Using Blaise 4.8 for Census Coverage Measurement

Roberto V. Picha, U.S. Census Bureau, USA

1. Introduction

This paper discusses a few of the innovations made to the Census Coverage Measurement Person Interview (CCM PI) survey instrument developed by the U.S. Census Bureau using Blaise 4.8. The original instrument was developed in preparation for the 2006 Dress Rehearsal using Blaise 4.6. As the 2010 Decennial Census operations approached, the Technologies Management Office Authoring Staff was requested to program the 2009 CCM PI Dress Rehearsal instrument by starting with the 2006 instrument and adding enhancements to it. The updated instrument specs contained new sets of requirements that presented challenges for the Authoring Staff.

This paper covers how the following functionality was enhanced or added to the 2009 CCM PI Dress Rehearsal instrument:

- ❑ Collecting, Displaying and Selecting Addresses
- ❑ Output Processing
  Roster Collections and Roster Review

  Navigating to the Last Question Asked From a Previous Session

2. Background

The CCM PI is conducted by personal visit and telephone using a computer-assisted data collection instrument on a laptop computer. The purpose of the CCM PI is to obtain person and address information about current residents of the sample housing unit, residents who may have moved since Census Day, and residents who may have moved out of the sample housing unit between Census Day and the time the CCM PI interview is conducted. The instrument collects demographic information for each person in the sample housing unit including name, gender, age, date of birth, race, and Hispanic origin. Information is also collected to determine where each current resident was living on Census Day and where each resident who has moved out since Census Day currently lives. Lastly, the instrument also collects information to determine if there are any other addresses where residents may have been counted on Census Day and other information necessary to geocode their alternate addresses. This data will then be compared to the decennial data in order to measure the accuracy of the person count of the 2010 Census.

In addition to developing the CCM PI instrument, a CCM Re-Interview (RI) instrument was created for verifying how the PI was conducted and will actually run the full PI if needed.

The CCM PI survey sample size is extremely large and will be implemented by Field Representatives (FRs) that have been temporarily hired to conduct this interview. Therefore, the premise for some of the CCM PI requirements is that the FRs collecting data do not have the usual experience, understanding, and training of current CAPI FRs. To this end, some of the requirements requested and implemented are ones that are not normally allowed in our current survey instruments.

Examples of these types of requirements are:

- ❑ Allowing empty responses for questions that should be answered
- ❑ Helping the FR with navigation upon re-entry
- ❑ Forcing forward navigation and locking complete sections that would alter the data collection if changed.

3. Collecting, Displaying, and Selecting Addresses

One of the more challenging requirements for the 2006 CCM PI instrument was to collect, display all collected addresses, and allow the FR to select an address from a list of up to 30 addresses per case. Each unique address entered for the various household members was added to an external ASCII file. The contents of this file were presented to the FR in the form of an answer list so the FR could select a previously entered address. When completing the address collection portions of the instrument, the FR either selected one from this list or added a new address. This same list of addresses was constantly updated and kept unique throughout the various sections in the instrument.

In the CCM PI instrument, an address is composed of seven components: house number, street name, unit designation, city, state, zip code, and country. Each unique address can be shared across different types of residencies throughout various sections of the instrument. In general, the FR can select an address previously entered in that section or another section for the same case. If the FR selects "Add New Address" at the Select Address question then all components will be available for editing. Otherwise, if the FR selects a previously entered address from the answer list (which displays all unique addresses entered for the case) the information displayed is transferred to each address component and then left as "show" only.

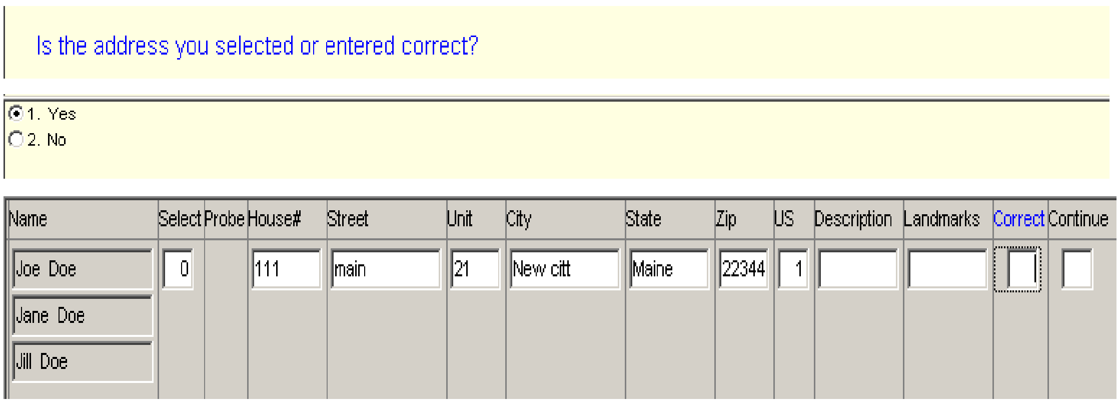
3.1 How Address Collection was accomplished in 2006

The 2006 CCM instrument attached an alien router written in Delphi that was exclusively built to write the address components collected through the Data Entry Program to an external file. Each unique address was written to an ASCII file by this alien router. In order for the alien router to work properly, the 2006 CCM instrument implemented a concept called gates. The gates were extra questions that triggered some events inside the alien router. The entry and exit gates were placed around the address components.   Some times there were two gates one after another; the purpose of these gates was to turn on and off the writing of the external file and avoid infinite writing of the same address.

The addition of these gates presented a challenge to the FR of extra, unnecessary, keystrokes needed when collecting data. This was unacceptable from the survey Methodologists perspective; however, the use of gates was considered unavoidable and left in place for 2006.

The graphic below demonstrates one type of gating utilized. The "Correct" questions locked the address information entered in the components. The locking was accomplished by setting flags in the section to disable the address information from editing.

In order to modify the entries made in any of the address information the FR had to change the value of the "Correct" variable and manually navigate to where the change was needed.

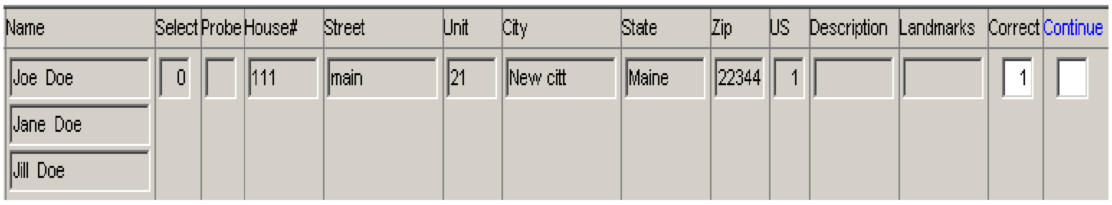


Is the address you selected or entered correct?

[✓ 1. Yes]
2. No

Name	Select	Probe	House#	Street	Unit	City	State	Zip	US	Description	Landmarks	Correct	Continue
Joe Doe	0		111	main	21	New citt	Maine	22344	1				
Jane Doe													
Jill Doe													

The "Continue" question enabled the next row if the next resident had an address in this section; otherwise the FR was taken to the next section.



