Th
res
of
thi
rep
ite
is
no
rea

Name	XDPB32-SECURE
Application	ST
Description	Canvas CSV file creator and updater
Default menu ID	
Release status	Internal Use Only
Added by	SPENNERS
Added on	06/18/2012
Changed by	SPENNERS
Changed on	06/18/2012
Jobstat cycles	10
Review process only	No
Force full record I/O	No
Uses bar graph	No
Component type	Main

Technical Documentation

```
Pacific Union College, Angwin, California
Canvas batch upload process backend
Created: 5/31/2011
Programmer: Steve Penners
Contact: spenners@puc.edu

History:
    6/18/12 - spenners - Added STTR.STUDENT.ACAD.CRED to Process
Elements.  Why this was missing, and how it generated while it was
missing without throwing an error is a mystery.  But it caused the
system to not work.  A gen on 6/15/12 caused this to generate without
that field in the Process Elements (where it went is a mystery), which
consequently broke the system.
Added log file dump of any errors calling the Ellucian sequential file
writing subroutines.  Just in case.
   4/24/12 - spenners - reprogrammed line 513 to account for bad
pointers to COURSE.SEC.FACULTY in COURSE.SECTIONS.  If there is one of
these, it can cause teachers to be assigned to incorrect courses.  Not
good.
                                    Also added lines around 597-604
changing a non person into "staff" for course listings
```

This program has two modes that it runs in: 1) full batch update, 2) incremental update
Full batch update is run nightly and deletes all data for a term, and then recreates it. It repeats this for each term if there are multiple terms configured to be updated.
The terms are found in PUC.CONTROL.POINT, CANVAS.SETTINGS and are edited by running XDP21.
Incremental update runs every 30 minutes and only uploads changed data from the given terms.

Full batch update creates 6 files at a time for each term in the HOLD area; enrollments.csv, users.csv, teachers.csv, courses.csv, sections.csv, xlists.csv.
These files contain active records for the given term.

The users who are selected to be included are checked as to having a PUC email account and if they don't have one, then they are run through account creation/activation.

Once the files are created in the HOLD area, the files are moved to a specified folder located at d:\ftp_folders\live\PUC.CANVAS.UPLOAD.DIR on colleague-live.

Then a Powershell process is called. This process zips and uploads the files into Canvas.

This program generally runs incremental uploads all day long. Once a night it runs and uploads all data. This is triggered by the system date change. A task is scheduled to run this every 30 minutes at 10 and 40 past the hour. See Task Manager on colleague-live: Canvas...

X.LOG.LEVEL should be set to either 'ERROR' or 'ALL'. ERROR is for normal production. ALL is for debugging. If you set X.DEBUG to TRUE it will increase the debug level to ALL automatically.

Technical Documentation Keywords

Process Data Element Name	Reference Only	Source	Type	Association
PUC.CONTROL.PC	No	PUC.CONTROL.PC	Key	
XFIELD1	No	PUC.CONTROL.PC	Data	
XFIELD2	No	PUC.CONTROL.PC	Data	
XFIELD3	No	PUC.CONTROL.PC	Data	
XFIELD4	Yes	PUC.CONTROL.PC	Data	
XFIELD6	Yes	PUC.CONTROL.PC	Data	
PUC.CANVAS.LOG	No	PUC.CANVAS.LOG	Key	
XPCL.DATE	No	PUC.CANVAS.LOG	Data	

Process Data Element Name	Reference Only	Source	Type	Association
XPCL.TIME	No	PUC.CANVAS.LOG	Data	
XPCL.LOG	No	PUC.CANVAS.LOG	Comments	
STUDENT.ACAD.C	Yes	STUDENT.ACAD.C	Key	
STC.TERM	Yes	STUDENT.ACAD.C	Data	
STC.STATUS	Yes	STUDENT.ACAD.C	Association	STC.STATUSES
STC.COURSE.NAM	Yes	STUDENT.ACAD.C	Data	
STC.STUDENT.CC	Yes	STUDENT.ACAD.C	SV Pointer	
STC.SECTION.NC	Yes	STUDENT.ACAD.C	Data	
PUC.CANVAS.STU	No	PUC.CANVAS.STU	Key	
XPCSAC.DATE	No	PUC.CANVAS.STU	Data	
XPCSAC.STATUS	No	PUC.CANVAS.STU	Data	
XPCSAC.TIME	No	PUC.CANVAS.STU	Data	
PUC.CANVAS.STU	No	PUC.CANVAS.STU	Key	
XPCST.DATE	No	PUC.CANVAS.STU	Data	
XPCST.STATUS	No	PUC.CANVAS.STU	Data	
XPCST.TIME	No	PUC.CANVAS.STU	Data	
STTR.STUDENT	Yes	STUDENT.TERMS	Key	
STTR.TERM	Yes	STUDENT.TERMS	Key	
STTR.STUDENT.A	Yes	STUDENT.TERMS	MV Pointer	
STTR.ACAD.LEVE	Yes	STUDENT.TERMS	Key	
SCS.COURSE.SEC	Yes	STUDENT.COURSE	SV Pointer	
COURSE.SECTION	Yes	COURSE.SECTION	Key	
SEC.SUBJECT	Yes	COURSE.SECTION	Data	
SEC.COURSE.NO	Yes	COURSE.SECTION	Data	
SEC.NO	Yes	COURSE.SECTION	Data	
SEC.FACULTY	Yes	COURSE.SECTION	MV Pointer	
SEC.SHORT.TITI	Yes	COURSE.SECTION	Data	
SEC.XLIST	Yes	COURSE.SECTION	SV Pointer	
SEC.TERM	Yes	COURSE.SECTION	Data	
SEC.FACULTY.IN	Yes	COURSE.SECTION	Data	
SEC.DEPTS	Yes	COURSE.SECTION	Association	SEC.DEPARTMENT
SEC.STATUS	Yes	COURSE.SECTION	Association	SEC.STATUSES
COURSE.SEC.FAC	Yes	COURSE.SEC.FAC	Key	
CSF.FACULTY	Yes	COURSE.SEC.FAC	SV Pointer	
FAC.USER3	Yes	FACULTY	Data	
COURSES.ID	Yes	COURSES	Key	

Process Data Element Name	Reference Only	Source	Type	Association
CSXL.COURSE.SE	Yes	COURSE.SEC.XLI	MV Pointer	
ID	Yes	PERSON	Key	
LAST.NAME	Yes	PERSON	Data	

Process Entry

```
EOL = ''
TRUE = 1
FALSE = 0
LOGS = ''
QUOTE = '"'
PGNAME = 'XDPB32'
X.FILE.CREATED = FALSE
X.WORK.DIR = 'PUC.CANVAS.UPLOAD.DIR'
X.LOG.LEVEL = 'ERROR'
X.CANVAS.URI = 'https://puc.instructure.com'

UFILE.NAME = 'users.csv'
UHEADING =
'"user_id","login_id","password","first_name","last_name","email","statu
:@FM
* "0253019","rhiebert","","Ryan","Hiebert","rhiebert@puc.edu","active"

EFILE.NAME = 'enrollments.csv'
EHEADING = 'course_id,user_id,role,section_id,status':@FM
* ,0253019,student,NURS-255-01-11SP,active

SWAP @FM WITH EOL IN UHEADING
SWAP @FM WITH EOL IN EHEADING

ENROLL.FILE.NAME = 'teachers.csv'
ENROLL.HEADING = 'course_id,user_id,role,section_id,status':@FM
* ,0000087,teacher,ESTH-497-01-11SU,active

XLISTS.FILE.NAME = 'xlists.csv'
XLISTS.HEADING = 'xlist_course_id,section_id,status':@FM
* HIST-360-01-11SU,RELH-360-01-11SU,active

SECTIONS.FILE.NAME = 'sections.csv'
SECTIONS.HEADING =
'"section_id","course_id","name","status","start_date","end_date"':@FM
* "AVIA-176-01-11SU","AVIA-176-01-11SU","AVIA-176-01 11SU","active","",'


COURSES.FILE.NAME = 'courses.csv'
COURSES.HEADING =
'"course_id","short_name","long_name","account_id","term_id","status"':
@FM

```

```
* "AVIA-176-01-11SU","AVIA-176-01 11SU","AVIA-176-01 PRIV FLIGHT TRAININ

 38
 39 SWAP @FM WITH EOL IN ENROLL.HEADING
 40 SWAP @FM WITH EOL IN XLISTS.HEADING
 41 SWAP @FM WITH EOL IN SECTIONS.HEADING
 42 SWAP @FM WITH EOL IN COURSES.HEADING
 43
 44 USERS = FALSE
 45 ENROLLMENTS = FALSE
 46 ENROLLS = FALSE
 47 SECTIONS = FALSE
 48 COURSES = FALSE
 49 XLISTS = FALSE
```

