Jenzabar CX

General Ledger Implementation and Maintenance

Technical Manual

Filename: tmglimmt
Distribution date: 12/05/2001

Contact us at www.jenzabar.com

JENZABAR, INC.
GENERAL LEDGER IMPLEMENTATION AND REFERENCE TECHNICAL MANUAL

TABLE OF CONTENTS

SECTION 1 - USING THIS MANUAL

Overview

Purpose of this Manual

This manual provides technical information required to install, support, and maintain the CX General Ledger module. For reference information about General Ledger tables, records, programs, menus, screens, scripts, etc., see Volume II of this manual.

Intended Audience

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

How to Use This Manual

If you are not familiar with the processes and features of the General Ledger module, read the manual for:

- Detailed reference information about how the module works
- Procedures for customizing and maintaining the module

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

Standard Product

This manual provides technical information about the CX standard product in GUI format. Your institution may or may not have all the features documented in this manual. Refer to this technical information when you want to customize screens for your institution.

Structure of This Manual

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

Overview information:

Section 1 - Information about using this guide
Section 2 - Overview information about the module

Module reference information:

Section 3 - Tables used in the module
Section 4 - Macros and Includes
Section 5 - Program Files
Section 6 - 23 - Program documentation and data flow diagrams
Section 24 - Menus, Screens and Scripts
Section 25 - Reports

Module procedures:

Section 26 - Procedures to install and customize your processes
Section 27 - Procedures to perform for maintaining the module

Error reference/Recovery procedures:

Section 28 - 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 General Ledger module.

QuickMate online help:

Using QuickMate
Getting Started User Guide

UNIX-based help:

Help command (Ctrl-w) in screens and menus

User guides:

Using General Ledger
Getting Started User Guide

Conventions Used in This Manual

Introduction

Jenzabar 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 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 cross-reference a section of text
- To represent a variable for which you substitute another variable (e.g., substitute *filename* with an appropriate filename)
- To represent a program name (e.g., *acquery*)

<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.

"Quotation marks"
Represent information written in this guide exactly as it appears on the screen (e.g., The message, "Now Running..." appears.).

CARS-Specific Terms

The following term conventions appear in this guide.

Application
A group of one or more software programs that enables you to perform a particular procedure, such as entering accounting 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 product, CX.

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).

Capitalized Words

The first letter of each word in a command, option, field name, or menu or screen title is capitalized in Jenzabar technical manuals to set those terms apart from regular text.

SECTION 2 - INSTALLING THE GENERAL LEDGER PROCESSES

Overview

Introduction

This section provides procedures for setting and installing the features of the General Ledger module. The following procedures are included:

- Assessing institution needs for the module
- Reviewing data in tables and records
- Setting up special purpose accounts and customizing the account number structure
- Implementing security features
- Entering first year budget information
- Creating a new subsidiary

Resources for Basic Information

This section contains detailed procedures specific to the General Ledger module. 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*

Assessing the General Ledger Setup

Introduction

CX provides several ways to implement the options of the General Ledger module. After assessing the needs of your institution, you can change the default settings of General Ledger enable macros and reinstall the module.

This section lists and describes the features that you must assess before you can modify the macros.

Order of Implementation

Jenzabar recommends that institutions implement the General Ledger module before any other financial modules or applications. The other financial areas include Cashier, Personnel/Payroll, Requisitioning and Accounts Payable, and Fixed Assets.

Because most activities on a campus impact the accounting area, the implementation of any CX module affects General Ledger. If your institution elects to implement one of the non-financial areas first (e.g., Student Billing, Registration, or Institutional Advancement), you must set up the required accounts in the Fund, Function, Object and Subfund tables. For example, tuition and fees for Registration, and donor holding accounts for Institutional Advancement are required. You must also set up the Defined Accounts record and the Fiscal Calendar record to validate the account combinations you create, and the dates you use on your financial transactions.

Account Number Structures

The standard CX account number includes the following four components:

- Fund
- Function
- Object
- Subfund

If the institution uses other information as part of its account number, you must modify macros and reinstall portions of the system. For more information about changing the account number structure, see *Implementing Customized Account Numbers* in this section.

Maintenance of More Than One Database

Jenzabar recommends that institutions use and maintain both a training and a live database. The training database enables new users to learn how to use the features of CX without impacting the actual live data.

If your institution trains users on the live database, you must ensure that any financial transactions created during the training process are nullified (e.g., the balances in accounts used for training must be zeroed, and the journals used for training must be terminated). To reset the journal numbers so the first live journal can use the number “1,” use the SQL statements in *Resetting Journal Numbers* in this section.

Use of Optional General Ledger Features

The institution can implement several options relating to General Ledger by enabling macros. The enable macros appear in the file $CARPATH/macros/custom.

The following list contains the General Ledger options that you can enable using macros. For more information about the purpose of all types of macros in General Ledger, see *General Ledger Macros* in this manual.

- ASCII posting menu options
- Associated Account report menu options
- Combined Center report menu options

Note: Many of the enable macros that appear in $CARSPATH/macros/custom relate to financial applications outside of General Ledger. For more information about the macros that implement the options, see the technical manual for the application.

Granting Permissions to Users

Granting permissions for users to access the financial area of CX is important because of the security issues involved.

You can restrict and grant access to screens and programs in the following ways:

- By adding users to the groups that can access the files
- By setting the users' login files to access specific menus
- By setting password protection on selected menus
- By entering permission information in the General Ledger Permission table and the Permission table

For more information about ways to maintain security in the system, see *Resolving Security Issues* in this section.

Customizing Reports for the Institution

The General Ledger module contains a variety of reports that users can run in different ways. For example, a report that displays balances in each function (or department) can be detailed or summarized, and can show current or prior year-to-date or monthly figures, or budgeted and encumbered amounts. To maximize reporting flexibility, a variety of report macros exist in macros/custom/finrpt. If the reporting needs of the institution require changes to the macros, see the *Reports* section of this manual.

Reviewing and Modifying Data in Tables and Records

Introduction

After assessing features of General Ledger and setting the appropriate enable macros, you must review the setup of CX tables and records.

Procedure

Follow these steps to review the values of the CX tables and records.

1. For each General Ledger table, review the codes supplied with 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 General Ledger system. Determine whether or not the records need to be updated to meet the needs of the CX reports. Make updates as appropriate.

Table and Record Information

For more information about the tables and records in the General Ledger module, see *General Ledger Tables and Records* in this guide.

Important General Ledger Tables and Records

Although CX, as delivered, contains a set of table values that an institution can use to get started processing General Ledger information, you must review the following tables in this order to ensure that the setup is correct.

1. General Ledger tables (fund_table, func_table, obj_table, subfund_table)
2. Financial Statement table (fs_table)
3. Defined Accounts record (gld_rec)
4. Fiscal Calendar record (fscl_cal_rec)
5. Document table (doc_table)
6. Amount Type table (atype_table)
7. Entry table (ent_table)
8. Journal table (vch_table)
9. Subsidiary tables (subs_table, subb_table, subt_table, subas_table)

Order of Table Information in This Section

Information about the setup of these tables appears in the following order in this guide:

- Tables that require modification appear first, in the order of implementation recommended by CARS.
- Tables that do not require modification appear after the required tables, in alphabetical order.

Order of Implementation

Jenzabar suggests that you implement the General Ledger tables and records in the following order:

Note: When you complete or verify the contents of these tables and records, the General Ledger module is available for use.

1. General Ledger tables:
   - Fund table
   - Function table
   - Object table
   - Subfund table
2. Financial Statement table
3. Defined Accounts record
4. Fiscal Calendar record
5. Document table
6. Other tables in any order

Order of Implementation of Subsidiary Tables

If your institution intends to use subsidiary processing, then Jenzabar recommends the following order for implementation of subsidiary tables:

1. Subsidiary table (subs_table)
2. Subsidiary Balance table (subb_table)
3. Subsidiary Total table (subt_table)
4. Subsidiary Association table (subas_table)

General Ledger Tables That Require Modification

Introduction

The tables that most institutions must modify during implementation are as follows:

1. General Ledger tables (fund_table, func_table, obj_table, subfund_table)
2. Financial Statement table (fs_table)
3. Defined Accounts record (gld_rec)
4. Fiscal Calendar record (fscl_cal_rec)
5. Document table (doc_table)

Implementation considerations for these tables appear in this section.

General Ledger Tables (fund_table, function_table, object_table, subfund_table)

You must complete the General Ledger tables before using the General Ledger module.

The chart of accounts contains several components. Standard CX includes four components: the fund, the function, the object, and the subfund. During implementation, you define all the valid values for each component in separate tables, then describe the valid combinations of the components in the Defined Accounts record.

Note: For information about changing the standard account number structure for your institution, see *Implementing Customized Account Numbers* in this section.

Within the General Ledger tables, you set up the following special purpose accounts:

Temporary subsidiary control accounts
For example, if you have not yet implemented Accounts Payable, you can define a temporary control account containing the total balance of payables at the institution. The balance in the temporary control account is a credit. Later, as you implement Accounts Payable and set up the subsidiary accounts you need to use, you debit the temporary control account and credit the correct subsidiary account for individual account balances. For more information about setting up a subsidiary control account, see *Establishing Subsidiary Control Accounts* in this section.

Budget contingency account
Many institutions define the object 9999 to use during the setup of the Budget module. The purpose of the contingency account is to enable you to use the *voucher* program to input budgeted amounts. Since *voucher* accepts balanced journal entries only, you can use a contingency account to help you enter the amounts. As you enter expense budgets, you debit the expense account, and credit the contingency account. As you enter revenue budgets, you debit the contingency account and credit the revenue account. If your institution operates at a breakeven point where revenues = expenses, the contingency account ends up with a net balance of $0.00. If your institution does not operate at a breakeven point, the contingency account will contain the net balance: a debit balance if expenses exceed revenues, or a credit balance if revenues exceed expenses.

Note: You define the budget contingency account object code in the Object table, and specify the code as the budget contingency account in the Defined Accounts record.

Donor holding account
The Institutional Advancement module of CX uses a donor holding account when gifts from donors are first received. Typically, the entry to record a donor gift debits a cash account

and credits the donor holding account. When the Alumni/Development staff determines the appropriate designation for the gift, an entry to debit the donor holding account and to credit the designation account is made.

Note: You define the donor holding account object code in the Object table, and specify the code as the donor holding account in the gld_rec.

Due to/from account

CX performs automatic fund balancing for interfund transactions. To maintain balanced funds, the system uses an interfund account, or a due to/from account. You define the due to/from account object code in the Object table, and specify the first two digits of the code in the macro *gl_interfund* in macros/custom/financial.

Cash over/short

When Cashiering functions result in an out-of-balance condition in a cash drawer, the account to debit with a shortage or credit with an overage is the cash over/short account. You define the cash over/short object code in the Object table, and specify the complete account in the macro CH_OS_DEFINE in macros/custom/financial.

Financial Statement Table (fs_table)

You must complete the Financial Statement table before using the General Ledger module.

The Financial Statement table controls the groupings of information on the standard financial statements. The statements contain blocks, groups, and schedules. Blocks are the broadest category, containing one or more groups. Groups are the second broadest category, containing one or more schedules.

The system generates three different types of logic for financial reporting, using the following sort criteria:

- Function
- Object
- Subfund

When sorted by *function*, the blocks, as the broadest category, might include Educational and General and Auxiliary Enterprises. Under the Educational and General block, the institution might include instruction, academic support, and student services as groups. Under the instruction group, possible schedules would include accounting, art, biology, and other departments.

When sorted by *object*, the blocks, as the broadest category, might include assets, liabilites, fund balances, revenues, and expenses. Groups, as categories within blocks, might include cash, investments and receivables under the asset block. Schedules, as the smallest categories, might include money market instruments and certificates of deposit under the investment group.

When sorted by *subfund*, the blocks, as the broadest category, might include Loan funds. Under the Loan funds block, the institution might include loan funds, and scholarship and grant funds.

Following is an example of the setup for the fs_table for the above examples by function, object, and subfund.

By function:

Block	Group	Schedule	Description
1000-????			Education and General
	1000-1299		Instruction
		1000-1099	Academic Departments
		1100-1199	Continuing Education
		1200-1299	Other Instruction
	1300-1349		Research
	1350-1399		Public Service
	1400-1499		Academic Support
	1500-1599		Student Services
	1600-1699		Institutional Support
	1700-1799		Plant Operations

By object:

Block	Group	Schedule	Description
1000-1999			Assets
	1000-1099		Cash
	1100-1199		Due to/from
		1110	Due to/from fund 10
		1121	Due to/from fund 21
	1200-1299		Investments
	1300-1399		Receivables
2000-2999			Liabilities
3000-3999			Fund balances
4000-5999			Revenue
	4000-4999		Tuition and other fees
	5000-5999		Other revenue
	6000-6999		Expenditures-general
	7000-7999		Capital expenditures

By subfund:

Block	Group	Schedule	Description
1000-1999			Restricted fund-subfunds
2000-3999			Loan fund-subfunds
	2000-2999		Loan funds
	3000-3999		Scholarship and grant funds
4000-5999			Endowment fund-subfunds
6000-6999			Plant fund-subfunds
7000-7999			Agency fund-subfunds

The following points are important as you complete the fs_table:

- Set up the axis codes for OBJ, FUNC or SUBF.
- Verify that a schedule exists for every group, even if the schedule is the same as the group.
- Ensure there are no gaps or overlaps in the block, group, and schedule numbering. For example, if the Object Group X ranges from 6400-6480, and Object Group Y begins at 6500, define the range for Object Group X in the fs_table as 6400-6499.
- Ensure that numbering for blocks, groups and schedules includes 1000 to 9999 (assuming the institution has not modified the size of any of the standard account component fields).
- The tfs report can verify that the fs_table is correct. Menu users can run this report by selecting Accounting: Table Maintenance: Financial (F-I) menu, and then running the menu option Financial Statement Report.

CAUTION: You must define the schedule level for every group to ensure reports contain consistent, complete totals.

Defined Accounts Record (gld_rec)

You must complete the Defined Accounts record before using the General Ledger module.

The Defined Accounts record performs the following three functions:

- Establishes rules for adding valid accounts
- Enables you to set up shortcut codes for commonly used accounts
- Defines special purpose accounts

Adding valid accounts

Within the Defined Accounts record, you can define valid accounts in the following two ways:

- Enter the exact account number in its entirety.
- Enter a masked value in the field, according to the *Rules for Entering Account Components* that follow.

Setting up shortcut codes for commonly used accounts

The Defined Accounts record contains a field (the gld) into which you can enter a single character to represent an account. To use this feature, you enter a code (e.g., **c**), then enter the account number that you want the code to represent (e.g., the Cash account, **10- -1010**). Then, whenever users access the *voucher* program to enter journal information, they can enter **c** in the first position of the account number field, and the Defined Accounts record will fill in the correct Cash account number automatically. The gle field in the Defined Accounts record is case-sensitive (i.e., it distinguishes between upper- and lower-case letters), and also accepts some special characters, so you can define up to 64 different codes.

Note: Some special characters are reserved for unique purposes and to designate special purpose accounts, and are not available for shortcut codes.

Defining special purpose accounts

After you set up the valid fund, function, object and subfund for your special purpose accounts in the General Ledger tables, you also use the gld field in the Defined Accounts record to define them. By using a reserved special character in the gld field, you indicate that the specified account performs a unique accounting function. The reserved codes and their purposes are as follows:

- $ (the budget contingency account)
- & (the donor holding account)

The following list describes all the fields in the Defined Accounts record:

Code

A one-character code that designates data entry-defined accounts (e.g., shortcut codes) or special purpose accounts (e.g., the donor holding account). Leave the field blank for entries that define valid account combinations.

Description

A 24-character description of the account code.

Function

The function code (e.g., 1010 for the Biology department).

Note: You can complete this field according to *Rules for Entering Account Components* that follow.

Fund

The fund number (e.g., 10 for the General Purpose fund).

Note: You can complete this field according to *Rules for Entering Account Components* that follow.

Object

The object code, e.g., 6100 for supplies.

Note: You can complete this field according to *Rules for Entering Account Components* that follow.

Subfund

The subfund code (e.g., 100 for a research project).

Note: You can complete this field according to *Rules for Entering Account Components* that follow.

Rules of Masking

You can mask the components of the account number that you want to validate, based on the following rules:

Any Alphanumeric Character

Any alphanumeric character (0-9, a-z, or A-Z) defines the exact code that must be used. For example, if an object is defined as 6100, the system accepts only 6100 as a valid account.

The + Character

The + character indicates that the code can include any alphanumeric character. For example, if an object is defined as ++++, the system accepts any valid alphanumeric string (e.g., 1000, 6100, 1abc, or mAf5), assuming the given account is valid in the Object table.

The ? Character

The ? character indicates that the code can include any alphanumeric character, or blank(s). For example, if an object is defined as ????, the system accepts any valid alphanumeric string (e.g., 1000, 6100, 1abc, mAf5, or 5 j), assuming the given account is valid in the Object table.

The " " (Blank) Character

The " " character indicates that the specified field must be blank. For example, you can define a cash account with an object code of 1010, and a function code of " ", causing the gld_rec to prevent the user from entering a function code with the cash account number.

Combinations of Characters

You can use the above characters in any combination. For example, if you define the Object code as "1+? ", the following conditions are required:

- The first character must be 1.
- The second character can be any alphanumeric value.
- The third character can be any alphanumeric value or blank.
- The fourth character must be blank.

Multiple Account Definitions

You can create more than one account definition for each code. If you need several account definitions to accurately define a valid code, the program reviews all the relevant gld_rec entries to validate your account. For example, if you create two gld_recs (one with an object code of 1+++, and the other with an object code of 2+++), the system will accept any objects that:

- Begin with 1 or 2
- Continue with any string of alphanumeric codes

Fiscal Calendar Record

The Fiscal Calendar record defines the posting dates that the *voucher* program can use. The institution must add a new Fiscal Calendar record every year, prior to the start of the new fiscal year.

For each amount type (ACT, BGT, ENC), you must define the posting dates for the fifteen standard posting periods for each fiscal year. In addition to the twelve calendar months, the General Ledger module uses a balance forward period (BAL), an adjustment period (ADJ), and a closing period (CLS).

Important considerations when modifying or adding Fiscal Calendar records are as follows:

- When entering information into the Fiscal Calendar record for the ledger type G/L, the prd_no must correspond to the fifteen posting periods. Valid posting period numbers are 00 - 14.
- The first four characters of the prd field are the fiscal year (e.g., 9596), and the last two characters are the corresponding period number. For example, the balance period (BAL) for fiscal year 9697 has a period of 969700, the July posting period has a period of 969701, the Adjustment posting period has a period of 969713, and the closing period (CLS) has a period of 969714.
- For subsidiaries, enter **0** in the prd_no field.

The following list describes all the fields in the Fiscal Calendar record:

Amount Type

The amount type (e.g., ACT, BGT, ENC) associated with the fiscal period.

Beginning Date

The beginning date of the fiscal period (e.g., 01/01/97 for January).

Close Balance

A flag indicating whether you want to close Subsidiary Balance records (subb_rec) with a zero balance. Valid codes are as follows:

- A (always close subb_rec when the balance is zero)
- C (close subb_rec when the balance is zero and the posting period is closed)
- N (never close the subb_rec)

During setup, use the following guidelines for completing the Close Balance field:

- For the subs type G/L, enter **N** in the field.
- For the subs type W/P, enter **A** in the field.
- For the subs type S/A, enter **C** in the field.
- For the subs type A/P, enter **A** in the field.

Closed Date

The last date that the period can be used for posting (e.g., 02/28/97 for January, if you want to permit posting for one month after the end of the period).

Ending Date

The ending date of the fiscal period (e.g., 01/31/97 for January).

Fiscal Year

The fiscal year (e.g., 9697). Designate the fiscal year 1999-2000 as 9900.

Ledger

The name of the ledger. Designate the general ledger as G/L. Designate subsidiary ledgers as the four-character code in the Subsidiary table.

Month/Period

The fiscal period code (e.g., BAL, JAN, ADJ).

Opened Date

The first date that the period can be used for posting (e.g., 12/01/96 for January, if you want to permit posting one month in advance).

Period Number

The period, from the base period of 0 used for Balances Forward, used by *voucher* to determine the posting period for the gl_amt_rec. For a July-to-June fiscal year, values include the following:

- 1 July
- 2 August

.
.
.

- 12 December

Period

The six-character code for the fiscal period (e.g., FA96). Subsidiary ledgers use the period codes rather than fiscal months. All allowable periods must appear in entries in this field.

Note: Complete this field for all Fiscal Calendar records, even for subsidiaries that do not use sessions or periods. Use the fiscal year followed by two zeroes (e.g., 969700).

Priority

For overlapping sessions, a code indicating the posting order for transactions. For example, in Student Accounts, you may want to permit early or late payments to occur. To permit such payments, you establish overlapping sessions (e.g., Fall 1996 and Spring 1997 are both open for posting for some period of time). During the overlapping time period, the lowest priority code is posted first. To post to Fall 1996 first, assign it a code of 1, then assign Spring 1997 a code of 2. For G/L, assign a code of 0.

Document Table

Most institutions modify the Document table to contain their unique codes and sequence numbers.

The Document table defines the document types that the institution uses. It also assigns numbering sequences to the document types, and the stations that use the documents. Each station should have a unique numbering sequence to correspond with the physical documents.

Setup issues for the Document table include the following:

- The Document table controls the form printing, the form numbering and the journals that are permitted to post using a selected document code. In check writing (A/P and W/P subsidiaries) it also defines the General Ledger cash account to use in the posting process.
- For the gift receipt (GR) document type, the void_prog (Void Subprogram) field must contain “grvoid.”
- Valid pmt_modes (payment modes) are as follows:
  - B (purchasing with a bill instead of a purchase order)
  - C (check)
  - D (direct deposit)
- During setup, you define the beginning form numbers for Cash Receipts (CR) and Purchase Orders (PO) for each station that uses these forms.

The following list describes the fields in the Document table:

Acct

The bank account number.

Amount Type

The three-character amount type code (e.g., ACT, BGT or ENC) to which you want to restrict the use of the document type.

Automatic Increment

A Y/N field indicating whether the user wants the program to automatically increment document numbers.

Code

The two-character user-defined code for the document type. This field, along with the Operator Station, serves as the primary key for the table.

Note: When you create a new document code that you want to use within General Ledger, you *must* define it for Operator Station 0. You can also define it for any other stations as needed.

You can also enter a 24-character description of the code in the Document Table screen.

Document Number Issued

A Y/N field indicating whether the user wants the program to issue document numbers for the station.

Note: Set this value to Y only if Automatic Increment is also set to Y.

Financial Institution ID

The ID number of the bank associated with this form.

Note: You can also enter the name of a contact person at the bank on the Document Table screen.

Form Layout

A fourteen-character field containing the name of the form file. The form is located in $CARSPATH/modules/forms/accounting/ voucher.

Form Printed

A Y/N field indicating whether the program must print a form when using the specified document type. Enter the name of the form type file in the Form Layout field.

Note: CX currently supports cash receipts only.

G/L Acct

The general ledger account (fund, function, object, and subfund) associated with the document type.

Note: The program uses this account when posting checks.

Journals Allowed for Document

A series of Y/N fields that enable the document type to be used with the specified journal type.

Number Last Issued

The system-maintained number of the last document issued (if the Document Number Issued field is set to Y) or the last document number used (if the Document Number Issued field is set to N).

Operator Station

The station number for the journal type. This field, along with the Code, serves as the primary key for the table.

Note: A station of zero (0) is unrestricted. Any number of users can use station zero. The program restricts station numbers of 1 or more to only one user at a time. This feature is particularly useful in Cashier, where each cashier terminal is assigned a different station with its own unique sequence of receipt numbers.

The program assigns a station to a journal when the journal is started. For security purposes, all future activity in the journal must originate from this station. The exception to this security measure is when a journal is started from station zero.

Pay Mode

The type of payment being made. Valid values are as follows:

- C (checks)
- D (direct deposit)

Reserved by User

The UNIX user ID number from etc/passwd for the user who is currently using the document station. If the station number is zero (0), then any number of users can use the station and the system does not use this field.

Transit

The bank transit number.

Void Sub Program

The name of the program that performs additional voiding actions after the *docvoid* program has reversed the general ledger entries for a document. Currently only *grvoid* (Gift Receipt Voiding) is supported. Leave this field blank for all document types that do not relate to gift receipts.

General Ledger Tables That May Not Require Modification

Introduction

The following tables are important to the successful use of General Ledger. As delivered, however, they usually contain values that do not require modification. Modification may be required as the applications to which the tables most closely relate are implemented.

During General Ledger implementation, you should review the contents of these tables to ensure they contain values that are compatible with your future use of the CX financial applications.

Amount Type Table

The Amount Type table defines the types of dollar amounts that users at the institution enter into the CX database. Most institutions do not need to change the values in the table.

Standard CX contains the Amount Type table with the following types:

- ACT
- BGT
- ENC

Center Combination and General Ledger Association Tables

The Center Combination table (cntrcomb_table) and the General Ledger Association table (glas_table) both define groupings of information that you want to consolidate on reports. Since the tables relate to reporting only, they are optional.

The Center Combination table defines groupings of functions. For example, if you need a financial report about the Performing Arts departments, you can define a group that includes the functions for Music, Theater and Dance. After you define the groupings of functions that you want (e.g., PA for Performing Arts), you enter the function codes that contain the desired information on the Center Combination records.

The General Ledger Association table defines groupings of objects. For example, if you need a financial report about the bookstore, you can define a group that includes all the accounts relating to the bookstore (e.g., books, supplies, apparel, and gift items). After you define the groupings of objects that you want, you enter the accounts that contain the desired information on the General Ledger Association records.

Entry Type Table

The Entry Type table contains the following information:

- Valid entry types
- Default payment form codes
- Journal types with which you can use the entry type
- Number of cash transactions you can use with the entry type
- Number of subsidiary transactions you can use with the entry type

For example, you can set up a CASH entry type that allows only debits to the institution's cash account by entering **D** in the Cash field on the Entry Type Table screen (the ca_tr_perm field in ent_table). You can complete the CASH entry type with an **I** in the S/L field on the Entry Type Table screen (the subs_tr_perm field in the ent_table), enabling an unlimited number of subsidiaries to be credited.

The following list describes the fields in the Entry Type table:

Amount Type

The three-character code for the amount type associated with the entry type (e.g., ACT, ENC, BGT).

Cash
An alphabetic code that indicates the types of cash transactions you want to permit for the entry type.

Note: For more information about valid codes, see the table at the end of this list.

Debit or Credit to A/P
A code (D for debit, C for credit) that the subsidiary logic uses to determine how to post a non-invoiced accounts payable application in an automatic mode.