Name	Select	Probe	House#	Street	Unit	City	State	Zip	US	Description	Landmarks	Correct	Continue
Joe Doe	0		111	main	21	New citt	Maine	22344	1			1	
Jane Doe													
Jill Doe													

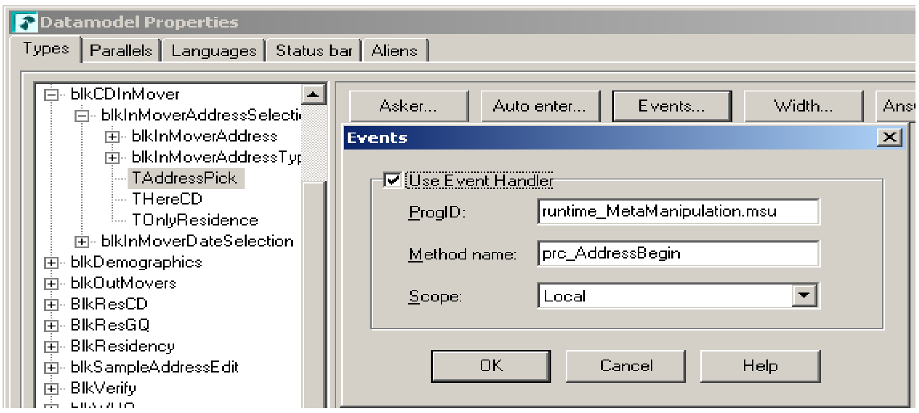
3.2 How Address Collection was implemented in 2009 using Blaise 4.8

For the new 2009 instrument, our sponsor wanted to make address collection more efficient and user-friendly by eliminating the extra questions added for gating the address components. This imposed a series of challenges to the authoring team. First, the original developers that worked with the Delphi application were no longer in the Division and secondly, our team no longer had a copy of this software.

Since Blaise 4.8 offers a robust capability to allow the Data Entry Program to interact with Manipula, a decision was made to migrate the original code written in Delphi to Manipula. This approach proved to be successful and straightforward, obtaining the same results as the original approach plus the additional new requirements requested by our sponsors. They requested that the 2009 CCM PI instrument filter addresses based on the section in the instrument and compare addresses for possible duplicates among other addresses already entered in the various sections of the instrument.

The CCM PI is composed of about twenty-five address sections, each of which collects the same type of address components. This similarity made the implementation of the Manipula script possible. In order to implement the changes, the team removed the four references to the DLL Delphi code and the extra gate fields. We then applied the new call to a Manipula script as an alien procedure. To do this, three questions were mapped to make the call and replace the function of the gates from the original instrument. The first gate was replaced in the "Select Address" question with Datatype TAddressPick, the second in the "probe" question with Datatype TyesNoProbeAddress and the third in the last address component "Country" question with Datatype TCountryLocation.

The graphic below is an example of how to attach to a Manipula script via the data model properties and associate to a data type used in the Answer List for selecting an address.



```
⊟ blkCDInMover
   ⊟ blkInMoverAddressSelecti
      ⊞ blkInMoverAddress
      ⊞ blkInMoverAddressTyp
        TAddressPick
        THereCD
        TOnlyResidence
   ⊞ blkInMoverDateSelection
⊞ blkDemographics
⊞ blkOutMovers
⊞ BlkResCD
⊞ BlkResGQ
⊞ BlkResidency
⊞ blkSampleAddressEdit
⊞ BlkVerify
```

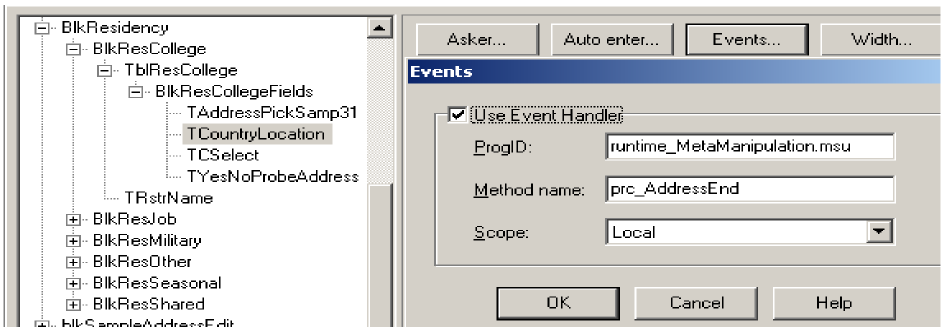
☑ Use Event Handler

ProgID: runtime_MetaManipulation.msu

Method name: prc_AddressBegin

Scope: Local

The same concept was used for writing to an ASCII file in the Country question as shown below.



BlkResidency
BlkResCollege
TblResCollege
BlkResCollegeFields
TAddressPickSamp31
TCountryLocation
TCSelect
TYesNoProbeAddress
TRstrName
BlkResJob
BlkResMilitary
BlkResOther
BlkResSeasonal
BlkResShared

Events

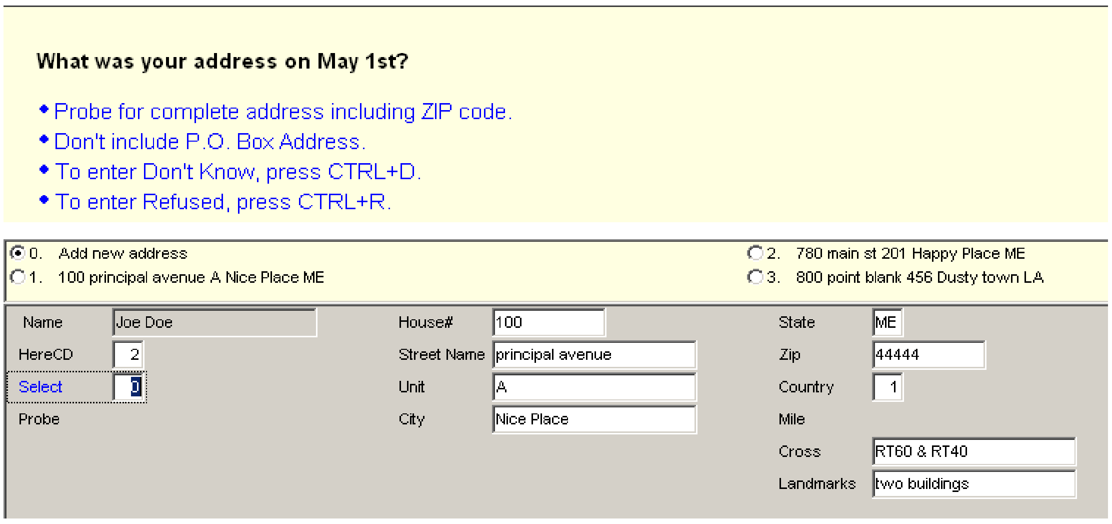
[✓] Use Event Handler

ProgID: runtime_MetaManipulation.msu

Method name: prc_AddressEnd

Scope: Local

The following screen demonstrates the new layout and style for collecting addresses without extra keystrokes or gates. Instead of the 2006 approach of displaying the address components in a table, our sponsor requested to only display one address at a time. An appropriate form pane had to be chosen so that the FR had a full view of the address items to be edited. Note that the answer list area expands as more addresses are listed or collected.



