter **INV** for fixed assets.

Note: Valid codes are as follows:

- INInvoice
- PAPayment

The Fixed Asset Type Table

The Fixed Asset Type table defines the different types of fixed assets that you use at your institution.

Fields in the Fixed Asset Type Table

The following lists and describes the fields on the Fixed Asset Type table.

Asset Type Code
An eight-character code defining the type of asset (e.g., BLDG for buildings, or FURN for furniture).

Summarize
A Y/N field indicating whether you want to group, or summarize assets of this type. This field is especially useful for capital assets that have a low per-unit cost.

Depreciate
A Y/N field indicating whether your institution intends to depreciate the asset type. For example, enter **Y** for buildings, but enter **N** for land.

Beginning Date
The beginning effective date for the asset type. Leave this field blank if your institution does not have a specific beginning date.

Description
A 24-character description of the code (e.g., “Buildings” or “Furniture”).

Capitalize
A Y/N field indicating whether your institution capitalizes the type of asset. For most fixed assets, enter **Y**. For any assets that you do not want to capitalize, but you still want to track in the Fixed Assets module, enter **N**. Assets you may not want to capitalize include fully depreciated assets (e.g., a 100 year old building).

Depreciation Method

The calculation method you want to use for the asset type. Valid codes, as defined in macros, are as follows:

- STRAIGHT_LINSL
- DECLINE_20D200
- DECLINE_15D150
- DECLINE_12D125
- ACRS_ AC3
- ACRS_ AC5
- ACRS_1 AC10
- ACRS_1 AC15
- ACRS_1 AC18
- ACRS_1 AC19
- MACRS_MA3
- MACRS_MA5
- MACRS_MA7
- MACRS_1MA10
- MACRS_1MA15
- MACRS_2MA20
- MACRS_REAMANP
- MACRS_RENTAMARP

Note: For more information about the Accelerated Cost Recovery System (ACRS) and Modified Accelerated Cost Recovery System (MACRS) methods of depreciation, see *Internal Revenue Service Publication 534*.

CAUTION: These values must appear in the Fixed Asset Type table exactly as they appear here in order for *fixpost* to compute depreciation correctly. Do not change any of the values in this file.

Ending Date

The ending effective date for the asset type. Leave this field blank if your institution does not have a specific ending date.

Capitalization Account

The fund, function, object, and subfund you want to debit when you acquire the asset type.

Note: You enter the account you want to credit (also called the source account) on the Fixed Asset record for each individual asset.

Accumulated Depreciation Account

The fund, function, object, and subfund that want to credit when you record depreciation for the asset.

Depreciation Expense Account

The fund, function, object, and subfund you want to debit when you record depreciation for the asset.

Setting Up Cashier Tables

Introduction

To initiate the features of the *cashier* program, you must set up and update the following CX database tables:

- Cash Transaction Type table (cashent_table)
- Document table (doc_table)
- Entry Type table (ent_table)
- General Ledger Permission table (glperm_table)
- Payment Form table (pay_frm_table)
- Payment Plan table (payplan_table)
- Subsidiary table (subs_table)
- Subsidiary Account record (suba_rec)
- Subsidiary Total table (subt_table)

Cash Transaction Type Table

The Cash Transaction Type table (cashent_table) controls default G/L account, default subsidiary, and default entry information. The following list contains the fields that you must update and describes the update required in each field:

Classification

One of four values (C, R, D, or T) that correspond to the four transaction classifications. The following values are valid:

C (Check Cashing)

Use this classification for transactions that do not affect the overall balance of the cash drawer (e.g., enter C for transactions that make change or cash checks out of the cash drawer).

D (Disbursements)

Use this classification for transactions involving payment or outlay of funds (e.g., enter D for transactions such as travel advances, refunds, and cash withdrawals).

R (Receipts)

Use this classification for transactions involving collection of funds (e.g., enter R for transactions such as student payments, gift receipts, dorm deposits, and miscellaneous income).

T (Transfers)

Use this classification for transactions involving fund transfers between the cash drawer account and any other account (e.g., enter T for transactions such as cash transfers from bank or vault, cash transfers to bank or vault, or cash transfers between cash drawers).

Classification Default

Do you want to indicate this cash transaction type as the default type?

- If yes, enter **Y** to indicate that this cash transaction type is the default.
- If no, enter **N** to indicate that this cash transaction type is not the default.

Note: Enter Y for the cash transaction type most often selected for each classification. Make sure that exactly one table entry for each classification has value of Y. If all table entries contain N, the *cashier* program requires you to select a default for each classification.

Code

A four-character alphanumeric field that represents a single cash transaction type. Jenzabar, Inc. suggests you use an abbreviation naming convention to make codes easier to remember.

Example: Enter SPMT for Student Payment or TADV for Tuition Advance.

Debit

One of two values (C or D). Valid values are:

- C (credit)

Specifies that the Default G/L Account field (in this table) is to be credited.

- D (debit)

Specifies that the Default G/L Account field (in this table) is debited.

Example: Default G/L Account is the control account for the S/A subsidiary. Enter **C** to *credit* a S/A subsidiary, as in a student payment, to imply that the cash drawer account is debited.

Default Balance Code

The subsidiary balance code you want to use with the transaction type. Review the Subsidiary Balance table for valid balance codes.

Note: Enter blanks for all non-subsidiary type transactions and for non-balance subsidiaries (e.g., tsubs_inv_bal_used = N and tsubs_pay_bal_used = N).

Example: Enter SB for Session Billing.

Default Balance Period

One of four values (C, D, N, or O) that correspond to the four methods of defaulting payment amounts against outstanding balances.

Note: The *cashier* program associates dates with balance periods through the fiscal calendar and determines the current balance period by using today's date. All four methods of defaulting add a balance record for the current balance period, if no open balances exist.

Valid values are

- C (Current balance period)

Applies remittance amounts to the current balance (e.g., if SP97 is the current balance period, the remittance amount entered defaults against the SP97 balance).

- D (Oldest debit balance period)

Applies remittance amounts to the balance period with the least-recent date balance greater than zero (e.g., if SP97 is the current balance period, and a student has a credit balance in FA95, a debit balance in FA96, and a debit balance in SP97, the remittance amount entered defaults against the FA96 balance).

- N (Newest balance period)

Applies remittance amounts to the balance period with the most-recent date (e.g., if SP97 is the current balance period, and a student has open balances for FA96, FA97, SP97, and FA98 balance periods, the remittance amount entered defaults against the FA96 balance).

- O (Oldest balance period)

Applies remittance amounts to the balance period with the least-recent date (e.g., if SP97 is the current balance period, and a student has open balance records for FA95, FA96, and SP97 balance periods, the remittance amount entered defaults against the FA95 balance).

Default G/L Account
The general ledger account number associated with the transaction type. For transaction types involving subsidiaries, you must enter the subsidiary control account number associated with the subsidiary.

Default G/L Entry Description
Description to use with the transaction type. This value overrides the default entry description from the Entry Type table.

Default G/L Entry Type
The entry code you want to use with the transaction type. The Entry Type table contains valid entry codes.

Note: You must enter the value, and you cannot override the value within the *cashier* program.

Default Subs Code
The subsidiary code to which *cashier* has permission, or blanks for all non-subsidiary type transactions.

Example: Enter S/A for Student Accounts, D/D for Dorm Deposit, or all blanks for Gift Receipts.

Default Total Code
Any subsidiary total code to which the *cashier* program has permission. Enter blanks for all non-subsidiary type transactions and for subsidiaries that do not use total codes (i.e., tsubs_inv_tot_used = N and tsubs_pay_tot_used = N).

Example: Enter PAID for Payment or FINE for Fine Payment.

Description
A 24-character alphanumeric description of the Code field entry.

Document Table

The Document table (doc_table) controls the cash drawer G/L account, default drawer closing G/L account, default drawer closing amount, and document station restrictions. Some of these data elements appear on the second Table Maintenance screen, which you can access using the **Screen** command.

The following list contains the fields that you must update and describes the update required in each field:

Beginning Drawer Balance
The beginning cash drawer balance as of the last reconciliation.

Note: You cannot update this field in the PERFORM screen; the *cashier* program maintains the field.

Closing Drawer Balance
An amount to remain in the cash drawer after closing.

Example: If the Current Drawer Balance is $1125.99 and the Closing Drawer Balance is $300.00, when closing or reconciling, *cashier* computes a deposit amount of $825.99 ($1125.99 - $300.00) and leaves a closing drawer amount of $300.00.

Note: You cannot override this amount within *cashier*; to change it, you must change the setup of the Document table.

Current Drawer Balance

The current cash drawer balance as of last transaction posted for this document station.

Note: You cannot update this field; *cashier* maintains it automatically.

Drawer Closing account

The general ledger account number to use as the default account debited during the closing and reconciliation processes.

G/L Account

The general ledger account number associated with the cash drawer. The *cashier* program debits or credits this account number in every entry it posts.

Note: Jenzabar, Inc. recommends you use unique account numbers for each cash drawer account since the amounts in each of the drawers is logically separate. However, *cashier* does not require unique account numbers to function properly.

Restricted To User

A code that restricts the document station to a single user. To restrict the station, enter the UNIX ID number (from the password file) of the desired user. To set up the station without restriction, enter a zero.

Example: If the UNIX ID number 149 is associated with user *jdoe* and you enter **149**, only the *jdoe* login can select this document station number.

Entry Type Table

The Entry Type table (ent_table) controls the number of debits and credits that *cashier* can post for a single transaction. The following figure displays an entry that you must make to the table:

```
                                ENTRY TYPE TABLE

Entry Type Code..[RECN] [Cash reconciliation     ]  Amount Type...........[ACT]
Default Description.....[Recon  /Cash Reconciled ]  Itemize on Statement..[I]
Payment Form.....[CA]    Cash                       Maximum Transactions..[  0]

Ignore: Subsidiaries....[N]  Constraints.......[N]
Debit or Credit to A/R..[ ]  A/P...............[ ]  Subsidiary Post.......[PAY]

     Journal Allowed To Post Entry     |       Chart of Permission Codes
     ------------------------------    |       -------------------------
       AC - [Y]         JC - [N]       |            DR  CR       DR  CR
       AP - [N]         PC - [N]       |        B - 1   1    H - 0>  0>
       AR - [N]         PD - [N]       |        C - 0   1    I - 0   0>
       BG - [N]         PR - [N]       |        D - 1   0    J - 0>  0
       CH - [Y]         PS - [N]       |        E - 1<  1<   K - 1>  1>
       CK - [N]         QC - [N]       |        F - 0   1<   L - 0   1>
       DA - [N]         SA - [N]       |        G - 1<  0    M - 1>  0
       FA - [N]         SB - [N]       |                     N - 0   0
                                       |        Cash..[E]    S/L...[N]
```

Required Entry Type Table Codes

To function properly, the *cashier* closing and reconciliation processes require the entry of the following recommended codes in the Entry Type table:

- RECN (Reconciliation)
- CLOS (Closing)

To use codes other than RECN and CLOS, you must update the CH_CLOSE_ENTTYPE_DEF and CH_RECON_ ENTTYPE_DEF macros.

General Ledger Permission Table

Modify the General Ledger Permission table (glperm_table) to ensure that you have access to all permissible general ledger and subsidiary control accounts. For more information about the glperm_table, see *General Ledger Technical Manual.*

Payment Form Table

The Payment Form table (pay_frm_table) controls the order of appearance of payment form types in the reconciliation process. To control this order, change the following field on the pay_frm_table.

Sort Order
A unique numeric value for all payment form types to form a priority order, with 0 being the highest priority, 1 being the next highest priority, etc. The order determines the order of appearance of each payment form on the reconciliation summary screen.

Note: The *cashier* program does not require a unique Sort Order value for the reconciliation process to function properly.

Required Payment Form Table Code

To function properly, the *cashier* closing and reconciliation processes require the entry of the DC (drawer closing) code in the Payment Form table. To use a code other than DC, you must update the CH_CLOSE_PAYFRM_DEF and CH_RECON_PAYFRM_DEF macros.

Payment Plan Table

The Payment Plan table (payplan_table) defines valid payment plans; *cashier* uses the table to generate payment coupons. The following list contains the fields that you must update and describes the update required in each field:

Note: If your institution does not use payment plans, omit updates to the Payment Plan table.

Code
A four-character code that designates the payment plan.

Day Activated
The due date of the payment (a number ranging from 1 to 31) within the month indicated in the Month field.

Description
A 24-character description of the value entered in the Code field.

Month
A number ranging from 1 to 99 representing the month, relative to the payment plan beginning date, in which the payment is due.

- Enter a zero (0) to indicate the payment is due during the current month.
- Enter one (1) to indicate the payment is due during the month after the current month.

Example: If a payment plan begins in August, a 0 in the Month field means the repayment is due in August. A 2 in the Month field means the repayment is due in October.

Per Cent Due
Enter the percentage of the outstanding balance that is due this payment.

Setting Up Payment Coupons

If the institution wants to distribute payment coupons with a student's statement, payment plans divide the student's current balance by the number of payments to determine the amount that you want printed on payment coupons.

You can use the payment plan and/or payment plan coupons to assist the school and the students in keeping track of the amount a student owes the school. The coupons help remind the students when their payments are due. To activate, enter the Payment Plan and the Payment Plan Date in the Subsidiary Account record.

Associating Minimum Payment

You can associate minimum payment with the payment plan with a macro-driven option, which can be turned on or off by the school.

Subsidiary Table

The Subsidiary table (subs_table) controls the posting to subsidiaries, statement form defaulting, and accounts receivable discounts in the *cashier* program. The following list contains the fields you must update and describes the update required in each field:

Allow CASHIER Post
A Y/N field indicating whether you want *cashier* to post transactions to the subsidiary in the tsubs_ch_post column. Enter **Y** for each subsidiary to which *cashier* can post transactions, and **N** for each subsidiary for which you do not want *cashier* to post transactions.

Note: Entering N overrides General Ledger (G/L) permissions.

CAUTION: Do *not* allow the *cashier* program permission to the Wages Payable (W/P) subsidiary. The defaulting of remittance amounts does not function properly if the *cashier* program has permission to the W/P subsidiary.

Default Statement Form
A default statement form type to associate with this subsidiary (e.g., stmt1, stmt2, stmt3, or stmt4). You can override this form name within *cashier*.

Note: This field is not required for the statement option to function properly. If you do not make an entry, however, you must enter a valid statement form name in *cashier*.

Discount
The general ledger (G/L) account number (fund, function, object, and subfund) to track discounts taken in the receivables subsidiary.

Note: The *cashier* program requires the account number only for receivable subsidiaries that use discounts.

Subsidiary Account Record

The Subsidiary Account record (suba_rec) controls payment plans and their beginning dates. The following list contains the fields that you must update and describes the update required in each field:

Note: If your institution does not use payment plans, omit updates to the Subsidiary Account record.

Payment Plan

A payment plan code that is valid in the Payment Plan table. Leave blank to indicate that a student is not on a payment plan.

Note: This field must contain a valid payment plan code to place the ID on a payment plan.

Payment Plan Date

The payment plan beginning date. The field determines the payment plan beginning month and year. The *cashier* program determines the due date of a payment from the Payment Plan table.

Note: The *cashier* program only uses the month and year portion of the field.

Subsidiary Total Table

The Subsidiary Total table (subt_table) controls posting to subsidiary total codes in the *cashier* program. You must update the following field:

CASHIER Post

A Y/N field indicating whether *cashier* should post transactions to the Subsidiary Total record.

Note: All subsidiary totals with non-zero balances appear in *cashier*, whether or not *cashier* can post to that subsidiary total.

Cash Receipt Forms

Introduction

To use cash receipt forms for printing cash receipts in *cashier*, you must make your receipt forms conform to the CX *cashier* program format, add other cash receipt forms as needed, or update default forms to match the receipts you currently use.

After you initially set up cash receipt forms, you may need to modify forms as your institution's needs change. This section also provides the steps to performing the following modifications:

- Adding/removing subsidiary total information
- Enabling/disabling manual cash receipt numbering

Note: To perform conversion and initialization tasks in this section, you must have experience in the following:

- Using the MAKE processor
- Using the *vi* editor
- Customizing forms

Cash Receipt Forms

CX supports four new cash receipt forms for printing cash receipts in *cashier*, including the following:

- cash_rcpt1
- cash_rcpt2
- cash_rcpt3
- cash_rcpt4

Cash Receipt Forms Directory

You must locate cash receipt forms in the following directory path: $CARSPATH/modules/accounting/forms/cashier.

The *cashier* program requires the use of this directory for printing cash receipts.

Merging Cash Receipt Form Modifications

Due to the number of new fields and modifications to old fields, Jenzabar, Inc. strongly recommends that you *not* use the Make merge command to merge modifications to your cash receipt form. A large number of overlaps can occur.

Converting Cash Receipt Forms

The following list describes the steps to convert your current cash receipt form into a format recognized by *cashier*:

1. Copy your current cash receipt form to the new location by entering the following:

   % cd $CARSPATH/modules/accounting/forms/cashier

   % make add F=cash_recpt

2. Does your institution currently use voucher to do cash receipting and now wants to use cashier?
   - If yes, do the following:
     - At the UNIX prompt, enter **vi cash_rcpt**.
     - Delete the file contents, from the revision control information to the end of the file.
     - Enter the following command: **:r ../voucher/cash_rcpt**. This command reads the contents of the voucher receipt into the cash_rcpt file.
     - Delete the revision control information from the old voucher file.
   - If no, go to step 3.
3. Delete the *attributes* section from your current form.
4. Copy the *attributes* section from one of the four standard CX cash receipt forms (cash_rcpt1, cash_rcpt2, cash_rcpt3, or cash_rcpt4) into your form.
5. Edit the *form* section of your form and modify references to fields where you have changed the field's name.
6. Edit the *attributes* section of your form and modify any format, group, and scroll clauses to correspond to your customized form.

   CAUTION: In order to maintain customizations to your cash receipt form other than those mentioned above, such as lookup clauses, add them to the *attributes* section of your form.
7. Install the converted form by entering the following: **% make cii F=cash_recpt L="Convert Cash Receipt Form"**

Adding Cash Receipt Forms

You can create any number of cash receipt forms. To name the file containing the cash receipt form, you can use any valid UNIX filename up to 14 characters in length. The following list describes the steps to add a cash receipt form:

1. Add the file for the cash receipt form by entering the following at the UNIX prompt:
   cd $CARSPATH/modules/accounting/forms/cashier
   make add F=<*filename*>
2. Copy one of the four standard CX cash receipt forms (cash_rcpt1, cash_rcpt2, cash_rcpt3, or cash_rcpt4) that most closely resembles the form you wish to create into your form.

