Jenzabar CX

Course/Class Schedule

Technical Manual

Filename: tmcrssch
Distribution date: 02/15/2002

Contact us at www.jenzabar.com

JENZABAR, INC.
COURSE/CLASS SCHEDULE TECHNICAL MANUAL

TABLE OF CONTENTS

SECTION 1 – USING THIS MANUAL

Overview

Purpose of This Manual

This manual provides technical information required to install, customize, and maintain the Course/Class Schedule product of CX.

Intended Audience

This manual is for use by those individuals responsible for installing, customizing, and maintaining CX.

How to Use This Manual

If you are not familiar with the processes and features of the Course/Class Schedule product, read the manual for:

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

If you are familiar with the processes and features of the Course/Class Schedule product, and just need specific reference information or a procedure, look through the table of contents or index and refer to the pages you need.

Product Differences

This manual contains information for using all features developed for the Course/Class Schedule product. Your institution may or may not have all the features documented in this manual.

Structure of This Manual

This manual contains both general reference information and procedures for installing, customizing, and maintaining the Course/Class Schedule product. The organization of the manual is as follows:

Overview information

Section 1 - Information about using this manual
Section 2 - Overview information about the product

Product reference information

Section 3 - Tables and records used in the product
Section 4 - Macros and includes
Section 5 - CX program files
Section 6 - Cancel Sections program (*cnclsec*)
Section 7 - Course Entry program (*crsent*)
Section 8 - History Equivalency program (*histeq*)
Section 9 - Roll Session program (*rollsess*)
Section 10 - Set Reference Number program (*setrefno*)
Section 11 - Menus, screens, scripts, and reports

Product procedures

Section 12 - Procedures to install and customize your processes
Section 13 - Procedures to maintain the product

Error reference/Recovery procedures

Section 14 - A reference of fatal errors and recovery procedures

Reference information

Index

Related Documents and Help

The following resources also are available to assist you in installing, customizing, maintaining, and using the Course/Class Schedule product.

QuickMate online help

QuickMate Installation Guide
Getting Started User Guide

Technical manuals

CX Implementation and Maintenance Technical Guide
CX System Reference Technical Guide

Terminology

Master Glossary

UNIX-based help

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

User guides

Course/Class Schedule User Guide
Getting Started User Guide

Conventions Used in This Manual

Introduction

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

Style Conventions

CX technical manuals observe the following style conventions.

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

Bulleted lists
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)

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

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

Numbered lists
Show ranking of items or sequence of performance.

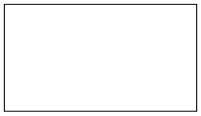
Parentheses
When used around a field name, indicate the field is unlabeled. The field description includes the location of the field.

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

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

Flowchart Conventions

Flowcharts representing a general overview of a particular application or process are included in this manual. Symbols are used in flowcharts as follows:



Process

Represents a processing function the system performs or the user must do.

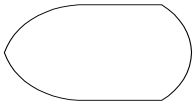
Predefined Process

Represents a subroutine or module that can be called by an application; i.e., generally programming code.

Note: Used only for exception situations. The Process symbol is the standard symbol.

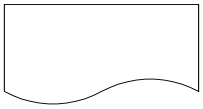
Terminator

Represents the very first or last symbol in a flowchart. Also used for references to another manual.



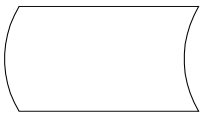
Display

Represents data displayed on a screen or window. Used for all menus including the master menu.



Document

Represents an output report, forms, or any readable data, either hardcopy or displayed on a screen.

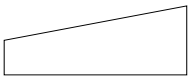


Stored data

Represents data maintained in the database including tables and records.

Manual operation

Represents any process performed manually by the user.



Manual input

Represents data entered by any manual method, such as data entry, scanning, or reading bar-codes.

Connector

Connects one section of the same flowchart to another. Usually, a connector contains a number where it exits a flowchart, and the same number in the connector showing re-entry.

Jenzabar -Specific Terms

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

Application

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

Data

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

Enter

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

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

F key

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

Hot key

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

ID

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

Parameter

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

Select

To execute a command by any of the following actions:

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

System

The Jenzabar, Inc. 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>**).

SECTION 2 – COURSE/CLASS SCHEDULE PROCESSES

Overview

Introduction

This section provides information on the purpose and process flow of Course/Class Schedule.

Purpose of Product

The primary purpose of Course/Class Schedule is to enable you to add, update, and maintain information about your institution's catalogs, classrooms, facilities, instructors, and meeting schedules.

Background Knowledge

The following list describes the necessary background information that you should know to implement and support the Course/Class Schedule product.

UNIX

Know the following about the UNIX operating system:

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

INFORMIX-SQL

Know about the following INFORMIX tools:

- SQL database
- PERFORM screens
- ACE reports

Jenzabar CX database tools and utilities

Know how to use the following database tools:

- MAKE processor
- Schemas
- Macros
- Includes
- Program screens and windows

Jenzabar CX

Know the following about the CX standard product:

- CX directory structure
- The menu processor
- The CX database engine
- The product update process

QuickMate features

Know the following about the CX Graphical Server:

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

C Programming

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

Course/Class Schedule policies and procedures

Know answers to the following questions:

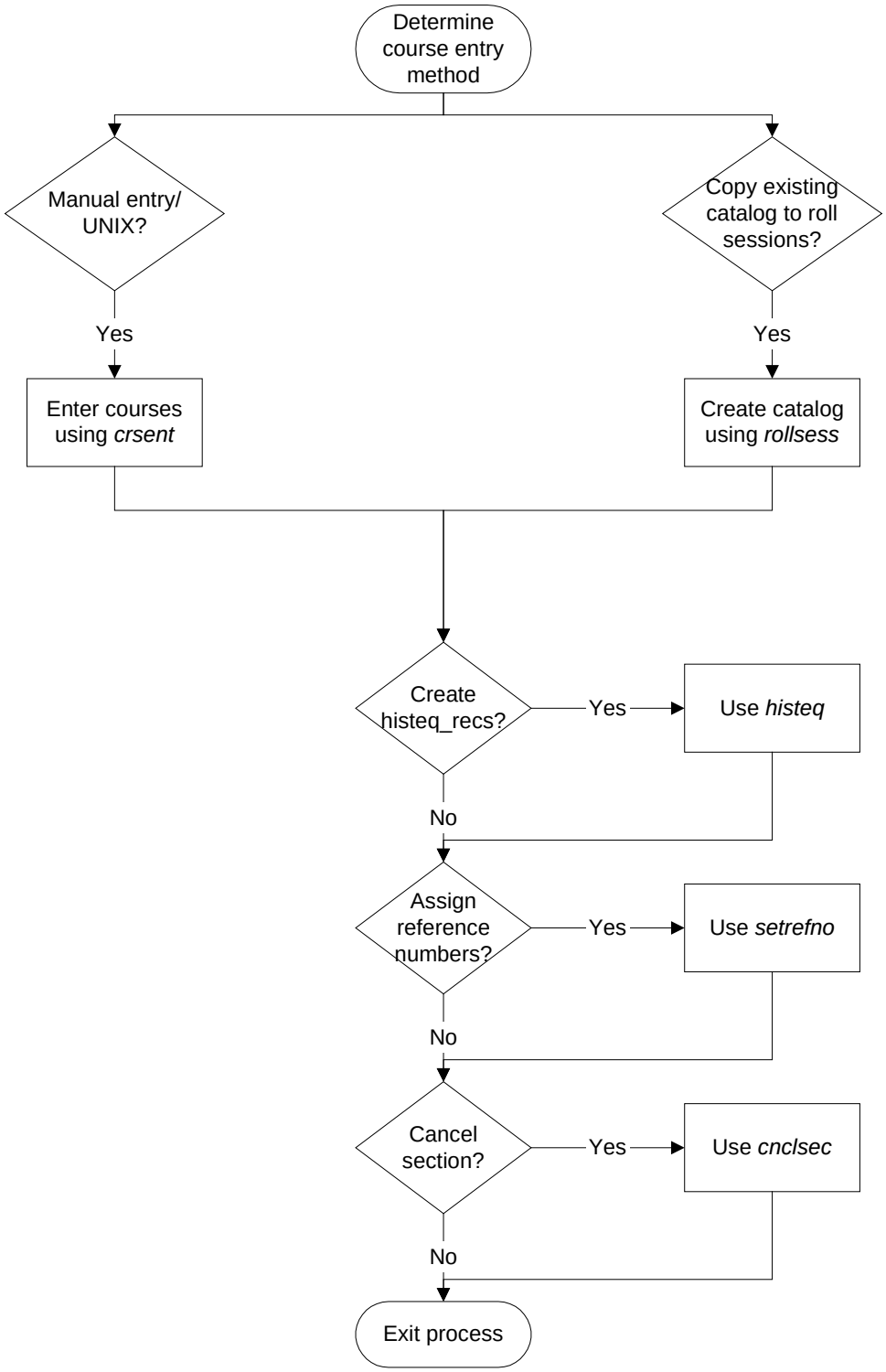
1. Who is authorized to create a course?
2. Who is authorized to cancel a section?

Process Flow

Diagram

The following diagram shows the process flow of the Course/Class Schedule product.

Note: For more information about program interrelationships and detailed data flow diagrams, see the program sections in this manual.



Process Description

The following list describes the processes involved with the Course/Class Schedule product.

1. Determine your method of entering course and class information.
   - Manual entry using UNIX

- Copy an existing catalog to roll over sessions

2. If you are performing manual course entry using UNIX, enter courses using *crsent*.
3. If you are copying an existing catalog to roll over sessions, create a new catalog using *rollsess*.
4. If you are creating histeq_recs, use *histeq*.
5. If you are assigning reference numbers to courses, use *setrefno*.
6. If you are canceling a section, use *cnclsec*.
7. Exit the Course/Class Schedule process.

Application Relationships

Related Jenzabar CX Applications

The Course/Class Schedule product interacts with several other applications and products in CX. The following list describes the interrelationships.

Registrar

Users of the Registrar application can change fees that are assessed from Course/Class Schedule data.

SECTION 3 – COURSE/CLASS SCHEDULE TABLES AND RECORDS

Overview

Introduction

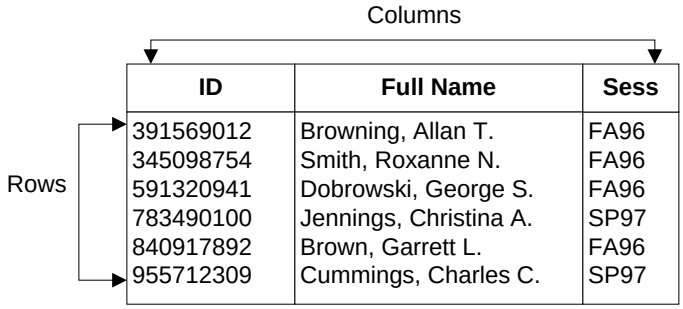
This section provides reference information about each table and record associated with the Course/Class Schedule product. It also provides definitions of SQL and CX table and record terminology and differentiates among common, shared, and product-specific tables and records.

Alphabetical Organization

The tables and records appear in alphabetical order in this section.

What Is an SQL Table?

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



ID	Full Name	Sess
391569012	Browning, Allan T.	FA96
345098754	Smith, Roxanne N.	FA96
591320941	Dobrowski, George S.	FA96
783490100	Jennings, Christina A.	SP97
840917892	Brown, Garrett L.	FA96
955712309	Cummings, Charles C.	SP97

What Is a Jenzabar CX Table?

A *table* in CX contains information that remains static and is denoted with the *_table* extension. For example, the State table, named *st_table,* contains the list of the states in the United States of America. On the CX menu, you can access most tables in Table Maintenance menus.

What Is a Jenzabar CX Record?

A *record* in CX is a table containing information that changes on a regular basis and is denoted with the *_rec* extension. For example, the Alternate Address record, named *aa_rec*, contains any other addresses where students can be contacted, such as a summer address. You access records in CX program screens and windows, scroll screens, and PERFORM screens. SQL makes no distinction between tables and records; all sets of data are tables.

Summary List of Tables and Records Used

Introduction

Tables and records used in Course/Class Schedule can be divided into the following categories:

- Common
- Shared
- Product-specific

Among these categories, some tables and records are required, while others are optional.

Impact of Changes to Tables and Records

If you make changes to schemas for any tables or records, you must reinstall each associated product or module.

Common Tables and Records

Modules in the Course/Class Schedule product use several tables and records that are common throughout CX. These tables and records are:

Note: For additional information about these common tables and records see the *CX System Reference Technical Manual* unless otherwise noted.

- Building table (bldg_table)
- Contact record (ctc_rec) (See *Communications Management Technical Manual*)
- Faculty record (fac_rec)
- Faculty Activity record (facact_rec)
- Facility table (facil_table)
- Faculty Qualification record (facqual_rec)
- ID record (id_rec)
- Office table (ofc_table)
- Profile record (profile_rec)

CAUTION: Before you begin to modify any common records and tables, check with other departments at the institution to determine whether any other Jenzabar, Inc. implementations have occurred and which common tables were affected by the prior implementation(s). It is very important that you perform this checking so that you do not inadvertently negate the work performed by individuals in other areas of the institution.

Shared Tables and Records

Some tables and records used in Course/Class Schedule originate, or are more frequently used, within other CX product areas. These tables and records, and their originating product areas, are:

Note: For additional information, see the technical manual for the primary product area. If multiple products are shown, the primary product is listed first.

- Academic record (stu_acad_rec)
  - Registration
- Course Work record (cw_rec)
  - Registration
- Noncatalog Equiv record (noncateq_rec)
  - Transcript
- Program Enrollment record (prog_enr_rec)

    - Registration
- Registration record (reg_rec)
    - Registration
    - Student Billing
- Stu Bill Custom record (sbcust_rec)
    - Registration
    - Student Billing
- Student Services record (stu_serv_rec)
    - Student Services
    - Student Billing
    - Registration
    - Financial Aid
- Student Status record (stu_stat_rec)
    - Registration

Course/Class Schedule Tables and Records

Programs in the Course/Class Schedule product use the following Course/Class Schedule tables and records. File information about these tables and records is in this section.

- Academic Calendar record (acad_cal_rec)
- Alternate Academic Calendar record (alt_cal_rec)
- Alternate Calendar table (altcal_table)
- Alternate Refund record (altrfnd_rec)
- Alternate Refund table (altrfnd_table)
- Articulation record (artic_rec)
- Attendance record (ada_rec)
- Attendance Totals record (adatot_rec)
- Course Catalog table (cat_table)
- Counting table (cntg_table)
- Course record (crs_rec)
- Course Catalog Abstract record (crsabstr_rec)
- Course Detail record (crsdtl_rec)
- Course Equivalence record (crseq_rec)
- Course History record (crshist_rec)
- Course Objectives Text record (crsobj_rec)
- Course Requirement Text record (crsrequir_rec)
- Course Syllabus Text record (crssyll_rec)
- Course Teaching Qualifications record (crsqual_rec)
- Department table (dept_table)
- Division table (div_table)
- Faculty Load table (flt_table)
- History Equiv record (histeq_rec)
- Instructional Method record (im_rec)
- Instructional Method table (im_table)
- Instructor record (instr_rec)
- Instructor Activity table (act_table)
- Instructor Qualifications table (qual_table)
- Meeting record (mtg_rec)
- Meeting Detail record (mtgdtl_rec)
- Non-Instructional table (noninstr_table)
- Non-teaching Date table (nonteach_table)
- Nonteaching FTE Assignment record (noninstr_rec)
- Operator Department table (oprdept_table)

- Period table (prd_table)
- Prerequisite record (crsreqgrp_rec)
- Repeat table (rep_table)
- Requirement Field table (secreqfld_table)
- Requirement table (secreq_table)
- Requisite Maintenance record (crsreq_rec)
- Schedule Comment record (schd_comment_rec)
- Section record (sec_rec)
- Section Bridge record (secbrdg_rec)
- Section Detail record (secdtl_rec)
- Section Meeting record (secmtg_rec)
- Section Requirements record (secreq_rec)
- Session table (sess_table)
- Subsession table (subsess_table)
- Textbook record (textbk_rec)

Table and Record Relationships

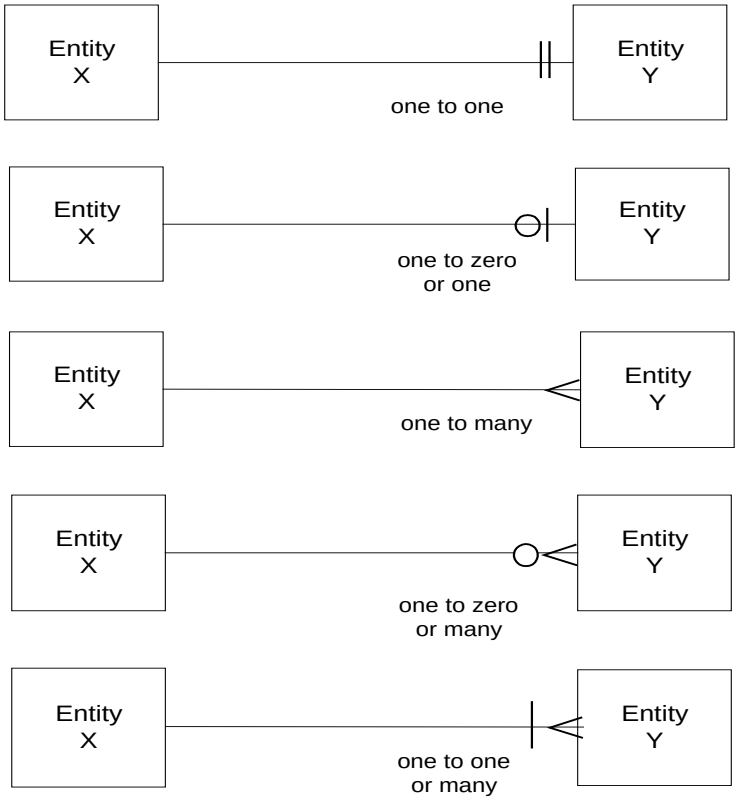
Key to Entity Relationship Diagrams

This diagram provides a key to interpreting the entity relationships in these diagrams.

Interpret the diagrams as follows:

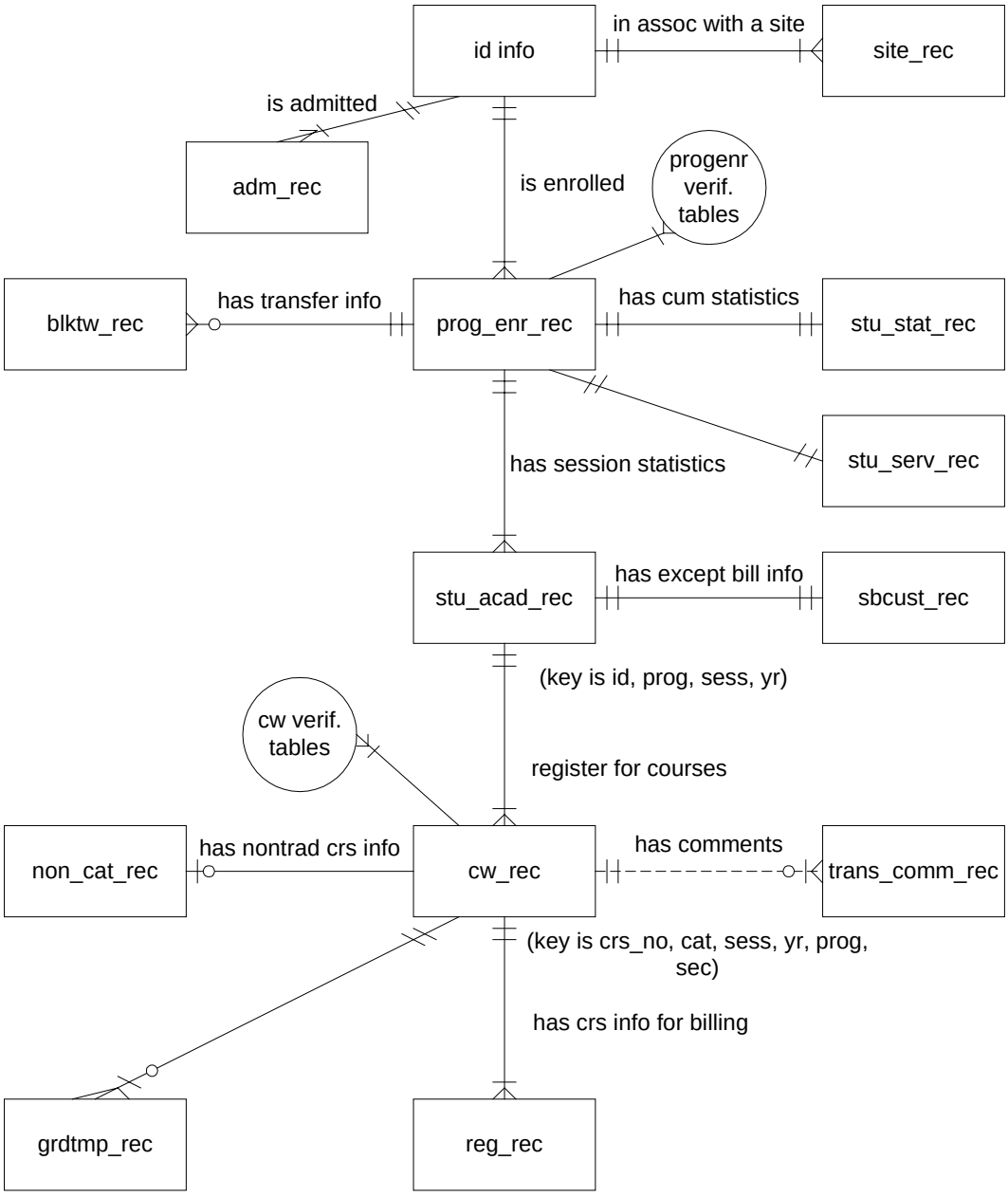
- Entity X has a one to (a) one or many relationship to Entity Y.
- Entity Y has a one to (b) one relationship to Entity X.

Relationship symbols and their meanings:

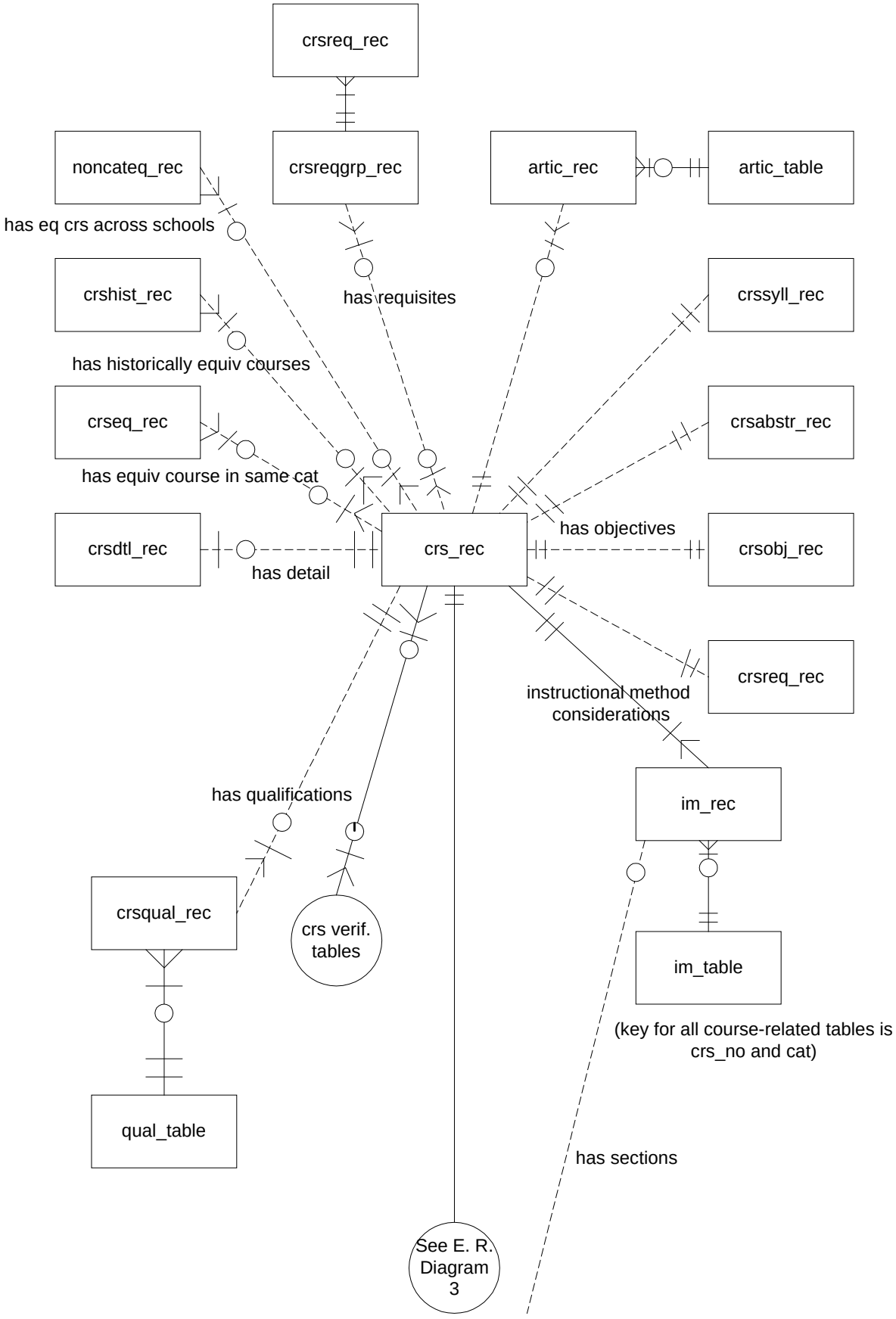


Entity Relationship Diagram 1

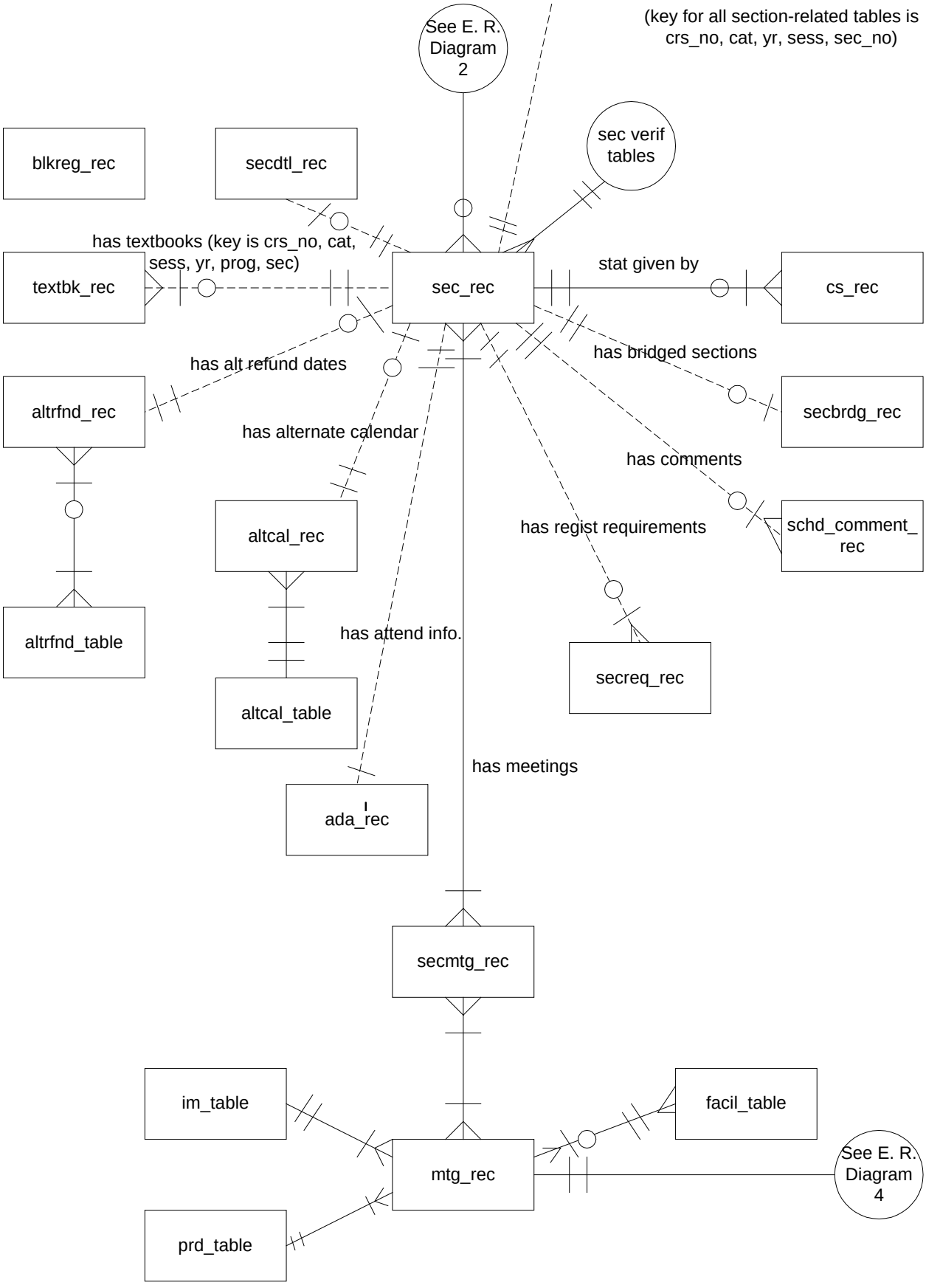
The following diagrams show the relationship between the tables and records the Course/Class Schedule product uses.



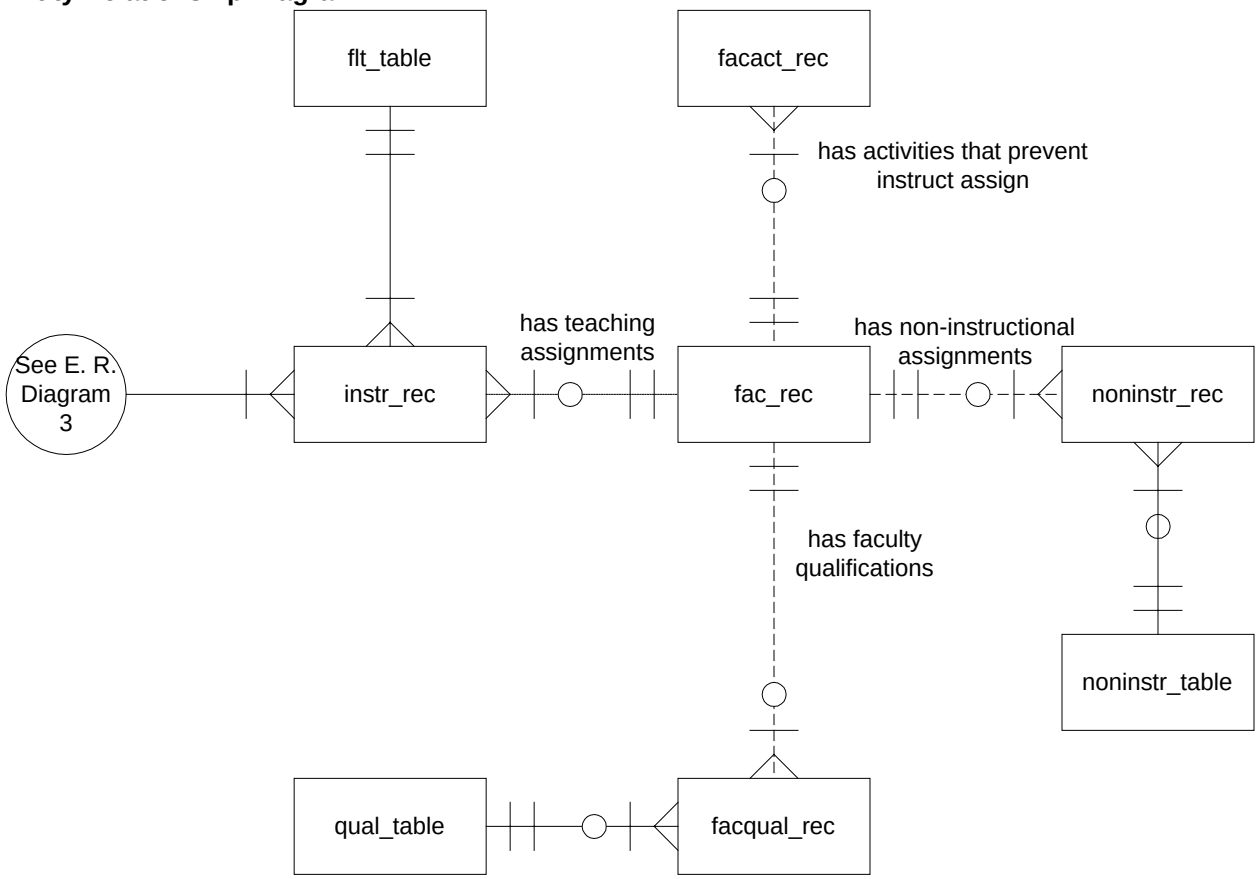
Entity Relationship Diagram 2


crsreq_rec
noncateq_rec
crsreqgrp_rec
artic_rec
artic_table
has eq crs across schools
has requisites
crshist_rec
crssyll_rec
has historically equiv courses
crseq_rec
crsabstr_rec
has equiv course in same cat
crsdtl_rec
crs_rec
has objectives
crsobj_rec
has detail
crsreq_rec
instructional method considerations
has qualifications
im_rec
crsqual_rec
crs verif. tables
im_table
(key for all course-related tables is crs_no and cat)
qual_table
has sections
See E. R. Diagram 3

Entity Relationship Diagram 3


See E. R. Diagram 2
(key for all section-related tables is crs_no, cat, yr, sess, sec_no)
blkreg_rec
secdtl_rec
sec verif tables
has textbooks (key is crs_no, cat, sess, yr, prog, sec)
textbk_rec
sec_rec
stat given by
cs_rec
has alt refund dates
has bridged sections
altrfnd_rec
secbrdg_rec
has alternate calendar
has comments
altcal_rec
schd_comment_rec
has regist requirements
altrfnd_table
has attend info.
secreq_rec
altcal_table
has meetings
ada_rec
secmtg_rec
im_table
facil_table
mtg_rec
See E. R. Diagram 4
prd_table

Entity Relationship Diagram 4



Schemas

Introduction

Schema files define the structure of database files and associated fields in the CX data dictionary. You can access schema files associated with the Course/Class Schedule product in the following directory path: $CARSPATH/schema/student.

File Naming Conventions

CX makes distinctions in the naming of schemas. For schema files containing definitions of CX tables, the UNIX filename begins with the letter *t* followed by characters describing the English name of the table (e.g., *tst* for the State table). For schema files containing definitions of CX records, the UNIX filename describes the English name of the record (e.g., *id* for the ID record).

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

Field Descriptions

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

Schema File Reports

Introduction

The standard CX product includes three reports that provide information about the tables and records used in each product area. When table implementation begins, you can run the reports to provide the installation team with the most current information about the tables and records, and the columns in each.

Report Locations

Access the reports from the System Management: Data Dictionary menu.

Report Descriptions

The following three reports provide information that is useful during the table setup phase of implementation:

Database Fields for the Jenzabar CX Database Dictionary

Report filename: *dbefield*

Menu option: Fields by File Report

Description: This report lists each column in the specified range of files (e.g., id_rec to profile_rec), including its name, short and long descriptions, field type, and size.

Database Files for the Jenzabar CX Database Dictionary

Report filename: *dbefile*

Menu option: Fields by Track Report

Description: This report lists each table that relates to the specified range of product areas (e.g., A for Admissions to F for Financial), including the table name, description, and purpose.

Database Files/Fields for the Jenzabar CX Database Dictionary by Track

Report filename: *dbetrack*

Menu option: File/Field by Track Report

Description: This report combines the contents of *dbefield* and *dbefile*, displaying the tables for the specified product areas (e.g., A for Admissions to F for Financial) and the columns in each table.

Tables and Records

Table and Record Information

The following list identifies the tables and records that originate from the Course/Class Schedule product. The list includes the filenames, location, purpose, and association of each table and record with programs, products, and other tables and records.

Note: The *Program interrelationships* in the list are included in the Course/Class Schedule product. The *Product interrelationships* in the list are not included in the Course/Class Schedule product.

Academic Calendar record

Describes academic sessions of an institution's academic calendar.

UNIX filename: acadcal

Informix filename: acad_cal_rec

Schema location: $CARSPATH/schema/student

Program interrelationships: cnclsec, crsent, rollsess, setrefno

Product interrelationships: Grading, Registration, Transcript

Table/record interrelationships: Associated with the Program table (prog_table) and Session table (sess_table).

Alternate Academic Calendar record

Describes the last add, drop, and withdrawal dates for nonstandard sections. Nonstandard sections are sections that do not use the academic calendar.

UNIX filename: altcal

Informix filename: altcal_rec

Schema location: $CARSPATH/schema/student

Program interrelationships: crsent, regent, rollsess

Product interrelationships: Registration

Table/record interrelationships: Associated with the Section record (sec_rec) and Alternate Calendar table (altcal_table).

Alternate Refund record

Describes refund dates for sections that do not follow the Refund table (rfnd_table) logic.

UNIX filename: altrfnd

Informix filename: altrfnd_rec

Schema location: $CARSPATH/schema/student

Program interrelationships: billing, crsent, regent, rollsess

Product interrelationships: Registration, Student Billing

Table/record interrelationships: Associated with the Section record (sec_rec) and Alternate Refund table (altrfnd_table).

Articulation record

Describes courses for reports to state or federal agencies. Also identifies the various articulation or transfer programs associated with a specific course.

UNIX filename: artic

Informix filename: artic_rec

Schema location: $CARSPATH/schema/student

Program interrelationships: crsent

Product interrelationships: None

Table/record interrelationships: Associated with the Course record (crs_rec) and Articulation table (artic_table).

Articulation table

Retains transfer articulation ties between courses at the institution and outside agencies or other institutions.

UNIX filename: tartic

Informix filename: artic_table

Schema location: $CARSPATH/schema/student

Program interrelationships: crsent

Product interrelationships: None

Table/record interrelationships: Associated with the Course record (crs_rec) and Articulation record (artic_rec).

Attendance record

Maintains average daily attendance for sections.

UNIX filename: ada

Informix filename: ada_rec

Schema location: $CARSPATH/schema/student

Program interrelationships: crsent

Product interrelationships: None

Table/record interrelationships: Associated with the Section record (sec_rec).

Attendance Totals record

Maintains attendance totals for sections.

UNIX filename: adatot

Informix filename: adatot_rec

Schema location: $CARSPATH/schema/student

Program interrelationships: crsent

Product interrelationships: None

Table/record interrelationships: Associated with the Attendance record (adatot_rec).

Course record

Describes a course an institution offers.

UNIX filename: crs

Informix filename: crs_rec

Schema location: $CARSPATH/schema/student

Program interrelationships: crsent, grading, grdrpt, regist, reglist, rollsess, setrefno, trans

Product interrelationships: Degree Audit, Grading, Registration, Transcript

Table/record interrelationships: Associated with the Articulation record (artic_rec), Course Syllabus Text record (crssyll_rec), Course Catalog Abstract record (crsabstr_rec), Course Objectives Text record (crsobj_rec), Requisite Maintenance record (crsreq_rec), Instructional Method record (im_rec), Course Teaching Qualifications record (crsqual_rec), Course Detail record (crsdtl_rec), Course Equivalence record (crseq_rec), Course History record (crshist_rec), Noncatalog Equiv record (noncateq_rec), Prerequisite record (crsreqgrp_rec).

Course Catalog table

Contains records created through an informer for the course catalog.

UNIX filename: tcat

Informix filename: cat_table

Schema location: $CARSPATH/schema/student

Program interrelationships: crsent, regist, trans

Product interrelationships: Degree Audit, Registration, Transcript

Table/record interrelationships: Associated with the Course record (crs_rec).

Course Catalog Abstract record

Describes an outline of a course for publication in the course catalog or in schedules.