What was your address on May 1st?

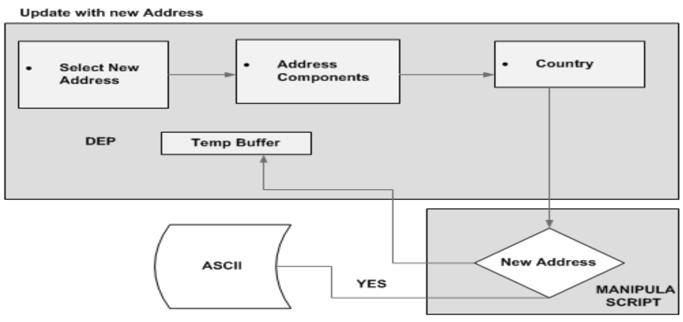
- Probe for complete address including ZIP code.
- Don't include P.O. Box Address.
- To enter Don't Know, press CTRL+D.
- To enter Refused, press CTRL+R.

● 0. Add new address
○ 1. 100 principal avenue A Nice Place ME
○ 2. 780 main st 201 Happy Place ME
○ 3. 800 point blank 456 Dusty town LA

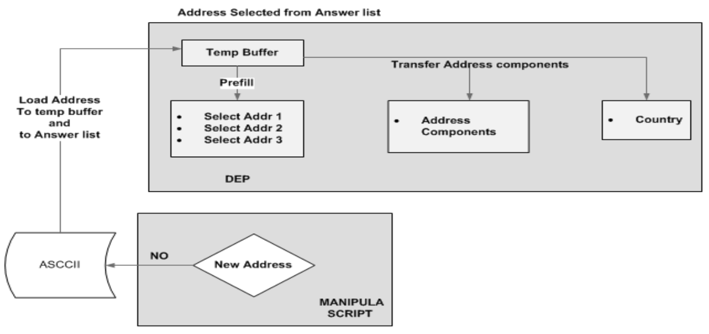
Name	Joe Doe	House#	100	State	ME
HereCD	2	Street Name	principal avenue	Zip	44444
Select	0	Unit	A	Country	1
Probe		City	Nice Place	Mile	
				Cross	RT60 & RT40
				Landmarks	two buildings

The general process of collecting addresses using a Manipula script to store to an external file and to retrieve the address data is as follows:

- As the FR lands on the “Select Address” question, a Manipula script populates a temp buffer variable from the external ASCII file and sets the answer list displaying the previous addresses collected.
- As soon as the FR leaves the “Select Address”, “Probe”, or “Country” question, the handler is passed to the Manipula script. The script will acknowledge the request as either “adding a new address” or “selecting an address already displayed”.
- When “adding a new address”, the script will verify the value from the “Select Address” question and the “Country” question. The Manipula script will compare this address against other addresses in that ASCII file. If there is a match the script does not append a new entry to the external file, otherwise the address components are written out and the temp buffer variable is updated for subsequent residents.



- When selecting an address already displayed, assignments are made from the temp buffer to the address components.



4. Output Processing

Our usual survey sponsors request either the Blaise blobs and conduct their own output processing or they request an ASCII Relational or ASCII output dump and use SAS to process the output data. Our CCM sponsor is not familiar with Blaise blobs, so they preferred a slightly different ASCII output.

4.1 How Output was processed for 2006

In 2006, the instrument output was an extremely long space delimited ASCII file; the dump came out in the order the fields were defined in the metafile. Since the instrument allowed for collection of data for 49 persons and up to 30 addresses in different sections, the output contained spaces reserved for all variables, thus the file totaled more than 1,000,000 characters per case.

Since the VAX machine used in 2006 could not deal with stream output records this long the output was split between variables. Manipula script accomplished this process of splitting it and cameleon was customized to follow the same concept as Manipula in order to produce the SAS script. There were no specified split locations but per case each of the forty-four output records averaged about 30,000 characters in length. The backend team processing this data had to determine where splits were and reorganize the data accordingly. This was obviously very challenging for the data processors to work with.

4.2 How Output is processed for 2009

Based on the output issues encountered back in 2006, a new output method was requested. The ASCII dump option was eliminated since the instrument was even larger for 2009 and because of the difficulty in linking information. The use of an ASCII Relational output was suggested and then eliminated because of the number of files to manage and the extra cleaning to be performed. Also, the current structure of this metafile was too convoluted to make drastic changes. Consequently, creating a customized output script was the only solution to this dilemma.

For the 2009 CCM instrument, the authoring team decided to utilize Blaise 4.8 new capabilities to create a Manipula script to output data in a well known format record typed used by the Census Bureau. One concern with using this approach was that we would need a second customized script to run output for the CCM PI RI instrument.

This approach undoubtedly imposed a concern in maintenance and coordination for any changes done to either instrument or the script to correct data issues.

The script was built from scratch. The following generic guidelines were considered for this ambitious approach:

- ❑ The script must be easy to maintain in the event of new enhancements
- ❑ The script must be able to read a file containing the information about the variables to be written for output and how (this file is also known as the layout file)
- ❑ The script must retrieve for data based on the layout file
- ❑ The script must generate a SAS script and by this we would not use cameleon to obtain information of the metafile since Manipula now is able to read it as well.
- ❑ The script must be generic enough so it can be used for both the CCM PI instrument and the CCM Re-Interview (RI) instrument using a different layout file.

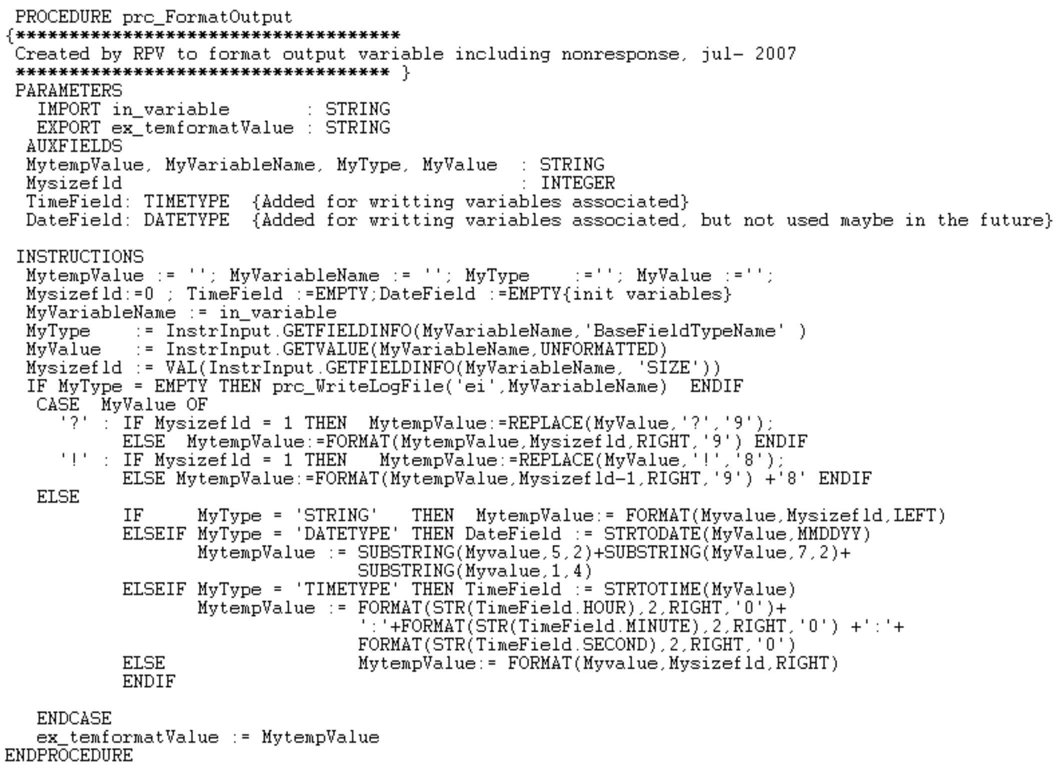
4.2.1 The script must be easy to maintain and allow for new enhancements

The customized script was modularized and broken into several procedures each one doing one specific task and properly parameterized and generic enough to become reusable.