   Example: Enter the following at the UNIX prompt:
   cp cash_rcpt2 <*filename*>
3. Edit your new form as desired using only valid cash receipt form fields. For a list of the fields, see *Valid Cash Receipt Form Fields* in this section.
4. Install your new form by entering the following at the UNIX prompt:
   make cii F=<filename> L="New Cash Receipt Form"

Valid Cash Receipt Form Fields

The following list describes the valid fields in cash receipt forms:

acct_table
Any field (e.g., acct_text)

addr1...addr6
Formatted ADR address

address1, address2, address3
Formatted ADR address

amt
G/L applied amount

appamts
Subsidiary applied amount

balcods
Total Detail Balance Code

balprds
Total Detail Balance Period

bals
Subsidiary balance amount

cashamt
Total amount received

chgamt
Total change given

cntr_table
Any field (e.g., cntr_text)

desc1s
Account description 1

desc2s
Account description 2

fund_table
Any field (e.g., fund_text)

gla_rec
Any field (e.g., acct_desc1)

gle_rec
Any field (e.g., gle_doc_no)

id_rec
Any field (e.g., id_no or ss_no)

netamt
Total applied amount

proj_table
Any field (e.g., proj_text)

recaddr1, recaddr2, recaddr3, recaddr4, recaddr5, recaddr6
Formatted ADR address

suba_rec
Any field (e.g., suba_subs)

subsaddr1, subsaddr2, subsaddr3, subsaddr4, subsaddr5, subsaddr6
Address for the subsidiary ID

subsids
Subsidiary ID number

subsnms
Subsidiary name

totamts
Total Detail Apply Amount

totdescs
Total Detail Description

unit_table
Any field (e.g., unit_text)

vch_rec
Any field (e.g., vch_no)

Adding or Removing Subsidiary Total Information

The following steps add or remove subsidiary total information from the Cash Receipt form:

1. Select the cash receipt form to modify by entering the following at the UNIX prompt:
   cd $CARSPATH/modules/accounting/forms/cashier
   make co F=<*filename*>

   Note: You can use one of the four standard cash receipt forms or a previously added form for <*filename*>. The standard cash receipt form cash_rcpt4 depicts the proper usage of all the fields available with this feature.

2. Edit the selected form using only valid cash receipt form fields.

   Note: The four scroll fields balcods, balprds, totamts, and totdescs apply to this feature. You must create a scrolling region large enough to accommodate the maximum number of lines of detail that you want on the form. The *cashier* program determines the scrolling region size. If the number of lines of detail exceeds this size, the *cashier* program does not generate another form and no excess information appears.

3. Install the modified form by entering the following at the UNIX prompt:
   cd $CARSPATH/modules/accounting/cashier/forms
   make cii F=<*filename*> L="Add/Remove Subs Total Info"

Enabling and Disabling Manual Cash Receipt Numbering

The following steps enable or disable manual cash receipt numbering. This feature is disabled in the standard CX product.

1. Enter the following at the UNIX prompt:
   cd $CARSPATH/src/accounting/cashier/SCR
   make co F=entry
   vi entry
2. Add or remove the recpt field using an existing line in the screen definition section.

   Note: The maximum width for the recpt field is 11.

   Example: The following is a portion of the screen definition section showing the inclusion of the recpt field:

```
screen
{
Transaction Type.[ctyp^ctext                  ]
Receipt Number...[recpt      ]
G/L Entry Type...[gltp^gldesc                 ]
}
```

3. Add the recpt field (or remove it from) the entry_group within the *attributes* section. This notifies *cashier* to allow/disallow entry of this field on the screen.

   Note: The order of fields in the entry_group represents the order in which *cashier* requests input.

   Example: The following portion of the *attributes* section shows inclusion of recpt field:

```
entry_group: optional,
     autonext,
     group = ( ctyp, recpt, idno, idnm, ssno, pf, pfno, sbno,
               payamt, appamt, gldesc );
```

4. Install the screen definition file by entering the following at the UNIX prompt:
   cd $CARSPATH/src/accounting/cashier/SCR
   make cii F=entry L="Enable/Disable Manual Receipt Number"

Statement Forms

Introduction

To use statement forms for printing statements in *cashier*, you must convert your existing statement forms to the CX *cashier* program format, and add other statement forms as needed.

After you initially set up statement forms, you may need to make modifications as your institution's needs change. This section also provides the steps to performing the following modifications:

- Adding/removing minimum payment information
- Adding/removing payment plan information
- Adding/removing pending financial aid

Note: To perform conversion and initialization tasks in this section, you must have experience in the following:

- Using the MAKE processor
- Using the vi editor
- Customizing forms

Statement Forms

CX supports four new statement forms for printing statements, including the following:

- stmt1
- stmt2
- stmt3
- stmt4

Statement Forms Directory

You must locate statement forms in the following directory path:
$CARSPATH/modules/accounting/forms/stmt

The *stmt*, *acquery*, *bursar*, and *cashier* programs require the use of this directory for printing statements.

Converting Statement Forms

The following list describes the steps to convert your current statement form into a format recognized by the *cashier* program (and the *acquery*, *bursar*, or *stmt* programs).

1. Copy your current statement form to the new location by entering the following at the UNIX prompt:
   cd $CARSPATH/modules/accounting/forms/stmt
   make add F=stmt
   vi stmt
2. Delete the file contents from just below the revision control information to the end of the file by entering the following while in *vi* Command mode:
   :r $CARSPATH/modules/acctsrecv/forms/stmt/stmt
3. Delete the old revision control information
4. Delete the *attributes* section from your current form.
5. Copy the *attributes* section from one of the four standard CX statement forms (stmt1, stmt2, stmt3, or stmt4) into your form.

6. Edit the *form* section of your form and modify references to fields whose names have changed.
7. Edit the *attributes* section of your form and modify any format, group, and scroll clauses to correspond to your customized form.

   CAUTION: To maintain customizations to your statement form other than those mentioned above, such as lookup clauses, add them to the *attributes* section of your form.
8. Install the converted form by entering the following at the UNIX prompt:
   make cii F=stmt L="Convert Statement Form"

How to Add Statement Forms

You can create any number of statement forms. To name the file containing the statement form, you can use any valid UNIX file name up to 14 characters in length. The following steps enable you to add a statement form.

1. Add file for cash receipt form by entering the following at the UNIX prompt:
   cd $CARSPATH/modules/accounting/forms/stmt
   make add F=<*filename*>
2. Copy one of the four standard CX statement forms (stmt1, stmt2, stmt3, or stmt4) most closely resembling the form you wish to create, into your form.

   Example: cp stmt2 <*filename*>
3. Edit your new form as desired using only valid statement form fields (see the following list).
4. Install your new form by entering the following at the UNIX prompt:
   make cii F=<*filename*> L="New Statement Form"

Valid Statement Form Fields

The following list describes the valid fields in statement forms:

addr1...addr7
Formatted ADR address

aids
Financial aid amount

balmsg
Balance text (e.g., Balance Due)

bals
Running statement balance

bdate1, bdate2
Beginning statement date

cduemsg
Current text (e.g., Current Due)

com
Statement comment

cont
Statement form continue message

crs
Credit transaction amount

curdue
Current due amount

dates
Transaction date

descs
Transaction description

drcrs
Credit/Debit transaction amount

drs
Debit transaction amount

edate1
Ending statement date

edate2
Ending statement date

fin_bal1, fin_bal2, fin_bal3
Statement balance

id_rec
Any field (e.g., id_no, ss_no)

mindue1, mindue2
Minimum balance due

minmsg1
Minimum payment due text

minmsg2
Minimum payment due text

paymsg
Payment due text

pdate
Statement print date

pdue
Past due amount

pduemsg
Past due test

prds
Trans. balance period

prev
Previous due amount

sbcust_rec
Any field (e.g., sbcust_test)

stmt_cr
Total statement credits

stmt_dr
Total statement debits

stu_acad_rec
Any field (e.g., stuac_cl)

stu_serv_rec
Any field (e.g., stusv_campus_box)

sub1, sub2
Student subsidiary number

subb_table
Any field (e.g., tsubb_text)

subs_table
Any field (e.g., tsubs_text)

totaid
Total financial aid

Add/Remove Minimum Payment Information

The following list contains the steps to add or remove Minimum Payment information from student statements:

1. Modify the enable macro for the statement minimum payment feature by entering the following at the UNIX prompt:
   cd $CARSPATH/macros/custom
   make co F=financial
   vi financial
2. Do you want to add minimum payments to the statement?
   - If yes, change the value of the ENABLE_MIN_ PLAN_LOGIC macro to Y.
   - If no, change the value of the ENABLE_MIN_ PLAN_LOGIC macro to N.
3. Install the macro file by entering the following at the UNIX prompt:
   cd $CARSPATH/macros/custom
   make cii F=financial L="Add/Remove Minimum Payments"
4. Install the C program include file referencing modified macro by entering the following at the UNIX prompt:
   cd $CARSPATH/include/util
   make reinstall F=libbill
5. Reinstall the statement library by entering the following at the UNIX prompt:
   cd $CARSPATH/src/Lib/libbill
   make reinstall
6. Reinstall all programs that include the statement library by entering the following at the UNIX prompt:
   cd $CARSPATH/src/accounting/acquery
   make reinstall >& make.out &
   cd $CARSPATH/src/accounting/bursar
   make reinstall >& make.out &
   cd $CARSPATH/src/accounting/cashier
   make reinstall >& make.out &
   cd $CARSPATH/src/acctsrecv/stmt
   make reinstall >& make.out &

   Note: You can reinstall these programs simultaneously or in any order.

How to Add or Remove Payment Plans

Use the following procedure to add or remove payment plan information from student statements:

1. Modify the enable macro for the payment plan feature by entering the following at the UNIX prompt:
   cd $CARSPATH/macros/custom
   make co F=financial
   vi financial
2. Do you want to add payment plans to the student statement?
   - If yes, change the value of the ENABLE_FEAT_ PAY_PLAN macro to Y.
   - If no, change the value of the ENABLE_FEAT_ PAY_PLAN macro to N.
3. Install the macro file by entering the following at the UNIX prompt:
   cd $CARSPATH/macros/custom
   make cii F=financial L="Add/Remove Payment Plan"
4. Install the C program include file that references the M4 macro changed in the above steps by entering the following at the UNIX prompt:
   cd $CARSPATH/include/util
   make reinstall F=libbill
5. Reinstall the statement library by entering the following at the UNIX prompt:
   cd $CARSPATH/src/Lib/libbill
   make reinstall
   cd $CARSPATH/src/Lib/libcars
   make reinstall
6. After you have reinstalled the statement library, reinstall all programs that include the statement library by entering the following at the UNIX prompt:
   cd $CARSPATH/src/accounting/acquery
   make reinstall >& make.out &
   cd $CARSPATH/src/accounting/bursar
   make reinstall >& make.out &
   cd $CARSPATH/src/accounting/cashier
   make reinstall >& make.out &
   cd $CARSPATH/src/acctsrecv/stmt
   make reinstall >& make.out &

   Note: You can reinstall these programs in any order or simultaneously.
7. Reinstall the menusrc file that includes the payment plan perform screen and the payment plan report on the menu by entering the following at the UNIX prompt:
   cd $CARSPATH/menusrc/fiscal/stubill/recmaint/
   make reinstall F=menudesc
8. Reinstall all program screens and statement forms that use the m4 macro you changed by entering the following at the UNIX prompt:
   cd $CARSPATH/modules/accounting/progscr/bursar
   make reinstall F=stmtinfo >& make.out &
   cd $CARSPATH/modules/accounting/progscr/stmt
   make reinstall F=ALL >& make.out &
   cd $CARSPATH/modules/accounting/forms/stmt
   make reinstall F=ALL >& make.out &

   Note: You can reinstall these programs in any order or simultaneously.

9. Reinstall the PERFORM screen using the m4 macro changed above by entering the following at the UNIX prompt:
   cd $CARSPATH/modules/acctsrecv/screens
   make reinstall F=subacct

How to Add or Remove Pending Financial Aid

Use the following procedure to add or remove pending financial aid from student statements:

1. Modify the enable macro for the statement pending financial aid feature by entering the following at the UNIX prompt:
   cd $CARSPATH/macros/custom
   make co F=financial
   vi financial
2. Do you want to add pending financial aid to the student statement?
   - If yes, change the value of the ENABLE_STMT_ DISP_PEND_AID macro to Y.
   - If no, change the value of the ENABLE_ STMT_ DISP_PEND_AID macro to N.
3. Install the macro file by entering the following at the UNIX prompt:
   cd $CARSPATH/macros/custom
   make cii F=financial L="Add/Remove pending financial aid"
4. Install the C program include file that references the m4 macro changed in the above steps by entering the following at the UNIX prompt:
   cd $CARSPATH/include/util
   make reinstall F=libbill
5. Reinstall the statement library by entering the following at the UNIX prompt:
   cd $CARSPATH/src/Lib/libbill
   make reinstall
   cd ../libcars
   make reinstall
6. After the statement library has been reinstalled, reinstall all programs that include the statement library by entering the following at the UNIX prompt:
   cd $CARSPATH/src/accounting/acquery
   make reinstall >& make.out &
   cd $CARSPATH/src/accounting/bursar
   make reinstall >& make.out &
   cd $CARSPATH/src/accounting/cashier
   make reinstall >& make.out &
   cd $CARSPATH/src/acctsrecv/stmt
   make reinstall >& make.out &

   Note: You can reinstall these programs in any order or simultaneously.
7. Reinstall the program screen that uses the m4 macro you just changed by entering the following at the UNIX prompt:
   cd $CARSPATH/modules/accounting/progscr/bursar
   make reinstall F=stmtparam

Cash Drawer Maintenance

Introduction

To activate the cash drawer reconciliation and closing functions of *cashier*, and to ensure that correct amounts exist in the cash drawer accounts, you must post current cash drawer account information in a format recognized by *cashier*. These pages explain how to add or remove the Close Drawer option from the Drawer Menu.

How to Add or Remove the Close Drawer Option

Use the following procedure to add or remove the Close Drawer option from the Drawer Menu. This feature is enabled in the CX standard product.

1. Modify the C program include macro by entering the following at the UNIX prompt:
   cd $CARSPATH/include/custom
   make co F=cashier
   vi cashier
2. Do you want to add the Close Drawer option to the Drawer menu?
   - If yes, comment out the DSPL_CLOSE_OPT macro to add the Close Drawer option to the Drawer menu.
   - If no, un-comment out the DSPL_CLOSE_OPT macro to remove the Close Drawer option from the Drawer menu.

   Syntax:
   ```
   Enabled:   #define DSPL_CLOSE_OPT
   Disabled:  /* #define DSPL_CLOSE_OPT */
   ```
3. Install the C program include file by entering the following at the UNIX prompt:
   cd $CARSPATH/include/custom
   make cii F=cashier L="Add/Remove Close Drawer Option"
4. Reinstall the cashier program by entering the following at the UNIX prompt:
   cd $CARSPATH/src/accounting/cashier
   make reinstall >& make.out

Replenishing Petty Cash in the Cashier's Office

Introduction

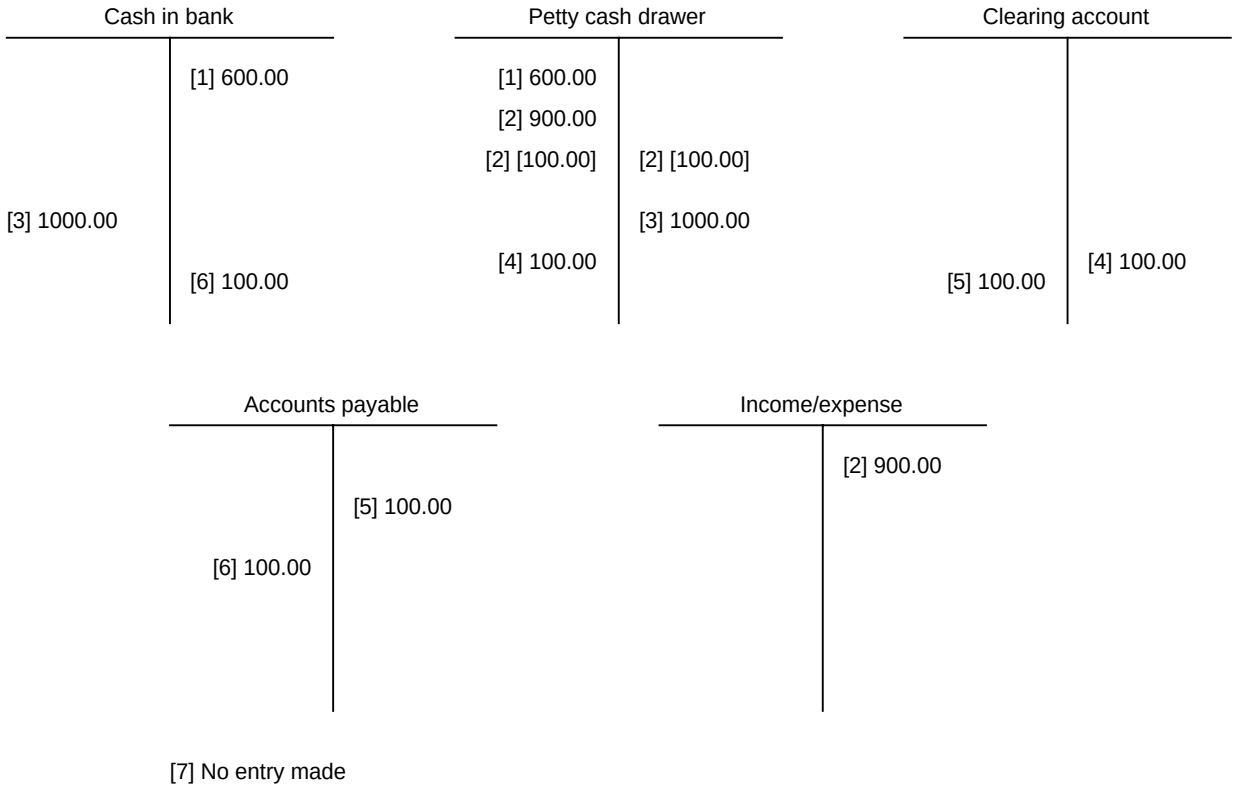
The *cashier* program enables you to replenish petty cash with physical money from the bank. These pages describe how to replenish petty cash by writing a check through Accounts Payable.

Transactions Affecting Cash Balance

You must use *cashier* to make any transactions that affect the cash balance and are recognized by the *glaudit* program. For more information about *glaudit*, see *General Ledger Technical Manual*.

Writing a Check Through Accounts Payable

The following figure depicts the process of writing a check through Accounts Payable to replenish petty cash. The numbered amounts that appear in the figure (e.g., [1]) are described below the figure.



The descriptions explain the numbered amounts displayed in the Accounts Payable Process figure (e.g., [1]).

[1]

The entry (600.00) opens or establishes the cash in the drawer balance.

Note: You must make an entry in *cashier* using the Transfer option in the Drawer menu. The *cashier* program makes a general ledger entry debiting the cash drawer account and crediting the cash in bank account.

[2]

The entry ($900.00) was taken in for tuition and other income. The entries below ($100.00) are cashed in checks.

Note: The cashed checks do not increase nor decrease the cash balance for the drawer. While the cash balance remains unchanged, the physical form of the money has changed, thus the money is not available for making change. Therefore, in this case, the actual cash in the drawer does not need to be replenished.