Debit or Credit to A/R
A code (D for debit, C for credit) that the subsidiary logic uses to determine how to post an account receivable in this entry type.

Default Description
A 24-character description of the entry type.

Entry Type Code
The four-character code that defines the entry type. Most codes are an abbreviation for the type of entry (e.g., CASH for cash receipts or DISB for cash disbursements).

Ignore: Constraints
A Y/N code that enables users to post to closed periods. For example, a Y in this field enables users to record a student's prepayment of some charges for an upcoming session that is not yet open. A Y in this field also enables *voucher* to open closed subb_recs.

Ignore: Subsidiaries
A Y/N code that enables *voucher* to allow transactions to be made against the subsidiary control account without making transactions against subsidiaries.
Set this value to Y only for the balance forward entry type

Itemize on Statement
A code that indicates how you want to display entry information on statements. Valid codes are as follows:

- I (ignore: causes entries with this entry type to be completely ignored by the statement program. Use this code only for entries that are known to balance to zero (e.g., some types of payroll funding codes.)
- N (no itemization: causes the entry to be summarized with one line on a statement, using the description from the sube_rec.)
- T (transactions: prints each transaction for an entry without summarizing them. Use this code for entry types that have only one transaction for each entry and which are adequately described on one line (e.g., balance forward entries).)
- Y (itemize entry: causes the entry to print in detail. The subsidiary entry line prints, followed by each subsidiary transaction on the next lines.)

Journal Allowed to Post Entry
A Y/N flag that controls the journals that can use the entry type. A journal can use a given entry type only if the permission flag for that journal contains Y. For example, for the CASH entry type, you would set the flag for the ch journal to Y.

Maximum Transactions
The maximum number of transactions that you want to permit for the entry type.
The maximum is a reminder only, not a fixed limitation.

Payment Form
The defaulting pay form code for the entry type.

S/L

An alphabetic code that indicates the types of subsidiary transactions you want to permit for the entry type.

Note: For more information about valid codes, see the table at the end of this list.

Subsidiary Post

A code (INV for invoice or PAY for payment) that determines which side of a subt_record is affected by the entry type.

CAUTION: Institutions should not modify this code. Use the codes as they appear in the CX standard product.

Cash and S/L Codes

The Cash and S/L fields in the Entry Type table contain an alphabetic code that indicates the number of debit or credit entries you want to permit. The following table lists the codes and their meanings:

Code	Meaning	Number of debits	Number of credits
B	one and only one debit and credit	1	1
C	one and only one credit, no debits	0	1
D	one and only one debit, no credits	1	0
E	at the most, one debit or credit	0-1	0-1
F	at the most, one credit, no debits	0	0-1
G	at the most, one debit, no credits	0-1	0
H	no restrictions on debits or credits	0<	0<
I	no restrictions on credits, no debits	0	0<
J	no restrictions on debits, no credits	0<	0
K	at least one debit and one credit	1<	1<
L	at least one credit, no debits	0	1<
M	at least one debit, no credits	1<	0
N	no debits or credits	0	0

General Ledger Association Table

The General Ledger Association table links accounts for reporting purposes only and is an optional part of General Ledger setup. For more information about the General Ledger Association table, see *Center Combination and General Ledger Association Tables and Records* in this section.

Journal Table

The Journal table contains all the allowable journal types. Because the types are used by hard-coded processes in the system, institutions must not modify this table except to change the default document type (def_doc) or the default entry type (def_ent_type), the defaults that appear when users access the *voucher* program.

Important fields in the Journal table are as follows:

Default Entry Type

The four-character entry type code that serves as the default for the journal type. The Entry Type table defines valid entry type codes.

Default Document

The two-character document type that serves as the default for the journal type. The Document table defines valid document types.

Group ID

The group ID number from the file in /etc/groups, indicating who can access the journal type.

Mode of Operation

One of the modes of *voucher* under which the journal can be run, including the following:

- 2 (interactive)
- 3 (semi-interactive)
- 4 (background)

Required Amt Type

The three-character code that indicates the amount type required for a given journal type (e.g., ACT for Actual). Only ACT and ENC can be used in subsidiary ledgers.

The following list contains all the valid journal types and their purposes within the *voucher* program:

- AC (accounting)
- AP (accounts payable)
- AR (accounts receivable)
- BG (budget)
- CH (cashier)
- CK (check posting)
- DA (donor accounting)
- FA (financial aid)
- JC (job costing)
- PC (purchasing)
- PD (payroll cash disbursement)
- PR (payroll)
- PS (personnel)
- QC (quick check)
- SA (student accounts)
- SB (student billing)

Note: The following journals are not used within CX and can serve any purpose that the institution requires:

- PS
- AR
- JC

Subsidiary Association Table

The Subsidiary Association table (subas_table) validates combinations of subsidiary balances (bals) and totals (tots), and links the combinations to the subsidiary. Before you can successfully complete the implementation of the Cashier application, you must have the correct information in this table.

Subsidiary Table

The Subsidiary table (subs_table) defines the subsidiaries that the institution uses. Most institutions use at least three subsidiaries: A/P (Accounts Payable), S/A (Student Accounts), and W/P (Wages Payable). Other subsidiaries appear in the standard Subsidiary table, and you should remove them if the institution does not use them.

Important considerations when modifying or adding Subsidiary table entries are as follows:

- The Subsidiary table must exist for the Voucher program to load.
- You cannot use A/P on two accounts. If you want to use more than one account with Accounts Payable, you must set up another subsidiary, (e.g., AA/P).
- The A/P subsidiary uses IDs and bals only.

- The S/A subsidiary uses IDs, bals and tots.
- The W/P subsidiary uses IDs, bals and tots.
- Subsidiaries such as Room Deposits (R/D) use IDs only.
- The Subsidiary Mode (subs_mode) must be **S** (standard) for all subsidiaries except A/P. Use a Subsidiary Mode of **I** for all A/P subsidiaries.
- Enter **Y** in the ID Number Used (id_no_used) field for every subsidiary.

The following list describes all the fields in the Subsidiary table:

Allow Bursar Display
A Y/N field that enables the *bursar* program to display a subsidiary.

Allow Cashier Post
A Y/N field that enables the *cashier* program to post to the subsidiary.

Default Alternate Address
A four-character code from the Alternate Address table that indicates the type of alternate address to use (e.g., PERM, SUMM).

Default Assoc
A Y/N field indicating whether the amount entered against a general ledger control account is to be prorated visually against the subt_recs of the subsidiary ledger.

Default Credit Rating
A two-character code for the default credit rating.

Default Discount
A two-character code for the default discount rate.

Default Dunning Letter
An eight-character code that denotes the default dunning letter to use for subsidiaries in the ledger.

Default Entries
A Y/N field indicating whether the amount entered against a general ledger control account is to be prorated against the subt_recs of the subsidiary ledger. The program displays the proration only if the Default Assoc code is set to Y.

Default Interest Waived
A Y/N field indicating whether interest is to be waived for the students' accounts.

Default Office for Check
A four-character code from the Office table that defines the default office to receive checks relating to the subsidiaries in the ledger.

Default Payment Terms
The default payment terms code from the Payment Terms table.

Default Statement Form
An eight-character code that defines the statement form used with the subsidiary.

Discount
The account (fund, function, object and subfund) against which you want the system to charge any discounts.

Function
The function component of the general ledger control account associated with the subsidiary ledger. The system uses the control account in background mode where the subsidiary entries and transactions are created before *voucher* automatically generates any general ledger transactions. The function must be valid in the General Ledger tables.

Fund

The fund component of the general ledger control account associated with the subsidiary ledger. The system uses the control account in background mode where the subsidiary entries and transactions are created before *voucher* automatically generates any general ledger transactions. The fund must be valid in the General Ledger tables.

ID No Used

A Y/N field indicating whether the user-entered subsidiary ID number must exist in the ID file.

Invoice Balances Used

A Y/N field indicating whether the subsidiary ledger can have balance and total records distinguishing between charging (inv) and paying (pay). For an example of using this code, see *Example of Using Invoice and Standard Balances and Totals* at the end of this list.

Invoice Totals Used

A Y/N field indicating whether the subsidiary ledger can have balance and total records distinguishing between charging (inv) and paying (pay). For an example of using this code, see *Example of Using Invoice and Standard Balances and Totals* at the end of this list.

Mode

A one-character code that defines the type of subsidiary ledger. Valid values include the following:

- I (invoice)
- S (standard)

Object

The object component of the general ledger control account associated with the subsidiary ledger. The system uses the control account in background mode where the subsidiary entries and transactions are created before *voucher* automatically generates any general ledger transactions. The object must be valid in the General Ledger tables.

Payment Subsidiary

Currently not in use.

Standard Balances Used

A Y/N field indicating whether the subsidiary ledger can have balance and total records distinguishing between charging (inv) and paying (pay). For an example of using this code, see *Example of Using Invoice and Standard Balances and Totals* at the end of this list.

Standard Totals Used

A Y/N field indicating whether the subsidiary ledger can have balance and total records distinguishing between charging (inv) and paying (pay). For an example of using this code, see *Example of Using Invoice and Standard Balances and Totals* at the end of this list.

Student Account

A Y/N field identifying whether a subsidiary pertains to student accounts. A Student Accounts subsidiary is a subsidiary that uses balance and total records for recording student billing charges, financial aid received, cash payments and any other miscellaneous account adjustments in each session.

Subfund

The subfund component of the general ledger control account associated with the subsidiary ledger. The system uses the control account in background mode where the subsidiary entries and transactions are created before *voucher* automatically generates any general ledger transactions. The subfund must be valid in the General Ledger tables.

Subsidiary

A four-character code for the name of a particular subsidiary ledger (e.g., S/A (student accounts) or K/D (key deposits)). You can also enter a description of the subsidiary in the Subsidiary Table screen.

Subsidiary Type

A three-character code that indicates whether the subsidiary ledger relates to accounts payable or accounts receivable, and determines debit and credit codes.

Example of Using Invoice and Standard Balances and Totals

The following four fields control balance and total processing for invoices and payments:

- Invoice Balances Used
- Invoice Totals Used
- Standard Balances Used
- Standard Totals Used

To allow charges and payments to be applied to individual total records (e.g., for the S/A subsidiary), set all the flags to Y.

To allow charges and payments to be applied to subsidiary records only (e.g., for the K/D subsidiary), set all the flags to N.

Subsidiary Balance Table

The Subsidiary Balance table defines the balances (bals) that the institution uses. Bals define a subset of subsidiary transactions. For example, in Accounts Payable, the institution must track the vendor name and the invoice number. Bals provide a way to track this type of information, and since no specific sessions must be tracked for Accounts Payable, it is not necessary to use Tots (as defined in the Subsidiary Total table).

Note: The Subsidiary Balance table provides information to the subsidiary logic of *voucher*, including the order to display subb_recs and which subb_recs relate to the selected subsidiary.

The following list describes all the fields in the Subsidiary Balance table:

Balance Code

A four-character code that identifies the subb_record (e.g., SB, used with the S/A subsidiary ledger).

Priority

A two-character numeric value that indicates the priority in which the system displays the subb_recs. The higher the number, the lower the priority. If you do not specify a number, the subb_recs appear in an oldest-to-newest sequence. If you do specify a number, the system displays records by priority, and then by date.

Subsidiary

The four-character code of the subsidiary ledger with which the subb_rec is associated (e.g., S/A is associated with the SB balance code).

Waive Interest

A Y/N field indicating whether you want to waive interest for the specified balance code.

Subsidiary Total Table

The Subsidiary Total table defines the totals (tots) that the institution uses. Tots define a specific type of transaction to the subsidiary. For example, in Student Accounts, the institution must track the student name, the total amount owed, the session(s) to which the owed amount(s) relate, and the type of charge. Tots provide a way to track the type of charge (e.g., tuition), any credits to the student's account (e.g., grants or scholarships), and the dollar amount of each charge or credit.

Most institutions implement the Subsidiary Total table during their setup of Student Billing. However, if General Ledger is the first module to be implemented at the institution, enter a tot of PAID. On the Subsidiary Total table entry for PAID, consider the following:

- Do not specify an account for the PAID tot.
- Use the PAID tot in Mode 2 (G/L posting mode) journals only. The primary Mode 2 journal is Cashier (CH); the Accounting (AC) journal is also Mode 2. Set the Lost flag to Y, unless the tot refers to a non-refundable fee.
- Set the Post flag to Y.

The following list describes the most important fields in the Subsidiary Total table:

Lost

A Lost tot refers to any charge that was posted at one time by Student Billing, but in a subsequent run, the Student Billing program cannot justify why the charge was posted to the student's account.

A more precise name for a Lost tot is Unlinked tot. For Student Billing to calculate a charge, the system must have both an Assessment table and a Charge table for the charge. In addition, the Assessment table must allow the student to be charged a given fee. If at some time, an Assessment table is changed or the student no longer satisfies conditions within an Assessment table, the Student Billing program cannot link from the Assessment to the Charge table to calculate a fee. Therefore, the calculated charge becomes lost or "unlinked." Student Billing refunds the charge at 100% if the Lost flag in the Subsidiary Total table is N.

Post

A one-character field containing one of the following codes:

- A (Financial Aid code). All Financial aid tot codes that you post through the Financial Aid Entry program (Faentry) must contain a Post flag set to A. In the case where a student is considered a no-show, the Student Billing program will not attempt to reverse the work study tots when the Post flag is set to A.
- B (Automatic Student Billing tot code) Use the B Post flag for student billing charges that the Student Billing program calculates based on the entries in the Charge table. Any tot with a B Post flag appears in the charges column on the Student Data Sheet.
- C (Manual Student Billing Charges) Use the C Post flag for any manual charge that the institution wants to reverse in the billing process if the student is a no-show, or for student billing charges that the institution enters through an SA journal. The C Post flag differs from the M Post flag in that a tot with a C Post flag displays in the charges column of the Student Data Sheet and can be refunded to the student if the Lost flag is set to Y.
- D (Display Manual Student Billing Charges) Use the D Post flag when the institution wants to display manual charges on the Student Data Sheet, and when the amounts are not refundable to students who are no-shows. To work properly, the D Post flag can only appear on tots that have the Lost flag set to Y.
- L (Financial Aid Loans) Use the L Post flag to identify tot codes that record the amount of financial aid loan check payments. For example, Stafford loans can post to a separate tot code for reporting and display purposes. The L Post flag, in contrast to the A Post flag, allows these types of credits to print in the Payments/Credits section of the Bursar screens.
- M (Manual Charges/Refunds) Use the M Post flag for any totally manual charge or refund posted to student accounts. The Student Billing program ignores tot codes with this value.
- I (Interest (optional)) Use the I Post flag in the case where the institution does not want Student Billing to reverse interest charges for students who are no-shows.
- Y (Cash Payment Tot Code (e.g., PAID)) Use the Y Post flag for tot codes to which cash payments are applied. The Student Billing program uses this flag to calculate and print the cash payments on the Student Data Sheet.

- P (Print Deduction of Check Stub (Payroll)) Use a P Post flag for wages payable deductions that must print on the check stub.
- N (Do Not Print Deduction on Check Stub (Payroll)) Use an N Post flag for. wages payable deductions that must not print on the check stub.

Priority

A two-character numeric value that indicates the priority in which the system displays the subt_recs. The higher the number, the lower the priority. If you do not specify a number, the subt_recs appear in an oldest-to-newest sequence. If you do specify a number, the system displays records by priority, and then by date.

Program

A four-character code that associates student account charge codes to an academic program code. You must only enter a value in this field if the Student Billing program is used to charge students enrolled in more than one academic program. When billing students who are enrolled in more than one program, this field associates charge/tot codes to a specific program. If the subsidiary total is not used in student billing, leave the field blank.

Note: The system ignores this field unless the Post code contains one of the following values: B, C, D, L or Y.

Subsidiary

A four-character code of the subsidiary to which the subt_rec is associated.

Total Code

A four-character code that describes the type of subt_rec (e.g., TUIT represents Tuition under the S/A subsidiary).

Considerations for Fee Collection

If you expect to use the Fee Collection application, note that a Total Association (totas) code must be associated with only one *Charge* type of Tot code (i.e., a Tot code with a Post code of B, D, or M that results in the posting of a debit). If you associate a totas code with multiple Tot codes that are the Charge type, the Fee Collection application will only apply fees against the first of the Tot codes.

Example: The following subt_table values are correct for use with Fee Collection:

Tot code	Description	(Debit or Credit?) (Not a field in the subt_table)	Post	Totas code
TUIT	Tuition	Debit	B, D, or M	TUIT
WTUI	Tuition waiver	Credit	B, D, or M	TUIT

In this example, since the WTUI code calls for a credit (as a contra-charge to Student Accounts Receivable), you can use the same totas code for both of these entries.

In contrast, the following subt_table values are *incorrect and will result in inaccurate posting* :

Tot code	Description	(Debit or Credit?) (Not a field in the subt_table)	Post	Totas code
TUIT	Tuition	Debit	B, D, or M	TUIT

Tot code	Description	(Debit or Credit?) (Not a field in the subt_table)	Post	Totas code
BOOK	Books	Debit	B, D, or M	TUIT

In this example, since both tot codes call for debits (as charges to Student Accounts Receivable), you cannot use the same totas code for both entries. In this incorrect table setup, assuming the TUIT code has the higher priority, the total amount applied by Fee Collection will be applied against the TUIT code, and no amounts will be applied against the BOOK code. The correct setup to ensure Fee Collection applies amounts correctly follows:

Tot code	Description	(Debit or Credit?) (Not a field in the subt_table)	Post	Totas code
TUIT	Tuition	Debit	B, D, or M	TUIT
BOOK	Books	Debit	B, D, or M	BOOK

Establishing Cash Accounts

Introduction

After the Financial tables are complete and contain the institution's codes and information, you must establish cash accounts, since the system handles cash transactions in special ways. For example, the Cashier application, when operating in Receipt mode, operates with debit entries to the cash account.

How to Establish Cash Accounts

The following procedure establishes cash accounts using the Voucher program. You must use this procedure for every cash account at the institution (e.g., Accounts Payable cash, Payroll cash, and the Cash Drawer account).

Note: Before you define the account as a cash account, you must set it up as a valid account in the Chart of Accounts tables.

1. From the Fiscal: Accounting menu, select Journal Processing. Then, select Accounting Entry. This selection loads *voucher*.
2. Select G/L.
3. Enter the number of a Cash account, then enter the following information about the account:
   - The Net Asset Indicator in the Ast field to reflect if the Cash account is (U)nrestricted, (T)emporarily restricted, or (P)ermanently restricted.
   - The Prior Account Number (if the institution wants to track the number from a previous system).
   - The Summarize flag set to **Y**.
   - The Cash Acct flag set to **Y**.
   - The Journals Allowed flags set to **Y** for the following journal types:
     - AC
     - CH
     - CK
     - DA
     - PD
     - QC

Note: To change an account that is already activated, reverse the account that was used in error, then terminate the account using the following menu selections:
- Accounting
- G/L Maintenance menu
- Terminate G/L Accounts

Establishing Due to/from Accounts

Introduction

The General Ledger setup includes defining due to/from accounts. The system uses these accounts for interfund transactions.

How to Establish Due To/From Accounts

1. Define the due to/from accounts in the General Ledger tables. The system automatically sets all flags and indicators for the due to/from accounts when the accounts are used, so no other table setup is required.
2. Define the interfund account structure in the *gl_interfund* macro.

Establishing Subsidiary Control Accounts

Introduction

The General Ledger setup includes defining subsidiary control accounts. The system uses these accounts as temporary holding accounts for amounts that have not yet been charged to the correct subsidiary, and to show beginning balances at the time of implementation.

How to Establish Subsidiary Control Accounts

The following procedure establishes subsidiary control accounts using the Voucher program.

Note: Before you define the account as a subsidiary control account, you must set it up as a valid account in the Chart of Accounts tables.

1. From the Fiscal: Accounting menu, select Journal Processing. Then, select Accounting Entry. This selection loads *voucher*.
2. Select G/L.
3. Enter the number of a subsidiary control account, then enter the following information about the account:
   - The Net Asset Indicator in the Ast field to reflect if the Due to/from account is (U)nrestricted, (T)emporarily restricted, or (P)ermanently restricted.
   - The Prior Account Number (if the institution wants to track the number from a previous system).
   - The Summarize flag set to **Y** (unless you do not have a large number of transactions for the subsidiary).
   - The Cash Acct flag set to **N**.
   - The Journals Allowed flags set to **Y** for the following journal types:
     - AC
     - CH
     - CK
     - PD
     - QC

Establishing Donor Holding Accounts

Introduction

The General Ledger setup includes defining donor holding accounts. The system uses these accounts as temporary holding accounts for amounts that have not yet been applied to the correct donor.

How To Establish Donor Holding Accounts

The following procedure establishes donor holding accounts using the Voucher program.

Note: Before you define the account as a donor holding account, you must set it up as a valid account in the Chart of Accounts tables.

1. From the Fiscal: Accounting menu, select Journal Processing. Then, select Accounting Entry. This selection loads *voucher*.
2. Select G/L.
3. Enter the number of a donor holding account, then enter the following information about the account:
   - The Net Asset Indicator in the Ast field to reflect if the donor holding account is (U)nrestricted, (T)emporarily restricted, or (P)ermanently restricted.
   - The Prior Account Number (if the institution wants to track the number from a previous system).
   - The Summarize flag set to **N**.
   - The Cash Acct flag set to **N**.
   - The Journals Allowed flags set to **Y** for the following journal types:
     - AC
     - CH
     - CK
     - PD
     - QC

Resetting Journal Numbers

Introduction

If the institution trains users on a live database, the trainees will create journals and journal numbers during the training. To begin live use of CX with journals starting with "1," you must reset the journal numbers in the Voucher table. SQL statements can reset the numbers.

CAUTION: Do not reset any journal numbers after implementation. *This is a setup consideration only.*

How to Reset Table Sequence Numbers for an Entire Table

If you need to reset the sequence numbers in a table, use the following SQL statement at the UNIX prompt:

% update vch_table set vch_table.last_issued_no = "0"

How To Reset Table Sequence Numbers for a Portion of a Table

If you only need to reset the sequence numbers in a portion of a table, (e.g., only for a journal type of AC in the Voucher table (vch_table), use an SQL statement in the same format). The following is an example:

% update vch_table set vch _table.last_issued_no = "0" where vch_table.jrnl = "AC"

Resolving Security Issues

Introduction

The General Ledger module contains several security features, since financial transactions can be sensitive or subject to misuse. Jenzabar suggests that institutions use the following security measures:

- Restrict menu access
- Add passwords to menus
- Limit permissions for accessing programs and accounts

How to Restrict Menu Access

To restrict menu access, you modify the menu users' .login file by entering an exec command into the .login file. After the keyword "exec," enter the path from the menusrc directory to which you want to limit access. For example, to limit menu access to the Fiscal Management Main menu, add the following line to the end of the users' .login files:

'exec menu fiscal', 'logout'

To limit menu access to the Cashier Main menu, add the following line to the end of the users' .login files:

'exec menu fiscal/cashier', 'logout'

Suggested .login Entries for Limiting Menu Access

Jenzabar suggests that you limit menu access using these guidelines:

V/P of Business:
.login entry: menu fiscal
Entry menu: Fiscal Management

Business Manager/Controller:
.login entry: menu fiscal
Entry menu: Fiscal Management

Budget Director:
.login entry: menu fiscal/finbgt
Entry menu: Financial Budgeting

Cashier:
.login entry: menu fiscal/cashier
Entry menu: Cashier

Accounts Receivable clerk:
.login entry: menu fiscal/acctsrecv
Entry menu: Accounts Receivable

Accounts Payable clerk:
.login entry: menu fiscal/acctspay
Entry menu: Accounts Payable

Purchasing clerk:
.login entry: menu fiscal/acctspay/purchasing
Entry menu: A/P Purchasing

Payroll supervisor:
.login entry: menu fiscal/payroll
Entry menu: Processing menu

Payroll clerk:
.login entry: menu fiscal/payroll/prprocess
Entry menu: Processing menu

Personnel clerk:
.login entry: menu fiscal/payroll/employee
Entry menu: Data Display/Entry

How to Add Passwords to Menus

Another security measure is to place password protection on a menu. Only those menu users who know the password can access the menu. Other menu users receive the message "Invalid Password" and cannot proceed to the protected menu.

The following example procedure, initiated at the UNIX prompt, adds the password *keepout* to the menu file for the General Ledger Maintenance menu. Use the same procedure for the other menusrc files that you want to restrict with a password.

1. Access the correct menusrc directory, (e.g., $CARSPATH/menusrc/fiscal/finacctg/glmaint).
2. Enter the following command to check out the menudesc file:
   make co F=menudesc
3. Edit the menudesc file, inserting the line **PW=keepout**. For convenience, insert the line near the top of the menudesc file.
4. Exit from the file, saving your changes.
5. Enter the following command to check in the menudesc file:
   make ci F=menudesc, L=added password to menu

How to Set a Standard Password for a Group of Menus

If you want to use the same password for a group of menus (e.g., all the General Ledger menus), you can use a keyword in the menusrc file and then define the keyword value in the passwd.s file, located in system/etc. If you use this method of setting passwords, you can change the password in a single location (the passwd.s file) and have the new password control all the menus that use the keyword.

Follow these steps to set a standard password:

1. Check out and access the menudesc file for the menu that you want to restrict with the standard password.
2. Enter the following line in the menudesc file:
   PW = @FISCAL
3. Exit the file and save your changes.
4. Access the following file: system/etc/menupw.s
5. Enter the following line in the menupw.s file:
   FISCAL = standard password
6. Exit the file and save your changes.

After you set the standard password in the menupw.s file, repeat steps 4 - 6 whenever you want to change the password on the menus that use FISCAL to define the password.

Limiting Permissions for Accessing Programs

Tables That Limit Permissions

Two tables, the Permission table (perm_table) and the General Ledger Permission table (glperm_table), enable you to limit program access for specific users.

The Permission table links individuals or groups to particular processes or product areas. The General Ledger Permission table links accounts to processes or product areas, therefore enabling institutions to limit the accounts that users can view or use for posting.

Interrelationships of the Tables

The Permission table and the General Ledger Permission table *must* work together to establish the security permissions for the institution. Neither table alone can provide the permissions you need.

General Ledger Permission Table

The General Ledger Permission table defines the account numbers that a user can access. The Permission code that appears in both the Permission table and the General Ledger Permission table links the table entries. Therefore, at the Permission table level, you define access to programs; and at the General Ledger Permission table level, you define access to accounts within the programs.

You can mask any of the account number component fields when you complete the General Ledger Permission table.

