# Output Processing for Consumer Expenditure Survey


## Xiaodong Guan, U.S. Census Bureau
## Howard R McGowan, U.S. Census Bureau


### Introduction

The Consumer Expenditure (CE) Survey is a large and complex survey conducted by the U.S. Census Bureau for the Bureau of Labor Statistics (BLS). The survey is being converted from paper to Computer-Assisted Personal Interview (CAPI) in April 2003. A CAPI instrument has been developed using Blaise and several field tests conducted. From the paper survey, the Census Bureau currently delivers 68 SAS data sets containing approximately 10,000 variables to the BLS. The deliverable from the CAPI output will be similar to the paper output.

The CE data model contains hundreds of blocks and thousands of fields. There are many types of data relationships in the CE data model. Additionally, the block structure and field names are constantly changed as the instrument is developed, tested and modified. The data output requires that we generate the SAS data sets for the selected blocks and fields from the CE data model. The data export system needs to be easily modified for changes in either the data model or output requirements.

We considered three output options.

- Use the Export ASCII data file option

  The Blaise system provides some tools to export data directly from the data model. Due to the size of the CE data model, the Export ASCII data file option does not work for the CE data model.

- Develop a custom-built Manipula output program for the CE data model

  Such a program would have to be written block by block, on a field by field basis and would contain hundreds or thousands of lines of code. It would be hard to develop and test. It also would be difficult to maintain.

- Generate ASCII relational data files for the data model and create SAS code for each ASCII data file.

  Hundreds of data files will be output from the CE data model. Unfortunately, the Blaise system does not come with a totally automated way to generate SAS data sets for the selected blocks and fields from the CE data model.

To accomplish the output requirement for the CE data model, we applied the third data output option. We developed an export system for the CE data model. . We modified Cameleon scripts and employed an output control file concept. The system is driven by an output control file developed and maintained by subject matter analysts. The system is flexible, efficient, and easy to maintain.

In this paper we describe in more detail the CE data model and data output requirements. We discuss and compare the data output methods in the Blaise system, and explain the reasons for our choice of method. We present the customized Cameleon scripts developed for the export system. We describe the concept and use of an output control file to select output blocks and fields. The paper concludes with a detailed description of the integrated data export system.

**CE data model and data export requirement**

Output processing is not a simple task for the CE CAPI survey data. It involves many aspects of the survey project, including the CE data model, output requirements, and other survey operation related issues. It is very important to choose the right tools to output data from the CE data model. In order to build a good data export processing system, it is critical that the developer understand every aspect of output processing. The following provides an overview of the CE data model and output requirement.

- **The CE data model**

  The CE data model was developed with Blaise software. It is a large and complex data model. It contains 270 blocks and 10,105 block instances. The CE data model produces over 250 ASCII relational output files. There is a total of 368,695 data fields with a total length of 13,853,483 bytes defined in the model. There are many different data types and relationships in the data model. The technical description for the CE data model is shown in tables 1 and 2. During the development stage of the data model, the block structure, block names and field names are constantly changed; for example, there were 255 blocks and 5,215 uniquely defined fields in the version 23 data model. But there were 296 blocks and 5,955 uniquely defined fields in the version 22 data model. Since the CE survey is a continuous survey, the data model will be released to the Field for data collection monthly. The block structure and field definition could be modified every month, especially, in the early stages of data collection. Because of this reality, it requires the data export system to handle a large, complex, and dynamic data model.

**Table 1. Block and data fields statistics in the CE data model**

| Overall Counts | Value |
|---|---|
| Number of uniquely defined fields *1 | 5,215 |
| Number of elementary fields *2 | 4,960 |
| Number of defined data fields *3 | 368,695 |
| Number of defined block fields *4 | 255 |
| Number of defined blocks | 270 |
| Number of embedded blocks | 14 |
| Number of block instances | 10,105 |
| Number of key fields | 1 |
| Number of defined answer categories | 1,207 |
| Total length of string fields | 13,079,645 |
| Total length of open fields | 0 |
| Total length of field texts | 191,668 |
| Total length of value texts | 56,187 |
| Number of stored signals and checks | 128,486 |
| Total number of signals and checks | 128,486 |

*1) All the fields defined in the FIELDS section
*2) All the fields defined in the FIELDS section, which are not of type BLOCK
*3) Number of fields in the data files (an array counts for more than one)
*4) Number of fields of type BLOCK