The Procedure below identifies if the variable for output is an array or set of. The "in_search" parameter would indicate what to look for. In this case the presence of the character "**[**" indicates array and "**–**" indicates "set of".

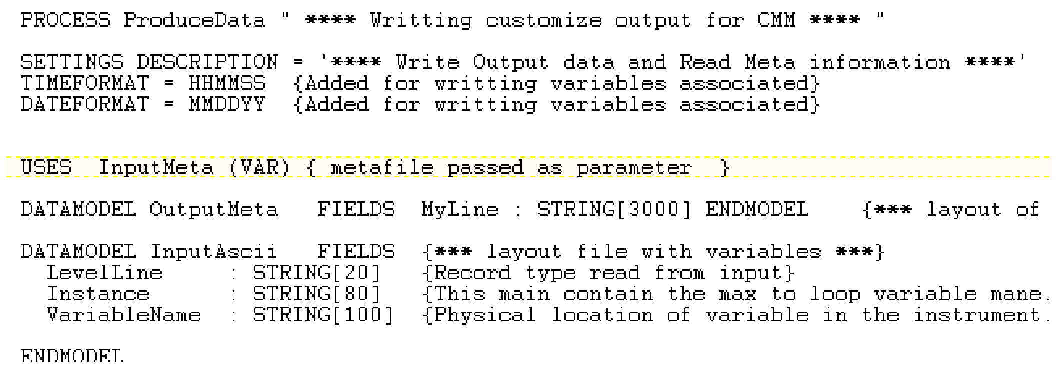
```
PROCEDURE prc_ParseTokenIfSet  {third procedure to hit and detect if the token is an arrayed
 PARAMETERS
   IMPORT inTokenfld : STRING  {token to be read}
   IMPORT inSearch   : STRING  {String to search on such as ; – @}
   EXPORT IsFound    : Integer
   EXPORT IndexRow   : STRING
   EXPORT LowRange   : STRING
   EXPORT HiRange    : STRING
AUXFIELDS
F,P : INTEGER

INSTRUCTIONS

     IF(POSITION(inSearch,inTokenfld)>0)THEN
        IndexRow := SUBSTRING(inTokenfld, (POSITION(inSearch,inTokenfld))-1,
                              (LEN(inTokenfld)-(POSITION(inSearch,inTokenfld)))+1 )
        LowRange  := SUBSTRING(IndexRow,1, (POSITION(inSearch,IndexRow))-1)
        HiRange   := SUBSTRING(IndexRow,(POSITION(inSearch,IndexRow))+1,len(IndexRow))
        IsFound   := 1
     ELSE
       IndexRow :=''; LowRange :=''; HiRange := '';   IsFound  := 2
     ENDIF
ENDPROCEDURE
```

The procedure below formats output values for the various data types. For example, it will correct the format for the nonresponse values, it will right justify numerical variables, and it will left justify alpha variables. This procedure will also check and format datetype or timetype for output.



```
 PROCEDURE prc_FormatOutput
{************************************
 Created by RPV to format output variable including nonresponse, jul- 2007
 *********************************** }
 PARAMETERS
   IMPORT in_variable       : STRING
   EXPORT ex_temformatValue : STRING
  AUXFIELDS
  MytempValue, MyVariableName, MyType, MyValue  : STRING
  Mysizefld                                     : INTEGER
  TimeField: TIMETYPE  {Added for writting variables associated}
  DateField: DATETYPE  {Added for writting variables associated, but not used maybe in the future}

 INSTRUCTIONS
  MytempValue := ''; MyVariableName := ''; MyType    :=''; MyValue :='';
  Mysizefld:=0 ; TimeField :=EMPTY;DateField :=EMPTY{init variables}
  MyVariableName := in_variable
  MyType    := InstrInput.GETFIELDINFO(MyVariableName,'BaseFieldTypeName' )
  MyValue   := InstrInput.GETVALUE(MyVariableName,UNFORMATTED)
  Mysizefld := VAL(InstrInput.GETFIELDINFO(MyVariableName, 'SIZE'))
  IF MyType = EMPTY THEN prc_WriteLogFile('ei',MyVariableName)  ENDIF
   CASE  MyValue OF
     '?' : IF Mysizefld = 1 THEN  MytempValue:=REPLACE(MyValue,'?','9');
           ELSE  MytempValue:=FORMAT(MytempValue,Mysizefld,RIGHT,'9') ENDIF
     '!' : IF Mysizefld = 1 THEN   MytempValue:=REPLACE(MyValue,'!','8');
           ELSE MytempValue:=FORMAT(MytempValue,Mysizefld-1,RIGHT,'9') +'8' ENDIF
   ELSE
           IF     MyType = 'STRING'   THEN  MytempValue:= FORMAT(Myvalue,Mysizefld,LEFT)
           ELSEIF MyType = 'DATETYPE' THEN DateField := STRTODATE(MyValue,MMDDYY)
                  MytempValue := SUBSTRING(Myvalue,5,2)+SUBSTRING(MyValue,7,2)+
                                 SUBSTRING(Myvalue,1,4)
           ELSEIF MyType = 'TIMETYPE' THEN TimeField := STRTOTIME(MyValue)
                  MytempValue := FORMAT(STR(TimeField.HOUR),2,RIGHT,'0')+
                                 ':'+FORMAT(STR(TimeField.MINUTE),2,RIGHT,'0') +':'+
                                 FORMAT(STR(TimeField.SECOND),2,RIGHT,'0')
           ELSE                  MytempValue:= FORMAT(Myvalue,Mysizefld,RIGHT)
           ENDIF

   ENDCASE
   ex_temformatValue := MytempValue
ENDPROCEDURE
```

Below we demonstrate the new feature in Blaise 4.8, passing the metafile as parameter (late binding), so we may no longer need to recompile the script; this is a powerful enhancement made to the software.



```
PROCESS ProduceData " **** Writting customize output for CMM **** "

SETTINGS DESCRIPTION = '**** Write Output data and Read Meta information ****'
TIMEFORMAT = HHMMSS  {Added for writting variables associated}
DATEFORMAT = MMDDYY  {Added for writting variables associated}


USES  InputMeta (VAR) { metafile passed as parameter  }

DATAMODEL OutputMeta   FIELDS  MyLine : STRING[3000] ENDMODEL    {*** layout of

DATAMODEL InputAscii   FIELDS  {*** layout file with variables ***}
  LevelLine     : STRING[20]   {Record type read from input}
  Instance      : STRING[80]   {This main contain the max to loop variable mane.
  VariableName  : STRING[100]  {Physical location of variable in the instrument.

ENDMODEL
```

Below we show under the Manipulate section the main entrance to the script. The script can extract data only, can create the SAS script only or if necessary can create both. More important, it is parameterized providing flexibility when running it.

