Using the BCP for Creating SAS Datasets and Codebooks

Lilia Filippenko, Mai Nguyen, and Chris Carson, RTI International

1. Introduction

Blaise Component Pack (BCP) allows us to create applications that read meta information from the Blaise meta file. This paper describes a .Net application (BlzToSAS) developed at RTI International which uses BCP to prepare SAS datasets easily and efficiently for any Blaise instrument regardless of its complexity and size.

The application uses an initialization file and creates a few intermediate files along with a driver program for creation of the SAS dataset. These files can be used during the development phase of the instrument to examine test data and during the production phase to prepare deliverable SAS datasets. The initialization file is created dynamically and holds a number of settings such as maximum length of characters allowed in variable names and variable labels, requirements for renaming Blaise variables, and many others. The application is used to check how variable names from the Blaise database will be presented in the SAS dataset and helps with preparation of codebooks.

2. Background

The initial version of this application was developed in Microsoft Visual Basic 6.0 when one of our studies used a Blaise instrument with almost 35 thousand fields. RTI's customized Cameleon script was already used successfully for many studies but didn't work properly with that instrument. We decided to use Blaise API to create a simple application that would be easy to use and easy to adapt later for other studies. The application was created in a very short amount of time and worked very well to produce SAS programs to create datasets. Some ideas are still used in the application and many new features have been added to it during the last nine years. A few years ago it was converted to VB.Net and uses the latest version of the Blaise API. The application is used for all studies at RTI International conducting CAPI interviews in Blaise.

3. Instrument Development

3.1 Blaise Instrument Requirements

Some standards are established at RTI for developing Blaise instruments to simplify the process of creating SAS datasets by using the BlzToSAS application:

- **Field Description** is used to specify a label for a field in the SAS dataset or to define a field as not needed in the delivery file. The latter is achieved by using key word "NoDeliv".
- **Field Tag** is used to define a variable name in the SAS dataset. If a field is defined as an array or a set, then an index will be added to the variable name.
- **User Defined Type** name should not allow a number as the last character due to SAS limitations.

Hatteras™, RTI's Web-based CAI system, is used to speed-up development of Blaise instruments by generating files defining Blaise fields. Figure 1 below shows how expected field name and labels in SAS dataset are entered in the Hatteras™ SurveyEditor.

Figure 1. Entering SAS Field Name and Label in Hatteras™ SurveyEditor

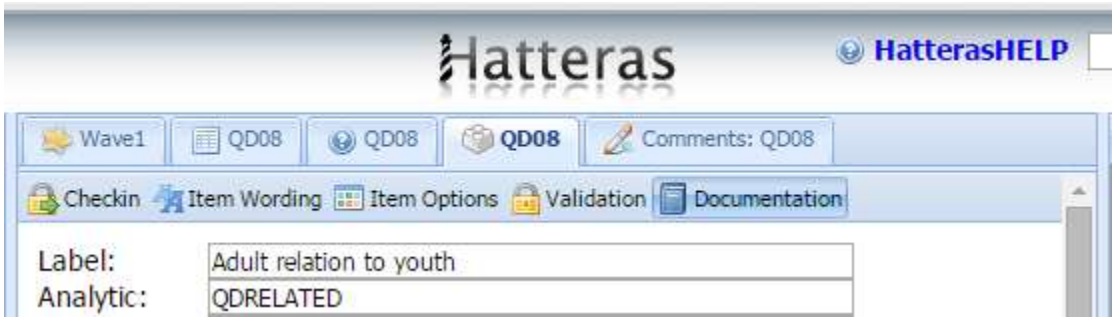


Figure 2. Example of Field Definition in Blaise Instrument

```
QD08 (QDRELATED)
ENUS
"HAND R SHOWCARD QD08.@/@/How are you related to ^prld.c_fname?"
/"Adult relation to youth"
: TREL
```

3.2. Initialization file

The application uses an initialization file (INI) to hold a few necessary settings in order to prepare the correct SAS input statements. If the name of the INI file is the same as the Blaise instrument name and exists in the folder where the BlzToSAS.exe is, it will be used by default. Otherwise, the name of the INI file should be provided interactively in a pop-up window.