UNIX filename: crsabstr

Informix filename: crsabstr_rec

Schema location: $CARSPATH/schema/student

Program interrelationships: crsent

Product interrelationships: None

Table/record interrelationships: Associated with the Course record (crs_rec)

Course Category table

Describes valid course category values. The system uses course categories to determine which courses appear on a transcript.

UNIX filename: tcrsctgry

Informix filename: crsctgry_table

Schema location: $CARSPATH/schema/student

Program interrelationships: crsent, trans

Product interrelationships: Transcript

Table/record interrelationships: Associated with the Course record (crs_rec)

Course Detail record

Describes specific course information to be tracked by the institution. This record is designed to be modified by the site and is not maintained by Jenzabar, Inc.

UNIX filename: crsdtl

Informix filename: crsdtl_rec

Schema location: $CARSPATH/schema/student

Program interrelationships: crsent

Product interrelationships: None

Table/record interrelationships: Associated with the Course record (crs_rec).

Course Equivalence record

Defines equivalent courses within a catalog (generally offered under different departments).

UNIX filename: crseq

Informix filename: crseq_rec

Schema location: $CARSPATH/schema/student

Program interrelationships: crsent

Product interrelationships: Degree Audit, Registration, Transcript

Table/record interrelationships: Associated with the Course record (crs_rec).

Course History record

Defines course number changes between catalogs.

UNIX filename: crshist

Informix filename: crshist_rec

Schema location: $CARSPATH/schema/student

Program interrelationships: crsent

Product interrelationships: Degree Audit, Registration, Transcript

Table/record interrelationships: Associated with the Course record (crs_rec).

Course Objectives Text record

Describes the teaching objectives of the course from both a student and faculty perspective using a text Binary Large Object (BLOB) field.

UNIX filename: crsobj

Informix filename: crsobj_rec

Schema location: $CARSPATH/schema/student

Program interrelationships: crsent

Product interrelationships: None

Table/record interrelationships: Associated with the Course record (crs_rec).

Course Requirement Text record

Provides an outline of the course requirements to include in course catalogs or schedules.

UNIX filename: crsrequir

Informix filename: crsrequir_rec

Schema location: $CARSPATH/schema/student

Program interrelationships: crsent

Product interrelationships: None

Table/record interrelationships: Associated with the Course record (crs_rec).

Course Syllabus Text record

Provides an overview of course lesson plans, instructional materials notations, evaluation procedures, attendance policy, and course education goals as a course syllabus using a text Binary Large Object (BLOB) field.

UNIX filename: crssyll

Informix filename: crssyll_rec

Schema location: $CARSPATH/schema/student

Program interrelationships: crsent

Product interrelationships: None

Table/record interrelationships: Associated with the Course record (crs_rec).

Course Teaching Qualifications record

Describes qualifications for teaching a course. This record does no auto-matching to the Faculty record and is driven by the Course Teaching Qualifications table (qual_table).

UNIX filename: crsqual

Informix filename: crsqual_rec

Schema location: $CARSPATH/schema/student

Program interrelationships: crsent

Product interrelationships: None

Table/record interrelationships: Associated with the Course record (crs_rec).

Discipline table

Describes the academic disciplines that may be associated with a course..

UNIX filename: tdisc

Informix filename: disc_table

Schema location: $CARSPATH/schema/student

Program interrelationships: crsent

Product interrelationships: None

Table/record interrelationships: Associated with the Course record (crs_rec).

Faculty Load table

Describes the faculty load equivalency data needed to compute faculty load values.

UNIX filename: tflt

Informix filename: flt_table

Schema location: $CARSPATH/schema/student

Program interrelationships: crsent

Product interrelationships: None

Table/record interrelationships: Associated with the Instructor record (instr_rec).

History Equiv record

Describes history and equivalencies associated with a given course and catalog prefix. This record is built from the Course Equivalence record (crseq_rec) and Course History record (crshist_rec).

UNIX filename: histeq

Informix filename: histeq_rec

Schema location: $CARSPATH/schema/student

Program interrelationships: histeq

Product interrelationships: Web Registration

Table/record interrelationships: Associated with the Course record (crs_rec).

Instructional Method record

Describes faculty load parameters for each course or section.

UNIX filename: im

Informix filename: im_rec

Schema location: $CARSPATH/schema/student

Program interrelationships: crsent

Product interrelationships: None

Table/record interrelationships: Associated with the Course record (crs_rec).

Instructional Method table

Describes valid instructional methods for courses and instructors.

UNIX filename: tim

Informix filename: im_table

Schema location: $CARSPATH/schema/student

Program interrelationships: crsent

Product interrelationships: None

Table/record interrelationships: Associated with the Instructional Method record (im_rec).

Instructor record

Maintains a tie to the Meeting record (mtg_rec) for instructional purposes and gathers load assessment data.

UNIX filename: instr

Informix filename: instr_rec

Schema location: $CARSPATH/schema/student

Program interrelationships: crsent

Product interrelationships: Registration

Table/record interrelationships: Associated with the Faculty record (fac_rec) and Faculty Load table (flt_table), Meeting record (mtg_rec).

Major table

Describes a school's valid academic major fields of study.

UNIX filename: tmajor

Informix filename: major_table

Schema location: $CARSPATH/schema/student

Program interrelationships: crsent

Product interrelationships: Registration

Table/record interrelationships: Associated with the Course record (crs_rec).

Meeting record

Describes the meeting times of sections of a course.

UNIX filename: mtg

Informix filename: mtg_rec

Schema location: $CARSPATH/schema/student

Program interrelationships: crsent

Product interrelationships: Registration

Table/record interrelationships: Associated with the Facility table (facil_table), Instructor record (instr_rec), Instructional Method table (im_table), Period table (prd_table), and Section Meeting (secmtg_rec).

Meeting Detail record

Contains the exact meeting date for each meeting occurrence.

UNIX filename: mtgdtl

Informix filename: mtgdtl_rec

Schema location: $CARSPATH/schema/student

Program interrelationships: crsent

Product interrelationships: None

Table/record interrelationships: Associated with the Meeting record (mtg_rec).

Nonteaching Date table

Specifies all days on which otherwise scheduled class meetings will not be held.

UNIX filename: tnonteach

Informix filename: nonteach_table

Schema location: $CARSPATH/schema/student

Program interrelationships: crsent

Product interrelationships: None

Table/record interrelationships: Indirectly associated with the Meeting record (mtg_rec).

Nonteaching FTE Assignment record

Maintains nonteaching assignments for each faculty member and is used on the term accrual fields (accrl_meth and accrl_date) of the Faculty record (fac_rec).

UNIX filename: noninstr

Informix filename: noninstr_rec

Schema location: $CARSPATH/schema/student

Program interrelationships: crsent

Product interrelationships: None

Table/record interrelationships: Associated with the Faculty record (fac_rec) and the Non-Instructional table (noninstr_table).

Period table

Describes the periods with their times and days of the week.

UNIX filename: tprd

Informix filename: prd_table

Schema location: $CARSPATH/schema/student

Program interrelationships: crsent

Product interrelationships: None

Table/record interrelationships: Associated with the Meeting record (mtg_rec).

Prerequisite record

Describes the group to which requisites and types of requisites associated with a course belong. Requisites in the same group are "anded" together. Requisites between groups are "ored" together. The group number is used to identify the groups.

UNIX filename: crsreqgrp

Informix filename: crsreqgrp_rec

Schema location: $CARSPATH/schema/student

Program interrelationships: crsent

Product interrelationships: Registration

Table/record interrelationships: Associated with the Course record (crs_rec) and Requisite Maintenance record (crsreq_rec).

Requirement table

Defines registration requirements that may be placed on sections of a course..

UNIX filename: tsecreq

Informix filename: secreq_table

Schema location: $CARSPATH/schema/student

Program interrelationships: crsent

Product interrelationships: Registration

Table/record interrelationships: Associated with the Requirement Field table ( ), Section record (sec_rec) Section Requirements record (secreq_rec).

Requirement Field table

Describes field values of the registration requirements that may be placed on courses and sections. This table is a detail table off of the Section Requirements table (secreq_table).

UNIX filename: secreqfld

Informix filename: secreqfld_table

Schema location: $CARSPATH/schema/student

Program interrelationships: crsent

Product interrelationships: Registration

Table/record interrelationships: Associated with the Section record (sec_rec).

Requisite Maintenance record

Describes course prerequisite, corequisite, and concurrent requisite information.

UNIX filename: crsreq

Informix filename: crsreq_rec

Schema location: $CARSPATH/schema/student

Program interrelationships: crsent

Product interrelationships: Registration

Table/record interrelationships: Associated with the Course record (crs_rec) and Prerequisite record (crsreqgrp_rec).

Schedule Comment record

Contains course schedule comments to be printed on course schedules.

UNIX filename: schdcomm

Informix filename: schd_comment_rec

Schema location: $CARSPATH/schema/student

Program interrelationships: crsent

Product interrelationships: Registration

Table/record interrelationships: Associated with the Section record (sec_rec).

Section record

Contains information about each section of each course being offered each session.

UNIX filename: sec

Informix filename: sec_rec

Schema location: $CARSPATH/schema/student

Program interrelationships: cnclsec, crsent, reglist, rollsess, setrefno

Product interrelationships: Grading, Registration

Table/record interrelationships: Associated with the Attendance record (ada_rec), Alternate Academic Calendar record (altcal_rec), Alternate Refund record (altrfnd_rec), Course record (crs_rec), Course Statistics record (cs_rec), Coursework record (cw_rec), Schedule Comment record (schd_comment_rec), Section Bridge record (secbrdg_rec), Section Detail record (secdtl_rec), Section Meeting (secmtg_rec), Section Requirements record (secreq_rec), Textbook record (textbk_rec).

Section Bridge record

Describes the next and previous bridge sections for a bridged section.

UNIX filename: secbrdg

Informix filename: secbrdg_rec

Schema location: $CARSPATH/schema/student

Program interrelationships: crsent

Product interrelationships: Registration

Table/record interrelationships: Associated with the Section record (sec_rec).

Section Detail record

Describes additional detail information about a section.

UNIX filename: secdtl

Informix filename: secdtl_rec

Schema location: $CARSPATH/schema/student

Program interrelationships: crsent

Product interrelationships: None

Table/record interrelationships: Associated with the Section record (sec_rec).

Section Meeting record

Contains information linking Section and Meeting records.

UNIX filename: secmtg

Informix filename: secmtg_rec

Schema location: $CARSPATH/schema/student

Program interrelationships: crsent

Product interrelationships: Registration

Table/record interrelationships: Associated with the Section record (sec_rec) and Meeting record (mtg_rec).

Section Requirements record

Describes registration requirements specific for a section.

UNIX filename: secreq

Informix filename: secreq_rec

Schema location: $CARSPATH/schema/student

Program interrelationships: crsent

Product interrelationships: Registration

Table/record interrelationships: Associated with the Section record (sec_rec) and Section Requirements table (secreq_table).

Section Requirements table

Describes the registration requirements that may be placed on course sections.

UNIX filename: secreq

Informix filename: secreq_table

Schema location: $CARSPATH/schema/student

Program interrelationships: crsent

Product interrelationships: Registration

Table/record interrelationships: Associated with the Section record (sec_rec).

Textbook record

Describes specific information about a textbook for a course.

UNIX filename: textbk

Informix filename: textbk_rec

Schema location: $CARSPATH/schema/student

Program interrelationships: crsent

Product interrelationships: None

Table/record interrelationships: Associated with the Section record (sec_rec).

SECTION 4 – COURSE/CLASS SCHEDULE MACROS AND INCLUDES

Overview

Introduction

This section provides reference information about macros and includes used to set up the Course/Class Schedule product.

Relationship Among Macros, Includes, and C Programs.

For all elements of the product other than C programs, m4 macros contain the definitions of features that have been designed to be modified for each institution. These macros, located under $CARSPATH/macros, are passed through the m4 processor.

CX contains an alternative method for the setting of features in C programs. Macros in the source code of C programs are not passed through the m4 processor because the C compiler has its own preprocessor, the cc.

To provide the same apparent functionality to C programs, a section in the include source directory has been allocated for a type of include file that acts as an intermediary between the m4 processor and the cc preprocessor. In operation, m4 macros are defined whose output is a valid cc macro. These m4 macros are placed in the include files. Then the include files are translated and the appropriate cc macro is placed in the include file. The installed include file is included by the C compiler at compile time so that the result of the compilation is influenced by the m4 macro.

General Installation Procedures

See the *CX Implementation and Maintenance Technical Manual,* for general procedures on setting and installing changes to macros and includes.

Configuration Table

The Configuration table (config_table) is used in some cases to provide run time configuration values.

Macros

Introduction

CX contains macros that define specific values used throughout the Course/Class Schedule product. The macros and includes enable you to change the available options and functionality of the Course/Class Schedule product without having to modify C code. By modifying macros, you can customize your implementation of the Course/Class Schedule product and make the product 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:

- Enable - 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 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 locate Course/Class Schedule macros, access the $CARSPATH/macros/custom/student file. To locate Common macros, access the $CARSPATH/macros/custom/common file.

Applocate Program

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

Note: To locate the macros used in Course/Class Schedule, using *Applocate*, you must specify the product name.

Follow these steps to run the *Applocate* program.

1. Select Utility Menu 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 Macro Category field. A list of module names appears in a Table Lookup box.
5. Select a module name (e.g., CATALOG/SCHEDULE), and click **OK**. The Table Lookup box disappears.
6. Select **Finish**. The Output Parameters and Scheduling window appears.
7. Enter the following:
   - In the Time field, enter **NOW**.

- In the Background field, enter **Y**.

8. Select **Finish**.

The system creates the file, *applocate.out*, and sends it to your home directory.

Enable Macros

The following lists the *crsent* enable macros located in the *student* macro file.

`ENABLE_FEAT_CRSDTL', `N'
Defines the inclusion of the crsdtl_rec in the course entry screen (i.e., Course Catalog screen). A Y means the inclusion is enabled, and an N means it is disabled. If it is disabled, the *crsent* program will not maintain the crsdtl_rec.

`ENABLE_FEAT_SECDTL', `N'
Defines the inclusion of the secdtl_rec in the section entry screen (i.e., Section Record screen). A Y means the inclusion is enabled, and an N means it is disabled. If it is disabled, the *crsent* program will not maintain the secdtl_rec.

'ENABLE_FEAT_CRSAREA', 'N'
Enables the option to maintain the course area records.

'ENABLE_FEAT_REGCTGRY', 'Y'
Enables the option to maintain the registration category records.

`ENABLE_FEAT_FACLOAD', `Y'
Defines the inclusion of the faculty load in the meeting and instructor screens (i.e., Section Record and Instructor Information screens). A Y means the inclusion is enabled, and an N means it is disabled. If it is disabled, the *crsent* program will not automatically maintain faculty load.

`ENABLE_FEAT_TOT_MTG_HRS', `Y'
Defines the inclusion of the maintenance of total hours value in Meeting records. Governs the existence of total hours fields (i.e., CalcHrs) on the Meeting Schedule window and also maintenance of the manual hours field (i.e., ManHrs). If this is set to N, the user is not forced to enter a total meeting hours value when one cannot be calculated automatically. If set to Y, the user will have to maintain total hours on Meeting records for TBA (to be announced) courses.

`ENABLE_FEAT_ALTCAL', `Y'
Defines the inclusion of the Section Calendar window in the *crsent* program.

`ENABLE_FEAT_CHANGE_FLT', `Y'
Specifies whether the faculty load type for Instructor records (instr_rec) should be changed automatically to preparatory for sections that are cancelled.

`ENABLE_FEAT_DISPLAY_FAC_INFO', `N'
Specifies whether the display only parameter in *crsent* should include the ability to view faculty information.

`ENABLE_FEAT_ADA', `N'
Enables the ADA (Average Daily Attendance) screen and the automatic maintenance of Attendance records (ada_rec) by the *crsent* program.

`ENABLE_FEAT_ARTIC', `N'
Defines the inclusion of the Course Articulation Program screen in *crsent*. This enable macro is designed for maintenance of California MIS data requirements.

`ENABLE_FEAT_CRSDTL_CA', `N'
Defines the inclusion of the California crsdtl_rec in the course entry screen (i.e., Course Catalog screen). A Y means the inclusion is enabled, and an N means it is disabled. The enabling of this option assumes that related California functionality also will be enabled (i.e.,

maintenance of Attendance records (ada_rec) records. This also enables MIS (Management Information Systems) fields related to the Course record (crs_rec).

`ENABLE_FEAT_SECDTL_CA', `N'
Defines the inclusion of the California secdtl_rec in the section entry screen (i.e., Section Record screen). A Y means the inclusion is enabled, and an N means it is disabled. The enabling of this option assumes that related California functionality also will be enabled (i.e., maintenance of Attendance records). This also enables MIS fields related to the Section record.

The following lists the *crsent* faculty load macros located in the *student* macro file.

`FACULTY_LOAD_CONSTANT', `16'
Defines the constant used in the calculation of faculty load. This is normally the number of weeks for a standard, primary session.

CAUTION: *Do not change this macro*. If you change this macro, Course/Class Schedule will not process properly.

`FAC_NON_INSTR_LOAD_TYPE', `NI'

`FAC_PREP_LOAD_TYPE', `PR'
Defines the load type for noninstruction assignments and for preparation given to instructors. If a class is cancelled, the *crsent* program automatically will exchange the faculty load macro FAC_PREP_LOAD_TYPE for the existing load type.

`FAC_ACCRL_VALID', `B,C,H,O,R,X'

`FAC_ACCRL_INCL', `include=(FAC_ACCRL_VALID),upshift'

`FAC_ACCRL_BANK', `B'

`FAC_ACCRL_CONTRACT', `C'

`FAC_ACCRL_HOURLY', `H'

`FAC_ACCRL_OVER', `O'

`FAC_ACCRL_RECORD', `R'

`FAC_ACCRL_OTHER', `X'
Defines the valid accrual methods for faculty load.

CAUTION: *Do not change this macro*. If you change this macro, Course/Class Schedule will not process properly.

`FAC_ACCRL_DEF', `C'
Defines the default faculty accrual method. Used in the instruction screen (i.e., Instructor Information screen) linked to the Meeting records (mtg_rec).

`FAC_LOAD_HRLY_ACCRL_MAX', `.64'

`FAC_LOAD_OVR_ACCRL_MAX', `.64'
Defines the maximum faculty load equivalency value that may be accrued by a faculty member in the hourly and overload accrual methods. When this value is exceeded, the *crsent* program will warn the operator when additional assignments are given to the instructor.

The following lists the *crsent* macros located in the *student* macro file.

`ALT_SKIP_WEEKEND', `N'

Defines whether to skip the weekends when determining an alternate refund or alternate calendar date. If set to Y and the alternate date falls on a Saturday or Sunday, the date put in the record will be Monday, not the weekend date. If set to N, the system will put the weekend date in the record.

`ALT_SKIP_FRIDAYS', `N'
Defines whether to skip Fridays in non-primary sessions when determining an alternate refund or alternate calendar date. If set to Y and the alternate date falls on a Friday, the date the system puts in the record will be the next suitable day, not Friday's date. The is_prim field in the Session table determines a non-primary session. If set to N, the date the system puts in the record will be Fridays.

`ALT_USE_NEXT_DAY', `N'
Defines whether to use the calendar day following the specified meeting day as the alternate date, rather than the calendar day of the meeting day.

`ALT_USE_MTG_BEGIN_DATE', `Y'
When alternate calendar and refund dates are calculated, if one is specifying the number of days, you can choose to begin counting those days either from the Meeting record begin date or from the day the first class for this meeting actually takes place. If this macro is set to Y, the system calculates the number of days from the Meeting record's begin date; otherwise, the system calculates the number of days from the date of the first actual meeting.

`CRS_STAT_VALID', `B,A,I'

`CRS_STAT_INCL', `include=(CRS_STAT_VALID),upshift'

`CRS_STAT_INACTIVE', `I'

`CRS_STAT_ACTIVE', `A'

`CRS_STAT_EX', `(B)anked, (A)ctive, (I)nactive.'

`CRS_STAT_DEF', `CRS_STAT_ACTIVE'
Defines valid course status codes. The status is used to determine what courses get rolled over for the next catalog. The statuses and their meanings are:

- B (Banked status prevents creation of section records, but does roll the course to a new catalog)
- A (Active status rolls the course and all associated records to new catalog)
- I (Inactive does not roll Course or any related records to the new catalog)

`FACIL_STAT_VALID', `A,I'

`FACIL_STAT_INCL', `include=(FACIL_STAT_VALID),upshift'

`FACIL_STAT_ACTIVE', `A'
Defines valid facility status codes. In the *crsent* program, an inactive facility cannot be used for meeting assignments.

`FACIL_STAT_DEF', `FACIL_STAT_ACTIVE'
Defines the default code for facility status.

`FACIL_TYPE_DEF', `C'

`FACIL_TYPE_CLASSRM', `C,A'

`FACIL_TYPE_OFFICE', `O,B'

`FACIL_TYPE_VALID', `C,O,A,B,D

`FACIL_TYPE_INCL', `include=(FACIL_TYPE_VALID),upshift')

Defines the type of facility available for classroom instruction and for office rooms. This is the value in the facil_table.ctgry field for classrooms and offices. Course Entry uses these values to define which facilities are valid for teaching assignments.

`MTG_DAYS', `UMTWRFS'
Defines the days string to use to indicate what days a class meets. The string must contain a unique set of characters.

`PREREQUISITE_TYPE',`P'

`CO_REQUISITE_TYPE',`C'

`CONCURRENT_REQUISITE_TYPE', `N'
Defines requisite types.

CAUTION: *Do not change this macro*. If you change this macro, Course/Class Schedule will not process properly.

`SEC_STAT_OPEN', `O'

`SEC_STAT_REOPEN', `R'

`SEC_STAT_HOLD', `H'

`SEC_STAT_CLOSED', `C'

`SEC_STAT_CANCL_IN_PROGRESS', `I'

`SEC_STAT_CANCEL', `X'

`SEC_STAT_VALID', `R,H,O,C'

`SEC_STAT_INCL', `include=(SEC_STAT_VALID),upshift')

`SEC_STAT_DEF', `SEC_STAT_OPEN'
Defines the valid section statuses. The following are the values and their meanings:

- O (Open status) allows registration based on the available number of seats.
- R (Reopened status) is assigned by the system automatically if the section was closed during registration, but the number enrolled dropped below the maximum registered. This status also may be set manually by the operator for closed and cancelled sections. The automatic assignment of this status is dependent on the definition of the macro for reopening closed sections.
- H (Hold status) blocks registration and prevents printing this section on the class schedule.
- C (Closed status) blocks registration. The *regent* program automatically sets this status when the number enrolled equals the maximum to be registered.
- I (Cancel in progress status) is set when the operator has executed the cancel option in *crsent*. Classes with this status are not available for registration.
- X (Cancelled status) indicates the section has been cancelled for purposes of registration. When *regent* is run to void those students in a section with cancel in progress, the status automatically is set to cancelled when all students have been successfully voided out of the section.

The following lists the *regent* enable macros located in the *student* macro file.

`CAT_EG', `, eg: CAT_DEF.'
Course Catalog Example

Note: No updating is necessary. Input of valid catalogs should have been accomplished in $CARSPATH/macros/custom/periodic to CAT_ VALID.

`CRS_LEVEL_VALID',`LL,UL," "'

`CRS_LEVEL_INCL',`include=(CRS_LEVEL_VALID)'

`CRS_LEVEL_EG', `, eg: LL,UL.'
Designates the level of instruction of the course. The example values are UL for upper level and LL for lower level.

`CRS_EG', `, eg: MATH201.'
Defines the course example, level example and default.

Note: Generally these macros will not require changes. These are not used extensively, but do have some applications in some informers.

`CRS_MIN_HRS',`3'

`CRS_MAX_HRS',`3'

`SEC_HRS',`3'
Defines the defaulted number of hours.

Note: Update as necessary. They should represent the credit hours that are established for the majority of their courses. If the majority of their courses are 5 hour, then use 5. This is used in the *crsent* screen defaults on *crsent*.

`HRS_FORMAT', `#&.&&'

`PERFORM_HRS_FORMAT', `##.##'
Defines the course hours format.

Note: Normally no changes are necessary during *regist* implementation.

`INSTR_RPTG_DEF',`B'

`INSTR_RPTG_VALID',`I,G,B,N'

`INSTR_RPTG_INCL',`include=(INSTR_RPTG_VALID),upshift'
Defines the default instructor FTE reporting method: Institutional, Government, Both, or None. The *crsent* program uses this macro at the Meeting record level.

`MAX_REG_DEF',`50'

`MAX_WAIT_DEF',`5'
Defines maximum values for sections.

Note: Update to the most frequently used values for registered and wait list students. If the class size that is used most by a client is 35, change to 35. If they don't want to use the wait list, use ‘0’ as the default.

`ADA_CENSUS_DATE1_PCT', `20'

`ADA_CENSUS_DATE2_PCT', `60'
Defines the percent used to calculate ADA census dates for nonstandard sections. This is normally 20% and 60%.

`ADA_AAM_EXEMPT', `EX'

`ADA_AAM_DAILY_ATTEND', `DC'

`ADA_AAM_WEEKLY_ATTEND', `WC'

`ADA_AAM_POS_ATTEND', `PA'

`ADA_AAM_BRIDGED', `BR'

`ADA_AAM_NONCRED', `PN'

`ADA_AAM_WORK_EXP_BY_WK', `WI'

`ADA_AAM_WORK_EXP_BY_DAY', `DI'

Defines each value of the Attendance Accounting Method (AAM) values. The AAM is related to the ADA/FTE processing incorporated into the Jenzabar, Inc. product. If enabled, the *crsent* program maintains the AAM interactively.

`SEC_MIN_SESS_WKS', `17'

`SEC_MAX_SESS_WKS', `18'
Defines the maximum and minimum number of weeks in a standard session. These values are used with the weeks value in the Section record to determine whether a section is standard with regard to census dates for ADA.

`IM_WORK_EXPERIENCE', `WE'

`IM_INDEPENDENT_STUDY', `IN'

`IM_IN_SERVICE', `IS'
Defines the values in the Instructional Method table (im_table) for the im method associated with independent student and work experience.

`FTES_RPT_SITES', `CARS'
Defines the sites that are processed in the FTES report.

`STUDENT_OVERRIDE_MAX_REPEATS'
Allows or disallows students to register for a class on the Web that has been repeated the maximum number of times.

Note: This macro only applies to the Web.

The following lists the MIS Elements enable macros located in the *student* macro file.

`CREDIT_STAT_VALID', `D,C,N,S')

`CREDIT_STAT_INCL', `include=(CREDIT_STAT_VALID),upshift'

`CREDIT_STAT_DEF', `D'
Defines valid credit status codes for the Course record (crs_rec).

`ENABLE_FEAT_SITE',`Y'
Enables the use site field.

`TRANSFER_STAT_VALID', `A,B,C'

`TRANSFER_STAT_INCL', `include=(TRANSFER_STAT_VALID),upshift'

`TRANSFER_STAT_DEF', `A'
Defines transfer status codes for the Course record (crs_rec).

`REMEDIAL_STAT_VALID', `P,N'

`REMEDIAL_STAT_INCL', `include=(REMEDIAL_STAT_VALID),upshift'

`REMEDIAL_STAT_DEF', `N'
Defines remedial status codes for the Course record (crs_rec).

`SPECIAL_CLASS_VALID', `S,P,N'

`SPECIAL_CLASS_INCL', `include=(SPECIAL_CLASS_VALID),upshift'

`SPECIAL_CLASS_DEF', `N'
Defines special class status codes for the Course Detail record (crsdtl_rec).

`COURSE_REPEAT_VALID', `P,N'

`COURSE_REPEAT_INCL', `include=(COURSE_REPEAT_VALID),upshift'

`COURSE_REPEAT_DEF', `N'
Defines course repeatability codes for the Course record (crs_rec).

`COOPED_STAT_VALID', `N,C'

`COOPED_STAT_INCL', `include=(COOPED_STAT_VALID),upshift'

`COOPED_STAT_DEF', `N'
Defines cooperative education status codes for the Course record (crs_rec).

`EXCESS_COSTS_DEF', `D'

`EXCESS_COSTS_VALID', `D,N,U'

`EXCESS_COSTS_INCL', `include = (EXCESS_COSTS_VALID), upshift'
Defines excess costs codes for the Course record (crs_rec).

The following lists the Student Services macros located in the *student* macro file.

`ENABLE_FEAT_R25', `N'
Defines whether the R25/S25 menu will display. A “Y” will display the menu, and an “N” will not display the menu.

`ENABLE_FEAT_S25', `N'
Defines whether the S25 menu options will display. A “Y” will display the menu, and an “N” will not display the menu.

`R25_DATABASE', `r25'
Defines the name of the Resource25 database.

CAUTION: If you change the value of this macro, you also must go to /opt/cisodbc/<release name> and change the name of the Resource25 database in the two files: cisinv.env.r25 and cisaps.env.r25.

`R25_VCAL_FILES', `$ENV{CARSPATH}/text/r25'
Defines the root directory where Resource25 reads and writes vCalendar files. The request and publish subdirectories must exist under this path.

`S25_FILES', `$ENV{CARSPATH}/text/s25'
Defines the directory where Schedule25 class descriptor files should be exported and imported.

`S25_FACIL_TYPES', `FACIL_TYPE_VALID’
Defines which facility types should be included in the export of the campus profile (e.g., C, A, O).

`S25_FACIL_DEF', `C'
Defines the default room type preference for all departments exported in the campus profile.

`S25_PARTITION_DEF', `CARS,MAIN'
Defines the default partition preference for all departments exported in the campus profile. This should consist of the campus and building.

`R25_EVENT_TYPE', `MEETING'
Defines the event type in Resource25 for CX meetings.

`DEGAUD_AUTO_SELECT_SUBAUDITS', `N'
Defines whether Audit Input records (audin_recs) are going to be used when adding Contact records (ctc_rec) for degree audit.

`WEB_CRSAUTH_REASON_DEF', `IA'
Defines the default reason code used for course authorizations via the Web.

The following lists the Course/Class Schedule periodic macros located in the *periodic* macro file.

Course catalog defaults

The following macros define course catalog defaults used in the Course Entry screens.

- `CAT_DEF', `UG92'
- `CAT_NEXT', `UG93'

Course catalog table valid and include values

The following macros define course catalog table valid and include values.

- `CAT_VALID', `CAT_DEF,CAT_NEXT,PREV,XFER,UG93,UG94,UG95,UG96,UG97'
- `CAT_INCL', `include=(CAT_VALID), upshift')

Includes

Introduction

The Course/Class Schedule product contains includes, which determine the features that are enabled in the product. An include can be either a compile option that enables or disables a feature, or a default value for a feature.

To enable a feature in the Course/Class Schedule product, you must define an include in $CARSPATH/include/custom. To disable an include, comment out the include in the same file. See the *CX System Reference Technical Manual,* for more information on enabling and disabling includes. By modifying includes, you can customize your implementation of the Course/Class Schedule product and make the product easier to maintain.

Purpose

An include allows you to activate or deactivate features in C programs without changing the C code. You also can specify compilation values for an entry program in the Course/Class Schedule product.

Macro Dependency

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

How to Locate Includes

To locate a Course/Class Schedule include, access the $CARSPATH/include/ directory.

Note: For more information about using the MAKE processor and modifying includes, see the *CX System Reference.*

Application Includes

The following lists and describes the application include files that affect or have an effect on the Course/Class Schedule product. The includes are located in the following directory path: $CARSPATH/include/applic.

crsent
Contains includes that define the common macros used within the Course Entry program.

SECTION 5 – JENZABAR CX PROGRAM FILES

Overview

Introduction

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

Program Files Detailed

This section contains details about the following files:

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

def.c

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

mac.h

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

Definition File

Every program uses a definition (def.c) file, located in the following paths:

- $CARSPATH/src/regist/cnclsec
- $CARSPATH/src/regist/crsent
- $CARSPATH/src/regist/histeq
- $CARSPATH/src/regist/rollsess
- $CARSPATH/src/regist/setrefno

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

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

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

- Includes for a mac.h file
- Global program variables
- Includes for schema files def.c files
- Form pointers that provide the location for forms
- Sqlda pointers that bind the file structure to the form
- dmm, dml, and dmlt definitions
- Program parameters

- Declarations of functions so the compiler can handle a call of that function

Example of a def.c File

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

Note: The legend for the file contents directly follows this example.

```
#include "mac.h"                                                                    1


#include <schema/student/acadcaldef.c>                                              2
#include <schema/student/adadec.h>


/* -----                                                                            3
        Global structures.
----- */
struct dbview facld_fields[] =
    {
        {"id"},
        {"facld_no"},


/* -----                                                                            4
        Define screen pointers.
----- */
SCREEN *artic_scrl;
SCREEN *crs_scr;
SCREEN *crs_scrl;

/* -----
        Define scr_bind list.                                                       5
      SCREEN_NAME, &screen, "scr_field", BIND_TYPE, (char *)&prog_buffer,
        BIND_MODE

----- */

SCR_MENUSTART(menu_getset)
    SCR_MENUOPT2(0, NULL, SCR_DONE, SCR_GMENABLE, NULL, NULL),
    SCR_MENUOPT2(0, NULL, SCR_ABORT, SCR_GMENABLE, NULL, NULL),                     6

am_list) / (sizeof(param_list[0])));
```

Legend for the def.c file:

1. mac.h include
2. schema file def.c's
3. global structures
4. screen pointers
5. screen binds
6. structure definitions

mac.h Files

Every program uses a macro header (mac.h) file, located in the following paths:

- $CARSPATH/src/regist/cnclsec
- $CARSPATH/src/regist/crsent
- $CARSPATH/src/regist/histeq
- $CARSPATH/src/regist/rollsess
- $CARSPATH/src/regist/setrefno

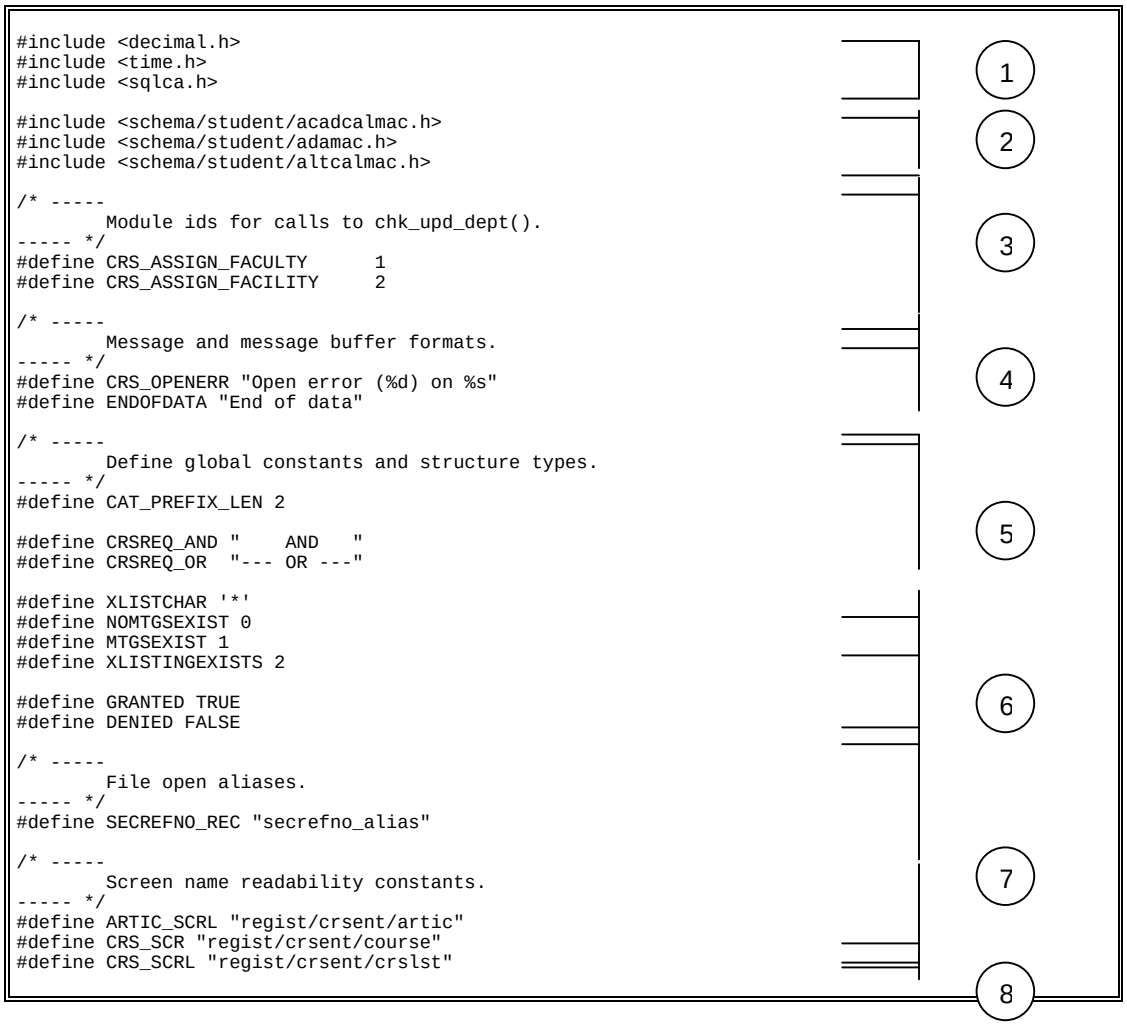
The *mac.h* file for a screen-oriented program can contain the following information:

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

Example of a mac.h File

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

Note: The legend for the file contents directly follows this example.


#include <decimal.h>
#include <time.h>
#include <sqlca.h>
#include <schema/student/acadcalmac.h>
#include <schema/student/adamac.h>
#include <schema/student/altcalmac.h>
/* -----
Module ids for calls to chk_upd_dept().
----- */
#define CRS_ASSIGN_FACULTY 1
#define CRS_ASSIGN_FACILITY 2
/* -----
Message and message buffer formats.
----- */
#define CRS_OPENERR "Open error (%d) on %s"
#define ENDOFDATA "End of data"
/* -----
Define global constants and structure types.
----- */
#define CAT_PREFIX_LEN 2
#define CRSREQ_AND " AND "
#define CRSREQ_OR "--- OR ---"
#define XLISTCHAR '*'
#define NOMTGSEXIST 0
#define MTGSEXIST 1
#define XLISTINGEXISTS 2
#define GRANTED TRUE
#define DENIED FALSE
/* -----
File open aliases.
----- */
#define SECREFNO_REC "secrefno_alias"
/* -----
Screen name readability constants.
----- */
#define ARTIC_SCRL "regist/crsent/artic"
#define CRS_SCR "regist/crsent/course"
#define CRS_SCRL "regist/crsent/crslst"
1
2
3
4
5
6
7
8

Legend for the mac.h file:

1. includes for header files
2. includes for schema files
3. module IDs
4. formats
5. global constants and structure types
6. program constant definitions
7. aliases
8. readability constants

SECTION 6 – CANCEL SECTIONS PROGRAM

Overview

Introduction

This section provides reference information about the Cancel Sections program (*cnclsec*). The Cancel Section program allows you to automatically cancel a course or specified section and update registration(s). Before running *cnclsec*, you must first mark the section for cancellation in the Course Entry program (*crsent*). You can run *cnclsec* for any of the following:

- All sections ready to be canceled
- All sections of a course number
- A single course and section

The *cnclsec* program creates a Contact record for the canceled section and an ACE report called *ltrcnclsec* that generates mailings to students. An institution must create an entry in the Contact table for cancellation letters if it wants to generate automated mailings.