Main Code Body

```
    1 * to switch from debug to live:
    2 * 1) set X.DEBUG = FALSE on line 8
    3
* 2) set X.DEBUG.PS.SCRIPT as you wish.  If set to TRUE it will dump the

    4
*                                        Set to FALSE to not dump the sc

    5 * 3) clean up log file by deleting a bunch of it (optional)
    6
    7 * set to false for live running
    8 X.DEBUG = FALSE
    9 X.DEBUG.PS.SCRIPT = FALSE
   10
   11 * main processing
   12
   13 GOSUB GET.TERMS
   14 GOSUB XDEBUG.001
   15 GOSUB GET.LOG.COUNTER
   16 GOSUB CHECK.CSV.EXIST
   17 IF (X.ABORT EQ FALSE) THEN
   18    IF (MASTER.RUN EQ TRUE) THEN
   19       X.TMP = TRIM(TERMS,"'")
   20       SWAP "'" WITH @VM IN X.TMP
   21       XL.TERM.LIST = X.TMP
   22
   23       X.MAIN.TERM.CNT = DCOUNT(XL.TERM.LIST,@VM)
   24
   25       X.TMP = TRIM(ADDED.TERMS,"'")
   26       IF (TRIM(X.TMP) NE '') THEN
   27          SWAP "'" WITH @VM IN X.TMP
   28          XL.TERM.LIST<1,-1> = X.TMP
   29       END
   30       X.FULL.TERM.CNT = DCOUNT(XL.TERM.LIST,@VM)
   31
   32       FOR X.LOOP = 1 TO X.FULL.TERM.CNT
   33          USERS = FALSE
   34          ENROLLMENTS = FALSE
```

```
         ENROLLS = FALSE
         SECTIONS = FALSE
         COURSES = FALSE
         XLISTS = FALSE
         TERMS = "'":FIELD(XL.TERM.LIST,@VM,X.LOOP,1):"'"
         ADDED.TERMS = "''"
         IF (X.LOOP GT X.MAIN.TERM.CNT) THEN
            ADDED.TERMS = TERMS
            TERMS = "''"
         END
         TERMS.PLUS.ADDED = TERMS:ADDED.TERMS
         SWAP "''''" WITH "'" IN TERMS.PLUS.ADDED
         GOSUB XDEBUG.002
         GOSUB CREATE.USERS
         GOSUB CREATE.ENROLLMENTS
         GOSUB CREATE.TEACHERS
         GOSUB CREATE.COURSES
         GOSUB CREATE.SECTIONS
         GOSUB CREATE.XLISTS
         GOSUB UPLOAD.TO.CANVAS
      NEXT X.LOOP
   END ELSE
      GOSUB CREATE.USERS
      GOSUB CREATE.ENROLLMENTS
      GOSUB UPLOAD.TO.CANVAS
   END
END
GOSUB WRITE.LOG
GOSUB WRITE.LOG.COUNTER

RETURN

****************************************
GET.TERMS:
****************************************

* retrieve settings (TERMS)

* note TERMS and ADDED.TERMS are not @VM separated.  they are terms in

* ADDED.TERMS is used to only upload courses and teachers, but not enro

V.PUC.CONTROL.POINT.ID = 'CANVAS.SETTINGS'
FOR_THIS SINGLE REFERENCED PUC.CONTROL.POINT.ID
   TERMS = FIELD(V.XFIELD1,@VM,2,1)
   LAST.RUN.DATE = FIELD(V.XFIELD2,@VM,2,1)
   ADDED.TERMS = FIELD(V.XFIELD3,@VM,2,1)
   UPLOAD.FILES = FIELD(V.XFIELD4,@VM,2,1)
END_THIS PUC.CONTROL.POINT.ID
MASTER.RUN = FALSE
IF (LAST.RUN.DATE NE DATE()) THEN MASTER.RUN = TRUE

IF (X.LOG.LEVEL EQ 'ALL') THEN
```

```
   LOGS<-1> = 'TERMS: ':TERMS
   LOGS<-1> = 'LAST.RUN.DATE: ':LAST.RUN.DATE
   LOGS<-1> = 'ADDED.TERMS: ':ADDED.TERMS
   LOGS<-1> = 'UPLOAD.FILES: ':UPLOAD.FILES
   LOGS<-1> = 'MASTER.RUN: ':MASTER.RUN
END

RETURN

****************************************
GET.LOG.COUNTER:
****************************************

* get log counter and increment it in case we write logs out
V.PUC.CONTROL.POINT.ID = 'PUC.CANVAS.LOG.CTR'
FOR_THIS SINGLE REFERENCED PUC.CONTROL.POINT.ID
   CANVAS.LOG.CTR = V.XFIELD1+1
END_THIS PUC.CONTROL.POINT.ID

RETURN

****************************************
CHECK.CSV.EXIST:
****************************************


* check for existence of csv files in Upload directory.  If there are ar
* files, abort all further processing as the previous process did not e

X.ABORT = FALSE
SQL = "CLS":@FM:'SELECT ':X.WORK.DIR:' WITH @ID LIKE "....csv"'
CALL S.EXECUTE('-C ':SQL)
IF (X.LOG.LEVEL EQ 'ALL') THEN
   LOGS<-1> = LOWER(SQL) ; LOGS<-1> = CAPTURED.OUTPUT
END
IF (@SYSTEM.RETURN.CODE GT 0) THEN
   MSG1 = PGNAME:' Error: .csv file(s) exist in upload directory ':X.WORK.DIR:', processing aborted.'
   LOGS<-1> = MSG1
   CALL S.POPMAIL('SEND','canvas@puc.edu',MSG1,MSG1)
   X.ABORT = TRUE
END

RETURN

****************************************
CREATE.USERS:
****************************************

   X.ERR = ''
   X.MSG = ''
   TOTAL.USERS = 0
   TOTAL.ENROLLMENTS = 0
   TOTAL.ENROLLS = 0
   TOTAL.SECTIONS = 0
   TOTAL.COURSES = 0
   TOTAL.XLISTS = 0


* get list of new students to add to canvas, and also of all teachers/faculty teaching this term
* also get list of all employees and add them as well.  these will have
* should they be added manually to a course, their records will already
   IF (X.LOG.LEVEL EQ 'ALL') THEN
      LOGS<-1> = '********* Creating users.csv (teachers, staff and students)'
   END
   SQLP = "CLS":@FM:"SELECT STUDENT.TERMS "
   SQL = "WITH STTR.TERM EQ ":TERMS:" AND STTR.CURRENT.STATUS EQ 'R''T''X''P''W' "
* if not running a master list then include the uploaded already flag i
   IF (MASTER.RUN EQ FALSE) THEN SQL := "AND XSTTR.XPCST.STATUS NE 'A' "
   SQL := "AND STTR.STUDENT.SORT.LAST.NAME UNLIKE '*...' BY STTR.STUDENT.SORT.NAM
   CALL S.EXECUTE('-C ':SQLP:SQL)
   IF (X.LOG.LEVEL EQ 'ALL') THEN
      LOGS<-1> = LOWER(SQLP):SQL ; LOGS<-1> = CAPTURED.OUTPUT
   END
   X.USERS.CNT = @SYSTEM.RETURN.CODE
   IF (X.USERS.CNT LT 0) THEN X.USERS.CNT = 0


* check teachers now and use the additional terms.  we only do teachers
* but we run them first because if a teacher is also a student, we want
   * and skip them as a student.
   IF (MASTER.RUN EQ TRUE) THEN
      SQLP2 = "CLS":@FM:"SELECT COURSE.SEC.FACULTY "
      SQL2 = "WITH SEC.TERM EQ ":TERMS.PLUS.ADDED:" "
      SQL2 := "BY CSF.FACULTY.LAST.NAME"
      CALL S.EXECUTE('-C ':SQLP2:SQL2)
      IF (X.LOG.LEVEL EQ 'ALL') THEN
         LOGS<-1> = LOWER(SQLP2):SQL2 ; LOGS<-1> = CAPTURED.OUTPUT
      END
      IF (@SYSTEM.RETURN.CODE GT 0) THEN
         X.USERS.CNT += @SYSTEM.RETURN.CODE
      END
   END

   IF (X.USERS.CNT GT 0) THEN
      USERS = TRUE
      XL.USER.LIST = ''
      * open hold file for users
      CALL S.OPEN.SEQ('HOLD',UFILE.NAME,8,'W',X.ERR,X.MSG)
      IF (X.ERR) THEN LOGS<-1> = UFILE.NAME:' Open ':X.MSG

      * write column headings on users file
      CALL S.WRITE.SEQ(UHEADING,8,X.ERR,X.MSG)
      IF (X.ERR) THEN LOGS<-1> = UFILE.NAME:' Write ':X.MSG

      * get list of all teachers and add them to the file
      IF (MASTER.RUN EQ TRUE) THEN
         FOR_EACH REFERENCED SELECTED COURSE.SEC.FACULTY.ID USING NEWLIST CRITERIA SQL2
            IF (V.CSF.FACULTY # '0156474') THEN
               FIND V.CSF.FACULTY IN XL.USER.LIST SETTING F,V,S ELSE
                  * teacher not found, so add them to the list
                  XL.USER.LIST<-1> = V.CSF.FACULTY
* check that puc email type is actually a puc.edu email.
                  X.EMAIL = ''
                  SKIP.USER = FALSE
                  CALL XS.GET.PUC.EMAIL(X.EMAIL,V.CSF.FACULTY)
                  IF (X.EMAIL # '' AND X.EMAIL[8] NE '@puc.edu') THEN
                     X.MSG = PGNAME:' Error: user has invalid PUC email: ':V.CSF.FACULTY
                     CALL S.POPMAIL('SEND','canvas@puc.edu',X.MSG,X.MSG)
                     LOGS<-1> = X.MSG
                     SKIP.USER = TRUE
                  END
                  IF (SKIP.USER EQ FALSE) THEN
                     X.EMAIL.NAME = ''
                     CALL XS.GET.PUC.EMAIL.NAME(X.EMAIL.NAME,V.CSF.FACULTY)
                     IF (X.EMAIL.NAME EQ '') THEN
                        * create account here
                        SLEEP 1
                        MSG1 = ''
                        LOG = ''
                        CALL XS.CREATE.ACTIVATE.ACCOUNT(V.CSF.FACULTY,X.EMAIL.NAME,MSG1,LOG)
                        LOGS<-1> = "Created: ":X.EMAIL.NAME:" - ":MSG1
                     END
                     X.NAME = ''
                     FOR_THIS REFERENCED SECONDARY CSF.FACULTY
                        * get the preferred name in last name, first order
                        X.FNAME = FIELD(V.FAC.USER3,",",2,1)
                        X.LNAME = FIELD(V.FAC.USER3,",",1,1)
                        CONVERT " " TO "" IN X.FNAME
                        CONVERT " " TO "" IN X.LNAME
                     END_THIS CSF.FACULTY
                     X.LINE = QUOTE:V.CSF.FACULTY:QUOTE:','
                     X.LINE := QUOTE:X.EMAIL.NAME:QUOTE:','
