

Th
res
of
thi
ref
ite
is
no
res

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.STATUSSES
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.STATUSSES
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

```

1  EOL = ''
2  TRUE = 1
3  FALSE = 0
4  LOGS = ''
5  QUOTE = '''
6  PGNAME = 'XDPB32'
7  X.FILE.CREATED = FALSE
8  X.WORK.DIR = 'PUC.CANVAS.UPLOAD.DIR'
9  X.LOG.LEVEL = 'ERROR'
10 X.CANVAS.URI = 'https://puc.instructure.com'
11
12 UFILE.NAME = 'users.csv'
13 UHEADING =
' "user_id","login_id","password","first_name","last_name","email","status":@FM
14
* "0253019","rhiebert","","Ryan","Hiebert","rhiebert@puc.edu","active"
15
16 EFILE.NAME = 'enrollments.csv'
17 EHEADING = 'course_id,user_id,role,section_id,status':@FM
18 * ,0253019,student,NURS-255-01-11SP,active
19
20 SWAP @FM WITH EOL IN UHEADING
21 SWAP @FM WITH EOL IN EHEADING
22
23 ENROLL.FILE.NAME = 'teachers.csv'
24 ENROLL.HEADING = 'course_id,user_id,role,section_id,status':@FM
25 * ,0000087,teacher,ESTH-497-01-11SU,active
26
27 XLISTS.FILE.NAME = 'xlists.csv'
28 XLISTS.HEADING = 'xlist_course_id,section_id,status':@FM
29 * HIST-360-01-11SU,RELH-360-01-11SU,active
30
31 SECTIONS.FILE.NAME = 'sections.csv'
32 SECTIONS.HEADING =
' "section_id","course_id","name","status","start_date","end_date":@FM
33
* "AVIA-176-01-11SU","AVIA-176-01-11SU","AVIA-176-01 11SU","active","","'
34
35 COURSES.FILE.NAME = 'courses.csv'
36 COURSES.HEADING =
' "course_id","short_name","long_name","account_id","term_id","status":@FM
37

```

```
* "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
```

```

35         ENROLLS = FALSE
36         SECTIONS = FALSE
37         COURSES = FALSE
38         XLISTS = FALSE
39         TERMS = "':FIELD(XL.TERM.LIST,@VM,X.LOOP,1)':'"
40         ADDED.TERMS = "''"
41         IF (X.LOOP GT X.MAIN.TERM.CNT) THEN
42             ADDED.TERMS = TERMS
43             TERMS = "''"
44         END
45         TERMS.PLUS.ADDED = TERMS:ADDED.TERMS
46         SWAP "''" WITH "''" IN TERMS.PLUS.ADDED
47         GOSUB XDEBUG.002
48         GOSUB CREATE.USERS
49         GOSUB CREATE.ENROLLMENTS
50         GOSUB CREATE.TEACHERS
51         GOSUB CREATE.COURSES
52         GOSUB CREATE.SECTIONS
53         GOSUB CREATE.XLISTS
54         GOSUB UPLOAD.TO.CANVAS
55     NEXT X.LOOP
56 END ELSE
57     GOSUB CREATE.USERS
58     GOSUB CREATE.ENROLLMENTS
59     GOSUB UPLOAD.TO.CANVAS
60 END
61 END
62 GOSUB WRITE.LOG
63 GOSUB WRITE.LOG.COUNTER
64
65 RETURN
66
67 *****
68 GET.TERMS:
69 *****
70
71 * retrieve settings (TERMS)
72
73 * note TERMS and ADDED.TERMS are not @VM separated.  they are terms in a
74
75 * ADDED.TERMS is used to only upload courses and teachers, but not enrollments
76
74 V.PUC.CONTROL.POINT.ID = 'CANVAS.SETTINGS'
75 FOR_THIS SINGLE REFERENCED PUC.CONTROL.POINT.ID
76     TERMS = FIELD(V.XFIELD1,@VM,2,1)
77     LAST.RUN.DATE = FIELD(V.XFIELD2,@VM,2,1)
78     ADDED.TERMS = FIELD(V.XFIELD3,@VM,2,1)
79     UPLOAD.FILES = FIELD(V.XFIELD4,@VM,2,1)
80 END_THIS PUC.CONTROL.POINT.ID
81 MASTER.RUN = FALSE
82 IF (LAST.RUN.DATE NE DATE()) THEN MASTER.RUN = TRUE
83
84 IF (X.LOG.LEVEL EQ 'ALL') THEN

```

```

85     LOGS<-1> = 'TERMS: ':TERMS
86     LOGS<-1> = 'LAST.RUN.DATE: ':LAST.RUN.DATE
87     LOGS<-1> = 'ADDED.TERMS: ':ADDED.TERMS
88     LOGS<-1> = 'UPLOAD.FILES: ':UPLOAD.FILES
89     LOGS<-1> = 'MASTER.RUN: ':MASTER.RUN
90 END
91
92 RETURN
93
94 *****
95 GET.LOG.COUNTER:
96 *****
97
98 * get log counter and increment it in case we write logs out
99 V.PUC.CONTROL.POINT.ID = 'PUC.CANVAS.LOG.CTR'
100 FOR_THIS SINGLE REFERENCED PUC.CONTROL.POINT.ID
101     CANVAS.LOG.CTR = V.XFIELD1+1
102 END_THIS PUC.CONTROL.POINT.ID
103
104 RETURN
105
106 *****
107 CHECK.CSV.EXIST:
108 *****
109
110
111 * check for existence of csv files in Upload directory.  If there are ar
112
113 * files, abort all further processing as the previous process did not ex
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128 *****
129 CREATE.USERS:
130 *****
131
132     X.ERR = ''

```

```

133     X.MSG = ''
134     TOTAL.USERS = 0
135     TOTAL.ENROLLMENTS = 0
136     TOTAL.ENROLLS = 0
137     TOTAL.SECTIONS = 0
138     TOTAL.COURSES = 0
139     TOTAL.XLISTS = 0
140
141
142 * get list of new students to add to canvas, and also of all teachers/
faculty teaching this term
142
143 * also get list of all employees and add them as well.  these will have
143
144 * should they be added manually to a course, their records will already
144
145     IF (X.LOG.LEVEL EQ 'ALL') THEN
146         LOGS<-1> =
147         '***** Creating users.csv (teachers, staff and students)'
148         END
149         SQLP = "CLS":@FM:"SELECT STUDENT.TERMS "
150         SQL = "WITH STTR.TERM EQ ":TERMS:
151         " AND STTR.CURRENT.STATUS EQ 'R'T'X'P'W' "
152
153 * if not running a master list then include the uploaded already flag in
153
154     IF (MASTER.RUN EQ FALSE) THEN SQL :=
155     "AND XSTTR.XPCST.STATUS NE 'A' "
156     SQL :=
157     "AND STTR.STUDENT.SORT.LAST.NAME UNLIKE '*...' BY STTR.STUDENT.SORT.NAME"
158
159     CALL S.EXECUTE('-C ':SQLP:SQL)
160     IF (X.LOG.LEVEL EQ 'ALL') THEN
161         LOGS<-1> = LOWER(SQLP):SQL ; LOGS<-1> = CAPTURED.OUTPUT
162     END
163     X.USERS.CNT = @SYSTEM.RETURN.CODE
164     IF (X.USERS.CNT LT 0) THEN X.USERS.CNT = 0
165
166
167 * check teachers now and use the additional terms.  we only do teachers
167
168 * but we run them first because if a teacher is also a student, we want
168
169 * and skip them as a student.
170     IF (MASTER.RUN EQ TRUE) THEN
171         SQLP2 = "CLS":@FM:"SELECT COURSE.SEC.FACULTY "
172         SQL2 = "WITH SEC.TERM EQ ":TERMS.PLUS.ADDED:" "
173         SQL2 := "BY CSF.FACULTY.LAST.NAME"
174         CALL S.EXECUTE('-C ':SQLP2:SQL2)
175         IF (X.LOG.LEVEL EQ 'ALL') THEN
176             LOGS<-1> = LOWER(SQLP2):SQL2 ; LOGS<-1> =
CAPTURED.OUTPUT