Program Features Detailed

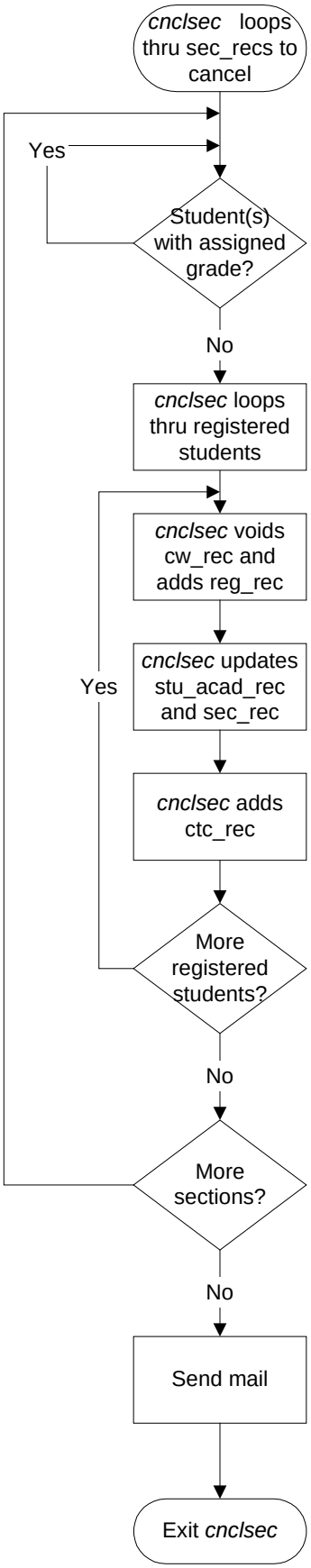
This section contains details about the following features of the Cancel Sections program:

- Process flow
- Parameters
- Program screens and windows

Process Flow

Diagram

The following diagram shows the flow of data in the Cancel Sections program.


cnclsec loops thru sec_recs to cancel
Yes
Student(s) with assigned grade?
No
cnclsec loops thru registered students
cnclsec voids cw_rec and adds reg_rec
cnclsec updates stu_acad_rec and sec_rec
Yes
cnclsec adds ctc_rec
More registered students?
No
More sections?
No
Send mail
Exit *cnclsec*

Data Flow Description

The following describes the data flow in the Cancel Sections program.

1. The *cnclsec* program loops through sec_recs to cancel.
2. The *cnclsec* program determines whether any student(s) in class have an assigned grade.
3. If the *cnclsec* program does not find any student(s) in class with an assigned grade, it loops through registered students. If the *cnclsec* program does find any student(s) in class with an assigned grade, it looks for more students in class with an assigned grade.
4. The *cnclsec* program voids cw_recs and adds reg_recs.
5. The *cnclsec* program updates stu_acad_recs and sec_recs.
6. The *cnclsec* program adds ctc_recs.
7. The cnclsec program looks for more registered students. If it finds more registered students, it returns to step 4. If it does not find any more registered students, it goes to the next step.
8. The *cnclsec* program looks for more sections. If it finds more sections, it goes to step 2. If it does not find more sections, it goes to the next step.
9. The *cnclsec* program creates an ACE report that generates mailings to students.
10. Exit the *cnclsec* program.

Program Relationships

The Cancel Sections program uses the following libraries.

- *Libcrs*
- *Libreg*

Tables and Records Used

The *cnclsec* program uses the following Common and Course/Class Schedule tables and records.

Note: For information about the Common tables and records, see the *CX System Reference Technical Manual*. For information about the Course/Class Schedule tables and records, see the section *Course/Class Schedule Tables and Records* in this manual.

Common tables and records

- ctc_rec

Course/Class Schedule tables and records

- cw_rec
- reg_rec
- sec_rec
- stu_acad_rec
- acad_cal_rec

Special Function Flags

The Cancel Sections program uses no records that have special function flags.

Parameters

Introduction

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

Note: You also can specify compilation values with the includes for the Course/Class Schedule product that affect the Cancel Sections program.

Parameter Syntax

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

The following is the correct usage for running the Cancel Sections program from the UNIX shell:

cnclsec [-C crs_no] -c cat -S sess -y year [-s sec_no] -t tick_code -r ctc_resrc -d ctc_due_date [-n No show reason code] [-l Last attended date] [-w Withdrawal date]

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

Parameters

The following lists the parameters for running Cancel Sections.

-C *crs_no*
Optional - Specifies the course number of the course to cancel.

Example: cnclsec –C ENG100

Note: You must use this parameter to pass a single course number.

-c *cat*
Required - Specifies the catalog of the section to cancel.

Example: cnclsec –c UG00

-S *sess*
Required - Specifies the session of the section to cancel.

Example: cnclsec –S FA

-y *year*
Required - Specifies the year of the section to cancel.

Example: cnclsec -2000

-s *sec_no*
Optional - Specifies the section number of the section to cancel.

Example: cnclsec –s 03

Note: You must use this parameter to pass a single course number.

-t *tick_code*
Required - Specifies the tickler code to use for the Contact record.

Example: cnclsec t RFND

-r *ctc_resrc*

Required - Specifies the contact resource code for the Contact record.

Example: cnclsec –r GRDRPT

-d *ctc_due_date*
Required - Specifies the due date for the Contact record.

Example: cnclsec –d 03/03/2000

-n *No show reason code*
Optional - Specifies the reason code for no shows to use with the stu_acad_rec.

Example: cnclsec –n P

-l *Last attended date*
Optional - Specifies the date the system uses for last attended.

Example: cnclsec –l 03/03/2000

-w *Withdrawal date*
Optional - Specifies the date the system uses as the withdrawal date.

Example: cnclsec – w 03/03/2000

Program Screens and Windows

Introduction

Cancel Sections has no screens and windows because it is a batch program.

SECTION 7 – COURSE ENTRY PROGRAM

Overview

Introduction

This section provides reference information about the Course Entry program (*crsent*). The Course/Class Schedule product uses Course Entry to maintain the course catalog. It also allows you to access the Course Catalog, Section, Meeting, Faculty, and Facility records. An institution can use the "-m" menu option to determine which of the Course Entry options are available for use. The menu option is located in the following directory path: $CARSPATH/menuopt/regist/programs/crsent.

You can establish permissions for individual users or permission groups in the Course Entry program using the Operator Department table (oprdept_table). For more information on how to complete the Operator Department table, see *Building the Course/Class Schedule Table and Records* in this technical manual.

CAUTION: Faculty maintenance makes changes to the ID record (id_rec), which is used throughout the system.

Special Note

You can cancel sections using the *crsent* program. It is important for individuals responsible for the cancellation process to have access to the *crsent* program.

Program Features Detailed

This section contains details about the following features of the Course Entry program:

- Process flow
- Parameters
- Program screens and windows

Process Flow

Diagram (Course Catalog) – Course Entry

The following diagrams show the flow of data in the Course Entry program.

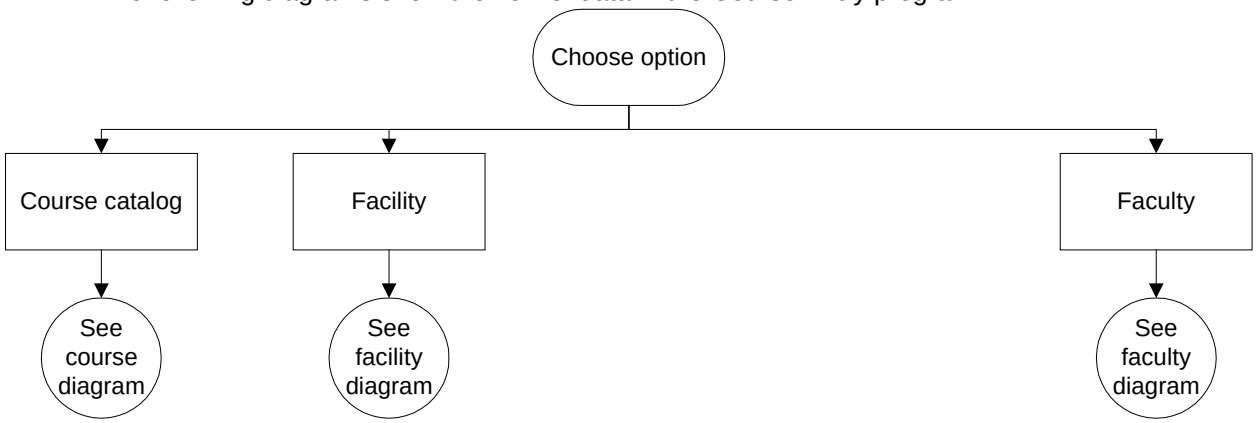


Diagram (Course) – Course Entry

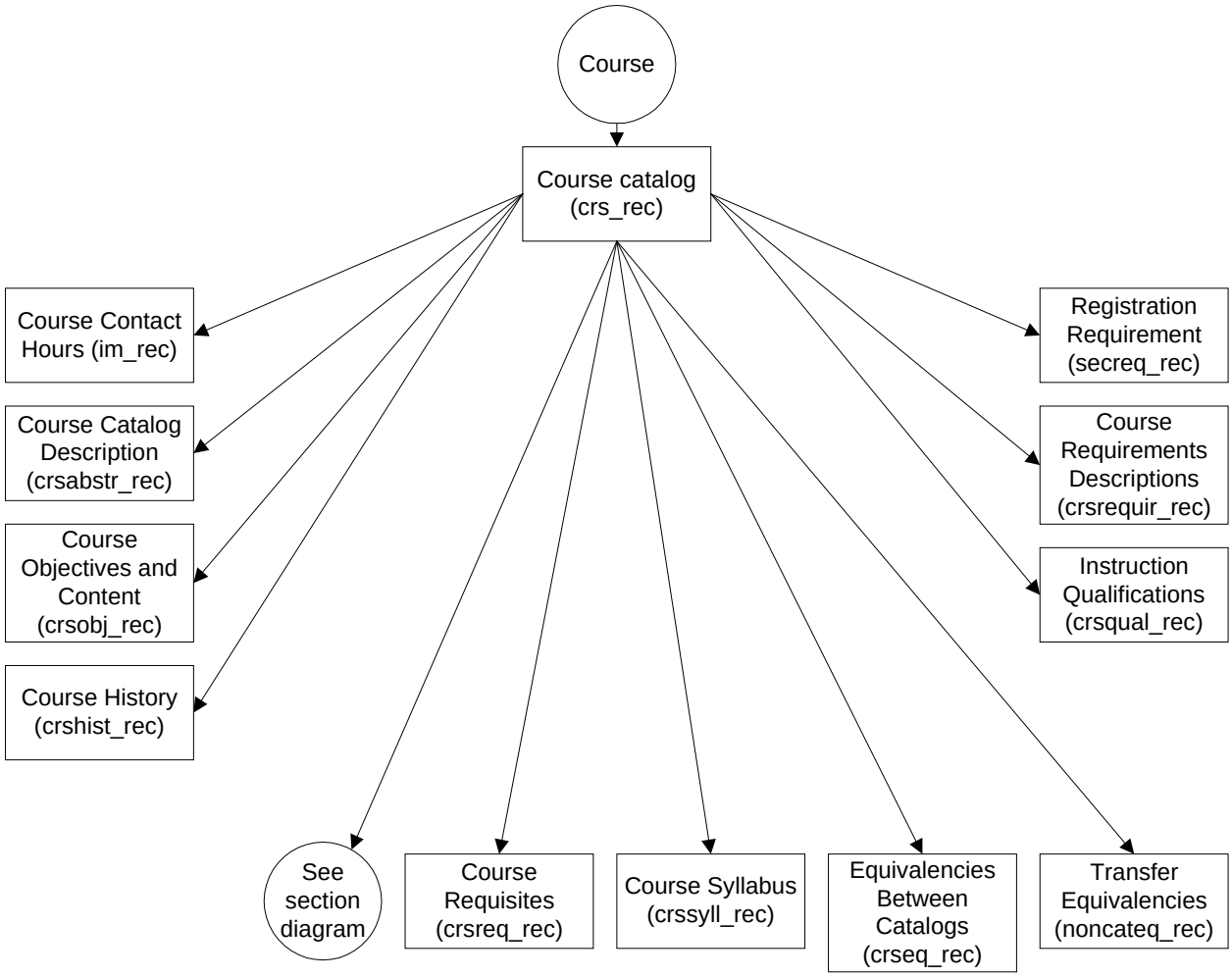


Diagram (Section) – Course Entry

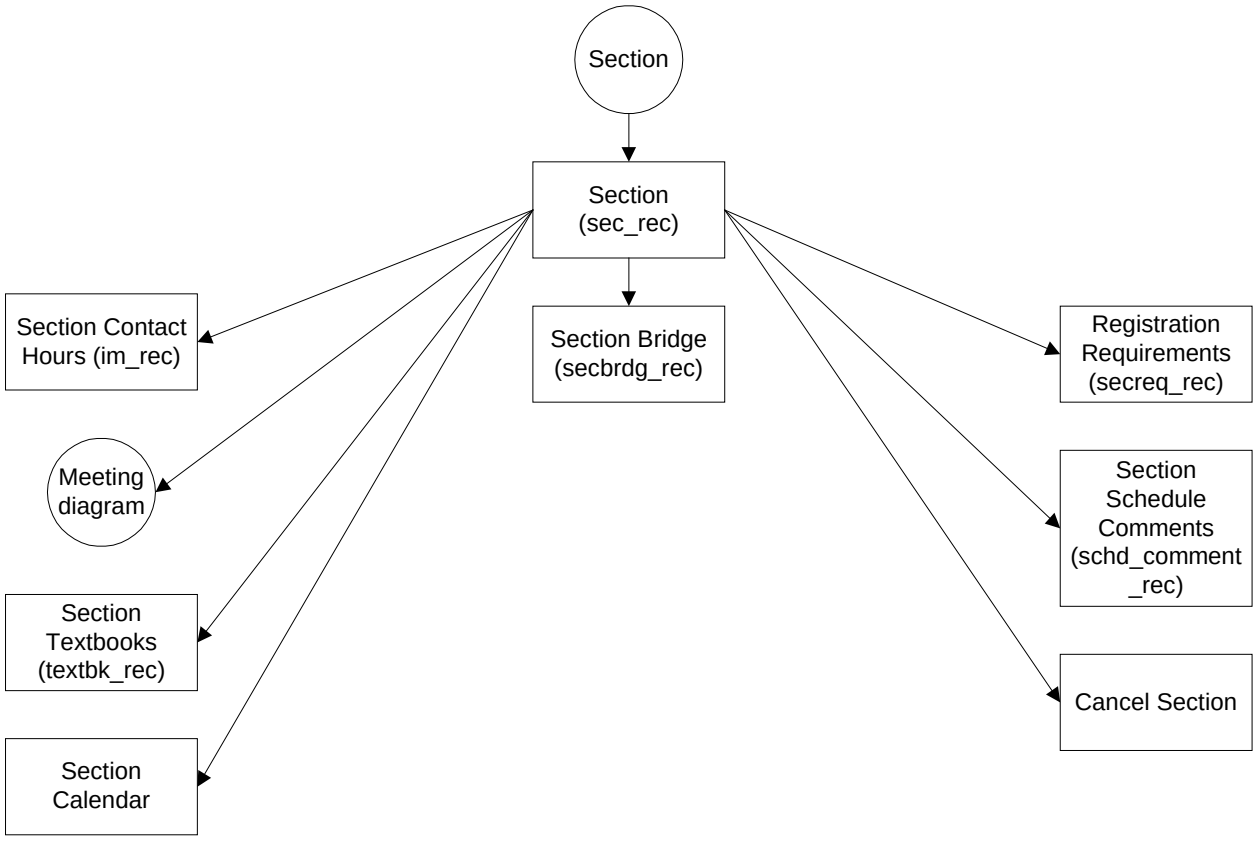


Diagram (Meeting) – Course Entry

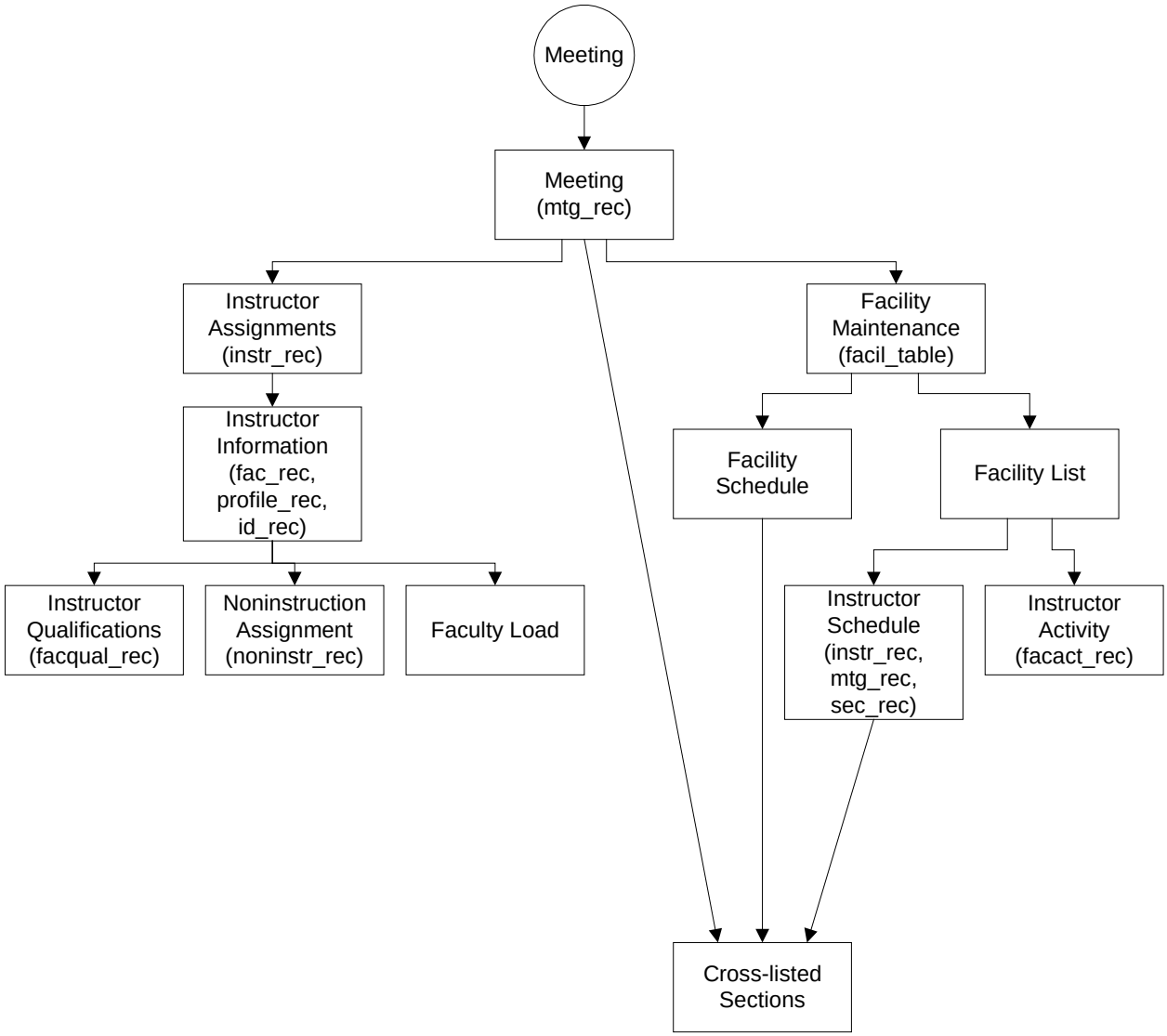


Diagram (Facility) – Course Entry

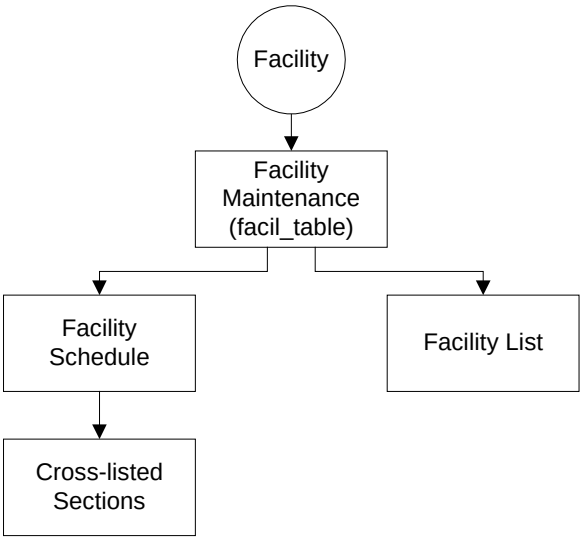
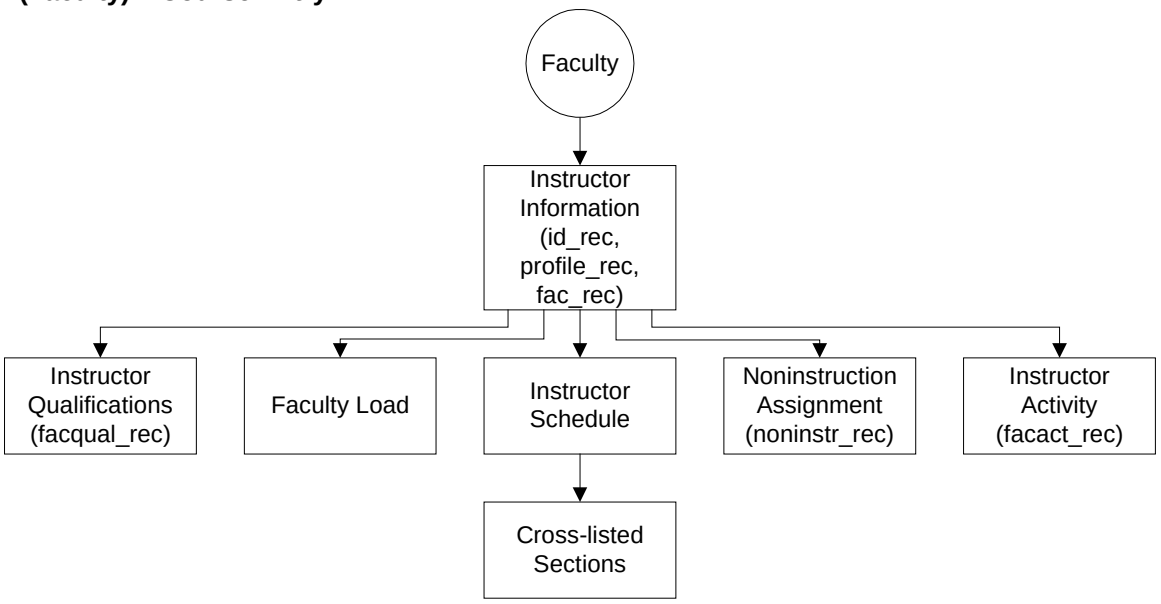


Diagram (Faculty) – Course Entry



Data Flow Description

The following describes the data flow in the Course Entry program.

1. The *crsent* program accesses the course catalog option, faculty option, or facility option.
2. If *crsent* accesses the course catalog option, it accesses the crs_rec and appropriate records related to it..
3. If there are sec_recs to view or update, *crsent* accesses the sec_rec and appropriate records related to it.
4. If there are mtg_recs to view or update, *crsent* accesses the mtg_rec and appropriate records related to it.
5. If facility maintenance must be done, *crsent* accesses the appropriate records.

6. If faculty maintenance must be done, crsent accesses the appropriate records.
7. Exit *crsent*.

Program Relationships

The Course Entry program uses the following libraries.

- *Libcrs*
- *Libreg*

Tables and Records Used

The *crsent* program uses the following Common and Course/Class Schedule tables and records.

Note: For information about the Common tables and records, see the *CX System Reference Technical Manual*. For information about the Course/Class Schedule tables and records, see the section *Course/Class Schedule Tables and Records* in this manual.

Common tables and records

- facact_rec
- fac_rec
- facil_table
- facqual_rec
- id_rec
- profile_rec

Course/Class Schedule tables and records

- acad_cal_rec
- ada_rec
- alt_cal_rec
- altrfnd_rec
- artic_rec
- cat_table
- crsabstr_rec
- crs_rec
- crsdtl_rec
- crseq_rec
- crshist_rec
- crsobj_rec
- crsqual_rec
- crsreq_rec
- crsreqgrp_rec
- crsrequir_rec
- crssyll_rec
- flt_table
- im_rec
- im_table
- instr_rec
- mtg_rec
- noncateq_rec
- noninstr_rec
- prd_table
- schd_comment_rec
- sec_rec
- secbrdg_rec
- secdtl_rec

- secmtg_rec
- secreq_rec
- textbk_rec

Special Function Flags

The Course Entry program uses no records that have special function flags.

Parameters

Introduction

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

Note: You also can specify compilation values with the includes for the Course/Class Schedule product that affect the Course Entry program.

Parameter Syntax

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

The following is the correct usage for running the Course Entry program from the UNIX shell:

crsent [-m modules] [-d] -L site [-l] [-g] [-u] -y sessyr list [-Y] [-v create R25 vcal files] [-R chk R25 conflicts]

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

Parameters

The following lists the parameters for running Course Entry.

-m *modules*

Optional – Specifies to restrict use to just facility (I) and/or faculty (U) modules and exclude the Course Entry functions.

- To restrict access to just the facility functions, enter **-m I**. To restrict access to facility and faculty function, enter **-m IU**.
- The "-m" parameter is optional and can be used to allow access into any combination of the following three processing areas:
  - Course/section entry and maintenance
  - Facility maintenance
  - Faculty maintenance

-d

Optional - Specifies access to Course Entry in display-only mode and overrides the permissions in the Operator Department table.

-L *site*

Required - Specifies the default value of the site (e.g., -L CARS).

-l

Optional - Specifies to allow access to all sites.

-g

Optional – Enables Meeting Detail record information (mtgdtl_rec) from the section screen.

-n

Optional – Enables e-mail to be sent to students in a section whose schedule information has changed. When passed, if a meeting change has occurred after the acad_cal_rec.schd_print, or if other changes have occurred and the TRACK_SEC_CHANGES is set to "Y", the operator will be asked if they want to e-mail the students about the schedule change for the course.

-u

Optional – Disables the display-only window that contains the unmet instruction method requirements.

-y *sessyr list*

Required - Specifies the session and year parameter list (e.g., -y FA0X SP0Y FA0Y SP0Z).

-Y

Optional – Allows access to all session and year entries in the Academic Calendar record (acad_cal_rec).

Note: You can access all year and session combinations in the academic calendar if you specify the “-Y” option.

-v *create R25 vcal files*

Optional – Enables the creation of R25 vcal files at the time of meeting updates.

-R *chk R25 conflicts*

Optional – Enables conflict checking with Resource25 data.

Program Screens and Windows

Introduction

Course Entry has screens and windows for performing the following interactive functions:

- Entering course information
- Entering instructor information
- Entering section information
- Maintaining the course catalog
- Maintaining section schedule information
- Updating course information
- Updating instructor information
- Updating section information

Access

The screen and window files for Course Entry are located in the following directory paths:

- $CARSPATH/modules/regist/progscr/crsent

Notes:

- You can access windows from each program screen in Course Entry.
- See the *CX System Reference Technical Manual* for information about common windows that appear in Course/Class Schedule.

Screen Files and Table/Record Usage

The Course Entry screens and windows appear in the following files and use the indicated tables and records.

artic

Contains the Course Articulation Program screen, which allows viewing or creation of an artic_rec to be associated with a course .

Access: $CARSPATH/modules/regist/progscr/crsent

Tables/Records:

- artic_rec
- artic_table

course

Contains the Course Catalog screen.

Access: $CARSPATH/modules/regist/progscr/crsent

Tables/Records:

- cat_table
- cntg_table
- crs_rec
- crscl_table
- crsctgry_table
- crsdtl_rec
- dept_table
- disc_table
- grdg_table
- major_table

- prog_table
- sam_table

crscattxt

Contains the Course Catalog Description window.

Access: $CARSPATH/modules/regist/progscr/crsent

Tables/Records:

- crs_rec
- crsabstr_rec

crseq

Contains the Equivalencies Within Catalogs window.

Access: $CARSPATH/modules/regist/progscr/crsent

Tables/Records:

- crs_rec
- crseq_rec

crshist

Contains the Course History window.

Access: $CARSPATH/modules/regist/progscr/crsent

Tables/Records:

- crs_rec
- crshist_rec

crslst

Contains the Course Listing window.

Access: $CARSPATH/modules/regist/progscr/crsent

Tables/Records:

- cat_table
- crs_rec
- dept_table
- prog_table

crsmenu

Contains the Catalog Maintenance menu.

Access: $CARSPATH/modules/regist/progscr/crsent

Tables/Records: None

crsobjtxt

Contains the Course Objectives and Content window.

Access: $CARSPATH/modules/regist/progscr/crsent

Tables/Records:

- crs_rec
- crsobj_rec

crsqual

Contains the Instruction Qualifications window.

Access: $CARSPATH/modules/regist/progscr/crsent

Tables/Records:

- crsqual_rec
- qual_table

crsreq

Contains the Course Requisites window.

Access: $CARSPATH/modules/regist/progscr/crsent

Tables/Records:

- crs_rec
- crsreq_rec
- exam_table

crsreqmt

Contains the Registration Requirements window.

Access: $CARSPATH/modules/regist/progscr/crsent

Tables/Records:

- secreq_rec
- secreq_table

crsreqtxt

Contains the Course Requirements Description window.

Access: $CARSPATH/modules/regist/progscr/crsent

Tables/Records:

- crs_rec
- crsrequir_rec

crssyltxt

Contains the Course Syllabus Description window.

Access: $CARSPATH/modules/regist/progscr/crsent

Tables/Records:

- crs_rec
- crssyll_rec

facact

Contains the Instructor Activity Record window.

Access: $CARSPATH/modules/regist/progscr/crsent

Tables/Records:

- act_table
- facact_rec

facasgn

Contains the Noninstruction Assignment window.

Access: $CARSPATH/modules/regist/progscr/crsent

Tables/Records:

- noninstr_rec
- noninstr_table

facilavl

Contains the Available Facilities window.

Access: $CARSPATH/modules/regist/progscr/crsent

Tables/Records:

- facil_table

facilcrt

Contains the Facility Criteria window.

Access: $CARSPATH/modules/regist/progscr/crsent

Tables/Records:

- dept_table
- facil_table

facility

Contains the Facility Maintenance screen.

Access: $CARSPATH/modules/regist/progscr/crsent

- bldg_table
- dept_table
- div_table
- facil_table

facillst

Contains the Facility Listing window.

Access: $CARSPATH/modules/regist/progscr/crsent

Tables/Records:

- facil_table

facilsch

Contains the Facility Meeting Schedule window.

Access: $CARSPATH/modules/regist/progscr/crsent

Tables/Records:

- im_table
- mtg_rec
- sec_rec

facld

Contains the Instructor Load window.

Access: $CARSPATH/modules/regist/progscr/crsent

Tables/Records:

- flt_table

facqual

Contains the Instructor Qualifications window.

Access: $CARSPATH/modules/regist/progscr/crsent

Tables/Records:

- facqual_rec
- qual_table

facsch

Contains the Instructor Schedule window.

Access: $CARSPATH/modules/regist/progscr/crsent

Tables/Records:

- instr_rec
- mtg_rec
- sec_rec

faculty

Contains the Instructor Information screen.

Access: $CARSPATH/modules/regist/progscr/crsent

Tables/Records:

- bldg_table
- id_rec
- ctry_table
- cty_table
- dept_table
- fac_rec
- facil_table
- profile_rec
- st_table
- title_table
- zip_table

imhrs

Contains the Course Contact Hours window.

Access: $CARSPATH/modules/regist/progscr/crsent

Tables/Records:

- im_rec
- im_table

instruct

Contains the Instruction Assignments window.

Access: $CARSPATH/modules/regist/progscr/crsent

Tables/Records:

- flt_table
- id_rec
- instr_rec

meeting

Contains the Meeting Schedule window.

Access: $CARSPATH/modules/regist/progscr/crsent

Tables/Records:

- bldg_table
- im_table
- mtg_rec
- prd_table

mtgconfl

Contains the Facility Conflicts window.

Access: $CARSPATH/modules/regist/progscr/crsent

Tables/Records:

- mtg_rec
- sec_rec

mtgimhrs

Contains the Course Contact Hours window.

Access: $CARSPATH/modules/regist/progscr/crsent

Tables/Records:

- im_rec
- im_table

noncat
Contains the Transfer Equivalencies window.

Access: $CARSPATH/modules/regist/progscr/crsent

Tables/Records:
- id_rec
- noncateq_rec

roomlkup
Contains the Available Facilities window.

Access: $CARSPATH/modules/regist/progscr/crsent

Tables/Records:
- facil_table

secbrdg
Contains the Section Bridge Record window.

Access: $CARSPATH/modules/regist/progscr/crsent

Tables/Records:
- secbrdg_rec

seccancel
Contains the Cancel Section window.

Access: $CARSPATH/modules/regist/progscr/crsent

Tables/Records: None

seccmnts
Contains the Section Schedule Comments window.

Access: $CARSPATH/modules/regist/progscr/crsent

Tables/Records:
- schd_comment_rec

secimhrs
Contains the Section Contact Hours window.

Access: $CARSPATH/modules/regist/progscr/crsent

Tables/Records:
- im_rec
- im_table

seclist
Contains the Section Listing window.

Access: $CARSPATH/modules/regist/progscr/crsent

Tables/Records:
- sec_rec

secreq
Contains the Registration Requirements window.

Access: $CARSPATH/modules/regist/progscr/crsent

Tables/Records:

- secreq_rec
- secreq_table

section

Contains the Section Record screen.

Access: $CARSPATH/modules/regist/progscr/crsent

Tables/Records:

- altcal_table
- altrfnd_table
- bldg_table
- cat_table
- crs_rec
- deggrp_table
- dept_table
- fac_rec
- flt_table
- grdg_table
- id_rec
- im_table
- instr_rec
- mtg_rec
- prd_table
- prog_table
- sec_rec
- secdtl_rec
- sess_table
- subsess_table

sectxtbk

Contains the Section Textbooks window.

Access: $CARSPATH/modules/regist/progscr/crsent

Tables/Records:

- textbk_rec

sessyrlst

Contains the Session/Year window.

Access: $CARSPATH/modules/regist/progscr/crsent

Tables/Records: None

xmtga

Contains the Add Cross-Listed Meeting window.

Access: $CARSPATH/modules/regist/progscr/crsent

Tables/Records:

- cat_table
- im_table
- mtg_rec
- sec_rec

xmtgalst

Contains the Meeting Records for Cross-Listing window.

Access: $CARSPATH/modules/regist/progscr/crsent

Tables/Records:

- mtg_rec

xsecvlst

Contains the Cross-Listed Sections window.

Access: $CARSPATH/modules/regist/progscr/crsent

Tables/Records:

- sec_rec

SECTION 8 – HISTORY EQUIVALENCY PROGRAM

Overview

Introduction

This section provides reference information about the History Equivalency program (*histeq*). The Course/Class Schedule product uses History Equivalency to track course history and equivalency relationships and create History Equivalency records (histeq_rec) for each course relationship.

The Web Registration application uses the History Equivalency program to analyze the crshist_rec and the crseq_rec, and then provides specific information about two equivalent courses in a History Equivalency record (histeq_rec).

Web Registration uses the generated History Equivalency records (histeq_recs) to determine historical and non-historical course equivalencies in the course registration process.

Program Features Detailed

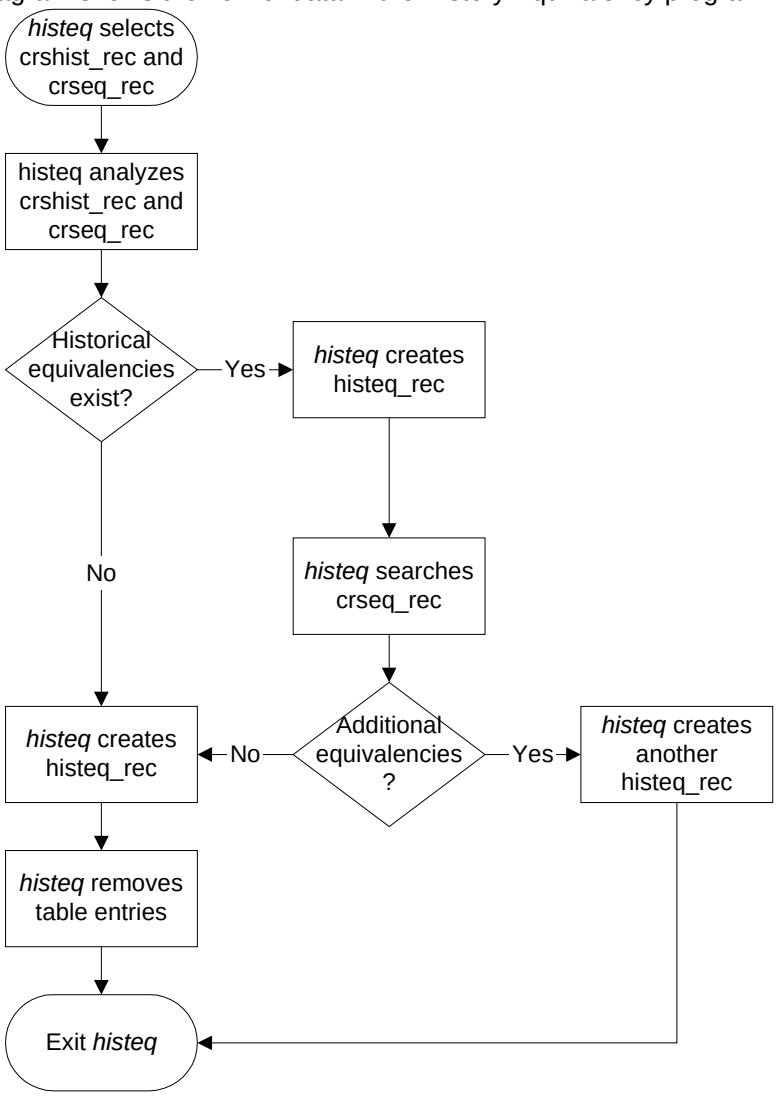
This section contains details about the following features of the History Equivalency program:

- Process flow
- Parameters
- Program screens and windows

Process Flow

Diagram

The following diagram shows the flow of data in the History Equivalency program.



Data Flow Description

The following describes the data flow in the History Equivalency program.

1. The *histeq* program selects all starting course numbers that have an entry either in the crshist_rec or in the crseq_rec.
2. The *histeq* program analyzes crshist_recs and crseq_recs.
3. The *histeq* program proceeds from the starting course number to all links to that course in the crshist_rec, and then it creates a histeq_rec for each historical equivalency.
4. For each historical equivalency the *histeq* program finds, the program searches the crseq_rec to see whether other courses equivalent to the historical course exist. If there is

another course or courses, the *histeq* program adds another histeq_rec for each additional course.

5. Once all historical equivalencies are processed, the *histeq* program reviews the crseq_rec to get all starting course numbers and builds the histeq_rec with all equivalencies to the course.
6. The *histeq* program removes all entries from the histeq_rec table.
7. Exit the *histeq* program.

Program Relationships

No programs use History Equivalency.

Tables and Records Used

The *histeq* program uses the following Common and Course/Class Schedule tables and records.

Note: For information about the Common tables and records, see the *CX System Reference Technical Manual*. For information about the Course/Class Schedule tables and records, see the section *Course/Class Schedule Tables and Records* in this manual.

Common tables and records

The History Equivalency program does not use any common tables and records.

Course/Class Schedule tables and records

- crseq_rec
- crshist_rec
- histeq_rec

Special Function Flags

The History Equivalency program uses no records that have special function flags.

Parameters

Parameters

There are no parameters for running History Equivalency.

Program Screens and Windows

Introduction

History Equivalency has no screens and windows because it is a batch program.

SECTION 9 – ROLL SESSION PROGRAM

Overview

Introduction

This section provides reference information about the Roll Session program (*rollsess*). The Course/Class Schedule product uses Roll Session to duplicate all the required records (i.e., sec_rec, secmtg_rec, mtg_rec, instr_rec, regctgry_rec, secexcept_rec) needed to create a new session.