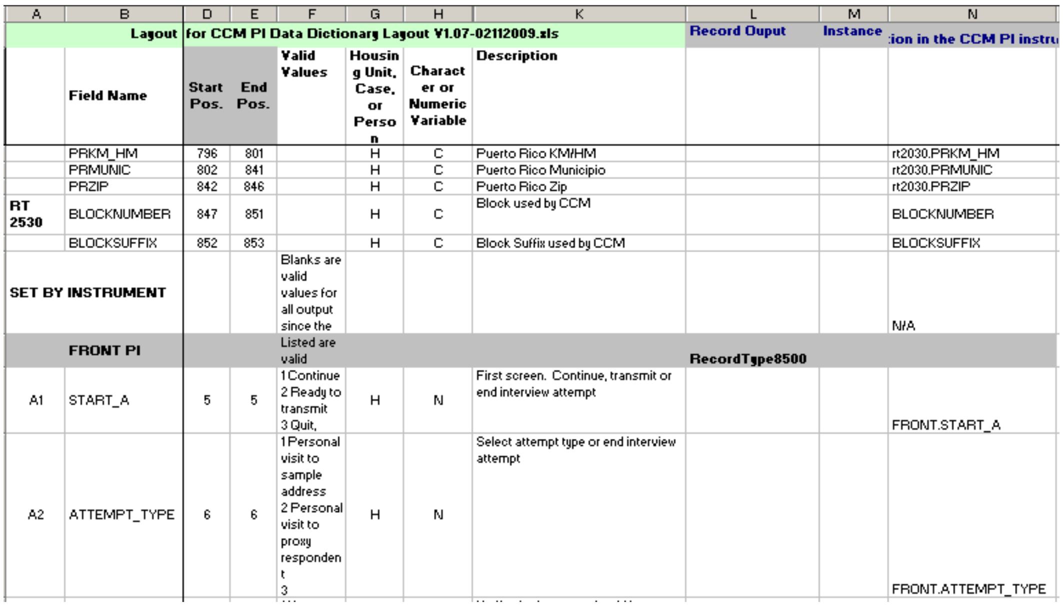
```
MANIPULATE

  typeProcess := PARAMETER(1) {'all'}
  IF PARAMETER(2) ='' THEN locationInput := 'MyData.inp'
  ELSE                     locationInput := PARAMETER (2)
  ENDIF
  IF PARAMETER(2) ='' THEN param_2 := 'NULL' ELSE param_2 :=  PARAMETER(2) ENDIF
  IF PARAMETER(1) ='' THEN param_1 := 'NULL' ELSE param_1 :=  PARAMETER(1) ENDIF

  prc_clearIt
  prc_Init
  CASE typeProcess OF
    'data': prc_InitDataDump('Start Custom output Only')
    'dict': prc_InitMetaDictionary('Start Custom Dictionary only')
    'all' : prc_InitDataDump('output and')
            prc_InitMetaDictionary('Dictionary')
  ELSE display('No parameters given! or wrong  order of parameters passed to this script!
               'This is what the script is acepting as parameters : param(1):'+param_1+' :
               'Contact Roberto Picha from TMO Authoring if more assistance is required.'
  ENDCASE
```

4.2.2 The script must read a file containing the information of the variables to be written for output

Our sponsors provided the team with a dictionary layout of all variables expected in output. The excel document had information such as variable name, length expected and expected values. Authoring added additional information to this excel file - including the physical location of the variable as defined in the metafile and whether or not the variable was "rostered" or "Set of". This file was then exported to an ASCII layout based on the layout defined for output.

The graphic below demonstrate the excel File layout that eventually will be converted into an ASCII TAB delimited. The last three columns from the excel dictionary were exported; this was a manual process preformed by copying the three columns from excel and pasting to a text editor and then saving it.

Excel Spreadsheet submitted by our sponsor.

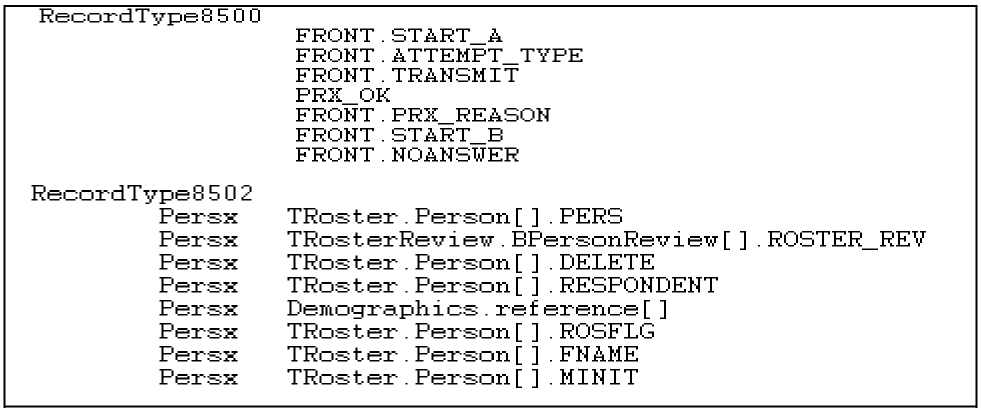


A	B	D	E	F	G	H	K	L	M	N
	Layout	for CCM PI Data Dictionary Layout V1.07-02112009.xls						Record Ouput	Instance	:ion in the CCM PI instru
	Field Name	Start Pos.	End Pos.	Valid Values	Housing Unit, Case, or Person	Character or Numeric Variable	Description			
	PRKM_HM	796	801		H	C	Puerto Rico KM/HM			rt2030.PRKM_HM
	PRMUNIC	802	841		H	C	Puerto Rico Municipio			rt2030.PRMUNIC
	PRZIP	842	846		H	C	Puerto Rico Zip			rt2030.PRZIP
RT 2530	BLOCKNUMBER	847	851		H	C	Block used by CCM			BLOCKNUMBER
	BLOCKSUFFIX	852	853		H	C	Block Suffix used by CCM			BLOCKSUFFIX
SET BY INSTRUMENT				Blanks are valid values for all output since the						N/A
	FRONT PI			Listed are valid				RecordType8500		
A1	START_A	5	5	1 Continue 2 Ready to transmit 3 Quit,	H	N	First screen. Continue, transmit or end interview attempt			FRONT.START_A
A2	ATTEMPT_TYPE	6	6	1 Personal visit to sample address 2 Personal visit to proxy respondent 3	H	N	Select attempt type or end interview attempt			FRONT.ATTEMPT_TYPE

The resulting ASCII layout file is shown below from the excel dictionary above.