```



```

169         END
170         IF (@SYSTEM.RETURN.CODE GT 0) THEN
171             X.USERS.CNT += @SYSTEM.RETURN.CODE
172         END
173     END
174
175     IF (X.USERS.CNT GT 0) THEN
176         USERS = TRUE
177         XL.USER.LIST = ''
178         * open hold file for users
179         CALL S.OPEN.SEQ('HOLD',UFILE.NAME,8,'W',X.ERR,X.MSG)
180         IF (X.ERR) THEN LOGS<-1> = UFILE.NAME:' Open ':X.MSG
181
182         * write column headings on users file
183         CALL S.WRITE.SEQ(UHEADING,8,X.ERR,X.MSG)
184         IF (X.ERR) THEN LOGS<-1> = UFILE.NAME:' Write ':X.MSG
185
186         * get list of all teachers and add them to the file
187         IF (MASTER.RUN EQ TRUE) THEN
188             FOR_EACH REFERENCED SELECTED COURSE.SEC.FACULTY.ID USING
NEWLIST CRITERIA SQL2
189                 IF (V.CSF.FACULTY # '0156474') THEN
190                     FIND V.CSF.FACULTY IN XL.USER.LIST SETTING F,V,S
ELSE
191                     * teacher not found, so add them to the list
192                     XL.USER.LIST<-1> = V.CSF.FACULTY
193
194 * check that puc email type is actually a puc.edu email.
195 X.EMAIL = ''
196 SKIP.USER = FALSE
197 CALL XS.GET.PUC.EMAIL(X.EMAIL,V.CSF.FACULTY)
198 IF (X.EMAIL # '' AND X.EMAIL[8] NE '@puc.edu')
THEN
199     X.MSG = PGNAME:
' Error: user has invalid PUC email: ':V.CSF.FACULTY
200     CALL S.POPMAIL('SEND','canvas@puc.edu',X.MSG
,X.MSG)
201     LOGS<-1> = X.MSG
202     SKIP.USER = TRUE
203     END
204     IF (SKIP.USER EQ FALSE) THEN
205         X.EMAIL.NAME = ''
206         CALL XS.GET.PUC.EMAIL.NAME(X.EMAIL.NAME,
V.CSF.FACULTY)
207         IF (X.EMAIL.NAME EQ '') THEN
208             * create account here
209             SLEEP 1
210             MSG1 = ''
211             LOG = ''
212             CALL XS.CREATE.ACTIVATE.ACCOUNT(
V.CSF.FACULTY,X.EMAIL.NAME,MSG1,LOG)
213             LOGS<-1> = "Created: ":X.EMAIL.NAME:" - "
:MSG1
214         END

```

```

214             X.NAME = ''
215             FOR_THIS REFERENCED SECONDARY CSF.FACULTY
216
217 * get the preferred name in last name, first order
218             X.FNAME = FIELD(V.FAC.USER3,"",2,1)
219             X.LNAME = FIELD(V.FAC.USER3,"",1,1)
220             CONVERT " " TO " " IN X.FNAME
221             CONVERT " " TO " " IN X.LNAME
222             END_THIS CSF.FACULTY
223             X.LINE = QUOTE:V.CSF.FACULTY:QUOTE:','
224             X.LINE := QUOTE:X.EMAIL.NAME:QUOTE:','
225             X.LINE := QUOTE:QUOTE:','
226             X.LINE := QUOTE:X.FNAME:QUOTE:','
227             X.LINE := QUOTE:X.LNAME:QUOTE:','
228             X.LINE := QUOTE:X.EMAIL.NAME:"@puc.edu":
229 QUOTE:','
230             X.LINE := QUOTE:'active':QUOTE
231             X.LINE := EOL
232             CALL S.WRITE.SEQ(X.LINE,8,X.ERR,X.MSG)
233             IF (X.ERR) THEN LOGS<-1> = UFILE.NAME:
234 ' Write ':X.MSG
235             TOTAL.USERS += 1
236         END
237     END
238 END
239     END_EACH COURSE.SEC.FACULTY.ID
240 END
241
242 * get list of all staff and add them to the file
243 IF (MASTER.RUN EQ TRUE) THEN
244     SQL3 = "CLS":@FM:
245 'SELECT HRPER WITH XHRP.STAFF.EMPLOYED EQ 1 BY XHRP.NAME'
246     CALL S.EXECUTE('-C ':SQL3)
247     IF (X.LOG.LEVEL EQ 'ALL') THEN
248         LOGS<-1> = LOWER(SQL3) ; LOGS<-1> = CAPTURED.OUTPUT
249     END
250     IF (@SYSTEM.RETURN.CODE GT 0) THEN
251         X.STATUS = 0
252         XL.LIST = ''
253         CALL S.READLIST(XL.LIST,0,X.STATUS)
254         IF (X.STATUS EQ 1 AND DCOUNT(XL.LIST,@FM) GT 0) THEN
255             COL = 1
256             LOOP
257                 REMOVE V.ID FROM XL.LIST AT COL SETTING
258             END.MARK
259             FOR_THIS SINGLE REFERENCED ID
260                 X.ID = V.ID
261                 FIND X.ID IN XL.USER.LIST SETTING F,V,S ELSE
262
263 * staff not found, so add them to the list
264             XL.USER.LIST<-1> = X.ID
265
266 * check that puc email type is actually a puc.edu email.
267             X.EMAIL = ''

```

```

261             SKIP.USER = FALSE
262             CALL XS.GET.PUC.EMAIL(X.EMAIL,X.ID)
263             IF (X.EMAIL # '' AND X.EMAIL[8] NE
'@puc.edu') THEN
264                 X.MSG = PGNAME:
' Error: user has invalid PUC email: ':X.ID
265                 CALL S.POPMAIL('SEND','canvas@puc.edu'
,X.MSG,X.MSG)
266                 LOGS<-1> = X.MSG
267                 SKIP.USER = TRUE
268             END
269             IF (SKIP.USER EQ FALSE) THEN
270                 X.EMAIL.NAME = ''
271                 CALL XS.GET.PUC.EMAIL.NAME(
X.EMAIL.NAME,X.ID)
272                 IF (X.EMAIL.NAME EQ '') THEN
273                     * create account here
274                     SLEEP 1
275                     MSG1 = ''
276                     LOG = ''
277                     CALL XS.CREATE.ACTIVATE.ACCOUNT(
X.ID,X.EMAIL.NAME,MSG1,LOG)
278                     LOGS<-1> = "Created: ":X.EMAIL.NAME
: " - ":MSG1
279                     END
280                     X.NAME = ''
281                     CALL XS.GET.NAME(X.NAME,X.ID,"LF")
282                     X.FNAME = FIELD(X.NAME,"",2,1)
283                     X.LNAME = FIELD(X.NAME,"",1,1)
284                     CONVERT " " TO " " IN X.FNAME
285                     CONVERT " " TO " " IN X.LNAME
286                     X.LINE = QUOTE:X.ID:QUOTE:','
287                     X.LINE := QUOTE:X.EMAIL.NAME:QUOTE:','
288                     X.LINE := QUOTE:QUOTE:','
289                     X.LINE := QUOTE:X.FNAME:QUOTE:','
290                     X.LINE := QUOTE:X.LNAME:QUOTE:','
291                     X.LINE := QUOTE:X.EMAIL.NAME:
"@puc.edu":QUOTE:','
292                     X.LINE := QUOTE:'active':QUOTE
293                     X.LINE := EOL
294                     CALL S.WRITE.SEQ(X.LINE,8,X.ERR,X.MSG)
295                     IF (X.ERR) THEN LOGS<-1> = UFILE.NAME:
' Write ':X.MSG
296                     TOTAL.USERS += 1
297                 END
298             END
299             END_THIS ID
300             WHILE END.MARK DO REPEAT
301         END
302     END
303 END
304
305
306             IF (TERMS # '') THEN