Program Features Detailed

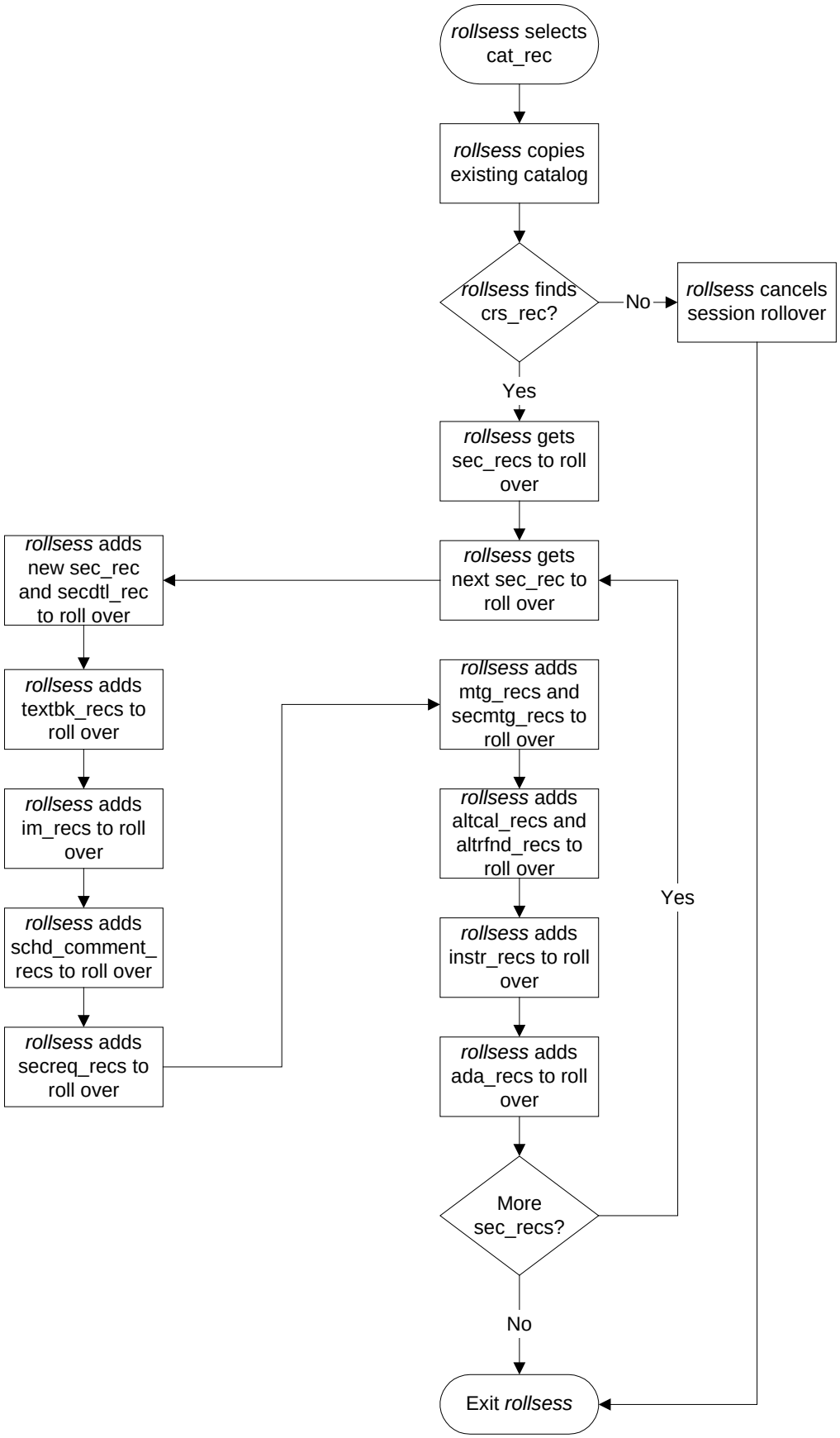
This section contains details about the following features of the Roll Session program:

- Process flow
- Parameters
- Program screens and windows

Process Flow

Diagram

The following diagram shows the flow of data in the Roll Session program.


rollsess selects cat_rec
rollsess copies existing catalog
rollsess finds crs_rec?
No
rollsess cancels session rollover
Yes
rollsess gets sec_recs to roll over
rollsess gets next sec_rec to roll over
rollsess adds new sec_rec and secdtl_rec to roll over
rollsess adds textbk_recs to roll over
rollsess adds im_recs to roll over
rollsess adds schd_comment_ recs to roll over
rollsess adds secreq_recs to roll over
rollsess adds mtg_recs and secmtg_recs to roll over
rollsess adds altcal_recs and altrfnd_recs to roll over
rollsess adds instr_recs to roll over
rollsess adds ada_recs to roll over
More sec_recs?
Yes
No
Exit *rollsess*

Data Flow Description

The following describes the data flow in the Roll Session program.

1. The *rollsess* program selects a cat_rec.
2. The *rollsess* program copies the existing catalog to create a new session for the new catalog.
3. If the *rollsess* program does not find a crs_rec, it cancels the session rollover process.
4. If the *rollsess* program finds a crs_rec and depending upon passed parameters, it creates the following new records to create a new schedule:
   - ada_rec
   - altcal_rec
   - altrfnd_rec
   - im_rec
   - instr_rec
   - mtg_rec
   - regctgry_rec
   - schd_comment_rec
   - sec_rec
   - secdtl_rec
   - secexcept_rec
   - secmtg_rec
   - secreq_rec
   - textbk_rec
5. If the *rollsess* program finds more sec_recs, it gets the next sec_rec to rollover and creates related records to create a new schedule.
6. If the *rollsess* program does not find more sec_recs, the *rollsess* program ends.

Program Relationships

The Roll Session program uses the following libraries.

- *Libcrs*
- *Libreg*

Tables and Records Used

The *rollsess* program uses the following Common and Course/Class Schedule tables and records.

Note: For information about the Common tables and records, see the *CX System Reference Technical Manual*. For information about the Course/Class Schedule tables and records, see the section *Course/Class Schedule Tables and Records* in this manual.

Common tables and records

The Roll Session program does not use any common tables and records.

Course/Class Schedule tables and records

- acad_cal_rec
- altcal_rec
- altrfnd_rec
- crs_rec
- crsdtl_rec
- im_rec
- instr_rec
- mtg_rec

- regctgry_rec
- schd_comment_rec
- sec_rec
- secdtl_rec
- secmtg_rec
- secreq_rec
- textbk_rec

Special Function Flags

The Roll Session program uses no records that have special function flags.

Parameters

Introduction

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

Note: You also can specify compilation values with the includes for the Course/Class Schedule product that affect the Roll Session program.

Parameter Syntax

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

The following is the correct usage for running the Roll Session program from the UNIX shell:

rollsess -y oldyr -s oldsess -c oldcat -Y newyr -S newsess [-C newcat] [-L site] [-p prog] [-a] [-n] [-m] [-i] [-d] [-r] [-x]

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

Parameters

The following lists the parameters for running Roll Session.

-y *oldyr*
Required - Specifies the year (yyyy) from which to roll the session.

Example: -y 2000

-s *oldsess*
Required - Specifies the code for the session from which to roll the session.

Example: -s FA

-c *oldcat*
Required - Specifies the code for the catalog from which to roll the session.

Example: -c UG

-Y *newyr*
Required - Specifies the year to which the system should roll the session.

Example: -Y 2001

-S *newsess*
Required - Specifies the session to which the system should roll the session.

Example: -S FA

-C *newcat*
Optional - Specifies the catalog to which the system should roll the session.

Example: -C UG

-L *site*
Optional - Specifies the default site.

Example: -L main

-p *prog*

Optional - Specifies code for the program of the session rollover.

Example: -p UNDG

-a

Optional - Specifies to include nonstandard sections in the rollover.

-n

Optional - Specifies to exclude canceled sections from the rollover.

-m

Optional - Specifies to exclude Meeting (mtg_rec), Instruction Assignment (instr_rec), Alternate Refund (altrfnd_rec), and Alternate Calendar (altcal_rec) records from the rollover.

-i

Optional - Specifies to exclude Instruction Assignment records from the rollover.

-d

Optional - Specifies to exclude Schedule Comment records from the rollover.

-r

Optional - Specifies to exclude Section Requirements records from the rollover.

-x

Optional - Specifies to exclude Text records (text_rec) from the rollover.

Program Screens and Windows

Introduction

Roll Session has no screens and windows because it is a batch program.

SECTION 10 – SET REFERENCE NUMBER PROGRAM

Overview

Introduction

This section provides reference information about the Set Reference Number program (*setrefno*). The Reference Number (*refno*) program allows you to insert a reference number in the Ref Number field of the Section Record screen for use in registration, grade scanning, and Interactive Voice Response (IVR). You can define the reference number with a beginning and ending number, and a "skip" number that indicates the size of the interval to skip between each number when the system assigns reference numbers.

You do not have to use reference numbers for the registration process, but you must use reference numbers for grade scanning and IVR.

Special Notes

- If you use the *refno* program to add a code in the Ref Number field, leave the Ref Number field blank in the Section record. The *refno* program does not overwrite sections containing reference numbers that have been manually inserted.
- To remove reference numbers that you created prior to the session schedule, you must prepare an appropriate SQL statement.

Program Features Detailed

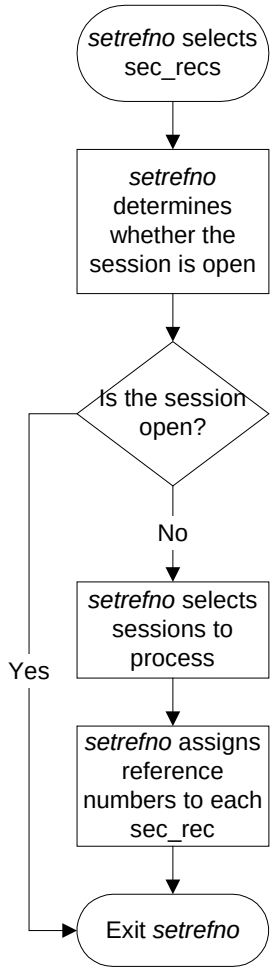
This section contains details about the following features of the Set Reference Number program:

- Process flow
- Parameters
- Program screens and windows

Process Flow

Diagram

The following diagram shows the flow of data in the Set Reference Number program.



Data Flow Description

The following describes the data flow in the Set Reference Number program.

1. The *setrefno* program selects sec_recs for a specified session.
2. The *setrefno* program determines whether the session is open. If the session is open, setrefno closes. If the session is not open, it selects sessions to process.
3. The *setrefno* program assigns reference numbers to each sec_rec.
4. Exit the *setrefno* program.

Program Relationships

No programs use Set Reference Number.

Tables and Records Used

The *setrefno* program uses the following Common and Course/Class Schedule tables and records.

Note: For information about the Common tables and records, see the *CX System Reference Technical Manual*. For information about the Course/Class Schedule tables and records, see the section *Course/Class Schedule Tables and Records* in this manual.

Common tables and records

The Set Reference Number program does not use any common tables and records.

Course/Class Schedule tables and records

- acad_cal_rec
- crs_rec
- sec_rec

Special Function Flags

The Set Reference Number program uses no records that have special function flags.

Parameters

Introduction

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

Note: You also can specify compilation values with the includes for the Course/Class Schedule product that affect the Set Reference Number program.

Parameter Syntax

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

The following is the correct usage for running the Set Reference Numbers program from the UNIX shell:

setrefno -s session -y year -p prog -L site -b begin -e end -k skip

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

Parameters

The following lists the parameters for running Set Reference Number.

-s *session*
Required – Specifies the session to process.

Example: setrefno –s FA

-y *year*
Required – Specifies the year to process.

Example: setrefno –y 2000

-p *prog*
Required – Specifies the program to process.

Example: setrefno –p UNDG

-L *site*
Required – Specifies the site to process.

Example: setrefno –L main

-b *begin*
Required – Specifies the beginning reference number to use.

Example: setrefno –b 001

-e *end*
Required – Specifies the ending reference number to use.

Example: setrefno –e 100

-k *skip*
Required – Specifies the size of the skip interval for assigning reference numbers.

Example: setrefno –k 005

Program Screens and Windows

Introduction

Set Reference Number has no screens and windows because it is a batch program.

SECTION 11 – MENUS, SCREENS, SCRIPTS, AND REPORTS

Overview

Introduction

This section provides reference information on the following features of the Course/Class Schedule product:

- Menu source files
- Menu option files
- PERFORM screens
- SQL scripts
- Csh scripts
- ACE reports
- Letters

Directory Locations

The features detailed in this section are located in the following directory paths:

Menu source files

$CARSPATH/menusrc/student/regist/schedule
$CARSPATH/menusrc/student/regist/schedule/reports
$CARSPATH/menusrc/student/regist/schedule/schdprint

Menu option files

$CARSPATH/menuopt/regist/informers
$CARSPATH/menuopt/regist/programs
$CARSPATH/menuopt/regist/reports
$CARSPATH/menuopt/stubill/informers

SQL scripts

$CARSPATH/modules/regist/informers
$CARSPATH/modules/stubill/informers

ACE reports

$CARSPATH/modules/regist/reports

Menu Structure

Introduction

CX menus provide access to a functionally related group of menu options. For example, a CX user in the Admissions office can access all the options necessary for processing prospects, recruits, and applicants from his/her menu, but typically cannot access options for processing accounting information. Depending on the work responsibilities of the CX users at your institution, you can customize their menus to offer access to as many or as few menu options as desired.

The selections that appear on the CX menus at your institution are controlled by a variety of interrelated directories and files. This section explains how the directories and files define CX menus, and it also provides information about the standard CX menu options.

Directories and Files that Define Menu Structures

The directories and files that control the standard CX menus are:

- Menu source directories
- Menu description files
- Menu option files

Menu Source Directories

Menu source (menusrc) directories define the branches of menus and submenus that offer access to CX features. The highest level of the menu source directory, located at $CARSPATH/menusrc, contains files and subdirectories similar to the following:

```
admit/                  student/
fiscal/                 system/
instdev/                utility/
menudesc
```

Note: In this example, six subdirectories (designated with a slash [/] after their names) and one file appear in the menusrc directory.

The Menu Description File: Example 1

Menusrc directories always contain a menu description (menudesc) file. The menudesc file defines the options available at the current menu level. To continue the example above, the menudesc file in the $CARSPATH/menusrc directory contains the following type of information:

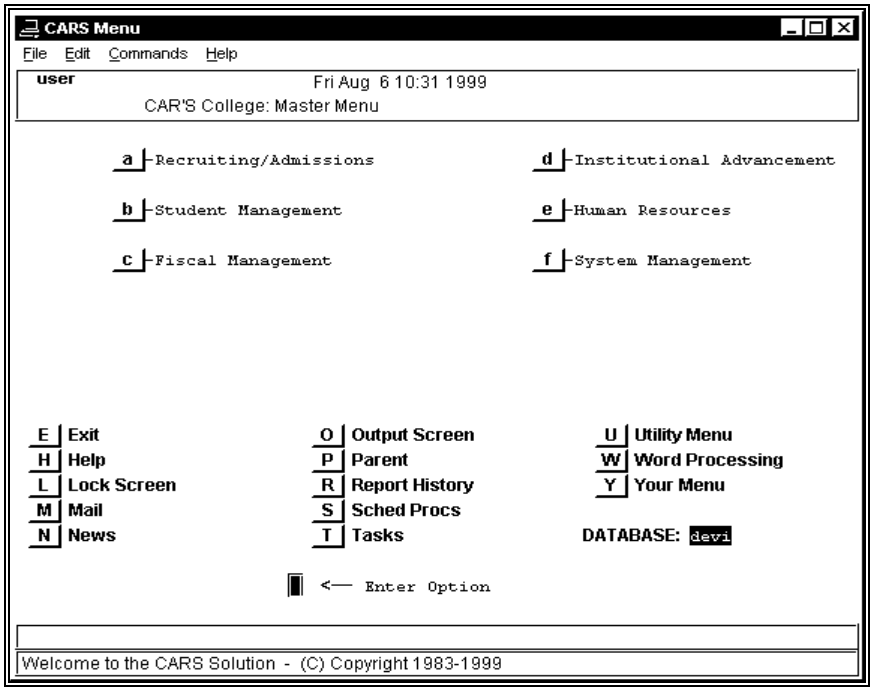
```
#
#Revision Information (Automatically maintained by 'make' - DON'T CHANGE)
#-------------------------------------------------------------------------
#$Header:
#$Log:
#
#-------------------------------------------------------------------------
#
#
TI=INST_NAME: Master Menu
SD=CARS Solution Master Menu
LD=Master Menu For INST_NAME
m4_keepif(ENABLE_MOD_ADMISSIONS, `Y')
MNU_SUB(admit)
m4_keepend
MNU_SUB(student)
MNU_SUB(fiscal)
m4_keepif(ENABLE_MOD_DEVELOP, `Y')
MNU_SUB(instdev)
m4_keepend
MNU_SUB(system)
```

Interpreting the menudesc File in Example 1

In this example, the menudesc file has the following information:

- Comment lines (lines beginning with a pound sign [#])
- Title and descriptive information (lines beginning with TI [title], SD [short description], or LD [long description])
- Keepif lines (lines indicating that a particular submenu is available if a corresponding ENABLE macro is set to Y)
- Keepend lines (lines indicating the end of a macro-controlled option)
- MNU_SUB lines (lines showing the name of the submenu)

Specifically, this example defines a menu called <institution name>: Master Menu. Assuming your institution has purchased the Admissions and Development applications and has enabled the ENABLE_MOD_ADMISSIONS and ENABLE_MOD_DEVELOP macros, your Master Menu would contain the following menu options:



Note: The names used for each of the submenus are controlled in their respective submenus.

Subdirectories in the menusrc Directory

Typically, the menusrc directory for a particular menu also contains subdirectories that relate to that menu's submenu options. In the example of the Master Menu, the following subdirectories are in the main $CARSPATH/menusrc directory, and they relate directly to the menu options that appear on the preceding menu screen example.

admit/
Recruiting/Admissions submenu

fiscal/
Fiscal Management submenu

instdev/
Institutional Advancement submenu

student/
Student Management submenu

system/
System Management submenu

utility/
Utility submenu (accessed from the lower part of the CX menu screen)

Contents of menusrc Subdirectories

The menusrc subdirectories have the same components as the main menusrc directory used in the preceding examples. Each contains subdirectories for more submenus, as well as a menudesc file that defines the appearance and contents of the submenu itself.

Navigating the menusrc Subdirectories

By using UNIX commands to move into any of the menusrc subdirectories and viewing its contents (i.e., its own subdirectories, if any, and its related menudesc file), you can view and map the menus, submenus, and menu options available at your institution. To continue the preceding example, if you enter **cd utility** at the UNIX prompt while in the $CARSPATH/menusrc directory, you can then enter **lsf** to display the contents of the utility subdirectory, as in the following:

```
Makefile            data/            iq/              ltrlblrep/       spooler/
RCS/                document/        login/           menudesc
adr/                files/           ltrlbl/          sbscr/
```

If you then enter **cd data** at the UNIX prompt and execute the **lsf** command again, you display the contents of the data subdirectory, as in the following:

```
Makefile            RCS              menudesc
```

In this example, only a menudesc file appears, indicating that no submenus exist at the $CARSPATH/menusrc/utility/data menu source level. The menudesc file in this case contains the actual menu options you can run from the Utilities/Data Entry menu. If desired, you can view the menudesc file using *vi*, *more*, or other UNIX commands.

The Menu Description File: Example 2

The menudesc file in Example 1 contains references only to submenus, that is, each line in the menudesc file that begins with MNU_SUB refers to a submenu. Another type of line can appear in the menudesc file: the MNU_OPT line. MNU_OPT lines do not refer to submenus; instead, they provide the link from the menu to the actual process (e.g., a program, script, report, or screen) you can run. For example, the menudesc file in the $CARSPATH/menusrc/utility/data directory contains the following type of information:

```
#
#Revision Information (Automatically maintained by 'make' - DON'T CHANGE)
#-------------------------------------------------------------------------
#$Header: menudesc,v 8.1 97/10/03 15:34:36 jdoe Released $
#-------------------------------------------------------------------------
#
TI=Utilities: Data Entry Menu
SD=Data Entry
LD=Data Entry
PW=@STD
MNU_OPT(common/programs/ide)
```

Interpreting the menudesc File in Example 2

In this example, the menudesc file has the following information:

- Comment lines (lines beginning with a pound sign [#])
- Title and descriptive information (lines beginning with TI [title], SD [short description], or LD [long description])
- Password information (the line beginning with PW)
- MNU_OPT line (the line showing the location of the menu option itself)

Notes:

- As with the menudesc file in Example 1, this file also can contain Keepif and Keepend lines to control the availability of optional processes.
- Although the menudesc file in Example 2 contains only one menu option (common/programs/ide), menudesc files can contain as many options as you care to display on a single menu.

Menu Option Files

The menudesc file for a particular menu or submenu points to the specific menu options you can run (e.g., Example 2 referred to the single menu option common/programs/ide, indicating that a common program is the only option available from the selected submenu).

To locate the menu option files for a particular menu, use the following command syntax:

cd menuopt/<directory on menudesc line>

For Example 2, the appropriate command is:

cd menuopt/common/programs

When you list the contents of the directory, the list resembles the following:

```
Makefile            ctcbatchr        ide              tuserid
RCS/                evnte            ide,fa
adrtest             fo.op            perm.a
ctcbatch            fo.stu           perm.c
```

This directory example contains several menu option files that are linked to other menudesc files. By their names, you usually can determine the process the menu option executes. For example, ide and ide.fa run versions of *identry*, and fo.op and fo.stu run versions of *forment*. More than

one version of these processes exist because, depending on the needs of each menu user, you may want to use different titles or parameters.

You can view the specific menu option file using *vi*, *more*, or other UNIX commands.

Interpreting the Menu Option File

The menu option file contains three primary types of information:

- A definition of the parameter screen that appears when a user selects the option from the menu, including information for comment lines
- The name of the screen, program, script, or report the menu option executes
- The parameters, if any, that determine how the menu option should execute

An excerpt from the ide file demonstrates these three types of information. The horizontal lines on the excerpt divide the types of information and have been added to help you interpret the file's contents.

```
------------------------------- Parameter screen ---------------------------------------
}
screen
{
                    m4_center_clipped(ID DATA ENTRY,40)
                    m4_center_clipped(PP_OFFICE[PA5], 40)
                    m4_center_clipped(PP_TICK[PA7], 40)
}
end
attributes
SD: optional,
    default = "ID Data Entry";
PW: optional,
    default = "@STD";
RD1: optional,
    default = "`Functions:'                          ";
RD2: optional,
    default = "";
RD3: optional,
    default = "`Enter ID Records for Individuals'    ";
RD4: optional,
    default = "Enter ID Records for Non-individuals";
PR: optional,
----------------------------------Executed Process-------------------------------------
    default = "BIN_PATH/identry";
--------------------------------Processing Parameters--------------------------------
m4_keepif(ENABLE_FEAT_AUTOMODE, `Y')
PA1: optional,
    default = "-a";
m4_keepelse
m4_keepif(ENABLE_FEAT_FORCEQUERY, `Y')
PA1: optional,
    default = "-F";
m4_keepend
m4_keepend
PA2: optional,
    default = "-D";
PA3: optional,
    default = "3";
PA4: optional,
    default = "-o";
LU5 = ofc_table.txt, optional;
PA5: optional,
    comments = "COMMENT_OFC_TBCODE  COMMENT_TBL",
    length = 4,
    lookup LU5 joining *ofc_table.ofc,
    upshift;
PA6: optional,
    default = "-T";
LU7 = tick_table.txt, optional;
PA7: optional,
    comments = "COMMENT_TICK  COMMENT_TBL",
    length = 4,
    lookup LU7 joining *tick_table.tick,
    upshift;
end
```

Determining the Type of Executed Process

In the preceding example, the location of the menu option ($CARSPATH/menuopt/common/programs) indicates that the menu option executes a program. The reference to BIN_PATH within the menu option file also indicates the execution of a program. However, many menu options execute other types of processes (e.g., screens, scripts, or reports).

The following list shows the type of process executed, the _PATH reference from the menu option file, and the type of menu option directory path in which the process resides.

Process type	_PATH reference	Menu option directory path
Program	BIN_PATH	/programs
C shell script	SCP_PATH	/scripts
Report	ARC_PATH OTH_PATH	/reports /others
SQL statement	INF_PATH	/informers
PERFORM screen	SCR_PATH FRM_PATH	/screens
Utility	UTL_PATH	/utilities

Summary of Menu Source, Menu Description, and Menu Option Information

When you need to modify any menu options in use at your institution, remember the following:

- Menu source directories reflect your CX menus and submenus.
- Menu source directories contain other menu source subdirectories and menu description files.
- Menu description files define the menu options that appear on each menu.
- Menu description files contain lists of the menu options and submenus.
- Menu option files contain the instructions needed to execute each CX process on every menu.

Menu Options

Programs, Screens, and Scripts

The following list associates each Course/Class Schedule program, screen, and script menu option and corresponding menuopt file and identifies the menuopt locations and what the menu option accesses.

Note: The following menus and options are listed in the order in which they appear on the standard CX menu structure. Italicized parameters indicate those that a user can enter or change.

Registrar: Course/Class Schedule Menu

Catalog Maintenance

Accesses: $CARSPATH/src/regist/crsent (Program)

File: $CARSPATH/menuopt/regist/programs/crsent

Parameters Passed:

- -L *site* (Indicates the site in which the processes are run)
- -y *sessyr list* (Indicates the session and year parameter list)
- -Y (Allows access to all session/year entries in the acad_cal_rec)
- -g (Enables the meeting detail information from the section screen)
- -R (Enable conflict checking with Resource25 data?)
- -v (Enable creation of R25 vcal files on meeting update?)

Print Course Catalog

Accesses: $CARSPATH/modules/regist/reports/catalog (ACE report)

File: $CARSPATH/menuopt/regist/reports/catalog

Parameters Passed:

- *Catalog*

District Catalog

Note: This menu option is available if the macro ENABLE_CA_CC is set to Y.

Accesses: $CARSPATH/modules/regist/reports/distcat (ACE report)

File: $CARSPATH/menuopt/regist/reports/distcat

Parameters Passed:

- *Catalog*
- *Division*
- *Department*
- *Course Status*

Course History Report

Accesses: $CARSPATH/modules/regist/reports/crshist (ACE report)

File: $CARSPATH/menuopt/regist/reports/crshist

Parameters Passed:

- *Beginning Course Number*
- *Ending Course Number*

- *Beginning Catalog*
- *Ending Catalog*

Create Hist/Eq Records

Note: This menu option is available if the macro ENABLE_MOD_WEB_REGIST is set to Y.

Accesses: $CARSPATH/src/regist/histeq (Program)

File: $CARSPATH/menuopt/regist/programs/heq

Parameters Passed: None

History/Equiv Report

Note: This menu option is available if the macro ENABLE_MOD_WEB_REGIST is set to Y.

Accesses: $CARSPATH/modules/regist/reports/histeq (ACE report)

File: $CARSPATH/menuopt/regist/reports/histeq

Parameters Passed:
- *Course Number*

Transfer Equiv by Inst

Accesses: $CARSPATH/modules/regist/reports/trnsfreq (ACE report)

File: $CARSPATH/menuopt/regist/reports/trnsfreq

Parameters Passed:
- ID#

Instructional Method Load

Note: This menu option is available if the macro ENABLE_FEAT_FACLOAD is set to Y.

Accesses: $CARSPATH/modules/regist/reports/imverify (ACE report)

File: $CARSPATH/menuopt/regist/reports/imverify

Parameters Passed:
- *Session*
- *Academic Year*
- *Catalog*
- *Division*
- *Department*

Faculty Load Sheet

Note: This menu option is available if the macros ENABLE_FEAT_CRSDTL and ENABLE_FEAT_CRSDTL_CA are set to Y.

Accesses: $CARSPATH/modules/regist/reports/facsheet (ACE report)

File: $CARSPATH/menuopt/regist/reports/facsheet

Parameters Passed:
- *Session*
- *Academic Year*

- *Division*
- *Department*

Cancel Sections

Accesses: $CARSPATH/src/regist/cnclsec (Program)

File: $CARSPATH/menuopt/regist/programs/cnclsec

Parameters Passed:

- -S *sess* (Indicates the session of the section to cancel)
- -y *year* (Indicates the year of the section to cancel)
- -c *cat* (Indicates the catalog of the section to cancel)
- -C *crs_no* (Indicates the course number of the course to cancel)
- -s *sec_no* (Indicates the section number of the section to cancel)
- -r *ctc_resrc* (Indicates the contact resource for the ctc_rec created)
- -t *tick_code* (Indicates the Tickler code to use)
- -d *ctc_due_date* (Indicates the date when the contact is due)

Create Reference Numbers

Note: This menu option is available if the macro ENABLE_REGIST_SET_REFNO is set to Y.

Accesses: $CARSPATH/src/regist/setrefno (Program)

File: $CARSPATH/menuopt/regist/programs/setrefno

Parameters Passed:

- -L *site* (Site value)
- -p *prog* (Academic program)
- -s *session* (Academic session to process)
- -y *year* (Year)
- -b *begin* (Beginning reference number to use)
- -e *end* (Ending reference number to use)
- -k *skip* (Size of skip interval for assigning reference numbers)

Update Registration Record

Accesses: $CARSPATH/modules/stubill/informers/updreg (SQL script)

File: $CARSPATH/menuopt/stubill/informers/updreg

Parameters Passed:

- *Session*
- *Academic Year*

Catalog Rollover

Accesses: $CARSPATH/modules/regist/informers/rollcat (SQL script)

File: $CARSPATH/menuopt/regist/informers/rollcat

Parameters Passed:

- *Old Catalog*
- *New Catalog*

Session Rollover

Accesses: $CARSPATH/src/regist/rollsess (Program)

File: $CARSPATH/menuopt/regist/programs/rollsess

Parameters Passed:

- -y *oldyr* (Old year)
- -s *oldsess* (Old session)
- -c *oldcat* (Old catalog)
- -Y *newyr* (New year)
- -S *newsess* (New session)
- -C *newcat* (New catalog)
- -L *site* (Site value)
- -p *prog* (Program)

Resource25 Interface

Notes:

- This submenu is available if the macro ENABLE_FEAT_R25 is set to Y.
- See the CX technical manual *Resource25/Schedule25 Interfaces Technical Manual* for additional information about this submenu.

Schedule25 Interface

Notes:

- This submenu is available if the macro ENABLE_FEAT_S25 is set to Y.
- See the CX technical manual *Resource25/Schedule25 Interfaces Technical Manual* for additional information about this submenu.

Registrar: Print Course/Class Schedules Menu

Print Division Schedule

Accesses: $CARSPATH/modules/regist/reports/schddiv (ACE report)

File: $CARSPATH/menuopt/regist/reports/schddiv

Parameters Passed:

- *Session*
- *Academic Year*
- *Catalog*

Print Department Schedule

Accesses: $CARSPATH/modules/regist/reports/schddep (ACE report)

File: $CARSPATH/menuopt/regist/reports/schddep

Parameters Passed:

- *Session*
- *Academic Year*
- *Catalog*

Print Faculty Schedule

Accesses: $CARSPATH/modules/regist/reports/schdfac (ACE report)

File: $CARSPATH/menuopt/regist/reports/schdfac

Parameters Passed:

- *Session*
- *Academic Year*

- *Catalog*

Print Conflict Schedule

Accesses: $CARSPATH/modules/regist/reports/schdflict (ACE report)

File: $CARSPATH/menuopt/regist/reports/schdflict

Parameters Passed:
- *Session*
- *Academic Year*
- *Catalog*

Print Time Schedule

Accesses: $CARSPATH/modules/regist/reports/schdtime (ACE report)

File: $CARSPATH/menuopt/regist/reports/schdtime

Parameters Passed:
- *Session*
- *Academic Year*
- *Catalog*

Registrar: Course/Class Reports Menu

Abbrev Schedule

Accesses: $CARSPATH/modules/regist/reports/schdabbr (ACE report)

File: $CARSPATH/menuopt/regist/reports/schdabbr

Parameters Passed:
- *Session*
- *Academic Year*
- *Catalog*

Cross Listed Sections

Accesses: $CARSPATH/modules/regist/reports/crosslist (ACE report)

File: $CARSPATH/menuopt/regist/reports/crosslist

Parameters Passed:
- *Session*
- *Academic Year*
- *Subsession*
- *Catalog*
- *Department*
- *Division*
- *ID#*
- *Campus*

Articulation

Note: This menu option is available if the macro ENABLE_FEAT_ARTIC is set to Y.

Accesses: $CARSPATH/modules/regist/reports/crsartic (ACE report)

File: $CARSPATH/menuopt/regist/reports/crsartic

Parameters Passed:
- *Catalog*

- *Division*
- *Department*
- *Course Status*

Course/Instructor

Accesses: $CARSPATH/modules/regist/reports/crsqual (ACE report)

File: $CARSPATH/menuopt/regist/reports/crsqual

Parameters Passed:
- *Session*
- *Academic Year*
- *Catalog*
- *Course Number*
- *Division*
- *Department*
- *ID#*

Course Overview

Accesses: $CARSPATH/modules/regist/reports/crsover (ACE report)

File: $CARSPATH/menuopt/regist/reports/crsover

Parameters Passed:
- *Session*
- *Academic Year*
- *Catalog*
- *Division*
- *Department*
- *Course Status*

Course Requisites

Accesses: $CARSPATH/modules/regist/reports/crsreq (ACE report)

File: $CARSPATH/menuopt/regist/reports/crsreq

Parameters Passed:
- *Catalog*
- *Department*

Room Allocation

Accesses: $CARSPATH/modules/regist/reports/roomalloc (ACE report)

File: $CARSPATH/menuopt/regist/reports/roomalloc

Parameters Passed:
- *Session*
- *Academic Year*
- *Catalog*
- *Division*
- *Department*
- *Course Status*

Room Usage

Accesses: $CARSPATH/modules/regist/reports/roomuse (ACE report)

File: $CARSPATH/menuopt/regist/reports/roomuse

Parameters Passed:

- *Session*
- *Academic Year*
- *Building*
- *Room*
- *Beginning Time*
- *Ending Time*
- *Minutes*

Room Usage - Weekend

Accesses: $CARSPATH/modules/regist/reports/roomuses (ACE report)

File: $CARSPATH/menuopt/regist/reports/roomuses

Parameters Passed:

- *Session*
- *Academic Year*
- *Building*
- *Room*
- *Beginning Time*
- *Ending Time*
- *Minutes*

Textbook

Accesses: $CARSPATH/modules/regist/reports/textbk (ACE report)

File: $CARSPATH/menuopt/regist/reports/textbk

Parameters Passed:

- *Session*
- *Academic Year*
- *Catalog*

PERFORM (Table Maintenance) Screens

Introduction

Course/Class Schedule uses no PERFORM screens for displaying maintenance tables and some records.

SQL Scripts

Introduction

The Course/Class Schedule product contains SQL scripts that perform queries and produce reports from database records. The scripts are located in the following directory path: $CARSPATH/modules/regist/informers

Note: Csh scripts can call ACE reports and SQL scripts. Such ACE reports and SQL scripts do not reside on the CX menu system.

SQL Scripts

The following lists the SQL scripts provided with Course/Class Schedule.

rollcat

Initiates a process that enables you to create a new catalog from a previous catalog.

Menu Access: Registrar: Course/Class Schedule menu: Catalog Rollover menu option.

- artic_rec
- crs_rec
- crsabstr_rec
- crsdtl_rec
- crsobj_rec
- crsqual_rec
- crsrequir_rec
- crssyll_rec
- im_rec
- secreq_rec

updreg

Initiates a process that updates the tuit_code, fee_code, and bill_code in reg_recs where the values in those fields are not equivalent to the values in the corresponding fields in the crs_rec or sec_rec.

Note: Values in the sec_rec, if present, always override those in the crs_rec. This informer must be run when changes are made to the tuit_code, fee_code, or bill_code in a crs_rec or sec_rec after students have registered.

Menu Access: Registrar: Course/Class Schedule menu: Update Registration Record menu option

Tables/Records Used:

- crs_rec
- cw_rec
- reg_rec
- sec_rec

Csh Scripts

Introduction

Course/Class Schedule contains no Csh scripts to automate the processing of information.

ACE Reports

Introduction

CX contains ACE reports for easy reporting of Course/Class Schedule database information. The ACE reports are grouped in the following categories:

- Miscellaneous Course/Class Schedule reports
- Print Course/Class Schedules reports
- Course/Class reports

Miscellaneous Course/Class Schedule Reports

The following lists the ACE reports accessed from the Registrar: Course/Class Schedule menu. The reports are listed by the report titles, which appear at the beginning of the printed reports. In some instances, the report title is the same as the menu option you use to produce the report. Some reports listed do not appear on the CX menu system because they are used only in Csh scripts.

Course Catalog

Provides the entire listing of courses in a specified course catalog.

Menu Access: Registrar: Course/Class Schedule menu: Print Course Catalog menu option.

File: $CARSPATH/modules/regist/reports/catalog

Course History and Equivalency Relationships Report

Provides historical and non-historical course equivalencies based on the histeq_rec, which is program-generated.

Menu Access: Registrar: Course/Class Schedule menu: History/Equiv Report menu option.

File: $CARSPATH/modules/regist/reports/histeq

Course History Report

Provides a list, in course number order, the changes a course name has undergone across catalogs. The system sorts the list by course number.

Menu Access: Registrar: Course/Class Schedule menu: Course History Report menu option.

File: $CARSPATH/modules/regist/reports/crshist

District Course Identification Report

Provides one biographical form for the id_no specified.

Menu Access: Registrar: Course/Class Schedule menu: District Catalog menu option.

File: $CARSPATH/modules/regist/reports/distcat

Faculty Load Sheet

Provides wide output of the course schedule by division, department, course, section, and meeting.

Menu Access: Registrar: Course/Class Schedule menu: Faculty Load Sheet menu option.

File: $CARSPATH/modules/regist/reports/facsheet

Instructional Method Verification Report

Provides verification of instructional methods.

Menu Access: Registrar: Course/Class Schedule menu: Instructional Method Load menu option.

File: $CARSPATH/modules/regist/reports/imverify

Transfer Course Equivalency Report

Provides a list of courses from a transfer institution that have been equated to the host institution's courses and catalogs.

Menu Access: Registrar: Course/Class Schedule menu: Transfer Equiv by Inst menu option.

File: $CARSPATH/modules/regist/reports/trnsfreq

Print Course/Class Schedule Reports

The following lists the ACE reports accessed from the Registrar: Print Course/Class Schedules menu. The reports are listed by the report titles, which appear at the beginning of the printed reports. In some instances, the report title is the same as the menu option you use to produce the report. Some reports listed do not appear on the CX menu system because they are used only in Csh scripts.

Course Conflict Schedule

Provides a list of conflicts within course schedules.

Menu Access: Registrar: Print Course/Class Schedules menu: Print Conflict Schedule menu option.

File: $CARSPATH/modules/regist/reports/schdflict

Course Schedule by Department

Provides wide output of the course schedule by department, course, section, meeting and instructor.

Note: The system ignores the sec_rec.fac_id if Instructor records for the section and meeting exist. If Instructor records do not exist, the system uses the sec.fac_id as the instructor for the meeting. If no sec.fac_id exists, the system uses "STAFF" as the instructor for the meeting.

Menu Access: Registrar: Print Course/Class Schedules menu: Print Department Schedule menu option.

File: $CARSPATH/modules/regist/reports/schddep

Course Schedule by Division

Provides wide output of the course schedule by division, department, course, section, meeting, and instructor.

Note: The system ignores the sec_rec.fac_id if Instructor records for the section and meeting exist. If Instructor records do not exist, the system uses the sec.fac_id as the instructor for the meeting. If no sec.fac_id exists, the system uses "STAFF" as the instructor for the meeting.

Menu Access: Registrar: Print Course/Class Schedules menu: Print Division Schedule menu option.

File: $CARSPATH/modules/regist/reports/schddiv

Course Schedule by Faculty and Meeting Times

Provides the course schedule by faculty and meeting times. The section status prints if maximum registration information is requested. Sections without Meeting records will print

with “TO BE ANNOUNCED” in the time column. This report requires paper 132 columns wide.

Menu Access: Registrar: Print Course/Class Schedules menu: Print Faculty Schedule menu option.

File: $CARSPATH/modules/regist/reports/schdfac

Course Schedule by Meeting Time and Days

Provides output identical to schddep, except the sort criteria are off the meeting time, then the meeting days (in descending order), and then on the meeting location before on course. This is a wide output report (132 columns).

Menu Access: Registrar: Print Course/Class Schedules menu: Print Time Schedule menu option.