                     X.LINE := QUOTE:QUOTE:','
                     X.LINE := QUOTE:X.FNAME:QUOTE:','
                     X.LINE := QUOTE:X.LNAME:QUOTE:','
                     X.LINE := QUOTE:X.EMAIL.NAME:"@puc.edu":QUOTE:','
                     X.LINE := QUOTE:'active':QUOTE
                     X.LINE := EOL
                     CALL S.WRITE.SEQ(X.LINE,8,X.ERR,X.MSG)
                     IF (X.ERR) THEN LOGS<-1> = UFILE.NAME:' Write ':X.MSG
                     TOTAL.USERS += 1
                  END
               END
            END
         END_EACH COURSE.SEC.FACULTY.ID
      END

      * get list of all staff and add them to the file
      IF (MASTER.RUN EQ TRUE) THEN
         SQL3 = "CLS":@FM:'SELECT HRPER WITH XHRP.STAFF.EMPLOYED EQ 1 BY XHRP.NAME'
         CALL S.EXECUTE('-C ':SQL3)
         IF (X.LOG.LEVEL EQ 'ALL') THEN
            LOGS<-1> = LOWER(SQL3) ; LOGS<-1> = CAPTURED.OUTPUT
         END
         IF (@SYSTEM.RETURN.CODE GT 0) THEN
            X.STATUS = 0
            XL.LIST = ''
            CALL S.READLIST(XL.LIST,0,X.STATUS)
            IF (X.STATUS EQ 1 AND DCOUNT(XL.LIST,@FM) GT 0) THEN
               COL = 1
               LOOP
                  REMOVE V.ID FROM XL.LIST AT COL SETTING END.MARK
                  FOR_THIS SINGLE REFERENCED ID
                     X.ID = V.ID
                     FIND X.ID IN XL.USER.LIST SETTING F,V,S ELSE
                        * staff not found, so add them to the list
                        XL.USER.LIST<-1> = X.ID
                        * check that puc email type is actually a puc.edu email.
                        X.EMAIL = ''
```

```
                        SKIP.USER = FALSE
                        CALL XS.GET.PUC.EMAIL(X.EMAIL,X.ID)
                        IF (X.EMAIL # '' AND X.EMAIL[8] NE '@puc.edu') THEN
                           X.MSG = PGNAME:' Error: user has invalid PUC email: ':X.ID
                           CALL S.POPMAIL('SEND','canvas@puc.edu',X.MSG,X.MSG)
                           LOGS<-1> = X.MSG
                           SKIP.USER = TRUE
                        END
                        IF (SKIP.USER EQ FALSE) THEN
                           X.EMAIL.NAME = ''
                           CALL XS.GET.PUC.EMAIL.NAME(X.EMAIL.NAME,X.ID)
                           IF (X.EMAIL.NAME EQ '') THEN
                              * create account here
                              SLEEP 1
                              MSG1 = ''
                              LOG = ''
                              CALL XS.CREATE.ACTIVATE.ACCOUNT(X.ID,X.EMAIL.NAME,MSG1,LOG)
                              LOGS<-1> = "Created: ":X.EMAIL.NAME:" - ":MSG1
                           END
                           X.NAME = ''
                           CALL XS.GET.NAME(X.NAME,X.ID,"LF")
                           X.FNAME = FIELD(X.NAME,",",2,1)
                           X.LNAME = FIELD(X.NAME,",",1,1)
                           CONVERT " " TO "" IN X.FNAME
                           CONVERT " " TO "" IN X.LNAME
                           X.LINE = QUOTE:X.ID:QUOTE:','
                           X.LINE := QUOTE:X.EMAIL.NAME:QUOTE:','
                           X.LINE := QUOTE:QUOTE:','
                           X.LINE := QUOTE:X.FNAME:QUOTE:','
                           X.LINE := QUOTE:X.LNAME:QUOTE:','
                           X.LINE := QUOTE:X.EMAIL.NAME:"@puc.edu":QUOTE:','
                           X.LINE := QUOTE:'active':QUOTE
                           X.LINE := EOL
                           CALL S.WRITE.SEQ(X.LINE,8,X.ERR,X.MSG)
                           IF (X.ERR) THEN LOGS<-1> = UFILE.NAME:' Write ':X.MSG
                           TOTAL.USERS += 1
                        END
                     END
                  END_THIS ID
               WHILE END.MARK DO REPEAT
            END
         END
      END


      IF (TERMS # "''") THEN
```

```
        FOR_EACH REFERENCED SELECTED STTR.STUDENT USING NEWLIST CRITERIA SQL
* this is necessary in case a person is actively registered in more tha
* we don't want to add them twice as they could have been added in a pre
           V.PUC.CANVAS.STU.TERMS.ID = V.STTR.STUDENT:"*": V.STTR.TERM:"*":V.STTR.ACAD.LEVEL
           FIND V.STTR.STUDENT IN XL.USER.LIST SETTING F,V,S ELSE
* student not found, so add them to the list and process away
              XL.USER.LIST<-1> = V.STTR.STUDENT
* check that puc email type is actually a puc.edu email.
              X.EMAIL = ''
              SKIP.USER = FALSE
              CALL XS.GET.PUC.EMAIL(X.EMAIL,V.STTR.STUDENT)
              IF (X.EMAIL # '' AND X.EMAIL[8] NE '@puc.edu') THEN
                 X.MSG = PGNAME: ' Error: user has invalid PUC email: ':V.STTR.STUDENT
                 CALL S.POPMAIL('SEND','canvas@puc.edu',X.MSG, X.MSG)
                 LOGS<-1> = X.MSG
                 SKIP.USER = TRUE
              END
              IF (SKIP.USER EQ FALSE) THEN
                 X.EMAIL.NAME = ''
                 CALL XS.GET.PUC.EMAIL.NAME(X.EMAIL.NAME, V.STTR.STUDENT)
                 IF (X.EMAIL.NAME EQ '') THEN
                    * create account here

                    MSG1 = ''
                    LOG = ''
                    CALL XS.CREATE.ACTIVATE.ACCOUNT( V.STTR.STUDENT,X.EMAIL.NAME,MSG1,LOG)
                    LOGS<-1> = "Created: ":X.EMAIL.NAME:" - ": MSG1
                 END
                 X.NAME = ''
                 CALL XS.GET.NAME(X.NAME,V.STTR.STUDENT,"LF")
                 X.FNAME = FIELD(X.NAME,",",2,1)
                 X.LNAME = FIELD(X.NAME,",",1,1)
                 CONVERT " " TO "" IN X.FNAME
                 CONVERT " " TO "" IN X.LNAME
                 X.LINE = QUOTE:V.STTR.STUDENT:QUOTE:','
                 X.LINE := QUOTE:X.EMAIL.NAME:QUOTE:','
                 X.LINE := QUOTE:QUOTE:','
                 X.LINE := QUOTE:X.FNAME:QUOTE:','
                 X.LINE := QUOTE:X.LNAME:QUOTE:','
```

```
                  X.LINE := QUOTE:X.EMAIL.NAME:"@puc.edu":QUOTE:','
                  X.LINE := QUOTE:'active':QUOTE
                  X.LINE := EOL
                  CALL S.WRITE.SEQ(X.LINE,8,X.ERR,X.MSG)
                  IF (X.ERR) THEN LOGS<-1> = UFILE.NAME:' Write ':X.MSG
* write data to the flag table.  A means added.  It is the only code use

                  IF (X.DEBUG EQ FALSE) THEN
                     FOR_THIS SINGLE PUC.CANVAS.STU.TERMS.ID
                        V.XPCST.STATUS = "A"
                        V.XPCST.DATE = DATE()
                        V.XPCST.TIME = TIME()
                     END_THIS PUC.CANVAS.STU.TERMS.ID
                  END
                  TOTAL.USERS += 1
               END
            END
         END_EACH STTR.STUDENT
      END

      * close hold files
      CALL S.WRITE.EOF(8,X.ERR,X.MSG)
      IF (X.ERR) THEN LOGS<-1> = UFILE.NAME:' Write ':X.MSG
      CALL S.CLOSE.SEQ(8,X.ERR,X.MSG)
      IF (X.ERR) THEN LOGS<-1> = UFILE.NAME:' Close ':X.MSG
      SQL = "CLS":@FM:"COPY FROM HOLD TO ":X.WORK.DIR:" ":UFILE.NAME
      CALL S.EXECUTE('-C ':SQL)
      IF (X.LOG.LEVEL EQ 'ALL') THEN
         LOGS<-1> = '' ; LOGS<-1> = LOWER(SQL) ; LOGS<-1> = CAPTURED.OUTPUT
      END
   * end of processing students and teachers for the terms given
   END

RETURN

****************************************
CREATE.ENROLLMENTS:
****************************************

   * process student enrollment changes only
   IF (X.LOG.LEVEL EQ 'ALL') THEN
      LOGS<-1> = "********* Creating enrollments.csv."
      LOGS<-1> = "Each dot represents one enrollment record uploaded."
   END
   DOTS = ''
   FILE.OPEN.YET = FALSE
   SQL = "WITH STTR.TERM EQ ":TERMS:" AND STTR.CURRENT.STATUS EQ 'R''T''X''P''W' "
```

```
* only deal with students who have been uploaded (their status flag set
SQL := "AND XSTTR.XPCST.STATUS EQ 'A' "
SQL := "AND STTR.STUDENT.SORT.LAST.NAME UNLIKE '*...' BY STTR.STUDENT.SORT.LAST
IF (TERMS # "''") THEN
   FOR_EACH REFERENCED SELECTED STTR.STUDENT USING NEWLIST CRITERIA SQL
*IF (X.DEBUG EQ TRUE) THEN CRT '----------':V.STTR.STUDENT:'--------':V
      XL.STUD.ACAD.REC = VL.STTR.STUDENT.ACAD.CRED
      COL = 1
      LOOP