```

```

307          FOR_EACH REFERENCED SELECTED STTR.STUDENT USING NEWLIST
CRITERIA SQL
308
* this is necessary in case a person is actively registered in more than
309
* we don't want to add them twice as they could have been added in a previous
310          V.PUC.CANVAS.STU.TERMS.ID = V.STTR.STUDENT:"*":
V.STTR.TERM:"*":V.STTR.ACAD.LEVEL
311          FIND V.STTR.STUDENT IN XL.USER.LIST SETTING F,V,S
ELSE
312
* student not found, so add them to the list and process away
313          XL.USER.LIST<-1> = V.STTR.STUDENT
314
* check that puc email type is actually a puc.edu email.
315          X.EMAIL = ''
316          SKIP.USER = FALSE
317          CALL XS.GET.PUC.EMAIL(X.EMAIL,V.STTR.STUDENT)
318          IF (X.EMAIL # '' AND X.EMAIL[8] NE '@puc.edu')
THEN
319          X.MSG = PNAME:
' Error: user has invalid PUC email: ':V.STTR.STUDENT
320          CALL S.POPMAIL('SEND','canvas@puc.edu',X.MSG,
X.MSG)
321          LOGS<-1> = X.MSG
322          SKIP.USER = TRUE
323          END
324          IF (SKIP.USER EQ FALSE) THEN
325          X.EMAIL.NAME = ''
326          CALL XS.GET.PUC.EMAIL.NAME(X.EMAIL.NAME,
V.STTR.STUDENT)
327          IF (X.EMAIL.NAME EQ '') THEN
328          * create account here
329
330          MSG1 = ''
331          LOG = ''
332          CALL XS.CREATE.ACTIVATE.ACCOUNT(
V.STTR.STUDENT,X.EMAIL.NAME,MSG1,LOG)
333          LOGS<-1> = "Created: ":X.EMAIL.NAME:" - ":
MSG1
334          END
335          X.NAME = ''
336          CALL XS.GET.NAME(X.NAME,V.STTR.STUDENT,"LF")
337          X.FNAME = FIELD(X.NAME,"",2,1)
338          X.LNAME = FIELD(X.NAME,"",1,1)
339          CONVERT " " TO " " IN X.FNAME
340          CONVERT " " TO " " IN X.LNAME
341          X.LINE = QUOTE:V.STTR.STUDENT:QUOTE:','
342          X.LINE := QUOTE:X.EMAIL.NAME:QUOTE:','
343          X.LINE := QUOTE:QUOTE:','
344          X.LINE := QUOTE:X.FNAME:QUOTE:','
345          X.LINE := QUOTE:X.LNAME:QUOTE:','

```

```

346             X.LINE := QUOTE:X.EMAIL.NAME:"@puc.edu":QUOTE:
', '
347             X.LINE := QUOTE:'active':QUOTE
348             X.LINE := EOL
349             CALL S.WRITE.SEQ(X.LINE,8,X.ERR,X.MSG)
350             IF (X.ERR) THEN LOGS<-1> = UFILE.NAME:' Write '
:X.MSG
351
* write data to the flag table. A means added. It is the only code use

352             IF (X.DEBUG EQ FALSE) THEN
353                 FOR_THIS SINGLE PUC.CANVAS.STU.TERMS.ID
354                     V.XPCST.STATUS = "A"
355                     V.XPCST.DATE = DATE()
356                     V.XPCST.TIME = TIME()
357                 END_THIS PUC.CANVAS.STU.TERMS.ID
358             END
359             TOTAL.USERS += 1
360         END
361     END
362     END_EACH STTR.STUDENT
363 END
364
365     * close hold files
366     CALL S.WRITE.EOF(8,X.ERR,X.MSG)
367     IF (X.ERR) THEN LOGS<-1> = UFILE.NAME:' Write ':X.MSG
368     CALL S.CLOSE.SEQ(8,X.ERR,X.MSG)
369     IF (X.ERR) THEN LOGS<-1> = UFILE.NAME:' Close ':X.MSG
370     SQL = "CLS":@FM:"COPY FROM HOLD TO ":X.WORK.DIR:" ":
UFILE.NAME
371     CALL S.EXECUTE('-C ':SQL)
372     IF (X.LOG.LEVEL EQ 'ALL') THEN
373         LOGS<-1> = '' ; LOGS<-1> = LOWER(SQL) ; LOGS<-1> =
CAPTURED.OUTPUT
374     END
375     * end of processing students and teachers for the terms given
376 END
377
378 RETURN
379
380 *****
381 CREATE.ENROLLMENTS:
382 *****
383
384     * process student enrollment changes only
385     IF (X.LOG.LEVEL EQ 'ALL') THEN
386         LOGS<-1> = "***** Creating enrollments.csv."
387         LOGS<-1> =
"Each dot represents one enrollment record uploaded."
388     END
389     DOTS = ''
390     FILE.OPEN.YET = FALSE
391     SQL = "WITH STTR.TERM EQ ":TERMS:
" AND STTR.CURRENT.STATUS EQ 'R'T'X'P'W' "

```

```

392
* only deal with students who have been uploaded (their status flag set)

393     SQL := "AND XSTTR.XPCST.STATUS EQ 'A' "
394     SQL :=
"AND STTR.STUDENT.SORT.LAST.NAME UNLIKE '*...' BY STTR.STUDENT.SORT.LAST"

395     IF (TERMS # "'') THEN
396         FOR_EACH REFERENCED SELECTED STTR.STUDENT USING NEWLIST
CRITERIA SQL
397
*IF (X.DEBUG EQ TRUE) THEN CRT '-----':V.STTR.STUDENT:'-----':VI

398         XL.STUD.ACAD.REC = VL.STTR.STUDENT.ACAD.CRED
399         COL = 1
400         LOOP
401             REMOVE V.STUDENT.ACAD.CRED.ID FROM XL.STUD.ACAD.REC
AT COL SETTING END.MARK
402             FOR_THIS SINGLE REFERENCED STUDENT.ACAD.CRED.ID
403
*IF (X.DEBUG EQ TRUE) THEN CRT V.STUDENT.ACAD.CRED.ID
404             * evaluate individual enrollment
405             * we will add it to the file if:
406             * XPCSAC.STATUS is blank or
407
* STC.STATUS<1,1> is 'N' 'A' and XPCSAC.STATUS is not 'A'
408
* STC.STATUS<1,1> is anything else and XPCSAC.STATUS is 'A' ('D' is the

409             * MASTER.RUN is true
410             V.PUC.CANVAS.STU.ACAD.CRED.ID =
V.STUDENT.ACAD.CRED.ID
411             FOR_THIS SINGLE PUC.CANVAS.STU.ACAD.CRED.ID
412             ADD.REC = FALSE
413             IF (V.XPCSAC.STATUS EQ '' OR MASTER.RUN EQ TRUE
) THEN
414                 ADD.REC = TRUE
415             END ELSE
416                 X.STC.STATUS = FIELD(VL.STC.STATUS,@VM,1,1)
417                 IF (X.STC.STATUS EQ 'N' OR X.STC.STATUS EQ
'A') THEN
418                     IF (V.XPCSAC.STATUS NE 'A') THEN ADD.REC
= TRUE
419                     END ELSE
420                     IF (V.XPCSAC.STATUS EQ 'A') THEN ADD.REC
= TRUE
421                     END
422                 END
423
* get the course and section numbers (these can be gotten from the student)

424
* but may be wrong, so this is more accurate hopefully)
425         FOR_THIS SINGLE SECONDARY