Permission Table

The purpose of the Permission table is to grant or deny a user access to information at the application or process level. For example, changes to the Permission table control access to *acquery* or *voucher*.

Examples of Table Setups

This section includes illustrations of how the General Ledger Permission table and the Permission table work together to establish permissions for access to applications and accounts.

Example General Ledger Permission Table Entry

Each of the following examples relies on the following General Ledger Permission table entry:

Permission Code: ALL

Fund: **

Function: ****

Object: ****

Subfund: ****

Exclude Permission: N

Using the Permission Table to Grant Permissions to a Group

To grant a group all General Ledger permissions, requires a single table entry for the UNIX user id number for the group. To grant permissions, enter the following information in the Permission table:

Program or Process: blank

Category: GLPERM

Permission Code: ALL

Group or User Number: *UNIX user id*

Note: Be sure to use the common uid number; as delivered, the common uid number for the General Ledger module is 120.

Type: G

Exclude Permission: N

Using the Permission Table to Deny Permissions to an Individual

To deny a user General Ledger permissions requires either a single table entry or no table entry. The lack of a Permission table entry implies that the user has no permissions. However, you can create a table entry that explicitly denies permission to a user. To deny permissions, enter the following information in the Permission table:

Program or Process: blank

Category: GLPERM

Permission Code: ALL

Group or User Number: *UNIX user id*

Type: U

Exclude Permission: Y

Using the Permission Table to Grant Permissions to a Group

To grant a group all General Ledger permissions requires a single table entry for the UNIX user id number for the group. To grant permissions, enter the following information in the Permission table:

Program or Process: blank

Category: GLPERM

Permission Code: ALL

Group or User Number: *UNIX user id*

Note: Be sure to use the common uid number; as delivered, the common uid number for the General Ledger module is 120.

Type: G

Exclude Permission: N

Using the Permission Table to Grant Permission to a Specific Application

To grant a user permission to use one specific application within General Ledger requires the following Permission table entry:

Program or Process:
<application name, e.g., *acquery*>
Category:
GLPERM
Permission Code:

ALL

Group or User Number:

UNIX user id

Type:

U

Exclude Permission:

N

Using Both Permission Tables to Deny Specific Accounts in a Specific Application

Denying a user access to a specific account (e.g., salary accounts that fall in the 6000-6999 object number range) in a specific application (e.g., *acquery*) requires four table entries (two for the General Ledger Permission table, and two for the Permission table). Example table entries for both tables are as follows:

General Ledger Permission table

Entry 1 Permission Code: ALL

Entry 1 Fund: **

Entry 1 Function: ****

Entry 1 Object: ****

Entry 1 Subfund: ****

Entry 1 Exclude Permission: N

Entry 2 Permission Code: SAL

Entry 2 Fund: 10

Entry 2 Function: ****

Entry 2 Object: 6***

Entry 2 Subfund: ****

Entry 2 Exclude Permission: N

Permission table

Entry 1 Program or Process: ACQUERY

Entry 1 Category: GLPERM

Entry 1 Permission Code: ALL

Entry 1 Group or User Number: <UNIX user id>

Entry 1 Type: U

Entry 1 Exclude Permissions: N

Entry 2 Program or Process: ACQUERY

Entry 2 Category: GLPERM

Entry 2 Permission Code: SAL

Entry 2 Group or User Number: <UNIX user id>

Entry 2 Type: U

Entry 2 Exclude Permissions: Y

Implementing Customized Account Numbers

Introduction

The standard CX account number conforms to the NACUBO account number structure by including Fund, Function, Object and Subfund. The default field lengths for these account number components are as follows:

- Fund (2)
- Function (4)
- Object (4)
- Subfund (4)

To allow for account number flexibility, CX can also support the following non-standard components in the account number:

- Class
- Location
- Sub-class
- Sub-location
- Sub-object
- Sub-type
- Sub-function
- Type

Institutions can vary their account number structures by changing the default lengths, or by using the non-standard components as part of the account number. In addition, some institutions prefer to use the traditional account structure of Fund/Center/Account instead of Fund/Function/Object, or to substitute Project codes for Subfunds if they do not need the self-balancing Subfund feature.

To change the standard account number, you must set macro values, build schemas, and reinstall macros, includes, modules, and source code according to the steps outlined in this section.

How to Implement Customized Account Numbers

The following procedure enables you to customize your account number by changing the lengths of the standard account number components, or by including non-standard components.

Access the macro file and incorporate the appropriate gl_define macros and field lengths into the account number. The macro file contains instructions for making this change.

1. Perform a make tinstall on the macro file.
2. Reinstall the include files by using the following commands:

   cd $CARSPATH/include

   make reinstall F=ALL
3. Build all the schemas that use the account number components by entering the following commands:

% cd $CARSPATH/schema/financial

% make buildy F=ALL

Note: You must also perform a make buildy for the following schemas that do not appear in the schema/financial directory:

- schema/common/tdoc

- schema/development/tdesg
- schema/development/desg
- schema/student/pbilling

4. Reinstall the entire system by entering the following commands:

% cd /$CARSPATH/modules/cvt2sql/scripts

% ./Inst_wrld.scp>&~/Inst.wrld.out &

Note: These commands place you in the correct directory to execute the script *Inst_wrld.scp*, then run the script, routing the standard error output to your home directory, and executing the process in the background.

When you complete this process to add components to the account number structure, the Informix database creates the tables you need to validate the components. For example, if you include location as part of the account number, Informix creates a Location table and a PERFORM screen into which you can enter valid Location codes.

Note: Menus in CX use macros to list the names of the Chart of Accounts tables, and automatically provide menu access to the tables you need for the additional components of your account number.

How to Substitute One Account Component for Another

If your institution would prefer to substitute one account number segment for another, you must change macro values, rebuild schemas, and reinstall components. The following procedure illustrates the changes you must make to substitute the Project code for the Subfund. You can adapt this procedure if you want to change other account number components.

1. Access the macros/custom/financial file.
2. Change the following four macros as indicated:
   - Change 'SUBFUND', 4, 'Sfund' to 'PROJ', 4, 'Proj'.
   - Change GL_SUBFUND_TABLE, 'subfund_table' to GL_SUBFUND_TABLE, 'proj_table'.
   - Change GL_SUBFUND_FIELD, 'subfund' to GL_SUBFUND_FIELD, 'proj'.
   - Change 'JRNLACCT_ORDER', 'fund, func, obj, subfund' to 'JRNLACCT_ORDER', 'fund, func, obj, proj'.
3. If you have an existing Subfund table, use the values in that table to populate the Project table (proj_table). You can use ISQL to unload your subfund_table to a flat file, then load it into the proj_table.
4. Reinstall the macro file.
5. Reinstall all the includes.
6. Reinstall all the macros.
7. Perform a buildy on the schema files.
8. Reinstall the modules.
9. Reinstall the source code.

Entering Budget Information

Introduction

Part of the implementation process can include entering budget information. Typically, implementation involves entering the current year's budget information; in subsequent years, budgeted amounts evolve from the prior year's actual charges to each account.

Program for Entering Budget Information

To enter budget information during implementation, use *voucher*.

Procedure

Follow these steps to enter budget information during implementation.

1. Select the following menu options from the Fiscal Management Main menu:
   - Accounting
   - Journal Processing menu
   - Budget Entry

   The Journal screen in *voucher* appears.

2. Select **Start**.

   Note: If you want to enter the budget information in an existing journal, select **Continue**.

3. Accept the default information that appears on the journal line, then select **Add**. The cursor moves to the entry line of the screen.
4. If desired, you can override the default posting date, using any date/period combination that is valid in the fscl_cal_rec (e.g., the beginning of the year).
5. Check the Entry Type. Is it OBGT?
   - If yes, go to step 9.
   - If no, go to step 6.
6. Terminate the journal and exit *voucher*.
7. Update the Journal table (vch_table) for the BG Journal Reference Code, changing the Default Entry Type to OBGT.
8. Repeat steps 1-6.
9. Enter your CX ID number, or the number of the person responsible for the budget. The name corresponding to the ID number appears on the line, and the cursor moves to the Fd (Fund) field in the transaction area of the screen.
10. Enter the account numbers that you want to debit, and the dollar amounts.
    - If you are entering expense budget items, enter the account numbers for the expenses.
    - If you are entering revenue budget items, enter the budget contingency account number.
11. Enter the the account numbers that you want to credit, then move your cursor to the credit column by pressing **<F3>** (GUI-based) or **<Tab>** (character-based), and enter the dollar amounts.
    - If you are entering revenue budget items, enter the account numbers for the revenues.

- If you are entering expense budget items, enter the budget contingency account number.

12. Select **Finish** to indicate that you have completed the entry.
13. Select **Write** to post the entry.
14. Do you want to add more budget information?
    - If yes, repeat steps 3-13.
    - If no, select **Finish**. The system displays a message about updating the journal, and notifies you when the update is complete.
15. Press **Y** to continue, then select **Bye**.

Setting Up a New Subsidiary

Introduction

Some institutions may need to establish a subsidiary that is not part of CX as delivered. If the implementation requires the setup of a subsidiary, you must review related tables, initialize the accounts or subsidiaries, change the values in macros, and reinstall menu options.

Table Changes

The tables in this section contain information about subsidiaries, and may require changes or additions if you set up a new subsidiary. The tables appear in this section in the order in which you must make changes or additions.

Object Table Changes

A new subsidiary account should use a new object number or at least a new account number combination (i.e., a unique fund/function/object combination). If you use a pre-existing account that has had previous non-subsidiary transactions posted to it, *glaudit* and *saaudit* will produce errors, and the accounts will be difficult to reconcile.

Setup considerations: To set up a new object, enter a new object number in the Object table, including a long and short description.

Defined Accounts Record Changes

The new account number combination that you want to use with the new subsidiary must be valid in the Defined Accounts record (gld_rec).

Setup considerations: To ensure the account is valid, review the table and verify it includes the new account. If a prior operation that used a shortcut code is a part of the new subsidiary, change the account number on the existing code, and use the new account number.

Subsidiary Table Changes

The new subsidiary must be valid in the Subsidiary table (subs_table).

Setup considerations: Add a new entry to this table for the new subsidiary. Review existing subsidiaries for setup hints.

Note: The Voucher program validates the subsidiary code in the Fiscal Calendar record against the codes in the Subsidiary table. Unless the codes are consistent, *voucher* cannot process entries or journals for the subsidiary.

CAUTION: Do not change the account number or name on a subsidiary that has been used in the past or present. If you want to move a subsidiary from one fund to another, you need to close out the old account and transfer any remaining balances to the new account, under the new subsidiary name.

Subsidiary Total Table Changes

If the new subsidiary requires the use of tots, you must define unique tot codes in the Subsidiary Total table.

Setup considerations: A tot code cannot be used in two different subsidiaries (i.e., subsidiaries must have unique tot codes.)

CAUTION: Tots also cannot be switched from one subsidiary to another, or deleted from the table, once they have been used. Do not change any existing entries in the Subsidiary Total table.

Subsidiary Association Table Changes

The Subsidiary Association table links bals and tots to subsidiaries.

Setup considerations: You must link the newly created subsidiary to the new bals and tots.

GL Permissions Table Changes

The GL Permissions table enables you to control the programs that access an account.

Setup considerations: After you add new accounts to accommodate the new subsidiary, you must enable the appropriate programs to access the accounts.

Document Table Changes

The Document table associates a document type with the journals and accounts.

Setup considerations: If the new subsidiary uses any of the accounts in the Document table, then you must change the references to those accounts. In addition, you may need to add a new document type to the Document table. For example, if you add a second A/P subsidiary, you might need to add a document type to use with it.

Cash Entry Table Changes

Used with the Cashier application, the Cash Entry table links a type of cash transaction with an account and subsidiary.

Setup considerations: If the new subsidiary relates to cashiering functions, you must enter the account numbers and subsidiary information on the related codes in the table, or create new entries.

Entry Type Table Changes

The Entry Type table validates the types of entries at use in the institution.

Setup considerations: If the new subsidiary uses an entry type that had not been used with subsidiaries before, you must update the Subsidiary Transaction Permission code to reflect the number of debits and credits you want to permit.

Fiscal Calendar Record Changes

The Fiscal Calendar record validates posting periods for a subsidiary/amount type.

Setup considerations: If the subsidiary uses bals or tots, you must add Fiscal Calendar records. Set up the Fiscal Calendar records for a new subsidiary, using the following guidelines:

- Set the Beginning date (beg_date) and the Ending date (end_date) as the beginning and end of the fiscal year, or of each subsidiary session (requiring an entry for each session)
- Set the following fields to blank:
  - Month (fscl_mo)
  - Period number (prd_no)
- Set the Close Balance (cls_bal) field to C.

Initializing the Subsidiary

When you add a new subsidiary, you must initialize it before you can post entries to it. To perform the initialization, use the following procedure after you update the tables related to the new subsidiary:

1. From the Fiscal: Accounting menu, select Journal Processing. Then, select Accounting Entry. This selection loads *voucher*.
2. Select G/L.
3. Enter the account number, then enter the following information about the account:
   - Set the Summarize flag to **Y** (unless you do not have a large number of transactions for the subsidiary).
   - Set the Cash Acct flag to **N** (unless the account is a cash account).
   - Set the Journals Allowed flags to **Y** for the journal types that you want to use with the subsidiary.

Changing the Values in Macros

When you set up a new subsidiary, you must review the following macros in $CARSPATH/macros/custom/financial. Ensure that you include the new subsidiary where appropriate.

- SUBS_AR_VALID
- SUBS_SA_VALID
- SR_SUBS_VALID

Reinstalling Menu Options

Example: When you set up a new subsidiary, you must reinstall the menu options that use the subsidiary. Reinstall the following menu options:

- $CARSPATH/menuopt/accounting
- $CARSPATH/menuopt/acctspay
- $CARSPATH/menuopt/acctsrecv
- $CARSPATH/menuopt/finaudit
- $CARSPATH/menuopt/stubill

Setting up General Ledger Account Auto-Fill

Introduction

The General Ledger Account Auto-Fill feature provides a table-driven way for CX users to put data directly into one or more segments of the General Ledger account number. Data automatically populates into a certain segment or certain segments when you enter a value into another predetermined segment of the General Ledger account number. This feature saves data entry time. For example, if the subfund segment of the General Ledger account number controls the value in the function segment of the General Ledger account number, the Account Auto-Fill feature can default the appropriate value into the function segment once the user enters a value in the subfund segment.

Configuration Table

If you wish to enable the General Ledger account Auto-Fill feature, you must add an entry to the config_table with the following values:

Product: Financial

Name: GL_ACCT_AUTO_ENTRY

Value: Y

General Ledger Substitution table

Once you have enabled your General Ledger Account Auto-Fill feature, you must add entries to the glsub_table to define your auto-fill strategy. To add entries, access the Accounting: Table Maintenance: Financial (F-Y) Menu, then select General Ledger Account Auto-Fill feature.

To fill the function segment with the value of 1002 automatically whenever you use the subfund 9000, make the following entry:

From G/L Field	subfund
From G/L Value	9000
To G/L Field	func
To G/L Value	1002

Setting Up Claims on Cash, Receivables and Payables

Introduction

Most institutions need to track detailed information about individual departments. CX provides flexible ways to gather this detailed information, including the use of the function code for expenses and revenues, and a variety of reports that show the results of departmental operations.

However, expense and revenue tracking may be insufficient for an institution. Some public institutions must track the effect on cash, receivables and payables (balance sheet accounts) for every transaction that impacts a *local* department (i.e., an operational department on a campus including the Computer Services department or the Athletic department). The CX Claim on Cash, Receivables and Payables feature (hereafter called *claim on cash*) enables institutions to report each local department as if it were a stand-alone entity with balance sheet accounts for cash, receivables and payables.

Comparison of Claim on Cash and Due To/From Accounting

Conceptually, there are similarities between the claim on cash feature and the due to/from accounting feature that has always been an integral part of fund accounting. In CX, both features generate entries to maintain equal debits and credits within self-balancing entities (i.e., funds and subfunds). The generated entries in traditional due to/from accounting result in a single balance for the entity that is either an asset (debit balance) or a liability (credit balance); it is not possible to identify the types of transactions that make up the due to/from balance.

In contrast, claim on cash generates transactions to various accounts, based on entries in the Claim table. The table designates the exact balance sheet account for posting of inter-fund or inter-subfund activity, depending on whether the input transaction impacts cash, receivables (student accounts) or payables. The end result is the ability to identify the type of balance sheet account effected by the inter-fund or inter-subfund activity.

Relevance of Due To/From Accounting with the Claim on Cash Feature

If implemented, both claim on cash transactions and due to/from transactions are created during *bgvoucher* processing. However, the claim on cash processing occurs first. Therefore, if the claim on cash feature is set up correctly, it can completely eliminate the need for due to/from accounting. If the claim on cash feature is not used, is not set up correctly, or if some accounts are omitted from the Claim table during setup, then due to/from accounting will generate any transactions necessary to keep the self-balancing funds or subfunds in balance.

Overall Effect of the Claim on Cash Feature

When institutions use the claim on cash feature, they can produce balanced financial statements at the local department level. The offsetting transactions for expenses and revenues impact *claim* accounts - Claim on Cash, Claim on Receivables, and Claim on Payables. They also create contra transactions that offset accounts in the Clearing House fund (CHF), the fund that contains cash, receivables, and payables.

Making Entries to the Claim on Cash and Contra Claim on Cash Accounts

Users cannot make entries to the Claim on Cash, Receivables, or Payables accounts or to the Contra Claim on Cash, Receivables, or Payables accounts. Only the claim on cash feature creates transactions for these accounts.

When you implement the claim on cash feature by defining the CLAIM_ON_CASH value in the Configuration table, you cause *bgvoucher* to verify that no input transactions are made to either the claim or the contra claim accounts.

Entries that Create Claim on Cash Transactions

The claim on cash feature generates entries under the following conditions:

- When you enable the feature in the Configuration table (config_table)
- When an input transaction matches an account in the Claim table (claim_table)

If you enable the feature, and an inter-fund or inter-subfund input transaction does *not* match an entry in the Claim table, then CX will create due to/from entries as required to maintain a balanced condition in the self-balancing entities. If the institution does not want to use due to/from accounting, then you must set up additional Claim table entries. To correct errors that result from incorrect table setup, see *Maintaining Claims on Cash, Receivables and Payables* in this manual.

Impact on Other Product Areas

Because the claim on cash feature creates journal entries and also impacts receivables and payables, it affects all the financial programs that generate financial transactions. The following programs use the claim on cash configuration parameter and the Claim table:

- *bgvoucher*
- *billing*
- *cashier*
- *ckpost*
- *faentry*
- *fee collection*
- *purch*
- *voucher*

Note:

- The claim on cash feature does not currently affect wages payable. Institutions requiring claim on cash transactions in the wages payable area must create manual entries.
- The fee collection process is required for claim on receivable processing.

Impact from Other Product Areas

Because of timing issues that frequently affect institutions, you must consider the following:

- Claim on cash cannot distinguish between actual cash received and the posting of deferments or financial aid.
- If your institution posts deferments through Third Party Billing, the claim on cash feature will generate all the local departmental journal entries even if the cash has not yet been received from the deferral agency.
- If your institution posts financial aid through *faentry* when it is awarded but before it is received, the claim on cash feature will generate journal entries as if the cash was received.
- You must run Fee Collection to allocate student payments to the related charges before performing claim on cash processing for student accounts.

For these reasons, Jenzabar recommends that you do not post disbursements or receipts until the cash is actually received and deposited by your institution.

Sample Trial Balance Using Claim on Cash

The Clearing House fund maintains the actual Cash, Accounts Receivable and Accounts Payable accounts. It also has contra accounts that reduce the amount of each account within the CHF,

allocating the account balances to other local operating departments at the institution. The net balance of the cash, receivable and payable accounts, offset by their respective contra accounts, is zero. The only occasion on which an account and its contra account will not net to zero is when timing differences arise. For example, in the area of student accounts, if the operator receives and records a cash receipt before the billing has occurred, a prepayment results, creating a claim on cash balance in the CHF.

The following sample trial balances show the impact of the use of claim on cash for the Clearing House fund (CHF) and a local department . Note that the local department does not have actual Cash, Accounts Receivable or Accounts Payable accounts, but it does claim part of the institution's overall cash, accounts receivable and accounts payable. Note also that the CHF balance sheet contains a Claim on Cash balance to offset the Prepaid Tuition Revenue liability.

TRIAL BALANCE FOR THE CLEARING HOUSE FUND	
Debits	
Cash	115000
Claim on cash	3000
Student receivables	119000
Contra claim on payables	122000
Total Debits	359000
Credits	
Contra claim on cash	115000
Contra claim on student receivables	119000
Prepaid tuition revenue	3000
Accounts payable	122000
Total Credits	359000

TRIAL BALANCE FOR THE ATHLETIC DEPARTMENT	
Debits	
Claim on cash	10000
Claim on student receivables	9000
Supplies expense	4000
Salaries expense	70000
Utilities expense	1000
Total Debits	94000
Credits	
Claim on payables	14000
Ticket revenue	40000
Student Fee revenue	15000
Local fund balance	25000
Total Credits	94000

How to Set Up the Claim on Cash Feature

Using claim on cash requires entries in the following tables and records:

Note: For more information about completing the Object table, the Subfund table and the Defined Accounts record, *see General Ledger Tables That Require Modification* in this section.

Configuration table (config_table)

You must define the Current Value of the CLAIM_ON_CASH entry in the Configuration table as Y. The default value is N.

Object table (obj_table)

You must define the valid object codes for the claim accounts.

Subfund table (subfund_table)

You must define the valid subfund codes that serve as the designation for each local department (e.g., the local Athletic department is Subfund 1550, and the local Computer department is Subfund 1700).

Defined Accounts record (gld_rec)

You must define the valid combinations of funds, functions, objects and subfunds for the newly defined codes in the Object table and the Subfund table.

Claim table (claim_table)

You must define the account combinations within the Clearing House fund that trigger the use of the claim on cash feature. The Claim table also includes information about the accounts to assess for the claims with the local department.

Other account component tables

Depending on your institution's requirements, you may also need to complete other tables for the components that you include as part of your account number. For example, if your institution uses the Type and the Classification as part of the account number, you must complete the Type table (typ_table) and the Classification table (clas_table). For more information about customizing your account number structure, see *Implementing Customized Account Numbers* in this section.

Completing the Claim Table

Introduction

CX uses the information in the Claim table only for the claim on cash feature. The table defines the CHF account that triggers the claim on cash feature, and also specifies the accounts for the local departmental transactions that result from using claim on cash. If your institution does not use this feature, do not create any table entries for the Claim table, and do not set the CLAIM_ON_CASH entry in the Configuration table (config_table) to Y

If you use the claim on cash feature, you *must* define *all* cash, accounts receivable (student accounts), and accounts payable accounts in the Claim table. You can designate the same contra and object accounts for two or more accounts. For example, both the Cash in Drawer and Cash in Bank accounts can use the same contra and object accounts.

Using the *dbmake* Audit Function

The *dbmake* utility can maintain an audit trail of changes you make to the institution's tables. Because of the complex nature of the claim on cash feature, Jenzabar recommends that you implement the *dbmake* audit function for the Claim table. You can then use the audit trail to track changes to the Claim table.

Accessing the Claim Table

Select the following CX menus to access the PERFORM screen for entering Claim table values:

1. Fiscal Management
2. Accounting
3. Table Maintenance
4. Financial (A-C)
5. Claim-on-Cash Table

Fields in the Claim Table

The Claim table contains the following fields:

Account Number:
Fund
Function
Object
Subfund

The account that appears on an input transaction and triggers the use of the claim on cash feature.

> **Note:** An input transaction must match these four fields (fund, function, object and subfund) *exactly* to cause the claim on cash feature to create entries.

Contra Account:
Fund
Function
Object
Subfund

The account that claim on cash charges (with the opposite sign, debit or credit) when an input entry matches the account in the Account Number: fields. The balance in this account offsets the balance in the Cash, Accounts Payable, or Accounts Receivable account.

Object Account:
Fund

Function
Object
Subfund

The account that serves as the claim account in the local department. When claim on cash creates entries, charges to this account have the same sign (debit or credit) as the original input entry.

Note: Complete the Object Account with an object code only. The entry based on the object code will obtain its fund and function codes from the original input transaction.

Active Date

The date on which the table entry becomes valid. Leave blank if the entry is valid for an indefinite period of time.

Inactive Date

The date on which the table entry becomes invalid. Leave blank if the entry is valid for an indefinite period of time.

Examples of Claim on Cash

Introduction

This section contains examples of claim on cash usage. The examples provide the following:

- T-accounts
- Journal entries
- Required values in the Claim table

Refer to the examples as you set up the claim on cash feature for the institution.

Chart of Accounts for Claim on Cash Examples

The following list contains the object, function, and subfund codes used in the claim on cash examples. The object codes that appear in italics are the codes used with the claim on cash feature.

Balance Sheet Objects

- 1100 - Cash in Bank
- *1101 - Contra Claim on Cash*
- 1105 - Cash in Drawer
- *1110 - Claim on Cash*
- 1112 - Allocation of Student Payments
- 1150 - Cash Held by State
- 1183 - Allocation of Cash for Student Payments
- 1184 - Allocation of Student Payments as Collections on A/R
- 1310 - Student Accounts Receivable
- *1311 - Contra Claim on Accounts Receivable*
- *1320 - Claim on Student Accounts Receivable*
- 1350 - Receivable from the State
- 1360 - Internal Transfer
- 1382 - Student Collections
- 2100 - Accounts Payable
- *2101 - Contra Claim on Accounts Payable*
- *2110 - Claim on Accounts Payable*
- 3100 - Fund Balance

Revenue/Expense Objects

- 0410 - Tuition Revenue
- 0420 - Athletic Fee Revenue
- 0430 - Revenue
- 0440 - General Revenue - Appropriation Income
- 0510 - Expense
- 0520 - Software Expense
- 0530 - Equipment Expense

Functions

- 21122 - Controllers Office

Subfunds

- 17071 - Athletics
- 17087 - Computer Services

Funds

- 00 - Clearing House fund

- 01 - Income fund
- 05 - Local Operations fund

Tot Codes

- TUIT - Tuition
- ATH1 - Athletic Fees

Classifications

- 97 - Appropriation year

Types

- G - General (from the state)
- I - Income (generated from the institution)

Example 1 - Claim on Cash

In the following example, you input four entries, and claim on cash generates all the required corresponding entries. Entry numbers with a letter (e.g., 2a) indicate generated entries.