Note that for rostered variables we used the " [ ]" to indicate script that we are dealing with arrays variables, "Persx" is a variable in the instrument and will instruct the Manipula script to loop as many lines as necessary.

Asccii Layout

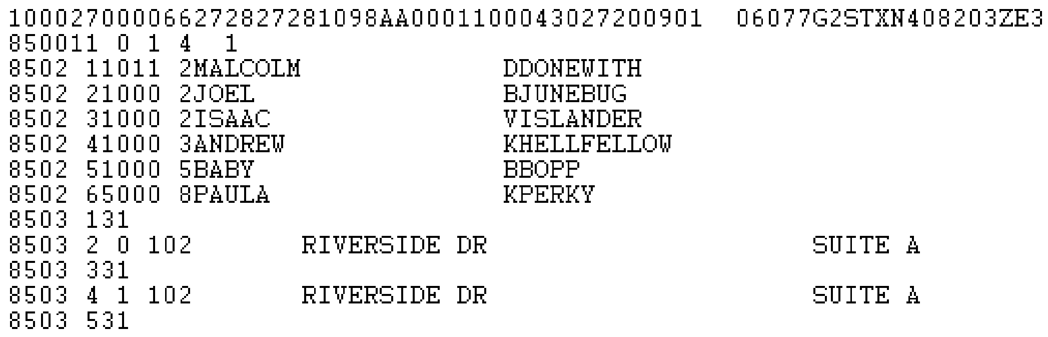


```
 RecordType8500
                   FRONT.START_A
                   FRONT.ATTEMPT_TYPE
                   FRONT.TRANSMIT
                   PRX_OK
                   FRONT.PRX_REASON
                   FRONT.START_B
                   FRONT.NOANSWER

RecordType8502
         Persx   TRoster.Person[].PERS
         Persx   TRosterReview.BPersonReview[].ROSTER_REV
         Persx   TRoster.Person[].DELETE
         Persx   TRoster.Person[].RESPONDENT
         Persx   Demographics.reference[]
         Persx   TRoster.Person[].ROSFLG
         Persx   TRoster.Person[].FNAME
         Persx   TRoster.Person[].MINIT
```

4.2.3 The script must retrieve data based on the layout file

Once the layout file was generated from the sponsor’s dictionary, the Manipula script retrieved items in the metafile using the GETFIELDINFO and GETTYPEINFO method. This was a critical step inside this script as using these methods allow us to format output for variables that where strings, integers, arrayed or variables defined as Set. Several checks where performed prior to final writing to maintaining data consistency.

The example below of the SCIF format generated by the customized output script.



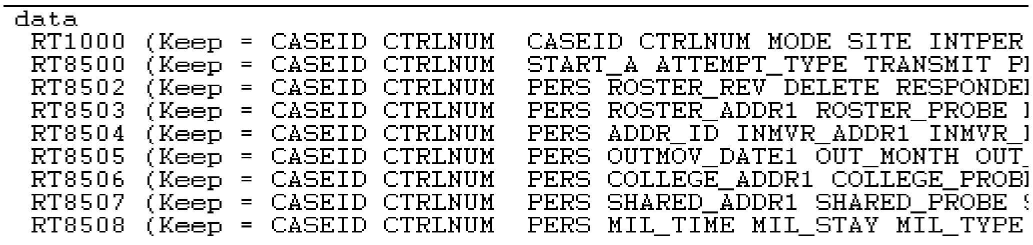
```
1000270000662728272810 98AA000110004302720 0901  06077G2STXN408203ZE3
850011 0 1 4  1
8502 11011 2MALCOLM            DDONEWITH
8502 21000 2JOEL               BJUNEBUG
8502 31000 2ISAAC              VISLANDER
8502 41000 3ANDREW             KHELLFELLOW
8502 51000 5BABY               BBOPP
8502 65000 8PAULA              KPERKY
8503 131
8503 2 0 102       RIVERSIDE DR                    SUITE A
8503 331
8503 4 1 102       RIVERSIDE DR                    SUITE A
8503 531
```

4.2.4 The Script must Generate a SAS script

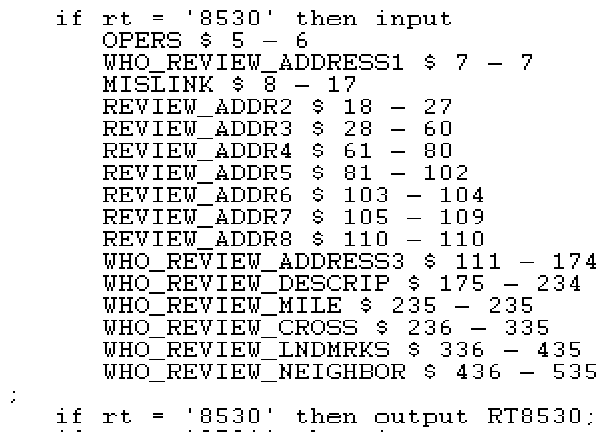
Since our sponsor’s use SAS to process the data we wanted to provide them with the corresponding SAS code to help in their processing. The customized Manipula script was generic enough to expand and enhance further to generate the SAS script in the same manner as it was used when producing data. Again the methods GETFIELDINFO and GETTYPEINFO played a major role here in providing relevant information of each variable listed for output.

The example below shows the result of the SAS script generated by the customized output Manipula script.

Generated SAS Script.



```
data
 RT1000 (Keep = CASEID CTRLNUM  CASEID CTRLNUM MODE SITE INTPER
 RT8500 (Keep = CASEID CTRLNUM  START_A ATTEMPT_TYPE TRANSMIT P
 RT8502 (Keep = CASEID CTRLNUM  PERS ROSTER_REV DELETE RESPONDE
 RT8503 (Keep = CASEID CTRLNUM  PERS ROSTER_ADDR1 ROSTER_PROBE
 RT8504 (Keep = CASEID CTRLNUM  PERS ADDR_ID INMVR_ADDR1 INMVR_
 RT8505 (Keep = CASEID CTRLNUM  PERS OUTMOV_DATE1 OUT_MONTH OUT_
 RT8506 (Keep = CASEID CTRLNUM  PERS COLLEGE_ADDR1 COLLEGE_PROB
 RT8507 (Keep = CASEID CTRLNUM  PERS SHARED_ADDR1 SHARED_PROBE
 RT8508 (Keep = CASEID CTRLNUM  PERS MIL_TIME MIL_STAY MIL_TYPE
```



```
   if rt = '8530' then input
      OPERS $ 5 - 6
      WHO_REVIEW_ADDRESS1 $ 7 - 7
      MISLINK $ 8 - 17
      REVIEW_ADDR2 $ 18 - 27
      REVIEW_ADDR3 $ 28 - 60
      REVIEW_ADDR4 $ 61 - 80
      REVIEW_ADDR5 $ 81 - 102
      REVIEW_ADDR6 $ 103 - 104
      REVIEW_ADDR7 $ 105 - 109
      REVIEW_ADDR8 $ 110 - 110
      WHO_REVIEW_ADDRESS3 $ 111 - 174
      WHO_REVIEW_DESCRIP $ 175 - 234
      WHO_REVIEW_MILE $ 235 - 235
      WHO_REVIEW_CROSS $ 236 - 335
      WHO_REVIEW_LNDMRKS $ 336 - 435
      WHO_REVIEW_NEIGHBOR $ 436 - 535
;
   if rt = '8530' then output RT8530;
```