         REMOVE V.STUDENT.ACAD.CRED.ID FROM XL.STUD.ACAD.REC AT COL SETTING END.MARK
         FOR_THIS SINGLE REFERENCED STUDENT.ACAD.CRED.ID
*IF (X.DEBUG EQ TRUE) THEN CRT V.STUDENT.ACAD.CRED.ID
            * evaluate individual enrollment
            * we will add it to the file if:
            * XPCSAC.STATUS is blank or
* STC.STATUS<1,1> is 'N''A' and XPCSAC.STATUS is not 'A'
* STC.STATUS<1,1> is anything else and XPCSAC.STATUS is 'A' ('D' is the
            * MASTER.RUN is true
            V.PUC.CANVAS.STU.ACAD.CRED.ID = V.STUDENT.ACAD.CRED.ID
            FOR_THIS SINGLE PUC.CANVAS.STU.ACAD.CRED.ID
               ADD.REC = FALSE
               IF (V.XPCSAC.STATUS EQ '' OR MASTER.RUN EQ TRUE ) THEN
                  ADD.REC = TRUE
               END ELSE
                  X.STC.STATUS = FIELD(VL.STC.STATUS,@VM,1,1)
                  IF (X.STC.STATUS EQ 'N' OR X.STC.STATUS EQ 'A') THEN
                     IF (V.XPCSAC.STATUS NE 'A') THEN ADD.REC = TRUE
                  END ELSE
                     IF (V.XPCSAC.STATUS EQ 'A') THEN ADD.REC = TRUE
                  END
               END
* get the course and section numbers (these can be gotten from the stude
* but may be wrong, so this is more accurate hopefully)
               FOR_THIS SINGLE SECONDARY
```

```
STC.STUDENT.COURSE.SEC
                           FOR_THIS SINGLE REFERENCED SECONDARY SCS.COURSE.SECTION
                              X.COURSE.SEC = V.SEC.SUBJECT:'-':V.SEC.COURSE.NO:'-':V.SEC.NO
                           END_THIS SCS.COURSE.SECTION
                        END_THIS STC.STUDENT.COURSE.SEC
* debug for me only just to see how accurate the data is
                        IF (X.COURSE.SEC # V.STC.COURSE.NAME:"-":V.STC.SECTION.NO AND V.STC.COURSE.NAME # '' AND V.STC.COURSE.NAME # 'WITH-000') THEN
*MSG1 = PGNAME:" Error: Student ":V.STTR.STUDENT:" Course Section may n
*CALL S.POPMAIL('SEND','lantern@puc.edu',MSG1,MSG1)
                        END
* checking stc.course.name because there has been a case of a blank
                        * course/section in the enrollments file. this should stop that from happening.
                        IF (ADD.REC EQ TRUE AND V.STC.COURSE.NAME # '') THEN
                           IF (FILE.OPEN.YET EQ FALSE) THEN
                              * open hold file for enrollments
                              CALL S.OPEN.SEQ('HOLD',EFILE.NAME,2,'W',X.ERR,X.MSG)
                              IF (X.ERR) THEN LOGS<-1> = EFILE.NAME:' Open ':X.MSG
* write column headings on enrollments file
                              CALL S.WRITE.SEQ(EHEADING,2,X.ERR,X.MSG)
                              IF (X.ERR) THEN LOGS<-1> = EFILE.NAME:' Write ':X.MSG
                              ENROLLMENTS = TRUE
                              FILE.OPEN.YET = TRUE
                           END
                           X.LINE = ','
                           X.LINE := V.STTR.STUDENT
                           X.LINE := ','
                           X.LINE := 'student'
                           X.LINE := ','
                           X.TERM1 = V.STC.TERM
                           SWAP "/" WITH '' IN X.TERM1
                           X.LINE := X.COURSE.SEC:'-':X.TERM1
                           X.LINE := ','
                           X.STC.STATUS = FIELD(VL.STC.STATUS,@VM,1,1)
                           IF (X.STC.STATUS EQ 'N' OR X.STC.STATUS EQ 'A') THEN
                              X.LINE := 'active'
* write data to the student.acad.cred record.  A means added, D means de
```

```
                        IF (X.DEBUG EQ FALSE) THEN V.XPCSAC.STATUS = "A"
                     END ELSE
                        X.LINE := 'deleted'
                        IF (X.DEBUG EQ FALSE) THEN V.XPCSAC.STATUS = "D"
                     END
                     X.LINE := EOL
                     CALL S.WRITE.SEQ(X.LINE,2,X.ERR,X.MSG)
                     IF (X.ERR) THEN LOGS<-1> = EFILE.NAME:' Write ':X.MSG
                     IF (X.DEBUG EQ FALSE) THEN V.XPCSAC.DATE = DATE()
                     IF (X.DEBUG EQ FALSE) THEN V.XPCSAC.TIME = TIME()
                     DOTS := '.'
                     TOTAL.ENROLLMENTS += 1
                  END
               END_THIS PUC.CANVAS.STU.ACAD.CRED.ID
            END_THIS STUDENT.ACAD.CRED.ID
         WHILE END.MARK DO REPEAT
      END_EACH STTR.STUDENT
   END
   IF (ENROLLMENTS EQ TRUE) THEN
      IF (X.LOG.LEVEL EQ 'ALL') THEN
         LOGS<-1> = DOTS
      END
      CALL S.WRITE.EOF(2,X.ERR,X.MSG)
      IF (X.ERR) THEN LOGS<-1> = EFILE.NAME:' Write ':X.MSG
      CALL S.CLOSE.SEQ(2,X.ERR,X.MSG)
      IF (X.ERR) THEN LOGS<-1> = EFILE.NAME:' Close ':X.MSG
      SQL = "CLS":@FM:"COPY FROM HOLD TO ":X.WORK.DIR:" ":EFILE.NAME
      CALL S.EXECUTE('-C ':SQL)
      IF (X.LOG.LEVEL EQ 'ALL') THEN
         LOGS<-1> = '' ; LOGS<-1> = LOWER(SQL) ; LOGS<-1> = CAPTURED.OUTPUT
      END
   END ELSE
      IF (FILE.OPEN.YET EQ TRUE) THEN
         CALL S.CLOSE.SEQ(2,X.ERR,X.MSG)
         IF (X.ERR) THEN LOGS<-1> = EFILE.NAME:' Close ':X.MSG
      END
      IF (X.LOG.LEVEL EQ 'ALL') THEN
         LOGS<-1> = 'No enrollment changes found.'
      END
   END
RETURN

****************************************
CREATE.TEACHERS:
****************************************

      * process teacher enroll records using additional terms
```

```
IF (X.LOG.LEVEL EQ 'ALL') THEN
   LOGS<-1> = "********* Creating teachers.csv."
   LOGS<-1> = "Each dot represents one teacher record uploaded."
END
DOTS = ''
ENROLLS = FALSE
SQL = "CLS":@FM:"SELECT COURSE.SECTIONS WITH SEC.TERM EQ ":TERMS.PLUS.ADDED:" AND WITH EVERY SEC.DEPTS NE 'ND' SAVING UNIQUE SEC.FACULTY"
SQL<-1> = "SELECT COURSE.SEC.FACULTY REQUIRE.SELECT SAVING UNIQUE CSF.FACULTY"
SQL<-1> = "SELECT FACULTY WITH @ID UNLIKE '0156474' REQUIRE.SELECT"
CALL S.EXECUTE('-C ':SQL)
IF (X.LOG.LEVEL EQ 'ALL') THEN
   LOGS<-1> = LOWER(SQL) ; LOGS<-1> = CAPTURED.OUTPUT
END
X.FAC.CNT = @SYSTEM.RETURN.CODE
CALL S.EXECUTE("CLS")
IF (X.FAC.CNT GT 0) THEN
   * open hold file for enrolls
   CALL S.OPEN.SEQ('HOLD',ENROLL.FILE.NAME,2,'W',X.ERR,X.MSG)
   IF (X.ERR) THEN LOGS<-1> = ENROLL.FILE.NAME:' Open ':X.MSG
   * write column headings on enrolls file
   CALL S.WRITE.SEQ(ENROLL.HEADING,2,X.ERR,X.MSG)
   IF (X.ERR) THEN LOGS<-1> = ENROLL.FILE.NAME:' Write ':X.MSG
   ENROLLS = TRUE
   XL.SORT.ARRAY = ''
   XL.ARR1 = ''
   XL.ARR2 = ''
   SQL = "WITH SEC.TERM EQ ":TERMS.PLUS.ADDED:" AND WITH EVERY SEC.DEPTS NE 'ND' BY SEC.TERM BY SEC.FACULTY.SORT.NAME'

   FOR_EACH REFERENCED SELECTED COURSE.SECTIONS.ID USING NEWLIST CRITERIA SQL
      X.COURSE.SEC = V.SEC.SUBJECT:'-':V.SEC.COURSE.NO:'-':V.SEC.NO
      FOR_EACH REFERENCED SECONDARY SEC.FACULTY
         IF (V.CSF.FACULTY NE '0156474' AND V.CSF.FACULTY NE '') THEN
            X.TERM1 = V.SEC.TERM
            SWAP "/" WITH '' IN X.TERM1
            XL.SORT.ARRAY<1,-1> = V.CSF.FACULTY:X.TERM1:X.COURSE.SEC
            XL.ARR1<1,-1> = V.CSF.FACULTY
            XL.ARR2<1,-1> = X.COURSE.SEC:'-':X.TERM1
            DOTS := '.'
            TOTAL.ENROLLS += 1
         END
      END_EACH SEC.FACULTY
```

```
         END_EACH COURSE.SECTIONS.ID
      END
      IF (ENROLLS EQ TRUE) THEN
         CALL S.SORT.ASSOC.AL(XL.SORT.ARRAY,XL.ARR1,XL.ARR2,"","","","","","","")
         * Write out the sorted lines
         FOR X = 1 TO TOTAL.ENROLLS
            X.LINE = ','
            X.LINE := XL.ARR1<1,X>
            X.LINE := ','
            X.LINE := 'teacher'
            X.LINE := ','
            X.LINE := XL.ARR2<1,X>
            X.LINE := ',active'
            X.LINE := EOL
            CALL S.WRITE.SEQ(X.LINE,2,X.ERR,X.MSG)
            IF (X.ERR) THEN LOGS<-1> = ENROLL.FILE.NAME:' Write ':X.MSG
         NEXT X

         IF (X.LOG.LEVEL EQ 'ALL') THEN
            LOGS<-1> = DOTS
         END
         CALL S.WRITE.EOF(2,X.ERR,X.MSG)
         IF (X.ERR) THEN LOGS<-1> = ENROLL.FILE.NAME:' Write ':X.MSG
         CALL S.CLOSE.SEQ(2,X.ERR,X.MSG)
         IF (X.ERR) THEN LOGS<-1> = ENROLL.FILE.NAME:' Close ':X.MSG
         SQL = "CLS":@FM:"COPY FROM HOLD TO ":X.WORK.DIR:" ":ENROLL.FILE.NAME