There are two sections in the INI file. A section with [Project] name is required to specify common settings available for all studies. Section [Rename] is optional.

3.2.1 Common settings

Some common settings in the [Project] section of the INI file are required. Some can be omitted if the default values are acceptable.

- **ProjectID** – optional. It is needed if custom code is added to the application for the project.
- **Folder –** optional. Name of the folder where output files will be created, it could be entered interactively in a pop-up window before output files are created.
- **LabelLength –** required. If no label text is assigned to a question, then the first LabelLength characters of the question text are used as a label.
- **FieldNameLength –** required. The allowable length of field names varies by SAS version.
- **RemoveTabName –** optional. Default setting is "False". If it is set to "True", then table names will be removed from the field name by the program.
- **UseTypeNames –** optional. Default setting is "False" and it should be used if requirements to user defined types as described in 3.1 above are not in place. If the value is set to "True", then SAS formats will use the names as defined in Blaise instrument. This is very helpful for creating SAS datasets for longitudinal studies to maintain names for the same formats across different waves of data collection.
- **UseTag –** optional. Default setting is "False". If it is set to "True" and a tag exists for a field, then that tag will be used as a field name. For variable that is an array, the variable name of the array element will be created as a tag and "_" and the array index.
- **DK –** required. This value will be assigned to a variable if it has a value of "Don't Know".
- **RF –** required. This value will be assigned to a variable if it has a value of "Refusal".

- **FileLabels –** optional. Not all variables can have labels defined during development of the instrument. This option is used when a file with missing labels becomes available.
- **FileRenames –** optional. Initial statements to remove block names can be entered as wildcard instructions in a file specified here or in the INI file in [Rename] section.

Example of the [Project] section in INI file with options described above.

```
[Project]
ProjectID = AAAA
LabelLength = 80
FieldNameLength = 30
RemoveTabName = True
UseTypeNames = True
UseTag = True
FileRenames = C:\AAAA\Inst_renames.txt
DK = -1
RF = -2
```

3.2.2 Rename Variables

Blaise instruments with embedded blocks and tables would have very long full field names and they cannot be used by SAS. A local Blaise field name is not always unique and can be defined as an array or a set. There are two ways that the BlzToSAS application can create output variable names suitable for SAS. Both of these options can be used in the same instrument:

- **Field Tag** is specified for a variable during development of the instrument.
- **Rename statements** are wildcard instructions specified in the INI file.

In addition, the following rules are used by default:

