Paper 30-28

'Watch Your Language!' Using SCL Lists to Store Vocabulary

Greg McLean, Project Leader, SAS Technology Centre
Statistics Canada, Ottawa, Ontario, Canada

ABSTRACT

Developing SAS® applications or programs almost always involves the use of spoken language or vocabulary. Whether it is text on a FRAME object (i.e. Pushbutton, List Box Title, etc.) or error messages written to the SAS LOG, we must deal with this issue. And quite often these applications or programs are either bilingual or multilingual.

The use of a "vocabulary dictionary" (one for each spoken language) can prove to be very useful and an efficient way to include vocabulary in your systems. This separation of vocabulary from the actual software is beneficial even if only one spoken language is used.

The focus of this paper is to provide a method to address the use of spoken language or vocabulary in SAS applications or programs, particularly those that are bilingual or multilingual. Discussion focuses on the use of SCL lists as a medium to store vocabulary. This concept of a "vocabulary dictionary" will be discussed in detail as well as a customized SAS/AF® application used to manage these external "vocabulary dictionaries" (SCL Lists).

INTRODUCTION

As software developers we often take for granted, or ignore the importance of the use of vocabulary in our applications, programs or systems. Without even thinking about it we are constantly embedding vocabulary in our SAS code, whether it be a title on a report, an error message to the user or text on a FRAME object (i.e. Pushbutton, List Box Title, etc.). The reason that most of us do not spend much time thinking about vocabulary when writing SAS code is that it is a simple task to incorporate. Text that we want displayed is simply placed in quotes and VOILA! This is definitely not "rocket science", however, there are some things that should be considered when incorporating vocabulary into medium to large systems.

This paper focuses on the use of SAS SCL (Source Component Language) lists and their functions to store and manipulate vocabulary. SAS SCL lists are very powerful and prove to be an efficient medium for storing vocabulary. To take this one step further, a SAS/AF application called **GULP** (**G**eneralized **U**tility for **L**anguage **P**rocessing) will also be introduced. This application was developed at Statistics Canada. Its purpose is to assist the developer, making it even easier to manage and maintain these SAS SCL vocabulary lists.

As you may have already realized, there is almost always several ways or methods of doing the same thing using SAS software. This is one of the great things about SAS. This paper will focus on one method of implementing vocabulary (SAS SCL Lists). The pros and cons of various other methods will also be addressed.

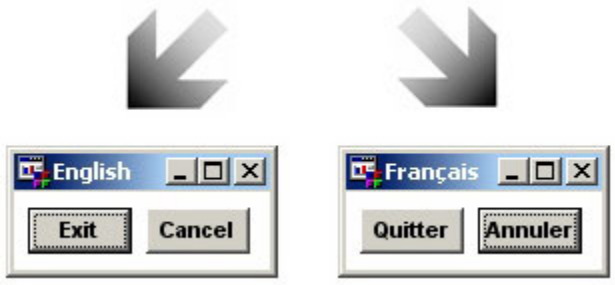
VOCABULARY DICTIONARY – THE CONCEPT

Before we get into the details of SAS SCL lists we will first discuss the concept of a "**vocabulary dictionary**". Traditionally, developers tend to place vocabulary directly in their SAS code. If there is more that one language, If-Then-Else statements are often used.

To best illustrate this would be to use an example. **Figure 1** is an excerpt of code from a SAS/AF application. This part of the SAS SCL code is responsible for placing vocabulary on pushbuttons on a dialog window based on the current language selected by the user. This is a bilingual application (French and English). Note that the vocabulary is "hard coded" within the SAS code.