File: $CARSPATH/modules/regist/reports/schdtime

Course/Class Reports

The following lists the ACE reports accessed from the Registrar: Course/Class Reports menu. The reports are listed by the report titles, which appear at the beginning of the printed reports. In some instances, the report title is the same as the menu option you use to produce the report. Some reports listed do not appear on the CX menu system because they are used only in Csh scripts.

Articulation Report

Provides one biographical form for the id_no specified.

Menu Access: Registrar: Print Course/Class Reports menu: Articulation menu option.

File: $CARSPATH/modules/regist/reports/crsartic

Course/Instructor Qualifications Report

Provides one biographical form for the id_no specified.

Menu Access: Registrar: Print Course/Class Reports menu: Course/Instructor menu option.

File: $CARSPATH/modules/regist/reports/crsqual

Course Overview Sheet

Provides one biographical form for the id_no specified.

Menu Access: Registrar: Print Course/Class Reports menu: Course Overview menu option.

File: $CARSPATH/modules/regist/reports/crsover

Course Requisite Report

Provides a list of courses and their prerequisite, corequisite, and concurrent requisite courses.

Menu Access: Registrar: Print Course/Class Reports menu: Course Requisites menu option.

File: $CARSPATH/modules/regist/reports/crsreq

Cross Listed Course Report

Provides a list of cross-listed courses.

Menu Access: Registrar: Print Course/Class Reports menu: Cross Listed Sections menu option.

File: $CARSPATH/modules/regist/reports/crosslist

Room Allocation Verification Report

Provides a room allocation verification report.

Menu Access: Registrar: Print Course/Class Reports menu: Room Allocation menu option.

File: $CARSPATH/modules/regist/reports/roomalloc

Room Usage Chart

Provides room usage information for weekdays.

Menu Access: Registrar: Print Course/Class Reports menu: Room Usage menu option.

File: $CARSPATH/modules/regist/reports/roomuse

Room Usage Chart for Weekend Days

Provides room usage information for Saturday and Sunday.

Menu Access: Registrar: Print Course/Class Reports menu: Room Usage - Weekend menu option.

File: $CARSPATH/modules/regist/reports/roomuses

Short Course Schedule By Division

Provides narrow output of the course schedule by division, department, course, section, meeting, and instructor.

Note: Note: The system ignores the sec_rec.fac_id if Instructor records for the section and meeting exist. If Instructor records do not exist, the system uses the sec.fac_id as the instructor for the meeting. If no sec.fac_id exists, the system uses “STAFF” as the instructor for the meeting.

Menu Access: Registrar: Print Course/Class Reports menu: Abbrev Schedule menu option.

File: $CARSPATH/modules/regist/reports/schdabbr

Textbook Table Report

Provides a list of textbooks used in courses and sections. The report is valid only if using the schedtext PERFORM screen to add textbooks.

Menu Access: Registrar: Print Course/Class Reports menu: Textbook menu option.

File: $CARSPATH/modules/regist/reports/textbk

SECTION 12 – CUSTOMIZING THE COURSE/CLASS SCHEDULE PROCESSES

Overview

Introduction

This section provides procedures for setting up and installing the features of the Course/Class Schedule product. It includes procedures for the following processes:

- Assessing institutional needs for the module
- Reviewing and modifying data in tables and records
- Changing default field values in macros
- Setting up alternate calendars
- Creating nontraditional courses
- Setting up and using requisites
- Setting up faculty workload
- Cross-listing sections
- Setting up and using bridged sections
- Full-time equivalency status (FTES) processing
- Attendance Accounting Method
- FTES census date calculation
- FTES contact hours calculation
- FTES reports

Basic Information

This section contains detailed procedures specific to the Course/Class Schedule product. For information on performing basic procedures, such as using the MAKE processor and reinstalling options, refer to the following resources:

- *Database Tools and General Utilities* course notebook (SYS200)
- *CX System Reference*

Implementation Process Checksheet

The implementation process checksheet, *Implementing Course/Class Schedule*, shows the general phases in the process to customize and install features of Course/Class Schedule. The checksheet defines the tasks you must complete to install the module and the correct sequence for the tasks.

Use the procedures in this section to help you complete your implementation of Course/Class Schedule.

Cross-Functional Issues

Introduction

As you implement Course/Class Schedule, various policy issues will arise about which you must make decisions. However, in addition to issues affecting only the Registrar's office, there are other issues that involve various offices at an institution. The following are some of these issues, as well as helpful information you can use in deciding how to resolve each issue.

List Of Cross-Functional Issues

The following lists each cross-functional issue, as well as a description of each, that should be addressed while implementing Course/Class Schedule.

Use of course, section, and reference numbers

To ensure that the original intent of an institution's numbering system remains intact, answer the following questions:

- Does the meaning of the course number indicate academic year (e.g., freshman)? Does it include the credit value or some other value?
- What are the institution's policy and procedure on reusing course numbers? How often are numbers reused? Why?
- Are section numbers or letters used? If so, is a meaning associated with them? Do they appear on transcripts?
- Are course or section numbers assigned to academic units in a centralized (e.g., Dean's list or Registrar's office, Curriculum Committee) or decentralized manner (e.g., each department assigns its own numbers)?
- Does the current system use a reference number for each course section? How is the reference number created? How is it used?

Developing the annual processing calendar for catalog and schedule information

The Academic Calendar record (acad_cal_rec) must reflect the course catalog and schedule dates. Therefore, all offices at an institution must understand and participate in developing the calendar used for course catalog and schedule information.

Access to student records, course catalog, schedule, registration, grading, and transcript information by other offices at an institution

To ensure that individuals outside a particular office are able to access student records, course catalog, schedule, registration, grading, and transcript information efficiently and cost-effectively, while guarding against misuse of this ability to access information, answer the following questions:

Student records

Which office is responsible for creating student ID cards? Does the card include information from admitted students' files? How is the information produced?

Catalog and Schedule

- Is an individual other than the registrar responsible for publishing the printed course catalog? How is the master course inventory maintained?
- What information does the bookstore need for ordering books? How does the bookstore get the information it needs?
- Which offices need facility schedule information (e.g., Information Office, Campus Security, Physical Plant)?

Registration

- Which offices need student schedule information (e.g., Financial Aid office, Dean of Students, Information office, Campus Security, Emergency contact office)?

- Is an office other than the Registrar's office (e.g., Institutional Research office) responsible for reports such as enrollment and IPEDS?
- Is an office other than the Registrar's office responsible for or involved in facility assignment or reporting (e.g., Academic Dean, individual academic departments, Physical Plant office)?

Course and section fees

Although the Registrar's and Bursar's offices usually coordinate course and section fees, sometimes fees or charges are handled manually by individual academic departments. CX supports virtually any kind of fee or charge associated with a course or section, and Jenzabar, Inc. recommends an institution-wide review of such charges.

All course or section requisites that govern enrollment

To implement the automated requisite checking feature of CX, complete these tasks:

- Establish requisites for each course that has requisites.
- Review the relationships among multiple requisites as they appear in the course catalog (e.g., review "and" and "or" logic).
- Identify non-course requisites (e.g., test scores and grades) that CX tracks.

All faculty or administration approvals required for students to enroll in courses and sections

Students occasionally must obtain approval from a faculty member, department, or dean in order to register for a course or section, if the requirement is course- or student-based. CX can help apply these requirements consistently. However, every office must completely understand the requirements so that the appropriate records can be created.

Decisions made during Course/Class Schedule implementations regarding student, Course, and Section records used in Registration

Review the decisions made and information entered in the Admissions and Course Class/Schedule applications regarding the following registration requisites to determine their effects elsewhere during registration:

- The academic status of registering students (e.g., admitted, continuing, or returning)
- The appropriate catalog and term for courses and sections

Responsibility (e.g., office or individual) for each of the registration and add/drop functions occurring in Jenzabar CX

The Registrar's office usually is responsible for registration and add/drop functions; however, some related registration processes often involve other offices. To improve the efficiency of the entire registration process and of the services provided to students, the offices involved in registration-related functions must do the following:

- Identify each registration-related function
- Identify who is responsible for each function
- Decide whether CX needs to be modified to accomplish each function

Identification, name, and address management

CX, because it is fully-integrated, is able to provide the following benefits to an institution regarding identification, name, and address management:

- Each individual or entity associated with the institution is assigned a single identification number. Course/Class Schedule deals with only faculty IDs.
- The social security number, an optional feature in CX, calls up an individual's or entity's record just as the ID number does

Formatting standards (e.g., address conventions)

For consistent, professional-looking correspondence, reports, and lists, the offices at an institution must agree on the following:

- How names and addresses are to be formatted
- What the valid values are for various data elements in the system (e.g., the names and abbreviations for degrees conferred, names of academic departments, majors)

Degree Audit

You can categorize courses for degree audit purposes using the degapply field of the crs_rec.

Assessing the Course/Class Schedule Setup

Introduction

CX provides several ways to implement the options of the Course/Class Schedule product. After assessing the needs of your institution, you can change the default settings of Course/Class Schedule enable macros and reinstall the product.

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

Resource25

Course/Class Schedule provides options in the Registrar: Course/Class Schedule menu for accessing the Resource25 interface. If the institution wants these options, you must change the setting of the macro, ENABLE_FEAT_R25, from N to Y.

Schedule25

Course/Class Schedule provides options in the Registrar: Course/Class Schedule menu for accessing the Schedule25 interface. If the institution wants these options, you must change the setting of the macro, ENABLE_FEAT_S25, from N to Y.

Reviewing and Modifying Data in Tables and Records

Introduction

After assessing features of Course/Class Schedule and setting the appropriate enable macros, you must review the setup of CX tables and records.

Procedure

The following procedure provides the steps to review the values of CX tables and records.

1. For each Course/Class Schedule table, review the codes supplied with CX. Determine whether the codes meet the needs of your institution. Make updates as appropriate.
2. Review the institution's records converted from the previous Course/Class Schedule system. Determine whether 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 Course/Class Schedule product, see the section *Course/Class Schedule Tables and Records* in this manual.

Table Setup Sequence

The following lists the sequence in which you should set up the Course/Class Schedule tables. Information about setting up each of the tables below follows in this section. The Course/Class Schedule tables appear in the Registrar: Table Maintenance menu.

Note: For reference information about the tables listed below, see the *Course/Class Schedule Tables and Records* section in this manual.

1. Division table (div_table)
2. Department table (dept_table)
3. Session table (sess_table)
4. Subsession table (subsess_table)
5. Course Catalog table (cat_table)
6. Operator Department table (oprdept_table)
7. Counting table (cntg_table)
8. Requirement table (secreq_table)
9. Requirement Field table (secreqfld_table)
10. Repeat table (rep_table)
11. Period table (prd_table)
12. Program table (prog_table)
13. Instructional Method table (im_table)
14. Faculty Load table (flt_table)
15. Instructor Activity table (act_table)
16. Instructor Qualifications table (qual_table)
17. Non-instructional table (noninstr_table)

18. Non-teaching Date table (nonteach_table)
19. Alternate Calendar table (altcal_table)
20. Alternate Refund table (altrfnd_table)
21. Grading table (grdg_table)
22. Articulation table (artic_table)
23. Discipline table (disc_table)
24. Major table (major_table)
25. Course Category table (crsctgry_table)
26. Degree Application table (degapply_table)
27. Exam table (exam_table)
28. Building table(bldg_table)
29. Degree Group table (deggrp_table)

Building the Course/Class Schedule Tables

Introduction

The processes in the Course/Class Schedule product use the Course/Class Schedule tables to control data entry and to provide necessary information for financial aid processing. You must build these tables before performing any Course/Class Schedule processing and before building any other Course/Class Schedule tables. Database names for each table and field appear in brackets.

Access

You access the Course/Class Schedule tables from the Registrar: Table Maintenance menu.

Alternate Calendar Table [altcal_table]

The following lists the fields in the Alternate Calendar table in the order in which you complete them. To access this table, select Alternate Calendar from the Table Maintenance: Registrar (A-B) menu.

Notes:

- You can enter codes in *crsent* only prior to students enrolling in a section. After one student enrolls, you cannot access the Alternate Calendar Code field.
- The table takes non-counting teaching dates into account for calculations when *crsent* determines the dates for an alternate calendar.
- If percentages of meeting times are used for calculation, the table rounds down the Last Add, Drop, or Withdrawal date depending on when the percentage is met. For example, if 20 percent is entered in the Percentage Withdrawn field, and 20 percent falls half way through a Wednesday meeting time in a Monday, Wednesday, Friday meeting schedule, the Last Withdrawal Date will be on the previous Monday.
- All table entries must be of the same type for an alternate calendar code (either all percentages or all days).

Alternate Calendar Code [altcal]
The alternate calendar code (e.g., 20).

Alternate Calendar Desc [descr]
A description of the alternate calendar (e.g., 20 Meetings).

Method [meth]
A code for the method for this entry. Valid codes (or values) are:

- M (Meetings)
- P (Percentage)
- D (Days)

Percentage Add [pct_last_add]
The percent of days (e.g., 20) before the last day to add.

Meetings To Add [mtg_last_add]
The number of meetings before the last day to add (e.g., 0).

Days To Add [days_last_add]
The number of days before the last day to add (e.g., 3).

Percentage Drop [pct_drop]
The percent of meetings before the last day to drop (e.g., 20).

Meetings To Drop [mtg_drop]

The number of meetings before the last day to drop (e.g., 2).

Days To Drop [days_drop]
The number of days.

Percentage Withdrawn [pct_wd]
The percent of meetings before the withdrawal period ends (e.g., 20).

Meetings To Withdraw [mtg_wd]
The number of meetings before the withdrawal period ends (e.g., 12).

Days To Withdraw [days_wd]
The number of days before the withdrawal period ends (e.g., 2).

Percentage To Charge [pct_charge]
The percent of meetings before the charge period begins (e.g., 20).

Meetings to Charge [mtg_charge]
The number of meetings before the charge period begins (e.g., 2).

Days To Charge [days_charge]
The number of days before the charge period begins (e.g., 0).

Percentage Change Grading [pct_last_grdg]
The percent of meetings (e.g., 20) before the last day to change the grading type.

Meetings Change Grading [mtg_last_grdg]
The number of meetings (e.g., 0) before the last meeting day to change the grading type.

Days To Change Grading [days_last_grdg]
The number of days (e.g., 0) before the last day to change the grading type.

Active Date [active_date]
The date (mm/dd/yyyy) the code will become active.

Inactive Date [inactive_date]
The date (mm/dd/yyyy) the code will become inactive.

Alternate Refund Table [altrfnd_table]

The following lists the fields in the Alternate Refund table in the order in which you complete them. To access this table, select Alternate Refund from the Table Maintenance: Registrar (A-B) menu.

Note: If you have three refund periods, you should have three entries in this table for a given altrfnd code. Be sure that the number of records in the altrfnd_table equals the number of date fields in the altrfnd_rec.

Refund Code [altrfnd]
The alternate refund code (e.g., C) for this group of refunds.

Refund Description [descr]
The description of the alternate refund code (e.g., CEU Refunds).

Method [meth]
The method for this entry. Valid codes (or values) are:

- M (Meetings)
- P (Percentage)
- D (Days)

Refund Number [break_no]
The sequence order number of this refund (e.g., 1).

Percentage of Meetings [pct_mtg]

The percent (e.g., 20) of class meetings before the refund ends.

Number of Meetings [no_mtg]
The number of meetings before the refund ends (e.g., 0).

Number of Days [no_days]
The number of days before the refund ends (e.g., 0).

Active Date [active_date]
The date (mm/dd/yyyy) the code will become active.

Inactive Date [inactive_date]
The date (mm/dd/yyyy) the code will become inactive.

Course Catalog table [cat_table]

The following lists the fields in the Course Catalog table in the order in which you complete them. To access this table, select Course Catalog from the Table Maintenance: Registrar (C) menu.

Notes:

- One blank record is required in the table.
- The addition of all table entries is required.
- Due to other capabilities of the system such as course history and course equivalency that are catalog dependent, a catalog should be established for implementation that will be consistent with other catalogs used. This is especially true if your institution plans to convert course-by-course as opposed to block entry. Since these capabilities will not cross catalogs, ensure that the first two positions of the catalog are consistent. For example, if the UG catalog scheme is to be used, the conversion catalog for undergraduate courses should be UG in the first two positions.

Code [cat]
The code (e.g., CE00) for the course catalog entry.

Description [txt]
The description for the code (e.g., Continuing Educ 2000-2001).

Site [site]
The site code (e.g., MAIN) for this course catalog.

Year [yr]
The year (yyyy) associated with this catalog.

Note: The Registration app server (Web Registration) uses this information to evaluate course history and course equivalencies.

Web Display [web_display]
Either Y (Yes) or N (No) indicating whether to have this course catalog available on the Web.

Web Link Avail [web_link_avail]
Either Y (Yes) or N (No) indicating whether to have course descriptions for this course catalog available on the Web.

Note: If the Web Display field is set to N, this field also should be set to N.

Active Date [active_date]
The date (mm/dd/yyyy) the code will become active.

Inactive Date [inactive_date]

The date (mm/dd/yyyy) the code will become inactive

Counting Table [cntg_table]

For information about the fields in the Counting table, see the *Grading Technical Manual*.

Department Table [dept_table]

The following lists the fields in the Department table in the order in which you complete them. To access this table, select Division/Department from the Table Maintenance: Registrar (D-E) menu.

Notes:

- Create the Division table before creating the Department table.
- Academic departments are frequently considered financial cost centers, which can affect enrollment reporting and grade distribution.
- This table functions with the Division table in a master/detail relationship. When adding or updating the department, the corresponding division code automatically is brought forward on the screen.
- Room ownership is linked to a specific department optionally in the Course Entry program.

Department Code [dept]
The code for the department (e.g., 1002).

Description [txt]
The description for the code (e.g., Geology Department).

Display on Web [web_display]
Either Y (Yes) or N (No) indicating whether to display this record on the Web.

Division Table [div_table]

The following lists the fields in the Division table in the order in which you complete them. To access this table, select Division/Department from the Table Maintenance: Registrar (D-E) menu.

Note: Leave one blank record for the Division table. The blank record is used for all departments if the institution does not use the division concept.

Division Code [div]
The code for the division (e.g., PALE).

Description [txt]
The description for the division (e.g., Paleontology).

Faculty Load Table [flt_table]

The following lists the fields in the Faculty Load table in the order in which you complete them. To access this table, select Faculty Load Type from the Table Maintenance: Registrar (F-L) menu.

Notes:

- You must create the Program table prior to the Faculty Load table
- You must create the Instructional Method table prior to the Faculty Load table

Faculty Load Type [flt]
The code for the faculty load type (e.g., LB).

Description [descr]

The description for the faculty load type (e.g., Lab).

Program [prog]
The code for the program associated with this faculty load type (e.g., UNDG).

Hours [hrs]
The number of hours per week required to be fulltime (e.g., 24.0).

Calculate by contact hours [by_hrs]
Either Y (Yes) or N (No) indicating whether the load is calculated by hours.

Calculate by Students [no_stu]
The number of students required to be fulltime (e.g., 0).

Active Date [active_date]
The date (mm/dd/yyyy) the code will become active.

Inactive Date [inactive_date]
The date (mm/dd/yyyy) the code will become inactive

Instructional Method Table [im_table]

The following lists the fields in the Instructional Method table in the order in which you complete them. To access this table, select Instructional Method from the Table Maintenance: (F-L) menu.

Instruction Method Code [im]
The code for the method of instruction.

Description [descr]
The description for the instructional method code (e.g., Classroom)

Faculty Load Type [flt]
The code (e.g., DS) for the faculty load type associated with the method of instruction.

Active Date [active_date]
The date (mm/dd/yyyy) the code will become active.

Inactive Date [inactive_date]
The date (mm/dd/yyyy) the code will become inactive

Instructor Activity Table [act_table]

The following lists the fields in the Instructor Activity table in the order in which you complete them. To access this table, select Instructor Activity from the Table Maintenance: (F-L) menu.

Activity Code [act]
The code for the activity (e.g., PAPE).

Description [descr]
The description of the instructor activity (e.g., Write Paper).

Sabbatical [sabb]
Either Y (Yes) or N (No) indicating whether the activity is considered a sabbatical.

Active Date [active_date]
The date (mm/dd/yyyy) the code will become active.

Inactive Date [inactive_date]

The date (mm/dd/yyyy) the code will become inactive

Instructor Qualifications Table [qual_table]

The following lists the fields in the Instructor Qualifications table in the order in which you complete them. To access this table, select Instructor Qualification from the Table Maintenance: (F-L) menu.

Qualifications Code [qual]
The code for the instructor's qualification (e.g., MAE).

Description [descr]
The description for the instructor's qualification code (e.g., Masters in English).

Active Date [active_date]
The date (mm/dd/yyyy) the code will become active.

Inactive Date [inactive_date]
The date (mm/dd/yyyy) the code will become inactive.

Non-instructional Table [noninstr_table]

The following lists the fields in the Non-instructional table in the order in which you complete them. To access this table, select Non-Instructional Activity from the Table Maintenance: Registrar (M-Q) menu.

Non-instructional Code [noninstr]
The code for the non-instructional activity (e.g., ADV).

Description [descr]
The description for the non-instructional activity (e.g., Academic Advisor).

Active Date [active_date]
The date (mm/dd/yyyy) the code will become active.

Inactive Date [inactive_date]
The date (mm/dd/yyyy) the code will become inactive.

Non-teaching Date Table [nonteach_table]

The following lists the fields in the Non-teaching table in the order in which you complete them. To access this table, select Non-Teaching Dates from the Table Maintenance: Registrar (M-Q) menu.

Non-teaching Day Date [nonteach]
The non-teaching date (mm/dd/yyyy).

Absolute Exclusion [abs]
Either Y (Yes) or N (No) indicating whether to exclude this date from all calculations?

> **Note:** If N and if a Meeting record (mtg_rec) begins and ends on this date, the date will be considered a teaching date.

Active Date [active_date]
The date (mm/dd/yyyy) the code will become active.

Inactive Date [inactive_date]

The date (mm/dd/yyyy) the code will become inactive.

Operator Department Table [oprdept_table]

The following lists the fields in the Operator Department table in the order in which you complete them. To access this table, select Operator Dept Perms from the Table Maintenance: Registrar (M-Q) menu.

Operator ID [toprdept_no]
The user/group ID number associated with this permission.

ID Type [uid_gid]
The code for the ID type (e.g., U for User).

User ID [sys_id_type]
The identification number from the ID table (id_table) associated with this permission.

Department [dept]
The code for the department or leave blank to select all codes.

Association Code
The Primary Association code of the course, facility, or faculty being evaluated for update permissions.

Priority
The priority of evaluation for the entry. The rows in the Permission table for the user are evaluated in priority order. The first row that matches the criteria is used to provide the permissions information for that user.

View Grades [vw_grades]
Either Y (Yes) or N (No) indicating whether this person has permission to view grades.

Update Grades [upd_crs]
Either Y (Yes) or N (No) indicating whether this person has permission to update grades.

Enter Grades [enter_grades]
Either Y (Yes) or N (No) indicating whether this person has permission to enter grades.

Faculty ID [fac_id]
The faculty ID of courses to which this user has access or enter zero (0) to allow access to courses regardless of faculty ID.

Update Courses [upd_crs]
Either Y (Yes) or N (No) indicating whether this person has permission to update courses.

Update Sections [upd_sec]
Either Y (Yes) or N (No) indicating whether this person has permission to update sections.

Update Meetings [upd_mtg]
Either Y (Yes) or N (No) indicating whether this person has permission to update meetings.

Assign Facilities [assign_facil]
Either Y (Yes) or N (No) indicating whether this person has permission to assign facilities.

Update Facilities [upd_facil]
Either Y (Yes) or N (No) indicating whether this person has permission to update facilities.

Assign Instructor [assign_fac]
Either Y (Yes) or N (No) indicating whether this person has permission to assign instructors.

Update Instructor [upd_fac]
Either Y (Yes) or N (No) indicating whether this person has permission to update instructors.

View Faculty [vw_fac]

Indicates whether the user may view faculty information.

Active Date [active_date]
The date (mm/dd/yyyy) the code will become active.

Inactive Date [inactive_date]
The date (mm/dd/yyyy) the code will become inactive.

Registration Category Table [regctgry_table]

The following lists the fields in the Registration Category and Registration Category Detail tables. To access this table, select Registration Categories from the Table Maintenance: Registrar (R-S) menu.

Note:

The Registration Category table allows your institution to define various categories of students, and associate any number of categories with a section. The Registration Category Detail table defines the data condition that must be met to be in a category. Each category tied to a section has its own registration limit. The categories are defined based on student data in the Program Enrollment record (prog_enr_rec). By using Registration Categories, you enable registration limits for any section that uses those predefined categories. During registration, the process will determine what category the student falls into, and whether space exists in that category. The use of Registration Categories is optional and is controlled via the ENABLE_FEAT_REGCTGRY macro under the CRSENT program.

Code [regctry]
The code for the Registration Category.

Description [desc]
The description for the Registration Category code.

Priority [priority]
Designate the priority in which the table entries are to be evaluated for a student. If a student conforms to the criteria for multiple categories, that student will be considered to be in the one first evaluated as true. If there are multiple entries for the same regctry, those entries should have the same priority.

And/Or [and_or]
Indicates whether the criteria associated with this category are to be and-ed together or or-ed together when evaluated.

Field Name [field_name]
The column name from the prog_enr_rec table, which provides criteria that need to be met for students in this regctry.

Current Value [curr_val]
The value that this column must have to meet the criteria for this regctry.

Operator [opr]
The logical operator used to evaluate the value of the column (=, <, >, !).

Period Table [prd_table]

The following lists the fields in the Period table in the order in which you complete them. To access this table, select Period from the Table Maintenance: Registrar (M-Q) menu.

Notes:

- Create the Session table before the Period table.
- Create a blank entry for this table.

- The period codes can be defined according to your institution's desires. Frequently, registrars will establish some scheme that means something to them. They may want to sequence their meeting times in some order (0800-0850, MWF before 0900-0950 MWF, etc.) and then call them F1, F2, etc. (F for Fall).
- Session and year can be used to input the session (FA) and a year (2000) associated with a period code. Although this implies that the period code is valid for only that session and year, actually the period code is unique and can be used for any session and year.

Period Code [prd]
The code for the period to use for the Meeting record (mtg_rec).

Site [site]
The code for the site (e.g., MAIN) associated with this period code.

Year [yr]
The year period code (e.g., 2001).

Session [sess]
The code for the session (e.g., SP).

Beginning Time [beg_time]
The time this period begins (e.g., 1100)

(Meridiem) [be_ampm]
This unlabeled field is located to the right of the Beginning Time field and designates whether the beginning time is a.m. (a) or p.m. (p).

Ending Time [end_time]
The time this period ends (e.g., 1200)

(Meridiem) [end_ampm]
This unlabeled field is located to the right of the Ending Time field and designates whether the ending time is a.m. (a) or p.m. (p).

Days [days]
A sequence of seven codes for the day(s) the class will be held beginning with Sunday. Valid codes (or values) might be:

- S (Sunday)
- M (Monday)
- T (Tuesday)
- W (Wednesday)
- T (Thursday)
- F (Friday)
- S (Saturday)
- - (No class)

Notes:

- Use the day code to indicate the day(s) the class will be held. Use the – to indicate a day with no class session.
- The Days field should use the same structure as defined in the macros/custom/student file for valid day abbreviations.

  Example: A Monday, Wednesday, Friday class code would be -M-W-F-.

Repeat Table [rep_table]

For information about the fields in the Repeat table, see the *Transcript Technical Manual*.

Requirement Field Table [secreqfld_table]

This table is a detail table to the secreq_table. It provides the criteria for meeting a section requirement. Multiple secreqfld_table rows tied to the same secreq are "OR"ed together when they are evaluated in registration.

The following lists the fields in the Requirement Field table in the order in which you complete them. To access this table, select Section Requirement from the Table Maintenance: Registrar (R-S) menu.

Table Name [secreq]
The Informix filename associated with the checking of this requirement (e.g., prog_enr_rec).

Column Name [field_name]
The column name in the table associated with this requirement.

Column Value [value]
The value this column must have to satisfy this requirement.

Operator [opr]
The relational operator to be applied against the value. Valid relational operators are:

- < (less than)
- > (greater than)
- ! (not equal to)

Requirement Table [secreq_table]

The following lists the fields in the Requirement table in the order in which you complete them. To access this table, select Section Requirement from the Table Maintenance: Registrar (R-S) menu.

Notes:

- Frequently an institution will specify that certain sections of courses are to have certain requirements placed upon them. Examples would include "this class only for majors," "this class is limited only to international students," "for upper division students with a 3.0 GPA or higher," and so forth. This table provides the capability to establish such requirements. Depending on the value of the Registration Check field, the section requirement can be evaluated within registration. If the value of the registration check is N, the requirement will appear only as a warning.
- Leave one blank entry in this table.

Code [secreq]]
The code for the section requirement (e.g., BIOM).

Description [txt]
The description for the code in the Code field (e.g., Biology Majors).

Registration Warning [warn]
Either Y (Yes) or N (No) indicating whether a warning should appear on the screen during registration.

Registration Check [reg_chk]
Either Y (Yes) or N (No) indicating whether the registration process should check this requirement.

Comment [comm]
A comment about this requirement (e.g., You must be a biology major to take this course).

Note: This is shown in Web Registration when a student has not met this type of section requirement.

Active Date [active_date]

The date (mm/dd/yyyy) the code will become active.

Inactive Date [inactive_date]
The date (mm/dd/yyyy) the code will become inactive.

Session Table [sess_table]

The following lists the fields in the Session table in the order in which you complete them. To access this table, select Session/Subsession from the Table Maintenance: Registrar (R-S) menu.

Notes:

- The Session table includes every session for which registration will occur. It is used in conjunction with the academic calendar.
- Create the Session table before creating the Subsession table.
- Add or update this table to reflect the sessions your institution uses. Typical sessions are FA (Fall), SP (Spring), and SU (Summer). An academic calendar must be available for every session for which registration occurs.
- The calendar year order is very important to the order sessions are printed on the transcript. Enter them in the order they appear in the calendar year.
- The academic year order is the order of the sessions within an academic year.
- The Transcript program uses three session codes:
  - Foot is used to define the location of transcript footers containing non-session/non-course work data (e.g., a high school attended)
  - Head is used to define the location of transcript headers containing non-session/non-course work data (e.g., a high school attended)
  - Prev is used for data conversion, to convert prior data to previous sessions

Session Code [sess]
The code for the session (e.g., FA).

Description [txt]
The description for the code in the Session Code field (e.g., Fall).

Calendar Year Order [yr_offset]
A number indicating the order in which the session occurs in the calendar year (e.g., SP=0, SU=1, FA=2).

Academic Year Order [ord]
A number indicating the order in which the session occurs in the academic year (e.g., FA=0, SP=1, SU=2).

Course Tuition [crs_tuit]
Either Y (Yes) or N (No) indicating whether the course tuition code overrides the program tuition code.

Web Display [web_display]
Either Y (Yes) or N (No) indicating whether to enable this session to appear on Web applications.

Web Order [web_ord]
A number indicating the order of session codes on Web applications.

Subsession Table [subsess_table]

The following lists the fields in the Subsession table in the order in which you complete them. To access this table, select Session/Subsession from the Table Maintenance: Registrar (R-S) menu.

Notes:

- The Subsession table contains a breakdown of sessions into shorter periods (e.g., first five weeks, second five weeks, third five weeks, etc.)
- Create the academic calendar before creating the Subsession table.
- Create the Session table before creating the Subsession table.
- A subsession must be linked with a session.
- Leave one blank entry in this table.
- Each subsession must have its own academic calendar.
- If subsessions are necessary, they affect both registration and billing.
- If a session consists only of subsessions, an academic calendar always is required for the period spanning the entire session with the subsession blank.

Subsession Code [subsess]

The code for the subsession (e.g., S1).

Description [txt]

The description for the code in the Subsession Code field (e.g., First summer session).

Setting Up Alternate Calendars

Introduction

It is common for an institution to offer courses that have a shorter duration than the entire session. When an institution offers short-term courses, frequently the following dates change:

- The add, drop, withdrawal, and charge dates differ from the traditional calendar also known as the standard academic calendar
- The refund dates differ from the traditional refund schedule

To address the nontraditional calendar dates and refund schedule, CX contains alternate calendar dates at the section and meeting level. Alternate calendars and alternate refund dates are an alternative to the standard session academic calendar (acad_cal_rec) and/or the subsession academic calendars. Using an alternate calendar should reduce, if not eliminate, the requirement for subsession calendars.

The requirement for alternate calendars is based on the assumption that an institution could offer these short-term courses for many different periods. As an institution adds the Section records, CX initially assumes that the course is a full-session course and uses the beginning and ending dates of the full session's academic calendar as defaults.

If the beginning and ending default dates do not correspond to the actual beginning and ending dates for the section, an institution can change the dates before any students register. Entering information into the Alternate Calendar table and then specifying the altcal code and/or altrfnd code at the section level enables the system to compute add, drop, withdrawal, and charge dates specific to that alternate calendar. Furthermore, entering information into the Alternate Refund table results in the system computing alternate refund dates.

Setup Introduction

The Course Entry program automatically creates the alternate calendar using the meeting days and dates that you enter at the meeting level. However, the system cannot complete the Alternate Calendar record accurately unless you already have added the Alternate Calendar and Alternate Refund tables. If no altcal or altrfnd code is specified for a section, Course Entry uses the dates from the full session's academic calendar.

The following list describes each of the three components used for setting up alternate calendars:

Macros

The directory path $CARSPATH/macros/custom/student contains the following macro that pertains to alternate calendars: m4_define ("ENABLE_FEAT_ALTCAL", "Y"). The "Y" is the default value in the base product, and if you leave the "Y," an alternate calendar detail window is available when menu users select the Options command in the Section Record screen. The "N" value prevents the detail window from appearing. Set the macro definition value to "N" only if you do not plan to use alternate calendars. For more information on how to set up macros, see the *Getting Started User Guide.*

Includes

The directory path $CARSPATH/includes/custom/billing contains the include "NO_NET_RFND", which you must define if you are using alternate refund calendars. For more information on how to define an include, see the *Getting Started User Guide.*

Tables

You must set up the following two tables before using alternate calendars:

- Alternate Calendar table (altcal_table)
- Alternate Refund table (altrfnd_table)

Alternate Calendar Table Setup

You must add a specific Alternate Calendar table (altcal_table) entry for each different combination of add, drop, withdrawal, and add and drop charge dates to be calculated.

Information listed in a table entry should be one of the following sets of fields in the Alternate Calendar table:

- All percentage of meetings (Percentage Add, Percentage Drop, Percentage Withdrawn, Percentage to Charge)
- All number of days (Days to Add, Days to Drop, Days to Withdraw, Days to Charge)
- All number of meetings (Meetings to Add, Meetings to Drop, Meetings to Withdraw, Meetings to Charge)

The Alternate Calendar Table

Do not mix the percentage meeting and days dates. It is common that the dates and percentages are based on the same ratio as the dates in the standard academic calendar. For example, if the session academic calendar allows you adds for the first five meetings of a 16-week session (three meetings each week), then the percentage is 10% of the meetings for the session (e.g., 48/5 = 10%). Therefore, when you add an Alternate Calendar table for a section that meets for only four weeks, you should set the percentage of Adds to 10%. Based on the number of meetings between the beginning and ending dates for the four-week course, less any non-teaching dates, the Course Entry program determines an actual last day to add the section.

Note: The Percentage to Charge and Days to Charge fields are the percent or days, respectively, before the system prompts the menu user in the Registration program (*regist*) to charge for adds or drops. A charge entry of "0" in the Percentage to Charge field means that the prompt for the add/drop charges in the registration process begins on the first day of classes (i.e., the beginning date) for the section.

The Process for Setting Up Alternate Calendars

It is critical that you enter the correct beginning and ending date in the Meeting Schedule window (mtg_rec). The default values are the full session beginning and ending dates. If you use an alternate calendar, you must enter the appropriate beginning and ending dates for the meeting. It is important that you specify an alternate calendar code and an alternate refund code. If you do not enter an alternate calendar code and an alternate refund code, the system bases the add, drop, and withdrawal dates on the full session calendar and bases refunds on the full session refund schedule. You cannot change the altcal or altrfnd codes after students register in the course.

Access the Meeting Schedule window and complete the fields required for all scheduled meetings of the course. The beginning and ending dates at the meeting level all should fall within the beginning and ending dates at the section level for all meetings.

The system establishes the actual alternate calendar dates based on the first beginning date and the last ending date for all meetings. The default dates in the Meeting Schedule window come from the dates entered at the section level. However, you can change the default dates when completing the fields on the Meeting Schedule window.

Access the Section Record screen for the section. Select **Options** and then select **Alternate Calendar/Refund**. A window appears that contains the section calendar and section refund calendar, based on section and meeting data already entered.

Note: If more than three periods are required, you must modify the Section Calendar window accordingly. The Section Calendar window is located in the following directory path: $CARSPATH/modules/regist/progscr/crsent/altcal.

Setting Up Alternate Refunds

Introduction

You must add an Alternate Refund table (altrfnd_table) for each combination of refund logic that you use. You must have as many rows (records) for each alternate refund code as you have refund periods in the Refund table (rfnd_table). Also, the refund for a given period can be based either on percentage of meetings, number of meetings, or number of days.

Note: You must have three entries for an Alternate Refund code in the altrfnd_table, whether or not you are using billing or refunding.

Example Setup

For example, an institution can have a policy that a student can receive refunds in the following situations:

- A 90% refund if the student drops a course by the time the course meets 20% of its meetings
- An 80% refund if the student drops a course by the time the course meets 40% of its meetings
- A 50% refund if the student drops a course by the time the course meets 60% of its meetings

You must add three Alternate Refund table entries in the Refund Code field for the same refund code. In this example, the first row for the first refund date contains a refund number of 1 and a percentage of meetings of 20%, the second row a refund number of 2 and a percentage of meetings of 40%, and the third row a refund number of 3 and a percentage of meetings of 60%. The Refund table (rfnd_table), established during the student billing process, determines the actual amount of the refund. In practice, the alternate refund dates calculated from the Alternate Refund table override the refund dates entered in the Refund table.

If a section contains all TBA meetings, the menu user can enter the dates manually in the Alternate Calendar window. This occurs only for TBA meetings, and the system overwrites the dates if a non-TBA Meeting record is assigned before the institution adds students to the section.

Creating Nontraditional Courses

Introduction

CX allows you to create courses that do not follow either the standard academic calendar or the alternate academic calendar for use in the registration process. These courses are called nontraditional courses.

Institutions typically define nontraditional courses as those with beginning or ending dates related to the enrollment date of the student. The system provides a calculated beginning or ending date to identify student enrollment.

Institutions commonly use nontraditional courses for the following types of courses:

- Open entry – open exit courses
- Correspondence courses

Setup

The nontraditional course design begins at the course catalog level. The following list contains the two fields in the Course record (crs_rec) that define the default standards for creating the Section record (sec_rec):

max_days (Days to Complete)
The maximum number of calendar days in which a student is allowed to complete a nontraditional course.

last_drop_days (Days to Drop)
The maximum number of calendar days in which a student is allowed to drop a nontraditional course.

Using Nontraditional Functionality

CX uses the Open Enroll field (open_enr) of the Section record to enable the nontraditional functionality. The feature uses the fields in the Course and Section records for nontraditional courses. Set the Open Enroll field in the Section Record screen to Y to use the nontraditional functionality.

Calculating Beginning and Ending Dates

The Section record can compute the beginning date in two different ways:

- If the beginning date in the Section record is blank, Registration takes the effective date of registration as the beginning date and calculates the ending date from the Days to Complete field in the Section record
- If the beginning date in the Section record is non-blank, Registration takes the date as the beginning date and calculates the ending date for the Days to Complete field in the Section record

To compute the ending date, the Days to Complete field in the Section record uses the Days to Complete field in the Course record as a default value. Normally, the Days to Complete field in the Section record overrides the Days to Complete field in the Course record. You can override the Days to Complete field in the Section record if the allotted time for the section differs.

The following list describes the processing of beginning and ending dates of nontraditional courses according to the entry in the Beginning Date, Ending Date, and Number of Days fields of the associated student's Course Work record.