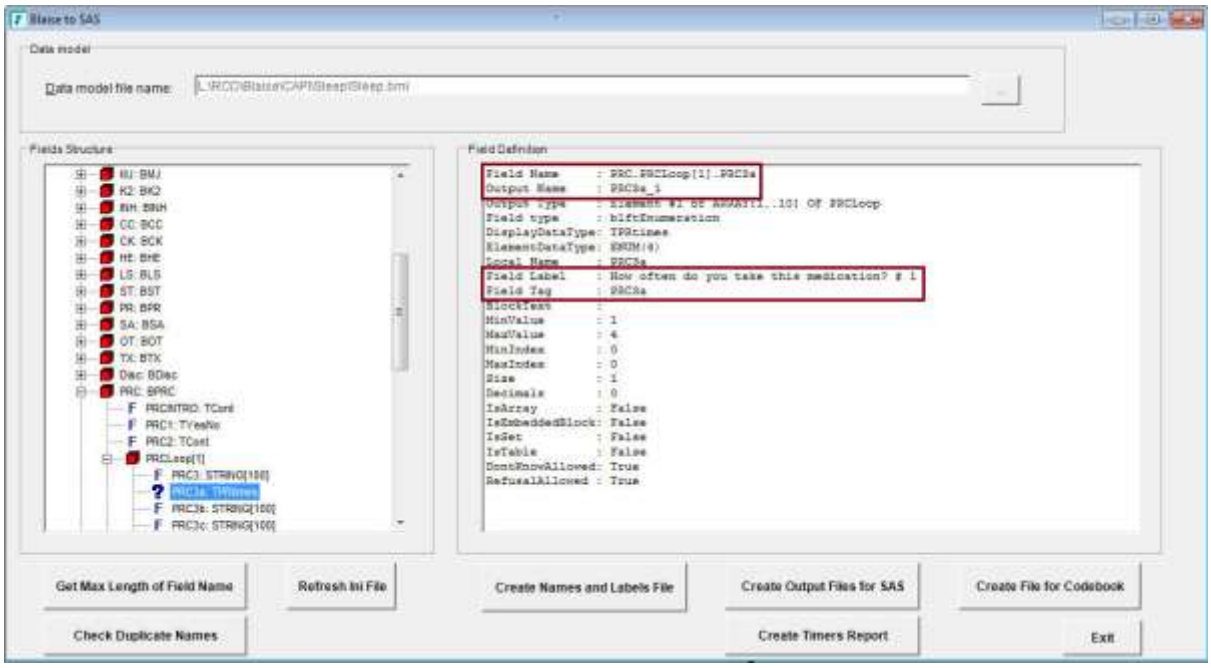
- Index representation for arrays of blocks is changed from "[i]." to "_i_".
- Index for an array or a set field is changed from "[i]" to "_i" at the end of the output variable name.
- Dots between block names in output variable name are replaced by "_".

Rename statements can be part of the [Rename] section in the INI file or can be entered in a standalone text file specified as "FileRenames" in the [Project] section. Use of the file with rename statements is more flexible and can start with the names of top level blocks that usually are not needed as part of the output variable name. For example, statement like (HHRoster.=) is used by the BlzToSAS application to create output variable names for all fields in "HHRoster" block without the block name. Other refinements can be added as the initial output is examined for possible improvements.

3.3 Review Variable and Type Names

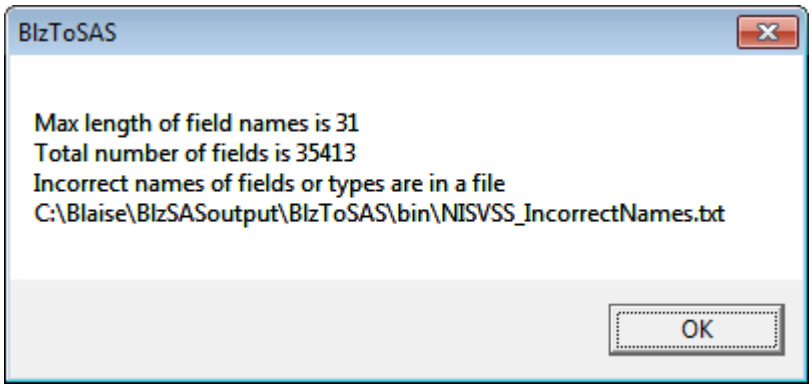
When the BlzToSAS application is launched and meta file of the instrument, along with the initialization file, are selected and loaded. The structure browser of the data model is displayed on the left panel and properties of any selected field in the tree are displayed on the right panel of the form. In addition to standard Blaise properties of an elementary field, "Output Name" and "Label" are presented as they will be defined in SAS dataset.

Figure 3. Blaise to SAS Form



The button “Get Max Length of Field Name” is used to examine the maximum length of output variable names and type names depending on the settings defined in the INI file. If there are some fields with lengths that exceed the allowable number of characters in expected output name, they are added to a file for review along with type names that cannot be used by SAS. It also shows the total number of fields in the Blaise instrument that will be outputted in an ASCII file. Depending on the available version of SAS, this can be used to decide if SAS dataset can be created for all of them.