Figure 1

```
Language:
   if (CurrentLanguage = "English") then do;
      Button1.Label = "Exit";
      Button2.Label = "Cancel";
   end;
   else if (CurrentLanguage = "French") then do;
      Button1.Label = "Quitter";
      Button2.Label = "Annuler";
   end;
return;
```

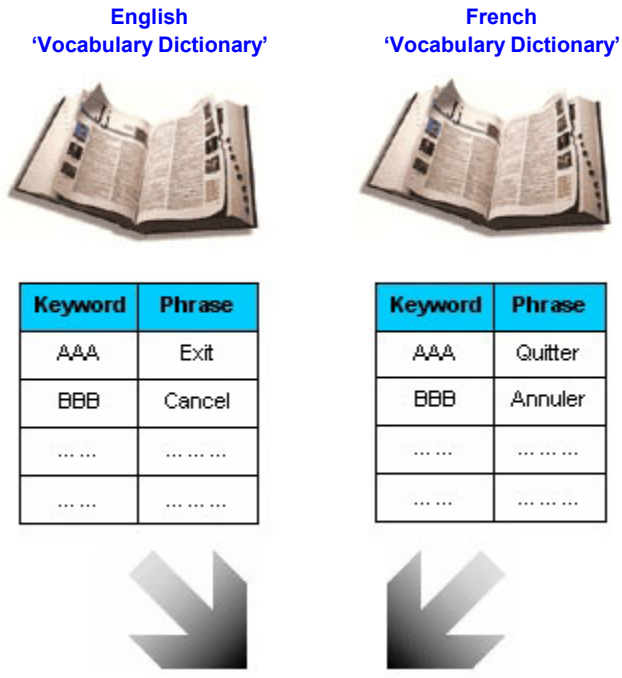


Example of "hard coded" vocabulary within SAS code.

Although this method works, a preferred approach is to separate the vocabulary from the SAS code. This would imply that all vocabulary be stored in some sort of external medium such as a SAS data set, SAS SCL list, SAS macro variable(s), flat file, etc. The external medium can then be thought of as a "**vocabulary dictionary**". This architecture would also require an interface between the SAS code and the external medium source.

Figure 2 illustrates the concept of separating vocabulary from the SAS code and storing it in a "**vocabulary dictionary**". Note that pseudo code is used within the SAS statements since the actual implementation has yet to be discussed.

Figure 2




```
Language:
   if (CurrentLanguage = "English") then do;
      Button1.Label = <Get Text From English
                       Vocabulary Dictionary
                       (Keyword  = AAA)>;
      Button2.Label = <Get Text From English
                       Vocabulary Dictionary
                       (Keyword = BBB)>;
   end;
   else if (CurrentLanguage = "French") then do;
      Button1.Label = <Get Text From French
                       Vocabulary Dictionary
                       (Keyword = AAA)>;
      Button2.Label = <Get Text From French
                       Vocabulary Dictionary
                       (Keyword = BBB)>;
   end;
return;
```

Conceptual illustration of separating vocabulary from SAS code.

When using a traditional dictionary, one would search or look-up a word to find the corresponding definition. When speaking of the concept of a "**vocabulary dictionary**" we are implying the look-up of vocabulary, phrases or text based on a keyword. A separate "**vocabulary dictionary**" would then exist for each language used in one's applications.

The common element between these "vocabulary dictionaries" is a **keyword**. The keyword is the means by which the SAS code interfaces with the desired phrase or vocabulary. It is then just a matter of ensuring that the SAS code is using the correct keyword and retrieving vocabulary from the desired "vocabulary dictionary".

In order to use such a method, a "vocabulary dictionary" must be selected before retrieving the desired phrase(s). The example from **Figure 2** could then be simplified. We no longer have to use if-then-else structures at every occurrence of vocabulary retrieval. Instead, we only need to determine which "vocabulary dictionary" to use. This is illustrated in the example in **Figure 3** (pseudo code).

Figure 3

```
Language:
   <Determine Vocabulary Dictionary To Use>;
   Button1.Label = <Get Phrase Corresponding To
                    Keyword AAA>;
   Button2.Label = <Get Phrase Corresponding To
                    Keyword BBB>;
return;
```

Conceptual illustration of selecting desired "vocabulary dictionary" before retrieving keyword / phrases.