Beginning date	Ending date	Number of days	Computed date(s)

Beginning date	Ending date	Number of days	Computed date(s)
Blank	Blank	>0	The system computes both beginning and ending dates.
Non-blank	Blank	>0	The system computes the ending date.
Non-blank	Non-blank	>0	The system computes the ending date.

Note: If you set the Number of Days field at zero (0), you must enter dates in the Beginning Date and Ending Dates fields.

Retaining the Calculated Dates

The system adds beginning and ending dates from the nontraditional section to the Course Work records of the individual student. You can view the calculated dates linked to a specific student by using the Course Options command and then the Enrollment Detail window in Registration.

The Enrollment Detail window is located in the following directory path: $CARSPATH/modules/regist/progscr/regent/stucrs.

Note: One of the primary purposes of the process is to set the beginning and ending dates of the course on the student transcript.

Nontraditional Course Limitations

The following are some limitations associated with nontraditional courses:

- You cannot set separate add, drop, or withdrawal dates with nontraditional sections as you can with sections in the alternate academic calendar.
- CX does not include a means to link the calculated registration dates to a corresponding set of calculated dates. All sections follow the standard refund calendar or the alternate refund calendar, or you must manipulate the financial record manually.
- CX contains no standard reports for the nontraditional sections. You must create local reports as you determine data reporting requirements.

Setting Up and Using Requisites

Introduction

Students must complete certain requirements called *requisites* before they are allowed to register for a particular course. Students must complete a requisite either prior to or during the same session that they take the desired course.

The Registration program uses requisites solely to ensure that students possess the proper academic background for a course. Therefore, you need to enter requisites only one time for the catalog in which the requisites become effective. Requisites remain effective for all subsequent catalogs unless the institution updates or removes the requisites. An institution should attach requisites to only one course and catalog. If the requisite changes in subsequent years, the institution should make new entries to the course and catalog for the catalog that the requisite is first effective. An institution must remove all past requisites for such a course.

Types of Requisites

The following list describes the three types of requisites used in CX:

Concurrent requisite
A requirement that a student must complete during the same session in which they take the course that has the concurrent requisite.

Example: Biology 210 (BIO210) has a companion laboratory course of Biology 202 (BIO202). A student must register for both Biology 210 and Biology 202 during the same session.

Corequisite
A requirement that a student must complete either prior to or during the same session in which they take the course that has the corequisite.

Example: History 302 (HIS302) has a corequisite of History 301 (HIS301). If a student wants to register for History 302, then the student must have successfully completed History 301 previously, and if not, then the student must register for both History 301 and History 302 in the same session.

Prerequisites
A requirement that a student must complete before registering for a particular course.

Example: A student wants to register for English 102 (ENG102), which has a prerequisite of English 101 (ENG101). The student must have successfully completed English 101 before registering for English 102.

Although highly unusual, an institution could have a course with all three categories of requisites attached to the course. A course can have as few as zero requisites (if the institution establishes that a course does not have any requisites associated with it), to as many as three requisites. The system displays a warning message for any portion of the requisite(s) that a student has not yet completed.

Note: Currently, CX administers only course and examination requisites. The course and examination requisites can be either conditions or combinations of conditions for the same course.

Setup

There are no macros or includes that pertain directly to setting up requisites. Since requisites are linked to a course for a specified catalog, the system adds the required records from the menu, using the Catalog Maintenance menu option from the Registrar: Course/Class Schedule menu.

How to Establish Prerequisites

A menu user can enter course and examination prerequisites in the Course Prerequisites window.

After performing a query on the course and catalog in the Course Catalog screen, a menu user can access the Course Prerequisites window by selecting **Options** and then selecting **Course Requisites**, then **Prerequisites** from the pull-down menu that appears.

Compound prerequisites are relatively common and use the features of logic. For example, the course AFR342 has a prerequisite of AFR211 and AFR212, or a composite score of 21 on the ACTP test. A student must have a D grade or higher in both AFR211 and AFR212 before registering for AFR342.

The following list describes what commands to use when a menu user wants to add more than one prerequisite in the Course Prerequisites window:

- If a menu user wants to add another prerequisite in the Course Prerequisites window using the and logic, then select the **Add** command. The word “AND” appears on the far right of the Course Prerequisites window.
- If a menu user wants to add another prerequisite in the Course Prerequisites window using the or logic, then select the **Add Group** command. The word “OR” appears on the far right of the Course Prerequisites window.

Course Corequisites Window

A menu user can enter course and examination corequisites in the Course Corequisites window.

For example, the course AFR342 has a corequisite of AFR214. This means that a student must complete AFR214 either prior to or during the same session as when he or she takes AFR342.

How to Establish Corequisites

After performing a query on the course and catalog in the Course Catalog screen, a menu user can access the Course Corequisites window by selecting **Options**, then selecting **Course Requisites**, and then **Corequisites** from the pull-down menu that appears.

A menu user can add compound corequisites that use the features of logic. The following list describes what commands to use when a menu user wants to add more than one corequisite in the Course Corequisites window:

- If a menu user wants to add another prerequisite in the Course Corequisites window using the and logic, select the **Add** command.

  Note: The word “AND” appears on the far right of the Course Corequisites window.

- If a menu user wants to add another prerequisite in the Course Corequisites window using the or logic, select the **Add Group** command.

  Note: The word “OR” appears on the far right of the Course Corequisites window.

Course Concurrent Requisites Window

A menu user can enter course and examination concurrent requisites in the Course Concurrent Requisites window.

For example, the course AFR342 has a concurrent requisite of AFR343, meaning that a student must take both courses during the same session. Concurrent requisites are common for lecture and laboratory courses.

How to Establish Concurrent Requisites

After performing a query on the course and catalog in the Course Catalog screen, a menu user can access the Course Concurrent Requisites window by selecting **Options**, then selecting **Course Requisites**, and then **Concurrent Requisites** from the pull-down menu that appears.

A menu user can add compound concurrent requisites that use the features of logic. The following list describes what commands to use when a menu user wants to add more than one concurrent requisite in the Course Concurrent Requisites window:

- If a menu user wants to add another concurrent requisite in the Course Concurrent Requisites window using the and logic, select the **Add** command.

  Note: The word "AND" appears on the far right of the Course Concurrent Requisites window.

- If a menu user wants to add another prerequisite in the Course Concurrent Requisites window using the or logic, select the **Add Group** command.

  Note: The word "OR" appears on the far right of the Course Concurrent Requisites window.

How the Registration Program Uses Requisites

Each time you add a course using the Registration program, the system automatically checks to see whether requisites exist. For more information on how the Registration program performs requisite checking, see *Adding and Updating a Course* in the *Registration User Guide*.

The following list describes what happens when the system finds an unmet requisite(s):

- If a student has partially satisfied an unmet requisite(s), then the system displays only the portion of the requisite that the student has not satisfied.
- If a student has not satisfied a requisite(s), then the menu user can do one of the following:
  - Override the requisite(s) by selecting **Finish**
  - Cancel the registration for the course by selecting **Cancel**

  Note: Selecting Cancel causes a message to appear at the bottom of the Registration screen. The message indicates that the student has not registered for the course due to an unmet prerequisite(s), corequisite(s), or concurrent requisite(s).

Setting Up Faculty Workload

Introduction

In CX, faculty workload is a mathematical model designed to accurately compute the workload of each faculty member. The faculty workload feature bases workload computations primarily on actual instruction, but it also records noninstructional duties. Both instructional and noninstructional values make up the total workload for each faculty member.

Faculty Workload Description

The faculty workload feature, a model based on contact hours, calculates the number of teaching platform hours for the instructor, rather than calculating credit hours for the courses taught. Institutions commonly use this method of calculation because different types of instruction can carry different weighted values.

For example, to be considered full-time, an instructor who teaches by the lecture method must work 12 platform hours per week while a laboratory instructor must work 24 platform hours per week.

Since most instructors use a variety of instructional methods, the faculty workload feature uses contact hours, which contain values for each type of instruction, as a common denominator. When the load is printed, displayed, or otherwise listed, it is dynamic and based on contact hours.

The base computations for instruction include the following:

- The system divides the contact hours for a course by the number of weeks for the session
- The system then divides the result by the number of hours required to be full-time for the course's method of instruction

For example, an instructor teaches a three-hour lecture course that meets for three hours per week, and the standard session is sixteen weeks. The total time that the course is required to meet is 48 hours (16 times 3), and twelve hours of instruction by lecture is considered a full-time load, also called a full-time equivalent (FTE). The system computes the instructor's workload for the course in the following way: it divides 48 by 16, and divides again by 12, resulting in .25. Therefore, the course counts as .25 (or 25%) of the instructor's workload.

The faculty workload feature computations include various other factors. Examples include the following:

- The ability to compute team-taught courses (i.e., multiple faculty per course).
- The ability to include or exclude a course from workload computations.
- The ability to place accounting in the following categories of contract:
    - Overload
    - Hourly
    - Banked
    - Instructor of record
    - Other category (for unique categories of instruction)
- The preparatory instructional type for instances in which an instructor is given credit for preparing to teach a class, but the class is subsequently canceled.
- Faculty workload computations are based on courses rather than the number of students in the courses. For possible future use, the faculty workload feature contains an initial definition of workload features based on the number of students taught.
- An instructor who teaches multiple courses and sections at the same time, same days, and same classroom (referred to as cross-listed courses) can earn workload for only one occurrence in this case.

Macros That You Must Update

You must review and update as necessary the following macros, located in the following directory path: $CARSPATH/macros/custom/student.

ENABLE_FEAT_FACLOAD

This macro determines whether to use and maintain the faculty workload. The default value is Y. If you do not plan to compute faculty workload, set the macro to "N."

> **Note:** If you set the above macro to "N," you do not need to review the following macros.

ENABLE_FEAT_CHANGE_FLT

When you set this macro to "Y" (the default value), the system sets the Instructor records of a canceled course to "preparatory," indicating that the institution will give workload factors to an instructor who has prepared to teach a course before the course was canceled. If you set the macro to "N," the system does not automatically set the Instructor records (instr_rec) to "preparatory" when a course is canceled; therefore, you must manually set the records if preparatory workload values are used. When the macro is set to "N," you also must set the Instructor records to some value other than lecture or lab so that the system does not compute faculty workload for canceled courses.

> **Note:** Jenzabar, Inc. recommends that you set this macro to "Y" to ensure that the workload calculation is correct.

FACULTY_LOAD_CONSTANT

This macro represents the constant number of weeks for a standard session used in faculty workload computations. The macro provides a set number of contact hours for all courses in a session even though the course lengths vary. For example, the system awards the same amount of credit to a course that meets three hours a week for 16 weeks as it does for a course that meets six hours a week for eight weeks. The default value is 16.

> **Note:** Do not change this macro after you initially set it.

FAC_LOAD_HRLY_ACCRL_MAX and FAC_LOAD_OVR_ACCRL_MAX

These macros define the maximum faculty load equivalency value that a faculty member can accrue in the hourly and overload accrual methods. The default for both macros is 64.

> **Note:** When the specified value is exceeded, Course Entry warns you when you attempt to enter additional assignments for the instructor.

IPEDS_SCH_ID

This macro defines the identification number of the school for which to include ed_recs. This ID almost always should be the identification number of the institution. According to rules established by the National Center for Education Statistics (NCES), degrees from other institutions should be covered under that institution's IPEDS reporting.

IPEDS_REPORT_TYPE

This macro defines the type of report.

Macros That You Do Not Have to Update

The following macros do not require modification; however, you should review them. The macros are located in the following directory path: $CARSPATH/macros/custom/student.

FAC_NON_INSTR_LOAD_TYPE

The code "NI" (the default value) represents a code used for non-instructional duties such as department chair and research. The code represents an institution's release time policy in which the instructor can obtain workload credit by performing noninstructional assignments.

FAC_PREP_LOAD_TYPE

This macro provides the value used for preparatory time given to instructors whose courses are canceled. The Faculty Load table (flt_table) must contain an entry for this value. The default value is PR.

FAC_ACCRL_DEF
The default value is C. This macro computes faculty workload based on the contract accrual method assigned in the Instructor Assignment record (instr_rec).

FAC_ACCRL_VALID
The values for this macro are B, C, H, O, R, and X. This macro can compute faculty workload using all contract accrual methods assigned in the Instructor Assignment record (instr_rec).

FAC_ACCRL_INCL
This include references all the values in the FAC_ACCRL_VALID macro.

FAC_ACCRL_BANK
The default value is B. This macro computes faculty workload based on the banked accrual method assigned in the Instructor Assignment record (instr_rec).

FAC_ACCRL_CONTRACT
The default value is C. This macro computes faculty workload based on the contract accrual method assigned in the Instructor Assignment record (instr_rec).

FAC_ACCRL_HOURLY
The default value is H. This macro computes faculty workload based on the hourly accrual method assigned in the Instructor Assignment record (instr_rec).

FAC_ACCRL_OVER
The default value is O. This macro computes faculty workload based on the overload accrual method assigned in the Instructor Assignment record (instr_rec).

FAC_ACCRL_RECORD
The default value is R. This macro computes faculty workload based on the record accrual method assigned in the Instructor Assignment record (instr_rec).

FAC_ACCRL_OTHER
The default value is X. This macro computes faculty workload based on the other accrual method assigned in the Instructor Assignment record (instr_rec).

Accrual Methods for Faculty Workload

The following list describes the accrual methods used by the last *nine* macros described.

Banked (B)
Computed workload not used in the session in which it is accrued.

> **Note:** The accrual method is available for future sessions. Currently, CX does not automatically consider banked workload in a subsequent session.

Contract (C)
Regular full-time faculty who have contracts with the institution.

Hourly (H)
Faculty that work either on an hourly or part-time basis.

Other (X)
The generic accumulation of workload information not accounted for in the other accrual methods.

Overload (O)
Instructional responsibilities beyond the normal workload.

Record (R)

The instructor of record for sections in which a number of registered students are assigned to a number of instructors.

Note: The instructor of record serves as a clearing agency for class lists and grading. The workload associated with this accrual type does not actually belong to the instructor of record, but must be distributed manually among the faculty.

Table Setup

You must establish the following tables for use in the faculty workload.

Instructional Method Table (im_table)
Contains all instructional methods used by the institution. The values typically are associated with a course in the Instructional Method record (im_rec) where the entries are standards for required hours (that a course meets per session) and instructional methods.

Note: You also can specify the actual instructional method(s) for a course section in the Meeting record (mtg_rec); the data may or may not correspond to the instructional methods established at the course level. The Instructional Method table also contains a field that points to the corresponding Faculty Load table value that defaults into the Instructor Assignment records when they are created.

Faculty Load Table (flt_table)
Contains actual types of instruction with the number of hours of instruction per week necessary for a full-time equivalent (FTE). For example, the code "LC" for lecture can contain a value of 15 hours, which indicates that an instructor must teach 15 platform (contact) hours per week to be equivalent to one FTE.

Note: CX currently functions only on contact hours and does not use the Calculate by Students field.

Non-Instructional Table (noninstr_table)
Contains all valid types of noninstructional activities that can be entered into the Non-Instructional record (noninstr_rec) for use in computing a total instructional and non-instructional workload. The Non-Instructional record is associated with the Faculty record (fac_rec).

Non-Teaching Date Table (nonteach_table)
Contains all the non-teaching days for an academic year. A meeting's contact hours are calculated based on actual meeting occurrences, which exclude nonteaching days.

Records Used in Faculty Workload Computations

Seven records are used in faculty workload computations. Descriptions of each of the records follow.

Faculty Record

As the first logical step to implementing faculty workload, you must ensure that every instructor has a valid Faculty record (fac_rec). You can access these records by accessing the Registrar: Course/Class Schedule menu and selecting the Catalog Maintenance option. From this option, select **Faculty Maintenance**.

After accessing the Instructor Information screen, perform the following:

- Perform a query to ensure that the record does not already exist
- Add a new record or update the existing record based on the results of the query

CAUTION: Faculty maintenance makes changes to the ID record (id_rec), which is used throughout the system.

You also can access the Instructor Information screen from the Instruction Assignments window by selecting **Faculty**. The system displays the instructor information entered in the Instruction Assignments window.

Nonteaching FTE Assignment Record

The Nonteaching FTE Assignment record table (noninstr_rec) records the administrative or other non-instructional assignments (commonly called release time) for an instructor. These assignments are considered in the total faculty workload. The table permits only entries that already have been added to the Non-Instructional table. After accessing the Instructor Information screen, perform the following:

- Access the Noninstruction Assignment window and select the **Noninstr assign** option.

  Note: The system displays a window that contains all non-instructional assignments.
- Add, update, or delete additional assignments.
- Enter the following when adding assignments:
  - The accrual method (Accrl)
  - The assignment (Assign), such as department chair
  - The full-time equivalency (FTE) for the assignment

Course Catalog Record

The Load field in the Course Catalog record is crucial for faculty load calculations. Enter "Y" for yes and "N" for no to determine whether the course counts for faculty workload calculations.

Course Contact Hours Record

The Course Contact Hours record links to the course because the record contains the standard contact hours by instructional method that the course must meet to fulfill instructional requirements. After accessing the course for which you want to enter contact hours, perform the following:

- Access the Course Contract Hours window.
- Enter the instructional method in the IM field and the contact hours per credit for the course in the Hours/Credit field.

For more information on how to complete the fields in the Course Contact Hours window, see *Using the Course/Class Schedule Screens and Windows* in the *Course/Class Schedule User Guide.*

Example: A lecture course that meets three hours per week for three credit hours during a session of 16 weeks, meets 16 contact hours per credit hour. Add an entry to the Course Contact Hours window for every instructional type associated with the course.

Example: If the course is lecture/lab, you can enter two instructional method entries: one for lecture and one for laboratory. The system compares the data that you entered to the amount of meeting time entered to determine whether the meeting time for the course is sufficient.

Section Record

The Section record (sec_rec) contains academic calendar information from the standard academic calendar for the session or an alternate calendar specified for the section. As you add the section, ensure that the beginning date and ending date are correct for the session. Enter the alternate calendar code, if applicable. The system computes contact hours based on entries in the Meeting record (mtg_rec) and the number of meetings between the beginning and ending dates of the calendar for the section, minus any non-teaching days.

The system then matches the computed contact hours to the course for the section and/or section standard to determine whether the meeting time is sufficient. If the meeting time is not sufficient, the system displays a message indicating so.

If you want to make the contact hours in the Section record greater or less than the contact hours in the Course record, perform the following:

- Access the Section Contact Hours window
- Complete the fields in this window

For more information on how to complete the fields in the Section Contact Hours window, see *Using the Course/Class Schedule Screens and Windows* in the *Course/Class Schedule User Guide.*

Note: Your entries at the course level appear as the first line on the Section Contact Hours window. If you want to reduce the time the section meets, enter the same instruction method and enter the new time in the Section Contact Hours window. However, if you want to change the instructional method from that entered for the course, make the following two entries: one entry to blank out the instructional method of the course and one entry to add the new instructional method and time.

Meeting Record

You can access the Meeting record (mtg_rec) in a section of a course by selecting **Meeting** in the Section Record screen. The Meeting Schedule window appears. Enter the following in the Meeting Schedule window:

- The actual instructional method.
- Either a period code (PC) or the actual days and time the section meets and the number of hours (Hrs) of each meeting. For example, a meeting on Monday from 0900a to 0950a meets for one hour.
- Complete the remaining fields in the window even though the displayed fields pertain to faculty load.
- Add as many Meeting records as required for the course, completing each field with the same steps as the first entry, even though the instructional method may be different.

You also can enter a manually calculated number of contact hours in the ManHrs/CalcHrs fields. Jenzabar, inc. recommends that you leave the field intact until you select **Finish** to save the data. Perform the following:

- Access the Meeting Schedule window, and an advisory window advises you if the course meets too few hours. If the advisory window does not appear, the Meeting Schedule window appears.
- View the number of calculated hours in the Hrs field. You can determine whether an entry is necessary in the ManHrs/CalcHrs fields.
- Enter a value in the ManHrs/CalcHrs fields to base the workload on the manual entry, not the computed value, if the course meets for too many hours, and the institution's policy indicates that workload cannot be computed for more hours than the standard.

Instructor Record

Access the Instructor record (instr_rec) from the Meeting Schedule window by selecting **Instr**. The Instruction Assignments window appears. Add the instructor(s) to each meeting schedule entry.

For information on how to complete the fields on the Instruction Assignments window, see *Using the Course Entry Screens and Windows* in the *Course/Class Schedule User Guide.*

- You can add multiple Instructor records for each meeting, but you must be cautious to ensure that each instructor gets proper credit. The accrual in the Acrl field value for one

instructor for a meeting is 100%. If the meeting is team-taught with two instructors who are both teaching the same amount, the value in the Acrl field should be 50% for each.

- The time for each meeting should total to 100% to indicate that a course has two meetings, one for lecture and one for laboratory. Therefore, you must make two entries.
- The accrual value in the Acrl field for each meeting should be 100%, distributed as 100% for one instructor, or possibly 50% each for two instructors for the same meeting.

When you re-enter the Instruction Assignments window, the system displays the system-computed FTE. If you wish to enter a manual FTE, you can enter the value.

Note: Changes in the Meeting Schedule window for days and times and/or changes in instructor assignments result in the system recomputing the FTE. However, returning to the Section Record screen and altering the calendar dates does not adjust the workload factors. Therefore, if you want to make changes to the workload, you must manually make the changes. You cannot change the calendar dates after students begin registering for the section.

How Faculty Workload is Computed

After you properly set the macros and add the necessary tables and records, the system can compute faculty workload. An example of the computations follows:

An instructor teaches a three-hour course that meets 16 hours per credit per session. The course, which counts in faculty load, has one instructor and one lecture method for 100% of the time.

- The Meeting record should have 48 (3 hours times 16 weeks) in the Hrs field. (The entry assumes that the meeting entered calculates to the same value as the standard for the course.)
- The system divides 48 by the value in the FTE_LOAD_CONSTAT field. The system sets the value to 16, which represents the weeks in the session (48 divided by 16 equals 3).

For these computations, if the hour requirement for lecture method is 15, then 3 is divided by 15, which equals 0.2.

Therefore, this course contributes 0.2 toward the instructor's FTE. The system adds all of the instructional assignments and non-instructional assignments together for a total workload by accrual type.

Note: Manual calculated hours (or manual FTE) always takes precedent over system-computed hours/FTE. However, the system computes the hours in the same way, whether based on manual or calculated hours.

CX contains two windows that display all faculty workload information for an instructor. To access these windows, you must first access the Instructor Information screen, locate the desired instructor, and then select **Options**.

Access the Instructor Schedule window by selecting **Schedule** from the Options window. The full schedule for the instructor appears. The faculty workload information appears, including the FTE for each course taught.

Access the Instructor Load window by selecting **Load** from the Options window. The full instructor workload appears with a summary line for the session. Horizontally, the window displays the six accrual methods. Vertically, the window displays the faculty load types that apply to the instructor for the session.

Faculty Load Report

You can access the Faculty Load report from the Registrar: End of Session Reports menu by selecting the Faculty Load menu option. The Faculty Load report lists the load for all faculty, or selectively by division and department as specified in the parameter screen. This report also includes non-instructional assignments.

Faculty Noninstruction Report

You can access the Faculty Noninstruction report from the Registrar: End of Session Reports menu by selecting the Faculty Non-Instruction menu option. The Faculty Noninstruction report lists all noninstructional assignments for all faculty.

Cross-Listing Sections

Introduction

In *CX*, a cross-listed section is a section that shares the same days, times, campus, building, and classroom (but not always the same course number) with one or more sections. It is not necessary for the cross-listed sections to share the same course number, section number, or title. Cross-listed sections share a common Meeting record (mtg_rec).

Cross-listing can be useful when working with equivalent courses. (An equivalent course is a course using a level of instruction equal to that of another course with a different course name and number.) For example, the course EDU313 and PSY313 both have the same title of Educational Psychology, and are ordinarily taught in the same classroom at the same time by the same instructor. Cross-listing sections of EDU313 and PSY313 makes it possible for the registration process to enroll students from a common pool of seats, and to combine class and grade lists.

A Description of Cross-Listing

CX approaches cross-listing from the perspective that you intend to cross-list two or more sections that are scheduled to meet at the same time and place. *CX* specifically does not require that the menu user make a prior conscious decision to cross-list two sections, then enter data to support the cross-listing. *CX* provides a prompt for cross-listing whenever a menu user enters data that indicates a potential cross-listing condition exists.

You can best understand cross-listing by looking at the relationship that exists between several records that you create when you build the schedule of courses for a specific session and year. You can create a Section record (sec_rec) from a valid course. When you enter meeting information for cross-listed sections, the Course Entry program creates the following records for each section:

- One or more Meeting records (mtg_rec)
- One or more Section Meeting records (secmtg_rec)

The Section Meeting records exist solely to link Section and Meeting records. A meeting number (mtg_no) is the link between the secmtg_rec and the mtg_rec. When sections are cross-listed, each section has its own sec_rec and a secmtg_rec, but the sections share a common mtg_rec.

Cross-listing is logically session-dependent, but it is not catalog-dependent. That is, it is possible to cross-list a course and section from the undergraduate catalog with another course and section from the graduate catalog. You can cross-list a section that is to be announced (TBA), and thus has no Meeting record, with another section that has a regular meeting pattern. The Meeting record that results from cross-listing the two sections contains the regular meeting information of the second session.

Cross-Listed Setup

You do not have to set up any macros, includes, or tables for cross-listing to occur. However, you must use the columns, options, and commands of the section and meeting screens and windows properly to create the appropriate relationships between records.

Section Record Screen

The cross-listing process begins with the Section Record screen. For more information on how to complete the fields on the Section Record screen, see *Using the Course/Class Schedule Screens and Windows* in the *Course/Class Schedule User Guide.*

The Cross Listing field on the Section Record screen is a two-part field that specifies whether a class list can be printed separately for cross-listed sections and whether a seat in one section can be exchanged for a seat in a cross-listed section.

The following list describes what happens when you enter a code in the Cross Listing field:

- If the section in the Section Record screen is cross-listed with another section and you want to print a separate class list for the section in the Section Record screen, enter a “Y” in the first part of the Cross Listing field.
- If the section in the Section Record screen is cross-listed with another section, and you do not want to print a separate class list for the section in the Section Record screen, enter an “N” in the first part of the Cross Listing field.
- If the section in the Section Record screen is cross-listed with another section, and you want to take an available seat from this section for use by the other section, enter a “Y” in the second part of the Cross Listing field.
- If the section in the Section Record screen is cross-listed with another section, and you do not want to take an available seat from this section for use by the other section, enter an “N” in the second part of the Cross Listing field.
- If the section in the Section Record screen is cross-listed with another section, and you want to take an available seat from this section for use by the other section, but you want a warning message to appear, enter a “W” in the second part of the Cross Listing field.

Note: At least one of the sections in a cross-list must contain a “Y” in the first part of the Cross Listing field for the system to produce class lists. All sections in a cross-list never should contain an “N” in the first part of the Cross Listing field. The code in the second part of the Cross Listing field has no effect on processing if the section is not cross-listed.

Meeting Schedule Window

Select **Meeting** from the Section Record screen to access the Meeting Schedule window. For more information on how to complete the fields on the Meeting Schedule window, see *Using the Course/Class Schedule Screens and Windows* in the *Course/Class Schedule User Guide*.

Facility Conflicts Window

If you create another Course, Section, and Meeting record with the same or overlapping days and times and in the same facility, the system recognizes that a facility conflict exists.

For more information on how to complete the fields in the Facility Conflicts window, see *Using the Course/Class Schedule Screens and Windows* in the *Course/Class Schedule User Guide*.

When a facility conflict exists, you can do one of the following:

- Select the command, **Select Meeting Record to Cross-List**, to create the cross-listed section. If the meeting pattern is not exactly the same, the meeting pattern of the initial section meeting determines the meeting pattern of the new combined Meeting record.
- Select **Cancel** to cancel the process and to indicate that the facility conflict was inadvertent and that you did not intend to cross-list.

Cross-Listed Sections Window

After you create the cross-listing relationship, select **Xlist** from the Meeting Schedule window. The system displays the Cross-Listed Sections window, which contains a complete listing of the courses and sections that are cross-listed with the course and section in the Section Record screen. Notice that the word "CROSSLISTED" appears in the upper right of the Section Record screen for any cross-listed section.

For more information on how to complete the fields on the Cross-Listed Sections window, see *Using the Course/Class Schedule Screens and Windows* in the *Course/Class Schedule User Guide*.

After you have completed the cross-listing process, select **Meeting** from the Section Record screen. The Meeting Schedule window appears containing a listing that indicates whether any of the meetings are part of a cross-listing. When you see an asterisk (*) in the column under the "X," the section for which you are adding a meeting schedule is cross-listed with one or more sections. You can view a screen showing all the cross-listed sections by selecting **Xlist**.

Facilities Check

The cross-listing process also includes a facilities check, in which the system compares the total number of seats (max_reg) for the combination with the available seats in the classroom. The system displays a window containing a warning message when there are insufficient seats for the combined total number of seats (max_reg). The warning message indicates that the menu user should update the number of seats (max_reg) in the Max Registration field on the Section Record screen. The system does not make the adjustment automatically.

Effects of Cross-Listing

The following list describes the effects of cross-listing on three different components in the registration process:

Registration

The cross-listing process enables the menu user to control the total enrollment in the combined sections by pooling the number of available seats. Jenzabar, Inc. recommends that the menu user set the Max Registration field in each of the Section Record screens so that the total number of seats for the combined sections is at the desired level, or at the capacity of the facility.

As the institution continues with registration, the system increments the enrollment counts for each section as seats are taken, and updates the section status from "O" for open to "C" for closed when the section reaches maximum capacity.

It is at this point that the pooling feature of cross-listed sections comes into use. If the second part of the Cross Listing field in the Section Record screen is set to "Y" or "W," a registration request for a closed section takes an available seat from an open section. To maintain the integrity of the combined total enrollment, the system decreases by one seat the Max Registration field (max_reg) of the section from which the seat was taken. The Max Registration field of the section that took the seat increases by one and remains closed.

Class Lists and Grade Lists

Both the Registration List (*reglist*) and ACE reports for class lists use the Cross Listing field on the Section Record screen, although you can use an option ("-i") in *reglist* that ignores the code to print separate class lists. Also in *reglist*, you can use the following options for cross-listed sections:

- Add the word “Sec:” before the section number in the upper right side of the form.
- Identify which cross-listed section each student is registered in and when combined lists are produced, without repeating the full department, course, and section identifier.

Cross Listed Course report

The Cross Listed Course report, available from the Registrar: Course/Class Reports menu, identifies and describes cross-listed courses for a specific session. The report shows full course and section identifiers, catalog, title, instructor, and full meeting information for each course and section that is cross-listed with the main listing. The fields from the Section Record screen are included in the listing, as well as the meeting number (mtg_no) that links the secmtg_rec and the mtg_rec.

Setting Up and Using Bridged Sections

Introduction

A bridged section is a section linked to a similar section in one or more subsequent sessions. A set of bridged sections, correctly created, allows the Registration Entry program (*regent*) to add automatically all bridged sections of a course for a student, beginning with the first session of the bridged sections.

Process for Building Bridged Sections

The following list describes the overall process involved in building bridged sections:

1. The Registration Entry program activates the Bridge field (sec_rec.brdge_ord) when the number in this field is greater than 0. The course number (sec_rec.crs_no) and section number (sec_rec.sec_no) must be the same in each session for each bridged course.
2. If the value in the Bridge field is greater than 0, the Registration Entry program searches for the Section Bridge record (secbrdg_rec) that points to the next session in which the section is offered.

   Note: If the Registration program cannot find the correct combination of values, an error message appears. Activate the bridged section feature using the enable macro, ENABLE_FEAT_BRIDGED_CRS, located in the directory path $CARSPATH/macros/custom/student.
3. The Registration Entry program enrolls the student in all linked sessions, but created Registration records (reg_rec) only in the section containing the Section record bridge order (sec_rec.brdg_ord).
4. An institution must complete all billing and refunding in the first bridged session according to the standard academic and standard refund calendar or the alternate academic and alternate refund calendar.

   Note: If you want to drop or void a bridged course, you must drop or void the course in the first bridged session.
5. You can define all attributes of the bridged section based on the institutional policy related to the use of bridged sections. The only requirement for matching data between Section records is through the index sec_prim, which matches course number, catalog, year, session, and section number.
6. Each bridged section appears on the appropriate session of the transcript or grade change mailer as if it has been added to that session through the Registration Entry program.

Section Record Screen

The Bridge field of the Section Record screen must reflect the correct order of sessions related to this section. A zero, the default, causes further processing, and the system assumes this is not a bridged section.

You must create subsequent Section records in sessions linked to the original bridged section prior to any registrations in this group of bridged sections.

Entering the value in the Bridge field for each bridged section in each session is a manual process.

Section Bridge Record Window

The Section Bridge Record window points to the previous or next session for the Registration Entry program. You must complete the Previous Session and Previous Year fields or the Next Session and Next Year fields for each bridged section prior to using the bridged sections in Registration Entry.

If an institution does not complete the Section Bridge Record window correctly, Registration menu users receive an error message indicating that the system loaded a missing Section Bridge record for a given course and section.

Full-Time Equivalency Status (FTES) Processing

Introduction

The FTES reporting performs the following processing functions:

- Creates and maintains the AAM
- Creates and maintains the census dates
- Calculates student contact hours for each section of a course
- Generates the CCFS-320 report and the supporting detail reports

Prerequisites

Before reviewing the FTES information on the following pages, make sure that you understand the relationship between the records listed below:

The processes described in the following pages relate to the following records:

- Academic record (stu_acad_rec)
- Course Detail record (crsdtl_rec)
- Course record (crs_rec)
- Course Work record (cw_rec)
- Meeting record (mtg_rec)
- Program Enrollment record (prog_enr_rec)
- Section Detail record (secdtl_rec)
- Section record (sec_rec)

How to Access FTES Process Information

Using the *CX* menu, access the Registrar: Session Processing menu. Select ADA Processing. (ADA stands for Average Daily Attendance.) The system displays the ADA Processing menu that contains information on processes and reports used in FTES processing.

FTES Records

The system uses the following Informix records to maintain data needed for FTES processing:

- Attendance record (ada_rec)
- Attendance Totals record (adatot_rec)

The directory path $CARSPATH/schema/student contains a description of the fields located in both the ada_rec and adatot_rec.

FTES Macros

All of the macros used by the FTES reporting system are located in the directory path $CARSPATH/macros/custom/student.

FTES Reporting Periods

The CCFS-320 report is due three times a year. The following list describes the FTES reporting periods for the CCFS-320 report:

Period 1

Your institution submits the CCFS-320 report on January 20, covering June 1 through December 31 of the previous year.

Period 2

Your institution submits the CCFS-320 report on April 30, covering June 1 of the previous year through April 15 of the current year.

Period 3

Your institution submits the CCFS-320 report on July 30, covering June 1 of the previous year through June 30 of the current year.

The FTES Reporting Process

The following list describes the overall process for FTES reporting:

1. The Course Entry program (*crsent*) maintains the AAM and census dates when any changes occur that can affect the AAM or census dates in the Meeting record or the secdtl_rec.ada and crsdetl_rec.ada fields.
2. Run the C code program called *adacalc* to calculate and update the enrollment and contact hours values for all Attendance records (ada_recs) for a specified session and year. Update the ada_rec for each section.

   Note: The *adacalc* program also recalculates the AAM and census dates and updates these dates if necessary (i.e., if the values change due to manual changes made outside of *crsent*).

3. Run the Annualizer report to compare period meeting time totals and annual meeting time totals and calculate a multiplier to be used to adjust period totals to annual estimates.

   Note: The Annualizer reports for periods 1 and 2, while providing actual accrued student contact hours for the period, are estimates of the FTES for the entire academic year.

4. Run the script called *adatot* to create Attendance Totals records (adatot_recs).

   Note: The adatot_recs contain the total values for each section and for each attendance accounting method at each site. The adatot_recs are necessary to generate the CCFS-320 report. The figures in the Annualizer report are used as parameters to the adatot script.

5. Run the *adarpt* program to generate the CCFS-320 report for each site and for the district.
6. Run the various FTES detail reports any time after you run the *adacalc* program to verify the census values on a section-by-section basis.

   Note: You can run the FTES detail reports using any of the following example criteria:

   - AAM
   - Division
   - Only bridged courses
   - Off-campus courses
   - SAM code

Attendance Accounting Method (AAM)

Introduction

The Attendance Accounting Method is an integral part of the CCFS-320 report because the system calculates the student contact hours in a different manner depending on the attendance accounting method for a section of a course.

AAM Types

There are five attendance accounting methods reported in the CCFS-320 reports. The following list describes each method type and describes its corresponding value:

Note: The term *contact hours* in the following table refers to the value in required parts II through V of the CCFS-320 report. The contact hours value is not the FTES value. The system uses the raw contact hours value to calculate the FTES value. For more information on each method type, see parts II through VI of the CCFS-320 report.

Weekly Census

The system calculates the weekly contact hours using the following information: the hours the section meets per week multiplied by the student enrollment as of the session census date.

Daily Census

The system calculates the contact hours using the following information: the total hours the section meets per day, multiplied by the number of days the section meets during the session, multiplied by the student enrollment as of the section census date.

Positive Attendance Census

The system calculates the contact hours using the following information: the total number of hours the section meets during the session, multiplied by the student enrollment, minus the student absent hours.

Note: The system further categorizes the positive attendance sections as credit and noncredit sections.

Work Experience/Independent Student Weekly Census

The system calculates the contact hours using the following information: the credit hours per section multiplied by the student enrollment as of the section census date.

Work Experience/Independent Student Daily Census

The system calculates the contact hours using the following information: the credit hours per section, multiplied by the term length multiplier, multiplied by the student enrollment as of the section census date.

AAM Calculation Logic

The following list describes the phases in the overall process for calculating the AAM. The system calculates the AAM by processing the information in each phase. When the system finds an "if" condition true for a given section, the system updates the section's ada_rec with the appropriate AAM, and AAM processing for that section is complete.

1. If a section is ADA-exempt, then the accounting method (ada_rec.acct_meth) is exempt and the code is “EX.”

   Note: A section is ADA-exempt if either of the following is true:
   - The value in the sec_rec.ada is “E”
   - The sec_rec.aua is blank and the crs_rec.ada is “E”

2. If a section is a bridged section, then the accounting method (ada_rec.acct_meth) is positive

attendance, and the code is “PA.”

Note: A section is a bridged section if the sec_rec.brdg_ord does not equal 0. A bridged section is in its final session if the secbrdg_rec.next_sess field is blank.

3. If a section is a bridged section and is not in its final session, then the accounting method (ada_rec.acct_meth) is bridged and the code is “BR.” Sections with an accounting method of “BR” are not included in the CCFS-320 report.

   Note: A section is a bridged section if the sec_rec.brdg_ord does not equal 0. A bridged section is not in its final session if the secbrdg_rec.next_sess field is not blank.

4. If a section is a non-credit section, then the accounting method (ada_rec.acct_meth) is positive attendance non-credit, and the code is “PN.”

   Note: A section is a non-credit section if either of the following is true:

   - The code in the sec_rec.cred field is “N”
   - The sec_rec.cred field is blank, and the crs_rec.cred field is “N”

5. If a section is an independent study section and a standard-term section, then the accounting method (ada_rec.acct_meth) is independent study/work experience weekly census, and the code is “WI.”

   Note: A section is an independent study section if the macro IM_INDEP_STUDY in the Meeting record (mtg_rec.im_ contains an instructional method of “IN.” A section is a standard-term section if sec_rec.weeks is greater than SEC_MIN_WKS.

6. If a section is an independent study section and a short-term section, then the accounting method (ada_rec.acct_meth) is independent study/work experience daily census, and the code is “DI.”

   Note: If a section is an independent study section and a short-term section, the accounting meeting record (mtg_rec.im) contains an instructional method of “IN.”

7. If the system designates a section as positive attendance, then the accounting method (ada_rec.acct_meth) is positive attendance, and the code is “PA.”