         CALL S.EXECUTE('-C ':SQL)
         IF (X.LOG.LEVEL EQ 'ALL') THEN
            LOGS<-1> = '' ; LOGS<-1> = LOWER(SQL) ; LOGS<-1> = CAPTURED.OUTPUT
         END
      END ELSE
         CALL S.CLOSE.SEQ(2,X.ERR,X.MSG)
         IF (X.ERR) THEN LOGS<-1> = ENROLL.FILE.NAME:' Close ':X.MSG
         IF (X.LOG.LEVEL EQ 'ALL') THEN
            LOGS<-1> = 'No teacher enrolls found.'
         END
      END

RETURN

****************************************
CREATE.COURSES:
****************************************

      * process courses
      IF (X.LOG.LEVEL EQ 'ALL') THEN
         LOGS<-1> = "********* Creating courses.csv."
```

```
   LOGS<-1> = "Each dot represents one course record uploaded."
END
DOTS = ''
SQLP = "CLS":@FM:"SELECT COURSE.SECTIONS "
SQL = "WITH SEC.TERM EQ ":TERMS.PLUS.ADDED:" AND WITH EVERY SEC.DEPTS NE 'ND' BY SEC.TERM BY SEC.SUBJECT BY SEC.COL
CALL S.EXECUTE('-C ':SQLP:SQL)
IF (X.LOG.LEVEL EQ 'ALL') THEN
   LOGS<-1> = LOWER(SQL) ; LOGS<-1> = CAPTURED.OUTPUT
END
X.CRS.CNT = @SYSTEM.RETURN.CODE
CALL S.EXECUTE("CLS")
IF (X.CRS.CNT GT 0) THEN
   * open hold file for enrolls
   CALL S.OPEN.SEQ('HOLD',COURSES.FILE.NAME,2,'W',X.ERR,X.MSG)
   IF (X.ERR) THEN LOGS<-1> = COURSES.FILE.NAME:' Open ':X.MSG
   * write column headings on enrolls file
   CALL S.WRITE.SEQ(COURSES.HEADING,2,X.ERR,X.MSG)
   IF (X.ERR) THEN LOGS<-1> = COURSES.FILE.NAME:' Write ':X.MSG
   COURSES = TRUE
   FOR_EACH REFERENCED SELECTED COURSE.SECTIONS.ID USING NEWLIST CRITERIA SQL
      X.COURSE.SEC = V.SEC.SUBJECT:'-':V.SEC.COURSE.NO:'-':V.SEC.NO
      X.TERM1 = V.SEC.TERM
      SWAP "/" WITH '' IN X.TERM1
      V.COURSE.SEC.FACULTY.ID = FIELD(VL.SEC.FACULTY,@VM,1,1)
      FOR_THIS SINGLE REFERENCED COURSE.SEC.FACULTY.ID
         V.ID = V.CSF.FACULTY
         FOR_THIS SINGLE REFERENCED ID
         END_THIS ID
      END_THIS COURSE.SEC.FACULTY.ID
      X.LINE = QUOTE:X.COURSE.SEC:'-':X.TERM1:QUOTE
      X.LINE := ','
      X.LINE := QUOTE:X.COURSE.SEC:' ':X.TERM1:QUOTE
      X.LINE := ','
      IF (V.LAST.NAME EQ '') THEN
         X.LAST.NAME = 'Staff'
      END ELSE
         X.LAST.NAME = V.LAST.NAME
      END
      X.LINE := QUOTE:X.COURSE.SEC:' ':TRIM(V.SEC.SHORT.TITLE):' (':X.LAST.NAME:' ':X.TERM1:')':QUOTE
      X.LINE := ','
      X.LINE := QUOTE:FIELD(VL.SEC.DEPTS,@VM,1,1):QUOTE
      X.LINE := ','
      X.LINE := QUOTE:X.TERM1:QUOTE
      X.LINE := ','
```

```
            * check status
            IF (FIELD(VL.SEC.STATUS,@VM,1,1) EQ 'A') THEN
               X.LINE := QUOTE:'active':QUOTE
            END ELSE
               X.LINE := QUOTE:'deleted':QUOTE
            END
            X.LINE := EOL
            CALL S.WRITE.SEQ(X.LINE,2,X.ERR,X.MSG)
            IF (X.ERR) THEN LOGS<-1> = COURSES.FILE.NAME:' Write ':X.MSG
            DOTS := '.'
            TOTAL.COURSES += 1
         END_EACH COURSE.SECTIONS.ID
      END
      IF (COURSES EQ TRUE) THEN
         IF (X.LOG.LEVEL EQ 'ALL') THEN
            LOGS<-1> = DOTS
         END
         CALL S.WRITE.EOF(2,X.ERR,X.MSG)
         IF (X.ERR) THEN LOGS<-1> = COURSES.FILE.NAME:' Write ':X.MSG
         CALL S.CLOSE.SEQ(2,X.ERR,X.MSG)
         IF (X.ERR) THEN LOGS<-1> = COURSES.FILE.NAME:' Close ':X.MSG
         SQL = "CLS":@FM:"COPY FROM HOLD TO ":X.WORK.DIR:" ":COURSES.FILE.NAME
         CALL S.EXECUTE('-C ':SQL)
         IF (X.LOG.LEVEL EQ 'ALL') THEN
            LOGS<-1> = '' ; LOGS<-1> = LOWER(SQL) ; LOGS<-1> = CAPTURED.OUTPUT
         END
      END ELSE
         CALL S.CLOSE.SEQ(2,X.ERR,X.MSG)
         IF (X.ERR) THEN LOGS<-1> = COURSES.FILE.NAME:' Close ':X.MSG
         IF (X.LOG.LEVEL EQ 'ALL') THEN
            LOGS<-1> = 'No courses found.'
         END
      END

RETURN

****************************************
CREATE.SECTIONS:
****************************************

      * process sections
      IF (X.LOG.LEVEL EQ 'ALL') THEN
         LOGS<-1> = "********* Creating sections.csv."
         LOGS<-1> = "Each dot represents one section record uploaded."
      END
      DOTS = ''
      SQLP = "CLS":@FM:"SELECT COURSE.SECTIONS "
```

```
            SQL = "WITH SEC.TERM EQ ":TERMS.PLUS.ADDED:
" AND WITH EVERY SEC.DEPTS NE 'ND' BY SEC.TERM BY SEC.SUBJECT BY SEC.COL

            CALL S.EXECUTE('-C ':SQLP:SQL)
            IF (X.LOG.LEVEL EQ 'ALL') THEN
               LOGS<-1> = LOWER(SQL) ; LOGS<-1> = CAPTURED.OUTPUT
            END
            X.CRS.CNT = @SYSTEM.RETURN.CODE
            CALL S.EXECUTE("CLS")
            IF (X.CRS.CNT GT 0) THEN
               * open hold file for enrolls
               CALL S.OPEN.SEQ('HOLD',SECTIONS.FILE.NAME,2,'W',X.ERR,
X.MSG)
               IF (X.ERR) THEN LOGS<-1> = SECTIONS.FILE.NAME:' Open ':
X.MSG
               * write column headings on enrolls file
               CALL S.WRITE.SEQ(SECTIONS.HEADING,2,X.ERR,X.MSG)
               IF (X.ERR) THEN LOGS<-1> = SECTIONS.FILE.NAME:' Write ':
X.MSG
               SECTIONS = TRUE
               FOR_EACH REFERENCED SELECTED COURSE.SECTIONS.ID USING
NEWLIST CRITERIA SQL
                  X.COURSE.SEC = V.SEC.SUBJECT:'-':V.SEC.COURSE.NO:'-':
V.SEC.NO
                  X.TERM1 = V.SEC.TERM
                  SWAP "/" WITH '' IN X.TERM1
                  X.LINE = QUOTE:X.COURSE.SEC:'-':X.TERM1:QUOTE
                  X.LINE := ','
                  X.LINE := QUOTE:X.COURSE.SEC:'-':X.TERM1:QUOTE
                  X.LINE := ','
                  X.LINE := QUOTE:X.COURSE.SEC:' ':X.TERM1:QUOTE
                  X.LINE := ','
                  * check status
                  IF (FIELD(VL.SEC.STATUS,@VM,1,1) EQ 'A') THEN
                     X.LINE := QUOTE:'active':QUOTE
                  END ELSE
                     X.LINE := QUOTE:'deleted':QUOTE
                  END
                  X.LINE := ','
                  X.LINE := QUOTE:QUOTE
                  X.LINE := ','
                  X.LINE := QUOTE:QUOTE
                  X.LINE := EOL
                  CALL S.WRITE.SEQ(X.LINE,2,X.ERR,X.MSG)
                  IF (X.ERR) THEN LOGS<-1> = SECTIONS.FILE.NAME:
' Write ':X.MSG
                  DOTS := '.'
                  TOTAL.SECTIONS += 1
               END_EACH COURSE.SECTIONS.ID
            END
            IF (SECTIONS EQ TRUE) THEN
               IF (X.LOG.LEVEL EQ 'ALL') THEN
                  LOGS<-1> = DOTS
               END
```

```
         CALL S.WRITE.EOF(2,X.ERR,X.MSG)
         IF (X.ERR) THEN LOGS<-1> = SECTIONS.FILE.NAME:' Write ':X.MSG
         CALL S.CLOSE.SEQ(2,X.ERR,X.MSG)
         IF (X.ERR) THEN LOGS<-1> = SECTIONS.FILE.NAME:' Close ':X.MSG
         SQL = "CLS":@FM:"COPY FROM HOLD TO ":X.WORK.DIR:" ":SECTIONS.FILE.NAME
         CALL S.EXECUTE('-C ':SQL)
         IF (X.LOG.LEVEL EQ 'ALL') THEN
            LOGS<-1> = '' ; LOGS<-1> = LOWER(SQL) ; LOGS<-1> = CAPTURED.OUTPUT
         END
      END ELSE
         CALL S.CLOSE.SEQ(2,X.ERR,X.MSG)
         IF (X.ERR) THEN LOGS<-1> = SECTIONS.FILE.NAME:' Close ':X.MSG
         IF (X.LOG.LEVEL EQ 'ALL') THEN
            LOGS<-1> = 'No sections found.'
         END
      END

RETURN

****************************************
CREATE.XLISTS:
****************************************

      * process xlists
      IF (X.LOG.LEVEL EQ 'ALL') THEN
         LOGS<-1> = "********* Creating xlists.csv."
         LOGS<-1> = "Each dot represents one xlist record uploaded."
      END
      DOTS = ''
      * open hold file for enrolls
      CALL S.OPEN.SEQ('HOLD',XLISTS.FILE.NAME,2,'W',X.ERR,X.MSG)
      IF (X.ERR) THEN LOGS<-1> = XLISTS.FILE.NAME:' Open ':X.MSG
      * write column headings on enrolls file
      CALL S.WRITE.SEQ(XLISTS.HEADING,2,X.ERR,X.MSG)
      IF (X.ERR) THEN LOGS<-1> = XLISTS.FILE.NAME:' Write ':X.MSG
      XLISTS = TRUE
      SQL = "CLS":@FM:'SELECT COURSE.SECTIONS WITH SEC.TERM EQ ':TERMS.PLUS.ADDED:' AND WITH EVERY SEC.DEPTS NE "ND" AND SEC.XLIST NE "" SAVING UNIQUE CS

      CALL S.EXECUTE('-C ':SQL)
      IF (X.LOG.LEVEL EQ 'ALL') THEN
         LOGS<-1> = LOWER(SQL) ; LOGS<-1> = CAPTURED.OUTPUT
      END
      X.STATUS = 0
      XL.LIST = ''
      CALL S.READLIST(XL.LIST,0,X.STATUS)
      IF (X.STATUS EQ 1 AND DCOUNT(XL.LIST,@FM) GT 0) THEN