VOCABULARY DICTIONARY – PROPERTIES

To summarize, the following is a list of properties that defines this concept of a "**vocabulary dictionary**":

- Separate Vocabulary Dictionary for each language used in an application
- Use of a **keyword** to identify each unique phrase
- Same set of keywords must exist in all vocabulary dictionaries used within an application

VOCABULARY DICTIONARY – ADVANTAGES

The following is a list of the advantages to using a "Vocabulary Dictionary":

- Required vocabulary changes need only be made in the vocabulary dictionary and not the SAS code (no need to recompile SAS code)
- Easier to list all phrases used in an application (i.e. to be sent to other departments for translation to other languages)
- Common phrases need only be defined once and not at every occurrence (i.e. "Exit" or "End")
- Other vocabulary dictionaries can be easily added (no change required to SAS code)

VOCABULARY DICTIONARY – IMPLEMENTATION

Now that we have discussed the concept of a "vocabulary dictionary" we must consider an actual implementation. There are several approaches that could be used to store keywords and corresponding phrases. In the following sections we will discuss a few of these, listing advantages and disadvantages of each. Please refer to SAS documentation for a more detailed explanation and description of these topics.

SAS MACROS

SAS Macros are dynamic variables that reside in system memory. Once a macro variable has been defined, one is able to easily retrieve its contents. The keyword would be the macro name and the contents would be the word or phrase. To simulate a "vocabulary dictionary", one would have to create a separate macro variable for each keyword / phrase for a selected language. Of course the keywords and phrases used to create the macro variables should be stored external to the SAS code (ie. SAS Data Set , Flat File, etc.)

Advantages

- Very efficient and fast retrieval of phrases since data resides in memory
- Each macro variable can contain up to 32K of data
- The number of macro variables and the size of their contents is only limited to memory size (Version 8)

Disadvantages

- Macro variables must be loaded or created at the beginning of your application, one by one
- The keyword names and phrases must be stored in some other medium before creating macro variables (i.e. SAS Data Set, Flat File, etc.)

SAS DATA SETS

A SAS data set is a viable medium to store the required keywords and phrases for a given application. A separate SAS data set would be created for each language. Each SAS data set would then contain two variables. One variable that would contain the keyword and a second variable that would contain the corresponding phrase. A SAS application could then open such a file, search for a keyword and retrieve the corresponding phrase.

Advantages

- Permanent and reside on a hard drive (no need to recreate at the start of an application)
- Easy to add, modify, or change the contents
- Most common data structure used in SAS

Disadvantages

- Slower access time since retrieving from disk.
- More steps required to retrieve desired keyword / phrase (OPEN, FETCH, GETVAR, etc. or WHERE clause)

Note:
It is possible to load a SAS data set into memory. This would improve access time for keywords / phrases. However, one would still require the same number of steps to do retrieval. To load a SAS data set into memory use the **SASFILE** command.

FLAT FILE

When we talk about a Flat File we are referring to a text-based file that can be used by many applications. Like a SAS data set, a flat file (text file) could also contain the keyword / phrase pairs. At the start of an application, the contents of the flat file would need to be loaded into another medium in which SAS could access more readily, such as a SAS data set or SAS macro variables.

Advantages

- Can be used by other applications, not just SAS
- Easier to translate to other languages (able to send flat file to a translator; no need to know SAS)
- Additions, modifications and deletions are easily made

Disadvantages

- Slower access time since retrieving from disk.
- More steps required to retrieve desired keyword / phrase (Create SAS Dataset, OPEN, FETCH, GETVAR, etc.)
- Some sort of SAS medium (i.e. SAS data set or SAS macros) must be created from the flat file each time an application runs

USER DEFINED FORMATS

It is not in the scope of this paper to discuss in detail the use of SAS User Defined Formats. However, in general we wish to address the possible use of such a method for storing vocabulary. These formats are created once and stored in a SAS catalog specified by the developer.