**Table 2. Data Type information in the CE data model**

| Data Fields | Number | Length |
|---|---|---|
| Integer | 147,339 | 635,940 |
| Real | 1,920 | 9,920 |
| Enumerated | 74,020 | 118,127 |
| Set | 5,500 | 9,555 |
| Classification | 0 | 0 |
| Datetype | 34 | 272 |
| Timetype | 3 | 24 |
| String | 139,879 | 13,079,645 |
| Open | 0 | 0 |
| Total in data model | 368,695 | 13,853,483 |

- **Output requirements**

  The data output requires the creation of about 70 SAS data sets, which contain over 1,800 variables, from the CE data model. All output data come from 90 blocks in the current data model. About 30 of the SAS data sets can be output from a single block. Other data sets need to be merged or manipulated from two or more blocks. Since the block and field name are not unique in the CE data model, the same block or field name can be used in many different sections. For example, the block "**brow"** and field "**amount"** are used in 20 different sections. Each one of the blocks produces a separate SAS data set. The field "**amount"** goes to different SAS data sets with a new SAS variable name. For instrument development reasons, the field names in the CE data model are often different than the SAS output variable names. For the purpose of data editing and analysis, the requirements may add or remove some variables from the current output. The CE output also requires exporting the timer information for each section in the data model. The data export system must be very flexible and efficient to handle the output requirements.

From the CE data model and output requirements, we see that the data export system will deal with a large, complex, and dynamic data model. It not only needs to generate the SAS data sets for selected blocks and fields from over 200 blocks and 5500 fields but also must easily handle changes either in data model or output requirements. Since the data output method is the cornerstone of the data export system, we will review the output method in the next section.

**The method selected for processing CE output**

The Blaise software provides three ways to output data from the data model in Version 4.4. The simplest and easiest way is the ASCII export. The ASCII export outputs data from the Blaise data model to an ASCII data file. This method is good for small sized data models. The length of the CE data model is 13,853,483 bytes. This method can not handle such a large data model. The CE data model is a highly structured data model. There are a lot of array blocks in the model. This method would not be suitable for the CE data model even if the method could handle the size of the CE data model.

The second way is to develop a customized Manipula output program. As [2] described, a custom-built Manipula output program must be done block by block, on a field by field basis and will contain hundreds

or thousands of lines of code. It is hard to develop, test and maintain this kind of program. Such custom-built output programs are less efficient during the output processing. It needs more time to connect the CE data model and the data output models. For example, it takes 16 minutes to output 300 cases from the CE data model to a special designed data output model. It only needs 3 minutes to output those 300 cases on the same desktop if we use the next method.

The third option is the ASCIIRELATIONAL export. This method can generate separate ASCII data files (ASCII relational data files), based on the data model block structure. For each ASCII relational data file, the Blaise system provides the Cameleon script to generate the data model associated with the ASCII file. Furthermore, the Blaise system can generate the SAS code based on the data model. As we described before, the CE data model is a large and complex data model. The CE data model is also highly structured, and the block structures are well defined to suit the data output requirements during the specifications developing phase. The third output option, export ASCII RELATIONAL data set, fits the output requirement of the CE data model compared to the other two methods. It gives us the basic steps to export the data from the CE data model. That is:

- Generate ASCII relational data sets from the CE database.
- Generate the SAS code for each data set.
- Run the SAS programs to create SAS data sets.

However, the output requires us to generate SAS data sets from the subset of blocks and fields in the CE data model. The export system can easily handle the changes in data model and output requirements. Unfortunately, the Blaise software does not come with a totally automated way to generate SAS data sets for selected ASCII relational data sets. There is considerable handwork involved if we use this method to export data from the CE data model. After we reviewed the Cameleon scripts provided by the Blaise software, the scripts did not provide enough information for the output in the CE data model. The output format of the scripts needed to be modified for an automatic data export system. Therefore, customized Cameleon scripts needed to be developed for our data export system.

**The Cameleon scripts for the data export of the CE data model**

The Blaise system provides many examples of Cameleon scripts. These examples can help users to generate a data model and create the code or script for other software. They also can obtain the information about the data model, such as field name and block number. These scripts needed to be modified to fit the data export requirement in the CE data model. For example, the Cameleon script outputs all data models for ASCII relational data sets to a big file. If we want to use some of the data models, we have to open the file, find the models, and cut and paste the models to separate files. For a large data model, such as the CE data model, it will be difficult and error prone work. For the output of the CE data model, we developed customized scripts, **ceasciirel, cedic** and **cesas,** which are modified from the scripts **asciirel, dic,** and **sas.** We modified these scripts to:

- Add some functions in the scripts to accomplish the requirement of the CE data model, such as, output more blocks or field related information for the CE data model.
- Remove some functions that are not needed for the output to reduce the chance to have errors during the batch processing.

