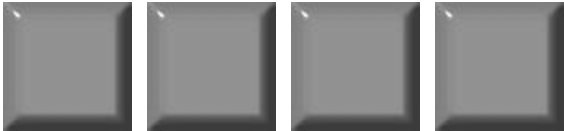


# Writing Maintenance-Friendly Customizations



Northwest Datatel Users  
Group 2001

Darcy Latremouille  
Datatel, Inc.  
June 26, 2001

## Overview



- What customizations and what maintenance?
- Development environment
- Overall techniques
- Specific techniques for coding
- Storing custom data elements
- Doing maintenance

---

---

---

---

---

---

---

---

## What Customizations?



- Modifications to standard Datatel processes
- Add-ons that interface with standard Datatel processes
- Standalone add-ons
- Modifications to generated source (standard forms, print routines, UT processes)

---

---

---

---

---

---

---

---

## What Maintenance?



- Software Updates (SOS patches)
- New releases
- What does maintenance involve?
  - Merging your custom changes into the updated standard version
  - Merging the standard version's changes into your custom version

---

---

---

---

---

---

---

---

## Why is Maintenance Important?



- Custom processes may fail due to change in standard product
- Include bug fixes and enhancements in your custom processes
- Customization may be obsolete
- Keeping up-to-date with Software Updates makes migrating to a new release easier

---

---

---

---

---

---

---

## Development Environment



- Per Datatel's "Managing Envision Custom Source" recommendations
- Two release accounts
- Develop in "X" applications
- Rename processes... or don't
- Use Move Utilities to move custom software (MOVEINFO records are the key)

---

---

---

---

---

---

---

## Documentation is the Key



- Document **why** the customization was created
- Document your changes
  - Mark the specific blocks you've changed
  - Leave the original code in (commented out)
  - Your MOVEINFO record is vital documentation
- Document which standard process your custom ones are based on

---

---

---

---

---

---

---

## General Recommendations



- Take advantage of Envision's power
  - Envision Basic is compact
  - Let Envision handle your I/O
  - Use the CDD and shared inserts
  - Don't modify generated source
- Isolate changes as much as possible
- Follow standard "good coding" practices

---

---

---

---

---

---

---

---

## Your MOVEINFO Records



- Exactly one MOVEINFO record per customization
- If it's not custom, it shouldn't be in the record
- ...including when you run MDEV / CATX
- MDEF's "Include" flag isn't always honored
- Not everything is included automatically

---

---

---

---

---

---

---

---

## Specific Techniques for Coding



- Use inserts
- GOSUB to internal subroutines
- Add your customizations "to the end"
- Isolate changes to as few hooks as possible

---

---

---

---

---

---

---

---

## Custom Data Elements



- User fields vs. co-files
- Co-files preferred
  - They're entirely custom: Datatel won't overwrite
  - User fields impact database independence
  - "WITH USER9 = 'X'" is not intuitive to read
  - Create virtual fields in the standard file
  - Performance impact is negligible

---

---

---

---

---

---

---

---

## If Data is Already In User Fields



- Options:
  - Move to a custom file
  - Build a logical view over the base file
  - Use the original "xxx.USERy" field names in processes; build virtual fields for queries
  - Rename the user fields to custom names
- Be careful moving appl.FILE.SPECS records

---

---

---

---

---

---

---

---

## The Real Test: It's Maintenance Time!



- Identify which processes are affected
- Identify what has changed in each affected process
- Update your process

---

---

---

---

---

---

---

---

## Identifying Processes That Need Maintenance



- Compare Express Load's control files to your MOVEINFO records
- Watch out for renamed items: a change to RGS001 will affect your custom XRG001
- Same with I\_INPUT.RGS001.SECTION vs. I\_INPUT.XRG001.SECTION
- Watch for appl.FILE.SPECS delivered for files you use
- Watch for subroutines whose argument lists change

---

---

---

---

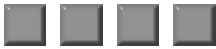
---

---

---

---

## Identifying Changes to a Process



- UNIX's "sdiff" compares files side-by-side (useful for code)
- UT - DIFF compares any records (useful for appl.FILE.SPECS, appl.PRC.S.DEF, appl.PRC.S.GEN)
- Other useful utilities exist, like X's "xdiff"