```

```
         COL = 1
         XL.SORT.ARRAY = ''
         XL.ARR1 = ''
         XL.ARR2 = ''
         XL.ARR3 = ''
         LOOP
            REMOVE V.COURSE.SECTIONS.ID FROM XL.LIST AT COL SETTING END.MARK
            * Get the primary xlist section
            FOR_THIS SINGLE REFERENCED COURSE.SECTIONS.ID
               X.TERM1 = V.SEC.TERM
               SWAP "/" WITH '' IN X.TERM1
               X.COURSE.SEC.TERM = V.SEC.SUBJECT:'-':V.SEC.COURSE.NO:'-':V.SEC.NO:'-':X.TERM1
               STAT1 = FIELD(VL.SEC.STATUS,@VM,1,1)
               MAIN.COURSE.SEC.ID = V.COURSE.SECTIONS.ID
               FOR_THIS SINGLE REFERENCED SECONDARY SEC.XLIST
                  FOR_EACH REFERENCED ASSOCIATED CSXL.COURSE.SECTIONS INTO COURSE.SECTIONS.ID
                     IF (MAIN.COURSE.SEC.ID # V.COURSE.SECTIONS.ID) THEN
                        X.TERM2 = V.SEC.TERM
                        SWAP "/" WITH '' IN X.TERM2
                        X.COURSE.SEC.TERM2 = V.SEC.SUBJECT:'-':V.SEC.COURSE.NO:'-':V.SEC.NO:'-':X.TERM2
                        STAT2 = FIELD(VL.SEC.STATUS,@VM,1,1)
                        XL.SORT.ARRAY<1,-1> = X.TERM1:X.COURSE.SEC.TERM:X.COURSE.SEC.TERM2
                        XL.ARR1<1,-1> = X.COURSE.SEC.TERM
                        XL.ARR2<1,-1> = X.COURSE.SEC.TERM2
                        IF (STAT1 EQ 'A' AND STAT2 EQ 'A') THEN
                           XL.ARR3<1,-1> = 'active'
                        END ELSE
                           XL.ARR3<1,-1> = 'deleted'
                        END
                        DOTS := '.'
                        TOTAL.XLISTS += 1
                     END
                  END_EACH CSXL.COURSE.SECTIONS
               END_THIS SEC.XLIST
            END_THIS COURSE.SECTIONS.ID
         WHILE END.MARK DO REPEAT
      END


* Process MUEN courses.  They all have a lower division (100) and upper

      SQL = "CLS":@FM:'SELECT COURSE.SECTIONS WITH SEC.TERM EQ ':TERMS.PLUS.ADDED:' AND WITH EVERY SEC.DEPTS NE "ND" AND SEC.SUBJECT EQ "MUEN"'
      CALL S.EXECUTE('-C ':SQL)
      IF (X.LOG.LEVEL EQ 'ALL') THEN
         LOGS<-1> = LOWER(SQL) ; LOGS<-1> = CAPTURED.OUTPUT
      END
      X.STATUS = 0
      XL.LIST = ''
      CALL S.READLIST(XL.LIST,0,X.STATUS)
      IF (X.STATUS EQ 1 AND DCOUNT(XL.LIST,@FM) GT 0) THEN
         COL = 1
         LOOP
            REMOVE V.COURSE.SECTIONS.ID FROM XL.LIST AT COL SETTING END.MARK
            * Get the primary xlist section
            FOR_THIS SINGLE REFERENCED COURSE.SECTIONS.ID
               X.TERM1 = V.SEC.TERM
               SWAP "/" WITH '' IN X.TERM1
               X.COURSE.SEC.TERM = V.SEC.SUBJECT:'-':V.SEC.COURSE.NO:'-':V.SEC.NO:'-':X.TERM1
               STAT1 = FIELD(VL.SEC.STATUS,@VM,1,1)
               MAIN.COURSE.SEC.ID = V.COURSE.SECTIONS.ID
               X.NO = V.SEC.COURSE.NO
               IF (X.NO LT 200) THEN
                  * change to upper division course number
                  X.NO[1,1] = '3'
                  SQL = 'WITH SEC.TERM EQ "':V.SEC.TERM:'" AND WITH EVERY SEC.DEPTS NE "ND" AND SEC.SUBJECT EQ "MUEN" AND SEC.C
:X.NO:'"'
                  FOR_EACH REFERENCED SELECTED COURSE.SECTIONS.ID USING NEWLIST CRITERIA SQL
* this should find a UD version, and only one even though we use FOR_EA

                     X.TERM2 = V.SEC.TERM
                     SWAP "/" WITH '' IN X.TERM2
                     X.COURSE.SEC.TERM2 = V.SEC.SUBJECT:'-':V.SEC.COURSE.NO:'-':V.SEC.NO:'-':X.TERM2
                     STAT2 = FIELD(VL.SEC.STATUS,@VM,1,1)
                     XL.SORT.ARRAY<1,-1> = X.TERM1:X.COURSE.SEC.TERM
                     XL.ARR1<1,-1> = X.COURSE.SEC.TERM
                     XL.ARR2<1,-1> = X.COURSE.SEC.TERM2
                     IF (STAT1 EQ 'A' AND STAT2 EQ 'A') THEN
                        XL.ARR3<1,-1> = 'active'
                     END ELSE
                        XL.ARR3<1,-1> = 'deleted'
                     END
                     DOTS := '.'
                     TOTAL.XLISTS += 1
                  END_EACH COURSE.SECTIONS.ID
               END
            END_THIS COURSE.SECTIONS.ID
         WHILE END.MARK DO REPEAT
      END


* Process LABs.  All sections get cross listed with the first section (

      SQL = "CLS":@FM:'SELECT COURSE.SECTIONS WITH SEC.TERM EQ ':
```

```
TERMS.PLUS.ADDED:' AND WITH EVERY SEC.DEPTS NE "ND" AND'
SQL := ' EVAL "INDEX(SEC.COURSE.NO,':"'L'":',
1)" GT 0 AND SEC.NO EQ "01"'
CALL S.EXECUTE('-C ':SQL)
IF (X.LOG.LEVEL EQ 'ALL') THEN
   LOGS<-1> = LOWER(SQL) ; LOGS<-1> = CAPTURED.OUTPUT
END
X.STATUS = 0
XL.LIST = ''
CALL S.READLIST(XL.LIST,0,X.STATUS)
IF (X.STATUS EQ 1 AND DCOUNT(XL.LIST,@FM) GT 0) THEN
   COL = 1
   LOOP
      REMOVE V.COURSE.SECTIONS.ID FROM XL.LIST AT COL
SETTING END.MARK
      * go through the primary (01) sections
      FOR_THIS SINGLE REFERENCED COURSE.SECTIONS.ID
         X.TERM1 = V.SEC.TERM
         SWAP "/" WITH '' IN X.TERM1
         X.COURSE.SEC.TERM = V.SEC.SUBJECT:'-':
V.SEC.COURSE.NO:'-':V.SEC.NO:'-':X.TERM1
         STAT1 = FIELD(VL.SEC.STATUS,@VM,1,1)
         MAIN.COURSE.SEC.ID = V.COURSE.SECTIONS.ID
         * locate any sub sections
         SQL = 'WITH SEC.TERM EQ "':V.SEC.TERM:
'" AND SEC.NO NE "01" AND SEC.COURSE.NO EQ "':V.SEC.COURSE.NO:
'" AND SEC.SUBJECT EQ "':V.SEC.SUBJECT:'"'
         FOR_EACH REFERENCED SELECTED COURSE.SECTIONS.ID
USING NEWLIST CRITERIA SQL

* there could be any number of these, so write each one out
            X.TERM2 = V.SEC.TERM
            SWAP "/" WITH '' IN X.TERM2
            X.COURSE.SEC.TERM2 = V.SEC.SUBJECT:'-':
V.SEC.COURSE.NO:'-':V.SEC.NO:'-':X.TERM2
            STAT2 = FIELD(VL.SEC.STATUS,@VM,1,1)
            XL.SORT.ARRAY<1,-1> = X.TERM1:X.COURSE.SEC.TERM
            XL.ARR1<1,-1> = X.COURSE.SEC.TERM
            XL.ARR2<1,-1> = X.COURSE.SEC.TERM2
            IF (STAT1 EQ 'A' AND STAT2 EQ 'A') THEN
               XL.ARR3<1,-1> = 'active'
            END ELSE
               XL.ARR3<1,-1> = 'deleted'
            END
            DOTS := '.'
            TOTAL.XLISTS += 1
         END_EACH COURSE.SECTIONS.ID
      END_THIS COURSE.SECTIONS.ID
   WHILE END.MARK DO REPEAT
END


* Process GNST401.  All sections get cross listed with the first sectior
```

```
SQL = "CLS":@FM:'SELECT COURSE.SECTIONS WITH SEC.TERM EQ ':TERMS.PLUS.ADDED:' AND WITH EVERY SEC.DEPTS NE "ND" AND'
SQL := ' SEC.COURSE.NO EQ "401" AND SEC.NO EQ "01" AND SEC.SUBJECT EQ "GNST"'
CALL S.EXECUTE('-C ':SQL)
IF (X.LOG.LEVEL EQ 'ALL') THEN
   LOGS<-1> = LOWER(SQL) ; LOGS<-1> = CAPTURED.OUTPUT
END
X.STATUS = 0
XL.LIST = ''
CALL S.READLIST(XL.LIST,0,X.STATUS)
IF (X.STATUS EQ 1 AND DCOUNT(XL.LIST,@FM) GT 0) THEN
   COL = 1
   LOOP
      REMOVE V.COURSE.SECTIONS.ID FROM XL.LIST AT COL SETTING END.MARK

* go through the primary (01) sections (there may be more than one becau

      FOR_THIS SINGLE REFERENCED COURSE.SECTIONS.ID
         X.TERM1 = V.SEC.TERM
         SWAP "/" WITH '' IN X.TERM1
         X.COURSE.SEC.TERM = V.SEC.SUBJECT:'-':V.SEC.COURSE.NO:'-':V.SEC.NO:'-':X.TERM1
         STAT1 = FIELD(VL.SEC.STATUS,@VM,1,1)
         MAIN.COURSE.SEC.ID = V.COURSE.SECTIONS.ID
         * locate any sub sections
         SQL = 'WITH SEC.TERM EQ "':V.SEC.TERM:'" AND SEC.NO NE "01" AND SEC.COURSE.NO EQ "401" AND SEC.SUBJECT EQ "GNS

         FOR_EACH REFERENCED SELECTED COURSE.SECTIONS.ID USING NEWLIST CRITERIA SQL

* there could be any number of these, so write each one out
            X.TERM2 = V.SEC.TERM
            SWAP "/" WITH '' IN X.TERM2
            X.COURSE.SEC.TERM2 = V.SEC.SUBJECT:'-':V.SEC.COURSE.NO:'-':V.SEC.NO:'-':X.TERM2
            STAT2 = FIELD(VL.SEC.STATUS,@VM,1,1)
            XL.SORT.ARRAY<1,-1> = X.TERM1:X.COURSE.SEC.TERM
            XL.ARR1<1,-1> = X.COURSE.SEC.TERM
            XL.ARR2<1,-1> = X.COURSE.SEC.TERM2
            IF (STAT1 EQ 'A' AND STAT2 EQ 'A') THEN
               XL.ARR3<1,-1> = 'active'
            END ELSE
               XL.ARR3<1,-1> = 'deleted'
            END
            DOTS := '.'
            TOTAL.XLISTS += 1
         END_EACH COURSE.SECTIONS.ID
      END_THIS COURSE.SECTIONS.ID
   WHILE END.MARK DO REPEAT
END