CHF - Contra Claim on Cash 00--1101

(3a)100	(1a)10,000
	(2a)100

CHF - Cash in Drawer 00--1105

(1)10,000	

CHF - Internal Transfer 00--1360

(2)100	(3)100

Comp. Svcs. - Revenue 05--0430-17087

	(1)10,000
	(2)100

Comp. Svcs. - Claim on Cash 05--1110-17087

(1a)10,000	
(2a)100	

Athletics - Expense 05--0510-17071

(3)100	

Athletics - Claim on Cash 05--1110-17071

	(3a)100

Example 1 Journal Entries

Example 1 reflects the following journal entries. In this example, the Internal Transfer account triggers the claim on cash feature.

Entry 1

Dr. CHF - Cash in Drawer
Cr. Comp.Svcs. - Revenue

Records a receipt of $10,000 cash from an outside source in exchange for computer services rendered.

Entry 1a

Dr. Comp. Svcs. - Claim on Cash
Cr. CHF - Contra Claim on Cash

Records the claim on assets that Computer Services has as a result of the revenue earned, and reflects the offsetting entry within Clearing House fund cash.

Entry 2

Dr. Internal Transfer
Cr. Comp. Svcs. - Revenue

Records revenue of $100 for a license sold by Computer Services to Athletics.

Entry 2a

Dr. Comp. Svcs. - Claim on Cash
Cr. CHF - Contra Claim on Cash

Records the claim on assets that Computer Services has as a result of providing the license to Athletics.

Entry 3

Dr. Athletics - Expense
Cr. Internal Transfer

Records expenditure side of Entry 2, to reflect cost of license to Athletics.

Entry 3a

Dr. CHF - Contra Claim on Cash
Cr. Athletics - Claim on Cash

Records the reduced claim on assets that Athletics has as a result of acquiring the license from Computer Services.

Example 1 Claim Table Entries

To cause CX to generate the Example 1 entries correctly, the following Claim table entries are required:

Account Number:

Fund 00
Function
Object 1100
Subfund

Contra Account:

Fund 00
Function
Object 1101
Subfund

Object Account:
Fund
Function
Object 1110
Subfund

Account Number:
Fund 00
Function
Object 1360
Subfund

Contra Account:
Fund 00
Function
Object 1101
Subfund

Object Account:
Fund
Function
Object 1110
Subfund

Account Number:
Fund 00
Function
Object 1105
Subfund

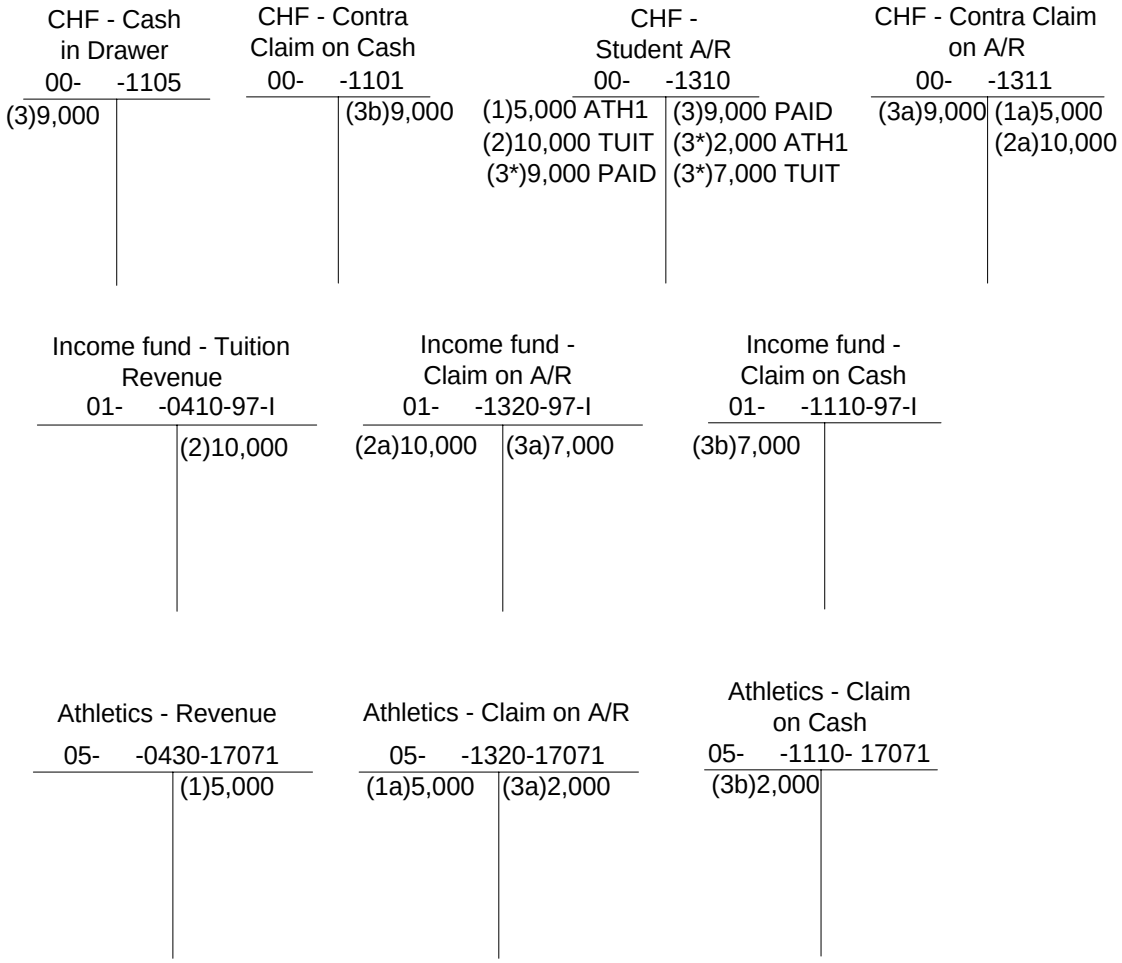
Contra Account:
Fund 00
Function
Object 1101
Subfund

Object Account:
Fund
Function
Object 1110
Subfund

Example 2 - Claim on Receivables

In the following example, you input three entries, and claim on cash generates all the required corresponding entries. Entry numbers with a letter (e.g., 2a) indicate generated entries from claim on cash. Entry numbers with an asterisk (e.g., 3*) indicate generated entries from Fee Collection.

Note: In this example, Fee Collection impacts entry 3. Fee Collection allocates the input transaction with a PAID tot code to the ATH1 and TUIT tot codes so it can be applied against the students' balances.



Example 2 Journal Entries

Example 2 reflects the following journal entries.

Entry 1

Dr. CHF - A/R
    Cr. Athletics - Revenue

Records a student billing of $5,000 for Athletic fees.

Entry 1a

Dr. Athletics - Claim on A/R
    Cr. CHF - Contra Claim on A/R

Records the claim on assets that Athletics has as a result of the revenue earned, and reflects the offsetting entry within Clearing House fund cash.

Entry 2

Dr. CHF - A/R
    Cr. Income fund - Revenue

Records a student billing of $10,000 for tuition
.

Entry 2a

Dr. Income fund - Claim on A/R
    Cr. CHF - Contra Claim on A/R

Records the claim on accounts receivable that the Income Fund has as a result of the revenue earned, and reflects the reduction in Clearing House fund receivables.

Entry 3

Dr. CHF - Cash
    Cr. CHF - A/R - PAID

Dr. CHF - A/R - PAID
    Cr. CHF - A/R - TUIT
    Cr. CHF - A/R - ATH1

Records the collection of $9,000 cash from students, and reflects the allocation of the money to the TUIT and ATH1 tot codes.

Entry 3a

Dr. CHF - Contra Claim on A/R
    Cr. Athletics - Claim on A/R
    Cr. Income fund - Claim on A/R

Records the reduced claim on receivables that Athletics and the Income Fund have as a result of the cash collection.

Entry 3b

Dr. Athletics - Claim on Cash
Dr. Income fund - Claim on Cash
    Cr. CHF - Contra Claim on Cash

Records the increased claim on cash that Athletics and the Income Fund have as a result of the cash collection.

Example 2 Claim Table Entries

To cause CX to generate the Example 2 entries correctly, the following two Claim table entries are required:

Account Number:
Fund 00
Function
Object 1310
Subfund

Contra Account:
Fund 00
Function
Object 1311
Subfund

Object Account:
Fund
Function
Object 1320
Subfund

Account Number:
Fund 00
Function
Object 1100
Subfund

Contra Account:
Fund 00
Function
Object 1101
Subfund

Object Account:
Fund
Function
Object 1110
Subfund

Example 3 - Claim on Payables with Partial Payments

In the following example, you input four entries, and claim on cash generates all the required corresponding entries. Entry numbers with a letter (e.g., 2a) indicate generated entries.

CHF - Cash in Bank

00- -1100	
	(3)600
	(4)200

CHF - Contra Claim on Cash

00- -1101	
(3b)600	
(4b)200	

CHF - A/P

00- -2100	
(3)600	(1)1,000
(4)200	(2)500

CHF - Contra Claim on A/P

00- -2101	
(1a)1,000	(3a)600
(2a)500	(4a)200

Comp. Svcs. - Expense

05- -0510-17087	
(1)1,000	

Comp. Svcs. - Claim on A/P

05- -2110-17087	
(3a)600	(1a)1,000

Comp. Svcs. - Claim on Cash

05- -1110- 17087	
	(3b)600

Athletics - Expense

05- -0510-17071	
(2)500	

Athletics - Claim on A/P

05- -2110-17071	
(4a)200	(2a)500

Athletics - Claim on Cash

05- -1110-17071	
	(4b)200

Example 3 Journal Entries

Example 3 reflects the following journal entries.

Entry 1

Dr. Comp. Svcs. - Expense
Cr. CHF - A/P

Records a $1,000 expense incurred by Computer Services.

Entry 1a

Dr. CHF - Contra Claim on A/P
Cr. Comp. Svcs. - Claim on A/P

Records the claim on liabilities that Computer Services has as a result of the expense incurred, and reflects the offsetting entry within Clearing House fund accounts payable.

Entry 2

Dr. Athletics - Expense
Cr. CHF - A/P

Records a $500 expense incurred by Athletics.

Entry 2a

Dr. CHF - Contra Claim on A/P
Cr. Athletics - Claim on A/P

Records the claim on liabilities that Athletics has as a result of the expense incurred, and reflects the offsetting entry within Clearing House fund accounts payable.

Entry 3

Dr. CHF - A/P
Cr. CHF - Cash

Records the invoice payment on account of $600.

Entry 3a

Dr. Comp. Svcs. - Claim on A/P
Cr. CHF - Contra Claim on A/P

Records the reduced claim on payables that Computer Services has as a result of the cash payment.

Entry 3b

Dr. CHF - Contra Claim on Cash
Cr. Comp. Svcs. - Claim on Cash

Records the decreased claim on cash that Computer Services has as a result of the cash payment.

Entry 4

Dr. CHF - A/P
Cr. CHF - Cash

Records the invoice payment on account of $200.

Entry 4a

Dr. Athletics - Claim on A/P
Cr. CHF - Contra Claim on A/P

Records the reduced claim on payables that Athletics has as a result of the cash payment.

Entry 4b

Dr. CHF - Contra Claim on Cash
Cr. Athletics - Claim on Cash

Records the decreased claim on cash that Athletics has as a result of the cash payment.

Example 3 Claim Table Entries

To cause CX to generate the Example 3 entries correctly, the following Claim table entries are required:

Account Number:
Fund 00
Function
Object 2100
Subfund

Contra Account:
Fund 00
Function
Object 2101
Subfund

Object Account:
Fund
Function
Object 2110
Subfund

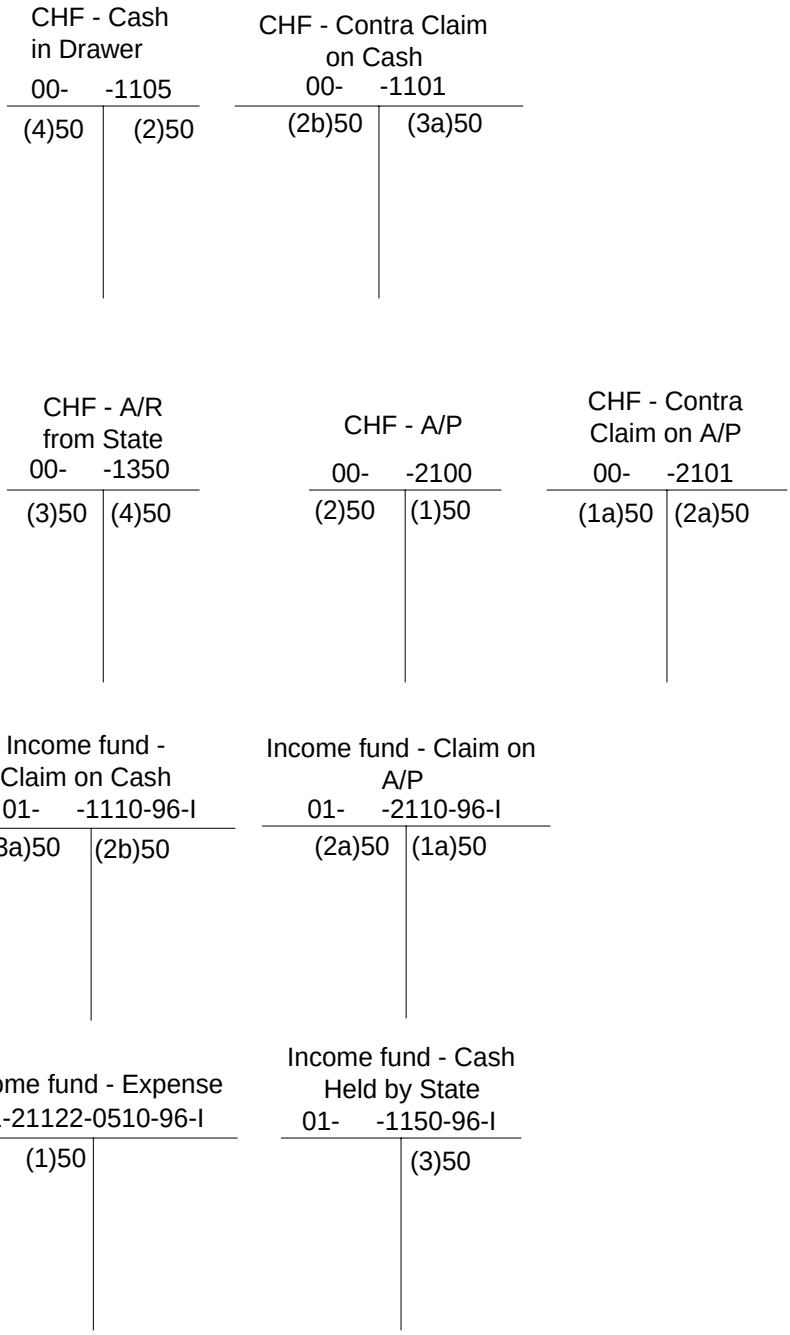
Account Number:
Fund 00
Function
Object 1100
Subfund

Contra Account:
Fund 00
Function
Object 1101
Subfund

Object Account:
Fund
Function
Object 1110
Subfund

Example 4 - Claim on Payables

In the following example, you input five entries, and claim on cash generates all the required corresponding entries. Entry numbers with a letter (e.g., 2a) indicate generated entries.



Example 4 Journal Entries

Example 4 reflects the following journal entries.

Entry 1

Dr. Income fund - Expense
Cr. CHF - A/P

Records a $50 expense incurred by a department that pays its expenses from the Income fund.

Entry 1a

Dr. CHF - Contra Claim on A/P
Cr. Income fund - Claim on A/P

Records the claim on liabilities that the Income fund has as a result of the expense incurred, and reflects the offsetting entry within Clearing House fund accounts payable.

Entry 2

Dr. CHF - A/P
Cr. CHF - Cash

Records a $50 expense paid from the Clearing House fund cash account.

Entry 2a

Dr. Income fund - Claim on A/P
Cr. CHF - Contra Claim on A/P

Records the reduction of accounts payable that the Income fund has as a result of paying for the expense incurred, and adjusts the balance in Clearing House fund claim on payables.

Entry 2b

Dr. CHF - Contra Claim on Cash
Cr. Income fund - Claim on Cash

Records the reduction in cash that the Income fund has as a result of paying for the expense incurred, and increases the balance in Clearing House fund claim on cash.

Entry 3

Dr. CHF - A/R from State
Cr. Income fund - Cash Held by State

Recognizes cash at the time of billing to the state.

Entry 3a

Dr. Income fund - Claim on Cash
Cr. CHF - Contra Claim on Cash

Records claim on cash in the Income fund, reflecting it in the Clearing House fund.

Entry 4

Dr. CHF - Cash in Drawer
Cr. CHF - A/R from State

Records the receipt of $50 cash from the state.

Example 4 Claim Table Entries

To cause CX to generate the Example 4 entries correctly, the following Claim table entries are required:

Account Number:

Fund 00
Function
Object 2100
Subfund

Contra Account:
Fund 00
Function
Object 2101
Subfund

Object Account:
Fund
Function
Object 2110
Subfund

Account Number:
Fund 00
Function
Object 1100
Subfund

Contra Account:
Fund 00
Function
Object 1101
Subfund

Object Account:
Fund
Function
Object 1110
Subfund

Example 5 - Claim on Payables for Local Expense Paid by Check

In the following example, you input two entries, and claim on cash generates all the required corresponding entries. Entry numbers with a letter (e.g., 2a) indicate generated entries.

CHF - Cash

00- -1100	
	(2)300

CHF - Contra Claim on Cash

00- -1101	
(2b)300	

CHF - A/P

00- -2100	
(2)300	(1)300

CHF - Contra Claim on A/P

00- -2101	
(1a)300	(2a)300

Comp. Svcs. - Claim on Cash

05- -1110-17087	
	(2b)300

Comp. Svcs. - Expense

05- -0510-17087	
(1)300	

Comp. Svcs. - Claim on A/P

05- -2110-17087	
(2a)300	(1a)300

Example 5 Journal Entries

Example 5 reflects the following journal entries.

Entry 1

Dr. Comp. Svcs. - Expense
Cr. CHF - A/P

Records a $300 expense incurred by Computer Services.

Entry 1a

Dr. CHF - Contra Claim on A/P
Cr. Comp. Svcs. - Claim on A/P

Records the claim on liabilities that Computer Services has as a result of the expense incurred, and reflects the offsetting entry within Clearing House fund accounts payable.

Entry 2

Dr. CHF - A/P
Cr. CHF - Cash

Records a $300 expense paid from the Clearing House fund cash account.

Entry 2a

Dr. Comp. Svcs. - Claim on A/P
Cr. CHF - Contra Claim on A/P

Records the reduction of liabilities that Computer Services has as a result of paying for the expense incurred, and adjusts the balance of Clearing House fund claim on payables.

Entry 2b

Dr. CHF - Contra Claim on Cash
Cr. Comp. Svcs. - Claim on Cash

Records the reduction in cash that Computer Services has as a result of paying for the expense incurred, and increases the balance in Clearing House fund claim on cash.

Example 5 Claim Table Entries

To cause CX to generate the Example 5 entries correctly, the following Claim table entries are required:

Account Number:

Fund 00
Function
Object 2100
Subfund

Contra Account:

Fund 00
Function
Object 2101
Subfund

Object Account:
Fund
Function
Object 2110
Subfund

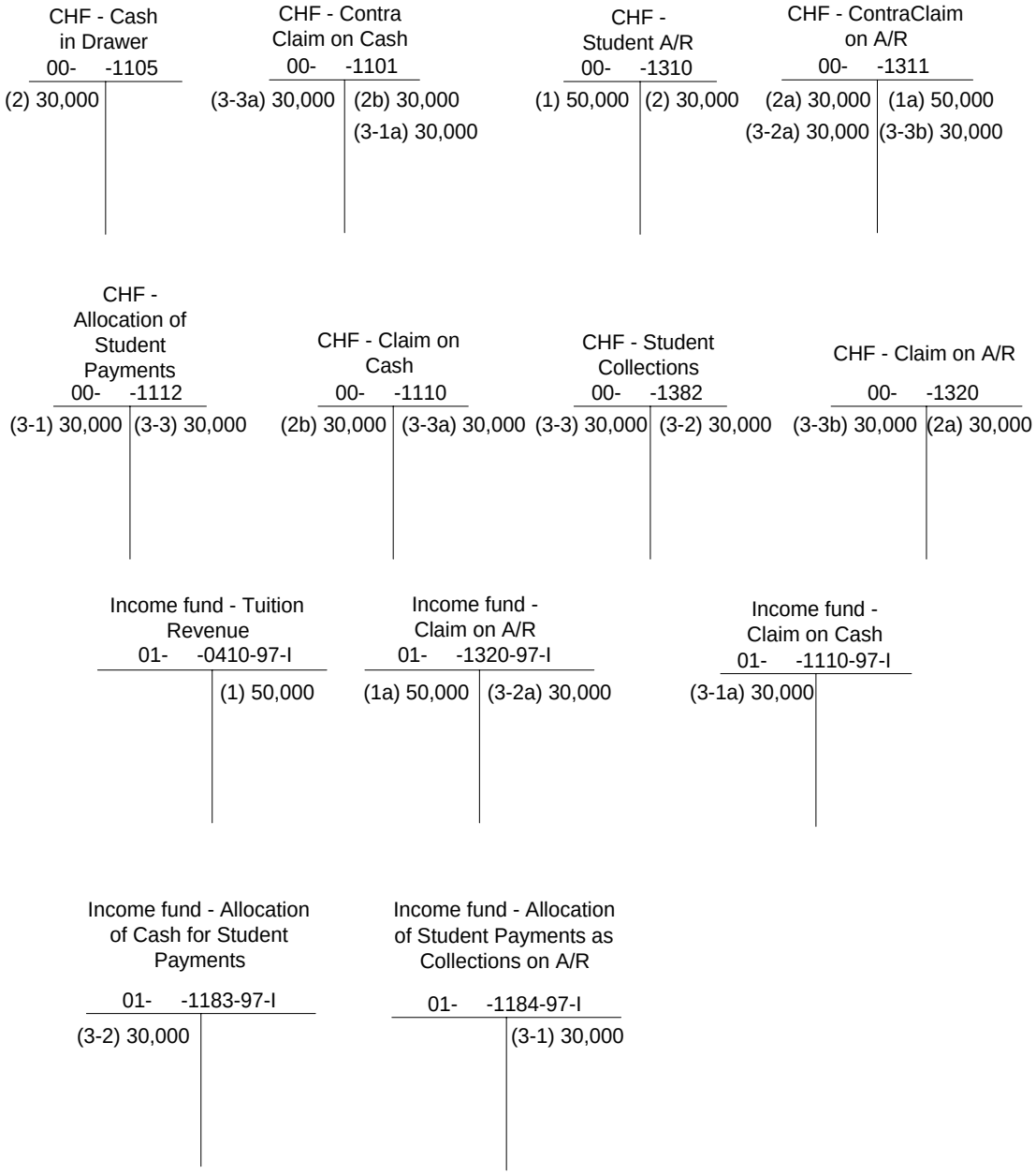
Account Number:
Fund 00
Function
Object 1100
Subfund

Contra Account:
Fund 00
Function
Object 1101
Subfund

Object Account:
Fund
Function
Object 1110
Subfund

Example 6 - Billing, Student Payments, and Allocation of Payments on Student Accounts

In the following example, you input two entries, and an ACE report creates transactions which allocate student payments against the receivables originally posted. Claim on cash generates all the required corresponding entries. Entry numbers with a letter (e.g., 2a) indicate generated entries, and entry numbers with a hyphenated number (e.g., 3-1) indicate entries created by the ACE report, which produces a journal to be posted.



Example 6 Journal Entries

Example 6 reflects the following journal entries.

Entry 1

Dr. Student A/R
Cr. Income fund - Tuition Revenue

Records a billing to students for tuition.

Entry 1a

Dr. Income fund - Claim on A/R
Cr. Contra Claim on A/R

Records the claim on receivables that the Income fund has as a result of the revenue earned, and reflects the offsetting entry within Clearing House fund accounts receivable.

Entry 2

Dr. CHF - Cash in Drawer
Cr. CHF - Student A/R

Records a total of $30,000 cash received from students in response to billing.

Entry 2a

Dr. CHF - Contra Claim on A/R
Cr. CHF - Claim on A/R

Relieves the Student A/R before allocating any payments to the individual account in the Income fund.

Entry 2b

Dr. CHF - Claim on Cash
Cr. CHF - Contra Claim on Cash

Records the claim on cash that the CHF maintains before allocating any payment to the individual account in the Income fund.

The following entries occur as a result of the ACE report's journal entry, which is posted after Fee Collection reviews the billing and receipts and allocates the payments against the charges. These entries occur automatically, using accounts created specifically to trigger claim on cash transactions. The balances of the allocation (Allocation of Cash for Student Payments and Allocation of Student Payments as Collections on A/R) accounts completely offset each other, so their zero net balance has no effect on reporting.

Entry 3-1

Dr. CHF - Allocation of Student Payments
Cr. Income fund - Allocation of Student Payments as Collections on A/R

Entry 3-1a

Dr. Income fund - Claim on Cash
Cr. CHF - Contra Claim on Cash

Entry 3-2

Dr. Income fund - Allocation of Cash for Student Payments
Cr. CHF - Student Collections

Entry 3-2a

Dr. CHF - Contra Claim on A/R
Cr. Income fund - Claim on A/R

Entry 3-3

Dr. CHF - Student Collections
Cr. CHF - Allocation of Student Payments

Entry 3-3a

Dr. CHF - Contra Claim on Cash
Cr. CHF - Claim on Cash

Entry 3-3b

Dr. CHF - Claim on S/A
Cr. CHF - Contra Claim on A/R

Example 6 Resulting Account Balances

When you make the types of entries in this example, and use Claim on Cash, Fee Collection and the related ACE report, the ending balances in the accounts are as follows. Note that these balances usually net to zero, therefore minimizing their effect on the institution's financial reports.

00- -1112 Allocation of Student Payments
Zero

00- -1110 Claim on Cash
Zero

00- -1382 Student Collections
Zero

00- -1320 Claim on Student A/R
Zero

00- -1105 Cash in Drawer
30,000 Dr.

00- -1101 Contra Claim on Cash
30,000 Cr.

00- -1310 Student A/R
20,000 Dr.

00- -1311 Contra Claim on A/R
20,000 Cr.

01- -0410-97-I Tuition Revenue
50,000 Cr.

01- -1320-97-I Claim on A/R
20,000 Dr.

01- -1110-97-I Claim on Cash
30,000 Dr.

01- -1183-97-I and **01- 1184-97-I Allocation of Cash for Student Payments** and **Allocation of Student Payments as Collections on A/R**
Net balance of zero.

