Paper #AD07

A SAS/AF® Utility for Managing Stored SCL Lists

Derek Morgan, Washington University Medical School, St. Louis, MO

ABSTRACT

SCL lists are an important component of SAS® System applications development. Unfortunately, there are no tools within SAS for the creation and editing of SCL lists that are stored as SLIST catalog entries. This describes one such application for the creation and management of stored SLIST entries. You can create an SLIST entry from an ASCII file, edit/delete existing SLIST entries, and write an existing SLIST entry to an ASCII file. The utility works on both named and ordered SLIST entries.

THE INITIAL PROBLEM

The Long Life Family Study uses a SAS/AF® application to administer several screening survey instruments over the phone to respondents. These survey instruments have long scripts that are to be read to the participant. The scripts must be displayed in text pad controls because of their length and because of limited real estate. It would be annoying to build the scripts at run-time through the frame's SCL code, especially since there are three field centers involved in the reading of these scripts, so there are three versions of each script. The scripts don't change, so why create them dynamically?

In addition, there are several places in the application where a pull-down list of items would be a great user convenience. There are several different types of item lists necessary, but most of them are fixed, and will not change for the duration of the study. Item lists for combo and list box controls, as well as pull-down lists attached to fields in table viewers are also SCL lists. While you can generate them dynamically in the code, as with the scripts, why should you?

Stored SCL lists are the obvious solution, but how do we create them for several scripts and several miscellaneous lists? Writing a program to create them from SCL code is just a variation on the creating umpteen different FRAMES, each of which displays a single, hard-coded script. Instead of building the scripts or the lists dynamically at run-time, you would be building them dynamically once during the application development process. The only net savings would be in computer resources at run-time.

The optimal solution would be to create a tool that would allow you to build your scripts and item lists from ASCII files. It would be flexible, re-usable, and once something is in a stored SCL list, you should be able to correct the list on the fly instead of reprocessing the ASCII file. That's exactly what this small SAS/AF application does.

THE APPROACH

The first step in the process of developing this tool was to determine exactly what functionality would be built into the application, and what tasks we wanted it to handle. The first limitation we imposed was to restrict the SCL list items to character items only. This allowed us to use a single approach, and not require the user to define the type of each list item. That also imposed a second limitation: the tool would not handle an SCL list made up of SCL lists or objects. Since the scope of our immediate needs were more or less restricted to character list items, allowing the tool to handle anything other than character list items could be put on the "future improvements" shelf right away. Next, we came up with a list of four basic tasks that the tool should be able to handle in its first release:

1) Creating and storing an SCL list from an external file OR by typing text.
2) Editing an existing SCL list entry.
3) Creating an external file from an existing stored SCL list.
4) Deletion of an existing stored SCL list.

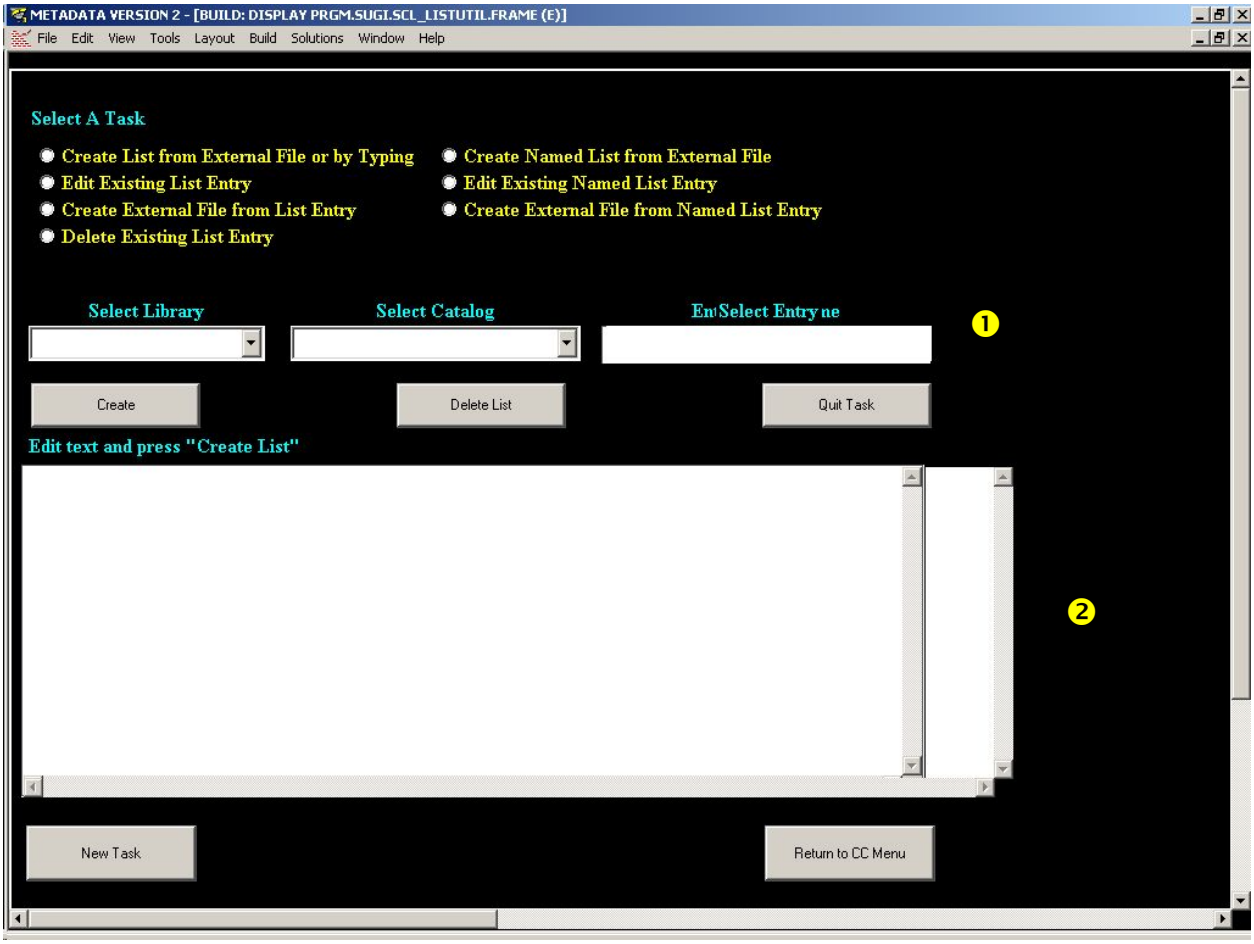
The first three above tasks needed to be built for both named and ordered SCL lists as well, while the fourth is common to both types of SCL lists.