---

---

---

---

---

---

---

---

## Identifying Changes to a Process



- General understanding of how Envision generated source looks helps
- Set "Uses Standard I/O" on SGP/BGP to Yes to see process fields (otherwise, they're hidden in the STRATEGIES record)
- If you've upgraded Envision, consider regenerating the standard process
- FR.UTILITIES - EXPAND.SOURCE.INSERTS

---

---

---

---

---

---

---

---

## What to Compare



- First step: run both versions
- Batch Processes:
  - Compare the appl.INSERTS for coding changes
  - Compare the appl.SOURCE or appl.SUBROUTINES for data element list
  - Watch out for changed inserts – use EXPAND.SOURCE.INSERTS to be sure

---

---

---

---

---

---

---

---

## What to Compare



- Screens:
  - appl.SOURCE
  - It would be nice to have a compare utility specifically for appl.PRCS.DEF records
  - Could write files out to UNIX with @VM's converted to newlines and tabs (two tabs for @SM's)
  - Then can use sdiff

---

---

---

---

---

---

---

---

## Updating Your Process



- Make the changes in the ToolKit then regen
- Watch for compiler uninitialized variables
- Test the process
- In the same session, run the standard version to find 'incorrect redefinition of common'
- Compare the generated source again

---

---

---

---

---

---

---

---

# WRITING MAINTENANCE-FRIENDLY CUSTOMIZATIONS

## NWDUG 2001 - HANDOUTS

---

### Contents

- Defining a Logical View Over User Fields 1
- Comparing the Express Load Control Files with MOVEINFO Records 2
- Setting Up to Use "sdiff" 3
  - Make Your Terminal Screen as Big as Possible 3
  - Make sdiff Ignore Whitespace 3
  - Let UNIX do the Walking 4
- Using "sdiff" Summary Mode 5
- Scrolling Through the Output 6
- Comparing Screens 7
- Comparing Batch Processes and Subroutines 10

### Defining a Logical View Over User Fields

Create the file on FS. Use a custom filename, but enter the name of the standard file as the physical file:

```
06/24/01 11:59          DEFINE FILE SPECIFICATIONS          T01ST    FS
                    *** T01.AR.ACCTS ***
Created On: 03/24/00 By: LCH          Changed On: 06/05/01 By: MISC
-----
 1 Physical File Name..: AR.ACCTS
 2 Alias File Names....: 1:
 3 Freeze Fld Placement: No
 4 Release Status.....: R Releasable
 5 Release Modules.....: 1: BASE          2: T01S          3:
 6 File Type.....: DYNAMIC    Dynamic allocation
 7 Category.....: T - Transaction
 8 * Purpose.: 1: Logical View of the AR.ACCTS file for Troy.
 9 * Tech Doc: 1:
10 Maximum Composite Field Size (Oracle ONLY):
11 Maximum Expected Row Size (Oracle ONLY)...:
-----
Press FIELD JUMP to make changes, RETURN to confirm or CANCEL to abort: _
```

Define the fields on DEL and DEP. On DEL, use the field locations of the user fields in the standard file. Here's what it looks like on the FILE when all the fields have been added:

06/24/01 12:00		FILE STRUCTURE DEFINITION			T01ST	FILE
T01.AR.ACCTS						
1 * Data Field Names		Pos	Data Type	Size	Justify	Conversion
1:	T01.PLAN.BAL	9	D Data	10	R Right	MD2,
2:	T01.PLAN.TERMS	10	A Associatio	5	L Left	
3:	T01.PLAN.DDATES	11	A Associatio	8	R Right	D2/
4:	T01.PLAN.TAMTS	12	A Associatio	10	R Right	MD2,
5:	T01.PLAN.OAMTS	13	A Associatio	10	R Right	MD2,
6:	T01.PLAN.TPAID	14	A Associatio	10	R Right	MD2,
7:	T01.PLAN.OPAID	15	A Associatio	10	R Right	MD2,
8:	T01.PLAN.PDUE	16	D Data	10	R Right	MD2,
9:	T01.PLAN.PDUE.PD	17	D Data	10	R Right	MD2,
10:	T01.PLAN.PBAL	18	D Data	10	R Right	MD2,
2 * Key Fields		Size	Justify	Lookup	Prompt	
1:	T01.ARA.PERSON	10	L Left			
3 Physical File...: AR.ACCTS						
4 * File Desc....: 1: Logical View of the AR.ACCTS file for Troy.						
5 File Modules....: 1: BASE                    2: T01S                    3:						