Figure 4. Examine Blaise Instrument



After the file with long names has been reviewed, changes can be made to the INI file. Then the button “Refresh Ini file” is used to update settings used by the application. Now Blaise fields with long names can be selected in the tree and new output names will be displayed for them. Usually renaming is added for a block and the change appears for all fields in that block.

During this renaming process some of variables could be assigned a non-unique output name. The button “Check Duplicate Names” is used to find such variables and then the issue can be fixed by changing rename statements or in the Blaise instrument itself. If unacceptable type names are found, they can also be fixed in the instrument.

A few iterations of revisions to the INI file and the Blaise instrument might be needed before all problems are resolved to have variable and type names acceptable for SAS.

3.4 Review Labels

Labels for the SAS dataset can be defined in the field descriptions in the Blaise instrument during development if they are known at that time. Alternatively, labels can be revised by researchers who work with SAS datasets. An option was added to the application to prepare a file for them to review default labels for every variable in the Blaise instrument. The button "Create Names and Labels File" is used to prepare an Excel file with the following columns:

- Module name.
- Blaise full name of a variable.
- Blaise local variable name as it was defined in specifications.
- Output variable name for SAS.
- Default label (field description or specified number of characters from question text).
- Question text in default language.

When the spreadsheet is reviewed by researchers, they prepare an updated Excel spreadsheet with only two columns - output variable name and new label for that variable. This file is specified in the INI file and will be used by the BlzToSAS application to prepare labels statements for the output files.

3.5 Output files

To create the programs needed to create a SAS dataset from a Blaise database, two main programs are created by invoking button "Create Output Files for SAS":

- Manipula setup to create an ASCII file.
- SAS program with include statements to create a SAS dataset from the ASCII file. Each include statement pulls in a major component to the SAS program such as label statements, format statements, and format associations with variables.

3.5.1 Manipula Setup

All fields from the Blaise database are exported by the Manipula setup generated by BlzToSAS. To simplify the process of recoding "Don't Know" and "Refusal" to desired values specified in the INI file, code has been added to the setup for "string" fields allowing these special values.

Example of Manipula Setup.

```
SETTINGS
  DESCRIPTION = 'BLAISE to ASCII'
  INPUTPATH  = '\\...\DemoInst'
  OUTPUTPATH  = '\\....\BlzToSAS\ DemoInst '
USES
  Meta1  '\\...\ DemoInst\ DemoInst.bmi'
INPUTFILE  InpFile: Meta1 (' DemoInst ', BLAISE)
  SETTINGS
    ACCESS = SHARED
OUTPUTFILE OutFile: Meta1 (' DemoInst.asc', ASCII)

MANIPULATE
    IF InpFile.RV1.NOTES = DK THEN OutFile.RV1.NOTES := '-1' ENDIF
    IF InpFile.RV1.NOTES = RF THEN OutFile.RV1.NOTES := '-2' ENDIF

    OutFile.WRITE
```

3.5.2 SAS Program

The SAS program created by the BlzToSAS application utilizes include files to pull in input statements, labels, formats, format library references, and other options.

Example of SAS program:

```
libname  library '\\....\BlzToSAS\DemoInst';
libname  out '\\....\BlzToSAS\DemoInst';
TITLE 'DemoInst';
%include '\\...\BlzToSAS\DemoInst\DemoInst_procfmt.inc';
DATA out.DemoInst;
INFILE '\\...\BlzToSAS\DemoInst\DemoInst.asc' LRECL = 324290 TRUNCOVER;
%include '\\...\BlzToSAS\DemoInst\DemoInst_input.inc';
%include '\\...\BlzToSAS\DemoInst\DemoInst_toggles.inc';
%include '\\...\BlzToSAS\DemoInst\DemoInst_labels.inc';
%include '\\...\BlzToSAS\DemoInst\DemoInst_formats.inc';
%include '\\...\BlzToSAS\DemoInst\DemoInst_dk_rf.inc';
%include '\\...\BlzToSAS\DemoInst\DemoInst_drop.inc';
RUN;
```