Advantages

- Permanent and reside on a hard drive (no need to recreate at the start of an application)
- Very simple syntax to use

Disadvantages

- More difficult to modify, add or delete keywords / phrases
- Slightly slower due to the fact that theses formats reside in catalogs on disk

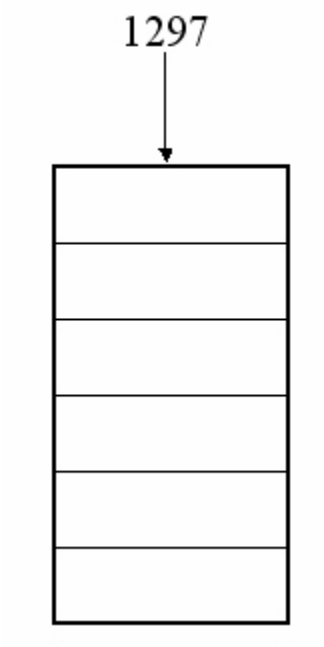
SAS SCL LISTS

Now that we have discussed alternate methods to store vocabulary, we will focus on the SAS SCL list. A quick lesson or review of SAS SCL lists will be beneficial to further our discussion with regards to vocabulary and SAS applications.

The SAS SCL list is a SAS data structure that resides in memory. Conceptually it resembles a list of "cells" linked together to form a chain. Each "cell" or item can contain character data, numeric data, or a list identifier for another SAS SCL list. SAS SCL lists are dynamic which means that they can grow or shrink in size. When first initialized, a list contains no cells, just a pointer into memory. This pointer is referred to as the **list identifier**. Using several SAS SCL list functions, one can then manipulate these lists, cells and their contents. The items in a list can be accessed by their name and/or their position in the list.

As indicated, it is possible to store a list identifier of another list in a "cell". This would be the foundation for multi-dimensional lists. However, for our discussion we will concentrate only on single dimensional lists as illustrated in **Figure 4**.

Figure 4



Conceptual view of a single dimensional SAS SCL list.

To make use of the 20 or so SAS SCL list functions, one must use the list identifier of the desired list. Each list that resides in memory will have a different list identifier. Typically the list identifier is stored in an SAS SCL local variable and passed to the various SAS SCL list functions. The list identifier will be in the form of an integer. This list identifier is a pointer used by SAS to locate the start of the list in memory.

Advantages

- Reside in memory (quick access)
- Very simple syntax to access contents
- Can be saved and loaded from/to SAS catalog or Flat File (text based file)
- Each cell can contain up to 32K of data
- Name of each "cell" can be up to 256 characters
- Name of each item or cell does not have to conform to SAS naming conventions (can use numbers and special characters)

Disadvantages

- Only accessible through SAS SCL code
- Changes to list structure or contents requires SAS SCL code

USING SCL LISTS AS VOCABULARY DICTIONARIES

Using a few of the basic SAS SCL list functions, we are able to create and use these lists to store vocabulary to be used in SAS programs and applications. Let us go over the steps required to make use of these SAS SCL lists as "vocabulary dictionaries".

CREATING SAS SCL LISTS

The first task would be to create a separate list for each language to be used in your application. The **MAKELIST** SAS SCL function is used to create an empty SAS SCL list in computer memory.

The sample code in **Figure 5** illustrates the creation of two SAS SCL lists. Once the lists have been created in computer memory, list identifiers are returned to the corresponding SAS SCL local variables. These list identifiers will be integers, normally 4 digits in length. These list identifiers will be used with various other SAS SCL list functions to manipulate the contents.

Figure 5

```
Init:
     DECLARE   list   English_List
                      French_List;

      English_List = MAKELIST();
      French_List  = MAKELIST();
return;
```

Example to illustrate the creation of SAS SCL Lists.

INSERTING DATA INTO SAS SCL LISTS

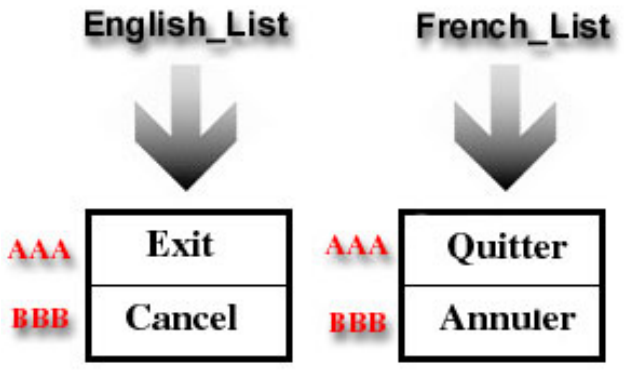
Once a SAS SCL list has been created for each language, we want to fill these with all of the vocabulary that we would use in our application(s). To insert items into a SAS SCL list we use the **INSERTC** SAS SCL function. This is usually a one-time task performed at the beginning of the development cycle. In an iterative fashion we need to add items to each list, insert desired phrases and name each item (or cell) with the desired keywords.