Press FIELD JUMP to make changes, RETURN to confirm or CANCEL to abort: \_

To use this in a process, just add your custom fields to the process element list. In a batch process, set V.T01.ARA.PERSON.ID = V.AR.PERSON.ID before your FOR\_EACH T01.ARA.PERSON.ID. If it's a screen, add the custom keys as phantoms after the phantoms from the original file, then set the input source hook to:

```
INPUT.DATA = V.ARA.PERSON.ID
```

(This example being a key with a multi-part file, we would do the same for the other part of the key.)

In other words, the logical view acts like a co-file.

### **Comparing the Express Load Control Files with MOVEINFO Records**

A utility must be written to compare Express Load control files with MOVEINFO records. Users will need the ability to define a cross-reference of custom names to standard names – i.e., RGS001 to XRGS001 – to map standard processes to their custom versinos.

At a minimum the utility must:

- Select all the EXPL.PATCH.CTL records for the software updates you've just loaded into your development account, saving ELPC.ITEMS (keys to the EXPL.ITEM.CTL file)
- Loop through the EXPL.ITEM.CTL records
- If ELIC.FILE is a file from the Envision file suite (i.e., ST.PRC.S.DEF), translate the application name to literally "appl". (So that we can compare X applications to base applications.)
- If ELIC.FILE and ELIC.ITEM match MI.SOURCE.FILES and MI.RECORD.NAMES in any MOVEINFO records, report the item. (Translate Envision file names from the MOVENIFO records before comparing them.)
- For all items in the user's translation table, swap any values (i.e. RGS001 to XRGS001) and look in MOVEINFO again.

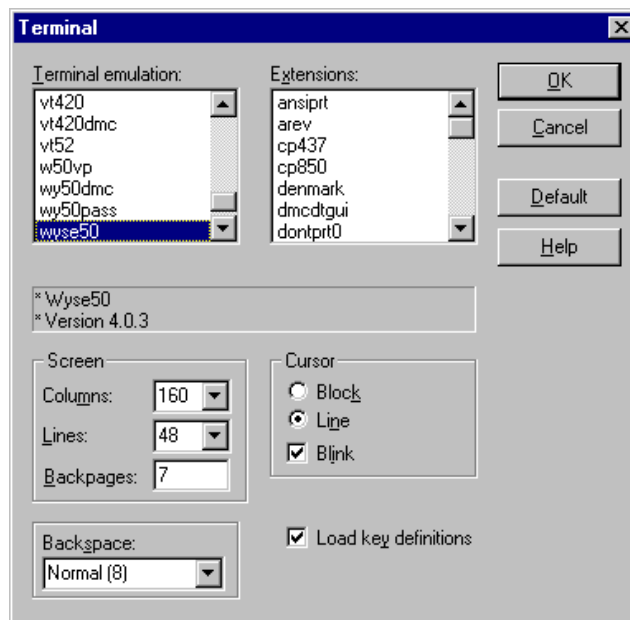
Nice-to-haves:

- A utility that makes a copy of the argument lists for all processes before loading Software Updates.
- The MOVEINFO compare utility above should look through all processes named in the MOVEINFO records and find all subroutine calls. Then it should compare the old argument lists with the new, and if the argument list for one of the subroutines changed, report the item and the subroutine.
- The MOVEINFO compare utility should also find all inserts included in any processes named in MOVEINFO records (whether the inserts were customized or not), and see if the inserts were re-delivered in any software updates. Since inserts are often shipped even if they haven't changed, the MOVEINFO compare utility could either compare the non-comment lines in development account version of the insert with the version in the live release account to see if the insert actually changed. Or perhaps the "pre-load" utility that makes copies of argument lists could generate a checksum for inserts.

### **Setting Up to Use "sdiff"**

#### **Make Your Terminal Screen as Big as Possible**

Set width to at least 132, but preferably 160 or more. In Datatel Terminal (aka FrontView Terminal aka wIntegrate), select Setup – Terminal:



Set the columns and lines appropriately. Note that you can type over them – you don't have to use what's in the drop-down box. Datatel Terminal will suggest that number of backpages to use. In earlier releases, you must set it by hand.