```

```
      IF (XLISTS EQ TRUE) THEN
         CALL S.SORT.ASSOC.AL(XL.SORT.ARRAY,XL.ARR1,XL.ARR2,XL.ARR3,"","","","","","")
         * Write out the sorted lines
         FOR X = 1 TO TOTAL.XLISTS
            X.LINE = XL.ARR1<1,X>
            X.LINE := ','
            X.LINE := XL.ARR2<1,X>
            X.LINE := ','
            X.LINE := XL.ARR3<1,X>
            X.LINE := EOL
            CALL S.WRITE.SEQ(X.LINE,2,X.ERR,X.MSG)
            IF (X.ERR) THEN LOGS<-1> = XLISTS.FILE.NAME:' Write ':X.MSG
         NEXT X

         IF (X.LOG.LEVEL EQ 'ALL') THEN
            LOGS<-1> = DOTS
         END
         CALL S.WRITE.EOF(2,X.ERR,X.MSG)
         IF (X.ERR) THEN LOGS<-1> = XLISTS.FILE.NAME:' Write ':X.MSG
         CALL S.CLOSE.SEQ(2,X.ERR,X.MSG)
         IF (X.ERR) THEN LOGS<-1> = XLISTS.FILE.NAME:' Close ':X.MSG
         SQL = "CLS":@FM:"COPY FROM HOLD TO ":X.WORK.DIR:" ":XLISTS.FILE.NAME
         CALL S.EXECUTE('-C ':SQL)
         IF (X.LOG.LEVEL EQ 'ALL') THEN
            LOGS<-1> = '' ; LOGS<-1> = LOWER(SQL) ; LOGS<-1> = CAPTURED.OUTPUT
         END
      END ELSE
         CALL S.CLOSE.SEQ(2,X.ERR,X.MSG)
         IF (X.ERR) THEN LOGS<-1> = XLISTS.FILE.NAME:' Close ':X.MSG
         IF (X.LOG.LEVEL EQ 'ALL') THEN
            LOGS<-1> = 'No xlists found.'
         END
      END

RETURN

****************************************
UPLOAD.TO.CANVAS:
****************************************

   * final processing
   LOGS<-1> = ''
   LOGS<-1> = 'Program: ':PGNAME
   LOGS<-1> = 'Term(s): ':TERMS.PLUS.ADDED
   LOGS<-1> = 'Total users uploaded: ':TOTAL.USERS
   LOGS<-1> = 'Total enrollments uploaded: ':TOTAL.ENROLLMENTS
   IF (MASTER.RUN EQ TRUE) THEN
```

```
      LOGS<-1> = 'Total teacher enrolls uploaded: ':TOTAL.ENROLLS
      LOGS<-1> = 'Total sections uploaded: ':TOTAL.SECTIONS
      LOGS<-1> = 'Total courses uploaded: ':TOTAL.COURSES
      LOGS<-1> = 'Total xlists uploaded: ':TOTAL.XLISTS
   END
   IF (USERS EQ TRUE OR ENROLLMENTS EQ TRUE OR ENROLLS EQ TRUE OR SECTIONS EQ TRUE OR COURSES EQ TRUE OR XLISTS EQ TRUE) THEN
   * points to note about this statement
   * 1) it runs Windows cmd first.  /c means execute a command and return from the shell.
* 2) echo . is required.  without it, powershell will flake out (or wil
* 3) cmd escape char is ^.  Without it cmd will freak out and pass a pa
   * 4) "" found in the script reduce to a single quote.
* 5) """ reduces to two quotes; the first to encompass the powershell a

      IF (X.LOG.LEVEL EQ 'ALL') THEN
         LOGS<-1> = 'Uploading file(s) to Canvas server.'
      END
      STMT = "!cmd /c "
      STMT := QUOTE:"echo . | powershell -ExecutionPolicy RemoteSigned C:\Users\Public\Documents\WindowsPowerShell\Upload-CanvasSIS.ps1 "
      STMT := "'canvasbatch' '<<put the password for the account to the left here>>' '
      STMT := "-CanvasURI '":X.CANVAS.URI:"' "
      STMT := "-CanvasAccountID '<<put your school's Account ID here>>' "
      STMT := "-CanvasAPIKey '<<put your school's API key here' "
      IF (MASTER.RUN EQ TRUE) THEN
         STMT := "-BatchMode -BatchModeTermID ":TRIM(TERMS.PLUS.ADDED,"/","A"):" "
      END
      STMT := "-Verbose -DeleteInputFiles -WorkingPath 'D:\ftp_folders\live\":X.WORK.DIR:"'":QUOTE
      IF (X.DEBUG EQ TRUE AND X.DEBUG.PS.SCRIPT EQ TRUE) THEN LOGS<-1> = STMT
      IF (UPLOAD.FILES EQ 'Y') THEN CALL S.EXECUTE('-C ':STMT)
      LOGS<-1> = "Powershell return code: ":CAPTURED.OUTPUT
   END
   * set the time of the last successful run
   V.PUC.CONTROL.POINT.ID = 'CANVAS.SETTINGS'
   FOR_THIS SINGLE PUC.CONTROL.POINT.ID
      V.XFIELD2 = "Last Run Date":@VM:DATE()
   END_THIS PUC.CONTROL.POINT.ID

RETURN