[3]

The entry (1000.00) is the entry *cashier* makes to close the drawer. The *cashier* program makes an entry debiting cash in the bank and crediting the drawer account.

Note: The deposit includes the cashed checks. When cashed checks are deposited, the actual cash balance decreases.

[4]

The entry (100.00) replenishes the cash balance back into the cash drawer account. The entry (100.00) must also be made within *cashier* using the Adjust function. The *cashier* program debits the cash drawer account and credits the clearing account.

Note: Jenzabar, Inc. recommends that you use a cash clearing account. You can use any account (it may even be the same account that you use for over/short); however, you cannot use a subsidiary account.

[5]

The entry (100.00) is the amount you want to replenish the drawer with by using the *purch* program to create a liability in the accounts payable subsidiary (A/P).

Note: The entry is a debit to the clearing account and a credit to the accounts payable liability subsidiary account.

[6]

The entry (100.00) is created by *cashier* for the written check. The entry debits the accounts payable liability account and credits the cash in bank account.

[7]

This represents when the check should be cashed and the money should be placed into the drawer account. The *cashier* program makes no entry for this action, and the actual cash is brought back into balance.

Summary Notes

- When the bank deposit in Entry [3] was made, the actual cash balance was reduced by $100.00 because $900.00 was taken in income and $100.00 in checks was cashed.
- The deposit of $1,000.00 (Entry [3]) included the cashed checks. Entry [4] increased the cash balance by $100.00 (the amount of checks that were cashed).

Setting Up Approve and Userid Tables for Purchasing

Introduction

The *approve* program controls the CX purchase order approval process. The process depends on restrictions placed on purchase order creation, which you establish during implementation through table setup.

The *approve* program uses information contained in various database files to implement a customized purchase order approval process. Entries made in the Approval table define restrictions placed on purchase order creation. Purchase orders that exceed these restrictions can require the approval of up to three budget managers.

How to Set Up the Userid Table

The *approve* program uses the Userid table to relate the UNIX system user id number to the ID number in the ID record file. This relationship enables the Approval table entries to use the ID number in the ID record. The following steps outline the procedures involved in creating appropriate entries in the Userid table:

1. Create a list of the users' UNIX ID numbers. The menu option to produce the report (UNIX ID Listing) resides on the Fiscal Management: Accounting: Table Maintenance: Financial Control menu.
2. Ensure that every user who will run the *approve* program is included on the list. If some users are not on the list, add them by using the menu option User ID on the Fiscal Management: Accounting: Table Maintenance: Financial Control menu.

   Note: This menu option runs *identry* so you perform queries to locate CX IDs and link them with the UNIX IDs.

How to Set Up the Approval Table

The Approval table is a multi- function table. It is used by two programs in differing capacities (the purchasing program *purch* and the purchase order approval program *approve*). A description of the different uses of the Approval table follows.

The *purch* Program

Uses the Approval table to determine the list of restrictions placed upon the current user. When *purch* is running, it compares the purchase orders and invoices prepared by this user. If this user prepares any item that exceeds his/her predefined restrictions, he/she is notified that this is the case. Subsequently, as the user commits items to the database, they are marked for needing approval, and each user responsible for approving or rejecting items receives electronic mail notification.

The *approve* program

Uses the Approval table to determine the list of ID numbers of users whose items need the budget manager's response. This list of ID numbers determines what items *approve* should select for the current user's response. Subsequently, as users approve or reject items, *approve* marks them in the database and sends electronic mail to each user who has had items approved or rejected.

Completing the Approval Table

The following example illustrates the data entry screen for the Approval table. The example displays the entry for a user (ID 21324) with no restrictions.

```
                  Purchase Order Approval Restriction Table
                  -----------------------------------------

       Id.....................[21324 ]  Moon, Alvin B.

       Category.......[    ]
       Code...........[    ]

       Approval Required....[N]
       Check Budget.........[N]

       Amount.................[$0.00      ]

       Approval Required by:

           Approval Id #1....[0     ]
           Approval Id #2....[0     ]
           Approval Id #3....[0     ]
```

For simplicity, additional table entry examples in this section will appear in the following form:

```
Check   Approval   Need Approval by:
 Id     Category   Code    Amount      Budget  Required    Id 1    Id 2    Id 3
-----  --------  ----  -----------  -------  ---------  ------  ------  ------
21324                 $      0.00     N         N          0       0       0
```

Setting Up Restrictions

The *purch* program searches the “Id” field for the current user's ID number to determine if that user has access to *purch*. The *approve* program searches the three "Need Approval by" ID fields for the current user's ID number in determining if that user has access to *approve*.

Any restrictions placed on an "Id" are applied to both the purchase orders and the invoices created by that "Id".)

Examples of Restrictions

Approval table entries combine to create various restriction possibilities. The following sub-sections describe in detail how *approve* interprets each restriction, and how these restrictions interact:

No restrictions

The following entry shows how to place no restrictions on an ID. This type of entry would typically be made for a budget manager. Although not necessary for a user to run *approve*, this entry is necessary for a user to run PURCH. The blank "Category" field instructs *approve* to ignore all other fields for ID number 21324, with the exception that the "Check Budget" and "Approval Required" fields are never ignored, regardless of the value of the Category field.

```
                                    Check   Approval   Need Approval by:
 Id    Category  Code    Amount      Budget  Required    Id 1    Id 2    Id 3
-----  --------  ----  -----------  -------  ---------  ------  ------  ------
21324                 $      0.00     N         N          0       0       0
```

Purchase order restrictions

You can create a table entry for an "Id" that restricts the "Id" to a specified "Amount" for a single purchase order. This type of entry also restricts the "Id" to the specified "Amount" for invoices against a single purchase order. To create an entry of this type, enter the following values in the Approve table:

Id	Category	Code	Amount	Check Budget	Approval Required	Need Approval by: Id 1	Id 2	Id 3
30454	PO		$ 1000.00	N	N	22363	0	0

A "Category" field value of "PO" instructs *approve* to allow ID number 30454 to create purchase orders totaling up to $1000.00 each without approval. The entry also allows ID number 30454 to create invoices totaling up to $1000.00 against each purchase order without approval. This type of entry allows ID number 30454 permission to any object code within any function. Any item created by ID number 30454 which exceeds $1000.00 will need the approval of id number 22363 before further processing (i.e. printing or distribution) of that item is allowed.

Note: If more than one entry is made with a "Category" value of "PO" for the same "Id", *purch* and *approve* will use the most restrictive entry to determine the status of items created by that "Id". If an item created by that "Id" exceeds more than one restriction, all relevant budget managers will be notified by electronic mail.

Function restrictions

You can create a table entry for an "Id" which restricts the "Id" to a specified "Amount" within a specified function code. The "Code" field is used in conjunction with the "Category" field to specify this type of restriction. To create this type of restriction, enter table values similar to the following:

Id	Category	Code	Amount	Check Budget	Approval Required	Need Approval by: Id 1	Id 2	Id 3
30454	FUNC	1603	$ 800.00	N	N	48353	0	0
30454	FUNC	1002	$ 700.00	N	N	48353	0	0

A "Category" field of "FUNC" instructs *purch* and *approve* to allow ID number 30454 permission to create a purchase order or an invoice against function code 1603 up to $800.00 and function code 1002 up to $700.00 without approval. These entries will restrict ID number 30454 to *only* these centers. Any attempts by ID number 30454 to create a purchase order or an invoice against any other function will not be allowed.

Purchase order and function restrictions

By combining the two "FUNC" restrictions (created by the *Function restrictions* above) with the PO restriction (created by the *Purchase Order restrictions* above), you create the following restrictions for ID number 30454:

ID number 30454 has permission to create a purchase order or an invoice against center 1603 or center 1002, up to specified "Amount" in each, provided the total for the purchase order or the total of the invoices does not exceed $1000.00.

Account restrictions

You can create a table entry for an ID that restricts the "Id" to a specified "Amount" within a specified account. The "Code" field works in conjunction with the "Category" field to specify this type of restriction. To create this type of restriction, enter table values similar to the following:

Id	Category	Code	Amount	Check Budget	Approval Required	Need Approval by: Id 1	Id 2	Id 3
30454	ACCT	6100	$ 450.00	N	N	48353	0	0
30454	ACCT	6002	$ 400.00	N	N	48353	0	0

A "Category" field of "ACCT" instructs *approve* to allow ID number 30454 permission to create a purchase order or an invoice against account 6100 up to $450.00 and account 6002 up to $400.00 without approval. These entries will restrict ID number 30454 to *only* these accounts. Any attempts to create a purchase order or an invoice against any other account (by ID number 30454) will not be allowed.

An "ACCT" restriction is the most restrictive of the three "Category" restrictions ("PO", "FUNC", "OBJ"). This means that *approve* will apply an "ACCT" restriction against a purchase order or an invoice before applying a "FUNC" or a "PO" restriction. An "Id" with "OBJ" restrictions (as shown above) will limit the "Id" to the specified accounts throughout all centers. If the "Id" also has "FUNC" restrictions, that "Id" will be limited to the specified accounts within the specified centers *only*.

Budget restrictions

It is possible to create a table entry for an ID which restricts the "Id" to remain within budget. The "Id" may create a purchase order or an invoice against any object code in any function as long as the amount of the item is within budget. The "Check Budget" is always checked by *approve* and, in the case where an item exceeds budget, the "Check Budget" field will override all "Category" restrictions. To create this type of restriction, enter table values similar to the following:

Id	Category	Code	Amount	Check Budget	Approval Required	Need Approval by: Id 1	Id 2	Id 3
76764			$ 0.00	Y	N	48353	0	0

A "Check Budget" field of "Y" instructs *approve* to allow ID number 76764 to create purchase orders and invoices against any object in any function as long as the amount applied to that function or object does not exceed its budgeted amount. You can specify a "Category" and "Code" combination (along with the "Check Budget" field set to "Y") to force budget restriction within a specified object or function.

Full restrictions

You can create a table entry for an "Id" which causes any purchase order or invoice created by that id to need approval. The "Approval Required" field is always checked by *approve* and will override any other restriction. This type of entry could be used to simulate a requisition system. The following entry depicts this type of situation:

```
                                    Check   Approval   Need Approval by:
 Id    Category  Code    Amount      Budget  Required    Id 1    Id 2    Id 3
-----  --------  ----  -----------  -------  ---------  ------  ------  ------
13734                  $      0.00     N          Y       48353      0       0
```

An "Approval Required" field of "Y" instructs *approve* to cause all items created by ID number 13734 to need approval. You can specify a "Category" and "Code" combination (along with the "Approval Required" field set to "Y") to force the approval restriction within a specified object or function.

Advanced Approval Examples

The preceding examples display a typical use of each type of restriction. This section contains sample groups of Approval table entries that vary in complexity, to demonstrate a variety of schemes for a purchase order approval process. To create a consistent display format, the following chart of ID numbers will be used for all examples throughout the Examples Section:

Note: Two types of users must be considered in creating a purchase order approval system: the individual who enters the purchase orders and invoices (i.e., a *purch* user), and the person will approve or reject purchase orders and invoices (i.e., an *approve* user).

```
  ID       TYPE OF USER              TITLE
-----     -------------   ----------------------------
11111 --- purch user   -- Head of Art Department.
19991 --- purch user   --    -Art Department Helper

22222 --- purch user   -- Head of Science Department.
29992 --- purch user   --    -Science Department Helper
28882 --- purch user   --    -Chemistry Department Head

55555 --- purch user   -- Accounts Payable Clerk. (Invoices only)

77777 --- purch user   -- Purchasing Director.
77777 --- approve user -- Purchasing Director.
79997 --- approve user --    -Purchasing Helper

99999 --- approve user -- Dean of School.
```

The following samples detail various combinations of Approval table entries. Each sample describes the table entries made for each "Id" as well as the restriction that is caused by each entry. Each sample demonstrates a unique feature of the *approve* program.

Note: Each group of entries represents a unique example. The relationships created within each sample are independent of the other samples, and each sample is more complex and involved that the sample before it. You can combine the concepts shown in each sample to create an unlimited number of possible scenarios.

Sample 1

Id	Category	Code	Amount	Check Budget	Approval Required	Need Approval by: Id 1	Id 2	Id 3
77777	PO		$ 1500.00	N	N	99999	0	0
79997	PO		$ 1500.00	Y	N	99999	0	0
55555	PO		$ 1000.00	N	N	99999	0	0

Sample 1 explanation

Sample 1 demonstrates a single level of approval. The Dean (ID 99999) appears as the only *approve* user. The Purchasing Director (ID 77777), the Purchasing Helper (ID 79997), and the Accounts Payable Clerk (ID 55555) each appear as *purch* users only.

In addition, the Purchasing Director has permission to any object in any function provided a single purchase order does not exceed $1500.00. The Accounts Payable Clerk is also unrestricted by "Category", provided the total of invoices against a single purchase order does not exceed $1000.00. The Purchasing Helper is also unrestricted, provided an item is under $1500.00 and does not exceed its budgeted amount.

Note: Each "Id" has permission to enter invoices (that exceed their individual restrictions) against existing purchase orders, provided the encumbered amount of the existing purchase order has been previously approved.

Sample 2

Id	Category	Code	Amount	Check Budget	Approval Required	Need Approval by: Id 1	Id 2	Id 3
11111	FUNC	1002	$ 0.00	Y	N	77777	0	0
22222	FUNC	1007	$ 0.00	Y	N	77777	0	0
77777	FUNC	1001	$ 0.00	Y	N	99999	0	0
77777	FUNC	1003	$ 0.00	Y	N	99999	0	0
77777	FUNC	1004	$ 0.00	Y	N	99999	0	0
77777	FUNC	1005	$ 0.00	Y	N	99999	0	0
77777	FUNC	1006	$ 0.00	Y	N	99999	0	0
55555	PO		$ 1000.00	N	N	99999	0	0

Sample 2 explanation

The above example shows a single level of approval where more than one individual is responsible for approval. The Purchasing Director (ID 77777) is both an *approve* user and a *purch* user. The Dean (ID 99999) serves as the only other *approve* user. The Head of the Art Department (ID 11111), the Head of the Science Department (ID 22222), and the Accounts Payable Clerk (ID 55555) appear as *purch* users only.

The Head of the Art Department and the Head of the Science Department are both restricted to their specified functions and they must remain within budget. The Purchasing Director is responsible for approving items created by the Department Heads which exceed budgeted amounts.

The Purchasing Director has permission to all other centers (assuming only seven centers) provided he/she remains within budget. The Accounts Payable Clerk is unrestricted by "Category", provided the total of invoices against a single purchase order does not exceed $1000.00. The Dean is responsible for approving items created by the Purchasing Director and the Accounts Payable Clerk which exceed restrictions.

Note: The ID number 77777 appears under both the "Id" column and the "Need Approval by" column. These two entry types are required to allow id number 77777 access to both *purch* and *approve*.

Sample 3

Id	Category	Code	Amount	Check Budget	Approval Required	Need Approval by: Id 1	Id 2	Id 3
11111	FUNC	1002	$ 0.00	Y	N	77777	99999	0
19991	FUNC	1002	$ 0.00	Y	N	11111	77777	99999
19991	FUNC	1002	$ 700.00	N	N	11111	77777	0
19991	OBJ	6100	$ 500.00	N	N	11111	0	0
19991	OBJ	6030	$ 500.00	N	N	11111	0	0

Sample 3 explanation

The above example depicts multiple levels of approval where one or more individuals is responsible for approval depending on the given situation. The Head of the Art Department (ID 11111) is represented as both an *approve* user and a *purch* user. The Purchasing Director (ID 77777) and the Dean (ID 99999) are represented as *approve* users only. The Art Department Helper is represented as a *purch* user only.

The Art Department Helper is restricted to function 1002 up to $700.00. Within this function, he/she is restricted to object 6100 up to $500.00 and to account 6030 up to $500.00. Any item created by the Art Department Helper that does not exceed his/her center restriction, only requires the approval of the Head of the Art Department.

However, any item created by the Art Department Helper that exceeds his/her function restriction, requires the approval of both the Head of the Art Department and the Purchasing Director. Further, any item created by the Art Department Helper that exceeds its budgeted amount also requires approval by the Dean. Any item created by the Head of the Art Department which exceeds its budgeted amount requires approval by the Purchasing Director and the Dean.

Note: The first two entries above effectively cause any item created by the Art Department that exceeds its budgeted amount, to need the approval of the Purchasing Director and the Dean.

Sample 4

Id	Category	Code	Amount	Check Budget	Approval Required	Need Approval by: Id 1	Id 2	Id 3
22222	FUNC	1007	$ 2000.00	N	N	99999	0	0
22222	FUNC	1007	$ 1000.00	N	N	77777	0	0
29992	FUNC	1007	$ 1000.00	N	N	22222	77777	0
28882	FUNC	1007	$ 1000.00	N	N	22222	77777	0
28882	OBJ	6100	$ 500.00	N	N	22222	0	0
28882	OBJ	6030	$ 0.00	N	Y	22222	0	0
28882	OBJ	6050	$ 0.00	N	Y	22222	0	0

Sample 4 explanation

The above example depicts multiple levels of approval where one or more individuals is responsible for approval depending on the given situation. The Head of the Science Department (ID 22222) is both an *approve* user and a *purch* user. The Purchasing Director (ID 77777) and the Dean (ID 99999) appear as *approve* users only. The Science Department Helper (ID 29992) and the Chemistry Department Head (ID 28882) are *purch* users only.

The Science Department Helper is restricted to function 1007 up to $1000.00. Within this function, he/she is restricted to object 6100 up to $500.00, but he/she always requires approval within account 6030 and account 6050. Any item created by the Science Department Helper that does not exceed his/her function restriction, only requires the approval of the Head of the Science Department. Any item created by the Science Department Helper or the Chemistry Department Head which exceeds $1000.00 but does not exceed $2000.00 requires approval by the Head of the Science Department and the Purchasing Director. However, if the item created exceeds $2000.00, it will require approval by the Dean as well.

Note: The first four entries above effectively cause any item created by the Science Department to be categorized in the specified range ($0.00 -- $1000.00 -- $2000.00) with subsequent notification of the appropriate user(s) based on this range.

Sample 5 (an incorrect example)

This section is designed to display representative table entries which should be avoided. A brief description of what is wrong with each entry is given below.

Id	Category	Code	Amount	Check Budget	Approval Required	Need Approval by: Id 1	Id 2	Id 3
11111	PO		$ 1000.00	N	Y	99999	0	0
22222	PO		$ 1000.00	N	N	22222	0	0
33333			$ 0.00	N	Y	0	0	0

Sample 5 explanation

The entry for "Id" number 11111 appears to restrict this "Id" to creating purchase orders which total less that $1000.00. However, the "Approval Required" field is set to "Y", causing all items created by this "Id" to need approval. This entry type effectively overrides the "Category" restrictions and may produce undesired results. This holds true for "Category" restrictions of "FUNC" and "OBJ".