4.2.5 The script must be Generic Enough to be used for the CCM PI-RI Instrument.

Since the same data processing team is processing the CCM PI RI output data we wanted to use an identical approach for the Re-Interview instrument. In view of the fact that the two instruments were very similar, the concept to parameterize the script was in order. We were able to use the same script for both instruments and just modify the variable layouts (using the excel sheet) and generate two different SAS scripts. The Output process is as follows:

- Each survey sponsor creates and maintains an Excel file containing the data items they require.
- Authoring adds appropriate instrument variable names and locations to the Excel file.
- ASCII layout file is generated manually from Excel File and delivered to data processors.
- Manipula script is run against the layout file to produce SAS scripts.
- Manipula script uses the ASCII layout file and is run against the consolidated Blaise database to generate customized record typed output

Considering the generic approach of this script, there is a prospective tendency to take advantage of using layout files against new instruments developed by the authoring team. Preliminary testing have assured that the script used for CCMPI can be slightly modified and used as another alternative output option by the Bureau.

5. Roster Collections and Roster Review.

Collecting names of residents with different status can be simple and straightforward, however, spec writers and subject matter people can have a different point of view in collecting rosters. The results can turn into the most unusual way to program rosters.

5.1 How Rostering worked in 2006

Due to the complexity of requirements in 2006, roster collection had to be performed in five separate rosters, each one collecting names of individuals living at a sample address. For each roster category, individuals were assigned a specific status flag stored in the field “RosFlag”.

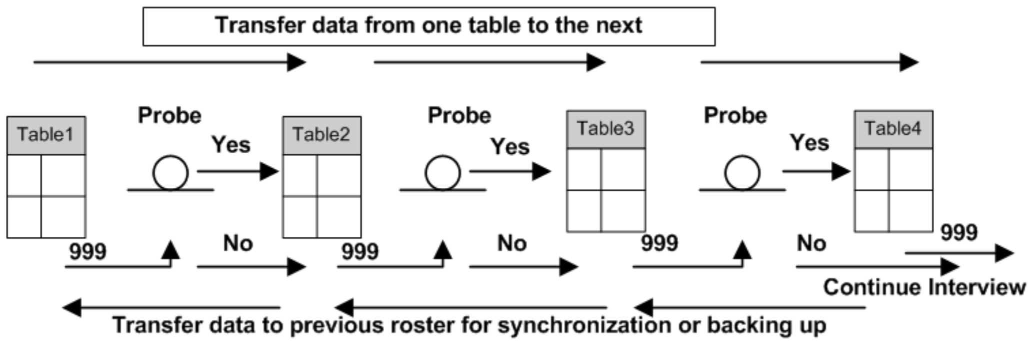
The approach turned out to be convoluted. Though it worked as expected the overhead involved caused some lagging in navigation from question to question. The team suggested that only one roster could have been used for this purpose. However, there was resistance from our sponsors to making changes to the instrument.

The first roster was asked for all individuals who are living at the sample address. Then subsequent probe questions were asked for:

- Individuals who stay at the sample address often,
- Individuals who are staying at the household until they find another place to live,
- Children who were not mentioned yet, and
- Any relatives staying at the sample address.

These five rosters were not combined into one table but they had to be in sync. They remained in separate rosters even in output, which resulted in duplicate data.

The graphic below demonstrates the process of collecting rosters in 2006. This approach required a great deal of code to synchronize all rosters at all times. Flags were required to turn off and on the rules based on the FR manipulation in the Data Entry Program.


Transfer data from one table to the next
Table1
Probe
Yes
Table2
Probe
Yes
Table3
Probe
Yes
Table4
999
No
999
No
999
No
999
Continue Interview
Transfer data to previous roster for synchronization or backing up

5.2 How Rostering worked in 2009 using Blaise 4.8

As part of the new specifications, the 2009 CCM instrument added additional rosters and a final Roster Review where the FR verified, corrected, and modified all names entered in the prior rosters.

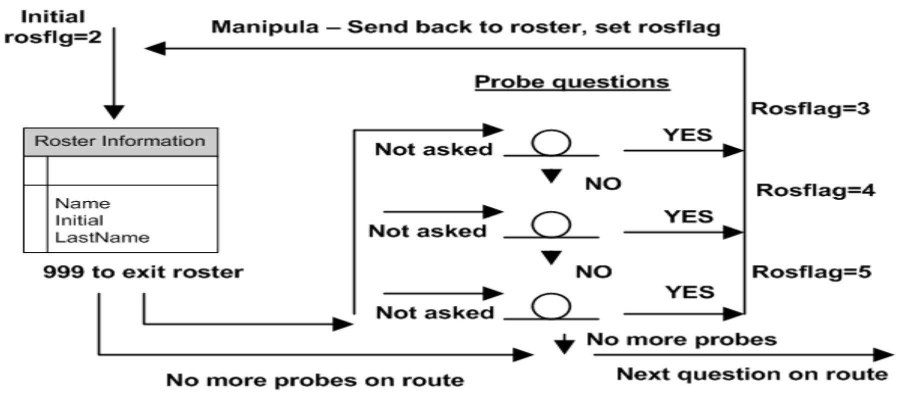
The original code was fairly complex and imposed some issues in situations of backing up to previous rosters. Furthermore, having the rosters synchronized at all times from within the Data Entry Program left an overhead that had to be controlled and simplified.

The authoring team decided that for simplicity there should be only one roster where all names and proper status flag (RosFlag) are stored.

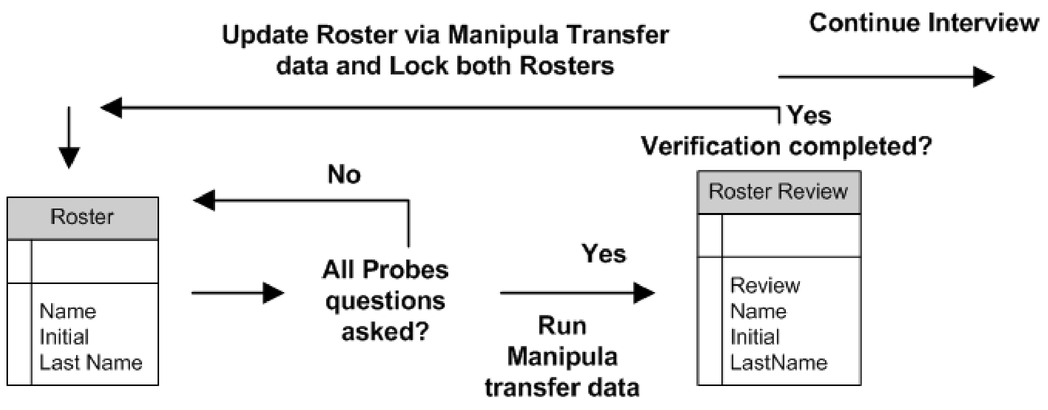
The same roster is also used to store relevant demographics information. This in contrast to the 2006 instrument where data information about individuals was scattered throughout the instrument, making data linkage difficult for analysis.