Example 6 Claim Table Entries

To cause CX to generate the Example 6 entries correctly, the following Claim table entries are required:

Account Number:
Fund 00
Function
Object 1112
Subfund

Contra Account:
Fund 00
Function
Object 1101
Subfund

Object Account:
Fund
Function
Object 1110
Subfund

Account Number:
Fund 00
Function
Object 1382
Subfund

Contra Account:
Fund 00
Function
Object 1311
Subfund

Object Account:
Fund
Function
Object 1320
Subfund

Setting Up Appropriation Years and Reporting Structures

Introduction

CX is a flexible product that can accommodate a variety of account structures, reporting requirements and budgetary demands. The setup information in this section outlines the ways institutions can modify CX to suit their particular needs.

In addition, CX can support a variety of special purpose approaches to tracking and reporting budgetary and departmental information. These approaches go beyond the most common usages of CX, and most institutions do not need to use them. The special purpose requirements include the following:

- Using appropriation years of other than twelve months in length, and reporting actual and budget amounts in a variety of ways. For example, an institution may use a fifteen month appropriation year, and want to track both calendar and appropriation year accounting information.
- Consolidating budgetary activities by budgeting at the master account level (e.g., travel expenses) while tracking actual charges at the detail account level (e.g., hotel expense and air travel).
- Producing reports of budget activity based on revenue source, regardless of original department. For example, many departments may incur travel expenses and pay the expenses from different resources (e.g., General Revenue or Educational Revenue). For some reports, only the total of all travel expenses for each resource may be required; the original department that incurred the expense may be unimportant for this type of report.

Setting Up and Using the Fifteen Month Appropriation Year

To set up and use a fifteen month appropriation year, do the following:

1. Define the Class as part of the account number, using the procedure *Implementing Customized Account Numbers* in this manual.
2. Set the GL_CLAS_ENABLE value in the Configuration table to Y.
3. Select the Appropriation Year command in Budget Review screens.

Setting Up and Using Consolidated Budget Reporting

To set up and use consolidated budget reporting, do the following:

1. Define interrelationships between blocks, groups, schedules and items in the Financial Statement table (fs_table). For more information about setting up the Financial Statement table, see *General Ledger Tables That Require Modification* in this manual.
2. Set the GL_POOL_ENABLED value in the Configuration table to Y.
3. Select the appropriate level in the Budget Review screens.

Setting Up and Using Budget Reports Based on Revenue Source

To set up and use budget reports based on revenue source, regardless of the original department charged, do the following:

1. Define the Type as part of the account number, using the procedure *Implementing Customized Account Numbers* in this manual.
2. Define the following two entries in the Configuration table:
   - GL_POOL_ENABLED set to Y.

- INST_WIDE_FUNC as the department number that you want to use to consolidate all budgetary activity for reporting purposes.

3. Using Budget Review, select the consolidated (aggregate) department number.

Using ascii Journal Transaction Files

Introduction

ACE reports, shell scripts, and other programs that interface with CX can create ascii transaction files that post to the General Ledger. An ascii transaction file minimizes the need for manual data entry, enhancing the accuracy and efficiency of your financial information. To post to the General Ledger, ascii transaction files must follow a specific format.

Role of Voucher and Background Voucher

All entries to the General Ledger must use the *voucher* or *bgvoucher* programs to post to the CX database to ensure they follow your rules for the account structure as set up in the database. The *voucher* and *bgvoucher* programs check the account numbers and all account codes for validity before any change is made to the General Ledger. During posting, they update the totals in the General Ledger Amount records, update subsidiary account totals in the subsidiary ledger, and maintain entry numbers, journal numbers, and other similar variables determined at posting time.

Role of Voucher Transaction Files

Many CX programs (e.g., *pay* and *ckpost*) use a "vt" (voucher transaction) file, which is then posted with *voucher* or *filepost*.

The "vt" file is, however, not very convenient when trying to post ascii information. The ascii transaction file is an updated version of the "vt" file that can be created by virtually any command that can create an ascii text file.

Ways to Create an ascii Posting File

The most common means of creating an ascii posting file is to write an ACE report that reads the requirements from the database, manipulates the data, and prints the required accounting entries in the ascii transaction format to a file in the $CARSPATH/vchpost/Filepost/ascii directory. Typically, a script controls the ACE output and checks for error conditions.

You can also create the ascii posting file with shell scripts, awk scripts, C programs, sed or ex editor scripts, and many other UNIX text manipulation tools.

Posting the ascii File

Once a process creates the ascii transaction file, it is posted using *filepost*. The *filepost* program reads the transaction file and passes it to *bgvoucher*.

Syntax for Posting ascii Files

The correct syntax for posting an ascii file is:

```
filepost -n filename -m a
```

where *filename* is the name of the file in the ascii directory, as in "filepost -n AC234222", and the "-m a" parameter instructs *filepost* to look for an ascii file in $CARSPATH/vchpost/Filepost/ascii. Do not enter the full path of the file to be posted.

Using Filepost in Interactive Mode

To run *filepost* in an interactive mode, which allows you to choose which files to post, enter the following:

filepost -i -m a

ascii Transaction File Format

An ASCII transaction file is a series of lines that tell *bgvoucher* what to post to the database. Each transaction line contains information, including strings, dates, codes that will be looked up in the database, dollar amounts, and numbers. Each field in the transaction line is separated from the next field and the previous field with a comma.

Composition of the Example ascii Transaction

For example, the subsidiary entry line contains seven fields as shown below:

SE,A/P,20240,IV,3043003,"Smith, John",AINV

Field 1 (SE)
The entry type. In the above example, the transaction code is "SE" (Subsidiary Entry).

Field 2 (A/P)
The subsidiary (A/P).

Field 3 (20240)
The subsidiary number (20240).

Field 4 (IV)
The document code (IV).

Field 5 (3043003)
The document number (the invoice number 3043003).

Field 6 (Smith, John)
The subsidiary description (the name of the person associated with the subsidiary number).

Field 7
The subsidiary entry type.

Each field is separated from the others with a comma.

Text Fields in Transaction Files

Some fields (e.g., the subsidiary description field) may have text containing commas. Enclose these fields in quotes to ensure that *filepost* interprets the commas as characters and not data separators. If the description contains a double-quote, you can quote the string with a single quote. Generally, you do not need to quote the numbers in a transaction line, while you will quote all the strings.

Omitted Fields in Transaction Files

If a field is not used or blank, it must still appear in the transaction file. It can either be filled in with a blank, or you can enter a comma and then the contents of the next field. For example, in the following line:

HD,"bfwd",0,7/15/89,,,AC

the fifth and sixth fields on the line have been omitted.

Transaction Types in the Transaction File

A total of seven different types of transactions can appear in the transaction file:

- Comment lines (lines that begin with "#")
- A header line (one that begins with "HD")
- General ledger entry lines (lines that begin with "GE")

- General ledger transaction lines (beginning with "GT")
- Subsidiary entry lines (beginning with "SE")
- Subsidiary transaction lines (beginning with "ST")
- A trailer line, beginning with "TR".

Required Components of the Transaction File

Of the seven types of transactions, only comment lines (#) can appear anywhere in the file. You must include one and only one header line (HD), which must be the first line in the file, and there must be one and only one trailer line (TR), which must be the last line in the file.

The rest of the transaction lines can be entered in the same order as you would enter an entry in *voucher*. You must include at least one general ledger entry line (GE), and for each general ledger entry you must supply at least two general ledger transaction (TR) lines. If a general ledger transaction is made to a subsidiary control account, that general ledger transaction line must be immediately followed by a subsidiary entry line. (This occurs in *voucher* when you enter a subsidiary control account and it automatically switches to the subsidiary screen.) Each subsidiary entry line must then be followed by at least one subsidiary transaction line.

Example of a Transaction File

An example transaction file follows:

```
#
#       This is a comment.  Comment lines exist throughout this file
#        to explain the file's contents.
#
#       The following is the header line.  Note that there is only
#       one header in the file.
#
HD,"bfwd",0,7/15/89,,,AC
#
#       Here we have the first general ledger entry line.  Note that you
#       must have a GE line before you can have any other type of
#       transaction.
#
GE,4,BF,1,"Subsidiary Bal Forward",SFWD,,,
#
#       Here is the first general ledger transaction line.  Each GE
#       line will always be immediately followed by a GT line.
#
GT,-500.00,10,,1301,,"   "
#
#       This GT happens to be a subsidiary control account and is
#       immediately followed by an SE line.  If this account was not
#       a control account, it would be followed by another GT line.
#
SE,S/A,4,4,BF,1,"Subsidiary Bal Forward",SFWD
#
#       Each SE is followed by at least one ST.
#
ST,PAID,FA88,SB,FA88,-500.00,0.00
#
#       You could have another SE for this subsidiary at this point,
#       as long as all the amounts in all the transactions totaled
#       up to the general ledger transaction amount.
#
GT,500.00,"10","    ","1301","    ","    "
SE,S/A,4,4,BF,1,"Subsidiary Bal Forward",SFWD
ST,PAID,SP89,SB,SP89,500.00,0.00
#
#       Here, we could have more GT lines (with or without SE lines),
#       as long as the total transactions for the GE added up to 0.0.
#
TR
```

Comment Lines

Any line that begins with a "#" character is considered a comment line. The *filepost* program will ignore comment lines, along with any blank lines that exist in the file.

The Header Line

Every transaction file must begin with a header line. The header line has the following eight fields:

HD
The first field in the header line contains the header transaction type, "HD".

Author
The second field in the header contains a string that references the program or process that created the transaction file. Usually, this is the name of the ACE report or script that created the transaction file.

User ID Number
The third field in the header contains the Unix user id of the user creating the posting file. This is for information purposes only (i.e., it is not used by *bgvoucher* or *filepost*). The person who runs *filepost* to post the file is the person associated with the journal in the Journal record.

Posting Date
The fourth field in the header is the posting date for this journal.

Station Number
The fifth field in the header is the document station number for the journal. This station number is generally applicable only to interactive programs, and is usually not used for ascii transaction files.

Posting Period
The sixth field in the header is the period to which the journal will be posted. This can be one of the twelve fiscal months, or it can be the "BAL" (balance forward), "ADJ" (adjustment), or "CLS" (closing) periods. If left blank, the posting date will be used to determine the default posting period (as defined in the Fiscal Calendar file).

Journal Code
The seventh field in the header is the journal to use for posting.

Document Code
The last field in the header specifies the document code for the journal. If the document code in the journal is given, all entries in the journal must use that document code. This is primarily used for the cashier journal, where the document code is used in reconciling cash receipts.

If you omit the document code, CX performs only the normal Document table checking on the document codes given in the general ledger entries.

General Ledger Entry Line

The general ledger entry line contains information about one set of general ledger and subsidiary transactions (e.g., GE,4,BF,1,"Subsidiary Bal Forward",SFWD,,,). It contains the following fields:

GE
The header transaction type, "GE".

Document ID
The ID number of the person associated with the document. If no document exists, or if the document is not associated with a person, this field can be left blank or zero.

Document Reference Code
The two-character document reference code. All general ledger entries must contain a document code.

Document Number

In interactive programs, a number that can be assigned by the system or entered by the user, such as the number of a check or the number on a purchase order. In background processes, the ACE report should assign a document number for reference.

Entry Description
The 24-character description of the entry.

Entry Type
The four-character entry code.

Form of Payment
The seventh field is the form of payment associated with this entry. If there is no form of payment associated with this type of entry, it can be left blank.

Form of Payment Number
A numeric identifier for the payment (e.g., the number of a check). If no form of payment exists, the number should be left blank.

General Ledger Transaction Line

The general ledger information line contains the account numbers and the amounts that are to be posted (e.g., GT,-500.00,10,,1301,," "). It contains the following fields:

GT
The transaction type "GT".

Amount
The second field in the general ledger transaction line is the amount of the transaction.

General Ledger Account Number
The third through sixth fields in the general ledger transaction line contain the fund, function, object, and subfund.

Subsidiary Ledger Entry

The subsidiary entry contains general information about the transactions as they affect the subsidiary account. If the general ledger transaction is not a subsidiary control account, any subsidiary information will be flagged as an error and the file will not be posted. Likewise, if the general ledger account is a subsidiary control account, and no subsidiary information is supplied, CX detects the error.

In many cases, the information in the subsidiary entry duplicates the general ledger entry. In *voucher*, for example, the description given for the general ledger entry commonly defaults into the subsidiary entry description. The entry type used for the general ledger entry is usually the same for the subsidiary entry. You may have one or more subsidiary entries for each general ledger transaction.

A subsidiary entry resembles the following:

SE,S/A,4,4,BF,1,"Subsidiary Bal Forward",SFWD

Fields in a subsidiary entry are:

SE
The subsidiary entry transaction type, "SE".

Subsidiary Code
The four-character code given in the General Ledger Account record (gla_rec) and the Subsidiary table. If this code does not match the code in the General Ledger Account record for the preceding transaction, the ascii transaction file cannot be posted.

Subsidiary Number

The ID number associated with the subsidiary.

Document ID
The document ID.

Document Reference Code
The document code.

Document Number
The next field is the document number. Note that the document number in the Subsidiary Entry record differs from the document number in the General Ledger Entry record. The General Ledger Entry document number is stored in the database as a long integer, and can only hold integer numbers. The Subsidiary Entry record, on the other hand, is a twelve-character field that can hold any alphanumeric value you want to put in it. The purchasing module, for example, uses this field to store vendor invoice numbers, which can contain dashes, numbers, and letters.

Subsidiary Description
The twenty-four character description of the subsidiary transactions.

Subsidiary Entry Type
The four-character entry type for the subsidiary entry. This entry type is subject to all of the restrictions imposed on the general ledger entry type. Only certain entry types are allowed in in some journals, and some entry types cannot be used in subsidiaries at all. Usually, the entry type for the subsidiary entry is the same as the entry for the general ledger entry.

`Subsidiary Ledger Transaction`

The subsidiary transactions provide the detail for the subsidiary entry (e.g., ST,PAID,SP89,SB,SP89,500.00,0.00. The Subsidiary Ledger Transaction contains the following fields:

ST
The subsidiary transaction type, "ST".

Subsidiary Total Code
The subsidiary total code for the transaction. This code is the deduction or earnings code for Wages Payable subsidiaries. It is the charge code for Student Accounts subsidiaries.

Subsidiary Total Period
The subsidiary total period for the transaction. In Wages Payable subsidiaries, this is a code for the calendar year ("CY98", for example). In Student Accounts subsidiaries, this is the session code.

Subsidiary Balance Code
The balance code for the transaction.

Subsidiary Balance Period
The balance period for the transaction. In Wages Payable subsidiaries, this is a combination of the payroll code and payroll number. In Student Accounts subsidiaries, it is the same as the total code. In Accounts Payable subsidiaries, it is always "INV".

Subsidiary Amount
The amount of the subsidiary transaction.

Subsidiary Associated Amount
The amount associated with the subsidiary amount. Payroll (Wages Payable subsidiaries) use this field to save the amount on which a deduction is based.

Trailer

The trailer consists of one field that contains "TR". This does nothing more than signify the end of the transaction file (e.g., TR).

ascii File Example

Following is an example of a simple ascii transaction file:

```
#
#       This is a file that illustrates an ASCII journal transaction file.
#

#   Author  Uid  Post Date  Station    Post Month  Journal
HD,"bfwd",   0,  7/15/98  ,         ,             ,   AC
GE,4,BF,1,"Subsidiary Bal Forward",SFWD,,,
GT,-500.00,10,,1301,,"    "
SE,S/A,4,4,BF,1,"Subsidiary Bal Forward",SFWD
ST,PAID,FA97,SB,FA97,-500.00,0.00
GT,500.00,"10","    ","1301","    ","    "
SE,S/A,4,4,BF,1,"Subsidiary Bal Forward",SFWD
ST,PAID,SP97,SB,SP97,500.00,0.00

TR
```

Purpose of Example Transaction File

The example transaction file demonstrates entries for a student account that forwards an amount from one session to another. This transaction file is similar to the one that is produced by *sabalfwd* (Subsidiary Balance Forward).

Explanation of Example Transaction File

The header line indicates to *filepost* that this file was created by a process called "bfwd". This file will be posted to "7/15/98". No station number is given (it will post to Station zero). No fiscal period is given, so it will default to the month that corresponds to "7/15/98" (typically JUL). It will be posted with an "AC" (accounting) journal.

The general ledger entry indicates to *filepost* that this entry was associated with id #4, is "BF" document #1, with a description of "Subsidiary Bal Forward" and an entry type of SFWD. Since no form of payment is given, it is assumed to be blank.

The general ledger transaction gives the amount and the G/L account number.

The subsidiary entry gives the subsidiary (S/A), the subsidiary number and document ID (both are #4 in this case), and the document reference and document number. Although the document reference and number are the same as those which appeared in the general ledger entry, this will not always be the case. The final two fields give the subsidiary description and the entry type.

The subsidiary transaction line provides the total code, balance code, session, and amount.

Example ACE Report

Following is a sample ACE report that might print the above transaction file.

```
database cars end

define
param[1] subs                   type character length 4
param[2] from_session           type character length 6
param[3] target_session         type character length 6
param[4] post_date              type date
variable quote type character length 1
variable comma type character length 1
end

output
left margin 0
top margin 0
bottom margin 0

read into a
subb_rec
where
subb_subs=subs
and subb_code="SB  "
and subb_prd=from_session
and subb_amt_act <> 0
end

sort by subb_subs subb_subs_no end

format
first page header

let comma=","
let quote=ascii(34)

print "HD", comma, quote, "bfwd", quote, comma, "0",
   comma, quote, post_date, quote, comma,comma,comma,"AC"

on every record

print "GE", comma, subb_subs_no, comma, "BF", comma, "1", comma,
   quote, "Subsidiary Bal Forward", quote, comma, "SFWD", comma,
   comma, comma

print "GT", -subb_amt_act using "-----&.&&", comma, "10",
   comma, comma, "1301", comma, comma, quote, "    ", quote

print "SE", comma, subb_subs, comma, subb_subs_no, comma,
   subb_subs_no, comma, "BF", comma, "1", comma, quote,
   "Subsidiary Bal Forward", quote, comma, "SFWD"

print "ST", comma, "PAID", comma, from_session, comma, subb_code,
   comma, from_session, comma, -subb_amt_act using "-----&.&&",
   comma, "0.00"

print "GT", subb_amt_act using "-----&.&&", comma, "10",
   comma, comma, "1301", comma, comma, quote, "    ", quote

print "SE", comma, subb_subs, comma, subb_subs_no, comma,
   subb_subs_no, comma, "BF", comma, "1", comma, quote,
   "Subsidiary Bal Forward", quote, comma, "SFWD"

print "ST", comma, "PAID", comma, target_session, comma,
   subb_code, comma, target_session, comma,
   subb_amt_act using "-----&.&&", comma, "0.00"

on last record
   print "TR"

end
```

Notes to Creating the Example ACE Report

Note the following in the example ACE report:

- First, all dollar amounts must be formatted with a "using" statement. The default format for money includes a dollar sign, and *filepost* cannot interpret dollar signs or commas in numeric fields.
- Second, note the use of the "comma" and "quote" variables in this report. Although these variables are not required, they help you test and debug your report.

SECTION 3 - MACROS, INCLUDES AND CONFIGURATION TABLE ENTRIES

Overview

Introduction

This section provides reference information about macros and includes used to set up the General Ledger module.

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.

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.

In the General Ledger product, the following processes and features use values in the Configuration table:

- Financial statement generation
- Institution-wide reporting department
- Class code as part of the account number for 15 month appropriation years
- Claim on cash

General Installation Procedures

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

General Ledger Macros

Introduction

CX contains macros that define specific values used throughout the General Ledger module. The macros and includes enable you to change the available options and functionality of the General Ledger module without having to modify C code. By modifying macros, you can customize your implementation of the General Ledger module, and make the module 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 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 example values for screen comment lines (_EG)
- 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

How to Locate Macros

To modify the macros that relate to General Ledger and other financial modules and applications, access the $CARSPATH/macros/custom directory. Three files within that directory contain macros that relate to General Ledger. The three macro files for customizing the installation of CX are as follows:

- financial
- periodic
- table

To access common macros, access the $CARSPATH/macros/custom/common file.

Macros for Reporting

Another macro file, $CARSPATH/macros/custom/finrpt, contains all the macros that you can use to customize reports. For more information about the report macros, see *Financial Reports* in this manual, and the comments sections of the macro file.

Applocate Program

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

Note: To use *applocate* to locate the macros used in General Ledger, you must specify the module's name.

The following steps run the *applocate* program.

1. Select Utilities from the CX 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.

Macros in the *financial* File

The following list describes the General Ledger macros located in the *financial* macro file. The macros appear in this list in the order in which they appear in the macro file.

Note: The *financial* macro file $CARSPATH/macros/custom/financial contains macros for all financial modules and applications (e.g., Purchasing and Personnel/Payroll). This list contains only the macros that relate to General Ledger.

m4_define('GL_OBJ_LEN','GL_ACCT_LEN')
m4_define('GL_OBJ_TITLE','GL_ACCT_TITLE')
m4_define('GL_OBJ_DSPL','GL_ACCT_DSPL')
m4_define('GL_OBJ_DASHES','GL_ACCT_DASHES')
m4_define('GL_OBJ_TEXT','GL_ACCT_TEXT')
m4_define('GL_OBJ_DEFAULT','GL_ACCT_DEFAULT')
m4_define('GL_OBJ_INTERFUND','GL_ACCT_INTERFUND')

Provide backward compatibility for institutions that use the fund/center/account structure for account numbers instead of the fund/function/object structure. These macros apply to institutions that use accounts, not objects.

m4_define('GL_FUNC_LEN','GL_CNTR_LEN')
m4_define('GL_FUNC_TITLE','GL_CNTR_TITLE')
m4_define('GL_FUNC_DSPL','GL_CNTR_DSPL')
m4_define('GL_FUNC_DASHES','GL_CNTR_DASHES')
m4_define('GL_FUNC_TEXT','GL_CNTR_TEXT')
m4_define('GL_FUNC_DEFAULT,'GL_CNTR_DEFAULT')
m4_define('GL_FUNC_INTERFUND','GL_CNTR_INTERFUND')

Provide backward compatibility for institutions that use the fund/center/account structure for account numbers instead of the fund/function/object structure. These macros apply to institutions that use centers, not function codes. You should not change the values in these macros.

m4_define('GL_SUBFUND_LEN','GL_PROJ_LEN')
m4_define('GL_SUBFUND_TITLE','GL_PROJ_TITLE')
m4_define('GL_SUBFUND_DSPL','GL_PROJ_DSPL')
m4_define('GL_SUBFUND_DASHES','GL_PROJ_DASHES')
m4_define('GL_SUBFUND_TEXT','GL_PROJ_TEXT')
m4_define('GL_SUBFUND_DEFAULT','GL_PROJ_DEFAULT')
m4_define('GL_SUBFUND_INTERFUND','GL_PROJ_INTERFUND')
m4_define('GL_SUBFUND_ENABLE','GL_PROJ_ENABLE')

Provide backward compatibility for institutions that use the fund/center/account structure for account numbers instead of the fund/function/object structure. These macros apply to

institutions that use project codes, not subfunds. You should not change the values in these macros.

gl_define('FUND', ...)
gl_define('FUNC', ...)
gl_define('OBJ', ...)
gl_define('SUBFUND', ...)

Provide field lengths, abbreviated names and common names for the default standard components of the account number. If your institution wants to track other information within the account number, you can add any of the following macros to this group.

gl_define('LOC', ...)
gl_define('SUBLOC', ...)
gl_define('SUBFUNC', ...)
gl_define('SUBOBJ', ...)
gl_define('CLASS', ...)
gl_define('SUBCLASS', ...)
gl_define('TYP', ...)
gl_define('SUBTYPE', ...)

Note: Provide field lengths, abbreviated names and common names for the optional components of the account number. Because these components are optional, these macros are contained in a comments section of the macro file. If your institution wants to track other information within the account number, you can copy the appropriate macro from the comments section and place it with the group above.

Note: If users at your institution want to change the name of one of these components, you can change the abbreviated name and the common name.

Example: If you want to use the class component in your account number to track the fiscal year, you can change the CLASS macro as follows:

Original:

gl_define('CLASS', 4, 'SubP', 'Class')

Revised:

gl_define('CLASS', 4, 'FY', 'Fiscal-year')

m4_define(GL_FUNC_TABLE, 'func_table')
m4_define(GL_OBJ_TABLE, 'obj_table')
m4_define(GL_SUBFUND_TABLE, 'subfund_table')

m4_define(GL_FUNC_FIELD, 'func')
m4_define(GL_OBJ_FIELD, 'obj')
m4_define(GL_SUBFUND_FIELD, 'subfund')

Provide backward compatibility for institutions that use the fund/center/account structure for account numbers instead of the fund/function/object structure. For example, if your institution uses centers instead of function codes, make the following changes:

Original:
m4_define(GL_FUNC_TABLE, 'func_table')
m4_define(GL_FUNC_FIELD, 'func')

Revised:
m4_define(GL_FUNC_TABLE, 'cntr_table')
m4_define(GL_FUNC_FIELD, 'cntr)

m4_define('GL_FUNC_INTERFUND', '0000')

Enables you to define a default function code for all your institution's interfund transfers. In this example, every time an interfund transfer occurs, it automatically receives a function

code of 0000. Because this feature is optional, this macro is contained in a comments section of the macro file. If your institution wants to define a default function code, remove this macro from the comment section and change the zeros to the desired function code.

gl_interfund('OBJ', '11')
Defines the means by which the system can recognize an interfund transfer. In this example, any object code with the first two digits "11" triggers the due to/due from accounting that is required for interfund transfers.

m4_define('PROJECTS_ARE_SUBFUNDS','Y')
Currently not in use in the standard system.

m4_define('GL_CASH_ACCOUNTING','N')
Enables you to implement or disable cash basis accounting within Cashier and Student Billing. If you accept the default value of N, the system does not implement cash basis accounting, but performs accrual basis accounting. If you change the value to Y, the system performs cash-basis accounting, using a holding account to track charges until the cash is actually received; at that time, the system recognizes the revenue.

m4_define('GL_THIRD_PARTY','Y')
Enables you to implement third party billing functionality. If you accept the default value of Y, *bgvoucher* will update deferment records in the system for use within third party billing. After you begin processing third party billing, you cannot change this macro value.

m4_define('GL_ASK_NEW_ACCT','Y')
Provides a prompt when a user first tries to use an account that is valid, but that has not been used before. The prompt indicates that the account is valid, but does not exist, and enables the user to decide whether or not to proceed. If you change the value to N, the system will permit the user to use a previously unused (but valid) account with no intervention.

m4_define('JRNLACCT_ORDER', 'fund, subfund, obj, func')
Defines the order in which information appears on subsidiary journal reports. If you change the information, you change the appearance of the journal reports.

m4_define('ENABLE_ASCII_MENU', 'Y')
Enables the menu for ASCII posting. This menu is applicable if your institution creates files of entries (in ASCII format) from customized processes (e.g., by using ACE reports). Accepting the default of Y for this macro gives access to the ASCII posting menu.

m4_define('ENABLE_BANK_RECON_TAPE', 'Y')
Enables the bank reconciliation tape options on the menu.