Each "include" file in the program has its own purpose and can be extremely long depending on the number of variables in the SAS dataset:

- "**_procfrm**" – SAS proc formats for enumerated types defined in Blaise instrument with additional options for "Don't Know" and "Refusal" with values specified in the INI file.
- "**_input**" - Input SAS statements
- "**_toggles**" – SAS statements to define new recoded variables needed to convert "set" Blaise fields to toggle variables. The RTI convention for set questions is to recode them to a set of yes/no toggles and add the new toggle variables to the end of the dataset.
- "**_labels**" – statement to assign SAS variable labels foe all variables including newly created recoded variables.
- "**_formats**" – SAS statement to assign format to all variables
- "**_dk_rf**" – SAS statements to assign "Don't Know" and "Refusal" values for variables that were not changed by Manupila Setup in ASCII file.
- "**_drop**" – SAS statements to remove variables from the SAS dataset that have label "NoDeliv" in the Blaise instrument.

This SAS program can be run in a batch mode to create the SAS dataset from the ASCII file created after running Manipula setup. For Blaise instruments with thousands variables, the total number of lines in such a program becomes hundreds of thousands. When it is created by the BlzToSAS application, every variable that is expected to be present in the SAS dataset is there with the correct value, type, and label.

3.6 Timers Report

At RTI International, a standard procedure is used to calculate time spent during an interview in every section defined in the Blaise instrument. For interviews completed in more than one session, total time includes cumulative time for all sessions. The button "Create Timers Report" is used to create a SAS program that will produce the timers report.

Figure 5. Example of Timing Report

DemoInst Interview Timing Report

The MEANS Procedure

Variable	Label	N	Mean	Minimum	Maximum
Tmr_Interview_TimeTotal	Total Interview Time	18	99.42	15.75	187.85
Tmr_QD_TimeTotal	Core Demographics	18	2.53	1.26	6.32
Tmr_AS_TimeTotal	Child Cognitive Assessments	9	56.35	43.39	67.96
Tmr_RV1_TimeTotal	Rey Auditory-Verbal Learning Test (RAVLT 1)	9	1.16	0.51	2.00
Tmr_cal_TimeTotal	Calendar	9	6.94	5.43	9.63
Tmr_TUT_TimeTotal	ACASI Tutorial	8	5.83	1.92	13.63

The name of each timer section is defined as a field description for a timer section block.

4. Data Delivery

The application BlzToSAS is also used to help prepare final SAS datasets and a codebook after data collection for a study is completed. In addition, if special types of SAS datasets are needed, study specific buttons can be made available on the form.

4.1 Final SAS Dataset and Codebook

For data delivery, a codebook is often required to accompany each SAS dataset. The BlzToSAS application can create a simple codebook directly from the Blaise database by using the Blaise API. Alternatively, it can produce input for a more elaborate, SAS based, codebook generation system RTI has developed. This generic "Codebook Generator" is used for other types of instruments as well, including Hatteras™ Web and others packages.

The button "Create File for Codebook" is used to create a spreadsheet with metadata containing complete list of variables and the corresponding labels, format names, and question texts.

The SAS dataset produced by the BlzToSAS application, discussed in detail in section 3.5, usually needs to be altered for these situations to prepare final SAS datasets:

- Remove personally identifiable information (PII) data
- Remove variables used for intermediate analysis during data collection
- Add imputed or derived variables

Using the metadata spreadsheet, project staff can add additional information not available from the Blaise instrument or revise the labels and question texts as they might contain formatters and/or fill variables. Project staff would also decide which variables to keep or drop in the final dataset. Example of the spreadsheet is shown in figure 6 below.