Inform UNIX of your new screen size:

```
$ stty rows 48 columns 160
```

#### **Make sdiff Ignore Whitespace**

You don't want differences in indentation levels to show up as differences if the text on the line is the same. We can put a shell wrapper around diff to prevent this from happening.

1. Find out and note the actual path to the diff command:

```
$ whence diff
```

2. Create a wrapper for diff in the "bin" directory under your home directory, for example, /home/dl/bin.

```
$ cd /home/dl/bin
$ vi diff
Insert the following text:
  #!/bin/ksh
  /path/to/diff -w $@
:wq    (to save and exit)
```

3. Make sure that the your personal "bin" directory is the first one in your PATH. This will either be set in your .profile or the standard profile that all users use – your system administrator will know where that file is. To see your current path, just do

```
% echo $PATH
```

4. That's it! sdiff calls diff to do the actual comparison, but now thanks to your wrapper it will include the "-w" argument which will cause it to ignore all whitespace. Much more useful output, and it saves us the trouble of either stripping leading spaces or manually comparing lines where the indentation was different.

An alternative would be to put the "diff" wrapper in the /usr/local/bin directory, which is probably already the first thing in the path anyway but it may end up affecting people or processes that don't expect this change. But then again, who runs diff other than you?

### Let UNIX do the Walking

You'll be comparing files between the directories for your custom application and the directories for base product. To save yourself from having to type the long paths over and over, set up shell environment variables pointing to the base product directories. Then you can reference them much more easily. You could set these up in your .profile file so that they're always available.

```
$ export STS=/datatel/development/CDEV16.461/ST/ST.SOURCE
$ export STI=/datatel/development/CDEV16.461/ST/ST.INSERTS
$ export STSUB=/datatel/development/CDEV16.461/ST/ST.SUBROUTINES
```

## Using "sdiff" Summary Mode

To get an overview of the extent of the changes, first run sdiff in summary mode. This mode only displays only lines that are different. (Normal mode displays all lines.)

- | means that the line exists in both files but is different
- < means that the line only exists in the file on the left
- > means that the line only exists in the file on the right

Summary mode is invoked with the "-s" argument. The "-w" argument tells sdiff the screen width. Be sure it matches your actual screen width.

```
$ cd XST.SOURCE
$ sdiff -s -w 132 XRG001 $STS/RGS001 | more
```

```
1,3c1,3
* Process XRG001 generated by ENVISION Screen Generator - Ve | * Process RGS001 generated by ENVISION Screen Generator -
* Date: 17:12:42 Jun 04 2001 | * Date: 09:36:09 Sep 01 2000
* Logname: MISC | * Logname: JCF
14d13
$INSERT I_COMMON FROM XST.SOURCE <
46,49c45
CALLABLE.PROCESSES = "CCDU12,RGS010,PASU74,S.REG.STU.COURSE,RGS | $INSERT I_RGS001.INI
CONVERT "," TO @FM IN CALLABLE.PROCESSES <
CALLABLE.PROCESS.MNEMONICS = "" <
$INSERT I_XTS006.INI <
89c85
$INSERT I_SCREEN.INIT FROM XST.INSERTS | $INSERT I_SCREEN.INIT FROM ST.INSERTS
107a104
> LOCAL.DEBUG = 1
109a107
> LOCAL.DEBUG = 0
115d112
RETAIN.DATA.BUFFERS = 1 <
117a115,117
> X.PROCESS = "RGS001"
> $INSERT I_PRCES.ENTRY.SET.DEBUG FROM CORE.INSERTS
> XL.DEBUG.MSG = ""
123c123
CRNT.FLD.INDX = 17 | CRNT.FLD.INDX = 16
132c132
CRNT.FLD.INDX = 20 | CRNT.FLD.INDX = 19
141c141
CRNT.FLD.INDX = 22 | CRNT.FLD.INDX = 21
```

```

145c145
CRNT.FLD.INDX = 26 | CRNT.FLD.INDX = 24
149c149
CRNT.FLD.INDX = 28 | CRNT.FLD.INDX = 26
158c158
CRNT.FLD.INDX = 31 | CRNT.FLD.INDX = 29
191c191
ON FLD.INDX GOSUB FLD.TODAY, FLD.NOW, FLD.PROCESS.TITLE, FLD.CU | ON FLD.INDX GOSUB FLD.TODAY, FLD.NOW, FLD.PROCESS.TITLE, F
243a244,248

> IF DATATEL.DEBUG THEN
> XL.DEBUG.MSG = ""
> XL.DEBUG.MSG<-1> = "BEFORE I_PRCS.END.RGS001"
> $INSERT I_DEBUG FROM CORE.INSERTS
> END

388a394,397

> IF DATATEL.DEBUG THEN
> XL.DEBUG.MSG<-1> = "AFTER I_PRCS.END.RGS001"
> $INSERT I_DEBUG FROM CORE.INSERTS
> END

stdin