The script **ceasciirel,** like **asciirel,** generates data models for ASCII relational data sets. It outputs each data model to a separate file. **Ceasciirel** also outputs the block information for the ASCII relational data sets. It connects the block name, block number, data model name, and ASCII data set name. The information can help us to generate batch files during the output processing. The following shows the details of the block information. From the information, we can easily determine the data model and ASCII

data set. If we need to create a SAS data set for block 26, we just use the script to generate SAS code for a26.bla with ceq_a1p1.a26 as an input file.

**ASCII relational block information**

| Sub Data Model | Block Name | Block Number | ASCII Data Set |
|---|---|---|---|
| A26.bla | Btabx | 26 | CEQ_a1p1.a26 |
| A71.bla | Bsect3i | 71 | CEQ_a1p1.a71 |
| A211.bla | Bback | 211 | CEQ_a1p1.c11 |

The **ceasciirel** provides the block information we need in the output processing. The script **cedic** generates the field-related information in the CE data model. The output of script **dic** lists every defined field (array field is counted more than once) in the data model. It is not suitable for the ASCII relational export method. The output is too large to use for the CE data model. The **cedic** only outputs uniquely defined fields. For each field, **cedic** outputs the field name, full name and block number for the field. The subject matter experts can use the information to identify the output fields. Since the output of **cedic** and **ceasciirel** include the block number, it will help us to identify the block to be output. For example, if CORRPRG is an output field, we can easily find out the sub-data model and ASCII data set for this field through the block number 71. An example of field information is:

**Field information**

| Field name | Block Number | Full Name |
|---|---|---|
| CORRPRG | 71 | Sect03.prop.sect3i.corrprg |
| Loantype | 58 | Sect03.sect3a1.curprop.loantype |
| Name | 42 | cc.unit.preson.name |
| Comb | 95 | Sect06.tablea.row.comb |

The script **sas,** provided by the Blaise system, generates the SAS code for the data model. It was developed for older versions of SAS software. The lengths of variable name are limited to 8 characters. The field name is used as the SAS variable name in the data model. For the CE data output, the export system should be able to pick the SAS variable name different from the field name. The **cesas,** modified from **sas**, adds these functions. The **cesas** also removes the **proc format** commands because the output of the CE data model does not need this function.

The three customized Cameleon scripts not only generate data model and SAS code for each ASCII relational data file but also provide more block and field information for the CE data model. This information is fundamental to developing the data export system. It becomes possible to output select blocks and fields automatically.

**The output control file for the data export of the CE data model**

There are about 270 data blocks and over 5,000 unique fields in the CE data model. Only 90 blocks and 1,800 fields will be output from the CE data model. It is very difficult to select the output blocks and fields by hand. We needed to find a way to select output blocks and fields automatically. The subject matter experts have developed a database to manage the CE project. The database includes much useful information for output processing. For example, it identifies all the fields that need to be output. The output file name (SAS data set name), is assigned to each output field name. For each output field in the database, if we can automatically identify its location in the CE data model, such as, full name or block number in the CE data model, it will help us to select the output blocks and fields. The field information produced by the script **cedic** can help us to accomplish this task. After merging the field information with

the control database, the subject matter experts can produce a data output control file or output specification. It contains the output field name, SAS data set name, and full name for each output field. An example of the control file is shown in table 3.

**Table 3. Example of data output control file**

| Blaise Field Name | SAS Data Set Name | Fullname1 | Fullname2 |
|---|---|---|---|
| Name | MEMB | cc.person.name | |
| Age | MEMB | Cc.person.age | |
| Apadesc | APA | Sect06.table.row.apadesc | Sect06.table.prechart.row.apadesc |
| Apaitem | APA | Sect06.table.row.apaitem | Sect06.table.prechart.row.apaitem |

Since the field name is not unique in the data model, there is more than one full name for some field names. From the output control file, we can produce two lists:

- **Output field list**

  For each output SAS data set, the SAS variables are listed in the output control file. We can create a variable keep list in the SAS code for every SAS data set. When the SAS data set is created, we only keep the variables in the variable keep list.