```

```

STC.STUDENT.COURSE.SEC
426          FOR_THIS SINGLE REFERENCED SECONDARY
SCS.COURSE.SECTION
427          X.COURSE.SEC = V.SEC.SUBJECT:'-':
V.SEC.COURSE.NO:'-':V.SEC.NO
428          END_THIS SCS.COURSE.SECTION
429          END_THIS STC.STUDENT.COURSE.SEC
430
* debug for me only just to see how accurate the data is
431          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
432
*MSG1 = PGNAME:" Error: Student ":V.STTR.STUDENT:" Course Section may no
433
*CALL S.POPMAIL('SEND','lantern@puc.edu',MSG1,MSG1)
434          END
435
* checking stc.course.name because there has been a case of a blank
436          * course/
section in the enrollments file. this should stop that from happening.
437          IF (ADD.REC EQ TRUE AND V.STC.COURSE.NAME # '')
THEN
438          IF (FILE.OPEN.YET EQ FALSE) THEN
439          * open hold file for enrollments
440          CALL S.OPEN.SEQ('HOLD',EFILE.NAME,2,'W',
X.ERR,X.MSG)
441          IF (X.ERR) THEN LOGS<-1> = EFILE.NAME:
' Open ':X.MSG
442
* write column headings on enrollments file
443          CALL S.WRITE.SEQ(EHEADING,2,X.ERR,X.MSG)
444          IF (X.ERR) THEN LOGS<-1> = EFILE.NAME:
' Write ':X.MSG
445          ENROLLMENTS = TRUE
446          FILE.OPEN.YET = TRUE
447          END
448          X.LINE = ','
449          X.LINE := V.STTR.STUDENT
450          X.LINE := ','
451          X.LINE := 'student'
452          X.LINE := ','
453          X.TERM1 = V.STC.TERM
454          SWAP "/" WITH ' ' IN X.TERM1
455          X.LINE := X.COURSE.SEC:'-':X.TERM1
456          X.LINE := ','
457          X.STC.STATUS = FIELD(VL.STC.STATUS,@VM,1,1)
458          IF (X.STC.STATUS EQ 'N' OR X.STC.STATUS EQ
'A') THEN
459          X.LINE := 'active'
460
* write data to the student.acad.cred record. A means added, D means de

```

```

461                                IF (X.DEBUG EQ FALSE) THEN
V.XPCSAC.STATUS = "A"
462                                END ELSE
463                                X.LINE := 'deleted'
464                                IF (X.DEBUG EQ FALSE) THEN
V.XPCSAC.STATUS = "D"
465                                END
466                                X.LINE := EOL
467                                CALL S.WRITE.SEQ(X.LINE,2,X.ERR,X.MSG)
468                                IF (X.ERR) THEN LOGS<-1> = EFILE.NAME:
' Write ':X.MSG
469                                IF (X.DEBUG EQ FALSE) THEN V.XPCSAC.DATE =
DATE()
470                                IF (X.DEBUG EQ FALSE) THEN V.XPCSAC.TIME =
TIME()
471                                DOTS := '.'
472                                TOTAL.ENROLLMENTS += 1
473                                END
474                                END_THIS PUC.CANVAS.STU.ACAD.CRED.ID
475                                END_THIS STUDENT.ACAD.CRED.ID
476                                WHILE END.MARK DO REPEAT
477                                END_EACH STTR.STUDENT
478                                END
479                                IF (ENROLLMENTS EQ TRUE) THEN
480                                IF (X.LOG.LEVEL EQ 'ALL') THEN
481                                LOGS<-1> = DOTS
482                                END
483                                CALL S.WRITE.EOF(2,X.ERR,X.MSG)
484                                IF (X.ERR) THEN LOGS<-1> = EFILE.NAME:' Write ':X.MSG
485                                CALL S.CLOSE.SEQ(2,X.ERR,X.MSG)
486                                IF (X.ERR) THEN LOGS<-1> = EFILE.NAME:' Close ':X.MSG
487                                SQL = "CLS":@FM:"COPY FROM HOLD TO ":X.WORK.DIR:" ":
EFILE.NAME
488                                CALL S.EXECUTE('-C ':SQL)
489                                IF (X.LOG.LEVEL EQ 'ALL') THEN
490                                LOGS<-1> = '' ; LOGS<-1> = LOWER(SQL) ; LOGS<-1> =
CAPTURED.OUTPUT
491                                END
492                                END ELSE
493                                IF (FILE.OPEN.YET EQ TRUE) THEN
494                                CALL S.CLOSE.SEQ(2,X.ERR,X.MSG)
495                                IF (X.ERR) THEN LOGS<-1> = EFILE.NAME:' Close ':X.MSG
496                                END
497                                IF (X.LOG.LEVEL EQ 'ALL') THEN
498                                LOGS<-1> = 'No enrollment changes found.'
499                                END
500                                END
501                                RETURN
502
503                                *****
504                                CREATE.TEACHERS:
505                                *****
506
507                                * process teacher enroll records using additional terms

```



```

508         IF (X.LOG.LEVEL EQ 'ALL') THEN
509             LOGS<-1> = "***** Creating teachers.csv."
510             LOGS<-1> =
"Each dot represents one teacher record uploaded."
511         END
512         DOTS = ''
513         ENROLLS = FALSE
514         SQL = "CLS":@FM:"SELECT COURSE.SECTIONS WITH SEC.TERM EQ " :
TERMS.PLUS.ADDED:
" AND WITH EVERY SEC.DEPTS NE 'ND' SAVING UNIQUE SEC.FACULTY"
515         SQL<-1> =
"SELECT COURSE.SEC.FACULTY REQUIRE.SELECT SAVING UNIQUE CSF.FACULTY"
516         SQL<-1> =
"SELECT FACULTY WITH @ID UNLIKE '0156474' REQUIRE.SELECT"
517         CALL S.EXECUTE('-C ':SQL)
518         IF (X.LOG.LEVEL EQ 'ALL') THEN
519             LOGS<-1> = LOWER(SQL) ; LOGS<-1> = CAPTURED.OUTPUT
520         END
521         X.FAC.CNT = @SYSTEM.RETURN.CODE
522         CALL S.EXECUTE("CLS")
523         IF (X.FAC.CNT GT 0) THEN
524             * open hold file for enrolls
525             CALL S.OPEN.SEQ('HOLD',ENROLL.FILE.NAME,2,'W',X.ERR,
X.MSG)
526             IF (X.ERR) THEN LOGS<-1> = ENROLL.FILE.NAME:' Open ':
X.MSG
527             * write column headings on enrolls file
528             CALL S.WRITE.SEQ(ENROLL.HEADING,2,X.ERR,X.MSG)
529             IF (X.ERR) THEN LOGS<-1> = ENROLL.FILE.NAME:' Write ':
X.MSG
530             ENROLLS = TRUE
531             XL.SORT.ARRAY = ''
532             XL.ARR1 = ''
533             XL.ARR2 = ''
534             SQL = "WITH SEC.TERM EQ ":TERMS.PLUS.ADDED:
" AND WITH EVERY SEC.DEPTS NE 'ND' BY SEC.TERM BY SEC.FACULTY.SORT.NAME"

535         FOR_EACH REFERENCED SELECTED COURSE.SECTIONS.ID USING
NEWLIST CRITERIA SQL
536             X.COURSE.SEC = V.SEC.SUBJECT:'-':V.SEC.COURSE.NO:'-':
V.SEC.NO
537             FOR_EACH REFERENCED SECONDARY SEC.FACULTY
538                 IF (V.CSF.FACULTY NE '0156474' AND V.CSF.FACULTY
NE '') THEN
539                 X.TERM1 = V.SEC.TERM
540                 SWAP "/" WITH '' IN X.TERM1
541                 XL.SORT.ARRAY<1,-1> = V.CSF.FACULTY:X.TERM1:
X.COURSE.SEC
542                 XL.ARR1<1,-1> = V.CSF.FACULTY
543                 XL.ARR2<1,-1> = X.COURSE.SEC:'-':X.TERM1
544                 DOTS := ' .'
545                 TOTAL.ENROLLS += 1
546             END
547         END_EACH SEC.FACULTY