   Note: The system designates a section as positive attendance if either of the following is true:

   - The sec_rec.ada is “P”
   - The sec_rec.ada equals ‘ ,’ and the crs_rec.ada value is “P”

8. If a section is not a co-terminus section, then the accounting method (ada_rec.acct_meth) equals positive attendance, and the code is “PA.”

   Note: A section is not a co-terminus section if the sec_rec.beg_date is less than the acad_cal_rec.beg_date, or the sec_rec.end_date is greater than the acad_cal_rec.end_date.

9. If a section is less than full term, and the section meets less than 5 days, then the accounting method (ada_rec.acct_,eth) is positive attendance, and the code is “PA.”

   Note: A section is less than full term if the sec_rec.weeks is less than 18.

10. If a section is less than full term, and the section is not irregular, then the accounting method (ada_rec.acct_meth) is daily census, and the code is “DC.”

Note: A section is less than full term if the sec_rec.weeks is less than 16.

11. If a section is less than full term, and the section is irregular, then the accounting method (ada_rec.acct_meth) is positive attendance and the code is “PA.”
12. If a section is not irregular, then the accounting method (ada_rec.acct_meth) is positive attendance, and the code is “PA.”

    Note: A section is not irregular if the SEC_MEETING_IRREGULAR_WC (section key) is regular.

Macros Used in AAM Calculations

The following list further defines how the system determines regular and irregular section patterns:

Irregular Short-Term Sections

Daily census procedure sections must meet regularly with respect to the number of hours during each scheduled day. This means that the section must be scheduled to meet the same number of hours on each day it is scheduled to meet. The meetings do not have to occur at the same time (i.e., have the same beginning and ending times) on each scheduled day, as long as the same number of hours are scheduled for each day.

Note: The processing used to determine whether short-term sections are irregular is to internally store the hours it meets per day for each day in the session that it meets. Then the system compares each day’s hours against each other day’s hours. If the hours values are all the same, the section is regular. If any one of the hours values is different, the section is irregular. If a section includes any arranged time meetings, the section is irregular.

Irregular Long-Term Sections

Weekly census procedure sections must meet regularly with respect to the number of days of the week and the number of hours the course meets on each scheduled day. *CX* interprets this to mean that the section must be scheduled to meet the same number of hours during each week (including arranged hours) that it is scheduled to meet. A section is not required to meet on the same days each week, nor is it required to meet at the same time or day on each meeting day.

Note: The processing used to determine whether long-term sections are irregular is to internally store the hours it meets per week, for each week the section meets. Then, the system compares each week’s hours against each other week’s hours. If the hours values are all the same, the section is regular. If any one of the hours values is different, the section is irregular.

FTES Census Date Calculation

Introduction

The system uses the census dates to determine whether a student is currently enrolled in a course. A student is considered enrolled in a course if the student enrolled prior to or on the census date.

Prior to the 1992-1993 FTES reporting year, an institution was required to report on two census dates for the CCFS-320 report. However, beginning in the 1992-1993 FTES reporting year, an institution reports on only one census date.

All courses, excluding courses that are TBA and courses that are not daily census, obtain the census date from the Academic Calendar record for the session. A TBA course has no days, times, building, or room assigned.

If a course is TBA, and a menu user at your institution does not enter a code in the Census Date field on the ADA screen in Course Entry, and the menu user exits the section, the FTES processing defaults in the census date from the Academic Calendar record. The menu user may change the census date at any time. The FTES processing does not overwrite any value in the Census Date field if the field is blank or null.

Census Date Calculation

The following list describes the overall process for calculating the census dates for census courses that are daily census and not TBA courses:

1. The system defines the percentage of meetings that must occur prior to the first and second census dates using the following macros:
   - ADA_CENSUS_DATE1_PCT
   - ADA_CENSUS_DATE2_PCT
2. The system calculates the number of meetings to occur prior to the census date using the following information: the total number of meetings, multiplied by the percentage in phase 1, divided by 100.
3. The system determines the census date using the first date on or after the number of meetings has occurred that is not a “non-teaching” day, a weekend, or, in the summer only, is not a Friday.

FTES Contact Hours Calculation

Introduction

Your institution sets the enrollment contact hours for each section according to the type of AAM used for the section.

Enrollment Types

The following are the four types of enrollment:

- Resident
- Nonresident
- Day
- Evening

Values for the First Census Date

Census values for the first census date (res_enr1 and nonres_enr1 in the ada_rec) are based on the number of students enrolled in the section as of the first census date (census_date1).

Criteria for an Evening Course

A course is considered an evening course if it meets after 4:29 p.m. any day, or if it meets on a Saturday or a Sunday. A TBA course in which you cannot determine whether it meets after 4:29 p.m. any day is considered a day course.

Weekly Hours Calculation

Weekly census calculations are based on the number of hours that a course meets per week. The system determines the calculation by accumulating all days and hours that a section meets and totaling the hours.

The following is an example from the Meeting record:

Days	Hours of Instruction
-M-T-F-	1
-M-W-	2

The result is that the value for total hours per week is 7 hours (3 * 1 + 2 * 2).

Total Section Hours Calculation

Daily census and positive attendance calculations are based on the total number of hours a section meets over the entire session. The system calculates these values using the values in the ManHrs field (mtg_rec.tot_hrs) and the CalcHrs field (mtg_rec.calc_tot_hrs) in Course Entry.

If the value in the tot_hrs field is greater than zero, then the system uses the value. If the value in the tot_hrs field is not greater than zero, the system uses the calc_tot_hrs value.

A menu user at your institution can manually enter the total hours value (tot_hrs). The system automatically calculates the calculated total hours (cal_tot_hrs) value in Course Entry each time the menu user adds or updates meeting information.

The system figures the calculated total hours by using the following information:

- The begin and end dates of the Meeting records
- The days the meeting occur (from the mtg_rec.days field)
- The Non-teaching Date table (nonteach_table) that defines the days that may not be counted as meeting days

The following example is from the Meeting record:

Manual Hours	Calculated Hours
(tot_hrs)	(calc_tot_hrs)
20	23.5
0	45

The result is that the total section hours equals 65 (20 + 45)

Student Enrollment

The system determines student enrollment according to the census date for any section that is not positive attendance. A student is considered enrolled at the time of the census date if the student has a "registered" status prior to the census date. (The student must have a Registration record [reg_rec] with a status of "R" and an effective date on or prior to the census date.)

Residency

The system determines residency by comparing the state code in the INST_ST macro to the state code in the student's Program Enrollment record (prog_enr_rec). The INST_ST macro is located in the following directory path: $CARSPATH/macros/custom/common.

Apprentice Sections

The system does not include a student in FTES values if both of the following conditions are met:

- The section is an apprentice section (the value in the field secdtl_rec.apprent is “Y”).
- The student is an apprentice student (the value in the stu_acad_rec.apprent field is “Y”).

CX uses a separate reporting for apprentice students enrolled in apprentice sections.

Census Value Calculations for Weekly Census Courses

For weekly census sections, the census value is the total hours the section meets per week multiplied by the number of students enrolled at the time of the given census date. The system calculates the total hours per week using the number of days from Meeting records that have the same begin and end dates.

The following is an example from Meeting records for a section.

Days	Hrs of Instruction	Begin date	End date
-M-W-	1.0	09/15/0X	12/30/0X
-T-	1.0	09/15/0X	12/30/0X

The result of the above example is that the total hours per week is three since Monday, Tuesday, and Wednesday count for one day each. The system stores the census value in the ada_rec.res_enr1 and ada_rec.nonres_enr1 fields. The value in the ada_rec.res_enr1 field indicates the number of state resident students reported in the first enrollment period. Part II of the CCFS-320 report contains the total census value.

Census Value for Calculations for Daily Census Courses

For daily census sections, the census value is the total hours the section meets (as determined by the sum of the total hours of all the Meeting records) multiplied by the number of students enrolled at the time of the given census date.

The system stores the census value for daily census sections in the ada_rec.res_enr1 and ada_rec.nonres_enr1 fields. Part II of the CCFS-320 report contains the total census value for daily census sections.

Census Value Calculations for Positive Attendance Courses

The following list describes the overall process for calculating the census value or positive attendance sections for both credit and noncredit sections:

Note: The system does not include students voided from a course in calculating the census value for positive attendance. The census value for positive attendance sections is stored in the ada_rec.res_enr1 and ada_rec.nonres_enr1 fields. Part IV of the CCFS-320 report contains the total census value for positive attendance sections.

1. The total Meeting record hours for the section multiplied by the number of phase 2.
2. The total number of students who were enrolled in the section at some point and did not drop the course prior to the section begin date, and whose absent hours were updated (i.e., the value in the cw_rec.absnt_upd field is “Y”), minus the number in phase 3.
3. The sum of all absent hours for all students as maintained in the cw_rec.absnt_hrs field.

Weekly Census Value Calculations for Independent Study/Work Experience Courses

The weekly census value for independent study/work experience sections is the credits of the section (defined in the sec_rec.hrs field) multiplied by the number of students enrolled at the time of the given census date.

The system stores this value in the ada_rec.res_enr1 and ada_rec.nonres_enr1 fields. Part V of the CCFS-320 report contains the total weekly census value for independent study/work experience sections.

Daily Census Value Calculations for Independent Study/Work Experience Courses

The daily census value for independent study/work experience sections is the credits of the section (defined in the sec_rec.hrs field) multiplied by the term length multiplier (currently 17.5), multiplied by the number of students enrolled at the time of the census date.

The system stores this value in the ada_rec.res_enr1 and ada_rec.nonres_enr1 fields. Part VI of the CCFS-320 report contains the total daily census value for independent study/work experience sections.

Census Value Calculations for Exempt Courses

The system maintains the census values for exempt sections as though these sections are weekly census sections.

FTES Reports

Introduction

The following pages describe the types of FTES reports that you can run.

Annualizer Report

The Annualizer report is used to calculate projections for future funding based on information from a current reporting period. An institution can run the Annualizer report for a site and an accounting method to project the next two reporting periods.

The begin and end dates, site, and period are the parameters for the Annualizer report. The begin and end dates are the dates for the yearly academic calendar that your institution is annualizing. The begin date is normally 7/1/XXXX and the end date is 6/30/XXXX+1. You must run the Annualizer report for each site.

The system uses the specified period to determine which sections from which period are being compared to the yearly total. The only valid values are 1 and 2, because by the third period, an annualizer is not necessary since the academic year is complete.

Annual Total Calculation on the Annualizer Report

For the annual total calculation on the Annualizer report, the system takes all weekly and daily census sections whose census date falls within the specified period. The Annualizer report takes all positive attendance sections whose section date falls within the specified period.

Period Totals and Contact Hours

For period totals on the Annualizer report, if the period equals 1, the system selects weekly and daily census sections whose census date falls between 7/1/XXXX and 12/31/XXXX. The system also selects positive attendance sections whose section date falls between 7/1/XXXX and 12/31/XXXX. The total contact hours for each section are derived from a calculation in the Meeting records.

The system compares the total contact hours to the total contact hours for all sections for the entire year, with the exception of summer sections.

For each attendance method, the system calculates the annualizer value by dividing the total yearly contact hours by the period contact hours.

The Annualizer and 320 Reports

When your institution submits CCFS-320 reports for periods 1 and 2, you must annualize the FTES figures for Part I of the CCFS-320 report. That is, the FTES figures must reflect an estimate of what the year's totals will be based on an annualizer value.

For example, if all weekly census sections that occur in period 1 have a total of 11,320 meeting hours and the total number of meeting hours for weekly census sections that occur any time in the entire FTES academic year (July 1- June 30) is 28,000, then the annualizer figure for period one for weekly census courses is 28,000/11,320 or 2.4735.

Criteria for Section Records

When selecting sections for the Annualizer report, the system uses the following criteria for section records:

- The census reporting date must fall within the begin and end dates used as parameters in the Annualizer report for weekly or daily census sections
- The section cannot be in the summer session (the sess value cannot be “SU”)

- The section must have a status other than “hold” or “cancel” as defined in the macros SEC_STAT_HOLD and SECC_STAT_CANCEL in the following directory path: $CARSPATH/macros/custom/student

Attendance Totals Records

The system totals the ada_recs for each attendance method for each site and stores the results in the adatot_rec. The system uses the annualizer values that the system already has calculated to create Attendance Totals records so that the values also can be stored in the Attendance Totals records.

The process for creating Attendance Totals records is an ACE report that writes an ISQL script. The ISQL script first removes all adatot_recs for the academic year that your institution is processing, and then adds adatot_recs for the period for all attendance methods for all sites.

The following list shows the logic for how the ACE report determines the term in which a particular section belongs if the section is not a summer session:

- If the census date is less than or equal to December 31, the term is the first primary term
- If the census date is less than or equal to April 15, the term is the second primary term
- If the census date is less than or equal to June 30, the term is the third primary term

The following list shows the logic for how the ACE report determines the term in which a particular section belongs if the section is a summer session:

- If the end date of the section is less than or equal to April 15, the term is a *summer intersession term* (for summer prior to the reporting period)
- Otherwise, the term is a *summer intersession term* (for the summer after the reporting period)

Criteria for Selecting Sections

When the system determines in which term a particular section falls, the system selects only courses that meet the following criteria:

- The section is positive attendance and the end date of the section falls between the begin and end dates of the academic year that your institution is processing (July 1 - June 30).
  - The section is weekly or daily census and meets the following criteria:
  - The section is not a summer section
  - The census date falls between the begin and end dates of the academic year that your institution is processing
- The section is weekly or daily census and meets the following criteria:
  - The section is a summer section
  - The section is the first summer session (the year of the section is the same as the begin year of the academic year that your institution is processing)
  - The census date falls after the begin date of the academic year (July 1)
- The section is weekly or daily census and meets the following criteria:
  - The section is a summer section
  - The section is the first summer session (the year of the section is the same as the begin year of the academic year that your institution is processing)
  - The menu user includes courses from the first summer session
  - The census date is less than the begin date of the academic year that your institution is processing.
- The section is weekly or daily census and meets the following criteria:
  - The section is a summer section
  - The section is the second summer session (the year of the section is the same as the end year of the academic year that your institution is processing)
  - The menu user includes courses from the second summer session

- The census date is less than the end date of the academic year that your institution is processing (June 30)

Other Totals in the Attendance Totals Record

In addition to storing term totals, the adatot_rec also stores totals on GAIN students, basic skill courses, in-service courses, and off-campus courses.

The following list describes how the system selects sections for GAIN students, and basic skills, in-service, and off-campus courses:

GAIN Students

For total GAIN FTES, the gain_enr1 field in the ada_rec accumulates the total. For a GAIN student enrolled in basic skills courses, the calculation includes the gain_enr1 field only if the section is a basic skills section (i.e., secdtl_rec.basic is "Y").

Basic Skills Courses

For basic skills courses excluding GAIN students, the system includes the res_enr1 field only if the section is a basic skills course (i.e., the secdtl_rec.basic field is "Y"). The system subtracts the gain_enr1 value from the res_enr1 value.

In-Service Courses

The system includes the res_enr1 field only if the section is an in-service course (i.e., the secdtl_rec.insvc field is "Y").

Off-Campus Courses

The system includes the res_enr1 field only if the section is an off-campus course (i.e., at least one Meeting record has an associated facil_table record in which the facil_table.on_campus field is "N").

In addition to the Attendance Totals record for each site, the system also creates district Attendance Totals records. The district Attendance Totals records store the values that are used in Parts II through VI of the CCFS-320 District report. The adatot_recs alone generate the final CCFS-320 reports.

CCFS-320 Site Report

A program in *CX* called *adarpt* uses the Attendance Totals records to print out the information required in the CCFS-320 Site report.

The parameters to the *adarpt* program are the begin and end dates (for the academic year that your institution is processing) and the period of the report (either 1, 2, or 3). You can generate a CCFS-320 Site report for each site and also generate the CCFS-320 District report. The information in the detail pages in parts II through VI comes from the adatot_rec.

The summary part (part I) of the CCFS-320 Site report comes from the appropriate detail sections (as given in the directions of the CCFS-320 Site report). The system divides the summary part value by 525 and multiplies the annualizer for the attendance method and the site when appropriate.

The system derives the values in part I of the CCFS-320 Site report as follows:

$$\frac{\text{Census value} * \text{multiplier} * \text{annualizer}}{525}$$

The census value is full student contact hours for sections that are other than weekly census. The census value is weekly student contact hours for weekly census courses. The multiplier is 17.5 for weekly census courses, and the multiplier is 1.0 for all other courses.

The system does not multiply summer ADA values by an annualizer because the summer ADA values are final when reported. When generating the final period report, the system does not use an annualizer because the year's totals are complete.

CCFS-320 District Report

The CCFS-320 District report is similar to the CCFS-320 Site Report, but the CCFS-320 District report provides totals across sites. The system generates this report at the same time as the CCFS-320 Site report using the same program (*adarpt*).

Note that the system computes all figures in Part I of the CCFS-320 District report for periods 1 and 2 using the annualizer values specific to each site. The system then totals these figures and the figures appear on the CCFS-320 District report.

Attendance Detail Reports

Detail reports exist that provide the census values, enrollment, and contact hours for specific sections. Use these reports to clarify data on the CCFS-320 reports.

Each of the Attendance Detail reports has the following parameters:

- Site
- Begin date
- End date
- Include first summer session
- Include second summer session

The system determines which Section records to include in the Attendance Detail reports in the same manner as it does for the Attendance Totals report. However, in the Attendance Detail reports, frequently the begin and end dates used in the report do not reflect the academic year, but only some portion of it.

Types of Detail Reports

The following list describes each Attendance Detail report used in FTES processing:

FTES Detail Report (adadtl)
Allows you to print section values by the attendance accounting method for sections that fall within a begin and end date specified as parameters to the report.

FTES Bridged Report (adabrdg)
Provides section values for bridged sections in the section's final term. Bridged sections are positive attendance sections and their FTES values are reported in the final term of the bridged series.

FTES Course Report (adacrs)
Provides section values for sections sorted by course prefix (i.e., the series of letters in the crs_no field prior to the first space).

FTES Division Report (adadiv)
Provides section values for sections sorted by course division.

FTES In-Service Report (adainsvc)
Provides section values for all in-service sections. A section is in-service if the secdtl_rec.insvc field is "Y."

FTES Off-Campus Report (adaoff)
Provides section values for all sections that meet off-campus. An off-campus section has at least one Meeting record indicating that the meeting occurs off-campus, and the associated facil_table record on_campus field is set to "N."

FTES Division Report (adadiv)

Provides section values for sections sorted by course SAM code (defined in the crsdtl_rec.sam field).

FTES Student Detail Report

Lists the student ID, name, and FTES value for each student included in the FTES total for the section.

SECTION 13 – COURSE/CLASS SCHEDULE MAINTENANCE PROCEDURES

Overview

Introduction

This section provides procedures you need to maintain the Course/Class Schedule product, including:

- Creating new catalog information from an existing catalog
- Creating new session information from an existing catalog

The maintenance procedures are organized in three groups in this section:

- Annual
- Session-based
- Ongoing

Definitions

The following defines the three types of maintenance procedures that you must perform in order to keep your database accurate and CX functioning properly.

Annual
You must perform these procedures annually.

Session-based
You must perform these procedures at the beginning, middle, or end of each session.

Ongoing
You must perform these procedures on an ongoing basis.

Process

This list shows the general phases for maintaining Course/Class Schedule. It identifies the order in which you should perform the maintenance procedures included in this section.

1. Create new catalog information from an existing catalog.
2. Create new session information from an existing session.

Session-Based Maintenance

Introduction

Course/Class Schedule does not require any session-based maintenance.

Annual Maintenance Procedures

Introduction

Course/Class Schedule does not require any annual maintenance.

Ongoing Maintenance Procedures

Introduction

These are the ongoing maintenance procedures in Course/Class Schedule.

- If the school has a new catalog each year, creating new catalog information from an existing catalog
- Creating for future sessions new session information from an existing catalog

Creating New Catalog Information from an Existing Catalog

Introduction

At the end of a session or the end of an academic school year, and institution might want to copy all of its courses from an existing catalog into a new catalog for the next school year. This process is called catalog rollover. Deciding when to perform the catalog rollover procedure is a policy decision that the institution makes.

The Catalog Rollover Process

The following list shows the phases in the overall process of rolling over course information from an existing catalog to a new catalog:

1. Create a new record in the Course Catalog table (cat_table).

   Note: The Course Catalog Table screen is a PERFORM screen that enables you to maintain the Course Catalog table. You must create a new record in this table for the Course/Class Schedule application to function correctly using the new catalog.

2. Run the SQL script called rollcat that creates a new catalog from an existing catalog.

   CAUTION: The institution can modify the rollcat script to include or exclude tables associated with the Course record (crs_rec). However, do not modify the rollcat script to roll over only a subset of the available course records because dependencies exist between the catalog rollover process and the session rollover process.

   CAUTION: After running the rollcat script, an institution can run the session rollover program called *rollsess*, which creates a new session for the new catalog. The *rollsess* program copies the section, meeting, and instructor assignment information from an existing session to a new session. The *rollsess* program expects to find existing Course records from the new catalog. If the rollcat script and the *rollsess* program do not use the same logic to roll over the Course records, then the *rollsess* program is unable to locate a Course record it expects to find, and it cancels the session rollover process.

The Rollcat SQL Statement

The Catalog Rollover menu option initiates an SQL script called rollcat that moves all Course records and corresponding records (in the Options window) from the old catalog year to the new catalog year. The SQL script is located in the following directory path: $CARSPATH/modules/regist/informers.

Before running the Catalog Rollover option, review the courses that you do not want to move to the new catalog.

The following list describes each catalog status and describes how the rollcat SQL statement processes each status:

Active (A)
When an "A" appears in the Course Status field on the Course Catalog screen, the process moves all Catalog records with the "A" status into the new catalog year.

Banked (B)
When a "B" appears in the Course Status field on the Course Catalog screen, the process does not create a Section record in the new catalog if the Catalog record is moved.

Inactive (I)

When an "I" appears in the Course Status field on the Course Catalog screen, the process does not create Catalog records for courses.

How to Roll Over the Course Catalog

If an institution does not want to roll over a course, change the status of the course to "Inactive" before performing the following procedure. The rollcat script does not copy an inactive course and any of its attached course-related records (e.g., Course Objectives record) from an existing catalog to a new catalog. Only those records associated with a course whose status is "Active" or "Banked" are copied over from an existing catalog to a new catalog.

The following list contains the steps to follow for copying course information from an existing catalog to a new catalog. Specifically, this list contains the steps for creating a new record in the Course Catalog table and running the rollcat script.

1. Access the Course Catalog table (cat_table) by selecting Course Catalog from the Table Maintenance: Registrar C menu. The Course Catalog Table screen appears.
2. Select **Add**.
3. Complete the fields on this screen. The system displays the information you enter in the fields.
4. Press the **<Esc>** key to save the information. The system saves the information.
5. Select **Exit** to exit the Course Catalog Table screen, and press **<Enter>.** The Table Maintenance Registrar (C) Menu appears.
6. Type **P** at the Enter Option prompt until the Student Management: Registrar Main menu appears.
7. Select **Course/Class Schedules**. The Registrar: Course/Class Schedule Menu appears.
8. Select **Catalog Rollover** and enter your password. The Catalog Rollover screen appears.
9. Enter the code for the institution's existing catalog (that you want to copy to a new catalog) in the Old Catalog field (e.g., UG90). The code for the existing catalog appears.

   Note: The existing catalog is made up of Course records and other course-related information (e.g., course syllabus). The rollover script uses the existing catalog as a template when creating the new catalog.
10. Enter the code for the institution's new catalog in the New Catalog field (e.g., UG95). The code for the new catalog appears.

    CAUTION: Do not enter a catalog name that already exists. Entering a catalog name that already exists may result in termination of the rollcat process.
11. Select **Finish**. The following occurs:
    - The system runs the rollcat script that copies the course catalog information from the existing catalog to the new catalog.
    - The system sends you an electronic mail message indicating that the rollcat script ran successfully.
    - The system creates an output file containing part of the rollcat script's execution.

Tables the Catalog Rollover Process Updates

When copying an existing catalog to a new catalog, the rollcat script creates new records for certain course catalog tables that contain course-related information.

The following list describes the course catalog tables for which the rollcat script creates new records:

- Articulation record (artic_rec)

Note: The rollcat script creates new records for this table only if this record exists at an institution.

- Course Catalog Abstract record (crsabstr_rec)
- Course Detail record (crsdtl_rec)

  Note: The rollcat script creates new records for this table only if this record exists at an institution.

- Course Objectives Text record (crsobj_rec)
- Course record (crs_rec)
- Course Requirement Text record (crsrequir_rec)
- Course Syllabus Text record (crssyll_rec)
- Course Teaching Qualifications record (crsqual_rec)
- Instructional Method record (im_rec)
- Section Requirements record (secreq_rec)

Notes:

- Only those secreq_recs that exist at course level (secreq_rec.sec.no is blank) are copied.
- The rollcat script rolls over only course-level records.

Tables the Catalog Rollover Process Does Not Update

The following tables are related to a specific catalog, but are not updated by the rollcat script. The rollcat script does not create new records for these tables.

Note: Rollover is not necessary for these tables due to their inherent nature and the CX applications that use the tables. For each of the following tables, an institution needs only one record for the catalog for which the record is initially attached. The Course/Class Schedule application applies the records in these tables to future catalogs.

- Course Equivalence record (crseq_rec)
- Course History record (crshist_rec)
- Noncatalog Equiv record (noncateq_rec)
- Prerequisite record (crsreqgrp_rec)
- Requisite Maintenance record (crsreq_rec)

Fields the Catalog Rollover Process Updates

During the catalog rollover process, the rollcat script modifies only one field value for the new catalog. This field is the Catalog field. For example, if an institution rolls over its existing 2000 undergraduate catalog (e.g., UG00) to the new 2001 undergraduate catalog (e.g., UG01), the script creates new table records and updates the Catalog field from the “UG00” code to “UG01” in each table.

Creating New Session Information from an Existing Session

Introduction

During a session, an institution might want to copy all of its session information from an existing session into a new session. This process is called session rollover. For example, during the Fall session of 2000, an institution may want to copy its Fall session information to the Fall session of 2001.

For each new session, you must create new sections and new meetings. Since sections and meetings are session-dependent, the session rollover process copies Section and Meeting records from an existing session to a new session.

CAUTION: If an institution creates a new session (e.g., Fall of 2000) that belongs in a new catalog (e.g., the 2000 undergraduate catalog), then create the new catalog first before creating the new session or else the *rollsess* program will not work properly. For the next session, the Fall of 2001, an institution does not need to create a new catalog when it creates the new Fall 2001 session. The institution can still use the 2000 undergraduate catalog. For more information on creating a new catalog from an existing catalog, see *Creating New Course Catalog Information From an Existing Catalog* in this manual.

The Session Rollover Process

Generally, an institution's Jenzabar coordinator performs the session rollover process. The following list shows the phases in the overall process of copying session information from an existing session to a new session:

1. Creates an Academic Calendar record (acad_cal_rec) for the new session.

   Note: The Academic Calendar record allows the *rollsess* program to assign beginning and ending dates to new Section and Meeting records and calculate total meeting hours and Full-time Equivalency (FTE) values.

2. Run the *rollsess* program, which copies the existing session information to a new session.
3. Manually update records in other tables that are linked to the Section Bridge record (secbrdg_rec).
4. Run the process to create section reference numbers since the *rollsess* program does not copy section reference numbers.

Parameters Used in the Session Rollover Process

An institution can run the *rollsess* program from the menu using the Session Rollover screen. However, if the institution wants to use any of the following parameters, which are not part of the CX base product, the institution can do either of the following:

- Modify the menu options associated with the Session Rollover screen located in the menuopt/regist/programs/rollsess directory path and add the desired parameter(s). For details on how to modify the menu options, see the *Getting Started User Guide*.
- Run the *rollsess* program from the UNIX shell.

The following list describes the parameters that an institution can use to control the tables used in the session rollover process:

Note: For more information on the “-a” and “-n” parameters, see *Preliminary Information: Non-Standard Sections* and *Preliminary Information: Canceled Sections* following this list.

-a

Include non-standard sections in session rollover.

-n
Exclude canceled section from session rollover.

-m
Exclude Meeting and Instructor records from session rollover.

-i
Exclude Instructor records from session rollover.

-d
Exclude Schedule Comment records from session rollover.

-r
Exclude section requirements from session rollover.

-x
Exclude Textbook records from rollover.

Preliminary Information: Sections Included in Session Rollover

The *rollsess* program does not use every section identified by the current year, session, and catalog that you specify in the Session Rollover screen at run time. Instead, only those sections associated with a course whose status is “Active” are copied over from the existing catalog to the new catalog. The *rollsess* program does not copy any sections or any records attached to the sections associated with a course having a status of “Banked” or “Inactive.”

Preliminary Information: Non-Standard Sections

The *rollsess* program always creates new Section and Meeting records using the standard beginning and ending dates defined in the Academic Calendar record. Non-standard sections have Section records whose beginning and ending dates differ from the dates defined in the Academic Calendar record.

If an institution uses non-standard sections in the session rollover process, the *rollsess* program generates a warning message to identify these non-standard sections. The institution must update the new records manually after running the *rollsess* program if the new sections are to remain non-standard.

Preliminary Information: Canceled Sections

If an institution includes canceled sections in the rollover process, the *rollsess* program creates the new section with an "Open" status. If the institution excludes canceled sections from the rollover process, *rollsess* generates a warning message for each canceled section it encounters.

How to Roll Over the Session

The following list contains the steps to follow for copying session information from an existing session to a new session. Specifically, this table lists the steps for creating a new Academic Calendar record for the new session and running the *rollsess* program.

Note: If an institution is creating a new session (e.g., Fall of 2000) that belongs in a new catalog (e.g., the 2000 undergraduate catalog), then create the new catalog first before creating the new session or else the *rollsess* program will not work properly.

1. Select Academic Calendar from the Table Maintenance: Registrar (A – B) to and then press **<Enter>** to access the Academic Calendar table. The Academic Calendar screen appears.
2. Select **Add**. The command line changes, and the cursor appears in the Academic Program field.
3. Complete the fields on this screen using the comment line as a guide.

4. Press **<Esc>** to save the information. The system saves the information to the database.
5. Select **Exit** to exit the Academic Calendar screen, and press **<Enter>.** The Table Maintenance Registrar (A-B) Menu appears.
6. Type **P** at the "Enter Option" prompt until you are at the Student Management: Registrar Main Menu. The Student Management: Registrar Main Menu appears.
7. Select Course/Class Schedules. The Registrar: Course/Class Schedule Menu appears.
8. Select **Session Rollover** and enter your password. The Session Rollover screen appears, with the cursor in the Old Academic Session field.
9. Complete the fields on the Session Rollover screen. The system displays the information you enter in the fields.
10. Select **Finish**. The Output Parameters and Scheduling window appears.
11. Complete the fields in the Output Parameters and Scheduling window and select **Finish**. The system runs the *rollsess* program.

    Note: See the *Getting Started User Guide* for information about how to complete the Output Parameters and Scheduling window.

12. Perform the procedure, *How to Create New Section Reference Numbers,* in this section.

How to Resolve Partial Schedule Creation

If an institution runs the *rollsess* program, and an error occurs that results in creating only a partial schedule, do the following:

- Resolve the error
- Rerun the *rollsess* program using the same field values in the Session Rollover screen and the parameters in the Output Parameters and Scheduling window.

CAUTION: Do not rerun *rollsess* if you have accessed the Course/Class Schedule application and made changes to affect cross-listed or Course records.

Making Course record status changes could result in an error message regarding a missing Course record and cause *rollsess* to terminate early.

How to Create New Section Reference Numbers

If an institution uses reference numbers, the *rollsess* program does not create new section reference numbers. After running *rollsess*, the institution must create section reference numbers using a menu option.

The following list contains the steps for creating reference numbers for the new sections in the new session:

1. Access the Registrar: Course/Class Schedule Menu. The Registrar: Course/Class Schedule Menu appears.
2. Select Create Reference Numbers, and enter your password. The Create Reference Numbers screen appears.
3. Complete the fields on this screen. The system displays the information you enter in the fields.
4. Select **Finish**. The Output Parameters and Scheduling window appears.
5. Complete the fields on the Output Parameters and Scheduling window. The system displays the information you enter in the fields.

   Note: See the *Getting Started User Guide* for information about how to complete the Output Parameters and Scheduling window.

6. Select **Finish**. The system creates the section reference numbers for the new session.

Tables the Session Rollover Process Updates

When copying an existing session to a new session, the *rollsess* program creates new records for certain session-related tables. The following list describes the tables for which the *rollsess* program creates new records:

- Attendance record (ada_rec)

  Note: The rollcat script creates new records for these tables only if these records exist at an institution.

- Alternate Academic Calendar record (altcal_rec)

  Note: The rollcat script creates new records for these tables only if these records exist at an institution.

- Alternate Refund record (altrfnd_rec)

  Note: The rollcat script creates new records for these tables only if these records exist at an institution.

- Instructional Method record (im_rec)

  Note: The rollcat script rolls over only course-level records.

- Instructor record (instr_rec)
- Meeting record (mtg_rec)
- Schedule Comment record (schd_comment_rec)
- Section record (sec_rec)
- Section Detail record (secdtl_rec)

  Note: The rollcat script creates new records for these tables only if these records exist at an institution.

- Section Meeting record (secmtg_rec)
- Section Requirements record (secreq_rec)
- Textbook record (textbk_rec)

Tables the Session Rollover Process Does Not Update

The rollsess process does not update the Section Bridge record due to the inherent nature of the table. Therefore, *rollsess* does not create a new record for the Section Bridge record.

A bridged section spans across more than one session. Rollover is not necessary for this record due to its inherent nature and the *CX* applications that use the record. For the Section Bridge record, your institution needs only one record for the session for which the record is initially attached. The Course/Class Schedule application applies the record in the Section Bridge record to future sessions.

Fields the Session Rollover Process Updates

During the session rollover process, *rollsess* modifies the following fields:

- instr_rec.fte FTE (Fulltime Equivalency)
- mtg_rec.beg_date (Begin Date)
- mtg_rec.calc_tot_hrs (Calculated Total Hours)
- mtg_rec.end_date (End Date)
- sec_rec.altrfd_no (Alternate Refund Number)
- sec_rec.brdg_ord (Bridged Order)
- sec_rec.cat (Catalog)
- sec_rec.end_date (End Date)

- sec_rec.ref_no (Reference Number)
- sec_rec.reg_num (Number Registered)
- sec_rec.stat (Status)
- sec_rec.stat_date (Status Date)
- sec_rec.wait_num (Waiting Number)

Using the Update Registration Process

Introduction

The Update Registration Record menu option updates any new bill codes, fee codes, or tuition codes from Course records or Section records if the corresponding field in the Registration records is different or blank. You can access this menu option from the Registrar: Course/Class Schedule menu.

Use the Update Registration Record menu option if a Section record was created that did not contain the correct billing codes structure, or a change was made in the codes after registration begins.

The Updreg SQL Script

The Update Registration Record option initiates an SQL script, updreg, that creates or revises tuition, fee, and bill codes. The SQL script is located in the following directory path: $CARSPATH/modules/stubill/informers.

Special Notes

Special notes:

- You must run the updreg SQL script when tuition, bill, or fee codes in the Course or Section record are added after registration begins
- Blank codes in the Course or Section record do not affect an update to the corresponding Registration record
- The process overwrites any tuition, fee, or bill codes unique to the student

SECTION 14 – PROGRAM ERRORS AND CRASH RECOVERY

Overview

Introduction

This section provides the following:

- A list of fatal errors
- Crash recovery procedures

Note: Refer to the *Course/Class Schedule User Guide* for a list of the more common status, field error, and warning messages that can occur when menu users execute the programs in Course/Class Schedule.

Fatal Errors

Fatal errors can occur in any program in Course/Class Schedule. If a fatal error occurs, use the following procedure:

1. Reload the program. Erroneous or corrupt data might cause the fatal error.
2. Try a different student's data to see whether errors occur in that process. If the error does not recur, the data for the first student may be causing the error.

Serious and Fatal Errors

Messages Received

The following lists alphabetically the serious and fatal error messages that can appear when you are using Course/Class Schedule. Future revisions of this document will go into greater detail regarding fatal error messages in other Jenzabar, Inc. Course/Class Schedule applications.

Note: To locate these errors, enter:

1. The source program (e.g., cd src/regist/crsent)
2. vic _FATAL

%s

Error message returned from chk4confl.ec, dbsec.ec, entartic.ec, entcrs.ec, entcrseq.ec, entcrshis.ec, entcrsqf.ec, entctrlt.c, entfac.ec, entfacact.ec, entfacil.ec, entfacld.ec, entfacsch.c, entgetset.c, entim.ec, entimv.c, entinstr.ec, entmtg.ec, entnoncat.ec, entpick.c, entrequis.ec, entrmsch.c, entsccm.ec, entsec.ec, entsecreq.ec, enttextbk.ec, entxmtga.ec, entxsecv.c, init.c, instr.ec, libscr.c, lkuproom.ec, main.ec, mntcatalog.c, mntcrs.c, mntfac.ec, mntfacil.ec, mntmtg.ec, mntsec.ec, mntseccan.ec, mtg.ec, sec.ec, secmtg.ec, sess.ec, sessyr.c, xlist.ec.

%s\n\n%s\n

Error message returned from main.ec.

An error occurred during the call to structinit(%s):\n %s

Error message returned from entfacact.ec.

Begin Work error %d

Error message returned from entartic.ec, entblob.ec, entcrs.ec, entcrseq.ec, entcrshis.ec, entcrsqf.ec, entfac.ec, entfacact.ec, entfacnia.ec, entfacqf.ec, entim.ec, entinstr.ec, entmtg.ec, entnoncat.ec, entrequis.ec, entsccm.ec, entsec.ec, entsecreq.ec, enttextbk.ec, mntalt.ec, mntsec.ec, mntseccan.ec.

Begin work error %d from within add_crs_db()

Error message returned from dbcrs.ec.

Begin Work error %d from within add_sec_db()

Error message returned from dbsec.ec.

Begin Work error %d from within del_facil_db()

Error message returned from dbfacil.ec.

Call to rollover_instrs() failed for the old meeting:\n mtg_no:%ld.

Error message returned from mtg.ec.

CX %d error on return from chk_max_occ()

Error message returned from chk4confl.ec.

CX error '%d' on dmm_size() in get_mr_crshist_rec()

Error message returned from mntcrshis.ec.

CX error '%d' on return from add_crsreqgrp_db() in crsreq_put()

Error message returned from entrequis.ec.

CX error '%d' on return from add_fac_db() in manage_fac_add_scr()

Error message returned from entfac.ec.

CX error '%d' on return from add_id_db() in manage_fac_add_scr(

Error message returned from entfac.ec.

CX error '%d' on return from add_profile_db() in manage_fac_add_scr()

Error message returned from entfac.ec.

CX error '%d' on return from add_profile_db() in manage_fac_upd()
Error message returned from entfac.ec.

CX error '%d' on return from add_sec_db() in manage_sec_add()
Error message returned from entsec.ec.

CX error '%d' on return from add_sec_db() in manage_sec_copy()
Error message returned from entsec.ec.

CX error '%d' on return from begin work in manage_fac_upd()
Error message returned from entfac.ec.

CX error '%d' on return from build_facil_avail_lst() in manage_facil_avail_scrl()
Error message returned from entfacil.ec.

CX error '%d' on return from build_manage_facil_avail_scrl() in manage_facil_criteria_scr()
Error message returned from entfacil.ec.

CX error '%d' on return from build_n_display_crs_list()
Error message returned from entcrs.ec.

CX error '%d' on return from build_n_display_tfacil_lst()
Error message returned from entfacil.ec.

CX error '%d' on return from build_sessyr_dmm()
Error message returned from main.ec.

CX error '%d' on return from build_unmet_req_dmm
Error message returned from mntimv.ec.

CX error '%d' on return from calc_fte_value() in instr_put()
Error message returned from mntseccan.ec.

CX error '%d' on return from calc_fte_value() in upd_instr_fte()
Error message returned from dbinstr.ec.

CX error '%d' on return from chk_4_unmet_im_req()
Error message returned from mntmtg.ec.