- **Output block list**

  First, we merge the output control file and the field information produced by script **cedic.** A new list is produced. The list includes field name, SAS data set name, block number for the field, and full name. (If the field is defined in more than one block in the data model, both block numbers will be listed in the file). The new list looks like:

| Field | SAS Dataset Name | Block Number | Full Name |
|---|---|---|---|
| Name | MEMB | 42 | Cc.person.name |
| Age | MEMB | 42 | cc.person.age |
| Apadesc | APA | 95 | Sect06.table.row.apadesc |
| Apadesc | APA | 94 | Sect06.table.prechart.apadesc |
| Apaitem | APA | 95 | Sect06.table.row.apaitem |
| Apaitem | APA | 94 | Sect06.table.prechart.apaitem |

  Second, we combine the above list with the block information produced by **ceasciirel**. It will create a list that contains the SAS data set name, ASCII relational data set name and data model name. We create an output block list. An example is:

| SAS Data Set Name | Block Number | Sub Data Model | ASCII Data Set Name |
|---|---|---|---|
| MEMB | 42 | A42.bla | CEQ_a1p1.a42 |
| APA | 95 | A95.bla | CEQ_a1p1.a95 |
| APA | 94 | A94.bla | CEQ_a1p1.a94 |
| APB | 95 | A95.bla | CEQ_a1p1.a95 |

From the output block list, we see the data for SAS data set MEMB come from block 42. The data for APA come from blocks 94 and 95. The data in block 95 will go to different data sets. For each block in the list, we can use script **cesas** to generate SAS code and create SAS data sets. The processing can be done automatically if a batch file is created.

The output control file, generated from the CE project database, is easy to update if there are any changes in the data model. For example, when a new CE data model is released, a subject matter expert can automatically combine the new output from **cedic** and their database to produce a new output control file. If anything is changed in the output requirement, the database will be updated. The output control file will include the new change. Thus, the data export system is driven by the output control file. This allows the export system to be flexible, efficient and easy to maintain.

**The data export system of the CE data model**

Implementing all the parts we discussed above to an export system is not just putting everything together. The order in which tasks are performed has a big impact on the efficiency of the system. Keeping the necessary middle processing information may reduce time to repeat running the whole system. We had to consider all different possibilities during the development of the export system. The following are what we have considered or experienced while developing the data export system:

- **Perform tasks between SAS and Blaise**

  The data export of the CE data model not only deals with a large, complex data model and special output requirements but also involves two data processing software, SAS and Blaise. Many tasks can be performed in either system. For any given task, we have to decide which software is best suited to perform that task. For example, we have two ways to keep the selected output fields. One way is to only keep selected fields when generating data models for the ASCII relational data sets. The advantage of this way is the system only carries the necessary data during processing. It reduces the size of files since the CE data model contains much duplicated information for display purposes. Some fields are extremely long. The disadvantage is that it requires us to develop more specific manipula programs or Cameleon scripts to accomplish it and needs more processing steps. Another way is to create a variable list of selected fields. When the SAS data set is created, we only want to keep selected fields and drop the rest of the fields. The unnecessary fields are kept until the last step. This is the easiest and simplest way to perform this task. After comparing the two methods during an early pretest, we chose to implement the second method for our export system.

  Some tasks can be performed in either software without much difference. We generally use SAS to perform those tasks because we have more experience in SAS programming.

- **Select method based on efficiency in the system**

  The Blaise system provides two ways to output ASCII relational data files. One is to output all ASCII relational blocks and another way is to output only selected blocks. In the output processing for the first two CE tests, we only output the selected blocks. We found that there is no big difference in performance between outputting selected blocks and all blocks. However, it took more time to process the data when we wanted to add more selected blocks later. We decided that output all blocks in the first step of the data export system.

Based on the control file and the customized Cameleon scripts, we developed the data export system for the CE data model. The system has two parts. The first part generates the SAS code for the selected output blocks and fields. This part can be executed every time to output the CE data. It may be executed

once and used for a while if the data model and output requirements are not changed. The export system includes the following steps:

- Generate field list from the data model.
- Generate the block list and data models associated with the ASCII relational data sets.
- Generate selected output field and block lists.
- Generate the batch files from the control list.
- Generate SAS code associated with the selected ASCII relational data model.

The programs, which are used and generated in this part, are listed in the table 4.