The entry for "Id" number 22222 appears to restrict this "Id" to creating purchase orders which total less that $1000.00. However, any item created by this "Id" that exceeds his/her restrictions, requires approval by the same ID number (22222). This entry type will cause *approve* to exit and should be avoided.

The entry for "Id" number 33333 appears to restrict this "Id" to need approval for any item he/she creates. However, the "Need Approval by" fields do not specify who is responsible for approving items created by this "Id". This entry type will cause *approve* to exit and should be avoided.

Setting Up Direct Deposit Macros

Introduction

The Direct Deposit process consists of two programs: *dirdep* and *ddtp*. These programs require information about your bank accounts to produce accurate direct deposit tapes.

Macros

The following macros relate to *dirdep*. Specifications for the variables inserted on the bank tapes may vary from bank to bank; consult your bank for formatting requirements.

m4_define(‘DIRDEP_INST_BANK_ACCT_NO’.’ ‘)

The account number for direct deposit. The *dirdep* program interprets this macro as the -a parameter and places the value in positions 13 -29 of the type 6 records on the tape. This macro is necessary if the bank requires a debit offset for all credit transactions.

m4_define(‘DIRDEP_BANK_DEST_CODE’.’ ‘)

The destination ACH for direct deposits. The *dirdep* program interprets this macro as both the -d and -m parameters and places the value in positions 4-13 and 41-63 of the type 1 records on the tape.

m4_define(‘DIRDEP_ORG_BANK_CODE’.’ ‘)

The originating ACH for direct deposits. The *dirdep* program interprets this macro as the -o parameter and places the value in positions 63-86 of the type 1 records on the tape.

m4_define(‘DIRDEP_REF_CODE’.’ ‘)

The reference code for direct deposits. The *dirdep* program interprets this macro as the -r parameter and places the value in positions 87-94 of the type 1 records on the tape.

SECTION 32 - FINANCIAL MAINTENANCE PROCEDURES

Overview

Introduction

This section provides you with procedures you need to maintain the financial products.

Definitions

There are two types of maintenance procedures that you must perform in order to keep your database accurate and the CX functioning properly. They are:

Annual
You must perform these procedures annually

Session-based
You must perform these procedures at the beginning, middle or end of a session.

Annual Maintenance Procedures

Introduction

Updating 1099 forms is an important part of your institution's annual maintenance. This section outlines the steps you must take to perform this update.

Procedure

Use the following procedure to prepare for 1099 processing each year.

1. Create a matching entry in the 1099 table for every f1099 code in the subb_rec. The program will perform a lookup in the 1099 table to determine the box in which the amount will be printed.

2. If you have not initially added the invoices with the appropriate f1099 code, you must update them with the applicable 1099 code. For example, if you added the invoices without the f1099 code during the year, you will have to change this at the end of the year to the applicable f1099 code. It is easier to set up a series of f1099 codes at the beginning of the year, and add invoices with these f1099 codes throughout the year.

   Example: You might set aside a "MISC" f1099 code to be used for invoices that should get 1099-Misc. There must be at least one f1099 code for each box that is to be filled in on the 1099. If you are reporting more than one type of miscellaneous income, for example, you might set aside "MIS1" and "MIS2" for the two different types of 1099-Misc.

Note:

- Not every box on the 1099s must have its own f1099 code. You need to set aside terms for only those boxes you will need to fill in the 1099s you print. For example, you will probably not need to have reserve a f1099 code for box 5 of 1099-Misc. This box is used to report fishing boat proceeds to the government.
- Also, in the f1099 table, you can specify a minimum dollar amount per record to inhibit the printing of a form. A form will not be generated if the dollar amount for a particular box on the form are below the specified minimum for that box. However, a 1099 form will be printed and all amounts reported if any one of the amounts in the 1099 record for an individual ID exceed the minimum or are not restricted by a minimum amount. For example, if for Form 1099-MISC the minimum amount to be reported in Box 7 is $600.00, enter $600.00 in the 1099 table for MISC, Box 7. This will restrict a form from being generated for an individual if the Box 7 amount accumulated in the 1099 record is below $600.00 and is the only box amount to be printed on the form. On the other hand, if the Box 7 amount is below $600.00, but another Box amount exceeds (or is not regulated by) a minimum amount, a form will be generated and the Box 7 amount will be printed as well, regardless of the minimum specified.

Macros in 1099 Processing

Introduction

Processing 1099-I (interest) and 1099-M (miscellaneous) forms uses macros in $CARSPATH/macros/custom/f1099. Processing 1099-R forms uses macros in $CARSPATH/macros/custom/r1099. This section describes these macros and provides information to help you complete the macro files.

Macros in $CARSPATH/macros/custom/f1099

The following macros are located in the file $CARSPATH/macros/custom/f1099. The macros are listed in the order in which they appear in the macro file.

m4_define(`F1099_PRIOR_YR', `')
A “P” if you are reporting prior year information; otherwise, a blank.

m4_define(`F1099_TCC_CODE', `99999')
The five-character alphanumeric Transmitter Control code (TCC) assigned by the IRS. You must have a TCC to file your 1099-I and 1099-M data.

m4_define(`F1099_TRANS_REPLACEMENT_CODE', `')
Required for replacement files only, the alphanumeric character which appears immediately after the TCC number on the Media Tracking Slip (Form 9267). The form 9267 accompanies media that has been returned by IRS/MCC because of procesing problems. *This field must be blank unless the original submission has been returned*. If the file is being replaced magnetically, or if the file was originally sent magnetically, but the replacement is being sent electronically, the information is required in this field.

m4_define(`F1099_TEST_CODE', `')
A “T” if the file is a test file; otherwise, a blank.

m4_define(`F1099_TRANS_FOREIGN_ENTITY', `')
A “1” if your institution is a foreign entity; otherwise a blank.

m4_define(`F1099_TRANSMITTER_NAME_1', `EMPR_NAME’)
The name of the institution as used in normal business. This value cannot exceed 40 characters. If the value you have assigned to EMPR_NAME exceeds 40 characters, the excess characters will be truncated. If desired, you can use the T1098_TRANSMITTER_NAME_2 macro for characters in excess of 40 characters.

m4_define(`F1099_TRANSMITTER_NAME_2', `')
If required, any additional information that may be part of the institution’s name. This value cannot exceed 40 characters.

m4_define(`F1099_COMPANY_NAME_1', `EMPR_NAME’)
The name of the location to which correspondence should be addressed, or to which media should be returned as a result of processing problems. This value cannot exceed 40 characters.

m4_define(`F1099_COMPANY_NAME_2', `EMPR_NAME’)
If required, any additional information that may be part of the location name (i.e., the place to which correspondence should be addressed, or to which media should be returned as a result of processing problems). This value cannot exceed 40 characters.

m4_define(`F1099_COMPANY_MAILING_ADDR', `EMPR_ADDR1’)
The address to which correspondence should be sent or media should be returned in the event the IRS is unable to process the submission. This value cannot exceed 40 characters.

m4_define(`F1099_COMPANY_CITY', `EMPR_CITY')
The city, town, or post office to which correspondence should be sent or media should be returned in the event the IRS is unable to process the submission. This value cannot exceed 40 characters.

m4_define(`F1099_COMPANY_ST',`EMPR_ST')
The U.S. Postal Service state abbreviation for the state.

m4_define(`F1099_COMPANY_ZIP', `EMPR_ZIP')
The nine-digit zip code assigned by the U.S. Postal Service. If you do not know the 9-digit code, you can define this value with the 5-digit code.

m4_define(`F1099_CONTACT_NAME', `EMPR_NAME')
The name of the person to contact if the IRS encounters problems with the file or your transmission. This value cannot exceed 40 characters.

m4_define(`F1099_CONTACT_PHONE_EXT', `999-999-9999-99999')
The telephone number and extension of the contact person. Format the number as area code-exchange-line number-extension. If no extension exists, you can omit the last hyphen and the last five digits. This value cannot exceed 15 digits (excluding hyphens).

Example: Enter the number 304-263-8700 extension 52345 as 304-263-8700-52345. Enter the number 304-263-8700 without an extension as 304-263-8700.

m4_define(`F1099_MAGNETIC_TAPE', `LS')
The letters “LS” if you are filing using magnetic tape; otherwise, blank.

m4_define(`F1099_ELECTRONIC_FILE_NAME', `')
Either the original filename or the name of the correction file as assigned by the IRP-BBS (e.g., 12345p01.DAT). Do not enter the replacement file name. If the submission is not an original or correction file, leave the field blank.

m4_define(`F1099_LAST_FILING_YR', `')
The number “1” if this is the last year the payer will file. Set this indicator if the payer will not file information returns under this payer name and TIN in the future (either magnetically, electronicallly, or on paper). Otherwise, leave the field blank.

m4_define(`F1099_ORIGINAL_CODE, `1')
A “1” if the information is original data; otherwise, blank.

m4_define(`F1099_PAYER_REPLACEMENT_CODE, `')
A “1” if the file replaces a previous file that IRS/MCC returned to the transmitter because of errors in processing; otherwise, blank.

m4_define(`F1099_CORRECTION_CODE, `')
A “1” if the file contrains corrected information that was previously submitted and processed by IRS/MCC, but contained errors; otherwise, blank.

Note: If you want to submit new, previously omitted information, it must not be added into a correction file, but must be submitted as original.

m4_define(`F1099_PAYER_FOREIGN_ENTITY', `')
A “1” if the payer is a foreign entity and income is paid by the foreign entity to a U.S. resident; otherwise, blank.

m4_define(`F1099_FIRST_PAYER_NAME', `EMPR_NAME')
The name of the payer. This value cannot exceed 40 characters.

m4_define(`F1099_SECOND_PAYER_NAME', `EMPR_NAME')
Contents based on the value you set for the macro F1099_TRANSFER_AGENT below. If that macro contains a value of “0”, this macro defines a continuation of the name of the payer; if that macro contains a value of “1”, this macro defines the name of the transfer (or paying) agent.

m4_define(`F1099_TRANSFER_AGENT', `0')
A “1” if the entity defined in F1099_SECOND_PAYER_NAME above is a continuation of the payer name; a “0” (zero) if the entity defined in F1099_SECOND_PAYER_NAME above is the transfer (or paying) agent.

m4_define(`F1099_PAYER_SHIPPING_ADDR', `EMPR_ADDR1')
Street address of the transfer (or paying) agent (if F1099_TRANSFER_AGENT is set to “0”), or the street address of the payer (if F1099_TRANSFER_AGENT is set to “1”.)

m4_define(`F1099_PAYER_CITY', `EMPR_CITY')
City of the transfer (or paying) agent (if F1099_TRANSFER_AGENT is set to “0”), or the city of the payer (if F1099_TRANSFER_AGENT is set to “1”.)

m4_define(`F1099_PAYER_ST', `EMPR_ST')
State of the transfer (or paying) agent (if F1099_TRANSFER_AGENT is set to “0”), or the state of the payer (if F1099_TRANSFER_AGENT is set to “1”.)

m4_define(`F1099_PAYER_ZIP', `EMPR_ZIP')
Zip code of the transfer (or paying) agent (if F1099_TRANSFER_AGENT is set to “0”), or the zip code of the payer (if F1099_TRANSFER_AGENT is set to “1”.)

m4_define(`F1099_PAYER_PHONE_EXT', `999-999-9999-99999')
The telephone number and extension of the payer. Format the number as area code-exchange-line number-extension. If no extension exists, you can omit the last hyphen and the last five digits. This value cannot exceed 15 digits (excluding hyphens).

Example: Enter the number 304-263-8700 extension 52345 as 304-263-8700-52345. Enter the number 304-263-8700 without an extension as 304-263-8700.

m4_define(`F1099_PAYER_OFFICE_CODE', `')
The four-character office code of the payer; otherwise, blank. For payers with multiple locations, this field may be used to identify the location of the office submitting the return.

Macros in $CARSPATH/macros/custom/r1099

The following macros are located in the file $CARSPATH/macros/custom/r1099. The macros are listed in the order in which they appear in the macro file.

m4_define(`R1099_PRIOR_YR', `')
A “P” if you are reporting prior year information; otherwise, a blank.

m4_define(`R1099_TCC_CODE', `99999')
The five-character alphanumeric Transmitter Control code (TCC) assigned by the IRS/MCC. You must have a TCC to file your 1099-R data.

m4_define(`R1099_TRANS_REPLACEMENT_CODE', `')
Required for replacement files only, the alphanumeric character which appears immediately after the TCC number on the Media Tracking Slip (Form 9267). The form 9267 accompanies media that has been returned by IRS/MCC because of procesing problems. *This field must be blank unless the original submission has been returned*. If the file is being replaced magnetically, or if the file was originally sent magnetically, but the replacement is being sent electronically, the information is required in this field.

m4_define(`R1099_TEST_CODE', `')
A “T” if the file is a test file; otherwise, a blank.

m4_define(`R1099_TRANS_FOREIGN_ENTITY', `')
A “1” if your institution is a foreign entity; otherwise a blank.

m4_define(`R1099_TRANSMITTER_NAME_1', `EMPR_NAME’)
The name of the institution as used in normal business. This value cannot exceed 40 characters. If the value you have assigned to EMPR_NAME exceeds 40 characters, the excess characters will be truncated. If desired, you can use the R1099_TRANSMITTER_NAME_2 macro for characters in excess of 40 characters.

m4_define(`R1099_TRANSMITTER_NAME_2', `')
If required, any additional information that may be part of the institution’s name. This value cannot exceed 40 characters.

m4_define(`R1099_COMPANY_NAME_1', `EMPR_NAME’)
The name of the location to which correspondence should be addressed, or to which media should be returned as a result of processing problems. This value cannot exceed 40 characters.

m4_define(`R1099_COMPANY_NAME_2', `EMPR_NAME’)
If required, any additional information that may be part of the location name (i.e., the place to which correspondence should be addressed, or to which media should be returned as a result of processing problems). This value cannot exceed 40 characters.

m4_define(`R1099_COMPANY_MAILING_ADDR', `EMPR_ADDR1’)
The address to which correspondence should be sent or media should be returned in the event the IRS is unable to process the submission. This value cannot exceed 40 characters.

m4_define(`R1099_COMPANY_CITY', `EMPR_CITY')
The city, town, or post office to which correspondence should be sent or media should be returned in the event the IRS is unable to process the submission. This value cannot exceed 40 characters.

m4_define(`R1099_COMPANY_ST',`EMPR_ST')
The U.S. Postal Service state abbreviation for the state.

m4_define(`R1099_COMPANY_ZIP', `EMPR_ZIP')
The nine-digit zip code assigned by the U.S. Postal Service. If you do not know the 9-digit code, you can define this value with the 5-digit code.

m4_define(`R1099_CONTACT_NAME', `EMPR_NAME')
The name of the person to contact if the IRS encounters problems with the file or your transmission. This value cannot exceed 40 characters.

m4_define(`R1099_CONTACT_PHONE_EXT', `999-999-9999-99999')
The telephone number and extension of the contact person. Format the number as area code-exchange-line number-extension. If no extension exists, you can omit the last hyphen and the last five digits. This value cannot exceed 15 digits (excluding hyphens).

Example: Enter the number 304-263-8700 extension 52345 as 304-263-8700-52345. Enter the number 304-263-8700 without an extension as 304-263-8700.

m4_define(`R1099_MAGNETIC_TAPE', `LS')
The letters “LS” if you are filing using magnetic tape; otherwise, blank.

m4_define(`R1099_ELECTRONIC_FILE_NAME', `')
Either the original filename or the name of the correction file as assigned by the IRP-BBS (e.g., 12345p01.DAT). Do not enter the replacement file name. If the submission is not an original or correction file, leave the field blank.

m4_define(`R1099_LAST_FILING_YR', `')
The number “1” if this is the last year the payer will file. Set this indicator if the payer will not file information returns under this payer name and TIN in the future (either magnetically, electronicallly, or on paper). Otherwise, leave the field blank.

m4_define(`R1099_ORIGINAL_CODE', `1')
A “1” if the information is original data; otherwise, blank.

m4_define(`R1099_PAYER_REPLACEMENT_CODE', `')
A “1” if the file replaces a previous file that IRS/MCC returned to the transmitter because of errors in processing; otherwise, blank.

m4_define(`R1099_CORRECTION_CODE, `')
A “1” if the file contrains corrected information that was previously submitted and processed by IRS/MCC, but contained errors; otherwise, blank.

> **Note:** If you want to submit new, previously omitted information, it must not be added into a correction file, but must be submitted as original.

m4_define(`R1099_PAYER_FOREIGN_ENTITY', `')
A “1” if the payer is a foreign entity and income is paid by the foreign entity to a U.S. resident; otherwise, blank.

m4_define(`R1099_FIRST_PAYER_NAME', `EMPR_NAME')
The name of the payer. This value cannot exceed 40 characters.

m4_define(`R1099_SECOND_PAYER_NAME', `EMPR_NAME')
Contents based on the value you set for the macro R1099_TRANSFER_AGENT below. If that macro contains a value of “0”, this macro defines a continuation of the name of the payer; if that macro contains a value of “1”, this macro defines the name of the transfer (or paying) agent.

m4_define(`R1099_TRANSFER_AGENT', `0')
A “1” if the entity defined in R1099_SECOND_PAYER_NAME above is a continuation of the payer name; a “0” (zero) if the entity defined in R1099_SECOND_PAYER_NAME above is the transfer (or paying) agent.

m4_define(`R1099_PAYER_SHIPPING_ADDR', `EMPR_ADDR1')
Street address of the transfer (or paying) agent (if R1099_TRANSFER_AGENT is set to “0”), or the street address of the payer (if R1099_TRANSFER_AGENT is set to “1”.)

m4_define(`R1099_PAYER_CITY', `EMPR_CITY')
City of the transfer (or paying) agent (if R1099_TRANSFER_AGENT is set to “0”), or the city of the payer (if R1099_TRANSFER_AGENT is set to “1”.)

m4_define(`R1099_PAYER_ST', `EMPR_ST')
State of the transfer (or paying) agent (if R1099_TRANSFER_AGENT is set to “0”), or the state of the payer (if R1099_TRANSFER_AGENT is set to “1”.)

m4_define(`R1099_PAYER_ZIP', `EMPR_ZIP')
Zip code of the transfer (or paying) agent (if R1099_TRANSFER_AGENT is set to “0”), or the zip code of the payer (if R1099_TRANSFER_AGENT is set to “1”.)

m4_define(`R1099_PAYER_PHONE_EXT', `999-999-9999-99999')

The telephone number and extension of the payer. Format the number as area code-exchange-line number-extension. If no extension exists, you can omit the last hyphen and the last five digits. This value cannot exceed 15 digits (excluding hyphens).

Example: Enter the number 304-263-8700 extension 52345 as 304-263-8700-52345. Enter the number 304-263-8700 without an extension as 304-263-8700.

m4_define(`R1099_PAYER_OFFICE_CODE', `')

The four-character office code of the payer; otherwise, blank. For payers with multiple locations, this field may be used to identify the location of the office submitting the return.