```

```

548         END_EACH COURSE.SECTIONS.ID
549     END
550     IF (ENROLLS EQ TRUE) THEN
551         CALL S.SORT.ASSOC.AL(XL.SORT.ARRAY,XL.ARR1,XL.ARR2,"",""
,","","","","")
552         * Write out the sorted lines
553         FOR X = 1 TO TOTAL.ENROLLS
554             X.LINE = ','
555             X.LINE := XL.ARR1<1,X>
556             X.LINE := ','
557             X.LINE := 'teacher'
558             X.LINE := ','
559             X.LINE := XL.ARR2<1,X>
560             X.LINE := ',active'
561             X.LINE := EOL
562             CALL S.WRITE.SEQ(X.LINE,2,X.ERR,X.MSG)
563             IF (X.ERR) THEN LOGS<-1> = ENROLL.FILE.NAME:' Write '
:X.MSG
564         NEXT X
565
566         IF (X.LOG.LEVEL EQ 'ALL') THEN
567             LOGS<-1> = DOTS
568         END
569         CALL S.WRITE.EOF(2,X.ERR,X.MSG)
570         IF (X.ERR) THEN LOGS<-1> = ENROLL.FILE.NAME:' Write ':
X.MSG
571         CALL S.CLOSE.SEQ(2,X.ERR,X.MSG)
572         IF (X.ERR) THEN LOGS<-1> = ENROLL.FILE.NAME:' Close ':
X.MSG
573         SQL = "CLS":@FM:"COPY FROM HOLD TO ":X.WORK.DIR:" ":
ENROLL.FILE.NAME
574         CALL S.EXECUTE('-C ':SQL)
575         IF (X.LOG.LEVEL EQ 'ALL') THEN
576             LOGS<-1> = ' ' ; LOGS<-1> = LOWER(SQL) ; LOGS<-1> =
CAPTURED.OUTPUT
577         END
578     END ELSE
579         CALL S.CLOSE.SEQ(2,X.ERR,X.MSG)
580         IF (X.ERR) THEN LOGS<-1> = ENROLL.FILE.NAME:' Close ':
X.MSG
581         IF (X.LOG.LEVEL EQ 'ALL') THEN
582             LOGS<-1> = 'No teacher enrolls found.'
583         END
584     END
585
586 RETURN
587
588 *****
589 CREATE.COURSES:
590 *****
591
592 * process courses
593 IF (X.LOG.LEVEL EQ 'ALL') THEN
594     LOGS<-1> = "***** Creating courses.csv."

```

```

595         LOGS<-1> =
"Each dot represents one course record uploaded."
596     END
597     DOTS = ''
598     SQLP = "CLS":@FM:"SELECT COURSE.SECTIONS "
599     SQL = "WITH SEC.TERM EQ ":TERMS.PLUS.ADDED:
" AND WITH EVERY SEC.DEPTS NE 'ND' BY SEC.TERM BY SEC.SUBJECT BY SEC.COU

600     CALL S.EXECUTE('-C ':SQLP:SQL)
601     IF (X.LOG.LEVEL EQ 'ALL') THEN
602         LOGS<-1> = LOWER(SQL) ; LOGS<-1> = CAPTURED.OUTPUT
603     END
604     X.CRS.CNT = @SYSTEM.RETURN.CODE
605     CALL S.EXECUTE("CLS")
606     IF (X.CRS.CNT GT 0) THEN
607         * open hold file for enrolls
608         CALL S.OPEN.SEQ('HOLD',COURSES.FILE.NAME,2,'W',X.ERR,
X.MSG)
609         IF (X.ERR) THEN LOGS<-1> = COURSES.FILE.NAME:' Open ':
X.MSG
610         * write column headings on enrolls file
611         CALL S.WRITE.SEQ(COURSES.Heading,2,X.ERR,X.MSG)
612         IF (X.ERR) THEN LOGS<-1> = COURSES.FILE.NAME:' Write ':
X.MSG
613         COURSES = TRUE
614         FOR_EACH REFERENCED SELECTED COURSE.SECTIONS.ID USING
NEWLIST CRITERIA SQL
615             X.COURSE.SEC = V.SEC.SUBJECT:'-':V.SEC.COURSE.NO:'-':
V.SEC.NO
616             X.TERM1 = V.SEC.TERM
617             SWAP "/" WITH ' ' IN X.TERM1
618             V.COURSE.SEC.FACULTY.ID = FIELD(VL.SEC.FACULTY,@VM,1,
1)
619             FOR_THIS SINGLE REFERENCED COURSE.SEC.FACULTY.ID
620                 V.ID = V.CSF.FACULTY
621                 FOR_THIS SINGLE REFERENCED ID
622                     END_THIS ID
623                 END_THIS COURSE.SEC.FACULTY.ID
624                 X.LINE = QUOTE:X.COURSE.SEC:'-':X.TERM1:QUOTE
625                 X.LINE := ', '
626                 X.LINE := QUOTE:X.COURSE.SEC:' ' :X.TERM1:QUOTE
627                 X.LINE := ', '
628                 IF (V.LAST.NAME EQ ' ') THEN
629                     X.LAST.NAME = 'Staff'
630                 END ELSE
631                     X.LAST.NAME = V.LAST.NAME
632                 END
633                 X.LINE := QUOTE:X.COURSE.SEC:' ':TRIM(
V.SEC.SHORT.TITLE):' (' :X.LAST.NAME:' ':X.TERM1:')':QUOTE
634                 X.LINE := ', '
635                 X.LINE := QUOTE:FIELD(VL.SEC.DEPTS,@VM,1,1):QUOTE
636                 X.LINE := ', '
637                 X.LINE := QUOTE:X.TERM1:QUOTE
638                 X.LINE := ', '

```

```

639         * check status
640         IF (FIELD(VL.SEC.STATUS,@VM,1,1) EQ 'A') THEN
641             X.LINE := QUOTE:'active':QUOTE
642         END ELSE
643             X.LINE := QUOTE:'deleted':QUOTE
644         END
645         X.LINE := EOL
646         CALL S.WRITE.SEQ(X.LINE,2,X.ERR,X.MSG)
647         IF (X.ERR) THEN LOGS<-1> = COURSES.FILE.NAME:
' Write ':X.MSG
648             DOTS := '.'
649             TOTAL.COURSES += 1
650         END_EACH COURSE.SECTIONS.ID
651     END
652     IF (COURSES EQ TRUE) THEN
653         IF (X.LOG.LEVEL EQ 'ALL') THEN
654             LOGS<-1> = DOTS
655         END
656         CALL S.WRITE.EOF(2,X.ERR,X.MSG)
657         IF (X.ERR) THEN LOGS<-1> = COURSES.FILE.NAME:' Write ':
X.MSG
658         CALL S.CLOSE.SEQ(2,X.ERR,X.MSG)
659         IF (X.ERR) THEN LOGS<-1> = COURSES.FILE.NAME:' Close ':
X.MSG
660         SQL = "CLS":@FM:"COPY FROM HOLD TO ":X.WORK.DIR:" ":
COURSES.FILE.NAME
661         CALL S.EXECUTE('-C ':SQL)
662         IF (X.LOG.LEVEL EQ 'ALL') THEN
663             LOGS<-1> = '' ; LOGS<-1> = LOWER(SQL) ; LOGS<-1> =
CAPTURED.OUTPUT
664         END
665     END ELSE
666         CALL S.CLOSE.SEQ(2,X.ERR,X.MSG)
667         IF (X.ERR) THEN LOGS<-1> = COURSES.FILE.NAME:' Close ':
X.MSG
668         IF (X.LOG.LEVEL EQ 'ALL') THEN
669             LOGS<-1> = 'No courses found.'
670         END
671     END
672 END
673 RETURN
674
675 *****
676 CREATE.SECTIONS:
677 *****
678
679     * process sections
680     IF (X.LOG.LEVEL EQ 'ALL') THEN
681         LOGS<-1> = "***** Creating sections.csv."
682         LOGS<-1> =
"Each dot represents one section record uploaded."
683     END
684     DOTS = ''
685     SQLP = "CLS":@FM:"SELECT COURSE.SECTIONS "