**Table 4. The programs used or generated in the data export system**

| Step | Program name | Purpose | Input | Output |
|---|---|---|---|---|
| 1 | Cedic.cif | Generate the field related information for the CE data model | The CE data model | An ASCII file, **CEDIC**, that includes all uniquely field name, full name. The block number for each field is also in the file. |
| 2 | Ceasciirel.cif | Generate data model for each ASCII data set and block information. | The CE data model | Over 200 Blaise data models |
| 3 | Cecontrol.sas | Create the output field and block lists | CEDIC and a control file provided by subject matter experts | Output field list for each SAS data set.<br>Output block list, **Blocklist**, for selected blocks.<br>**Ceoutput.sas** |
| 4 | Ce4batch.man | Create batch files to generate .bmi, .sas and sas data set in output block list | **Blocklist** | **Bla2bmi.bat**<br>**Bmi2sas.bat**<br>**Cesas.sas** |
| 5 | Bla2bmi.bat | Prepare the selected data model to generate *.bmi files | The selected data model, such as, a98.bla | *.bmi files for the selected data models (e.g., a98.bmi) |
| 6 | Bmi2sas.bat | Generate SAS code for each selected data model | *.bmi files for selected data model | *.sas file for selected data model, (e.g., a98.sas) |

The second part exports the data from the CE data model and generates the SAS data sets. The steps are:

- Generate ASCII relational data sets from the CE data model.
- Generate the SAS data sets for the selected blocks.
- Generate the final CE output.
- Generate timing information for each section in the CE data model.

The files used for this part are listed in table 5.

**Table 5. The files used or generated in the data export system**

| Step | Program name | Purpose | Input | Output |
|---|---|---|---|---|
| 1 | Ceascii.man | Generate ASCII relational data sets for the CE data model | The CE data collecting database and data model | Over 200 ASCII data files (e.g., CEQ.a98, CEQ.a99) |
| 2 | CESAS.sas | Output the SAS data sets for selected data model | The selected ASCII relational data file (e.g., CEQ.a98) | SAS data sets for selected data file (e.g., a98, a99) |
| 3 | Ceoutput.sas | Create SAS data set for the CE output | The SAS data set for selected output blocks (e.g., a98, a99) | SAS data set for the CE output (e.g., APA, APB, etc.) |
| 4 | Ceaduit.sas | Creat a SAS data set for the time information. | The audit trail file for each case | A SAS data set contains the time information for each section in the CE data model |

A **Winbatch** program links all the programs in the data export system. The system achieves our goal of exporting data from the CE data model and generating SAS data sets for the selected blocks and fields automatically. The system can run everything at one time. It can also be run separately depending on the need. In the SAS code development part, the system can produce field lists, output control lists, and SAS code separately. Users have the option to create the ASCII relational data sets and the SAS data sets at different times. In the data export system, we also include a program to create the timing information for each section in the CE data model. It reads time information for each field in the audit trail file and computes the total time for each section and creates a SAS data set.

The output control file plays the key role in the data export system. It drives the entire output processing. The advantages of this system are:

- **Flexibility**

  It can output any number of SAS data sets depending on how the output control file is created. For example, if subject matter experts just want to review one data set, they can create an output control file for the data set. It can also output any number of the fields. It is especially useful during the early stages of data output development and testing.

- **Efficiency**

  The system can export each stage's data separately. It avoids repeating the same output procedure if the change in the output control file does not effect the early stages of data output. It is very useful for large data sets. Most of our data processing is run on a Unix system during production. The data export system allows us to perform our tasks on the Unix system once the ASCII relational data sets are created in the microcomputer environment.

- **Easy to maintain and adapt to other surveys**

  The data export system is independent of any specific Blaise data model. The data model and output control files are the inputs to the system. The system itself does not need to be modified if there are changes in these inputs. The output can be obtained by re-running the system with a corrected output control file and data model. The system can be used to export data for other Blaise data models. The user would just have to identify the data model name and the control file name.

The goal of the data export system is to output data from the Blaise data model and to generate SAS data sets for selected blocks and fields automatically. We have discussed all the steps needed to achieve this goal. The data export system for the CE data model is based on three assumptions:

- The block structure of the CE data model corresponds well to the requirements of output data sets.
- The data model is well defined and organized.
- The output control file is easy to update and can drive the data export system.

The data export system for the CE data model can not be successful without these assumptions. The export system could not have been built without the work of the specification writers, data model developers, and subject matter experts.

**References**


1. Statistics Netherlands (1999) Blaise Developer's Guide

2. Pierzhzchala, Mark and Farrant , Graham (2000) "Helping non-Blaise Programmers to Specify a Blaise Instrument", Proceedings of the 6th International Blaise Users Conference

3. Frey, Richard (2000) "Developing Blaise instrument for the Spanish Bladder Cancer Survey", Proceedings of the 6th International Blaise Users Conference