```

The numbers that prefix each change represent line numbers in each file. Enter “`man diff`” for full details.

### **Scrolling Through the Output**

“`stdin`” is the “more” prompt. Useful (case-sensitive) commands at this prompt are:

- [Spacebar] – forward one page
- [Return] – forward one line
- b – scroll up one page
- u – scroll up half a page
- d – scroll down half a page
- /*string* – search forward for *string*
- ?*string* – search backwards for *string*
- G – Return to the top of the file
- h – Display help

## Comparing Screens

Use sdiff in “normal” mode to display the full contents of each file. This lets you see the context, which you will need to identify the hook where the change is located.

```
$ cd XST.SOURCE
$ sdiff -w 132 XRG001 $STS/RGS001 | more
```

Here we see a change where the custom version has replaced standard standard code:

```
X.EDITED.DATA = ";SEC.CRNT.STATUS.INDEX 1":X.EDITED.DATA
END
END
END
EDITED.DATA = X.EDITED.DATA
ERROR.OCCURRED = X.ERROR.OCCURRED
MSG = X.MSG
IF XL.RGC.SECTION.LOOKUP.CRITERIA THEN
CONVERT @VM TO " " IN XL.RGC.SECTION.LOOKUP.CRITERIA
XL.RGC.SECTION.LOOKUP.CRITERIA = TRIM(XL.RGC.SECTION.LOOKUP.CRI
LKUP.ATTR<LKUP.ATTR.SELECT> = LKUP.ATTR<LKUP.ATTR.SELECT>:" ":X
END
END
IF NOT(ERROR.OCCURRED) THEN
IF WARNING.OCCURRED THEN GOSUB ERROR.MESSAGE

X.EDITED.DATA = ";SEC.CRNT.STATUS.INDEX 1":X.EDITED.DATA
END
END
END
EDITED.DATA = X.EDITED.DATA
ERROR.OCCURRED = X.ERROR.OCCURRED
MSG = X.MSG
IF VL.RGC.SECTION.LOOKUP.CRITERIA THEN
CONVERT @VM TO " " IN VL.RGC.SECTION.LOOKUP.CRITERIA
VL.RGC.SECTION.LOOKUP.CRITERIA = TRIM(VL.RGC.SECTION.LOOKU
LKUP.ATTR<LKUP.ATTR.SELECT> = LKUP.ATTR<LKUP.ATTR.SELECT>:
END
END
IF NOT(ERROR.OCCURRED) THEN
IF WARNING.OCCURRED THEN GOSUB ERROR.MESSAGE
```