In addition, the following enable macros exist in the $CARSPATH/macros/custom/r1099 file:

m4_define('ENABLE_R1099_NAME_SORT', 'N')
m4_define('ENABLE_R1099_SSN_SORT', 'N')
m4_define('ENABLE_R1099_ZIP_SORT', 'N')
m4_define('ENABLE_R1099_ZIP_NAME_SORT', 'Y')

The means to control the sorting of 1099-R output. You can sort by name, social security number, zip code, or by a combination of zip code and name. As delivered, the last of these four macros is set to Y, and the others are set to N, causing the output to sort first by zip code, then by name (i.e., by name within zip code). Only one of these macros can be set to Y; if none is set to Y, the default sort criteria is the ID. Includes stored in $CARSPATH/include/custom/r1099 work with these macros and should not be changed.

SECTION 33 - PROGRAM ERRORS AND CRASH RECOVERY

Overview

Introduction

This section provides you with the following:

- A list of serious and fatal errors
- Crash recovery procedures

Note: Refer to the following manuals for a list of the more common status, field error, and warning messages that can occur when menu users execute the programs in the financial products.

- *Using Financial Budgeting*
- *Using Fixed Assets*
- *Using Cashier*

Error and Crash Recovery Procedures

Introduction

The procedures to recover from a crash are organized by the seriousness of the error.

Core Dump Recovery

The following procedure describes the steps to recover from a core dump of a financial program.

1. Run *vchrecover* to reset the status of the journal.
2. Access the program screens directory for the program.

   Example: cd/ $CARSPATH/modules/accounting/progscr
3. Reinstall each program screen file.

   Example: make reinstall F=<filename>

   Note: You can also reinstall all of the screens by entering the following at the UNIX prompt:
   make reinstall F=all
4. Attempt to execute the program. Did the reinstall of the program screens fix the error?
   - If yes, you are done.
   - If no, go to step 5.
5. Access the source code directory of the program.

   Example: cd/$CARSPATH/src/accounting/cashier
6. Reinstall the source code for the program.

   Example: make reinstall
7. Attempt to execute the program. Did the reinstall of the program source code fix the error?
   - If yes, you are done.
   - If no, go to step 8.
8. In the source code for the program, delete the old compiled code for the program.

   Example: make cleanup

9. Reinstall the program source code.

   Example: make reinstall

10. Attempt to execute the program. Did the deletion of the old code and reinstallation of the program source code fix the error?
    - If yes, you are done.
    - If no, go to step 11.

11. Review the libraries for the program. In the source code for the program, review the file, Makefile. In the file, search for the parameter, ADDLIBS, which identifies the libraries that you must reinstall.

    Example: % vi Makefile

    /ADDLIBS

12. Reinstall the libraries for the program and reinstall the source for the program (e.g., *cashier*) by entering the following at the UNIX prompt:

    cd <to appropriate library>
    make cleanup reinstall
    cd $CARSPATH/src/financial/cashier
    make reinstall

    Note: You must reinstall the source program to include any library changes.

13. Attempt to execute the entry program. Did the reinstallation of the libraries for the entry program fix the error?
    - If yes, you are done.
    - If no, call Jenzabar Support Services.

Processing Errors

Errors From the *purch* Program

The following errors can occur when using *purch*:

A Form cannot be specified without a printer
When *purch* is first loaded, the menu option supplies many default options. The *purch* program prints this message if the -f (form) command line option was used without the -o (output device) option. You cannot provide the *purch* program with a form to use without also providing it an output device. If this error occurs, *purch* will go to the Initialization screen and force you to enter both the printer and the form to be used.

A Negative Amount Cannot Be Entered
When you are adding charges to a purchase order, *purch* will not allow you to enter a negative value. You must always enter a positive value.

A documentation station number must be selected before an add
The *purch* program requires that each purchase order added to the system be controlled by a specific station number. This gives *purch* complete control over the purchase order numbers that will be printed. Before you can add a purchase order, you must therefore indicate your station number to *purch*. Once you have done so, *purch* will use that station number for all your purchase orders.

A form has not been Initialized

The *purch* program prints this error when you attempt to use the Output, Form, or Write commands and have not yet specified a form to use. When this happens, go to the Initialization screen and enter the correct form.

Aborted
Add Aborted
Complete Aborted
Dufindid Aborted
Erase Aborted
Query Aborted
Terminate Aborted
Update Aborted

These status messages appear when you interrupt a *purch* function.

Account is valid though it does not exist. Do you wish to use it (Y, N)

If the account you enter on the charge screen does not exist, *purch* will check it to make sure it follows the general ledger rules as set forth by the gld_rec and the accounting tables. If it is a valid account, you will have the option of using it. Make sure the accounts you entered is correct, then press **Y** if you want to use it. If you do not want to use it, press **N** and *purch* will allow you to correct the account number.

Invalid FUND according to the Fund Table
Function not found in the Function table:Code
Object not found in the Object table:Code
Invalid subfund according to the Subfund Table

The *purch* program looks at the fund, function, object, and subfund tables to validate the parts of the account number. If the fund, function, object, or subfund is not in the corresponding table, the account number is invalid. The *purch* program will not allow you to enter that account. You must correct the fund, function, object, or subfund code you entered or correct the table.

Add Invoice Amounts against departments

This message displays as a prompt for you to begin entering the amounts on the invoices.

Amount type must be 'Type' for the purchasing document

The *purch* program encumbers departments for the amounts on the purchase order. This message displays when the document station number you entered does not allow encumbering. If the document chosen does not allow encumbering, you cannot add a purchase order using this station. Change to a station that does allow purchase orders.

An Amount Must be Specified

When you are adding charges to a purchase order, you must enter a value for each account.

An invoice amount has been specified against this charge

You cannot delete a charge that has an invoice amount associated with it. Once an invoice amount has been specified, the department has been charged for the invoice. You must first delete the invoice amount against this account, and then delete the charge.

An invoice amount is required in an add

When you add an invoice, you must also specify an amount for the accounts affected by the invoice.

BIND:fieldname on screen name:Message

The *purch* program uses the CX screen package to display and accept input. To use a screen, it has to assign its internal variables to the proper locations on the screen. If *purch* encounters an error when it attempts to do this, it prints the above error message, where fieldname is the name of the screen field it is attempting to bind, screen name is the name of the screen it is attempting to use, and *message* is the screen package error message. This message usually occurs when you have revised the purchase order form and accidentally modified or deleted a field *purch* is expecting to use. If you do not want something printed on your purchase order form, specify that it is optional in the attributes section.

Bgv_output error:

This message occurs when *purch* tries to create an output file to be used in later posting of the general and subsidiary ledger entries and transactions. It will be followed by one or two error lines from PTP that further describe the problem. This error results from a permission problem on the output file directories. Make sure there is a Filepost/binary directory in the standard $CARSPATH/vchpost voucher transaction file directory. Make sure the person trying to run *purch* has permission to write to the posting directory.

Bgvoucher connection error:

When *purch* is run with the *bgvoucher* Verify or Post options, it will attempt to load and run *bgvoucher* during program initialization. If it cannot establish the connection, the above message appears, along with the PTP error that caused the connection attempt to fail. This message should normally not occur. If it does occur, check BIN_PATH and make sure *bgvoucher* exists.

Bgvoucher load error:

When *purch* is run with the *bgvoucher* Verify or Post options, it will start *bgvoucher* during initialization. The first time you execute a command that might result in general ledger entries, *purch* will check to make sure *bgvoucher* loaded correctly. If *bgvoucher* has encountered an error, it will send *purch* a short description of the problem, which *purch* will display. The *purch* program will then exit. A *bgvoucher* initialization error is usually caused by *vchrecover* being run at the same time.

Loading bgvoucher
Opening Database
Opening Files
Initializing Variables
Verifying Arguments
Binding Screens
Loading Tables

The *purch* program prints several status messages while it is initializing. On a slow system, it may be helpful to know where *purch* is in its initialization process.

Can only delete charges which have not been written
You cannot delete charges after you have written them.
Can't find gla_rec: fund-center-account-project-unit. Status:nnnn
Can't find glamt_rec: fscl yr-amt type-fund-center-acct-project-unit. Stat:nnnn

As part of the process for verifying an account number, *purch* attempts to find the gla_rec and glamt_rec for this account in the database. If *purch* gets a status from the dbfind call that it cannot interpret, it will print this message.

Cannot Complete the Purchase Order until a 'W'rite or 'E'rase is executed
Cannot Exit Program until a 'W'rite or 'E'rase is Executed
Cannot Initialize Parameters until a 'W'rite or 'E'rase is executed
Cannot Query on a Purchase Order until a 'W'rite or 'E'rase is executed
Cannot Terminate the Purchase Order until a 'W'rite or 'E'rase is executed
Cannot add a new purchase order until a 'W'rite or 'E'rase is executed
Cannot change the Station number until a 'W'rite or 'E'rase is executed
Cannot do a 'N'ext until a 'W'rite or 'E'rase in executed (sic)
Cannot do a 'P'revious until a 'W'rite or 'E'rase in executed (sic)

These messages are printed when you have updated the purchase order but have not told *purch* what to do with the changes. Your command would either change the status of the purchase order or change to a new purchase order. In either case, *purch* cannot act on your command until you have used the Write command for your changes to the database or you have canceled your modifications with Erase.

Cannot Delete a Charge without Purchasing Permission

You must have Purchasing permission to be able to affect any purchase order with encumbered amounts.

Cannot Look Up an Id From This Field

The *purch* program prints this message when you attempt to do an id lookup from a field that is not an id field. You can only look up an id from the vendor id, payee id, or responsible id fields on the purchase order screen.

Cannot Terminate a PO with Invoices without Accounts Payable Permission

You cannot add, update, or delete invoices without Accounts Payable permission. Terminating a purchase order will automatically delete any associated invoices. If you try to terminate a purchase order and there are invoices with amounts, *purch* will not allow you to terminate the purchase order.

Cannot Terminate a PO with encumbered amounts without purchasing permission

You must have Purchasing permission to be able to terminate any purchase order with encumbered amounts.

Cannot Terminate a purchase order where an invoice has been paid

Once a payment has been made to a vendor, you cannot terminate the purchase order. If for some reason you must terminate the purchase order, you must void the check used to pay the invoice, and only then terminate the purchase order.

Cannot Update:Message

This message is printed if the screen package returns an error when you tried to update the purchase order information. Message will contain additional information about the error that may assist you in tracking down the error. This error should not normally occur. Make sure the screens for *purch* are properly installed in $SCRPATH/purchasing/PURCH.

Cannot add invoices without accounts payable permission
Cannot delete invoices without accounts payable permission
Cannot update invoices without accounts payable permission

Without Accounts Payable permission, you can view the invoices and the invoice detail but you cannot add, update, or delete anything associated the invoice or invoice detail.

Cannot delete invoices which have been paid or selected for payment

Once an invoice has been paid, or selected for payment, you cannot delete or update that invoice. The only way you can change this invoice is by voiding the check with which you paid the invoice.

Cannot erase after a 'T'erminate

Once you have Terminated a purchase order, you cannot change the purchase order.

Cannot find doc_table entry with 'nn' station number. Status:'nnnn'

The *purch* program prints this message if you have entered an invalid station number for this document type. You must enter a document station number that is in the Document table for this document code.

Cannot find due to from acct in gld_rec. Status:nnnn

You must have the due to/from account defined in your gld_rec. As part of its account checking, *purch* does not allow you to add charges to accounts in the due to/from series. If *purch* cannot find this gld_rec when it is loading, it will exit with a fatal error.

Cannot find fund 'nn ', cntr 'nnnn', proj 'nnnn', unit 'nnnn', status %d

The *purch* program prints this error in the review option when there is no data in the gl_amt_rec.

Cannot find gle_rec. 'Vch Ref'-'Vch Number'-'Entry Number'. Status=nnnn

This error occurs when *purch* has found the subsidiary information for the purchase order but was unable to find the general ledger information. The gle_rec may have a corrupt index.

Cannot find id 'nnnn' in id_rec. Status:nnnn

The *purch* program looks in the id file to find the name and other information about the vendor. If it cannot find the id it is looking for, and receives an unexpected status from the *dbfind* operation, it will print the above message and quit. If the id is not in the id file, *purch* will print an error message and continue operation.

Cannot find id 'nnnnn' in suba_rec. Status:nnnn

At various points in the purchasing process, *purch* must find and verify the suba_rec for vendors or payees. If it cannot find a suba_rec and receives a status it does not understand, it will print the above message.

Cannot find po_rec. 'PO'-'nnnnn'. Status=xxxx

When you query on an invoice, *purch* reads the subsidiary records that matches your query and then attempts to find the po_rec that goes with the subsidiary information it found. If it cannot find the po_rec, it prints the above message, where PO is the document code of the purchase order, nnnnn is the number of the purchase order, and xxxx is the status returned by the Informix applications language library. This is usually caused by using an AP voucher to add an invoice. The *purch* program cannot handle invoices it did not add. Use an AP voucher to maintain any invoices you added with voucher, and use *purch* to maintain any purchase orders and invoices you originally added with purch.

Cannot find subb_rec. 'subs'-'subs no'-'bal code'-'bal period'. Stat:nnnn

When *purch* is loading the invoice detail or updating subsidiary information in the database, it looks for the invoices associated with the purchase order. If it cannot find an invoice that (according to the supporting detail) should be in the database, *purch* will exit with a fatal error.

Cannot handle a subsidiary control account

The *purch* program does not allow any departmental subsidiary control accounts. If it prints this message, you have tried to charge a purchase to an account that is a subsidiary control account. Change the account number to a different number and try it again.

Cannot insert a line. This would move the current last line off the form.

You can update, add, or delete information from the purchase order that is to be printed. You cannot, however, add more information to a purchase order than will fit on one printed form. You must add another purchase order for the line items you were not able to add to the first purchase order.

Cannot load bgvoucher.

When *purch* is told to use *bgvoucher* with the -m P or -m V options, the first thing it does during the program loading process is try to start *bgvoucher*. If it gets an error, it will print *bgvoucher's* error message and the above message, and then *purch* will quit.

Cannot print a po that is being 'A'dded.

When a purchase order is being added, it will be output automatically when you select the Write command for the purchase order. You do not have to output a purchase order you are adding.

Cannot query on invoices. Selfield on 'key name' failed. Status=nnnn
Cannot select 'key name' in 'file name', dbselfield status nnnn.

When it was accessing and manipulating the database files, *purch* could not select the key name key. If this error occurs, check the database files and make sure the key in the error message has been built with the schema. If it has not been built, you must add the index and rebuild the file. This error should never occur because the applicable indexes should have been built during your implementation.

Cannot select a BILL MODE document without accounts payable permission

Bill Mode is a shorthand method of adding invoices for continuing services, or for products you do not normally print purchase orders for. Accounts payable mode is used when you will be dealing with invoices and paying invoices. Adding a bill to the system implies that you are going to be working with invoices.

Cannot select a true purchase order without purchasing permission

To add a purchase order to the system, you must have purchasing permission. The *purch* program does not allow you to select a purchase order document station without purchasing permissions. If you have neither accounts payable permission or purchasing permission, you will not be allowed to select a document station, and you will not be able to add, update, or delete purchase orders or invoices.

Cannot start a AP voucher:
Cannot start a PC voucher:

The *purch* program prints these messages when it cannot start an Accounts Payable or Purchasing voucher with *bgvoucher*. This message will be followed by the *bgvoucher* error message.

Cannot update the total encumbrance to be less than the total actual

An encumbered amount could be considered to be the amount you expect to pay to an account. An actual amount is the amount for which you have been charged on an invoice. The *purch* cannot accept that a department expects to pay less than the amount it was actually charged.

Cannot update 'file name' (serial no), dbupdate status nnnn.

The *purch* program prints this message when it cannot update a record in the specified file. The message gives the serial number of the record that could not be updated, and the status the dbupdate Informix call returned.

Cannot use the Output command in batch printing

When you are printing forms in Batch mode, your forms are going to a printer that is assumed to contain pre-numbered purchase order forms. You cannot Output a purchase order that would disrupt the normal number sequence on the purchase order forms.

Cannot work with form 'form type A', form type 'form type B' is loaded

You cannot work with purchase orders with a printed form that is not the same as the form you currently have loaded. You must change your station number and form type to be able to query on and modify purchase orders with form type B.

Changes Erased

This message is printed when *purch* successfully erases the changes you have made to a purchase order you are adding or updating.

Clearing Buffers

The *purch* program prints this status message when it is reinitializing during the Query command. Command was sent to function name() that was not understood.

Complete the Purchase Order (Y or N)

When you complete a purchase order, *purch* removes the remaining encumbrances and changes the purchase order status to C. This should only be done after all the invoices for the purchase order have been received. To avoid accidentally selecting the wrong command and completing a purchase order by mistake, *purch* verifies that you do really want to complete the purchase order before it continues with the Complete process.

Completing Purchase Order
Printing
Terminating Purchase Order
Writing

Since the complete, output, or write commands can involve several seconds of processing, *purch* prints out a status message to let you know that it has started the completion or termination process.

Could not add purchase order body record. 'code-number'. Line n.
Status nnnn

Each individual line item in the purchase order form is stored in a separate database record. When you add or update a line, it must be added or updated in the database. The *purch* program prints the above message when it tries to add it to the database and the add fails.

Could not add subb record 'subs'-'subs number'-'bal code'-'bal prd'.
Status=nnnn

The *purch* program stores invoices in the subsidiary balance file. If it cannot add an invoice, it will get a fatal error. If the status it reports is 6017, the most probable cause of this error is that the suba_cust_serial is incorrect. You can verify this by checking the last subsidiary balance record against the suba_cust_serial for that vendor. If the suba_cust_serial is correct, the subb_code should be equal to the suba_cust_serial. If they are not equal, you should try to determine how the suba_cust_serial was disrupted.

Could not add the purchase order record 'doc code'-'nnnnnn'.
Status=nnnn

All purchase orders *purch* adds must have a unique document code and purchase order number. The most common reason for the failure to add a purchase order is if you manually entered a purchase order number and code that duplicated a purchase order that already exists in the system. It could also happen if the document numbers overlapped with purchase orders printed from another station.

Could not add the suba record 'subs'-'subs no'. Status=nnnn

All vendors in the purchasing process must have subsidiary accounts. The subsidiary account is used for the invoice serial number and other information. If *purch* cannot add a suba_rec, the most likely explanation is that the suba_rec has a corrupted index.

Could not close ap voucher connection, trying again...
Second try also failed. Problem in ap bgvoucher communications.

Could not close pc voucher connection, trying again...

Second try also failed. Problem in pc bgvoucher communications.

If *purch* is using *bgvoucher*, it attempts to finish any vouchers it has started and tries to close the connections it has established to *bgvoucher* before it exits. If *bgvoucher* will not allow it to close the connection, there is probably a problem with *purch-bgvoucher* communications. If there are errors on the first attempt, purch will print the error messages it gets from *bgvoucher* and try to close the connection again. If it cannot do so, it will quit without closing the ptp connection to *bgvoucher*. You will probably receive mail from *bgvoucher* as well as *purch*.

Could not continue ap vch after verification.
Could not continue pc vch after verification.