In order to accomplish the goal, the instrument will navigate back to the single rosterso ti can be updated with missing information when the probe questions are answered with a “yes” response. The “RosFlag” is updated with the appropriate probe code. The probe questions were attached to a Manipula script to handle the setup of “RosFlag” when returning to the roster. Once the table is exited via the 999 token, the instrument asks the subsequent probe question or continues to Roster Review.

The graphic below demonstrates the flow of the Roster Collection with the probe questions flowing back to the roster via to the attached Manipula script.



The graphic below demonstrates the interaction between the Roster and Roster Review. Once the roster is completed via the probes or the “no more people” 999 token, data is transferred to the Roster Review by invoking Manipula through the rules as an alien procedure. The Roster Review process allows the FR to verify, add, and/or update any information that was entered incorrectly. After completing this verification, the updated information is copied back to the Roster via Manipula. The roster and roster review are then put off path so that the roster information cannot be changed again.


Update Roster via Manipula Transfer data and Lock both Rosters
Continue Interview
Yes
Verification completed?
No
Roster
Name
Initial
Last Name
All Probes questions asked?
Yes
Run Manipula transfer data
Roster Review
Review
Name
Initial
LastName

6. Navigating to the Last Question Asked From a Previous Session

One of the new requirements requested by our sponsors for CCM PI 2009 was to have the instrument automatically take the FR to the question they were last on when they previously exited the case (for partial interviews).

Initially the authoring team suggested the use of the "End" key since it is available in the DEP by default; the purpose is similar to the requirement and it is standard function key used in other Census surveys. The "End" key by default goes to the next unanswered question that is not empty enable and meets the universe logic. However the End key does not fully accomplish the request because many of the address components attributes allow for empty and therefore the "End" key could potentially skip over some questions so they may never get asked. Since the CCM interviewers are not experienced FRs, the sponsor believed that this could lead to possible confusion when re-entering partial cases.

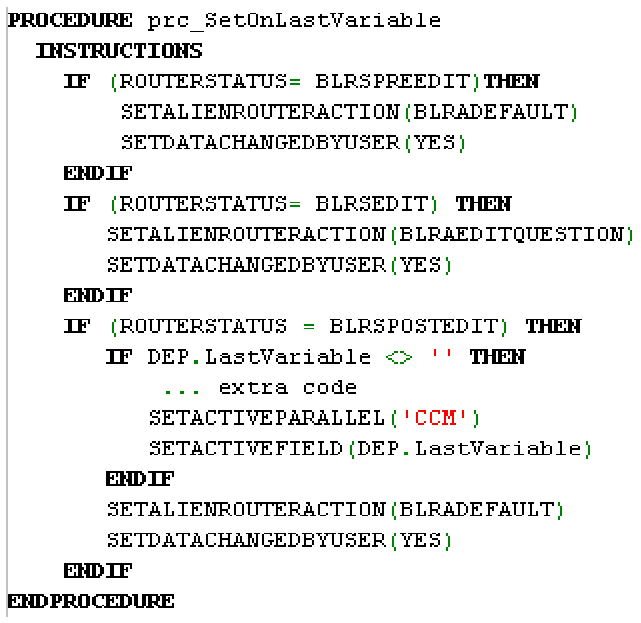
In order to accomplish this new requirement, the authoring team used a Manipula script.

The script consists of two procedures and works as follows:

- As soon as the F10 key (exit to back) is pressed, the path location is captured by the Manipula script using the ACTIVEFIELD method and stored into a field called "LastVariable". "LastVariable" only gets captured when the FR presses the F10 key from the main path of the instrument and the Roster Review is completed. Capturing the last variable does not take place if the user invokes the F10 key from any parallel block in the instrument. The focus is then returned back to the Data Entry Program to resume exit.

```
PROCEDURE prc_LastVariable
 INSTRUCTIONS
  IF ACTIVEPARALLEL = 'CCM' THEN
    IF (DEP.Gate_TRoster = 1) OR
       (DEP.WHOutmovers.Gate_TRoster = 1) THEN
        IF (substring(ACTIVEFIELD,1,6) <> 'Front.') AND
           (substring(ACTIVEFIELD,1,5) <> 'Back.') AND
           (substring(ACTIVEFIELD,1,6) <> 'bBack.') THEN
            DEP.LastVariable := ACTIVEFIELD
        ENDIF
    ENDIF
  ELSE
     DEP.LastVariable :=''
  ENDIF
ENDPROCEDURE
```

- Once a checkpoint is passed in the re-entered case, Manipula is used to retrieve the content of the “LastVariable” field and sets the focus back to the DEP using the SETACTIVEFIELD function.



```
PROCEDURE prc_SetOnLastVariable
  INSTRUCTIONS
    IF (ROUTERSTATUS= BLRSPREEDIT)THEN
        SETALIENROUTERACTION(BLRADEFAULT)
        SETDATACHANGEDBYUSER(YES)
    ENDIF
    IF (ROUTERSTATUS= BLRSEDIT) THEN
       SETALIENROUTERACTION(BLRAEDITQUESTION)
       SETDATACHANGEDBYUSER(YES)
    ENDIF
    IF (ROUTERSTATUS = BLRSPOSTEDIT) THEN
       IF DEP.LastVariable <> '' THEN
           ... extra code
          SETACTIVEPARALLEL('CCM')
          SETACTIVEFIELD(DEP.LastVariable)
       ENDIF
       SETALIENROUTERACTION(BLRADEFAULT)
       SETDATACHANGEDBYUSER(YES)
    ENDIF
ENDPROCEDURE
```

7. Conclusion

By using the new capabilities of Blaise 4.8, the TMO Authoring team was able to quickly modify and add enhancements to the CCM PI instrument to meet some of the challenging new requirements for 2009.

We were able to improve the FR/instrument interface for collecting address components to make it smooth, practical, and more efficient by using Manipula with the DEP. The Roster collection and review was made more efficient and cleaner for output purposes. Some user enhancements, such as the navigation to the last question, were also helpful.

However, the most important enhancement for this survey was modifying the way output was created for the data processors. This process not only makes the data processor jobs easier, it also allowed the subject matter specialists to quickly review their data. During early testing, subject matter specialist observed several issues with the output - such as missing variables from the data dictionary and getting extra data that was not expected. These issues were quickly identified and corrected. The new output process was crucial in the success of this project due to the tight turn around with testing the instrument, systems, and reviewing their output data. By quickly identifying data issues, instrument corrections were made in time for later tests and production. The dictionary and layout file seemed to play a big role in defining the dataset expected in output. Also, the size of the output data file per case was significantly reduced compared with the output in 2006, allowing more insightful data analysis.

8. References

Blaise for windows Developer’s guide

9. Acknowledgements

The author would like to acknowledge the following people for input into this paper and input work on this project.

Robert Wallace, U.S. Census Bureau
Jason Arata, U.S. Census Bureau
Diane Cronkite, U.S. Census Bureau
Susanne Frank, U.S. Census Bureau
Shadana Myers, U.S. Census Bureau