---

Here we see new code that was added. Note that this isn't exactly the code you'll see in the ToolKit. Part of it is the generated version of CALL\_SCREEN SUSPEND using the \$SECONDARY keyword. Which is why I would like to be able to compare appl.PRC.SDEF records directly instead of looking at the generated source...

```
IF SCREEN.REFRESH THEN GOSUB REFRESH.SCREEN
MSG = "NODETAIL"; WARNING.OCCURRED = 1
END
END
END
GOSUB INSRT.INPT.RGS001.CHECK.CONFLICT
IF LEN(EDITED.DATA) THEN <
IF NOT(LEN(V.LIST.VAR13)) THEN <
V.LIST.VAR13 = V.SEC.LOCATION <
VL.LIST.VAR13<1,WNDW.VAR.SCS.COURSE.SECTION.INDX> = V.SEC.LOCAT <
END <
PRCS.TO.EXEC = "XS.CHECK.PREREQ.TESTS" <
IF LEN(PRC.SDEF) THEN <
IF NOT(LEN(EDITED.DATA)) THEN <
TEMP.VAR.SCS.COURSE.SECTION = V.VAR.SCS.COURSE.SECTION <
END ELSE <
```

```

TEMP.VAR.SCS.COURSE.SECTION = EDITED.DATA <
END <
GOSUB REBUILD.RECORDS <
CALL @PRCS.TO.EXEC(V.ID,TEMP.VAR.SCS.COURSE.SECTION,X.WARNING.O <
IF (RETURN.CODE NE CMD.CA) THEN GOSUB PARSE.RECORDS <
IF LEN(TEMP.VAR.SCS.COURSE.SECTION) AND (NOT(LEN(EDITED.DATA)) <
IF (EDITED.DATA[1,4] EQ "$NEW") THEN <
EDITED.DATA = V.VAR.SCS.COURSE.SECTION <
END ELSE <
EDITED.DATA = TEMP.VAR.SCS.COURSE.SECTION <
END <
END <
END <
END <
IF X.WARNING.OCCURRED THEN <
CALL S.ARG.ERROR.MESSAGE(' ',X.WARNING.OCCURRED,X.MSG,'') <
END <
IF X.AUTO.REG AND ERROR.OCCURRED THEN IF X.AUTO.REG AND ERROR.OCCURRED THEN
CALL S.ARG.ERROR.MESSAGE(ERROR.OCCURRED," ",MSG,MSG.ARGUMENTS) CALL S.ARG.ERROR.MESSAGE(ERROR.OCCURRED," ",MSG,MSG.ARGUMEN
ERROR.OCCURRED = 1 ERROR.OCCURRED = 1
MSG = "RG041" MSG = "RG041"
X.AUTO.REG = 0 X.AUTO.REG = 0

```

---

Changes to the PARSE and REBUILD generated routines indicate that the custom process accesses different fields than the standard version. They might be painted on the screen, they might be on SXR, or they might possible be in one of the other “demand” hooks (such as on SWC or SEH).

```

END END
GOSUB EVALUATE.SECURITY.STATE GOSUB EVALUATE.SECURITY.STATE
RETURN RETURN
*-----* *-----*
PARSE.STUDENT.COURSE.SEC: PARSE.STUDENT.COURSE.SEC:
V.SCS.STUDENT = R.STUDENT.COURSE.SEC<SCS.STUDENT> V.SCS.STUDENT = R.STUDENT.COURSE.SEC<SCS.STUDENT>
V.SCS.USER2 = R.STUDENT.COURSE.SEC<SCS.USER2> <
V.SCS.COURSE.SECTION = R.STUDENT.COURSE.SEC<SCS.COURSE.SECTION> <
V.SCS.LOCATION = R.STUDENT.COURSE.SEC<SCS.LOCATION> <
RETURN RETURN
*-----* *-----*
READ.COURSE.SECTIONS: READ.COURSE.SECTIONS:
KEY.COURSE.SECTIONS = V.COURSE.SECTIONS.ID KEY.COURSE.SECTIONS = V.COURSE.SECTIONS.ID
IF NOT(LEN(V.COURSE.SECTIONS.ID)) THEN IF NOT(LEN(V.COURSE.SECTIONS.ID)) THEN
KEY.COURSE.SECTIONS = "" KEY.COURSE.SECTIONS = ""

```

Here's a set of changes that aren't logic changes at all. The original process was generated under Envision 4.4.2 and the custom version was generated under Envision 4.6.1. Envision 4.6.1 generates slightly different code to call processes. (You'll see the WIN.DBCAPT line often as it contains the process mnemonic.)

```

PRCS.TO.EXEC = "ACS017"
THREAD.FLAG<2> = PROCESS.SECURITY.LEVEL
IF LEN(P RCS.TO.EXEC) THEN
PROCESS.OPERANDS(1) = V.ID
GOSUB NONINPUT.CALL.SCREEN
IF THIN.GUI THEN
WINTSCR = WINT.NAME
CALL WIN.DBENALL(WINTSCR,1,1)
CALL WIN.DBENALL(WINTSCR,0,0)
CALL WIN.DBCAPT('MDI','','Datatel - ':SN.CURRENT.APPLICATION:'
END

V.ID = PROCESS.OPERANDS(1)
V.STUDENT.HIATUS.ID = PROCESS.OPERANDS(2)
END
END ELSE
X.MAX = DCOUNT(XL.ORIG.STUDENT.ACAD.CRED.ID,@VM)
FOR X.CTR = 1 TO X.MAX