Each list should contain the same number of items with the same item names. The only difference should be the contents of the cells. This is illustrated in **Figure 6**.

Figure 6

```
Add_Items:
   English_List = INSERTC(English_List,"Exit",-1,"AAA");
   English_List = INSERTC(English_List,"Cancel",-1,"BBB");

    French_List  = INSERTC(French_List,"Quitter",-1,"AAA");
   French_List  = INSERTC(French_List,"Annuler",-1,"BBB");
return;
```



Example to illustrate the insertion of keywords / phrases into SAS SCL Lists.

STORING SAS SCL LISTS

Due to the fact that SAS SCL lists reside in memory, it is necessary to save these "vocabulary dictionary" lists so that they may be referenced and used again in the future. If they are not saved, they will be lost when you terminate your SAS session. Use the **SAVELIST** SAS SCL function to create a copy of the SAS SCL list in memory to disk (SAS catalog entry or external file). Although SAS SCL lists may be stored in flat files (text-based files) it is recommended to store them in SAS catalogs. When stored as entries in SAS catalogs, these saved lists will have the extension "SLIST". **Figure 7** illustrates the way in which we would store our "vocabulary dictionaries" to disk (SAS catalog).

Figure 7

```
Store_Lists:
   RC = LIBNAME( "MYLIB","C:\Application XYZ" );

   RC = SAVELIST ( "CATALOG",
                   "MYLIB.CATNAME.ENGLISH.SLIST",
                   English_List );

   RC = SAVELIST ( "CATALOG",
                   "MYLIB.CATNAME.FRENCH.SLIST",
                   French_List );
return;
```

Example to illustrated the storing of SAS SCL Lists.

Due to the fact that SAS SCL lists reside in memory, it is good programming practice to delete them once the SAS application terminates. To remove SAS SCL lists from memory use the **DELLIST** SAS SCL function. Now that we have saved our "vocabulary dictionary" lists to SAS catalogs as illustrated in **Figure 7**, we can now delete them from memory as seen in **Figure 8**.

Figure 8

```
Delete_Lists:
   RC = DELLIST ( English_List );

   RC = DELLIST ( French_List );
return;
```

Example to illustrated the deletion of SAS SCL Lists.

LOADING SAS SCL LISTS

Once a SAS SCL list is saved to disk (SAS catalog or flat file) one can load it back into memory at any time using the **FILLIST** SAS SCL function. An empty SAS SCL list must be first created (using the MAKELIST SAS SCL function) before using the **FILLIST** SAS SCL function.

Following our example, it would be necessary to provide the user with a choice of language on an application start-up or "splash" screen. Once the user selects the language of choice, we would simply load the corresponding saved SAS SCL "vocabulary dictionary" list into memory. If "English" is selected we would load the English "vocabulary dictionary" list, or if "Français" was selected we would load the French "vocabulary dictionary" list. **Figure 9** illustrates how one would load a SAS SCL "vocabulary dictionary" list into memory based on a user's language preference (pseudo code).

Figure 9

```
Load_Lists:
    DECLARE    LIST    Language_List;

    RC = LIBNAME( "MYLIB","C:\Application XYZ" );

    Language_List = MAKELIST();

    If (<User Selected English>) then
       RC = FILLIST ( "CATALOG",
                      " MYLIB.CATNAME.ENGLISH.SLIST",
                        Language_List );
    Else if (<User Selected French>) then
       RC = FILLIST ( "CATALOG",
                      "MYLIB.CATNAME.FRENCH.SLIST",
                       Language_List );
return;
```

Example to illustrated the loading of SAS SCL Lists.