```

```
****************************************
WRITE.LOG:
****************************************

* write log file out
V.PUC.CANVAS.LOG.ID = CANVAS.LOG.CTR
FOR_THIS SINGLE PUC.CANVAS.LOG.ID
   V.XPCL.DATE = DATE()
   V.XPCL.TIME = TIME()
   CONVERT @FM TO @VM IN LOGS
   SWAP CHAR(192):"21":CHAR(192):"23":CHAR(192) WITH '' IN LOGS
   VL.XPCL.LOG = LOGS
END_THIS PUC.CANVAS.LOG.ID

RETURN

****************************************
WRITE.LOG.COUNTER:
****************************************


*uncomment these 4 lines if you want to write the log info to a sequent

*CALL S.OPEN.SEQ('HOLD','SDP',2,'W',X.ERR,X.MSG)
*CALL S.WRITE.SEQ(LOGS,2,X.ERR,X.MSG)
*CALL S.WRITE.EOF(2,X.ERR,X.MSG)
*CALL S.CLOSE.SEQ(2,X.ERR,X.MSG)

* write log counter out
V.PUC.CONTROL.POINT.ID = 'PUC.CANVAS.LOG.CTR'
FOR_THIS SINGLE PUC.CONTROL.POINT.ID
   V.XFIELD1 = CANVAS.LOG.CTR
END_THIS PUC.CONTROL.POINT.ID

RETURN


****************************************
XDEBUG.001:
****************************************
IF (X.DEBUG EQ TRUE) THEN
   MASTER.RUN = TRUE
   X.WORK.DIR := '.DEV'
   X.LOG.LEVEL = 'ALL'
   X.CANVAS.URI = 'https://puc.beta.instructure.com'
END
RETURN

****************************************
XDEBUG.002:
****************************************
RETURN
IF (X.DEBUG EQ TRUE) THEN
   XT = TERMS:ADDED.TERMS
   XT = CONVERT("'/","",XT):"_"
   IF (INDEX(UFILE.NAME,"_",1) GT 0) THEN
      UFILE.NAME = XT:FIELD(UFILE.NAME,"_",2,1)
      EFILE.NAME = XT:FIELD(EFILE.NAME,"_",2,1)
      ENROLL.FILE.NAME = XT:FIELD(ENROLL.FILE.NAME,"_",2,1)
      XLISTS.FILE.NAME = XT:FIELD(XLISTS.FILE.NAME,"_",2,1)
      SECTIONS.FILE.NAME = XT:FIELD(SECTIONS.FILE.NAME,"_",2,1)
      COURSES.FILE.NAME = XT:FIELD(COURSES.FILE.NAME,"_",2,1)
   END ELSE
      UFILE.NAME = XT:UFILE.NAME
      EFILE.NAME = XT:EFILE.NAME
      ENROLL.FILE.NAME = XT:ENROLL.FILE.NAME
      XLISTS.FILE.NAME = XT:XLISTS.FILE.NAME
      SECTIONS.FILE.NAME = XT:SECTIONS.FILE.NAME
      COURSES.FILE.NAME = XT:COURSES.FILE.NAME
   END
END
RETURN
```