```

```

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

687      CALL S.EXECUTE('-C ':SQLP:SQL)
688      IF (X.LOG.LEVEL EQ 'ALL') THEN
689          LOGS<-1> = LOWER(SQL) ; LOGS<-1> = CAPTURED.OUTPUT
690      END
691      X.CRS.CNT = @SYSTEM.RETURN.CODE
692      CALL S.EXECUTE("CLS")
693      IF (X.CRS.CNT GT 0) THEN
694          * open hold file for enrolls
695          CALL S.OPEN.SEQ('HOLD',SECTIONS.FILE.NAME,2,'W',X.ERR,
X.MSG)
696          IF (X.ERR) THEN LOGS<-1> = SECTIONS.FILE.NAME:' Open ':
X.MSG
697          * write column headings on enrolls file
698          CALL S.WRITE.SEQ(SECTIONS.HEADING,2,X.ERR,X.MSG)
699          IF (X.ERR) THEN LOGS<-1> = SECTIONS.FILE.NAME:' Write ':
X.MSG
700          SECTIONS = TRUE
701          FOR EACH REFERENCED SELECTED COURSE.SECTIONS.ID USING
NEWLIST CRITERIA SQL
702              X.COURSE.SEC = V.SEC.SUBJECT:'-':V.SEC.COURSE.NO:'-':
V.SEC.NO
703              X.TERM1 = V.SEC.TERM
704              SWAP "/" WITH ' ' IN X.TERM1
705              X.LINE = QUOTE:X.COURSE.SEC:'-':X.TERM1:QUOTE
706              X.LINE := ', '
707              X.LINE := QUOTE:X.COURSE.SEC:'-':X.TERM1:QUOTE
708              X.LINE := ', '
709              X.LINE := QUOTE:X.COURSE.SEC:' ':X.TERM1:QUOTE
710              X.LINE := ', '
711              * check status
712              IF (FIELD(VL.SEC.STATUS,@VM,1,1) EQ 'A') THEN
713                  X.LINE := QUOTE:'active':QUOTE
714              END ELSE
715                  X.LINE := QUOTE:'deleted':QUOTE
716              END
717              X.LINE := ', '
718              X.LINE := QUOTE:QUOTE
719              X.LINE := ', '
720              X.LINE := QUOTE:QUOTE
721              X.LINE := EOL
722              CALL S.WRITE.SEQ(X.LINE,2,X.ERR,X.MSG)
723              IF (X.ERR) THEN LOGS<-1> = SECTIONS.FILE.NAME:
' Write ':X.MSG
724              DOTS := ' .'
725              TOTAL.SECTIONS += 1
726          END EACH COURSE.SECTIONS.ID
727      END
728      IF (SECTIONS EQ TRUE) THEN
729          IF (X.LOG.LEVEL EQ 'ALL') THEN
730              LOGS<-1> = DOTS
731          END

```

```

732      CALL S.WRITE.EOF(2,X.ERR,X.MSG)
733      IF (X.ERR) THEN LOGS<-1> = SECTIONS.FILE.NAME:' Write ':
X.MSG
734      CALL S.CLOSE.SEQ(2,X.ERR,X.MSG)
735      IF (X.ERR) THEN LOGS<-1> = SECTIONS.FILE.NAME:' Close ':
X.MSG
736      SQL = "CLS":@FM:"COPY FROM HOLD TO ":X.WORK.DIR:" ":
SECTIONS.FILE.NAME
737      CALL S.EXECUTE('-C ':SQL)
738      IF (X.LOG.LEVEL EQ 'ALL') THEN
739          LOGS<-1> = '' ; LOGS<-1> = LOWER(SQL) ; LOGS<-1> =
CAPTURED.OUTPUT
740      END
741      END ELSE
742      CALL S.CLOSE.SEQ(2,X.ERR,X.MSG)
743      IF (X.ERR) THEN LOGS<-1> = SECTIONS.FILE.NAME:' Close ':
X.MSG
744      IF (X.LOG.LEVEL EQ 'ALL') THEN
745          LOGS<-1> = 'No sections found.'
746      END
747      END
748
749      RETURN
750
751      *****
752      CREATE.XLISTS:
753      *****
754
755      * process xlists
756      IF (X.LOG.LEVEL EQ 'ALL') THEN
757          LOGS<-1> = "***** Creating xlists.csv."
758          LOGS<-1> =
"Each dot represents one xlist record uploaded."
759      END
760      DOTS = ''
761      * open hold file for enrolls
762      CALL S.OPEN.SEQ('HOLD',XLISTS.FILE.NAME,2,'W',X.ERR,X.MSG)
763      IF (X.ERR) THEN LOGS<-1> = XLISTS.FILE.NAME:' Open ':X.MSG
764      * write column headings on enrolls file
765      CALL S.WRITE.SEQ(XLISTS.HEADING,2,X.ERR,X.MSG)
766      IF (X.ERR) THEN LOGS<-1> = XLISTS.FILE.NAME:' Write ':X.MSG
767      XLISTS = TRUE
768      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)

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

```

```

777         COL = 1
778         XL.SORT.ARRAY = ''
779         XL.ARR1 = ''
780         XL.ARR2 = ''
781         XL.ARR3 = ''
782     LOOP
783         REMOVE V.COURSE.SECTIONS.ID FROM XL.LIST AT COL
SETTING END.MARK
784         * Get the primary xlist section
785         FOR_THIS SINGLE REFERENCED COURSE.SECTIONS.ID
786             X.TERM1 = V.SEC.TERM
787             SWAP "/" WITH ' ' IN X.TERM1
788             X.COURSE.SEC.TERM = V.SEC.SUBJECT:'-':
V.SEC.COURSE.NO:'-':V.SEC.NO:'-':X.TERM1
789             STAT1 = FIELD(VL.SEC.STATUS,@VM,1,1)
790             MAIN.COURSE.SEC.ID = V.COURSE.SECTIONS.ID
791             FOR_THIS SINGLE REFERENCED SECONDARY SEC.XLIST
792                 FOR_EACH REFERENCED ASSOCIATED
CSXL.COURSE.SECTIONS INTO COURSE.SECTIONS.ID
793                     IF (MAIN.COURSE.SEC.ID #
V.COURSE.SECTIONS.ID) THEN
794                         X.TERM2 = V.SEC.TERM
795                         SWAP "/" WITH ' ' IN X.TERM2
796                         X.COURSE.SEC.TERM2 = V.SEC.SUBJECT:'-':
V.SEC.COURSE.NO:'-':V.SEC.NO:'-':X.TERM2
797                         STAT2 = FIELD(VL.SEC.STATUS,@VM,1,1)
798                         XL.SORT.ARRAY<1,-1> = X.TERM1:
X.COURSE.SEC.TERM:X.COURSE.SEC.TERM2
799                         XL.ARR1<1,-1> = X.COURSE.SEC.TERM
800                         XL.ARR2<1,-1> = X.COURSE.SEC.TERM2
801                         IF (STAT1 EQ 'A' AND STAT2 EQ 'A') THEN
802                             XL.ARR3<1,-1> = 'active'
803                         END ELSE
804                             XL.ARR3<1,-1> = 'deleted'
805                         END
806                         DOTS := '.'
807                         TOTAL.XLISTS += 1
808                     END
809                 END_EACH CSXL.COURSE.SECTIONS
810             END_THIS SEC.XLIST
811         END_THIS COURSE.SECTIONS.ID
812     WHILE END.MARK DO REPEAT
813 END
814
815
* Process MUEN courses.  They all have a lower division (100) and upper
816     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"'
817     CALL S.EXECUTE('-C ':SQL)
818     IF (X.LOG.LEVEL EQ 'ALL') THEN
819         LOGS<-1> = LOWER(SQL) ; LOGS<-1> = CAPTURED.OUTPUT
820     END