```

```

PRCS.TO.EXEC = "ACS017"
THREAD.FLAG<2> = PROCESS.SECURITY.LEVEL
<
PROCESS.OPERANDS(1) = V.ID
GOSUB NONINPUT.CALL.SCREEN
IF THIN.GUI THEN
WINTSCR = WINT.NAME
CALL WIN.DBENALL(WINTSCR,1,1)
CALL WIN.DBENALL(WINTSCR,0,0)
CALL WIN.DBCAPT('MDI','','Datatel - ':SN.CURRENT.APPLICATI
END
>
V.ID = PROCESS.OPERANDS(1)
V.STUDENT.HIATUS.ID = PROCESS.OPERANDS(2)
<
END ELSE
X.MAX = DCOUNT(XL.ORIG.STUDENT.ACAD.CRED.ID,@VM)
FOR X.CTR = 1 TO X.MAX

```

---

Same thing here. Something to do with Oracle (as evidenced by the presence of SN.SQLATOR.RESPOSITORY).

```

IF NOT(ERROR.OCCURRED) THEN
IF COMMAND NE CMD.CA THEN
V.SYS.USER.ID = EDITED.DATA

GOSUB READ.RECORDS
CRNT.FLD.INDX = 7
IF SN.SQLATOR.REPOSITORY THEN
IF (LEN(V.SYS.USER.ID) GT CRNT.FLD.WDTH) THEN
IF OPERS.ADD.MODE THEN
MSG.ARGUMENTS=CRNT.FLD.WDTH; MSG="LONGFLD"
ERROR.OCCURRED=1
RECORD.CANCEL=2
END ELSE
MSG.ARGUMENTS=CRNT.FLD.WDTH; MSG="LONGFLD.WARN"
WARNING.OCCURRED=1
END
END
END

```

```

IF NOT(ERROR.OCCURRED) THEN
IF COMMAND NE CMD.CA THEN
V.SYS.USER.ID = EDITED.DATA
> IF SCREEN.REFRESH THEN GOSUB REFRESH.SCREEN
GOSUB READ.RECORDS
<
<
<
<
<
<
<
<
<
END
END
END

```

Here's a legitimate change to the standard version. (Granted, a poor example.) Changes to the standard version should be merged into the custom version if they don't conflict with your customizations.

```
CALL S.ARG.ERROR.MESSAGE("1","",X.TEMP.MSG,X.TEMP.ARGS)
NEXT X.MSG.CTR
WARNING.OCCURRED = ""
MSG = ""
MSG.ARGUMENTS = ""
END
END
END
END
END
END
```

```
X.PROCESS.END = PROCESS.END
X.THREAD.FLAG = THREAD.FLAG
PROCESS.END = ""
THREAD.FLAG = ""
```

```
IF X.REPORT.ERRORS.OPTION = 0 OR X.REPORT.ERRORS.OPTION = 1 OR
CONVERT @FM TO @VM IN XL.RECORD.IDS
```

```
CALL S.ARG.ERROR.MESSAGE("1","",X.TEMP.MSG,X.TEMP.ARGS)
NEXT X.MSG.CTR
WARNING.OCCURRED = ""
MSG = ""
MSG.ARGUMENTS = ""
END
END
END
END
END
END
```

```
> IF DATATEL.DEBUG THEN
> XL.DEBUG.MSG<-1> = "AFTER I_PRC.S.END.RGS001"
> $INSERT I_DEBUG FROM CORE.INSERTS
> END
```

```
X.PROCESS.END = PROCESS.END
X.THREAD.FLAG = THREAD.FLAG
PROCESS.END = ""
THREAD.FLAG = ""
```

```
> IF DATATEL.DEBUG THEN
> XL.DEBUG.MSG<-1> = "BEFORE I_EVALUATE.RULES.DETAIL"
> $INSERT I_DEBUG FROM CORE.INSERTS
> END
```

```
IF X.REPORT.ERRORS.OPTION = 0 OR X.REPORT.ERRORS.OPTION = 1 OR
CONVERT @FM TO @VM IN XL.RECORD.IDS
```