One reason for using *bgvoucher* in *purch* is to be able to post both the ap and pc vouchers simultaneously. To do this, *purch* sends the general ledger information to *bgvoucher* to be verified. When *bgvoucher* has verified both general ledger entries, *purch* tells *bgvoucher* to post both entries. This error message will result if *purch* was unable to instruct *bgvoucher* to post the entries after they were verified. There is probably a problem with the *purch-bgvoucher* ptp link.

Could not delete purchase order body record. serial number 'serial'.
Status nnnn

Each individual line item in the purchase order form is stored in a separate database record. When you delete a line, it must be deleted from the database. The *purch* program prints the above message when the delete fails.

Could not find fscl cal record for 'subsidiary' 'amt type' 'date'. Status:nnnn
Could not find fscl cal record for 'subsidiary' 'amt type' 'date'.

These messages appear if *purch* cannot find a Fiscal Calendar record for this subsidiary, amount type, and date combination. Make sure your Fiscal Calendar records contain appropriate entries for your Accounts Payable subsidiary and your general ledger. The Accounts Payable subsidiary must include entries for amount type ENC and ACT for the date you give, and your general ledger entries must include entries for ENC, ACT, and BGT for the program date.

Could not find purchase order body record 'code'-number. Status nnnn

Each individual line item in the purchase order form is stored in a separate database record. When you initially view the purchase order, *purch* loads the line items on the form from the database. The *purch* program prints the above message when it gets an error on the database read.

Could not find purchase order body record serial number 'serial'.
Status nnnn

Each individual line item in the purchase order form is stored in a separate database record. When you display the purchase order, *purch* reads the purchase order line items from the database. If you modify a line item, *purch* must update it in the database. If it cannot find the line item you updated in the database, it will print the above message.

Could not find subb record 'subs'-'subs no'-'bal code'-'bal prd' to update.
Status=nnnn

When you are updating a purchase order or its invoices, *purch* reads the subsidiary balance records at the time you load the purchase order detail. Later, when you Write your updated information, *purch* goes back to the subsidiary balance records and updates any amounts that have changed. This error occurs when *purch* has previously located a subsidiary balance record in the database but cannot find the record to update it. This error probably shows that you have an index problem in the subsidiary balance records.

Could not find sube_rec:'subs'-<subs number>. Status=nnnn

This error occurs if *purch* gets an unexpected error status from its *dbfinds* on the subsidiary entry records.

Could not find the comparison purchase order record 'doc code'-'po number'. Status=nnnn

The *purch* program prints this error if it cannot find the purchase order it is currently working with in the database.

Could not find the document table to update 'document'-'station'. Status=nnnn

When a user selects a document station in *purch*, *purch* reserves that document station for you. When you exit the program, when you change stations, or when you print a document, *purch* tries to find the entry in the Document table so it can update it appropriately. If it cannot find the station number in the Document table it will print this message. This error should not occur because the Document table entry *purch* tries to retrieve should always exist.

Could not find the purchase order record 'doc code'-'po number' to update. Status=nnnn

When you query on a purchase order, *purch* reads the purchase order from the database. If you update the purchase order, *purch* will try to update the purchase order. If it cannot find it in order to update it, *purch* will print this message. This error should not occur because the purchase order record *purch* tries to retrieve should always exist.

Could not find the suba record 'subs'-'subs no' to update. Status=nnnn

When *purch* reads the subsidiary information for a vendor, it either reads the existing suba_rec in the database or it adds one. This error will occur if *purch* cannot find the suba_rec that it located previously.

Could not finish the ap voucher.
Could not finish the pc voucher.

When you leave *purch*, or when you change the date on the parameter screen, *purch* attempts to finish any vouchers you have started. If *purch* cannot do so, it will print one or both of the above messages. This is probably because of a problem in the ptp communications between *purch* and *bgvoucher*.

Could not incomplete pc the voucher. (sic)
Could not incomplete the ap voucher.

When *purch* encounters a fatal error, it attempts to incomplete any journals it may have started via *bgvoucher*, under the assumption that if an error exists, it is easiest for the user to work with an incomplete journal. The *purch* program therefore will report the error and leave the journal alone. Run *vchrecover* for that journal. In addition, there is probably a ptp communication problem between *purch* and *bgvoucher* that should be resolved.

Could not locate 'pay terms code' in the payterm_table
Could not locate 'document code' in the doc table
Could not locate 'document code' in the doc_table. Error:nnnn
Could not locate 'subs code' in the subs_table.
Could not locate 'subs code' in the subs_table. Error:nnnn

When you enter a subsidiary or a document code on the parameter screen, or payment terms on the invoice screen, *purch* checks the appropriate table to verify that the code you entered is correct. If *purch* cannot find it, *purch* will tell you that it could not locate the code in the table. Furthermore, if it gets an error it does not understand when it looks the code up in the table, it will report that error and exit.

Could not locate acct 'fund'-'cntr'-'acct'-'proj' internally

This is an internal error, produced when *purch* cannot find an account that it previously added to its internal list of accounts.

Could not open database: nnnn

The *purch* program must open the database to be able to manipulate the database files. The most common cause of its failure to open the database is a *dbbuild* or *dbstatus* updating the database dictionary. When the *dbbuild* or *dbstatus* is finished, or if there is no *dbbuild* or *dbstatus*, run *purch* again.

Could not post entry:

When *purch* is posting entries to the database with *bgvoucher*, it does so in a two-step process. First, it verifies any purchasing and accounts payable general ledger information. Then it tells *bgvoucher* to post both entries. Once an entry has been verified, it should post without any trouble. If, however, *bgvoucher* does not successfully post the entry, it probably is caused by a problem with the database that *purch* and *bgvoucher* cannot resolve. The *purch* program will print the above message, followed by *bgvoucher's* description of the problem.

Could not rename the accounts payable vchpost file
Could not rename the purchasing vchpost file

When *purch* is using *voucher* transaction files (-m O), it must rename the output files to files that *voucher* will find. If it cannot rename the files, it will exit with a fatal error. If this error occurs, you may be able to rename the pc and ap files so that *voucher* will post them normally.

Could not select index 'pobody_key1'. Status nnnn.
Could not select index 'pobody_serial'. Status nnnn.

When you first display the purchase order form, *purch* reads the line items based on the purchase order document code and number. When you update line items on the form, *purch* updates the line items based on the line item serial number. If *purch* cannot select the proper key for reading or updating, it will print the above message. Check the pobody_rec database file and make sure the key in the error message has been built with the schema. If it has not been built, you will have to add the index and rebuild the file. You should not delete or change indexes or fields in the database without consulting Jenzabar, Inc. If you have not done anything to the pobody_rec file, this error should never occur because these indexes should have been built when you installed the purchasing system.

Could not send accts payable entry to bgvoucher:
Could not send purchasing entry to bgvoucher:

The *purch* program will report this error if the ptp call to send the general ledger information to *bgvoucher* fails. This message should be followed by a better description of the error by the ptp package or by *bgvoucher*. This error could happen if *bgvoucher* had been killed, or if there is a problem in the ptp communications between *purch* and *bgvoucher*.

Could not update purchase order body record. serial number '%ld'.
Status %d

Each individual line item in the purchase order form is stored in a separate database record. When you update a line, it must be updated in the database. The *purch* program prints the above message when the update fails.

Could not update subb record 'subs'-'subs no'-'bal code'-'bal prd'.
Status=nnnn
Could not update the document table 'doc code'-'stn number'. Status=nnnn
Could not update the purchase order record 'doc code'-'po'. Status=nnnn

Could not update the suba record 'subs'-'Subs No'. Status=nnnn

When *purch* tries to update the database and gets a status it does not understand, it will report the name of the file, the status it received from the dbupdate call, and the primary key of the record it is attempting to update. These messages probably indicate database problems. You should run a "make check" on the file to check for crashed indexes.

DBFIND error on acct_table: nnnn
DBFIND error on cntr_table: nnnn

These errors are printed when *purch* is trying to find a record in the database and it gets a status it does not understand.

Deleting Invoice

There may be a short delay between the time *purch* gets the command to delete an invoice and when it can finish the database processing involved. To show you that it received the delete command, it will print this status message.

Do you wish to leave the program (Y or N)

If all the possible *purch* options were not given on the command line, or if a parameter was invalid, *purch* will go to the parameter screen and allow you to enter the parameters. If you interrupt the parameter screen, *purch* assumes that you want to leave the program. It will ask you to verify your intention, and if you answer Y, *purch* will exit.

Document Code, Subsidiary, and Bgvoucher Mode Must be Entered

When you first enter *purch*, *purch* will go to the parameter screen and allow you to enter parameters. If you do not enter all the required parameters, *purch* will print this message and stay on the parameter screen until you interrupt the parameter screen or until you enter the required parameters.

Enter the Station Number for the Document

The *purch* program prints this prompt after you entered the Station command and before you have entered the documentation station number.

Erase all changes since last Write, Complete, or Terminate ? (Y or N)

When you erase the work you have done, *purch* will ask for verification to avoid accidentally deleting work you have done. When you verify that you really do want to erase your changes, *purch* will go ahead and erase them.

Erasing Changes since last Write, Complete, or Terminate

Since the erase process may take several seconds, *purch* prints this status message to the screen to tell you it is processing your command.

Error in getset on Form:Message

The *purch* program prints this message if it gets an error while you are adding or updating information on the purchase order form. If you have modified the form, you probably made a mistake in your modifications. Using the screen package error message, you can probably determine where your mistake is and correct it. If you have not modified the form, this error should not happen. It probably shows an error in the purchase order form that you did not discover during implementation.

Error occurred in verifying purchase order. Status='nnnn'

Before *purch* adds a purchase order, it checks to make sure a purchase order with that document code and number is not already in the database. If it gets an Informix error when it is trying to find an existing purchase order, it will print the above message.

Error occurred when loading gle detail on 'doc code'-'Doc Number'.
Status:nnnn

When you query on a purchase order, *purch* does not automatically load the general ledger information for that purchase order. When you change levels to the Charge level or the Invoice level, or when you Complete or Terminate a purchase order, *purch* will go out to the database and load the general ledger information. If it gets an Informix error when it is trying to find the general ledger entries for this purchase order, it will print the above message.

Error on database unlock on po_rec. Status=nnnn (sic)

When you query on a purchase order, *purch* locks that record so no one else can touch it. When you move to a different purchase order, *purch* then unlocks the old one and proceeds to lock the new one. If *purch* cannot unlock a record, it will print the above message and exit.

Error on dbfind 'gl_amt_rec', status nnnn

In the Review screen, *purch* must find the budgeted and actual general ledger amounts for the centers you specify. If it encounters an error when it is finding the general ledger amount records, it will print this message.

Error on dbfind of po_rec in query. Status=nnnn

The *purch* program will print this error and exit if it encounters an Informix error when it is finding the purchase order records in a *purch* query.

Error on selfield on po_rec. Key:'key name'. Status=nnnn

During program initialization and when it is reading and writing to the database, *purch* must select the appropriate key in the po file so it can look up purchase orders. If it cannot select the key it needs, it will print this message, where key name is the key it is attempting to use and nnnn is the Informix error code. Make sure the key name is a part of the specified database file. If it is not, you will have to add the key to the schema file and rebuild the schema. This error should never occur because the applicable indexes should have been built when we installed the purchasing system.

Error on structinit of pobody_rec
Structinit on pobody_rec returned an error

This error happens when *purch* cannot correctly initialize the line items on the purchase order form.

Error returned from dufindid

When you enter the Find Id function from an id field, *purch* will call the CX *dufindid* utility. If *dufindid* returns an error, *purch* will print the above message.

Error returned on a getset().

In the purchase order query, *purch* will print this message if the screen package gives it an error.

Error while finding 'file name' 'key value'. Status:nnnn

When *purch* is attempting to load the purchase order detail, it will look up the subsidiary entries, the subsidiary transactions, the invoices, the general ledger entries, and the general ledger transactions. If it encounters an error when it is doing so, it will print the above message, where file name is the name of the file, key value is the value of the key it is using, and nnnn is the Informix error number.

Finding Purchase Order Records Based Upon Query

When you execute a query for purchase orders, *purch* will print the above message to tell you it started finding the purchase order records.

Finishing loading bgvoucher...

When *purch* uses *bgvoucher*, it starts *bgvoucher* and then goes on about its business. You can query on purchase orders and invoices without ever touching *bgvoucher*. When the time comes to add or update a purchase order, however, *purch* goes back to *bgvoucher*, waits for it to finish loading, verifies that it is posting. If *bgvoucher* has not finished loading, *purch* will print the above message. There may be a short delay while *bgvoucher* finishes initializing.

For a Bill the Update is Done on the Invoice Level

If you are updating a Bill, you must do so from the Invoice level, not the Charge level. A Bill does not have charges; all it has are actual Invoice amounts.

Form 'Form Name' is not currently loaded. 'Form Name' is loaded

When you use the Output command to print a copy of a purchase order, *purch* will check to make sure you have the proper form loaded. If you do not have the proper form loaded, you will have to use the Initialize command to load the form. Then you will be able to use the Output command.

Form Not Initialized. Form 'Form Name' requested

If you have not specified a form on the Parameter screen, *purch* will not allow you to select a document station that requires a printed form. Select the proper form for that station, and then try selecting that station again.

Form is specified for this station but device not passed to program

In immediate mode, you must have an output device to use a document station that prints purchase orders. You will have to go to the parameter screen and tell *purch* where you want your output to be printed before you will be able to select this document station number.

Id not found in ID record

When you enter a vendor id, payee id, or responsible id, this id number must be in the id file. The *purch* program will not allow you to enter an invalid id.

In Batch printing the computer must issue the purchase order numbers

In the Batch mode, you can only select document station numbers in which *purch* automatically assigns a purchase order number. You cannot manually enter purchase order numbers for batch printing. Either you should select a different station number, or enter the program without the Batch option specified.

Initialized Form is 'old form'. Form 'new form' associated with this station

The *purch* program will not allow you to select document station numbers that do not have the same form as your current form. To select this station number, you will have to change your current purchase order form to the form specified in the document table.

Interfund account cannot be used

In generating accounts to be charged and encumbered, *purch* makes several validation checks. One of these checks verifies that none of the accounts used will effect an interfund account, as specified by the gld records. Correct your account number.

Internal Problem. Invoice = 'Amount'. Detail structure = 'Amount'

When you add or change a purchase order, *purch* goes through its internal list of accounts and amounts and determines what needs to be written to the database. If these amounts do not agree, there is a serious problem in *purch*.

Invalid General Ledger account

When you enter an account number, *purch* attempts to validate it. If there is a problem with the account, it will print out the specific problem and then the above message. Enter a correct account number.

Invalid Option From This Field

The *purch* program prints this message when you attempt to use the Find ID function from a field that is not an id field.

Invalid Printer

The valid printers are given by the CARSPRINTERS environment variable. You cannot tell *purch* to use any printer other than those given in CARSPRINTERS.

Invalid command
Invalid Option
Unknown command.

The *purch* program prints these messages when you type a command that is not shown on the prompt line. Look at the prompt line to find the valid commands.

Invalid pobodyget screen command nn!
Invalid pobodyput screen comm nn!

These messages show an internal problem with *purch*'s handling of the screen package.

Invalid selection, 'Your Selection' is not in the allowable range

When you query on several purchase orders, you can only choose from the purchase orders displayed on the screen. You must restrict your selection to the letter range of purchase orders displayed on the query screen.

Invoice Deleted

The *purch* program prints this message when it has successfully deleted an invoice.

Invoice Number is Required

When you enter an invoice on the invoice level, you must enter an invoice number. If there is no invoice number on the vendor's invoice, assign one to the invoice.

It appears that the PO record is incorrect or a voucher needs posted
The <actual or encumbered> amts ='Amount 1' the po says ='Amount 2'

When *purch* is not using *bgvoucher*, it creates a voucher transaction file that must be posted before you will be able to do anything to the purchase order. Sometimes you might try to do some work on a purchase order that has not been completely posted to the database. If this message occurs, make sure you have posted all the AC and PC voucher transaction files and then try the purchase order again. If *purch* still does not permit you to touch the purchase order, you should run *purchaudit* to fix the purchase order record.

Note: Before running *purchaudit*, try to determine how the purchase order amounts were erroneously modified. If it was not caused by an unposted voucher transaction file, please try to duplicate the problem. The only way this message should occur is if you have not posted a voucher transaction file.

Load of help screen 'screen name' failed

The *purch* program could not load the help screen. Make sure all the screens in the program source area have been installed and try the option again. If the error still occurs, there is a problem with the screen package.

Load of the screen package failed

The *purch* program is a screen oriented program. When it is loading, it attempts to initialize the screen so that its screen displays will work correctly. If it cannot do so, it will exit with a fatal error.

Loading Purchase Order Detail
Loading Purchase Order Information

When you change levels from the Purchase order level to the Charge level, or when you complete or terminate a purchase order, *purch* must look up the general ledger information for the current purchase order. The *purch* program prints this message when it begins, and clears the message when the purchase order information has been successfully loaded.

MSG_MQUEUE returned nn
MSG_SENDMAIL returned nn

If either of these errors occur, the program tried to send mail but the mail utilities failed.

No Records Found

There were no purchase order records that matched your query conditions. One of your query parameters is probably too restrictive. Another possible reason for not finding any purchase orders is that they were added by *purchinv*. The *purch* program will not work with purchase orders added by *purchinv*.

No free text fields available

There are several optional fields on the purchase order form that can be used for any purpose. These fields can be updated and modified from the Form screen. These fields are optional. If they are not on the screen, *purch* will print the above message.

No *bgvoucher* error status. Press <Return> for *bgvoucher* error message.

When *purch* gets an error from *bgvoucher*, it is usually a ptp error or a verification or posting error. If some other error occurred, The *purch* program will print the above message and exit. This error should not happen, and probably shows problems in the ptp communications between *purch* and *bgvoucher*.

No entries to be posted.

The *purch* program builds a list of accounts payable and purchasing transactions and then writes these to a voucher transaction file. It prints the above message when there are no transactions to be written.

No form specified in purchase order record.

The *purch* program did not print a form for this purchase order when the purchase order record was originally added. There is nothing to print or view for this purchase order.

Not allowed to work with subsidiary that has a unit of 'Unit Code'.

The subsidiary account in the subsidiary table does not agree with the unit specified by the CARSUNIT environment variable. You must either have the CARSUNIT environment variable set to the correct unit when you log in, or you must set the unit used in the subsidiary table to the correct unit.

Output completed successfully
Purchase Order Completed
Purchase Order Terminated
Successful Write -- Don't Forget 'C'omplete (if necessary)
Successful Write

Since the output, complete, and terminate processes do not finish until the form has been sent to the printer or until the information has been posted to the general ledger, there may be a delay between the time you start the command and the time it finishes. This message informs you that *purch* is finished and ready for your next command.

Permission Not Granted For Updating Encumbrances

You must have purchasing permission to update any charges.