Note: Bank reconciliation tapes have no standard format and the tape options available on the system must be customized before you enable this option. Since these processes are optional, this macro appears in a comment section of the macro file.

m4_define('ENABLE_MANUAL_BGT', 'Y')
Enables users to manually enter budget information using a BG journal. Accepting the default of Y for this macro gives access to the menu options that users need to enter budget information.

m4_define('ENABLE_ASSOC_ACCT_RPT', 'Y')
Enables the Associated Account reports and tables to appear on the General Ledger menus. These tables link accounts for reporting purposes.

m4_define('ENABLE_COMB_CNTR_RPT' 'Y')
Enables the Combined Center reports and tables to appear on the General Ledger menus. These tables link centers for reporting purposes.

m4_define('ENABLE_FEE_COLLECTION' 'Y')

Enables the fee collection menu options to appear on the General Ledger menus. If you set this macro value to N, no fee collection options will appear on the menus.

m4_define('ATYPE_EG', ', eg: ATYPE_DEF.')

Defines the default value for amount types. You define the value for ATYPE_DEF (e.g., ACT for Actual) in the macro file macros/custom/table, then this macro causes the value you set to appear on comment lines as in the following example:
"Enter the amount type, eg: ACT"

m4_define('ATYPE_BGT_VALID', 'ACT, BGT, APP, PRJ, REQ')
m4_define('ATYPE_BGT_INCL', 'include=(ATYPE_BGT_VALID), upshift')
m4_define('ATYPE_BGT_DEF', 'BGT')
m4_define('ATYPE_BGT_EG', ', eg: ATYPE_BGT_DEF.')

Defines the amount types that relate to budgeting. The first macro defines valid amount types (e.g., Actual, Budgeted, Approved, Projected, or Requisitioned). The second macro defines how to display the valid amount types. The third macro defines the default value. The fourth macro defines how to display the default on comment lines, as in the following example:
"Enter the amount type, eg: BGT"

m4_define('PSTMT_ATYPE_DEF','A')
m4_define('PSTMT_ATYPE_VALID','A,B')
m4_define('PSTMT_ATYPE_INCL','include=(PSTMT_ATYPE_VALID), upshift')

Defines the contents of subsidiary statements. Valid values are A (Actual) and B (both Actual and Encumbered). If your institution never uses encumbered amounts on statements, you can change the definition of the second macro PSTMT_ATYPE_VALID from A,B to A.

m4_define('OBJ_ASSET_BEG','1000')
m4_define('OBJ_ASSET_END','1999')
m4_define('OBJ_LIAB_BEG','2000')
m4_define('OBJ_LIAB_END','2999')
m4_define('OBJ_FB_BEG','3000')
m4_define('OBJ_FB_END','3999')
m4_define('OBJ_REV_BEG','4000')
m4_define('OBJ_REV_END','5999')
m4_define('OBJ_EXP_BEG','6000')
m4_define('OBJ_EXP_END','9999')

Defines the object code ranges for assets, liabilities, fund balances, revenues, and expenses. The system uses the ranges in these macros to format reports.

m4_define('ASSOC_CODE_DEF', 'BOOK')
m4_define('ASSOC_CODE_VALID', 'BOOK, HSKP, INSU, DORM, CASH')
m4_define('ASSOC_CODE_INCL', 'include=(ASSOC_CODE_VALID, upshift')

Defines the values from the General Ledger Association record that you want to use for reporting. The second macro defines the valid values, and the first and third macros define the display of the codes and the default.

m4_define('FS_AXIS_DEF','FUNC')
m4_define('FS_AXIS_VALID','OBJ, FUNC, SUBFUND')
m4_define('FS_AXIS_INCL','include=(FS_AXIS_VALID), upshift')
m4_define('FS_AXIS_EX', ' (OBJ) (FUNC) (SUBFUND).')

Defines the information that your institution displays on financial reports. If the institution uses centers, accounts, and projects, you can modify the macro values to reflect this terminology.

m4_define('ENT_EG',', eg: ENT_DEF.')

Defines valid entry types. You define the value for ENT_DEF (e.g., ADJ for Adjustment) in the macro file macros/custom/table, then this macro causes the value you set to appear on comment lines as in the following example:
"Enter the entry type, eg: ADJ"

m4_define('ENT_TRANS_TYPE_DEF','H')
m4_define('ENT_TRANS_TYPE_VALID','B,C,D,E,F,G,H,I,J,K,L,M,N')
m4_define('ENT_TRANS_TYPE_INCL','include=ENT_TRANS_TYPE_VALID), upshift')

m4_define('ENT_DEBIT_CREDIT_VALID','d,c')
m4_define('ENT_DEBIT_CREDIT_INCL','include=(ENT_DEBIT_CREDIT_VALID, downshift')
Define valid transaction types and valid debit/credit indicators.

CAUTION: The General Ledger programs use the values set in these macros. Do *not* change these values.

m4_define('FS_CODE_DEF', ' ')
m4_define('FS_CODE_ORD',
'BAL,JULY,AUG,SEPT,OCT,NOV,DEC,JAN,FEB,MAR,APR,MAY,JUNE,ADJ,CLS')
m4_define('FS_CODE_VALID', 'FS_CODE_ORD')
m4_define('FS_CODE_INCL', 'include=(FS_CODE_VALID), upshift')
m4_define('FS_CODE_EG', ', eg: BAL, JAN, ADJ.')
m4_define('FS_CODE_ALL_EG', ', eg: JAN, ALL.')
Defines the order of the months in your institution's fiscal year, and the abbreviations that you want to use on the comment lines for your screens. If your institution uses a fiscal year other than July-June, or if you use different abbreviations for the names of the months, you can change the values of these macros.

m4_define('PRD00_TEXT','"BAL"')
m4_define('PRD01_TEXT','"JULY"')
m4_define('PRD02_TEXT','"AUG"')
m4_define('PRD03_TEXT','"SEPT"')
m4_define('PRD04_TEXT','"OCT"')
m4_define('PRD05_TEXT','"NOV"')
m4_define('PRD06_TEXT','"DEC"')
m4_define('PRD07_TEXT','"JAN"')
m4_define('PRD08_TEXT','"FEB"')
m4_define('PRD09_TEXT','"MAR"')
m4_define('PRD10_TEXT','"APR"')
m4_define('PRD11_TEXT','"MAY"')
m4_define('PRD12_TEXT','"JUNE"')
m4_define('PRD13_TEXT','"ADJ"')
m4_define('PRD14_TEXT','"CLS"')

m4_define('PRD00_DESC','"Balance"')
m4_define('PRD01_DESC','"July"')
m4_define('PRD02_DESC','"August"')
m4_define('PRD03_DESC','"September"')
m4_define('PRD04_DESC','"October"')
m4_define('PRD05_DESC','"November"')
m4_define('PRD06_DESC','"December"')
m4_define('PRD07_DESC','"January"')
m4_define('PRD08_DESC','"February"')
m4_define('PRD09_DESC','"March"')
m4_define('PRD10_DESC','"April"')
m4_define('PRD11_DESC','"May"')
m4_define('PRD12_DESC','"June"')

m4_define('PRD13_DESC','"Adjustment"')
m4_define('PRD14_DESC','"Closing"')

Defines the abbreviations and the actual names of the fiscal posting periods that your institution uses. If your institution uses a fiscal year other than July-June, or if you use different abbreviations for the names of the months, you can change the values of these macros.

m4_define('FS_PRD_NUM_INCL', 'include=(0:14)')
m4_define('FS_PRD_NUM_EG', ', eg: BAL=0, JULY=1, AUG=2')

Defines the fiscal posting periods that appear on the comment lines of screens. If your institution uses a fiscal year other than July-June, or if you use different abbreviations for the names of the months, you can change the values of these macros.

m4_define('FS_YR_EG', ', eg: FS_YR_DEF.')

Defines the default or example, fiscal year that appears on the comment lines of screens. This macro uses the value of FS_YR_DEF (as defined in the macro file macros/custom/periodic).

m4_define('FUND_EG', ', eg: FUND_DEF.')

Defines the default, for example, fund number that appears on the comment lines of screens. This macro uses the value of FUND_DEF (as defined in the macro file macros/custom/table).

m4_define('SAE_DIST_DEF', 'B')
m4_define('SAE_DIST_VALID', 'A,B,D')
m4_define('SAE_DIST_INCL', 'include=(SAE_DIST_VALID), upshift')
m4_define('SAE_DIST_EX', '(A)mount only, (B)oth amount and distribute, (D)istribute only')

Defines valid codes for standard accounting entries.

CAUTION: The program uses the codes that appear in these macros. Do not change any of these values.

m4_define('SUBS_EG', ', eg: SUBS_DEF.')

Defines the default or example, subsidiary name that appears on the comment lines of screens. This macro uses the value of SUBS_DEF (as defined in the macro file macros/custom/table).

m4_define('SUBS_BAL_EG', ', eg: SUBS_BAL_DEF.')

Defines the default or example, subsidiary balance code that appears on the comment lines of screens. This macro uses the value of SUBS_BAL_DEF (as defined in the macro file macros/custom/table).

m4_define('SUBS_BAL_CLOSE_DEF','C')
m4_define('SUBS_BAL_CLOSE_VALID','A,C,N')
m4_define('SUBS_BAL_CLOSE_INCL','include=(SUBS_BAL_CLOSE_VALID), upshift')

Defines values for the Fiscal Calendar record relating to the closing of the Subsidiary Balance record.

CAUTION: The program uses the codes that appear in these macros. Do not change any of these values.

m4_define('SUBS_TOT_DEF','PAID')
m4_define('SUBS_TOT_VALID','BORD,ATHL,CWSP,NDSL')
m4_define('SUBS_TOT_INCL','include=(SUBS_TOT_VALID," "),upshift')
m4_define('SUBS_TOT_EG',', eg: SUBS_TOT_DEF.')

Defines the default subsidiary total code that your institution uses, and the examples and valid values that appear on the comment lines of screens. If the institution uses different total codes, you can modify the macro values to reflect your codes.

m4_define('SUBS_TOT_BFWD','BFWD')
Provides the total code that you want to use for balance forward amounts. Most institutions do not change this code.

m4_define('SUBS_TOT_PAID','PAID')
Provides the total code that you want to use for student payments. Most institutions do not change this code.

m4_define('VCH_EG',', eg: VCH_AP_DEF, VCH_SA_DEF.')
Provides the example values for journal types. You define the example values in the macro file macros/custom/table, and refer to them in this macro.

CAUTION: Although you can select example values, the values you select must exist in the Voucher table as delivered with standard CX.

m4_define('VCH_YE_DEF', 'AC')
m4_define('VCH_YE_VALID', 'AC,PC')
m4_define('VCH_YE_INCL', 'include=(VCH_YE_VALID), upshift')
m4_define('VCH_YE_EG', ', (AC) or (PC).')
Defines the journal types that the system uses for year-end processing.

CAUTION: The program uses the codes that appear in these macros. Do not change any of these values.

m4_define('fs_bal00_bgt', 'total_00')
m4_define('fs_bal00_act', 'total_01')
m4_define('fs_bal00_enc', 'total_02')
m4_define('fs_bal00_tot', 'total_03')
m4_define('fs_balnn_bgt', 'total_04')
m4_define('fs_balnn_act', 'total_05')
m4_define('fs_balnn_enc', 'total_06')
m4_define('fs_balnn_tot', 'total_07')
m4_define('fs_ytd_bgt', 'total_08')
m4_define('fs_ytd_act', 'total_09')
m4_define('fs_ytd_enc', 'total_10')
m4_define('fs_ytd_tot', 'total_11')
m4_define('fs_fy_bgt', 'total_12')
m4_define('fs_fy_act', 'total_13')
m4_define('fs_fy_enc', 'total_14')
m4_define('fs_fy_tot', 'total_15')
Provides information to the program about the fs_table.

CAUTION: The program uses the codes that appear in these macros. Do not change any of these values.

Macros in the *table* File

The following list describes the General Ledger macros located in the *table* macro file. The macros appear in this list in the order in which they appear in the macro file.

Note: The macro file $CARSPATH/macros/custom/table contains macros for all modules and applications (e.g., Personnel/Payroll and Recruiting and Admissions). This list contains only the macros that relate to General Ledger.

m4_define('ATYPE_DEF', 'ACT')

Defines the default amount type. Change this macro if you do not want a default type of ACT.

m4_define('ATYPE_VALID', 'ACT,ENC,BGT')
m4_define('ATYPE_INCL', 'include=(ATYPE_VALID), upshift')
Defines the valid amount types that display on the comment line of screens. Change this macro if you want to change the suggested amount types.

m4_define('ENT_DEF', 'CASH')
Defines the default entry type. Change this macro if you do not want a default type of CASH.

m4_define('ENT_VALID', 'AID, ADJ, BILL, CASH, CHK, CHR, FADJ, SADJ, SFWD, VOID')
m4_define('ENT_INCL', 'include=(ENT_VALID), upshift')
Defines the valid entry types that display on the comment line of screens. Change this macro if you want to change the suggested entry types.

m4_define('SUBS_DEF', 'S/A')
Defines the default subsidiary code. Change this macro if you do not want a default subsidiary code of S/A (student accounts).

m4_define('SUBS_VALID',
'"S/A","A/P","W/P","K/D","R/A","R/H","D/D","ADV","F/S","PIP","PIPC","NP-A"')
m4_define('SUBS_INCL', 'include=(SUBS_VALID, ALL), upshift')
Defines the valid subsidiary codes that display on the comment line of screens. Change this macro if you want to change the suggested subsidiary codes.

m4_define('SUBS_BAL_DEF', 'SB')
Defines the default subsidiary balance code. Change this macro if you do not want a default code of SB.

m4_define('SUBS_BAL_VALID', 'SB,MISC,EMPR,NPAY," "')
m4_define('SUBS_BAL_INCL', 'include=(SUBS_BAL_VALID), upshift')
Defines the valid subsidiary balance codes that display on the comment line of screens. Change this macro if you want to change the suggested subsidiary balance codes.

Macros in the *periodic* File

The following list describes the General Ledger macros located in the *periodic* macro file. The macros appear in this list in the order in which they appear in the macro file. These macros require updating every year.

Note: The macro file $CARSPATH/macros/custom/periodic contains macros for all modules and applications (e.g., Personnel/Payroll and Recruiting and Admissions). This list contains only the macros that relate to General Ledger.

m4_define('FS_YR_BEG_DEF', '07/01/1995')
m4_define('FS_YR_END_DEF', '06/30/1996')
Define the beginning and ending dates for the current fiscal year. You must update these macros every year during your year-end closing procedures.

CAUTION: Many CX defaults depend on these macros. To operate correctly, the defaults require that you update these macros promptly at year-end.

m4_define('FS_YR_PREV', '9495')
m4_define('FS_YR_CUR, '9596')
m4_define('FS_YR_NEXT, '9697')

Define the previous, current, and next fiscal year. You must update these macros every year during your year-end closing procedures.

CAUTION: Many CX defaults depend on these macros. To operate correctly, the defaults require that you update these macros promptly at year-end.

m4_define('FS_YR_DEF', 'FS_YR_CUR')
m4_define('FS_YR_VALID', 'FS_YR_NEXT, FS_YR_CUR, FS_YR_PREV, 8990, 9091, 9192, 9293, 9394, 9495 9596 9697')
m4_define('FS_YR_INCL', 'include=(FS_YR_VALID), upshift')

Define the valid years that appear on the comment line on screens. You must update these macros every year during your year-end closing procedures so the correct valid values appear.

m4_define('SUBS_PRD_OTH', 'PREV')
m4_define('SUBS_PRD_DEF', 'SESSYR_DEF')
m4_define('SUBS_PRD_VALID', 'SESSYR_VALID,INV,SUBS_PRD_OTH')
m4_define('SUBS_PRD_INCL', 'include=(SUBS_PRD_VALID), upshift')
m4_define('SUBS_PRD_ALL_INCL', 'include=(SUBS_PRD_VALID,ALL), upshift')

Use values from the common macro file to define valid subsidiary sessions.

CAUTION: The program uses the codes that appear in these macros. Do not change any of these values.

Configuration Table Entries

Introduction

Although most General Ledger features and processes rely on macros, a few programs access the Configuration table for processing instructions. This section contains descriptions of the Configuration table entries that the General Ledger product uses.

Features that Use the Configuration Table

If you use any of these features, define the specified values in the Configuration table:

- Institution-wide reporting department
- Class code as part of the account number for 15 month appropriation years
- Claim on cash
- General Ledger Account Auto-Fill

Institution-wide Reporting Function Code (Department)

CX can create an aggregate function code (usually considered a department) that sums budget information from all other functions. The charges to the function consolidate amounts by function while preserving separate accounts.

Example: An institution uses the following setup:

- Tracks budget items by source (e.g., as general revenue or as education revenue).
- Maintains the revenue source as another part of the account number (e.g., the Type).
- Uses the institution-wide (aggregate) function code to define a reporting department.

Users could view or report amounts for all functions combined, while having the ability to identify amounts by source. Users do not post any amounts directly to this aggregate department; its purpose is to facilitate budget reporting.

After an institution sets up this feature, all budget transactions to individual departments are duplicated and posted *as memorandum entries only t*o the aggregate department. Then, users can review balances in the aggregate department, rather than having to create reports or perform queries that summarize a large number of departments. The use of the aggregate department allows CX to produce budget reports and queries as quickly and efficiently as possible.

Set the following values in the Configuration table to use the aggregate reporting department:

ENABLE_SOURCE_FUNDS_BGT
Implements the institution-wide (aggregate) function code for budget reporting purposes.

INST_WIDE_FUNC
Defines the function number that you want to use as your aggregate function code (e.g., 9999).

Class Code to Track 15-month Appropriation Years

If the institution uses a 15-month year to track appropriations, you must track appropriation year as the Class. You must therefore set the following parameter:

GL_CLAS_ENABLE
Implements the use of the Class field to track appropriation year. Set the value to Y to use this feature.

Claim on Cash

The claim on cash feature enables users to track the sources and uses of funds that cross function line. For more information about claim on cash, see *Setting Up Claims on Cash, Receivables and Payables* in this guide.

To implement claim on cash, you must set the following Configuration table value:

CLAIM_ON_CASH
Indicates that the institution uses the claim on cash feature to generate entries. Define the value as Y in the Configuration table to implement the feature.

General Ledger Account Auto-Fill

The General Ledger Account Auto-Fill feature automatically defaults data into one or more segments of the General Ledger account number. Based on instructions in the glsub_table, the feature keys off the account segment entered by the user and automatically places one or more of the remaining segments into the account number field. For example, if a line item (object code) is only used in a particular cost center (function code), you could define the relationship between the line item and the cost center in the glsub_table. Then, when the user enters the line item in one segment of the account number field, the feature would automatically place the cost center in the appropriate field.

To implement Account Auto-Fill, you must set up the glsub_table and also define the following Configuration table value:

GL_ACCT_AUTO_ENTRY
Indicates that the institution uses the General Ledger Account Auto-Fill to automatically fill data into one or more segments of the General Ledger account number. Define the value as Y in the Configuration table to enable the feature.

General Ledger Includes

Introduction

The General Ledger module contains includes that determine the features that are enabled in the module. 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 General Ledger module, you must define an include in $CARSPATH/include/common. To disable an include, comment out the include in the same file. See the *Implementing Jenzabar CX* section in the *Jenzabar CX Technical Manual, Volume 1*, for more information on enabling and disabling includes. By using includes, the General Ledger module is easier for you to customize and maintain.

Purpose

An include activates or deactivates features in C programs without changing the C code. It can also specify compilation values for some entry programs in CX.

Macro Dependency

Includes have a dependency on macros. In the General Ledger module, you do not directly modify includes for the module. You must modify a corresponding macro's value and then reinstall the dependent include.

SECTION 4 - FINANCIAL REPORTS

Overview

Introduction

This section lists the types of accounting reports that appear on the Accounting: Reports Menu and describes the contents of these reports. Topics in this section include the following:

- Information about initializing report indexes
- Descriptions of the accounting reports and features available from each reports menu
- Descriptions of the macros associated with these accounting reports
- Information about using the macros
- Information about issues that arise when using macros

Initializing Report Indexes

The Initialize Report Indexes menu option uses a script to create the temporary file *glatemp_rec*. The purpose of the glatemp_rec file is to enable the accounting reports to run efficiently.

The glatemp_rec file stores all General Ledger account combinations for two fiscal years, that is, the fiscal year that you enter as a parameter and the fiscal year immediately preceding that fiscal year. For example, if you enter 9697 as the parameter when you initialize report indexes, the system adds all General Ledger account combinations for fiscal years 9697 and 9596 to the glatemp_rec file.

If the user runs a report for fiscal years other than those in the glatemp_rec, the output may contain unreliable amounts. For example, if glatemp_rec contains fiscal years 9697 and 9798 and you enter 9697 as the fiscal year parameter, the selected amounts will be 9697 fiscal year amounts. If the report contains a prior year column output, the prior year amounts (i.e., those for 9596) will be zero since they are not in glatemp_rec.

The system deletes all existing records from the glatemp_rec file before adding new records. This clears all records added previously. As a result, you can rerun the Initialize Report Indexes option at any time. Jenzabar recommends that you initialize the indexes before each report run (e.g., at least monthly).

CAUTION: Make sure you do not rerun the option while you are running one of the reports using the glatemp_rec file.

Although important to reporting in the accounting area, some reports do not use the glatemp_rec. For example, transaction reports and budget reports, in which macros are used extensively, do not use the glatemp_rec file since one report can span more than two fiscal years.

Most reports, with the exception of the Transaction reports, use the glatemp_rec file to provide the data they display.

Row and Column Output

Row output refers to the individual lines on a report. For example, a Trial Balance contains a row, or line, for each fund/function/object/subfund. Summary reports may include rows of subtotals only. Object reports contain rows for assets, liabilities, expenses or revenues, and function reports contain rows for departments or other areas of responsibility at the institution.

Column output refers to the type of information presented for each row. For example, the Trial Balance may include current year-to-date and prior year-to-date information. However, the Trial Balance could also include current year-to-date and the prior year's totals. Budget or encumbrance figures could also appear on the report. The column output for reports is maintained in report macros, so changes can be maintained in a single location, the macro file

macros/custom/finrpt. For most reports, users designate the type of column output they desire each time they run the report. Since a variety of column outputs are defined in standard CX, the reports are very flexible in providing information.

When users select a report menu option with the desired row output, they can designate the type of column output they want to display. The Jenzabar coordinator can create additional column output macros that meet the needs of the institution.

Account Number Ranges

Users can request reports for a range of funds, functions, objects, and subfunds, in any combination and at any time. The only exception for the timing of report processing is that users should not run reports when the script to initialize indexes is running. For consistency, report parameter screens prompt the user to enter all ranges in the order of fund, function, object, and subfund, whether it is a function, object or subfund type report.

Non-Display Accounts

CX accommodates a user's need to restrict the display of some accounts on financial reports. A field in the General Ledger Account tables controls the display of the account in the Budgeting module. The accounting reports use the same field to determine whether or not to display the account.

Subtotals by Block, Group and Schedule

Some CX General Ledger reports can include subtotals at the block, group and/or schedule level. Transaction and some combined center and associated function reports produce detail information only.

Sorting by Responsible Person

Based on responses to prompts on the report screen, users can produce reports sorted and totaled for the person responsible for functions, objects, or subfunds. The person can have either primary or secondary responsibility.

Trial Balance

The Trial Balance options can produce detailed reports. They also can span the full account range, from the beginning of assets through expenditures. As a result, every General Ledger account combination (fund/function/object/subfund) displays a line item.

Object Reports

Object reports total the amounts for each account. Within an account group, a line item exists for each general ledger account and identifies the function to which it belongs. If subfunds are requested, the subfund is also identified. Each account begins on a separate page. The condensed versions, the Account Summary reports contain only the totals for each account.

Detail by Month Report

The Detail by Month option has its own separate source file because it uses different column output (months vs. budget/actual). It does not have different columns for the actual, budget and variance amounts. Instead, it has separate rows for each of these items because the individual months (or fiscal periods) use up all of the columns.

The following example shows how the Detail by Month report columns appear, with the standard object detail appearing first.

```
                          Object Detail Report
6000  Salaries

                                  |-------- Year to Date ----------|
Func Description                     Actual      Budget    Variance
---- --------------------------  ----------- ----------- ----------
1001 Art Department                  $15,000     $20,000     $5,000
```

```
                          Object Detail by Month
NOTE:  The row marked "B" is for Budget, "A" is for Actual

   BAL+July    August   September  October      . . .      June+ADJ
  ---------- ---------- --------- ----------    . . .     ----------
6000 Salaries                                   . . .
                                                . . .
B     10,000      3,000     1,000      1,000    . . .         10,000
A      9,000      2,950     1,200      2,000    . . .          9,000
  ---------- ---------- --------- ----------    . . .     ----------
       1,000         50      (200)    (1,000)   . . .          1,000
```

Function Reports

Function reports total the amounts for each function, or center. Within a function, a line item exists for each general ledger account and identifies the function to which it belongs. If subfunds are requested, the subfund is also identified. Each function begins on a separate page.

The Detail/Transactions and Profit Center options are the only two special types of reports on the Accounting: Function Reports Menu. The Detail/Transactions option is a combination of the Detail and the Transactions options. They are combined and sorted together for distribution to individual functions, or departments.

The Profit Center option is a summary report that prints a function if revenue and expenditure accounts are found for it.

Subfund Reports

Most of the subfund reports resemble the object and function summary and detail reports. The main difference is that the subfund summary sorts first and prints in the page header.

The Summary and Summary by Funds reports, which are similar to a function summary, print one row for each subfund. The Summary and Summary by Funds options are the only two using the SUBFUND logic, that is, the grouping and subtotaling of projects resulting from the use of a SUBFUND categorization in the fs_table.

Associated Object Reports

The Associated Object reports use a table with an associated code (glas_table) to determine which General Ledger accounts to select. This table must include a code associated with any number of General Ledger objects. For example, when considering bookstore (BOOK) accounts, the selected objects can be assets, liabilities, revenues or expenditures. The type of report selected determines how these accounts appear. For example, if the user selects a revenue and expenditure statement for the code BOOK, then only the revenue or expenditure General Ledger objects appear in the report.