### **Comparing Batch Processes and Subroutines**

It's much easier maintaining batch processes, as you can compare the INSERTS records to pick up most of the changes. Here you see the original Envision Basic, not the generated version. Plus comments remain intact, so surrounding your changes with comments makes it clear what's custom and what's not. Do remember to also compare the appl.PRC.S.DEF records or the appl.SOURCE/appl.SUBROUTINES to identify if additional data elements were added to BED, or if code was added to BXR or the reporting hooks.

```
$ cd XST.INSERTS
$ sdiff -w 132 XS.PRINT.REG.STATEMENT $STI/S.PRINT.REG.STATEMENT
```

```
X.NBR.LINES = DCOUNT(XL.STMNT.DETAILS.FA,@FM)
FOR IDX = 1 TO X.NBR.LINES
  X.PRINT.LINE = XL.STMNT.DETAILS.FA<IDX>
  GOSUB PRINT.A.LINE
NEXT IDX
END
```

```
;* x.stmnt.unsent.fa.in
```

```
X.NBR.LINES = DCOUNT(XL.STMNT.DETAILS.FA,@FM)
FOR IDX = 1 TO X.NBR.LINES
  X.PRINT.LINE = XL.STMNT.DETAILS.FA<IDX>
  GOSUB PRINT.A.LINE
NEXT IDX
END
```

```
;* x.stmnt.unsent.
```

```
* 5/15/00 DL - Print payment plan info
GOSUB PRINT.PPLAN
* end changes
```

```
<
<
<
<
```

```
IF AL.MESSAGE THEN
  * there's a message to be printed

  * no column header (in case of a page break in the middle of
  * for the message section
  XL.COLUMN.HEADING = ""
  X.COLUMN.HEADER.LINES = DCOUNT(XL.COLUMN.HEADING,@FM)
```

```
IF AL.MESSAGE THEN
  * there's a message to be printed

  * no column header (in case of a page break in the midd
  * for the message section
  XL.COLUMN.HEADING = ""
  X.COLUMN.HEADER.LINES = DCOUNT(XL.COLUMN.HEADING,@FM)
```

The PRINT.PPLAN subroutine was added to the very bottom to keep it isolated from the main code. It could also have been an insert.

---

Here's code that was added to the standard version. We really need this in our custom version.

```
IF DATATEL.DEBUG THEN
  XL.DEBUG.MSG<-1> = "Entering FORMAT.STMNT.SUMMARY.FINANCIAL"
END
```

```
IF DATATEL.DEBUG THEN
  XL.DEBUG.MSG<-1> = "Entering FORMAT.STMNT.SUMMARY.FINAN
END
```

```
GOSUB GET.PAYMENT.AMOUNTS
```

```
GOSUB GET.PAYMENT.AMOUNTS
```

```
X.TOTAL.CHARGES = X.BALANCE.FORWARD + X.TOTAL.CURRENT.CHARGES
```

```
> *
> * 3/30/00 10:32:38 AM Tyler G. Spires (Activity 215.27)
> * The ballance forward was was being used when
> * REG.CONTROLS/REG.USERS parameter is set to "NO"
> *
> * X.TOTAL.CHARGES = X.BALANCE.FORWARD + X.TOTAL.CURRENT.CH
> *
> IF X.STMNT.BAL.FORWARD.IND EQ "Y" THEN
|   X.TOTAL.CHARGES = X.BALANCE.FORWARD + X.TOTAL.CURRENT.C
| END ELSE
>   X.TOTAL.CHARGES = X.TOTAL.CURRENT.CHARGES
> END
> *
> * 3/30/00 11:11:59 AM Tyler G. Spires (end of Activity 215
> *
```

```
X.TOTAL.AMOUNT.DUE = X.TOTAL.CHARGES
X.TOTAL.AMOUNT.DUE -= X.CASH.PMTS
X.TOTAL.AMOUNT.DUE -= X.FA.PMTS
```

```
X.TOTAL.AMOUNT.DUE = X.TOTAL.CHARGES
X.TOTAL.AMOUNT.DUE -= X.CASH.PMTS
X.TOTAL.AMOUNT.DUE -= X.FA.PMTS
```