```

```

821      X.STATUS = 0
822      XL.LIST = ''
823      CALL S.READLIST(XL.LIST,0,X.STATUS)
824      IF (X.STATUS EQ 1 AND DCOUNT(XL.LIST,@FM) GT 0) THEN
825          COL = 1
826          LOOP
827              REMOVE V.COURSE.SECTIONS.ID FROM XL.LIST AT COL
SETTING END.MARK
828      * Get the primary xlist section
829      FOR_THIS SINGLE REFERENCED COURSE.SECTIONS.ID
830          X.TERM1 = V.SEC.TERM
831          SWAP "/" WITH ' ' IN X.TERM1
832          X.COURSE.SEC.TERM = V.SEC.SUBJECT:'-':
V.SEC.COURSE.NO:'-':V.SEC.NO:'-':X.TERM1
833          STAT1 = FIELD(VL.SEC.STATUS,@VM,1,1)
834          MAIN.COURSE.SEC.ID = V.COURSE.SECTIONS.ID
835          X.NO = V.SEC.COURSE.NO
836          IF (X.NO LT 200) THEN
837              * change to upper division course number
838              X.NO[1,1] = '3'
839              SQL = 'WITH SEC.TERM EQ "' :V.SEC.TERM:
'" AND WITH EVERY SEC.DEPTS NE "ND" AND SEC.SUBJECT EQ "MUEN" AND SEC.CO
:X.NO:'"'
840              FOR_EACH REFERENCED SELECTED COURSE.SECTIONS.ID
USING NEWLIST CRITERIA SQL
841
* this should find a UD version, and only one even though we use FOR_EAC
842              X.TERM2 = V.SEC.TERM
843              SWAP "/" WITH ' ' IN X.TERM2
844              X.COURSE.SEC.TERM2 = V.SEC.SUBJECT:'-':
V.SEC.COURSE.NO:'-':V.SEC.NO:'-':X.TERM2
845              STAT2 = FIELD(VL.SEC.STATUS,@VM,1,1)
846              XL.SORT.ARRAY<1,-1> = X.TERM1:
X.COURSE.SEC.TERM
847              XL.ARR1<1,-1> = X.COURSE.SEC.TERM
848              XL.ARR2<1,-1> = X.COURSE.SEC.TERM2
849              IF (STAT1 EQ 'A' AND STAT2 EQ 'A') THEN
850                  XL.ARR3<1,-1> = 'active'
851              END ELSE
852                  XL.ARR3<1,-1> = 'deleted'
853              END
854              DOTS := '.'
855              TOTAL.XLISTS += 1
856              END_EACH COURSE.SECTIONS.ID
857          END
858      END_THIS COURSE.SECTIONS.ID
859      WHILE END.MARK DO REPEAT
860      END
861
862
* Process LABs. All sections get cross listed with the first section ((
863      SQL = "CLS":@FM:'SELECT COURSE.SECTIONS WITH SEC.TERM EQ '':

```



```

TERMS.PLUS.ADDED: ' AND WITH EVERY SEC.DEPTS NE "ND" AND'
864      SQL := ' EVAL "INDEX(SEC.COURSE.NO,':'L':',
1)" GT 0 AND SEC.NO EQ "01"'
865      CALL S.EXECUTE('-C ':SQL)
866      IF (X.LOG.LEVEL EQ 'ALL') THEN
867          LOGS<-1> = LOWER(SQL) ; LOGS<-1> = CAPTURED.OUTPUT
868      END
869      X.STATUS = 0
870      XL.LIST = ''
871      CALL S.READLIST(XL.LIST,0,X.STATUS)
872      IF (X.STATUS EQ 1 AND DCOUNT(XL.LIST,@FM) GT 0) THEN
873          COL = 1
874          LOOP
875              REMOVE V.COURSE.SECTIONS.ID FROM XL.LIST AT COL
SETTING END.MARK
876          * go through the primary (01) sections
877          FOR_THIS SINGLE REFERENCED COURSE.SECTIONS.ID
878              X.TERM1 = V.SEC.TERM
879              SWAP "/" WITH ' ' IN X.TERM1
880              X.COURSE.SEC.TERM = V.SEC.SUBJECT: '-':
V.SEC.COURSE.NO: '-':V.SEC.NO: '-':X.TERM1
881              STAT1 = FIELD(VL.SEC.STATUS,@VM,1,1)
882              MAIN.COURSE.SEC.ID = V.COURSE.SECTIONS.ID
883              * locate any sub sections
884              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:'''
885              FOR_EACH REFERENCED SELECTED COURSE.SECTIONS.ID
USING NEWLIST CRITERIA SQL
886
887          * there could be any number of these, so write each one out
887              X.TERM2 = V.SEC.TERM
888              SWAP "/" WITH ' ' IN X.TERM2
889              X.COURSE.SEC.TERM2 = V.SEC.SUBJECT: '-':
V.SEC.COURSE.NO: '-':V.SEC.NO: '-':X.TERM2
890              STAT2 = FIELD(VL.SEC.STATUS,@VM,1,1)
891              XL.SORT.ARRAY<1,-1> = X.TERM1:X.COURSE.SEC.TERM
892              XL.ARR1<1,-1> = X.COURSE.SEC.TERM
893              XL.ARR2<1,-1> = X.COURSE.SEC.TERM2
894              IF (STAT1 EQ 'A' AND STAT2 EQ 'A') THEN
895                  XL.ARR3<1,-1> = 'active'
896              END ELSE
897                  XL.ARR3<1,-1> = 'deleted'
898              END
899              DOTS := '.'
900              TOTAL.XLISTS += 1
901              END_EACH COURSE.SECTIONS.ID
902              END_THIS COURSE.SECTIONS.ID
903              WHILE END.MARK DO REPEAT
904          END
905
906
* Process GNST401. All sections get cross listed with the first section

```

```

907      SQL = "CLS":@FM:'SELECT COURSE.SECTIONS WITH SEC.TERM EQ ':
TERMS.PLUS.ADDED:' AND WITH EVERY SEC.DEPTS NE "ND" AND'
908      SQL :=
' SEC.COURSE.NO EQ "401" AND SEC.NO EQ "01" AND SEC.SUBJECT EQ "GNST"'
909      CALL S.EXECUTE('-C ':SQL)
910      IF (X.LOG.LEVEL EQ 'ALL') THEN
911          LOGS<-1> = LOWER(SQL) ; LOGS<-1> = CAPTURED.OUTPUT
912      END
913      X.STATUS = 0
914      XL.LIST = ''
915      CALL S.READLIST(XL.LIST,0,X.STATUS)
916      IF (X.STATUS EQ 1 AND DCOUNT(XL.LIST,@FM) GT 0) THEN
917          COL = 1
918          LOOP
919              REMOVE V.COURSE.SECTIONS.ID FROM XL.LIST AT COL
SETTING END.MARK
920
* go through the primary (01) sections (there may be more than one becau

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

929          FOR_EACH REFERENCED SELECTED COURSE.SECTIONS.ID
USING NEWLIST CRITERIA SQL
930
* there could be any number of these, so write each one out
931              X.TERM2 = V.SEC.TERM
932              SWAP "/" WITH ' ' IN X.TERM2
933              X.COURSE.SEC.TERM2 = V.SEC.SUBJECT:'-':
V.SEC.COURSE.NO:'-':V.SEC.NO:'-':X.TERM2
934              STAT2 = FIELD(VL.SEC.STATUS,@VM,1,1)
935              XL.SORT.ARRAY<1,-1> = X.TERM1:X.COURSE.SEC.TERM
936              XL.ARR1<1,-1> = X.COURSE.SEC.TERM
937              XL.ARR2<1,-1> = X.COURSE.SEC.TERM2
938              IF (STAT1 EQ 'A' AND STAT2 EQ 'A') THEN
939                  XL.ARR3<1,-1> = 'active'
940              END ELSE
941                  XL.ARR3<1,-1> = 'deleted'
942              END
943              DOTS := '.'
944              TOTAL.XLISTS += 1
945          END_EACH COURSE.SECTIONS.ID
946      END_THIS COURSE.SECTIONS.ID
947      WHILE END.MARK DO REPEAT
948      END
949