Permission denied for this account

When you enter an account, *purch* attempts to verify that it follows the rules set up in the database for general ledger accounts. If it cannot be used with a PC or AP voucher, *purch* will print the above message. You will either have to correct the account number, or add AP and PC permissions to the account in the gla_rec.

Permission not granted execute station command

If neither Accounts Payable nor Purchasing permissions are granted, you cannot choose a station number. There is no reason to choose a station number when you will never be able to use any output option.

Permission not granted for adding Non Bill Purchase Order

If you do not have Purchasing permission, you cannot add a purchase order.

Permission not granted for adding a Bill 'Pay Out'

If you do not have Accounts Payable permission, you cannot add a Bill purchase order, or a Pay Out.

Permission not granted to complete purchase order

If you do not have Accounts Payable or Purchasing permission you will not be able to complete a purchase order. If you have Accounts Payable permissions, you will only be able to complete a purchase order if there are no remaining encumbered amounts.

Permission not granted to execute add command

If you do not have Purchasing permission, you cannot Add a purchase order.

Permission not granted to execute erase command

If you have neither purchasing nor accounts payable permission, *purch* will not allow you to add or modify anything. The *purch* program will not allow you to use the erase command when you cannot possibly have anything to erase.

Permission not granted to execute output command

If you do not have purchasing permission, you cannot print the purchase order.

Permission not granted to execute terminate command

If you have neither purchasing nor accounts payable permission, you cannot terminate a purchase order. If you only have accounts payable permission, *purch* will not let you terminate a purchase order with encumbered amounts. If you only have purchasing permission, you cannot terminate purchase orders that have quantities and amounts on the associated invoices.

Permission not granted to execute write command

If you have neither purchasing nor accounts payable permission, *purch* will not allow you to add or modify anything. The *purch* program will not allow you to use the write command when you cannot possibly have anything to write to the database.

Permission not granted to update purchase order

If you do not have purchasing permission, you cannot change anything on the purchase order.

Posting error

When *purch* is posting entries with *bgvoucher*, it first sends the entries to *bgvoucher* to be verified. When the entries have been validated, they are posted to the database. The two-step process prevents data inconsistency, and avoids touching the database unless everything is correct. The validation process is supposed to flag all errors. If a Posting error occurs, this generally shows that an index is crashed or that someone has the suba_rec or the subb_rec unnecessarily locked. Examine the error message for file names, and see if there are any other programs using that database file. Run a make check on that file to make sure the indexes are not corrupt.

Posting general ledger entries...

Purch prints this message when it has validated the general ledger purchasing and accounts payable entries and is posting the information to the database.

Press <Return> for the *bgvoucher* error message
Press <Return> to see the ptp error message

These messages are printed when *purch* needs to display an error message that is more than two lines long. The *purch* will first print a general error message, and when you press **<Return>** it will print the specific *bgvoucher* or ptp error message.

Printer has not be Initialized (sic)

You cannot Output a form when you have not specified a printer. Use the Initialize command to select a printer, then print the form.

Printer must be specified before selecting form. Blank form and enter printer

You must tell *purch* the printer you want to use before you can enter the form type.

Purch cannot Complete a purchase order added by Inventory Purchasing.
Purch cannot Output an Inventory Purchasing purchase order.
Purch cannot Terminate a purchase order added by Inventory Purchasing.
Purch cannot Update a purchase order added by Inventory Purchasing.
Purch cannot add charges to an Inventory Purchasing purchase order.
Purch cannot add invoices to an Inventory Purchasing purchase order.
Purch cannot delete charges to an Inventory Purchasing purchase order.
Purch cannot modify a purchase order added by Inventory Purchasing.
Purch cannot modify an Inventory Purchasing purchase order.
Purch cannot update charges for an Inventory Purchasing purchase order.
Purch cannot update invoices for an Inventory Purchasing purchase order.
Purch cannot view an Inventory Purchasing purchase order.

The *purch* program and the Purchasing Inventory coexist on the same system. They use many of the same database files, and do exactly the same general ledger processing. The *purch*, however, does not handle the printed purchase orders like *purchinv* does. The form *purch* prints is a loosely structured form that can be maintained as much or as little as you want. The *purchinv* program has much stronger control over what is printed and requires that everything be integrated into the purchasing inventory system. With *purch*, you can print off a purchase order and add line items and comments to the purchase order at the time you print it. The *purchinv* program, however, expects these line items to have been preprocessed by the Requisition Entry program and uses a different database record to store the purchase order line items. What this means is that *purch* should not and cannot be used for *purchinv* purchase orders, and *purchinv* should not and cannot have anything to do with *purch* purchase orders. This restriction is enforced by both *purch* and *purchinv*. The *purchaudit* program can still audit both types of purchase orders.

Purchase Order 'doc code'-'PO Number' has been changed. Internal buffers do not match the database. Please Query on PO after clearing the buffers in Query option

When you query on a purchase order, there may be a period of time between the time *purch* displays the list of purchase orders that match your query and the time you reach that purchase order. During this time, it is possible that someone updated the purchase order. This means the purchase order as displayed by your query may not be the same as the purchase order that is contained in the database. When you do any work on a purchase order, *purch* first checks the purchase order in the database to make sure the purchase order it has in memory is the same as the one in the database. If they are not the same, it prints the message above. You will have to re-execute your query to update *purch* with the latest information from the database.

Purchase Order Record with that number is already in the system
Purchase order record 'doc code'-'number' is already in the system
The purchasing system does not allow duplicate purchase orders. You must enter purchase order numbers for your document code that are not already in the database.

Read Line Number 'nn' from database. Not enough lines on form. Line Ignored
The purchase order form has a fixed number of lines. When *purch* reads the purchase order line items, it will only read as many lines as the purchase order form will print. Any other lines will be ignored. This should only occur if you revise the number of lines on your purchase order form.

Record is currently locked. 'doc code'-'PO Number'
Before you can modify a purchase order, *purch* attempts to lock the purchase order record so that no one else can use it. If someone else has it locked, however, you will not be able to change anything.

Rules Table is Not Set Up Correctly
Your gld_rec is not set up correctly. See *General Ledger Technical Manual* for information about defining values for the gld_rec.

STBIND: 'fields' on 'screen': Message
The *purch* program uses the CX screen package to display and accept input. To use a screen, it has to assign an internal variable to the screen field. One way it does this is to assign that screen field to a field in the database. If *purch* gets an error when it tries to do this, it prints out the above error message, where fields describes the record it is attempting to bind, screen is the name of the screen it is binding, and Message is the screen package error message. This message should not occur unless the screens or the database fields have been modified. If you have customized the database file mentioned in the error message, the program screens may need to be modified to reflect your customization.

Status n was returned to ctrlloop() and is not understood
This error is an internal error that should never occur.

Status from {FILECLOSE or FILEOPEN} 'file name': nnnn
When you start *purch*, it must open the Informix database files it will be using. When you enter the Bye command, *purch* finishes any vouchers it has started and closes all its open files. If it cannot open or close a database file, it will print the above message, where *filename* is the name of the file that it could not close and nnnn is the Informix error number. A *fileopen* error of 6001 may be caused by a change in the database dictionary or when the file is locked in Informer, *dbstatus*, or *dbbuild*. If you do get this error, try *purch* again.

Status from SELFIELD 'file name' 'key': nnnn
During program initialization and when it is reading and writing to the database, *purch* must select the appropriate keys in the database files so it can look up various values. If it cannot select a key, it will print this message, where file name is the database file it is trying to use, key is the key in the file, and nnnn is the Informix error code. Make sure the key is a part of the specified database file. If it is not, you will have to add the key to the schema file and rebuild the schema. This error should never occur because the applicable indexes should have been built when you installed the purchasing system.

Status from STRUCTVIEW 'file name': nnnn
After *purch* has opened a database file, it must select the fields in the file it is going to use. If it cannot do so, it will print the above message, where *filename* is the name of the file it was trying to use and nnnn is the Informix error number the dbstructview function returned. The only way this error should occur is if you have deleted a field in the files that *purch* uses and have not updated the program include files for that schema.

Terminate Purchase Order (Y or N)

When you terminate a purchase order, *purch* erases all encumbrances and charges. To avoid accidentally hitting the wrong key and terminating a purchase order by mistake, *purch* verifies that you do really want to terminate the purchase order before it continues with the terminate process.

The Encumbrance Bal Code is Blank which should not happen in charge detail

It should not be possible to have a blank balance code in the Subsidiary Balance records.

The Payee Id is Required
The Vendor Id is Required

When adding a purchase order, you must fill in both the payee and the vendor id fields.

The Purchase Order Detail Must be Loaded First ('L'evel)

If you have made any changes to the purchase order record, you must load the supporting detail before you can write the changes. This is done to ensure that no one else can change the records for this purchase order while you are modifying it.

The payables subsidiary cannot use subsidiary totals
The payables subsidiary must use subsidiary balances

When *purch* is loading, and when you change anything on the parameter screen, *purch* checks to make sure the subsidiary given defines a valid accounts payable subsidiary. Accounts payable subsidiaries do not use subsidiary total records, and accounts payable subsidiaries always use subsidiary balance records. If the subsidiary you enter uses subsidiary total records, or if it does not use subsidiary balance records, it cannot be an accounts payable subsidiary.

The requested station is reserved

The Document table controls the documents that are printed throughout the CX. If a document station is locked, that means someone else is using that station. To prevent purchase order numbers from being duplicated, you must use a different document station.

The subs_table does not have valid default pay terms

When you add an invoice, *purch* defaults the payment terms to the default payment terms in the subsidiary table. If these payment terms are not in the payment terms table, they cannot be used and *purch* will not allow you to add an invoice with these terms. Either the Subsidiary table should be corrected, or the payment terms should be added to the Payment Terms table.

There are n Accounts Payables Files to be Posted
There are n Purchasing Files to be Posted

When *purch* is finishing up, it checks to see how many voucher transaction files have not been posted. If there are any accounts payable or purchasing files to post, you should run *voucher* or *filepost* and post them. The database will not correctly reflect the purchasing system until the voucher transaction files have been posted.

There are at least two sube for gle 'Vch Ref'-'Vch No'-'Ent No'.
Cannot handle

The *purch* generates general ledger and subsidiary entries in a specific format that allows it to easily load and display the purchase order information for a given purchase order. It does not create multiple subsidiary entries for one general ledger entry. It cannot handle this situation. You have probably used an Accounts Payable journal to manually enter entries and transactions. To allow *purch* to correctly maintain purchase orders, you cannot use both *voucher* and *purch* on the same purchase order, and you cannot use both *voucher* and *purch* to enter invoices for the purchase order.

There are not any charges

The *purch* program prints this message when you tried to add an invoice to a purchase order that had no charges.

There are not any charges to update

You cannot update or delete charges when there are no charges.

There are not any invoices to update
There is not an invoice to delete

You cannot use the Update or Delete commands on the Invoice level if there are no invoices to update or delete.

There is no current purchase order
There is no purchase order, charge, or invoice information to display!
There is not a current purchase order record
There is not a current purchase order to 'O'utput
There is not a current purchase order to 'W'rite
There is not a current purchase order to Complete
There is not a current purchase order to Terminate
There is not a current record to update
There is not a current record

These messages are displayed when you attempt to delete, modify, or display purchase order information without first querying on or adding a purchase order. You must have purchase order information loaded to be able to do any of these actions.

There is not a record in the direction you are going

The *purch* program prints this message when you attempted to do a Next or a Previous and there are no purchase order records in that direction. You can only do Next or Previous when you are adding purchase orders. When you query on a purchase order, that will clear any purchase orders you have added from memory.

There is not enough room on the form for all the account numbers to print

This status message is printed when the account number section on the purchase order form does not have enough space to print all the account numbers affected by the purchase order. The *purch* program will still print the purchase order.

This Query Combination will Cause a Sequential Search. Continue? (Y or N)

When you Query on a purchase order, *purch* tries to select a key that will result in the fastest execution of your query. If your query is too general, *purch* will have to look through all the purchase order records in the database to find those records that meet your query conditions. If this is the case, the query will take a long time. To help you avoid waiting for a lengthy query that you may not need, *purch* asks you to verify your intentions before it executes the query.

This invoice is already in the list. Use 'U'pdate

The *purch* program prints this message when you attempt to add an invoice that is already in the system. Vendors should not send you two or more invoices with the same number for any given purchase order. If they do, update the previous invoice and add the items that appear on this invoice to the previous invoice.

This would cause the line item to exceed budget. Override budget (Y, N, R)

If the amount you enter for the charge or invoice causes that department to exceed its budget, *purch* will print the above message. If you want to override the budget, type Y. If you do not want to override the budget, type N. The *purch* program will allow you to reenter the account. If you are not sure and want to review the budget and actual amounts for this account, type R. This will display the Review screen for this account.

Trs do not equal bal amt: inv='Invoice Number' trs='Amount 1' bal='Amount 2'

This is an error message *purch* prints when it discovers that the transactions for an invoice do not add up to the amount on the invoice. This is not a fatal error, because *purch* can continue processing, but you should run *saaudit* and *purchaudit* as soon as possible to find out what is wrong with the transactions. For more information about saaudit, see *General Ledger Technical Manual*.

Unknown Action in chg_rec.Action='x'
Unknown Action in chgdtl_rec.Action='x'
Unknown Action in inv_rec.Action='x'
Unknown bgmode n.

During processing, *purch* stores the current status of the records it has read from the database. It knows when a record has been updated, when it has been added, and when it has been read from the database but has not been changed. The above messages are printed when these internal statuses have been disrupted.

VT file cannot be opened: Filename

When *purch* cannot create a voucher transaction file, it will print the above message giving the name of the file it attempted to create. The most likely explanation for this error is that there is a permission problem with the voucher posting directory. Make sure the voucher posting directory and the installed version of *purch* have the correct permissions.

Verifying general ledger entries...

During a Write, when *purch* begins verifying the transactions through *bgvoucher*, it will print the above status message.

When adding record could not find doc table 'doc code'-'Station Number'. Status:nnnn

Before *purch* allows you to begin the add or update process, you must select a document station. This allows *purch* to control the document numbers it issues, and prevents others from improperly printing purchase orders out of sequence. This means that by the time *purch* gets down to adding the purchase order, it knows the document station it is using. It finds the Document table entry so it can update the last document issued. If it cannot find this Document table entry, it exits with a fatal error. The most likely cause of this is that someone has deleted or changed the Document table between the time you selected a document station and the time *purch* tried to print the purchase order.

When loading purchase order detail the voucher 'Vch Ref'-'Vch No' Entry Type 'Ent Code'. was found. The program cannot proceed with this purchase order.

The *purch* program expects purchasing general ledger entries to be written in a specific format. It can handle only certain entry types and entry groupings and cannot correctly load and update the general ledger if other entry types are present. The most common cause of this error is that you added invoices to this purchase order with *voucher*. You cannot use *voucher* and *purch* on the same purchase order.

You cannot add invoices to a Terminated purchase order.
You cannot delete invoices to a Terminated purchase order.
You cannot update invoices to a Terminated purchase order.

When you Terminate a purchase order, you erase all processing you have done to it and block all access to that purchase order in the future. You should only Terminate purchase orders that will never be used again. If you receive an invoice for that Terminated purchase order, you will not be able to add it that to that purchase order. You will have to add another purchase order or send the invoice back to the vendor.

bgv_instruction(BG_POSTLATER) error:
bgv_instruction(BG_WAIT) error:

bgv_setnotify(BG_POST+BG_VERIFY+BG_NOSIG) error:

When *purch* is initializing *bgvoucher*, it sends *bgvoucher* several instructions that describe the actions *bgvoucher* is to take during posting and verification. If *bgvoucher* does not accept these instructions, there is probably a communications problem between *purch* and *bgvoucher*. The *purch* program will print one of the above messages, and then print *bgvoucher's* error message. This error should never happen.

fscl_cal_rec for 'Type' does not have the same fscl yr for all amount types

This message is caused by a problem in the Fiscal Calendar records. An amount type (Actual, Budgetary, or Encumbered) probably has overlapping records or has been added with the wrong beginning and ending dates. Check the Fiscal Calendar record for your Accounts Payable subsidiary and for your general ledger. Make sure each fiscal year is distinct and that the dates are correct.

getset() Error: Message
getset() Param Error: Message
scget() Error: Message

The *purch* program prints this error when it gets an error in the screen package on the parameter screen. This error should not happen.

lseek on {header, trailer} of 'ap or pc' voucher transaction file
failed: vtptr='n'

This error can occur when Purch is trying to write the header or the trailer of the appropriate voucher transaction file. This error should never occur. The file *purch* has been creating may have been removed, or this error may be caused by disk problems on the portion of the disk the pc or ap file is on.

'filename' is locked will try to access in 5 seconds

At different points in the purchasing process, *purch* has to update the suba_rec and the subb_rec without interference from any other users. To do this, *purch* must lock the record it is going to update. If another program has this record locked, *purch* will not be able to lock the record and will have to retry the lock.

Resolving System Failures in the *purch* Program

Steps to Recovery

In the event of a system or program failure, you must complete the following recovery steps regardless of the action that was being executed in *purch*.

1. Reset the value in the field rsv_by_uid to "0" (zero) in the doc_table entry for the document code and station number that was used with *purch*.
2. Update the doc_table field last_issued_num to reflect the correct purchase order number.
3. Rename all unposted VT files ($CARSPATH/vchpost/), as in the following examples:
   mv $CARSPATH/vchpost/pc012345 $CARSPATH/vchpost/PC012345
   mv $CARSPATH/vchpost/ap012345 $CARSPATH/vchpost/AP012345
4. Use *voucher* to post the renamed files described in step 3 before continuing with the following steps.
5. Re-access the *purch* program.
6. If the operator was using the Write, Terminate or Complete command at the time of the system failure, additional processing is required. The additional processing verifies that the accounting system is in agreement with the Purchase Order record. Test the data integrity by completing the following steps:
   - Restart *purch* .
   - Query on the purchase order that was in progress when the program was aborted.
   - Once the Purchase Order record is found, select the Write command. The *purch* program notifies you if the Purchase Order record does not match what is in the accounting system.
   - Continue to work on the purchase order to obtain the desired results.
7. If *purch* does show data inconsistency, update the Purchase Order record to match the accounting system.

CAUTION: The *only* fields that you can update outside *purch* are the fields po_amt_enc and po_amt_act. Any other updating outside *purch* results in erroneous, unsupported data.

Error Messages from the *approve* Program

Introduction

This section lists possible error messages which may be encountered while running *approve*. With each error message will be an explanation detailing the probable cause as well as a possible resolution. The resolution of some errors is beyond the scope of this document and should be referred to the Jenzabar, Inc. support center.

DENIED:Need userid_table entry to run approve. *See* Coordinator.
DENIED:Need proper appr_table entry to run approve. *See* Coordinator.
The *approve* program encountered an error while attempting to locate appropriate table entries for the current user. This error is caused by improper table setup.

No Purchase Orders of Invoices Needing Approval
The *approve* program has encountered no error. This message is displayed on the screen when a user attempts to run *approve* and there is currently nothing awaiting his/her approval.