CX error '%d' on return from chk_facact_4_confl() in chk_4_faculty_confl()
Error message returned from chk4confl.ec.

CX error '%d' on return from chk_mtg_dmm_4_confl() in chk_4_faculty_confl()
Error message returned from chk4confl.ec.

CX error '%d' on return from commit() in manage_fac_upd()
Error message returned from entfac.ec.

CX error '%d' on return from del_assoc_sec_level() in manage_sec_del()
Error message returned from entcrs.ec, entsec.ec.

CX error '%d' on return from delete() using '%s %s'
Error message returned from entcrs.ec.

CX error '%d' on return from delete in upd_crsreq_scrl()
Error message returned from entrequis.ec.

CX error '%d' on return from delete sec and secdtl records
Error message returned from entsec.ec.

CX error '%d' on return from deleting facil_rec.
Error message returned from entfacil.ec.

CX error '%d' on return from display_pick_scr in man_pick_scrl()

Error message returned from entpick.c.

CX error '%d' on return from dmm_add() in get_mr_crshist_rec()
Error message returned from mntcrshis.ec.

CX error '%d' on return from dmm_add() in ld_crshist_recs()
Error message returned from mntcrshis.ec.

CX error '%d' on return from dmm_get() in change_instructor_assignment_4_section()
Error message returned from entsec.ec.

CX error '%d' on return from dmm_next() in manage_crs_add()
Error message returned from entcrs.ec.

CX error '%d' on return from dmm_next() in manage_tfacil_add()
Error message returned from entfacil.ec.

CX error '%d' on return from do_tbas_exist() in man_ada()
Error message returned from mntada.ec.

CX error '%d' on return from do_tbas_exist() in man_alt_upd_scr()
Error message returned from mntalt.ec.

CX error '%d' on return from fetch no2_cursor
Error message returned from mntcrseq.ec.

CX error '%d' on return from fetch on crseq_cursor.
Error message returned from mntcrseq.ec.

CX error '%d' on return from fetch secmtg_rec
Error message returned from autodel.ec, entsec.ec.

CX error '%d' on return from get_fac_rec() in mnt_fac()
Error message returned from mntfac.ec.

CX error '%d' on return from get_facil_rec() in mnt_facil()
Error message returned from mntfacil.ec.

CX error '%d' on return from get_sessyr() in call_sec_opts()
Error message returned from mntsec.ec.

CX error '%d' on return from ld_crs_dmm() using '%s %s
Error message returned from entcrs.ec.

CX error '%d' on return from ld_facil_avail_dmm()
Error message returned from entfacil.ec.

CX error '%d' on return from ld_sec_recs() in call_sec_opts()
Error message returned from mntsec.ec.

CX error '%d' on return from ld_sec_recs() in crs_ada_upd()
Error message returned from entcrs.ec.

CX error '%d' on return from ld_sec_recs() in mnt_sec()
Error message returned from mntsec.ec.

CX error '%d' on return from ld_tfacil_dmm()
Error message returned from entfacil.ec.

CX error '%d' on return from insert in crsreq_put()
Error message returned from entrequis.ec.

CX error '%d' on return from insert in facqual_put()
Error message returned from entfacqf.ec.

CX error '%d' on return from insert in im_put()

Error message returned from entim.ec.

CX error '%d' on return from insert in manage_facil_add()
Error message returned from entfacil.ec.

CX error '%d' on return from insert in manage_facil_copy()
Error message returned from entfacil.ec.

CX error '%d' on return from insert in mtg_put()
Error message returned from entmtg.ec.

CX error '%d' on return from insert in noncateq_put()
Error message returned from entnoncat.ec.

CX error '%d' on return from insert in sccm_put()
Error message returned from entsccm.ec.

CX error '%d' on return from insert in secreq_put()
Error message returned from entsecreq.ec.

CX error '%d' on return from insert in textbk_put()
Error message returned from enttextbk.rec.

CX error '%d' on return from insert row using '%s'
Error message returned from dbrequis.ec.

CX error '%d' on return from insert using '%s'
Error message returned from db.c, dbfac.ec.

CX error '%d' on return from man_CTRLt_win() in get_sessyr()
Error message returned from sessyr.c.

CX error '%d' on return from man_pick_scrl() in manage_sec_del()
Error message returned from entsec.ec.

CX error '%d' on return from man_pick_scrl() in call_crs_opts()
Error message returned from mntcrs.c, mntfacil.ec.

CX error '%d' on return from man_pick_scrl() in manage_sec_qry()
Error message returned from entsec.ec.

CX error '%d' on return from manage_fac_add_trans() in manage_fac_qry()
Error message returned from entfac.ec.

CX error '%d' on return from manage_fac_add_trans() in manage_facid_qry()
Error message returned from entfac.ec.

CX error '%d' on return from manage_sec_qry() in call_sec_opts()
Error message returned from mntsec.ec.

CX error '%d' on return from reg_get_mtg_rec()
Error message returned from entsec.ec, mntsec.ec, mntseccan.ec.

CX error '%d' on return from reg_get_mtg_rec() in do_tbas_exist()
Error message returned from mntmtg.ec.

CX error '%d' on return from reg_seccal_display() in man_alt_upd_scr()
Error message returned from mntalt.ec.

CX error '%d' on return from rollback work in manage_fac_upd()
Error message returned from entfac.ec.

CX error '%d' on return from scr_dialog() in upd_artic_scrl()
Error message returned from entartic.ec.

CX error '%d' on return from scr_dialog() in upd_crsreq_scrl()

Error message returned from entrequis.ec.

CX error '%d' on return from select in man_ada_n_alt_upd()
Error message returned from mntsec.ec.

CX error '%d' on return from select sec_rec
Error message returned from entcrs.ec.

CX error '%d' on return from select using '%s'
Error message returned from entcrs.ec.

CX error '%d' on return from update crs_rec using '%s %s'
Error message returned from entcrs.ec.

CX error '%d' on return from update facil_table in manage_facil_upd()
Error message returned from entfacil.ec.

CX error '%d' on return from update in manage_fac_upd()
Error message returned from entfac.ec.

CX error '%d' on return from update instr in upd_instr_fte()
Error message returned from dbinstr.ec.

CX error '%d' on return from validate_sessyr() in get_sessyr()",
Error message returned from sessyr.c.

CX error '%d' on return from validate_sessyr() in parse_sessyr_list()
Error message returned from sessyr.c.

CX error '%d' trying to delete fac_rec for %d.
Error message returned from entfac.ec.

CX error '%d' when deleting from crsqual_rec.
Error message returned from entcrsqf.ec.

CX error '%d' when deleting textbk_rec, line %d in file %s.
Error message returned from autodel.ec.

CX error (%d) on return from scr_dialog
Error message returned from entcrshis.ec, entcrsqf.ec, entfacact.ec, entfacnia.ec, entfacqf.ec, entim.ec, entinstr.ec, entmtg.ec, entnoncat.ec, entsccm.ec, entsecreq.ec, enttextbk.ec, mntimv.ec.

CX error %d on return from build_facsch_rec() in build_facsch_dmm()
Error message returned from mntfacsch.ec.

CX error %d on return from chk_4_confl()
Error message returned from chk4confl.ec.

CX error %d on return from chk_max_occ()
Error message returned from mntxmtga.ec.

CX error %d on return from chk_room_avail()
Error message returned from entmtg.ec.

CX error %d on return from delete of %s.
Error message returned from entblob.ec.

CX error %d on return from fetch mtg_rec in build_rmsch_rec()
Error message returned from mntrmsch.ec.

CX error %d on return from free_where_clause_recs() in build_n_display_crs_list()
Error message returned from entcrs.ec.

CX error %d on return from free_where_clause_recs() in build_n_display_tfacil_lst()

Error message returned from entfacil.ec.

CX error %d on return from free_where_clause_recs() in upd_facil_criteria_scr()
Error message returned from entfacil.ec.

CX error %d on return from from man_xsec_add()
Error message returned from entmtg.ec.

CX error %d on return from from man_xsecv()
Error message returned from entfacsch.c, entmtg.ec, entrmsch.c.

CX error %d on return from man_pick_scrl() in build_n_display_crs_list()
Error message returned from entcrs.ec.

CX error %d on return from man_pick_scrl() in build_n_display_tfacil_lst()
Error message returned from entfacil.ec.

CX error %d on return from man_pick_scrl() in manage_facil_avail_scrl()
Error message returned from entfacil.ec.

CX error %d on return from man_room_scrl()
Error message returned from entmtg.ec.

CX error %d on return from man_xmtga_scrl()
Error message returned from mntxmtga.ec.

CX error %d on return from manage_facil_criteria_scr() in upd_mtg_scrl()
Error message returned from entmtg.ec.

CX error %d on return from qry_fac_id()
Error message returned from entinstr.ec.

CX error %d on return from qry_func at line %d in %s
Error message returned from entgetset.c.

CX error %d on return from query on %s
Error message returned from entartic.ec.

CX error %d on return from select facil in chk_max_occ()
Error message returned from chk4confl.ec.

CX error %d on return from undo_section_cancel() in sec_chk()
Error message returned from entsec.ec.

CX error %d on return from upd_instr_fte()
Error message returned from entmtg.ec.

CX error %d on return from update %s
Error message returned from entblob.ec, entgetset.c.

CX error %d on return from updating secmtg_rec()
Error message returned from chk4confl.ec.

CX error %d when deleting from %s.
Error message returned from entgetset.c.

CX error after execution of dmm_delete(): dmm_size == %d
Error message returned from entxmtga.ec.

CX error during switch on '%c' in mnt_crsreq()
Error message returned from mntrequis.ec.

CX error during switch on '%c' in set_crsreq_scrl_prompts()
Error message returned from entrequis.ec.

CX error in call_crs_opts()

Error message returned from mntcrs.c.

CX error in call_fac_opts()
Error message returned from mntfac.ec.

CX error in call_sec_opts()
Error message returned from mntsec.ec.

CX error on return from add_crs_db()
Error message returned from entcrs.ec.

CX error on return from add_xref_2dmm() in manage_xlisting().
Error message returned from xlist.ec.

CX error on return from add_xref_2dmm() in rollover_xlisting().
Error message returned from xlist.ec.

CX error on return from chk_xref_dmm() in rollover_xlisting().
Error message returned from xlist.ec.

CX error on return from db_dabind(): %s
Error message returned from crs.ec, dbcrs.ec, dbfacil.ec, xlist.ec.

CX error on return from db_dabind() in rollover_sess():\n%s
Error message returned from sess.ec.

CX error on return from db_dabind() in rollover_xlisting(): %s
Error message returned from xlist.ec.

CX error on return from db_dabind() of newmtg in rollover_mtg():\n%s
Error message returned from mtg.ec.

CX error on return from db_dabind() of oldmtg in rollover_mtg():\n%s
Error message returned from mtg.ec.

CX error on return from dmm_add in chk_4_faculty_confl()
Error message returned from chk4confl.ec.

CX error on return from dmm_add()
Error message returned from entcrseq.ec, entcrsqf.ec, entfacact.ec, entnoncat.ec, entmxtga.ec.

CX error on return from dmm_add() in build_unmet_im_req_dmm()
Error message returned from mntimv.ec.

CX error on return from dmm_add() in manage_crs_add()
Error message returned from entcrs.ec.

CX error on return from dmm_add() in manage_facil_add()
Error message returned from entfacil.ec.

CX error on return from dmm_add() in parse_sessyr_list()
Error message returned from sessyr.c.

CX error on return from dmm_add() in upd_artic_scrl()
Error message returned from entartic.ec.

CX error on return from dmm_add() in upd_crshist_scrl()
Error message returned from entcrshis.ec.

CX error on return from dmm_add() in upd_crsreq_scrl()
Error message returned from entrequis.ec.

CX error on return from dmm_add() in upd_facqual_scrl()

Error message returned from entfacqf.ec.

CX error on return from dmm_add() in upd_im_scrl()
Error message returned from entim.ec.

CX error on return from dmm_add() in upd_mtg_scrl()
Error message returned from entmtg.ec.

CX error on return from dmm_add() in upd_sccm_scrl()
Error message returned from entsccm.ec.

CX error on return from dmm_add() in upd_secreq_scrl()
Error message returned from entsecreq.ec.

CX error on return from dmm_add() in upd_textbk_scrl()
Error message returned from enttextbk.ec.

CX error on return from dmm_back() in del_mntcrsreq_from_dmm()
Error message returned from entrequis.ec.

CX error on return from dmm_back() in manage_crs_del()
Error message returned from entcrs.ec.

CX error on return from dmm_back() in manage_facil_del()
Error message returned from entfacil.ec.

CX error on return from dmm_delete() in manage_facil_del()
Error message returned from entfacil.ec.

CX error on return from dmm_delete() in manage_sec_del()
Error message returned from entsec.ec.

CX error on return from dmm_delete() in manage_xlisting().
Error message returned from xlist.ec.

CX error on return from dmm_delete() in rollover_xlisting().
Error message returned from xlist.ec.

CX error on return from dmm_get() in manage_crs_del()
Error message returned from entcrs.ec.

CX error on return from dmm_get() in manage_facil_del()
Error message returned from entfacil.ec.

CX error on return from dmm_get() in manage_xlisting().
Error message returned from xlist.ec.

CX error on return from dmm_get() in rollover_xlisting().
Error message returned from xlist.ec.

CX error on return from dmm_insert() in manage_crs_add()
Error message returned from entcrs.ec.

CX error on return from dmm_insert()in manage_tfacil_add()
Error message returned from entfacil.ec.

CX error on return from dmm_position() in man_pick_scrl()
Error message returned from entpick.c.

CX error on return from dmm_put()in chk_dmm_4_confl()
Error message returned from entcrsqf.ec.

CX error on return from dmm_put() in manage_crs_del()
Error message returned from entcrs.ec.

CX error on return from dmm_put() in manange_crs_upd()

Error message returned from entcrs.ec.

CX error on return from dmm_put() in manange_facil_upd()
Error message returned from entfacil.ec.

CX error on return from dmm_put() in mntcrsreq_put()
Error message returned from entrequis.ec.

CX error on return from dmm_put() in upd_crsqual_dmm()
Error message returned from entcrsqf.ec.

CX error on return from dmm_put() in upd_mtg_scrl()
Error message returned from entmtg.ec.

CX error on return from dmm_put() in update_mntsec_rec_mtg_status_flags()
Error message returned from mntsec.ec.

CX error on return from getmenu()
Error message returned from mntfac.ec.

CX error on return from ld_im_recs() in mnt_im()
Error message returned from mntim.ec.

CX error on return from ids_select: %s
Error message returned from entfac.ec.

CX error on return from insert statement in insert_mtg(). SQLCA_S.SQLCODE: %d
Error message returned from mtg.ec.

CX error on return from insert statement in insert_secmtg(). SQLCA_S.SQLCODE: %d
Error message returned from secmtg.ec.

CX error on return from insert statement in rollover_sec(). SQLCA_S.SQLCODE: %d
Error message returned from sec.ec.

CX error on return from open statement in ld_crs_dmm(). SQLCA_S.SQLCODE: %d
Error message returned from dbcrs.ec.

CX error on return from reg_findxl() in build_rmsch_rec(): tmpsec_dmm_size == 0: missing section record.
Error message returned from mntrmsch.ec.

CX error on return from reg_get_mtgrec() - size == %d
Error message returned from entxmtga.ec.

CX error on return from strtrim() in validate_sessyr():'%s'.
Error message returned from sessyr.c.

CX error on size of pick_dmm in man_pick_scrl()
Error message returned from entpick.c.

CX error on switch of '%d' in call_crs_opts()
Error message returned from mntcrs.c.

CX error switching on '%d' in call_facil_opts()
Error message returned from mntfacil.ec.

CX error: invalid value '%d' passed to rollover_mtg() for dup_sec_flg parameter.
Error message returned from mtg.ec.

CX error: meeting record not found:\n(yr, sess, cat, crs_no, sec_no) '%d' '%s' '%s' '%s' '%s'\nnew meeting record expected during rerun for duplicate section.
Error message returned from mtg.ec.

CX error: Undefined facility type specified:'%d' on

Error message returned from dbfacil.ec.

CX error: unexpected status '%d' on return from $FETCH crs_cursor in get_crs().
Error message returned from crs.ec.

CX error: unexpected status '%d' on return from $open crs_cursor in get_crs().
Error message returned from crs.ec.

CX error: unexpected status '%d' on return from '$OPEN newmtg_cursor' in rollover_mtg().
Error message returned from mtg.ec.

CX error: unknown quickkey'%d' returned from scr_dialog() in mnt_catalog()
Error message returned from mntcatalog.c.

CX error: unknown status '%d' on return from:\n '$EXECUTE IMMEDIATE %s'\n in rollover_instrs().\n%s\n
Error message returned from instr.ec.

CX error: unknown status '%d' on return from:\n '$SELECT flt FROM im_table WHERE im MATCHES %s' in rollover_instrs().\n%s\n
Error message returned from instr.ec.

CX ERROR: unknown status '%d' on return from '$BEGIN WORK in rollover_sec_trans().\n\n%s
Error message returned from sec.ec.

CX error: unknown status '%d' on return from '$ CLOSE crs_cursor' in ld_crs_dmm().\n\n%s
Error message returned from dbcrs.ec.

CX error: unknown status '%d' on return from '$ CLOSE facil_cursor' in ld_tfacil_dmm().\n\n%s
Error message returned from dbfacil.ec.

CX ERROR: unknown status '%d' on return from '$COMMIT WORK in rollover_sec_trans().\n\n%s
Error message returned from sec.ec.

CX error: unknown status '%d' on return from '$DROP TABLE tmp_im' in rollover_secim().\n\n%s
Error message returned from secperph.ec.

CX error: unknown status '%d' on return from '$DROP TABLE tmp_instr' in rollover_instr().\n%s\n
Error message returned from instr.ec.

CX error: unknown status '%d' on return from '$DROP TABLE tmp_sccm' in rollover_sccm().\n\n%s
Error message returned from secperph.ec.

CX error: unknown status '%d' on return from '$DROP TABLE tmp_secdtl' in rollover_secdtl().\n\n%s
Error message returned from secperph.ec.

CX error: unknown status '%d' on return from '$DROP TABLE tmp_secreq' in rollover_secreq().\n\n%s
Error message returned from secperph.ec.

CX error: unknown status '%d' on return from '$DROP TABLE tmp_textbk' in rollover_textbk().\n\n%s

Error message returned from secperph.ec.

CX error: unknown status '%d' on return from '$ FETCH facil_cursor using descriptor q_desc' in ld_tfacil_dmm().\n\n%s
Error message returned from dbfacil.ec.

CX error: unknown status '%d' on return from '$FETCH newmtg' in rollover_mtg().
Error message returned from mtg.ec.

CX error: unknown status '%d' on return from '$FETCH oldmtg' in rollover_mtg().
Error message returned from mtg.ec.

CX error: unknown status '%d' on return from '$FETCH oldsec_cursor' statement in rollover_sess().\n\n%s
Error message returned from sess.ec.

CX error: unknown status '%d' on return from '$INSERT INTO im_rec' in rollover_secim().\n%s\n
Error message returned from secperph.ec.

CX error: unknown status '%d' on return from '$INSERT INTO instr_rec' in rollover_instr().\n%s\n
Error message returned from instr.ec.

CX error: unknown status '%d' on return from '$INSERT INTO schd_comment_rec' in rollover_sccm().\n%s\n
Error message returned from secperph.ec.

CX error: unknown status '%d' on return from '$INSERT INTO secreq_rec' in rollover_secreq().\n%s\n
Error message returned from secperph.ec.

CX error: unknown status '%d' on return from '$INSERT INTO textbk_rec' in rollover_textbk().\n%s\n
Error message returned from secperph.ec.

CX error: unknown status '%d' on return from '$INSERT INTO tmp_secdtl' in rollover_secdtl().\n\n%s
Error message returned from secperph.ec.

CX error: unknown status '%d' on return from '$ OPEN crs_cursor' in ld_crs_dmm().\n\n%s
Error message returned from dbcrs.ec.

CX error: unknown status '%d' on return from '$ OPEN facil_cursor' in ld_tfacil_dmm().\n\n%s
Error message returned from dbfacil.ec.

CX error: unknown status '%d' on return from '$ PREPARE crs_id from $q_string' in ld_crs_dmm().\n\n%s
Error message returned from dbcrs.ec.

CX error: unknown status '%d' on return from '$ PREPARE facil_id from $q_string' in ld_tfacil_dmm().\n\n%s
Error message returned from dbfacil.ec.

CX ERROR: unknown status '%d'on return from '$ROLLBACK WORK in rollover_sec_trans().\n\n%s
Error message returned from sec.ec.

CX error: unknown status '%d' on return from '$SELECT * FROM im_rec' in rollover_secim().\n\n%s

Error message returned from secperph.ec.

CX error: unknown status '%d' on return from '$SELECT * FROM instr_rec' in rollover_instr().\n%s\n
Error message returned from instr.ec.

CX error: unknown status '%d' on return from '$SELECT * FROM schd_comment_rec' in rollover_sccm().\n\n%s
Error message returned from secperph.ec.

CX error: unknown status '%d' on return from '$SELECT * FROM secdtl_rec' in rollover_secdtl().\n\n%
Error message returned from secperph.ec.

CX error: unknown status '%d' on return from '$SELECT * FROM secreq_rec' in rollover_secreq().\n\n%s
Error message returned from secperph.ec.

CX error: unknown status '%d' on return from '$SELECT * FROM textbk_rec' in rollover_textbk().\n\n%s
Error message returned from secperph.ec.

CX error: unknown status '%d' on return from '$UPDATE tmp_im' in rollover_secim().\n\n%s
Error message returned from secperph.ec.

CX error: unknown status '%d' on return from '$UPDATE tmp_sccm' in rollover_sccm().\n\n%s
Error message returned from secperph.ec.

CX error: unknown status '%d' on return from '$UPDATE tmp_secdtl' in rollover_secdtl().\n\n%s
Error message returned from secperph.ec.

CX error: unknown status '%d' on return from '$UPDATE tmp_secreq' in rollover_secreq().\n\n%s
Error message returned from secperph.ec.

CX error: unknown status '%d' on return from '$UPDATE tmp_textbk' in rollover_textbk().\n\n%s
Error message returned from secperph.ec.

CX error: unknown status '%d' on return from add_mtg() in add_mtg().
Error message returned from mtg.ec.

CX error: unknown status '%d' on return from add_mtg() in is_sec_ok_2_roll().
Error message returned from xlist.ec.

CX error: unknown status '%d' on return from chk_xref_dmm() in manage_xlisting().
Error message returned from xlist.ec.

CX error: unknown status '%d' on return from FETCH statement in manage_xlisting().
Error message returned from xlist.ec.

CX error: unknown status '%d' on return from FETCH statement in rollover_xlisting().
Error message returned from xlist.ec.

CX error: unknown status '%d'on return from get_crs() in is_sec_ok_2_roll().
Error message returned from sec.ec.

CX error: unknown status '%d' on return from get_crs() in rollover_sess().
Error message returned from sess.ec.

CX error: unknown status '%d'on return from insert_secmtg() in rollover_xlisting().

Error message returned from xlist.ec.

CX error: unknown status '%d'on return from is_sec_ok_2_roll() in add_xref_2dmm().
Error message returned from xlist.ec.

CX error: unknown status '%d' on return from is_sec_ok_2_roll() in rollover_sess().
Error message returned from sess.ec.

CX error: unknown status '%d' on return from manage_xlisting() in rollover_mtg().
Error message returned from mtg.ec.

CX error: unknown status '%d' on return from open statement in rollover_mtg().
Error message returned from mtg.ec.

CX error: unknown status '%d' on return from rollover_aaa_recs() in rollover_secperph().
Error message returned from secperph.ec.

CX error: unknown status '%d'on return from rollover_im_rec() in rollover_secperph().
Error message returned from secperph.ec.

CX error: unknown status '%d' on return from rollover_instrs() in add_mtg().
Error message returned from mtg.ec.

CX error: unknown status '%d' on return from rollover_mtg() in rollover_sec().
Error message returned from sec.ec.

CX error: unknown status '%d' on return from rollover_schd_comment_rec() in rollover_secperph().
Error message returned from secperph.ec.

CX error: unknown status '%d' on return from rollover_sec_trans() in rollover_sess().
Error message returned from sess.ec.

CX error: unknown status '%d'on return from rollover_secdtl() in rollover_secperph().
Error message returned from secperph.ec.

CX error: unknown status '%d' on return from rollover_secperph() in rollover_sec().
Error message returned from sec.ec.

CX error: unknown status '%d' on return from rollover_secreq_rec() in rollover_secperph().
Error message returned from secperph.ec.

CX error: unknown status '%d' on return from rollover_textbk_rec() in rollover_secperph().
Error message returned from secperph.ec.

CX error: unknown status '%d' on return from rollover_xlisting() in rollover_mtg().
Error message returned from mtg.ec.

Commit Work error %d
Error message returned from entartic.ec, entblob.ec, entcrs.ec, entcrseq, entcrshis.ec, entcrsqf.ec, entfac.ec, entfacact.ec, entfacnia.ec, entfacqf.ec, entim.ec, entinstr.ec, entmtg.ec, entnoncat.ec, entrequis.ec, entsccm.ec, entsec.ec, entsecreq.ec, enttextbk.ec, mntalt.ec, mntsec.ec, mntseccan.ec.

Commit Work error %d from within add_crs_db()
Error message returned from dbcrs.ec.

Commit Work error %d from within add_sec_db()
Error message returned from dbsec.ec.

Commit Work error %d from within del_facil_db()
Error message returned from dbfacil.ec.

Db error '%d' on fetch at line %d in file %s.

Error message returned from mntartic.ec.

DBE_ENFIND Error %d on %s
Error message returned from dbsec.ec.

Declare cursor error %d at line %d in file %s, %s.
Error message returned from dbsec.ec.

Declare cursor error at line %d in file %s, status = %d
Error message returned from autodel.ec, chk4confl.ec, entcrs.ec, entfacil.ec, entsec.ec, lkuproom.ec, mntfacld.ec, mntfacqf.ec, mntfacsch.ec, mntim.ec, mntimv.ec, mninstr.ec, mntmtg.ec, mntnoncat.ec, mntrmsch.ec, mntsccm.ec.

ERROR: Cannot locate academic calendar record for:\n(prog, yr, sess, subsess) '%s' '%d' '%s' '%s'
Error message returned from sec.ec.

ERROR: Column '%s' not found in crs table.
Error message returned from entsec.ec.

ERROR: Column '%s' not found in sec table.
Error message returned from entsec.ec.

ERROR: Environmental variable CARSDB not set
Error message returned from main.ec.

ERROR: Missing course record for: (crs_no, cat) '%s' '%s'
Error message returned from sec.ec.

ERROR: unexpected status '%d' on return from '$OPEN oldmtg_cursor' in rollover_mtg().
Error message returned from mtg.ec.

ERROR: unexpected status '%d' on return from $open oldsec_cursor in rollover_sess().
Error message returned from sess.ec.

ERROR: unexpected status '%d' on return from '$OPEN oldxsec_cursor1' in manage_xlisting().
Error message returned from xlist.ec.

ERROR: unexpected status '%d' on return from '$OPEN oldxsec_cursor2' in rollover_xlisting().
Error message returned from xlist.ec.

Error '%d' at line %d in file %s on return from del_ada_rec() in del_assoc_sec_level_recs()
Error message returned from autodel.ec.

Error '%d' at line %d in file %s on return from del_all_crs_im_recs () in del_assoc_crs_level_recs()
Error message returned from autodel.ec.

Error '%d' at line %d in file %s on return from del_all_crsabstr_recs() in del_assoc_crs_level_recs()
Error message returned from autodel.ec.

Error '%d' at line %d in file %s on return from del_all_crartic_recs() in del_assoc_crs_level_recs()
Error message returned from autodel.ec.

Error '%d' at line %d in file %s on return from del_all_crseq_recs() in del_assoc_crs_level_recs()
Error message returned from autodel.ec.

Error '%d' at line %d in file %s on return from del_all_crshist_recs() in del_assoc_crs_level_recs()

Error message returned from autodel.ec.

Error '%d' at line %d in file %s on return from del_all_crsqf_recs() in del_assoc_crs_level_recs()
Error message returned from autodel.ec.

Error '%d' at line %d in file %s on return from del_all_crsreq_recs() in del_assoc_crs_level_recs()
Error message returned from autodel.ec.

Error '%d' at line %d in file %s on return from del_all_sccm_recs() in del_assoc_sec_level_recs()
Error message returned from autodel.ec.

Error '%d' at line %d in file %s on return from del_all_sec_im_recs() in del_assoc_sec_level_recs()
Error message returned from autodel.ec.

Error '%d' at line %d in file %s on return from del_all_secreq_recs() in del_assoc_sec_level_recs()
Error message returned from autodel.ec.

Error '%d' at line %d in file %s on return from del_all_textbk_recs() in del_assoc_sec_level_recs()
Error message returned from autodel.ec.

Error '%d' at line %d in file %s on return from del_altcal_rec() in del_assoc_sec_level_recs()
Error message returned from autodel.ec.

Error '%d' at line %d in file %s on return from del_altrfnd_rec() in del_assoc_sec_level_recs()
Error message returned from autodel.ec.

Error '%d' at line %d in file %s on return from del_crsobj_recs() in del_assoc_crs_level_recs()
Error message returned from autodel.ec.

Error '%d' at line %d in file %s on return from del_crsrequir_recs() in del_assoc_crs_level_recs()
Error message returned from autodel.ec.

Error '%d' at line %d in file %s on return from del_crssyll_recs() in del_assoc_crs_level_recs()
Error message returned from autodel.ec.

Error '%d' at line %d in file %s on return from del_secbrdg_rec() in del_assoc_sec_level_recs()
Error message returned from autodel.ec.

Error '%d' at line %d in file %s on return from ld_im_recs() in del_all_sec_im_recs()
Error message returned from autodel.ec.

Error '%d' at line %d in file %s updating section or section detail.
Error message returned from mntalt.ec.

Error '%d' at line %d in file %s when adding secreq_rec.
Error message returned from mntsecreq.ec.

Error '%d' at line %d in file %s when declaring secreq_cursor.
Error message returned from mntsecreq.ec.

Error '%d' at line %d in file %s when deleting ada_rec.

Error message returned from autodel.ec.

Error '%d' at line %d in file %s when deleting altcal_rec.
Error message returned from autodel.ec.

Error '%d' at line %d in file %s when deleting altrfnd_rec.
Error message returned from autodel.ec.

Error '%d' at line %d in file %s when deleting artic_rec.
Error message returned from autodel.ec.

Error '%d' at line %d in file %s when deleting crsabstr_rec.
Error message returned from autodel.ec.

Error '%d' at line %d in file %s when deleting crseq_rec.
Error message returned from autodel.ec.

Error '%d' at line %d in file %s when deleting crshist_rec.
Error message returned from autodel.ec.

Error '%d' at line %d in file %s when deleting crsobj_rec.
Error message returned from autodel.ec.

Error '%d' at line %d in file %s when deleting crsqual_rec.
Error message returned from autodel.ec.

Error '%d' at line %d in file %s when deleting crsreq_rec.
Error message returned from autodel.ec.

Error '%d' at line %d in file %s when deleting crsreqgrp_rec.
Error message returned from autodel.ec.

Error '%d' at line %d in file %s when deleting crsrequir_rec.
Error message returned from autodel.ec.

Error '%d' at line %d in file %s when deleting crssyll_rec.
Error message returned from autodel.ec.

Error '%d' at line %d in file %s when deleting im_rec.
Error message returned from autodel.ec.

Error '%d' at line %d in file %s when deleting instr_rec.
Error message returned from autodel.ec.

Error '%d' at line %d in file %s when deleting mtg_rec.
Error message returned from autodel.ec.

Error '%d' at line %d in file %s when deleting noncateq_rec.
Error message returned from autodel.ec.

Error '%d' at line %d in file %s when deleting schd_comment_rec.
Error message returned from autodel.ec.

Error '%d' at line %d in file %s when deleting secbrdg_rec.
Error message returned from autodel.ec.

Error '%d' at line %d in file %s when deleting secmtg_rec
Error message returned from autodel.ec.

Error '%d' at line %d in file %s when deleting secreq_rec.
Error message returned from autodel.ec, mntsecreq.ec.

Error '%d' at line %d in file %s when fetching secreq_cursor.
Error message returned from mntsecreq.ec.

Error '%d' at line %d in file %s when opening secreq_cursor.

Error message returned from mntsecreq.ec.

Error '%d' at line %d, file %s trying to update mtg_rec.
Error message returned from mntseccan.ec.

Error '%d' at line %d, file %s when trying to update instr_rec
Error message returned from mntseccan.ec.

Error '%d' on return from select in get_facil_rec(
Error message returned from mntfacil.ec.

Error '%d' on return from structinit()
Error message returned from mntada.ec, mntcrsabs.ec, mntcrsobj.ec, mntcrssyl.ec, mntrequir.ec, mntsecbdg.ec.

Error '%d' on select of fac_fields in get_fac_rec()
Error message returned from mntfac.ec.

Error '%d' on select of profile_fields in manage_facid_qry()
Error message returned from entfac.ec.

Error '%d' trying to update section or section detail record
Error message returned from entsec.ec.

Error '%d' when declaring cursor at line %d in file %s.
Error message returned from mnttextbk.ec.

Error '%d' when declaring facact_cursor in chk_facact_4_confl()
Error message returned from chk4confl.ec.

Error '%d' when declaring facact_cursor in ld_mntfacact_recs
Error message returned from mntfacact.ec.

Error '%d' when deleting crsreq_rec %d.
Error message returned from entrequis.ec.

Error '%d' when fetching cursor at line %d in file %s.
Error message returned from mnttextbk.ec.

Error '%d' when fetching facact_cursor in ld_mntfacact_recs
Error message returned from mntfacact.ec.

Error '%d' when open facact_cursor in ld_mntfacact_recs
Error message returned from mntfacact.ec.

Error '%d' when opening cursor at line %d in file %s.
Error message returned from mnttextbk.ec.

Error '%d' when selecting fac_fields in upd_instr_scrl()
Error message returned from entinstr.ec.

Error '%d' when selecting fac_rec in manage_fac_qry()
Error message returned from entfac.ec.

Error '%d' when selecting id fields in get_fac_rec()
Error message returned from mntfac.ec.

Error '%d' when selecting profile fields in get_fac_rec()
Error message returned from mntfac.ec.

Error '%d' when updating instr_rec at line %d in file %s.
Error message returned from entinstr.ec, entsec.ec.

Error '%d' when updating mtg_rec at line %d, file %s.

Error message returned from mntseccan.ec.

Error '%d' when updating sections at line %d, file %s.
Error message returned from dbsec.ec, mntsec.ec.

Error '%d' when updating sections in mnt_seccan()
Error message returned from mntseccan.ec.

Error %d on insert of profile_rec at line %d in file %s.
Error message returned from entfac.ec.

Error %d on return from updating secmtg_rec.
Error message returned from chk4confl.ec.

Error %d when adding ada_rec.
Error message returned mntada.ec.

Error %d when adding crsabstr_rec.
Error message returned from mntcrsabs.ec.

Error %d when adding to crsobj_rec.
Error message returned from mntcrsobj.ec.

Error %d when adding crsrequir_rec.
Error message returned from mntrequir.ec.

Error %d when adding to crssyll_rec.
Error message returned from mntcrssyl.ec.

Error %d when deleting ada_rec.
Error message returned mntada.ec.

Error %d when deleting crsabstr_rec.
Error message returned from mntcrsabs.ec.

Error %d when deleting crsrequir_rec.
Error message returned from mntrequir.ec.

Error %d when deleting crssyll_rec.
Error message returned from mntcrssyl.ec.

Error %d when deleting from crsobj_rec.
Error message returned from mntcrsobj.ec.

Error %d when finding crsobj_rec.
Error message returned from mntcrsobj.ec.

Error %d when finding crsrequir_rec.
Error message returned from mntrequir.ec.

Error %d when selecting ada_rec.
Error message returned from mntada.ec.

Error %d when selecting secbrdg_rec.
Error message returned from mntsecbdg.ec.

Error %d when trying to find crssyll_rec.
Error message returned from mntcrssyl.ec.

Error %d when updating ada_rec.
Error message returned mntada.ec.

Error %d when updating crsabstr_rec.
Error message returned from mntcrsabs.ec.

Error %d when updating from crsobj_rec.

Error message returned from mntcrsobj.ec.

Error %d when updating secbrdg_rec.
Error message returned from mntsecbdg.ec.

ERROR (%d) on scr_init()\n
Error message returned from main.ec.

ERROR on return from dmm_add() in add_xref_2dmm()
Error message returned from xlist.ec.

Error openning file '%s' for R25 conflict checking: error %d.
Error message returned from common.c.

Invalid year specified in course catalog: '%s'. The last two characters of the catalog specifier must represent a year.
Error message returned from mntcrshis.ec.

mnt_seccan: malloc(%d): Out of memory.
Error message returned from mntseccan.ec.

No academic calendar records found for %s
Error message returned from main.ec.

No non-instruction assignment records to display
Error message returned from mntfacnia.ec.

No row found to update instr_rec number %d
Error message returned from entinstr.ec.

Open cursor error %d at line %d in file %s, %s.
Error message returned from dbsec.ec.

Open cursor error at line %d in file %s, status = %d
Error message returned from autodel.ec, chk4confl.ec, entcrs.ec, entfacil.ec, entsec.ec, lkuproom.ec, mntcrseq.ec, mntcrshis.ec, mntcrsqf.ec, mntfacld.ec, mntfacnia.ec, mntfacqf.ec, mntfacsch.ec, mntim.ec, mntimv.ec, mntinstr.ec, mntmtg.ec, mntnoncat.ec, mntrmsch.ec, mntsccm.ec, mntsec.ec.

OPEN ERROR: %d on '%s'
Error message returned from main.ec.

Rollback Work error %d
Error message returned from dbsec.ec, entartic.ec, entblob.ec, entcrs.ec, entcrseq, entcrshis.ec, entcrsqf.ec., entfac.ec, entfacact.ec, entfacnia.ec, entfacqf.ec, entim.ec, entinstr.ec, entmtg.ec, entnoncat.ec, entrequis.ec, entsccm.ec, entsec.ec, entsecreq.ec, enttextbk.ec, mntalt.ec, mntsec.ec, mntseccan.ec.

Rollback Work error %d from within add_crs_db()
Error message returned from dbcrs.ec.

Rollback Work error %d from within add_sec_db()
Error message returned from dbsec.ec.

Rollback Work error %d from within del_facil_db()
Error message returned from dbfacil.ec.

Section detail record not found
Error message returned from dbsec.ec.

SELECT Error %d on %s
Error message returned from dbsec.ec.

Select/fetch error %d when fetching tmpnoninstr_cursor

Error message returned from mntfacld.ec.

TBL(acad_cal_rec): %s
Error message returned from main.ec.

Error and Crash Recovery Procedure

Introduction

If a CX program crashes, you may need to perform one or more corrective steps to recover. The recovery procedure in this section begins with the simplest steps and progresses to the most complex step.

Core Dump and Fatal Error Recovery

The following procedure describes the steps to recover from a core dump or fatal error in a CX program.

1. Access the program screens directory for the CX program.

   Example: % cd $CARSPATH/modules/regist/progscr/crsent

2. Reinstall each program screen file.

   Example: % make reinstall F=<filename>

   Note: You also can 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/regist/crsent

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, remove the currently compiled program objects and allow for a fresh reinstall of the code.

   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, then reinstall the source for the program.

Example: % cd <to appropriate library>

% make REINSTALL

% cd $CARSPATH/regist/crsent

% 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, you are done
    - If no, go to step 13
13. Remove the currently compiled program objects and allow for a fresh reinstall of the code.

    Example: % make cleanup
14. Reinstall the library or libraries, and then reinstall the source for the program.

    Example: % make REINSTALL

    % cd $CARSPATH/src/regist/crsent

    % make REINSTALL

    Note: You must reinstall the source program to include any library changes.
15. Attempt to execute the program after updating the libraries. Did it execute properly?
    - If yes, you are done
    - If no, call Jenzabar, Inc. Support Services

INDEX

D

E

F

G

I

M

N

O

P

R

S

T

U

W