RETRIEVING DATA FROM SAS SCL LISTS

Once the desired SAS SCL "vocabulary dictionary" list is loaded into memory, we can easily retrieve the desired phrases based on corresponding keywords. To retrieve data from a SAS SCL list we use the **GETNITEMC** SAS SCL function. This particular SAS SCL function retrieves data based on the name of the item or cell. There are other SAS SCL list functions that can be used to retrieve data based on other criteria such as position or data type (numeric or another list identifier). For this discussion we will use the **GETNITEMC** SAS SCL function, as this is what we need for this application.

Figure 10 illustrates how our SAS SCL program would populate two pushbuttons from the current "vocabulary dictionary" list that resides in memory:

Figure 10

```
Populate_Buttons:

   BUTTON1.Label = GETNITEMC ( Language_List, "AAA" );
   BUTTON2.Label = GETNITEMC ( Language_List, "BBB");

return;
```

Example to illustrated how to retrieve data from SAS SCL Lists using the item's name.

It's just that simple!

GENERALIZED UTILITY FOR LANGUAGE PROCESSING (GULP)

As we have seen, SAS SCL lists are a useful medium or SAS data structure used for storing and retrieving vocabulary. However, there is one major drawback to using SAS SCL lists. They are only accessible through SAS SCL code. If one wishes to access a SAS SCL list from BASE SAS or some other modules in SAS, one must indirectly go through SAS SCL code. Therefore, it would be useful to build a generalized tool that would allow the developer to easily access and manipulate SAS SCL lists, in particular, these "vocabulary dictionaries".

At Statistics Canada, we are required to make all of our applications bilingual (French and English). Therefore, we must deal with this issue on every new system that we develop. A few years back we decided to build a generalized utility called **GULP** (**G**eneralized **U**tility for **L**anguage **P**rocessing) that would allow us to easily manipulate and maintain these SAS SCL "vocabulary dictionary" lists.

This application called **GULP** is intended to be used by the SAS developer. It is a small SAS/AF application that makes use of a user-friendly, point and click approach to manage and maintain these SAS SCL "vocabulary dictionary" lists.

GULP allows the SAS developer to:

- Organize up to 5 "vocabulary dictionaries" (add, delete, modify)
- Organize unlimited amount of keywords / phrases (add, delete, modify)
- Export and import vocabulary data out of / into "vocabulary dictionaries"
- View summary statistics on defined "vocabulary dictionaries"

A full discussion of **GULP** would be another paper in itself. Like the name, **GULP** is a unique application that has a very specific function; to improve our use of vocabulary within SAS applications at Statistics Canada.

SUMMARY

If there is only one thing to be learned from this discussion, it should be:

Seperate your vocabulary from your SAS code

But hopefully we have shown that the SAS SCL lists are an excellent medium for storing vocabulary to be used in SAS applications. Although we did not discuss all of the available SAS SCL list functions, we covered the majority of the ones needed to implement a strategy to handle vocabulary.

As developers, we must address this issue in just about every system that we work on. Separating vocabulary from our SAS programs and applications and storing them in SAS SCL lists (or any other SAS medium) will prove to be a time saver in terms of development, testing and maintenance.

ACKNOWLEDGEMENTS

The author would like to acknowledge the entire SAS Technology Centre Team at Statistics Canada for their hard work, dedication and support. It is this kind of team work that motivates SAS knowledge and passion.

CONTACT INFORMATION

For information on topics cover in this paper including the **GULP** (**G**eneralized **U**tility for **L**anguage **P**rocessing) application mentioned please contact:

Statistics Canada Statistique Canada

Greg McLean

Project Leader – SAS Technology Centre
System Development Division
R.H. Coats Building, 14th Floor, Section Q

Ottawa, Ontario, Canada K1A 0T6
(613) 951-2396 Fax (613) 951-0607
greg.mclean@statcan.ca

Canada

Statistique Canada Statistics Canada

Greg McLean

Chef de projet – Centre de Technologie SAS
Division de développment des systèmes
Édifice R.H. Coats Building, 14ième étage, section Q

Ottawa, Ontario, Canada K1A 0T6
(613) 951-2396 Fax (613) 951-0607
greg.mclean@statcan.ca

Canada