Missing filename. (reference)
The *approve* program encountered an error while attempting to locate a record in <*filename*>. Pertinent information about error is contained in <reference>. This error may be caused from improper Approval table setup. An examination of the Approval table entries associated with the specified <reference> may indicate the source of the problem. Otherwise, contact the Jenzabar, Inc. support center for resolution of the problem.

Database Errors

This section outlines errors which may occur during database related operations within *approve*. These errors will be sent by electronic mail to the user and will cause the program to exit. These are uncommon errors and their resolution should be referred to the Jenzabar, Inc. support center.

DBOPEN(dbname): Error NNNN.
The *approve* program encountered error <NNNN> while attempting to open database <dbname>.

FILEOPEN(filename): Error NNNN.
The *approve* encountered error <NNNN> while attempting to open <*filename*>.

DBSTRUCT(filename): Error NNNN.
The *approve* program encountered error <NNNN> while attempting to define the structure of <filename>.

DBSELFIELD(fieldname): Error NNNN.
The *approve* encountered error <NNNN> while attempting to select<fieldname> for searching.

DBUPDATE(filename): Error NNNN. (reference)
The *approve* program encountered error <NNNN> while attempting to update <*filename*>. Pertinent information about the update record is contained in <reference>.

DBFIND(filename): NNNN (cfile). (reference)
The *approve* program encountered error <NNNN> while attempting to find record in <*filename*>. The error was produced in the C code source file <cfile>. Pertinent information about the search record is contained in <reference>.

TBL(tablename): reference.

The *approve* program encountered an error while attempting to load <tablename>. Pertinent information regarding exact error is contained in <reference>.

Non-Database Errors

This section outlines errors which may occur during non-database related operations in *approve*. These errors will be sent by electronic mail to the user and will cause the program to exit. These errors are uncommon and their resolution should be referred to the Jenzabar, Inc. support center.

SCRINIT: reference.

The *approve* encountered an error initializing screen package. Pertinent information regarding the exact error is contained in <reference>.

SCREXIT: reference.

The *approve* program encountered an error exiting screen package. Pertinent information regarding the exact error is contained in <reference>.

LOAD: reference.

The *approve* program encountered an error loading a screen. Pertinent information regarding screen name and exact error is contained in <reference>.

BIND: reference.

The *approve* program encountered an error binding a field to a screen or a form. Pertinent information regarding screen or form name, field name and exact error is contained in <reference>.

TEXT: reference.

The *approve* program encountered an error displaying the text layout of a screen. Pertinent information regarding screen name and exact error is contained in <reference>.

SCROLL dir: reference.

The *approve* program encountered an error scrolling the information on a screen. Pertinent information regarding screen name and exact error is contained in <reference>. Error occurred while attempting to scroll in direction <dir>.

FPS_OPEN: reference.

The *approve* program encountered an error while attempting to load a form. Pertinent information regarding form name and exact error is contained in <reference>.

FPS_INST: reference.

The *approve* program encountered an error while attempting to perform an instruction. Pertinent information regarding instruction and exact error is contained in <reference>.

FPS_PUTSET: reference.

The *approve* program encountered an error while attempting to add a line to a form. Pertinent information regarding form name and exact error is contained in <reference>.

FPS_WRITE: reference.

The *approve* program encountered an error while attempting to add text to a form. Pertinent information regarding form name and exact error is contained in <reference>.

PRM: reference.

The *approve* program encountered an error while parsing program parameters. A detailed description of the parameter format is contained in <reference>.

ADROPEN: reference.

The *approve* program encountered an error while initializing Alternate Address. Pertinent information regarding exact error is contained in <reference>.

ADRLOAD: reference.
The *approve* program encountered an error while loading Alternate Address. Pertinent information regarding exact error is contained in <reference>.

ADR: functname() failed. (idno).
The *approve* program encountered an error in <functname> within Alternate Address. The error was encountered while processing id number <idno>.

Mail Messages Received During Crash Recovery

The *approve* program has been designed to be easily rerun in the event of an unanticipated crash due to system failure or program error. One side effect of this design is the possible inaccuracy of mail messages. The *approve* program may send you mail stating some items have been Approved or Rejected when the database has not actually been updated to reflect this status. Erroneous mail messages can occur when a program crash which takes place before the message "Database Update Successful. . ." is displayed on the screen. This message appears only when the database is current and has been updated successfully.

Voucher Errors

This section outlines errors which may occur during the posting of transactions to *bgvoucher* within *approve*. These errors will be sent to the screen and will cause the program to exit. A detailed description of the error will be sent by electronic mail to the user. These errors are uncommon and their resolution should be referred to the Jenzabar, Inc. support center.

BGV_CONNECT: Unable to connect. See Mail.
The *approve* program encountered an error while attempting to connect to *bgvoucher*. This message is sent to the screen and the exact error conditions are sent by electronic mail to the user.

BGV_START: Unable to start journal. See Mail.
The *approve* program encountered an error while attempting to start a journal with *bgvoucher*. This message is sent to the screen and the exact error conditions are sent by electronic mail to the user.

BGV_PUTENTRY: Unable to post entry. See Mail.
The *approve* program encountered an error while attempting to post journal entry with *bgvoucher*. This message is sent to the screen and the exact error conditions are sent by electronic mail to the user.

BGV_GETSTAT: Unable to get status. See Mail.
The *approve* program encountered an error while attempting to determine the status of an entry made through *bgvoucher*. This message is sent to the screen and the exact error conditions are sent by electronic mail to the user.

BGV_SETNOTIFY: Unable to set notify. See Mail.
The *approve* program encountered an error while attempting to stop processing on a connection to *bgvoucher*. This message is sent to the screen and the exact error conditions are sent by electronic mail to the user.

BGV_FINISH: Unable to finish journal. See Mail.
The *approve* program encountered an error while attempting to finish a journal created in *bgvoucher*. This message is sent to the screen and the exact error conditions are sent by electronic mail to the user.

BGV_GETVCH: Unable to get voucher. See Mail.
The *approve* program encountered an error while attempting to determine the voucher code and voucher number from *bgvoucher*. This message is sent to the screen and the exact error conditions are sent by electronic mail to the user.

BGV_CLOSE: Unable to close journal. See Mail.
The *approve* program encountered an error while attempting to close a connection to *bgvoucher*. This message is sent to the screen and the exact error conditions are sent by electronic mail to the user.

Error Messages from the Checkwriting Process

Introduction

The check writing process includes several different steps and six programs that must work correctly for the process to finish. If there is a breakdown in any of these procedures, you must back up to the previous step, correct the problem, and continue the process. This section contains information on correcting specific problems that may occur during the check writing process.

Using *ckabort* to Correct Errors

The Check Abort option (*ckabort*) can be run at any time from the creation of the check group record up to the time checks are posted. Its primary purpose is to restore the system to the state it was in before checks were selected. It should be run if the system goes down while checks are being selected or if the check selection program unexpectedly gets a fatal error. If there were problems during the printing of checks you can run the prepare for Check Abort menu option then run *ckabort* to return to the pre select state.

The *ckabort* program removes the check requests that *ckslct* created, and unlocks the invoices that the selection reserved for the given check group.

Common Errors in Check Writing

Listed below are several of the most common problems that may occur and the procedures that can be done to correct them.

A Check Should Not Have Been Printed
Voiding a check is the most common means of correcting a check that should not have been printed. If a document has passed through the check writing process (e.g., it has been printed and posted), and a problem is discovered with the check, you should manually void the check and then use the Void a Document (*docvoid*) option. This option reverses the check and restores the invoices on that check to the way they were before the checks were selected.

Note that the invoice remains in the system, and still ready to be selected. The next check selection run will locate the invoice and print a check for it unless you use the *purch* program (or a journal entry) to remove the liability for the invoice.

The *ckslct* program does not select all invoices
If *ckslct* does not select all the invoices you anticipated, examine several of the invoices in question. The most common reasons for an invoice not to be selected are:

- The invoice due date is not a valid date to select. All invoices selected on a given run must have an invoice due date that is less than or equal to the due date in the Check Group record. Either change the due date in the Check Group record or the due date in the invoice to select the invoice.
- The invoice is locked by a previous check run, and the previous check run was not successfully completed or terminated. This can occur if the check request number is not zero or if the status is L.

- If the check request number is not zero and the status is L, then the invoice has been selected for payment. Do not interfere with the processing unless you are absolutely certain this invoice was not paid on another check run. The invoice may have been selected but a check may not have been printed yet. In any case, the check run on which this invoice was selected has not yet been posted. Make sure that all checks have been printed and posted.
- If the check request number is not zero but the status is O, then the system printed and posted the checks, but the journal that contained the posting was later terminated. To reset the check request numbers in the invoice records, call the Computer Center.
- If an invoice has been held for payment, it will not be selected. An invoice will also not be paid until it has been approved. If a check needs to be written for the invoice, set the hold payment flag to "N" or have the appropriate person approve the invoice.
- A manual check (sometimes known as a "Quick Check") was posted to the wrong invoice. This is particularly common when initially implementing the accounts payable system, or when someone unfamiliar with the manual check procedures writes and posts a check. The journal for posting a quick check uses (by default) the oldest eligible invoice for a vendor. The person posting the check may not realize this was the wrong invoice, or he/she may not know how to change the default selection. Look at the transactions for the invoice. If a quick check journal has been used for the invoice, and if a partial payment or an overpayment has been made, it is likely that the check was posted to the wrong invoice. To correct this error, make an adjusting journal entry between the incorrectly posted invoice and the correct invoice, debiting and crediting accounts payable.

Checks Were Printed with the Wrong Check Numbers

This error occurs only if a quick check or a previous check run has not been successfully posted. Users discover this error when *ckpost* sends a mail message as follows:

Example: The first physical check number is '24034' but the document table's last number issued is '24038'. The physical check number should be one greater than the document table number.

The primary solution for checks printed with the wrong numbers is to reprint the entire check run. Use the Prepare for FPS Restart option to restore the forms file so it can be re-printed. Then use *fps* to print the checks again. The Document table's last issued number will default into in the First Used in Setup field. Enter the number of the next check in the stack of pre-printed checks that will be printed on for the First to be Printed field. The range of checks to void will cover the range of checks printed for the first run. These documents must be voided by hand.

If the system sends a the above mail message, but your review of the checks verifies that the checks are correct, then one of the following error conditions has occurred.

Document Table Number Greater than or Equal to Check Number

Look at the numbers given in the message. If the number in the Document table is greater than or equal to the first check number, the system determines that the range of checks in the error message has already been posted. You can cause the system to post the checks you have just printed by updating the last issued number in the table to be one less than the first physical check number as given in the error message. It is important, however, to determine the reason the Document table was incorrect. Verify (through the *acquery* program or through journal reports) that the document numbers in question have not already been posted to the database.

If a check has already been posted using that document number, the error may have occurred by entering the First Used for Setup number incorrectly. This error results from manual intervention and should not occur if *fps* locates and defaults the next document number to be used. In the definition file for the form, located in $CARSPATH/modules/acctspay/forms/ckslct, the instructions section should be similar to the following:

```
formtype = apcheck;
number = ckno;
alignment;
tracking;
voiding;
getnum = $CARSPATH/install/scp/acctspay/lastpr.scp;
```

(where "apcheck" is the name of the form.) If the line beginning with "getnum" is not present, FPS will not find and default the document number correctly.

The checks can be posted by changing the tracking file in $CARSPATH/spool/forms. Look for entries in the tracking file corresponding to the incorrect document numbers that look like this:

```
>>f:FPS_SETUP:13491:VOID
or
>>f:FPS_ALIGNMENT:13491:VOID
```

You can delete the incorrect FPS_SETUP or FPS_ALIGNMENT setup entries from the tracking file.

If a check that was printed on the current check run corresponds to a check that has already been posted, there was a problem with the previous check posting run or with the forms you are using to print the checks. The previously posted check should not have been posted to the check number just printed. If the correct check writing procedures have been followed, this should not happen

Document Table Less than First Check Number

If the number in the Document table is less than the first check number, then there are two possibilities.

1. The most common possibility is that a manual check or another check run has not been posted. Post the other check run (or manual check) and then try the check posting option again.
2. If there are no other check runs and if no manual checks have been written, you may have entered the incorrect document numbers while printing. The system will normally default the correct numbers from the document table, and it is not possible to change the number of the first document used for setup. This also can be corrected by modifying the tracking file. Insert additional lines into the tracking file that read as follows:

```
>>f:FPS_SETUP:13491:VOID
>>f:FPS_SETUP:13492:VOID
>>f:FPS_SETUP:13493:VOID
```

There should be one FPS_SETUP line for each missing document number.

Note again that you should not modify the tracking file unless you are absolutely certain there has in fact been a manual error entering the document numbers and there have been no other checks printed since the last posted check run.

If you need to reprint some of the checks, the "Prepare for FPS Restart" option will allow you to restart the check run and resume normal processing.

The Printer Jammed During Printing

If the printer jams during printing, interrupt the printing procedure, reset the printer and realign the forms, then use the Restart command. It prompts you for the last correctly printed form. Enter this number. The forms printing program pauses while it restores the tracking file to the correct location, a step that can take several minutes. Next, the system prompts you to enter the number of the next form to be printed. Enter the number of the next check, and continue printing forms as normal.

Occasionally, a printer will jam on one of the last checks to be printed. Use the Prepare for FPS Restart option to restore the tracking file to a point where you can restart the forms, then use the Restart option in *fps* to enter information about the forms that were printed correctly.

Error Messages from the 1099 Processes

List of Errors

Several combinations of warnings and errors can occur in the processing of 1099s, including the following:

Error occurred during parameter processing.
f1099bld usage: f1099bld -s subs -y year
Invalid year. Must be a two-digit number (eg, 96 instead of 1996)
No year specified for '-y'
You must specify at least one subsidiary (-s subs)
You must specify the calendar year (-y year)

You entered parameters incorrectly from the menu option. Try the option again, entering the parameters correctly. If the same error messages occur, the menu option may contain an error.

Dbfind error on 'subb_rec', status 107.
Current subb_prim: 'A/P -5003', '0015', 'INV '
The database read was retried for 10 times with no effect.

Another user is working with the selected invoice. All the invoices for the correct calendar year should have already been paid. If this error occurs, either another user is voiding the check that paid the invoice, or you entered the incorrect calendar year on the parameter screen. Attempt to run the program again, using the correct calendar year. If the error still occurs, your database may contain an error.

Unknown Command Line Argument 'xx'

The creation of the 1099 FPS file requires only the optional id specification. This error indicates that an error exists in the menu option file.

Country Code 'CAN' Not Found, ID=1005.

This message indicates that a country in the id_rec was not found in the Country table. To correct this error, either add the appropriate code to the Country table or change the country code in the individual's id_rec, then use the Create 1099 FPS File option again.

List of Warnings

You may receive the following types of warning messages:

All payments for invoice '3', vendor 100 are not in the same year. Subsidiary balance 'A/P - 100', code '0003', period 'INV'. Examine this balance carefully to avoid printing an invalid 1099.

This message occurs when you pay an invoice with two or more payments that span two (or more) calendar years. Review the invoice to determine the calendar year for which the applicable 1099 should be printed. You may have to delete the unneeded 1099 manually.

Boxes 5 and 6 on 1099-INT

Box 5 reports foreign tax withheld and paid on interest. CX does not determine foreign taxes paid on interest. If you need to use this box, add the 1099 record manually, using the appropriate amount. If the address of the person who will receive this 1099 is in a foreign country, the 1099 programs will print the name of that country in box 6. If the address of the 1099 recipient is in the United States, the 1099 programs will not determine the contents of box 6. You must fill this box manually after the 1099 has been printed.

Boxes 9 and 10 on 1099-MISC

Box 9 indicates sales of $5,000 or more of consumer products to someone on a commission basis. Box 10 is a box indicating that an amount in box 7 was crop insurance proceeds. The CX cannot at this time automatically track crop insurance proceeds, or commission sales. If your institution needs to use either of these boxes, add a 1099 record with the amount of $1.00 for that box.

Boxes 11 and 12 on 1099-MISC

Box 11 reports any state income tax withheld. Box 12 reports the state abbreviation combined with the payer's state identification number. The IRS provides these boxes for your convenience, and does not require that you complete them.

Tape Production Error Messages

The f1099tape program can generate the following error messages:

Unknown Command Line Argument 'xx'

This message typically results from an error in the menu option, or a problem during installation. Contact Jenzabar, Inc. for more information.

Cannot Open File ‘usr/carsi/spool/tape/f003945.dat’'

This message results from restricted permissions on the /usr/carsi/spool/tape directory. Verify that complete permission to access that directory have been assigned, and rerun *Create 1099 Tape File*.

*****Warning*** No 1099s Have Been Processed**

This error probably happens because you did not run the "Create 1099 Records" option, or the options you gave to "Create 1099 Records" did not cause any 1099 records to be added. If you did print some 1099 forms, there is either a problem with the 1099 tape program or with the database file.

Tape Verification Error Messages

The *taperead* allows you to verify that a tape was successfully created. It will display the information on the tape, line by line, until you are satisfied that the information has been successfully written. The following errors can occur when you use this utility.

Can’t Open Tape Device: ‘dev/tape’

This message indicates that the tape drive is not ready. Check the tape drive. Make sure the tape is loaded properly and that it is on line. Review any error messages from the system (e.g., ts0: not online) that could clarify the 1099 tape writing error. If you continue to receive the error message, your standard tape device may not be linked to /dev/tape. If /dev/tape does not exist on your system, use the UNIX *ln* command to link it to your standard tape device. For example, *ln /dev/rmt/0m /dev/tape* links the standard tape device of /dev/rmt/0m to /dev/tape.

Error in Scanning Spool Directory ‘usr/carsi/spool/tape'

This message indicates that you do not have permission to access the tape spooler directory. Verify your permissions to make sure you have permission to access this directory.

No More Files to Process in Spool Directory ‘/usr/carsi/spool/tape’

This message occurs when you did not choose to write any of the applicable files in /usr/carsi/spool/tape to the tape drive.

No Files to Process in Spool Directory ‘/usr/carsi/spool/tape’

This message appears when the Prepare Tape (*f1099tape*) option did not create any intermediate files in the /usr/carsi/spool/tape directory. Rerun this option and then try to execute *f1099tp* again.

openfile: stat: Could Not Obtain Status on File
Can’t Open Data File: ‘/usr/carsi/spool/tape/f010934.dat
Can’t Open Label File: ‘/usr/carsi/spool/tape/f010934.lab

These messages occur if you do not have permission to access the intermediate tape file. This may occur when different users attempt to create the intermediate files and write the tape. The same user should perform both operations.

Budget Processing Error Messages

Budget processing requires the use of several programs. The following errors can occur:

BG_FINISH: -1. errno -1. ptp_errno 0
Bgvoucher cannot finish a journal it has not started or continued.

Period 'BAL ': --> There are no changes to be posted for BAL

Period 'JULY':
BGV_START(JULY): -1. errno -1. ptp_errno 0
This period ('07/01/1995', 'JULY') is not open.

INDEX

D

E

F

G

I

J

L

M

N

P

R

S

T

U

V