Figure 6. Metadata Spreadsheet

	A	B	C	D	E	F	G
1	SAS_Variable_Name	Length	Var_Order	Label_of_Variable	SAS_Forma	Drop_Keep	QuestionText
2	MC_zrid	8	1	Case ID	$ch2fmt	Keep	Case ID
3	MC_sum_evt	4	2	Current Event Code		drop	Current Event Code
4	MC_sum_evtxt	15	3	Description of current event code		drop	Description of current event code
5	MC_sum_stat	4	4	Summary Status of case	$statscr	Keep	Summary Status of case
6	MC_sum_sttxt	15	5	Summary Status of case - text	$ch2fmt	Keep	Summary Status of case - text

Once the review process is completed, the SAS dataset and metadata spreadsheet become the input to the codebook generation process consisting of a set of SAS programs. The main steps in this process are:

1. Read spreadsheet metadata
2. Generate SAS scripts to
    a. Retain/drop variables
    b. Rename variables
    c. Assign labels and formats
3. Read SAS dataset, apply the above scripts to create the final dataset
4. Generate dataset contents and frequencies from the final dataset
5. Create codebook document

The codebook uses HTML structure and is styled using cascading style sheet (CSS). It can be delivered as is or brought into Word for additional editing, such as adding headers, footers, or creating 2-column format. Example of a sample codebook is shown in figure 7 below.

Figure 7. Codebook Example

Screener Codebook

September 9, 2013

Variable:	Person_1_SCQ_11
Description:	Is HH member 1 eligible
Stem:	HH MEMBER
Notes:	

Label	Value	Frequency	Percent	Cumulative Frequency	Cumulative Percent
Legitimate skip	.	3	1.52%	3	1.52%
YES	1	195	98.48%	198	100.00%
DELETE	4	0	0.00%	198	100.00%
Dontknow	-1	0	0.00%	198	100.00%
Refusal	-2	0	0.00%	198	100.00%

Variable:	Person_2_ageR
Description:	Age of person 2
Stem:	How old is "firstname"? ENTER 0 FOR LESS THAN 1 YEAR
Notes:	

Label	Value	Frequency	Percent	Cumulative Frequency	Cumulative Percent
Legitimate skip	.	58	29.29%	58	29.29%
Data present		137	69.19%	195	98.48%
Dontknow	-1	2	1.01%	197	99.49%
Refused	-2	1	0.51%	198	100.00%

4.2 Special SAS Datasets

The standard SAS dataset has one record per respondent. For studies collecting information about household members or people with whom respondents interact, the same questions are asked in loops. The data is collected in tables or arrays and for analysis it can be grouped the same way as it was collected. For example, for one respondent there can be blocks of questions for each household member. Clients sometimes prefer reorganization of this type of data into separate relational datasets.

The BlzToSAS application uses BCP to simplify this process of creating relational SAS datasets when they are requested by the study. A list of common variables and blocks with the required data must be defined. Then custom code can be added to the application to process them. Standard procedures are used to define variable names, formats, labels, and input statements along with the SAS program. The time needed to add custom code to the application is significantly less than creating a comparible SAS program from scratch.

To create special SAS datasets, the Blaise database name should be provided to create two files with data – an Excel spreadsheet and an ASCII text file. The Excel spreadsheet can be used to easily check the output data.

5. Conclusion

SAS has become the standard for file deliveries at RTI International. Projects producing extremely large data models or data models with complex nested structures and/or relational structures benefit from the enhanced system provided by BlzToSAS for generating SAS datasets. RTI's adoption of

this system has paid off in efficiency gains and improvements in the quality and reliability of file deliveries. It has also been well integrated into our new codebook generation system, so that a visually appealing codebook document can accompany our file deliveries.

6. References

Blaise 4.8 API Reference Manual

Roger Schou, "Creating Code to Convert Blaise to ASCII for SAS Input (Using the Blaise API within Visual Basic)", 2003, Proceedings of 8th International Blaise Users Conference

Barbara S. Bibb & R. Suresh, "Automatic Customization of Datasets Using Cameleon", 2004, Proceedings of 9th International Blaise Users Conference