```

```

950         IF (XLISTS EQ TRUE) THEN
951             CALL S.SORT.ASSOC.AL(XL.SORT.ARRAY,XL.ARR1,XL.ARR2,
XL.ARR3,"","","","","")
952             * Write out the sorted lines
953             FOR X = 1 TO TOTAL.XLISTS
954                 X.LINE = XL.ARR1<1,X>
955                 X.LINE := ','
956                 X.LINE := XL.ARR2<1,X>
957                 X.LINE := ','
958                 X.LINE := XL.ARR3<1,X>
959                 X.LINE := EOL
960                 CALL S.WRITE.SEQ(X.LINE,2,X.ERR,X.MSG)
961                 IF (X.ERR) THEN LOGS<-1> = XLISTS.FILE.NAME:' Write '
:X.MSG
962             NEXT X
963
964             IF (X.LOG.LEVEL EQ 'ALL') THEN
965                 LOGS<-1> = DOTS
966             END
967             CALL S.WRITE.EOF(2,X.ERR,X.MSG)
968             IF (X.ERR) THEN LOGS<-1> = XLISTS.FILE.NAME:' Write ':
X.MSG
969             CALL S.CLOSE.SEQ(2,X.ERR,X.MSG)
970             IF (X.ERR) THEN LOGS<-1> = XLISTS.FILE.NAME:' Close ':
X.MSG
971             SQL = "CLS":@FM:"COPY FROM HOLD TO ":X.WORK.DIR:" ":
XLISTS.FILE.NAME
972             CALL S.EXECUTE('-C ':SQL)
973             IF (X.LOG.LEVEL EQ 'ALL') THEN
974                 LOGS<-1> = '' ; LOGS<-1> = LOWER(SQL) ; LOGS<-1> =
CAPTURED.OUTPUT
975             END
976             END ELSE
977             CALL S.CLOSE.SEQ(2,X.ERR,X.MSG)
978             IF (X.ERR) THEN LOGS<-1> = XLISTS.FILE.NAME:' Close ':
X.MSG
979             IF (X.LOG.LEVEL EQ 'ALL') THEN
980                 LOGS<-1> = 'No xlists found.'
981             END
982             END
983
984 RETURN
985
986 *****
987 UPLOAD.TO.CANVAS:
988 *****
989
990     * final processing
991     LOGS<-1> = ''
992     LOGS<-1> = 'Program: ':PGNAME
993     LOGS<-1> = 'Term(s): ':TERMS.PLUS.ADDED
994     LOGS<-1> = 'Total users uploaded: ':TOTAL.USERS
995     LOGS<-1> = 'Total enrollments uploaded: ':TOTAL.ENROLLMENTS
996     IF (MASTER.RUN EQ TRUE) THEN

```

```

997     LOGS<-1> = 'Total teacher enrolls uploaded: ':TOTAL.ENROLLS
998     LOGS<-1> = 'Total sections uploaded: ':TOTAL.SECTIONS
999     LOGS<-1> = 'Total courses uploaded: ':TOTAL.COURSES
1000    LOGS<-1> = 'Total xlists uploaded: ':TOTAL.XLISTS
1001    END
1002    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
1003        * points to note about this statement
1004        * 1) it runs Windows cmd first. /
c means execute a command and return from the shell.
1005
* 2) echo . is required. without it, powershell will flake out (or will
1006
* 3) cmd escape char is ^. Without it cmd will freak out and pass a pa
1007
* 4) "" found in the script reduce to a single quote.
1008
* 5) """" reduces to two quotes; the first to encompass the powershell a
1009
1010    IF (X.LOG.LEVEL EQ 'ALL') THEN
1011        LOGS<-1> = 'Uploading file(s) to Canvas server.'
1012    END
1013    STMT = "!cmd /c "
1014    STMT := QUOTE:"echo . | powershell -
ExecutionPolicy RemoteSigned C:\Users\Public\Documents
\WindowsPowerShell\Upload-CanvasSIS.ps1 "
1015    STMT :=
"'canvasbatch' '<put the password for the account to the left here>>' '
1016
1017    STMT := "-CanvasURI ':'X.CANVAS.URI:'' "
1018    STMT := "-
CanvasAccountID '<put your school's Account ID here>>' "
1019    STMT := "-CanvasAPIKey '<put your school's API key here' "
1020    IF (MASTER.RUN EQ TRUE) THEN
1021        STMT := "-BatchMode -BatchModeTermID ":TRIM(
TERMS.PLUS.ADDED,"/", "A"): "
1022    END
1023    STMT := "-Verbose -DeleteInputFiles -WorkingPath 'D:
\ftp_folders\live\":X.WORK.DIR:"":QUOTE
1024    IF (X.DEBUG EQ TRUE AND X.DEBUG.PS.SCRIPT EQ TRUE) THEN
LOGS<-1> = STMT
1025    IF (UPLOAD.FILES EQ 'Y') THEN CALL S.EXECUTE('-C ':STMT)
1026    LOGS<-1> = "Powershell return code: ":CAPTURED.OUTPUT
1027    END
1028    * set the time of the last successful run
1029    V.PUC.CONTROL.POINT.ID = 'CANVAS.SETTINGS'
1030    FOR_THIS SINGLE PUC.CONTROL.POINT.ID
1031    V.XFIELD2 = "Last Run Date":@VM:DATE()
1032    END_THIS PUC.CONTROL.POINT.ID
1033    RETURN
1034

```

```

1035 *****
1036 WRITE.LOG:
1037 *****
1038
1039 * write log file out
1040 V.PUC.CANVAS.LOG.ID = CANVAS.LOG.CTR
1041 FOR_THIS SINGLE PUC.CANVAS.LOG.ID
1042     V.XPCL.DATE = DATE()
1043     V.XPCL.TIME = TIME()
1044     CONVERT @FM TO @VM IN LOGS
1045     SWAP CHAR(192):"21":CHAR(192):"23":CHAR(192) WITH '' IN LOGS
1046     VL.XPCL.LOG = LOGS
1047 END_THIS PUC.CANVAS.LOG.ID
1048
1049 RETURN
1050
1051 *****
1052 WRITE.LOG.COUNTER:
1053 *****
1054
1055
1056 *uncomment these 4 lines if you want to write the log info to a sequenti
1057
1058 *CALL S.OPEN.SEQ('HOLD','SDP',2,'W',X.ERR,X.MSG)
1059 *CALL S.WRITE.SEQ(LOGS,2,X.ERR,X.MSG)
1060 *CALL S.WRITE.EOF(2,X.ERR,X.MSG)
1061 *CALL S.CLOSE.SEQ(2,X.ERR,X.MSG)
1062
1063 * write log counter out
1064 V.PUC.CONTROL.POINT.ID = 'PUC.CANVAS.LOG.CTR'
1065 FOR_THIS SINGLE PUC.CONTROL.POINT.ID
1066     V.XFIELD1 = CANVAS.LOG.CTR
1067 END_THIS PUC.CONTROL.POINT.ID
1068
1069 RETURN
1070
1071 *****
1072 XDEBUG.001:
1073 *****
1074 IF (X.DEBUG EQ TRUE) THEN
1075     MASTER.RUN = TRUE
1076     X.WORK.DIR := '.DEV'
1077     X.LOG.LEVEL = 'ALL'
1078     X.CANVAS.URI = 'https://puc.beta.instructure.com'
1079 END
1080 RETURN
1081
1082 *****
1083 XDEBUG.002:
1084 *****
1085 RETURN
1086 IF (X.DEBUG EQ TRUE) THEN
1087     XT = TERMS:ADDED.TERMS

```

```
1087     XT = CONVERT( "'/'", "", XT): "_"
1088     IF ( INDEX(UFILE.NAME, "_", 1) GT 0) THEN
1089         UFILE.NAME = XT:FIELD(UFILE.NAME, "_", 2, 1)
1090         EFILE.NAME = XT:FIELD(EFILE.NAME, "_", 2, 1)
1091         ENROLL.FILE.NAME = XT:FIELD(ENROLL.FILE.NAME, "_", 2, 1)
1092         XLISTS.FILE.NAME = XT:FIELD(XLISTS.FILE.NAME, "_", 2, 1)
1093         SECTIONS.FILE.NAME = XT:FIELD(SECTIONS.FILE.NAME, "_", 2, 1)
1094         COURSES.FILE.NAME = XT:FIELD(COURSES.FILE.NAME, "_", 2, 1)
1095     END ELSE
1096         UFILE.NAME = XT:UFILE.NAME
1097         EFILE.NAME = XT:EFILE.NAME
1098         ENROLL.FILE.NAME = XT:ENROLL.FILE.NAME
1099         XLISTS.FILE.NAME = XT:XLISTS.FILE.NAME
1100         SECTIONS.FILE.NAME = XT:SECTIONS.FILE.NAME
1101         COURSES.FILE.NAME = XT:COURSES.FILE.NAME
1102     END
1103 END
1104 RETURN
```