Combined Functions Reports

The Combined Functions reports are similar to the Associated Objects reports. However, the Combined Functions table with its associated code only contains function codes, not the full General Ledger account. This enables users to produce reports that print the totals of functions that are not ordered numerically. The reports allow either the printing of one row for each function (detail), or only printing one row for the total of the combined functions (summary).

An example of the object detail with the detail option follows:

```
                    Combined Object Detail
6000  Salaries

Func  Description                  Budget     Actual
----  -------------------------  ---------- ----------
1001  Art Department                   1000       2000
1002  Biology Department                500       1000
1003  Chemistry Department             2000       1000
                                  ---------- ----------
ABC   Art/Biology/Chemistry            3500       4000
```

An example of the summary option follows:

```
                    Summary Object Report
6000  Salaries

Func  Description                  Budget     Actual
----  -------------------------  ---------- ----------
ABC   Art/Biology/Chemistry            3500       4000
```

Revenue/Expense Reports

The Revenue/Expense summary options are a combination of an object summary and a function summary report. The revenue output is an object summary (one row per object). The expenditures output is a function summary (one row per function).

The Revenue/Expense Detail and the Rev/Exp Detail by Funds options display a separate row for every combination of object, function, and subfund.

The following example shows a standard object summary report (and the revenue side of the revenue and expenditure statement).

```
                     Object Summary Report
Obj  Description                  Actual
---- ----------------------------  ---------
5001 Tuition - Undergraduate         $500,012
5002 Tuition - Graduate              $287,185
5003 Other Fees                       $29,946
```

This example shows a sample detailed version of the same object numbers shown in the preceding example.

```
              Object Report
Obj  Description                    Actual
---- ----------------------------  ---------
5001 Tuition - Undergraduate
     1001 Art Department             $120,928
     1002 Biology Department         $185,280
     1003 Chemistry Department       $137,872
  ...
  Total Tuition - Undergraduate      $500,012

5002 Tuition - Graduate
     1010 English Department          $62,280
       SPEC Special Project            $8,198
     1090 Computer Center             $81,872
  ...
  Total Tuition - Graduate           $287,185

5003 Other Fees
     1010 English Department          $29,946
  Total Other Fees                    $29,946
```

:

Report Formatting and Macros

Introduction

To effectively use report macros and format reports, you must know the following:

- The components of CX reports
- The purpose and location of macros
- The types of macros
- How to use and change macros
- The issues that relate to macro usage

Purpose of Report Macros

Macros enable you to do the following to accounting reports:

- Change the text descriptions
- Allow you to customize how reports work at the institution (e.g., define the beginning and ending objects for expenditures)

Report Source File

To accommodate your needs, and to make maintaining the report macros an efficient process, multiple kinds of reports can be combined in one source file. This allows one source report, regardless of whether sorting is done by responsible person, by fund, or by some other criteria. If all of the various combinations of reports required separate source files, maintaining the macros would be extremely difficult. In addition, each version would require a separate menu option, complicating the menu structure.

CAUTION: Without exception, change *only* the macros identified in this section. Never change a macro in the $CARSPATH/macros/user/finrpt file that follows the warning line shown below:

```
*****************************************************************************
** BEGINNING OF ACCOUNTING/REPORTS MACROS THAT SHOULDN'T NEED MODIFICATION **
*****************************************************************************
```

The file macros/user/finrpt contains revisions to text descriptions (such as changes to wording of text that appears on a screen). Changes to the macros in this file can cause errors in dollar amount calculations and other serious problems with CX reports.

Record Selection for Reports

On all the General Ledger reports generated from reporting macros, an account prints if *any* record exists for either the year being run, or the year prior to it. The account prints even if the column output specified does not include previous year amounts. This printing occurs because the system first reads all records for both fiscal years, and then prints all records in the format specified by the macro.

Macros and Accounting Report Sections

Each section of CX accounting reports uses macros to define the appearance and content of the output. Accounting reports contain the following sections:

- The DEFINE section
- The OUTPUT section
- The READ section
- The SORT section

- The FORMAT section

DEFINE Section

The macros used in the DEFINE section of the accounting reports begin under the following heading in the file $CARSPATH/macros/user/finrpt:

```
*********************************************************************
*************** Macros Used in the Define Section of ACE. ***********
*********************************************************************
```

The DEFINE section includes two types of macros. The first type is the the REP_ACCT_PARAM, which defines parameters. The second type includes all others, which are used to define the required variables.

REP_ACCT_PARAM Macro

The REP_ACCT_PARAM macro expands to include the correct number of parameters based on the type of column output. REP_ACCT_PARAM has a number of related macros. As a result, using these macro combinations is more complex; however, their basic function is to determine which parameters are required for each of the different report types. For example, fund reports uses many more parameters than non-fund reports.

Not all parameters passed by the menu option are handled as normal ACE parameters; the program does not define a parameter number for them. Instead, the menu option passes a string to change along with the value to which to change it.

For example, when selecting a report, the user can specify whether to sort the report by responsible person. The part of the menu option that handles this option looks like this: OPT_SORT_RESP(acct); the option is not defined as an ACE parameter.

In this example, the OPT_SORT_RESP macro expands as follows when used in the menu option:
PP=
PA=PA_DEFINE(SORTFIELD)
PP=Enter responsible person sort field, (prim)ary, (sec)ondary or blank
PA=length 4, dwshift, include=(prim,sec," ")

This causes any place in the report (or in the macros that the report has in it) to change the value of "SORTFIELD" to the entered value (prim, sec or blank).

Other DEFINE Section Macros

Other DEFINE macros also define the required variables. These other macros include the following:

- REP_ACCT_DEFINE
- REP_VAR
- REP_VAR_TEMP

REP_ACCT_DEFINE Macro

All the acounting reports use the REP_ACCT_DEFINE macro. It includes a variety of variables; not every report, however, uses each of these variables.

REP_VAR Macros

The REP_VAR macros define the arrays used for storing all the various amounts available within the account type (ACCT, ACCTPRD, ACCTFD and BGT). Reports require a separate REP_VAR macro for each subtotal needed. Different ACE reports have different numbers of REP_VAR macros, depending on how many subtotals they allow.

The REP_VAR_TEMP macros provide temporary storage. While the program stores the actual amounts in separate arrays; calculations are sometimes done on some of these amounts. Calculations must be done outside the array, so the program calls another macro, the REP_GET_DETAIL macro, to transfer the amount from the array into the temporary variable. The program uses the temporary variable for the calculation.

OUTPUT Section Macros

The standard General Ledger module uses OUTPUT Section macros that do *not* require modification. The merged function and subfund reports use an additional feature called *sortpage*. *Sortpage* allows the report output to be resorted so that the line amounts and the transactions for each individual center can be merged.

Sortpage uses additional lines in the report's output. These lines are deleted later as *sortpage* uses them. As a result, for each individual sort criteria, one additional line must be added to the page length. For example, if you have six sort fields to be used by *sortpage*, the page length changes to 66 + 6. After adding one line for spacing differences, the total number of lines becomes 73.

Note: The number of *sortpage* sort fields changes, depending on whether the user sorts the report by responsible person, and whether an ACCTFD column output is used. (The report uses PRESORT_SORT_BY for this purpose. PRESORT_SORT_BY must be located before the OUTPUT section of the ACE so that the page length can be adjusted correctly).

Both PRESORT_SORT_BY and REP_CHECK_SORT_RESP must precede the OUTPUT section. REP_CHECK_SORT_RESP determines whether to sort the report by responsible person. REP_CHECK_SORT_RESP also determines the number of sort fields; consequently, it affects the page length.

READ Section Macros

READ section macros include the following:

- REP_ALIAS
- REP_READ_FS_GROUP_BEFORE
- REP_OBJ_READ
- REP_READ_SORT_RESP
- REP_JOIN_SORT_RESP
- REP_READ_DISPLAY
- REP_READ_FS_GROUP_AFTER

REP_READ_FS_GROUP_BEFORE

REP_READ_FS_GROUP_BEFORE follows REP_ALIAS. This macro reads the fs_table so that the system can determine blocks, groups and schedule boundaries. One of the parameters passed to this macro is the OBJ, FUNC or SUBFUND sort sequence code (axis) for the report to use.

Note: Not all function reports use the FUNC axis in reporting. A function summary does use the FUNC axis; however, a function detail report would use the OBJ axis. The system uses this axis because the individual lines that print in a function detail report are objects, and users can request subtotals by the group or schedule from the OBJ axis.

REP_ACCT_READ

The REP_ACCT_READ macro performs the main read clause which reads in the fields needed by the ACE report. It adjusts automatically to read either the glatemp_rec or the bgtamt_rec file based on the account type (BGT type only reads the bgtamt_rec). To determine the correct responsible person fields, the program uses two macros: (REP_READ_SORT_RESP and REP_JOIN_SORT_RESP).

Note: The responsible person macros have associated parameters. Within this macro, the macro REP_READ_SORT_RESP has "($3)" after it. This means that it uses the third parameter passed to the REP_ACCT_READ macro as its first parameter. Within the REP_READ_SORT_RESP macro, it defines the fields to sort by as "$1_prim_resp_id". The third parameter to REP_ACCT_READ is the file to be used within the report.

Example: In the case of an object report, which passes "obj" as the third parameter, the REP_READ_SORT_RESP to be defined as obj_prim_resp_id.

A fourth parameter passed to REP_ACCT_READ is the axis used in the fs_table read. This parameter joins the two files together, since no other common field exists between the read with the fs_table and the read with the amounts.

The fifth parameter can pass additional restrictions to the selection criteria.

REP_READ_DISPLAY

The REP_READ_DISPLAY macro discards unwanted records from the selection. REP_READ_DISPLAY uses the parameter "want_display" to determine if the user wants the non-display accounts. It also clears records with no responsible person if the user requests the report to be sorted by responsible person, and also enters **N** at the prompt, "Do you want to include on report if no responsible person?".

REP_READ_FS_GROUP_AFTER

REP_READ_FS_GROUP_AFTER causes the fs_table read fields to join to the amount read. This causes multiple records for each General Ledger account since, for each record, there are three fs_records (one with each fs_type: S, B, and G).

SORT Section Macros

The only macro used for ACE sorting is the REP_ACCT_SORT macro. REP_ACCT_SORT automatically adjusts to sort by the responsible person and/or fund, depending on the parameters that the user enters.

FORMAT Section Macros

The FORMAT Section of ACE uses the following phrases:

- The FIRST PAGE HEADER
- The PAGE HEADER
- The BEFORE GROUP OF
- The ON EVERY RECORD
- The AFTER GROUP OF
- The ON LAST RECORD.

FIRST PAGE HEADER

INFORMIX calls the FIRST PAGE HEADER an ACE phrase. This phrase is used as follows:

- To produce the first page, which contain the parameters along with their answers.
- To initialize some variables (REP_LAST_REC is used in most ACE reports to revise and locate information to be printed in the page trailer). Since a separate page lists the parameters, the revision and location information appear at the bottom of this page. The

rep_pageno variable is set to -1 to ensure that this parameter page is marked as page zero (0) and that page one (1) begins the actual report output.

REP_FORMAT

The REP_FORMAT macro determines the width of the page header information. The system passes the parameter REP_WIDTH to it. Each column output defines this REP_WIDTH so that the page header is the same width as the output.

REP_VAR

The REP_VAR macro initializes the arrays that ACE uses in the report.

REP_NUM_SPACES

REP_NUM_SPACES determines how many spaces are needed to print in front of each line of the parameter page so that it will be centered with the REP_WIDTH output.

REP_PARAM_HEAD

REP_PARAM_HEAD produces the heading "RUN TIME PARAMETERS USED".

REP_OBJ_TYPE_HEAD
GL_OBJ_PARAMS_HEAD
REP_NOTE_COMMENT

These macros are located at the bottom of macros/user/finrpt. These macros print the parameter prompt text and the associated parameter value that the user entered.

The same macro prints the parameter prompts within the menuopts and the text on the first page header. This enables you to change the wording to the parameter in one place and update it in both locations. For example, REP_OBJ_TYPE_HEAD uses the macro OUTPUT_TYPE_TEXT for the text. The default value for the macro is "Enter type of column output".

To change the wording to "Enter column output type desired", do the following:

1. Go to $CARSPATH/macros/user.
2. Check out the "finrpt" file.
3. Update OUTPUT_TYPE_TEXT.
4. Check the file back in, reinstall the menuopt files and reinstall the menusrc files.

The changed text now appears in the menu options as well as in the first page header section.

Note: Some reports have more macros in the first page header because of additional parameters needed for the report.

GL_ACCT_PARAMS_HEAD

GL_ACCT_PARAMS_HEAD prints the beginning and ending fund, function, object and subfund values you entered. If you are using fund column output, then the beginning and ending fund values are omitted.

REP_NOTE_COMMENT

REP_NOTE_COMMENT is a free-format macro that you can use to print a message, note or warning to those who view the reports. It is printed only on the first page header; therefore, it does not print on each individual page. The default message for this macro alerts the person who ran the report that the output is different than expected, and to run the Initialize Report Indexes option on the menu and to rerun the report.

Note: Keep this message concise. Lengthy messages can cause some reports to be too large, displaying the error, "user string space".

PAGE HEADER Macros

The PAGE HEADER macros print information at the top of each page. Some of the macros translate differently, depending on the column output and/or whether the report must sort by responsible person. For example, if the report must sort by responsible person, REP_PRINT_RESP_DTL (used in acctdtl and cntrdtl) expands to print only the name and ID number of the primary or secondary responsible person, depending on the selected sort. If the report does not sort by responsible person, the macro expands to print both the primary and secondary responsible person or persons for that object or function.

Some reports also print a different page header on the last page. The formatting for these reports is controlled by the following macros:

- REP_TOTALS_PAGE_TEXT
- REP_RESET_LAST_PAGE (in the page header)
- REP_LAST_PAGE (at the bottom of the report)

These macros can cause the page header to print the following:

- Grand totals page
- Total for responsible person
- Total by fund

The print values depend on the column output selected and/or the report sorts by responsible person.

REP_HEAD_DESC
REP_HEAD_DASH

Two macros, REP_HEAD_DESC and REP_HEAD_DASH print the correct column headings in the upper left portion of the page. For example, in an account detail report, the macros REP_HEAD_DESC_ACCT_DTL and REP_HEAD_DASH_ACCT_DTL cause the following to print:

```
Obj/Subfund Description
-----------------------
```

These macros appear in macros/custom/finrpt because of the need to print the correct number of dashes under the account and center codes for those who do not use four digit codes.

Note: Do not modify these macros. As part of standard CX, they are set to GL_OBJ_DASHES and GL_OBJ_TITLE, which are set automatically in macros/user/acct.

REP_HEAD

The REP_HEAD macro is the first macro in the file with "32" in it. The 32 is a parameter to the macro identifying where the amounts are to begin printing.

CAUTION: Do not change this number; Jenzabar does not support other start positions.

Note: The detail by month reports *acctdtlmon* and *cntrdtlmon* reports have start positions set at 2, since 130 characters are needed for printing the dollar amounts.

BEFORE GROUP OF Macros

The BEFORE GROUP OF macros vary depending on the type of report; however, explanations of some of the macros follow.

REP_SKIP_FIRST_PAGE

The REP_SKIP_FIRST_PAGE macro skips the rest of the first page header so that the detail begins on the top of the next page. It also resets the rep_record variable back to zero

so that the revision and location information does not print on the bottom of every other page.

REP_SKIP_TO_TOP_OF_PAGE

The REP_SKIP_TO_TOP_OF_PAGE macro issues a "skip to top of page" command after printing one blank line. In case the detail prints to the very last line of the page, the next page header does not contain incorrect information.

REP_CHECK_REVEXP

The REP_CHECK_REVEXP checks whether any fund balance, revenue or expenditure accounts were found. Each time a new account is printed in some of the reports, it checks to determine if there has been revenue printed and if a subtotal is required.

ON EVERY RECORD Macros

All the reports described in this section use the ON EVERY RECORD clause to accumulate totals and store them into the defined arrays. The amounts stored are based on the column output type specified, which relates to one of the four types (ACCT, ACCTFD, ACCTPRD or BGT).

REP_FSCODE

The REP_FSCODE macro stores the text descriptions and the beginning and ending numbers of the current block, group and schedule.

In the last read clause, each amount record is reproduced in triplicate with three records from the fs_table. As a result, each amount record has one record with the code of B, one with the code of G, and one with the code of S, corresponding to the beginning and ending number surrounding the object number.

Later, the report compares the current values of the block, group, and schedule with the previous values. If they have changed, the subtotals are then printed, then zeroed out for the next batch. Next, each report uses the REP_ACCUM_AMTS macro to accumulate the amounts into the appropriate array positon. An *if* section checks for the tfs_grp_code of S, since this code appears only once (if the *if* section did not perform this check, the amounts would triple).

REP_FACTOR

The REP_FACTOR macro is directly above the REP_ACCUM_AMTS macro. REP_FACTOR enables institutions to determine if they want to display revenues as positive numbers and expenditures as negative numbers. In standard CX, revenues, as credits, are stored as negatives, and expenditures, as debits, are stored as positives. The macros REV_FACTOR, ASSET_FACTOR, and EXP_FACTOR appear in macros/custom/finrpt.

AFTER GROUP OF Macros

A variety of reports use macros in the AFTER group of clauses. One of these is the REP_GRP_SCHED, which is used in the object and function detail reports.

REP_GRP_SCHED

REP_GRP_SCHED checks whether the current group or schedule is different from the previous group or schedule. If it is, and the report requires subtotals, then subtotals print and the report stores a new current group or schedule.

REP_BLK_GRP
REP_BLK_GRP_AFTER

The object summary reports use macros like REP_BLK_GRP and REP_BLK_GRP_AFTER, which are similar to REP_GRP_SCHED with one exception: they check block and/or group subtotaling.

REP_BLK_GRP_SUM
REP_BLK_GRP_AFTER_SUM

The function summary reports use macros similar to those used by the object summary reports. For example, function summary reports use REP_BLK_GRP_SUM and REP_BLK_GRP_AFTER_SUM. These two macros check revenue and/or expenditure amounts and print two lines and a variance of the two if both are found, as in the following example:

```
    Instruction

    Cntr Description                Budget    Actual  Encumber Variance
    ---- -------------------------  --------  --------- -------- ---------
    1001 Art Dept
           Revenue                  200,000   100,000        0  (100,000)
           Expenditures             100,000   200,000        0  (100,000)
    1002 Anthropology Department
           Expenditures              10,000   100,000        0   (90,000)
                                    -------- --------- -------- ---------

    Instruction
           Revenue                  200,000   100,000       0  (100,000)
           Expenditures             110,000   300,000       0  (190,000)
```

REP_DETAIL

The REP_DETAIL macro prints the information based on the column output that you specify. Some reports might not have this macro in the source of the report because the REP_DETAIL macro is called within other macros.

REP_ACCUM
REP_ACCUM_TYPE
REP_INIT

Other macros (e.g., REP_ACCUM and REP_ACCUM_TYPE), accumulate the amounts from one array into another array. Usually, after using these macros, the report calls a function (_varstore) to zero out the arrays.

ON LAST RECORD Macros

The ON LAST RECORD macros print the totals after the report finishes. One macro used in most of the reports (except subfund reports) is called REP_OBJ_TOTALS_CLAUSE. This macro expands to *on last record*, *after group of name*, or *after group of fund*, depending on whether fund column output is used and/or the report requires a sort by responsible person.

If the report sorts by responsible person, then the macro becomes *after group of name*. The macros following then print the totals and initialize the total macros. Since the report is not finished, it begins again with new totals for the name of the next responsible person. If the report is not fund column output, the REP_OBJ_TOTALS_CLAUSE macro expands to *after group of fund*, since a range of funds may have been specified. The totals then print and initialize, and the report continues running.

If the report is a fund column output and the sort was *not* by responsible person, then the REP_OBJ_TOTALS_CLAUSE macro expands to *on last record*. This occurs because all the fund totals are within the columns; therefore, the totals up to this point are the final report totals. There can be no more than one occurrence of any phrase. Consequently, there cannot be another *on last record* clause. Therefore, if the report were one of the first two cases, there would be no grand total for the report.

Column Output

Introduction

The CX product includes a variety of financial reports and budget reports. Most of the reports allow for several different formats of output based on the response to prompts when the users select report menu options. Generally the appearance of a report is determined by three factors: rows, columns and macros.

Column Output

When reports with multiple column output are run, the user is prompted for column output desired. Most of the financial reports allow the user to enter one of many predefined column outputs, but some reports are specialized and their columns are hard-coded. Of the reports with variable column outputs, column outputs can be modified or added, based on the users' requirements.

Types of Column Output

There are 4 types of column output formats. The term 'type' means a group of column outputs which have a certain number of variables in common. The four types are:

- OBJ
- OBJPRD
- OBJFD
- BGT

Accounting Type Reports (OBJ)

This type has the most varied amounts, and the largest number of standard formats. All amounts are taken from the General Ledger Amount record (glamt_rec) and are used strictly in accounting reports.

The following list details which amounts are stored (and therefore available to print) for the OBJ type.

```
actytd      - Current Year      Year-to-date      Actual (ACT)
actprvbal   - Current Year      Previous Balance  Actual (ACT)
actprd      - Current Year      Month             Actual (ACT)
actpytd     - Prior Year        Year-to-date      Actual (ACT)
actpprd     - Prior Year        Month             Actual (ACT)
actptot     - Prior Year        Total Year        Actual (ACT)
actpprvbal  - Prior Year        Previous Balance  Actual (ACT)
bgtfy       - Current Year      Total Year        Budget (BGT)
bgtytd      - Current Year      Year-to-date      Budget (BGT)
bgtprvbal   - Current Year      Previous Balance  Budget (BGT)
bgtprd      - Current Year      Month             Budget (BGT)
bgtpytd     - Prior Year        Year-to-date      Budget (BGT)
bgtpprd     - Prior Year        Month             Budget (BGT)
bgtptot     - Prior Year        Total Year        Budget (BGT)
bgtpprvbal  - Prior Year        Previous Balance  Budget (BGT)
encytd      - Current Year      Year-to-date      Encumberance (ENC)
encprvbal   - Current Year      Previous Balance  Encumberance (ENC)
encprd      - Current Year      Month             Encumberance (ENC)
encpytd     - Prior Year        Year-to-date      Encumberance (ENC)
encpprd     - Prior Year        Month             Encumberance (ENC)
encptot     - Prior Year        Total Year        Encumberance (ENC)
encpprvbal  - Prior Year        Previous Balance  Encumberance (ENC)
```

The amounts including the letters *prd* are for one month only (e.g., actprd), while the amount types including the letters *prvbal* is for the months up to but not including the ending month parameter. For example, to obtain the period amount, take the ytd amount - prvbal amount (e.g., actytd - actprvbal).

Some of the amounts enable printing variance amounts between ACT and BGT types. They consider the reversal of signs (for Revenue and Expenditure) and switching the formula for Revenue and Expenditure.

- For Revenue, the formula is A - B.
- For Expenditure, the formula is B - A.

```
                                                A               B
       prddiff     - Difference between (actprd + encprd) and bgtprd
       ytddiff     - Difference between (actytd + encytd) and bgtytd
       totdiff     - Difference between (actytd + encytd) and bgtfy
                                          A         B
       pprddiff    - Difference between actpprd and bgtpprd
       pytddiff    - Difference between actpytd and bgtptot
       ptotdiff    - Difference between actptot and bgtptot
```

A positive variance amount is desirable, and a negative variance amount is undesirable. The following example assumes both revenue and expenditures are printing as positive.

```
                                    Budget          Actual          Variance
Revenue                              100             200             100
Expenditures                         100             200             (100)
```

For revenue, if you budgeted to receive $100, but you actually received $200, you have a $100 desirable variance. For expenditures, if you budgeted to spend $100, but you actually spent $200, you have a $100 undesirable variance.

Period (Monthly) Type Reports (OBJPRD)

This type provides column information about each individual month or period. The information can come from either General Ledger amount records (glamt_rec) or budget amount records (bgtamt_rec). The information can be printed in any ranges or columns. For example, if you want to print quarterly amounts in each column, you could take the first three months added together to make the first column, then the next three months added together for the second column.

The report determines the periods from the beginning of the fiscal year. If the fiscal year begins on July 1, then period 1 is July, and period 2 is August. This information is stored in macros which are customized at installation time.

Users designate the fiscal year as a parameter at the time the report is run. The standard report uses one fiscal year at a time.

The following list details which amounts are stored (and therefore available to print) for the OBJPRD type.

```
prd01_act  - Specified Year     1st Period        Actual (ACT)
prd02_act  - Specified Year     2nd Period        Actual (ACT)
prd03_act  - Specified Year     3rd Period        Actual (ACT)
prd04_act  - Specified Year     4th Period        Actual (ACT)
prd05_act  - Specified Year     5th Period        Actual (ACT)
prd06_act  - Specified Year     6th Period        Actual (ACT)
prd07_act  - Specified Year     7th Period        Actual (ACT)
prd08_act  - Specified Year     8th Period        Actual (ACT)
prd09_act  - Specified Year     9th Period        Actual (ACT)
prd10_act  - Specified Year     10th Period       Actual (ACT)
prd11_act  - Specified Year     11th Period       Actual (ACT)
prd12_act  - Specified Year     12th Period       Actual (ACT)
prdtot_act - Specified Year     All Periods       Actual (ACT)
prd01_bgt  - Specified Year     1st Period        Budget (BGT)
prd02_bgt  - Specified Year     2nd Period        Budget (BGT)
prd03_bgt  - Specified Year     3rd Period        Budget (BGT)
prd04_bgt  - Specified Year     4th Period        Budget (BGT)
prd05_bgt  - Specified Year     5th Period        Budget (BGT)
prd06_bgt  - Specified Year     6th Period        Budget (BGT)
prd07_bgt  - Specified Year     7th Period        Budget (BGT)
prd08_bgt  - Specified Year     8th Period        Budget (BGT)
prd09_bgt  - Specified Year     9th Period        Budget (BGT)
prd10_bgt  - Specified Year     10th Period       Budget (BGT)
prd11_bgt  - Specified Year     11th Period       Budget (BGT)
prd12_bgt  - Specified Year     12th Period       Budget (BGT)
prdtot_bgt - Specified Year     All Periods       Budget (BGT)
```

Fund Type Reports (OBJFD)

This type provides column information about ranges of funds. All amounts are taken from the General Ledger Amount record (glamt_rec) and are used strictly in accounting reports. The actual amounts stored are based on four parameters set up in the file $CARSPATH/macros/custom/finrpt.

The macros are as follows:

```
FUND_COL_VAR_PREVYR_ACT - Previous Year Actual (ACT)    PF?_ACT
FUND_COL_VAR_PREVYR_BGT - Previous Year Budget (BGT)    PF?_BGT
FUND_COL_VAR_CURYR_ACT  - Current Year Actual (ACT)     F?_ACT
FUND_COL_VAR_CURYR_BGT  - Current Year Budget (BGT)     F?_BGT
```

The question mark (?) stands for any of the different fund ranges 1 to 7, as well as OTH for Other and TOT for Total. Thus, for the first example, the amount names are PF1_ACT, PF2_ACT, PF7_ACT, PFOTH_ACT and PFTOT_ACT.

These variables determine the amounts stored in the amount fields. Temporary amounts calculated in the report can be then associated with any of the 4 different groups of amounts.

The temporary amounts are:

```
tot           Total Year Amount
ytd           Year-to-date amount up to and including ending period
prd           Monthly amount (of ending period)
prvbal        Year-to-date up to but not including beginning period
ytd - prvbal  Period range amount (from begin period to end period)
```

Note: For each report using these amounts, two menu option parameters designate a beginning and ending period. The first parameter determines the amount *prvbal*. The ending period determines all the other amounts.

Each of the four groups of amounts can be associated with any of the four temporary amounts. For example, if you want the previous year's Actual amounts to have year-to-date values, but you want the previous year's Budget amounts to be total year values, define the macro FUND_COL_VAR_PREVYR_ACT to be *ytd* and define the macro FUND_COL_VAR_PREVYR_BGT to be *tot*.

The *prd* and *ytd* amount types assume monthly budgeting. If your institution does not use monthly budgets, and you want them to be prorated, the syntax for *prd* is *tot/12* and the syntax for *ytd* is *tot/12 * prdno* where *prdno* is a known variable signifying the number of periods up through the ending month.

Use the following amounts with the OBJFD type:

```
F1_ACT     - Year Specified    1st Fund Range     Actual (ACT)
F2_ACT     - Year Specified    2nd Fund Range     Actual (ACT)
F3_ACT     - Year Specified    3rd Fund Range     Actual (ACT)
F4_ACT     - Year Specified    4th Fund Range     Actual (ACT)
F5_ACT     - Year Specified    5th Fund Range     Actual (ACT)
F6_ACT     - Year Specified    6th Fund Range     Actual (ACT)
F7_ACT     - Year Specified    7th Fund Range     Actual (ACT)
FOTH_ACT   - Year Specified    Other Fund Range   Actual (ACT)
FTOT_ACT   - Year Specified    Total All Funds    Actual (ACT)
F1_BGT     - Year Specified    1st Fund Range     Budget (BGT)
F2_BGT     - Year Specified    2nd Fund Range     Budget (BGT)
F3_BGT     - Year Specified    3rd Fund Range     Budget (BGT)
F4_BGT     - Year Specified    4th Fund Range     Budget (BGT)
F5_BGT     - Year Specified    5th Fund Range     Budget (BGT)
F6_BGT     - Year Specified    6th Fund Range     Budget (BGT)
F7_BGT     - Year Specified    7th Fund Range     Budget (BGT)
FOTH_BGT   - Year Specified    Other Fund Range   Budget (BGT)
FTOT_BGT   - Year Specified    Total All Funds    Budget (BGT)
PF1_ACT    - Previous Year     1st Fund Range     Actual (ACT)
PF2_ACT    - Previous Year     2nd Fund Range     Actual (ACT)
PF3_ACT    - Previous Year     3rd Fund Range     Actual (ACT)
PF4_ACT    - Previous Year     4th Fund Range     Actual (ACT)
PF5_ACT    - Previous Year     5th Fund Range     Actual (ACT)
PF6_ACT    - Previous Year     6th Fund Range     Actual (ACT)
PF7_ACT    - Previous Year     7th Fund Range     Actual (ACT)
PFOTH_ACT  - Previous Year     Other Fund Range   Actual (ACT)
PFTOT_ACT  - Previous Year     Total All Funds    Actual (ACT)
PF1_BGT    - Previous Year     1st Fund Range     Budget (BGT)
PF2_BGT    - Previous Year     2nd Fund Range     Budget (BGT)
PF3_BGT    - Previous Year     3rd Fund Range     Budget (BGT)
PF4_BGT    - Previous Year     4th Fund Range     Budget (BGT)
PF5_BGT    - Previous Year     5th Fund Range     Budget (BGT)
PF6_BGT    - Previous Year     6th Fund Range     Budget (BGT)
PF7_BGT    - Previous Year     7th Fund Range     Budget (BGT)
PFOTH_BGT  - Previous Year     Other Fund Range   Budget (BGT)
PFTOT_BGT  - Previous Year     Total All Funds    Budget (BGT)
```

Budget Type Reports (BGT)

This type provides information from budget amount records (bgtamt_rec) and are used strictly in budget reports. Column amounts are for a full year only.

Standard Column Outputs

The following is a sample list of the standard column outputs. There is no limit to the number of column outputs allowed; however, adding too many column outputs can overflow the report buffers.

(ACCT)	(ACCTFD)	(ACCTPRD)	(BGT)
STANDARD	7_COL	MONTH	4COL
FULL	6_OTH	QTRACT	SINGLE
PREV	CURFDPREV	QTRBGT	BGTREQ
PREVCENTS	CURFDBGT		COMPARE
PREVACT			
PREVACTPRD			
PREVYTD			
RANGE			
CURYTD			
YTDNARROW			
BGTCOMP			
TRIALBAL			

Size Restrictions

Although there is no limitation as to the number of column outputs allowed or the number of columns available within a column output, a size limitation exists within ACE that can limit some types of column outputs. Some of the more complicated reports, such as the Cost Center Summary, are close to the limit of ACE compile space available. If a complex column output were added, ACE could run out of compile space. Therefore, avoid using complicated percentages and amount differences. Also, be advised that some reports are more complicated than others; if you succeed in using a complicated column output on one report, there is no assurance that the same column output will succeed on other reports with a different degree of complexity.

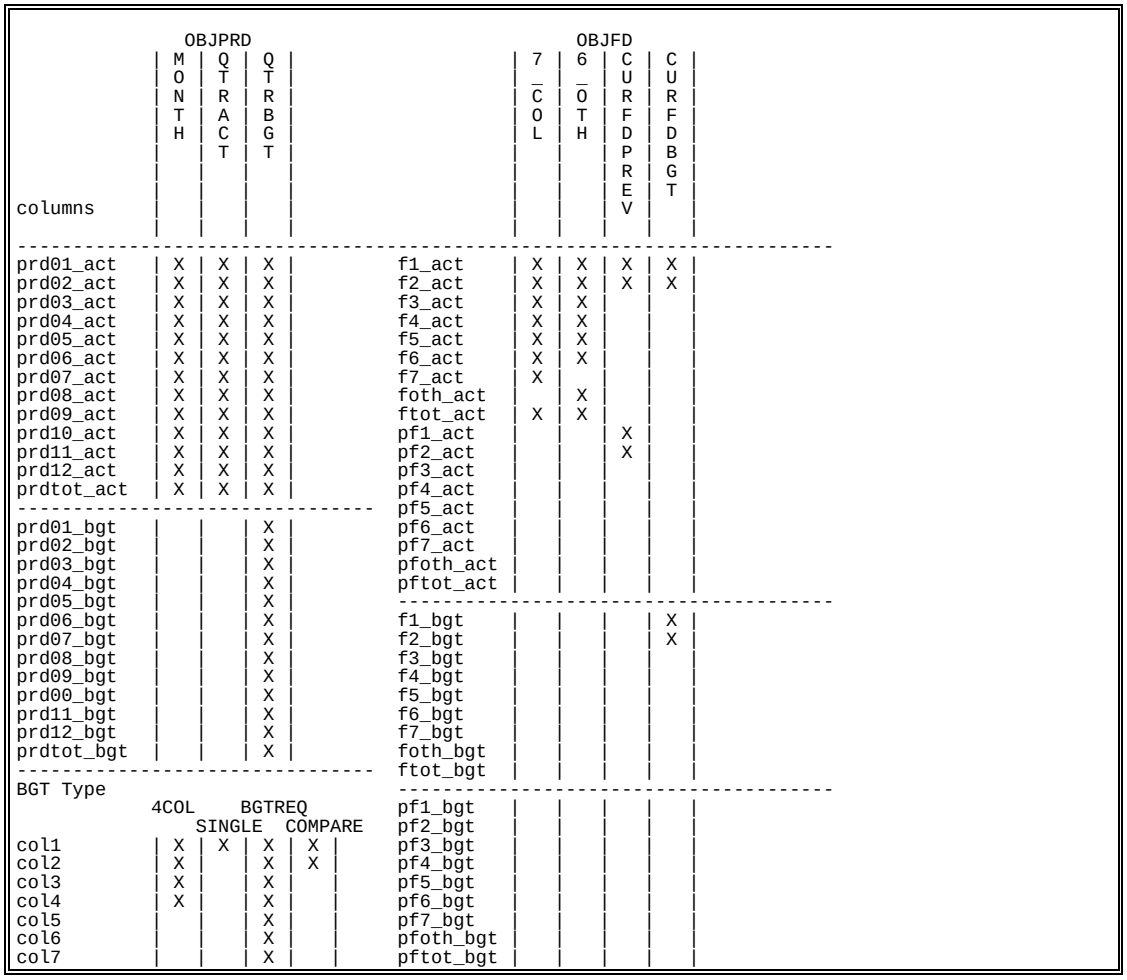
Variables Used in Column Outputs

Standard CX contains the following column outputs using the variables designated with an “X”:

OBJ Type Column Output

columns	STAND ARD	FULL	PREV	PREVCENTS	PREVACT	PREVACTPRD	PREVYTD	RANGE	CURYTD	YTDNARROW	BGTCOMP	TRIALBAL				
actytd	X	X	X	X	X	X	X	X	X	X	X	X				
actprd	X	X				X			X							
actprvbal												X				
(actytd - actprvbal)								X			X	X				
actpytd						X	X									
actpprd						X										
actpprvbal																
actptot			X	X	X											
(actpytd - actpprvbal)								X								
bgtytd		X	X	X			X	X	X	X	X					
bgtprd		X							X							
bgtprvbal																
bgtfy	X	X														
(bgtytd - bgtprvbal)								X								
bgtpytd							X									
bgtpprd																
bgtpprvbal																
bgtptot			X	X												
(bgtpytd - bgtpprvbal)								X								
encytd	X	X	X	X				X	X	X						
encprd		X							X							
encprvbal																
encpytd																
encpprd																
encpprvbal																
encptot																
ytddiff		X	X	X				X	X	X						
prddiff		X							X							
totdiff	X	X														
pytddiff																
pprddiff																
ptotdiff																

Standard CX contains the following variables for OBJPRD, OBJFD, and BGT column output.



	OBJPRD				OBJFD			
columns	MONTH	QTRACT	QTRBGT		7_COL	6_OTH	CURFDPREV	CURFDBGT
prd01_act	X	X	X	f1_act	X	X	X	X
prd02_act	X	X	X	f2_act	X	X	X	X
prd03_act	X	X	X	f3_act	X	X		
prd04_act	X	X	X	f4_act	X	X		
prd05_act	X	X	X	f5_act	X	X		
prd06_act	X	X	X	f6_act	X	X		
prd07_act	X	X	X	f7_act	X			
prd08_act	X	X	X	foth_act		X		
prd09_act	X	X	X	ftot_act	X	X		
prd10_act	X	X	X	pf1_act			X	
prd11_act	X	X	X	pf2_act			X	
prd12_act	X	X	X	pf3_act				
prdtot_act	X	X	X	pf4_act				
				pf5_act				
prd01_bgt			X	pf6_act				
prd02_bgt			X	pf7_act				
prd03_bgt			X	pfoth_act				
prd04_bgt			X	pftot_act				
prd05_bgt			X					
prd06_bgt			X	f1_bgt				X
prd07_bgt			X	f2_bgt				X
prd08_bgt			X	f3_bgt				
prd09_bgt			X	f4_bgt				
prd00_bgt			X	f5_bgt				
prd11_bgt			X	f6_bgt				
prd12_bgt			X	f7_bgt				
prdtot_bgt			X	foth_bgt				
				ftot_bgt				

BGT Type	4COL	BGTREQ		
		SINGLE	COMPARE	
col1	X	X	X	X
col2	X		X	X
col3	X		X	
col4	X		X	
col5			X	
col6			X	
col7			X	

pf1_bgt	
pf2_bgt	
pf3_bgt	
pf4_bgt	
pf5_bgt	
pf6_bgt	
pf7_bgt	
pfoth_bgt	
pftot_bgt	

Notes to Amount Codes

Consider the following when using the amount codes:

- *prd01* includes the BAL period and *prd12* includes the ADJ Period.
- *QTRACT* and *QTRBGT* columns are 3 variables added together (e.g., *prd01 + prd02 + prd03*).
- *COMPARE* also has a percentage and a difference amount as columns.

Resolving Reporting Issues

Introduction

When using reports or modifying report macros, issues or errors may occur. This section contains information to help you resolve the issues.

Omitted Groups of Accounts

Occasionally, some groups of accounts do not display on reports, even though the accounts existed with balances for the fiscal year specified.

To resolve this issue, verify that the accounts are in the temporary account record, glatemp_rec. Use the menu option Reports: Review Report Indexes to query and check the records. If the glatemp_recs are complete and correct, then use *acquery* to verify that General Ledger Amount records exist for the fiscal year(s) specified.

Another explanation for missing accounts on reports is an incorrect Financial Statement table (fs_table). The fs_table contains the breaks for the account and center axes, including block, group and schedule ranges. Accounting reports read the fs_table and select the associated records for each account. The reports read in different axes, depending on their purpose. For example, the *acctsum* report reads the OBJ axis, but *cntrsum* reads the FUNC axis. The *revexp* report reads in both axes, but for different ranges. For the revenue accounts, the reports use the OBJ axis, since they are sorted by object, but for the expenditure accounts, the reports use the FUNC axis, since they are sorted by function.

If the fs_table is not built correctly, reports can produce incomplete or incorrect results. Each function and object must be included in one and only one range (i.e., ranges must include all codes, but must not overlap). For more information about setting up the fs_table, see *Installing the General Ledger Processes* in this manual.

If the schedule record (tfs_grp_code = S) does not exist for an object or function (depending on the axis), the report will access the General Ledger Amount record, but the totals will display as zero. Each report that uses this logic accumulates amounts when it locates the schedule record.

Inconsistency in Number of Defined Parameters

Parameter problems can generate the following message:

> "The number of defined parameters in the ACE program does not equal the number of actual parameters passed to ACEGO on the command line. See error number 9051."

This type of error results when the menu option passes the wrong number of parameters to the ACE report. If the user is running a FUND type report, (e.g., the column output type is 7_COL, 6OTH, CURFDPREV, or CURFDBGT), the macro called OPT_FUND_PARAM may be incorrectly defined.

Debugging Report Macros

CX provides a script that expands the accounting report macros. You can use the script, *acctexp*, to assist you in identifying the errors, or help you see the way the macros interact. It will expand the report that you specify, and put it in a file in your home directory with the report name and an ".err" extension.

Parameter Syntax for acctexp

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

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

acctexp -f report [-s <prim|sec>] [-c column] [-o output] [-d dir]

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

Parameters for acctexp

The following lists the parameters for running *acctexp*.

-f report
Required - The name of report to expand.

-s <prim|sec>
Optional - Either *prim* or *sec*, to indicate that you want the output to sort by primary or secondary responsible person.

-c column
Optional - The column output type, (e.g., YTDNARROW)

-o output
Optional - The output filename, if different from the report name.

-d dir
Optional - The directory location of report. Valid values are *accounting* (the default) or *budget*.

You can use the script for any report that uses column output logic, including the budget reports. Budget reports, however, are in a separate directory. To expand one of them, such as *bgtacctdtl*, you must include the parameter **-d budget** when running *acctexp*.

When you run the script, you receive an error message from *runreports* in your mail, since that is the script that does the final expand. You also receive a “grammatical error” message at the end of the expanded report because the final *end* statement of the report is removed from the temporary file used before expanding it. If you want to compile the report (using *aceprep*), edit the output file (with the .err extension), remove the error lines at the bottom of the file, and add an "end" on the last line. Before you compile the report, change the *.err* extension to *.ace*.

Complexity in Accounting Reports

The accounting reports are internally complex. This complexity limits the ability to expand their contents. For example, some reports do not contain every level of totaling because of internal limitations of the compiler.

If you create a complex report, you can create an error condition resulting from *saceprep* running out of logic room. The error message for this type of error is as follows:

> "The ACE report specification is too complex or large to be properly compiled. The number of instructions needed to implement this specification is greater than the space allocated in the compiler's instruction tables. See error number -8002."

Test your changes to macros and column output on the most complex report used at your institution, since it will provide the most complete test of both your changes and the ACE space constraints. One of the most complex reports is *cntrdtlsrt*, which is the first report run by the *sortcntr* script, so Jenzabar suggests you use it for testing. This report becomes even more complex if the user requests the output to sort by responsible person.

Insufficient User Variable Space

Another space constraint can occur when using ACE. This type of limitation appears when the number of strings (which includes variables defined and literals) exceeds the limit set by ACE. Defining character types uses as much space as their length. The error message for this type of error is as follows:

"The string '(user variable space)' does not fit into remaining space in the compiler's string table. See error number 8054."

Note: Occasionally, ACE is unable to process a complex report, even if the report has compiled successfully.

Variations in Output

If your printed output does not contain the expected rows, columns or totals, any of the following conditions may be the cause:

- Running the report for fiscal years different than those in the glatemp_rec file
- Missing accounts or incorrect dollar amounts
- Incorrectly modified macros

Different Fiscal Years

To determine if the error condition is because the glatemp_rec does not include information for the correct years, review the report indexes that the report is using. To perform the query, examine the first and last fiscal years in the record.

Missing Accounts or Incorrect Dollar Amounts

If your reports have missing accounts or incorrect dollar amounts, perform the following steps:

1. Run the Initialize Report Indexes option, then rerun the report.
2. If you still have missing or incorrect amounts, perform a query on the specific account combination that is missing.
3. If you locate the specific account combination, make sure the amount type is correct. Separate glatemp_recs exist for each amount type (i.e., ACT, BGT, and ENC).
4. If you do not locate the specific account combination, use *acquery* to make sure a gla_rec exists for the combination.
5. If the gla_rec exists, then you may have incorrectly changed a macro in the macros/user/finrpt file. This can happen, for example, if you change macros that affect the array positions of the amounts.

Report Already in Use

When you run some of the reports located in the directory called *others*, the source file is copied to /tmp, run through the m4 processor, then compiled. If you start another account detail report running before a previously started report is finished, then the script that runs the report checks /tmp to determine if there is an existing source file. If a source file exists, you will receive the message,

"The filename report is already being run. Try later."

This message will also appear if the ACE ended abnormally and the script did not clean out the temporary files

Users can run more than one account detail report at the same time by using the optional filename parameter to send the report to a user-defined output filename. The source file in /tmp then becomes the optional filename that you specified.

Note: The *cntrdtlcom*, *sortcntr*, and *sortproj* reports are exceptions. You can run only one of these reports at a time, because they each use scripts that save temporary processing results in files with duplicate names.

If you have not given an optional filename to a report that is currently running, or if you give a report the same optional filename as the other report that is running, the previous message will appear. You can check to determine if this has happened by performing the following:

1. From the shell prompt, go to /tmp.
2. From within /tmp, use the ll command on the filename.
3. Look at the date and time of the file to determine whether it is currently running.

   Note: If the file is running, there are files that begin with "ac" associated with it.

4. Use the ls -lt ac* command to display files with the beginning "ac".
5. Use the rm command to remove all the filenames.

   CAUTION: Use the rm command only if you are sure the report is not running.

6. Restart the report.

SECTION 5 - GENERAL LEDGER MAINTENANCE PROCEDURES

Overview

Introduction

This section provides procedures for performing maintenance within the General Ledger module. The primary reasons to perform these maintenance steps are to validate the new year's posting periods, and to display the correct dates as defaults or options on reports or screens.

The Process

The following list shows the general phases that occur when you add and update general ledger information:

1. Review and update the macros/custom/periodic file to define the correct dates. For more information about the macros that affect the module, see *General Ledger Macros* in this guide.
2. Copy Fiscal Calendar records based on the previous fiscal year.

   Note: Use the following guidelines for copying a previous Fiscal Calendar record:
   - If no data exists for the new fiscal year, use the Add Fiscal Calendar process, which copies records from a previous year.
   - If some data already exists for the new fiscal year, use the Fiscal Calendar option on the Table Maintenance: Financial menu to add remaining records for the new year, or update existing records.
3. Copy gla_recs from the previous year, using the Add G/L Accounts option on the Accounting: General Ledger Closing menu.

How to Add Fiscal Calendar Records

The following list contains the steps for using the *fiscalnxt* script to copy the Fiscal Calendar records from a previous fiscal year.

Note: Remember the following points as you use the *fiscalnxt* script.
- The script may fail if records already exist for the new fiscal year (e.g., if duplicate records exist).
- The *fiscalnxt* script does not overwrite existing Fiscal Calendar records.
- You can exclude up to three specific subsidiaries from carrying over to the new fiscal year.

1. Select Add Fiscal Calendar from the Table Maintenance: Financial menu.
2. Enter the following information:
   - The previous year from which you want to copy
   - The new fiscal year for which you want to create records
   - The subsidiaries, if any, that you want to exclude from the copying process

     Note: Any subsidiary that uses the session/year combination (e.g., FA96) must be excluded.
3. Select **Finish** to display the Output Parameters and Scheduling window.
4. Enter the time and day that you want to run the process, or enter **NOW** to run the process immediately.
5. Indicate whether or not you want to run the process in the background.

6. Select **Finish**. The program schedules or initiates the Add Fiscal Calendar process.

How to Add General Ledger Account Records

The following list contains the steps for using the *addgla* script to copy the General Ledger Account records from a previous fiscal year.

Note: Remember the following points as you use the *addgla* script.

- The script may fail if records already exist for the new fiscal year (e.g., if duplicate records exist).
- The *addgla* script does not overwrite existing General Ledger Account records.

1. Select Add G/L Accounts from the Accounting: Period End Processing: General Ledger Closing menu.
2. Enter the following information:
   - The previous year from which you want to copy
   - The new fiscal year for which you want to create records
3. Select **Finish** to display the Output Parameters and Scheduling window.
4. Enter the time and day that you want to run the process, or enter **NOW** to run the process immediately.
5. Indicate whether or not you want to run the process in the background.
6. Select **Finish**. The program schedules or initiates the Add G/L Accounts process.

Maintaining Claims on Cash, Receivables and Payables

Introduction

The claim on cash feature is designed to be set up during implementation and to produce claim and contra claim entries without user intervention. If for any reason your institution modifies the Claim table that controls the feature, or if you implement the claim on cash feature after your initial CX implementation, you may have unwanted inter-fund or inter-subfund transactions in your due to/from accounts. This section provides instructions for resolving inconsistencies or errors in your accounts that result from claim on cash processing.

Procedure

Use the following procedure to resolve inconsistencies or errors in your accounts that result from incorrect claim on cash processing.

1. Update the Claim table, ensuring that all of the institution's cash, student accounts receivable, and payables have table entries.
2. If any of the errors resulted from student accounts, run the audit deferral (*defaudit*) process in Fee Collection. This process reconciles errors and reconstructs entries that originated in student accounts. The reconstructed entries then automatically generate correcting claim on cash entries. For more information about *defaudit*, see *Using Fee Collection*.
3. Run *glaudit* in report mode, then verify the results. The *glaudit* program locates all inconsistencies and verifies that amounts in claim on cash entries and their respective contra accounts match. For more information about *glaudit*, see *General Ledger Audit* in this manual.
4. After verifying that *glaudit* has identified all the errors, run the program again, using the update parameter.

Running Subsidiary Account Balance Forward

The Subsidiary Balance Forward process creates updated subb_recs for the current period, which are needed if your institution is using the Automated Holds feature. To ensure that Automated Holds works correctly, you must always run the Subsidiary Balance Forward option before the Automated Holds - Subs option. For more information, see the *Student Billing - Automated Holds Script* section in *Systems Manual: Student.*

SECTION 6 - CRASH RECOVERY

Overview

Introduction

This section provides a procedure to recover from a crash in a General Ledger program.

Note: Refer to *Using General Ledger* for a list of the more common status, field error, and warning messages that can occur when menu users execute the programs in General Ledger.

Fatal Errors

Fatal errors can occur in the following programs in General Ledger:

- *acquery*
- *bgtreview*
- *cashier*
- *voucher*

Crash Recovery

Introduction

When a user is accessing an accounting program and a system failure occurs, a core dump may result. This section contains the procedure for recovering from a core dump, including the additional recovery steps required if the *voucher* program was in use at the time of the failure.

Core Dump Recovery

The following list describes the steps to recover from a core dump of a program.

1. Access the program screens directory for the program.

   Example: % cd $CARSPATH/modules/accounting/progscr/*program*

2. Reinstall each program screen file.

   Example: % make reinstall F=*filename*

   Note: You can also reinstall all of the screens by entering the following:
   % make reinstall F=ALL

3. 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 4.

4. Access the source code directory of the program.

   Example: % cd $CARSPATH/src/accounting/*program*

5. Reinstall the source code for the program.

   Example: % make reinstall

6. 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 7.

7. In the source code for the program, delete the old compiled code for the entry program.

   Example: % make cleanup

8. Reinstall the program source code.

   Example: % make reinstall

9. 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 10.

10. 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

11. Reinstall the libraries for the program and reinstall the source for the program.

    Example: % cd *appropriate library*

% make reinstall

% cd $CARSPATH/src/accounting/*program*

% make reinstall

Note: You must reinstall the source program to include any library changes.

12. Attempt to execute the program. Did the reinstallation of the libraries for the program fix the error?
    - If yes, go to step 13.
    - If no, call Jenzabar Support Services.
13. Did the crash occur while a journal was open?
    - If yes, go to step 14.
    - If no, you are done.
14. Run *vchrecover*.
15. Run *glaudit* with the update option. If the crash occurred while *cashier* was in use, run *glaudit* with the Cash Balance flag set to Y.
16. Did the crash occur while attempting to post to a subsidiary account?
    - If yes, go to step 17.
    - If no, you are done.
17. Run *saaudit* with the update option.

INDEX

D

E

F

G

H

I

J

L

M

S

T

V

Y