The SCL list creation and maintenance wound up being a surprisingly small piece of the application. SCL list functions such as LISTLEN(), SAVELIST() and CLEARLIST(), and INSERTC() do most of the communication between application object to saved SCL list entry. We took advantage of the fact that the items attribute of the text pad control is an SCL list itself for the interaction with ordered SCL lists. However, since named lists have two pieces (name and item,) using a text pad for them would have required some fancy parsing footwork. The easier approach was to put the named list information into a temporary dataset, and access it via a table viewer control. Place these items on top of each other, and you have no screen real estate issues.

Since ordered SCL lists are sequential and item-driven, you can use the SCL external file handling functions to get from ASCII text file to SCL list and vice versa. Named lists are, outside of SAS, the same way in that reading/writing them in sequential fashion does not alter their utility as long as the name always goes along with the item.

All of this means that the majority of the code in the application is devoted to the user interface. Objects on the frame need to be turned on and off according to the context of the user's task within the application. The basic frame design looks like this:

Build Mode: The SCL List Editor



The apparently odd screen layout is to accommodate the SLC list editor's appearance on a LINUX system using X-win 32, which is one of the operating environments within which this utility would be functioning. We have two sets of overlaid objects (❶ and ❷, above). The field for naming a new SCL list entry overlays the combo box for existing list entries (❶) and the text pad for ordered lists overlays the table viewer for named lists (❷.) Which of the overlaid objects is visible to the user depends on the task they have chosen from the task box.

In order to provide a standard layout for the table viewer at run-time, we ran the program and created a named list, which gave us a dataset template for the dataset to be viewed via a model in the table viewer. After putting this template data set name in the SAS Data Set model, we sized everything as we desired and then created a DATAFORM entry for the model. Once the DATAFORM entry was created, we removed both the DATAFORM entry and the template data set from the model, because in use, the table is created dynamically, and then it is assigned to the SAS Data Set model at run-time. Since you cannot assign a non-existent dataset to a model, you cannot set items such as column width or other column properties for the model. By creating the DATAFORM entry during the build process, you can associate it with the model at run time in the SCL once the model has a data set defined:

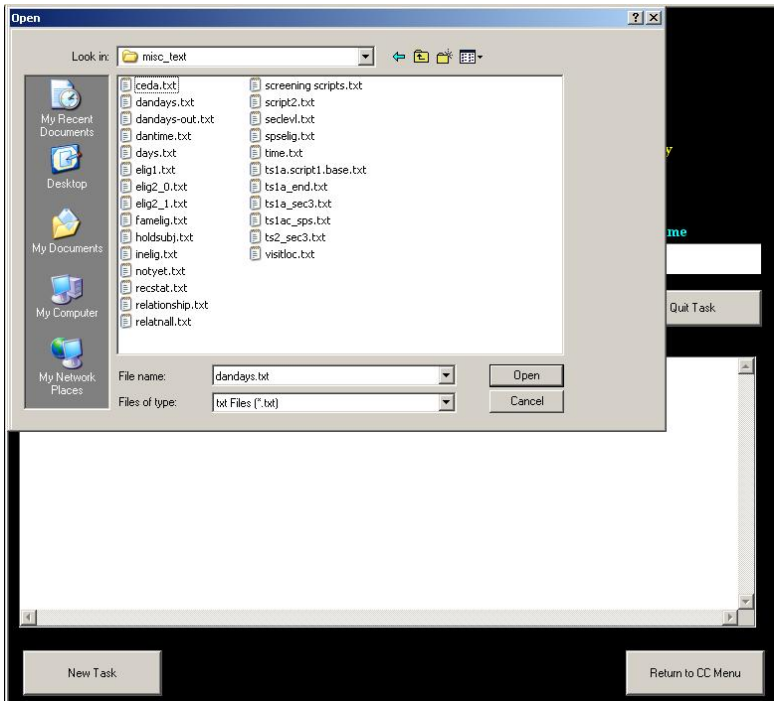
```
nlist.table = "temp";
nlist.dataformEntry = "prgm.sugi.scledit.dataform";
```

WHAT DOES IT LOOK LIKE WHEN I USE IT?

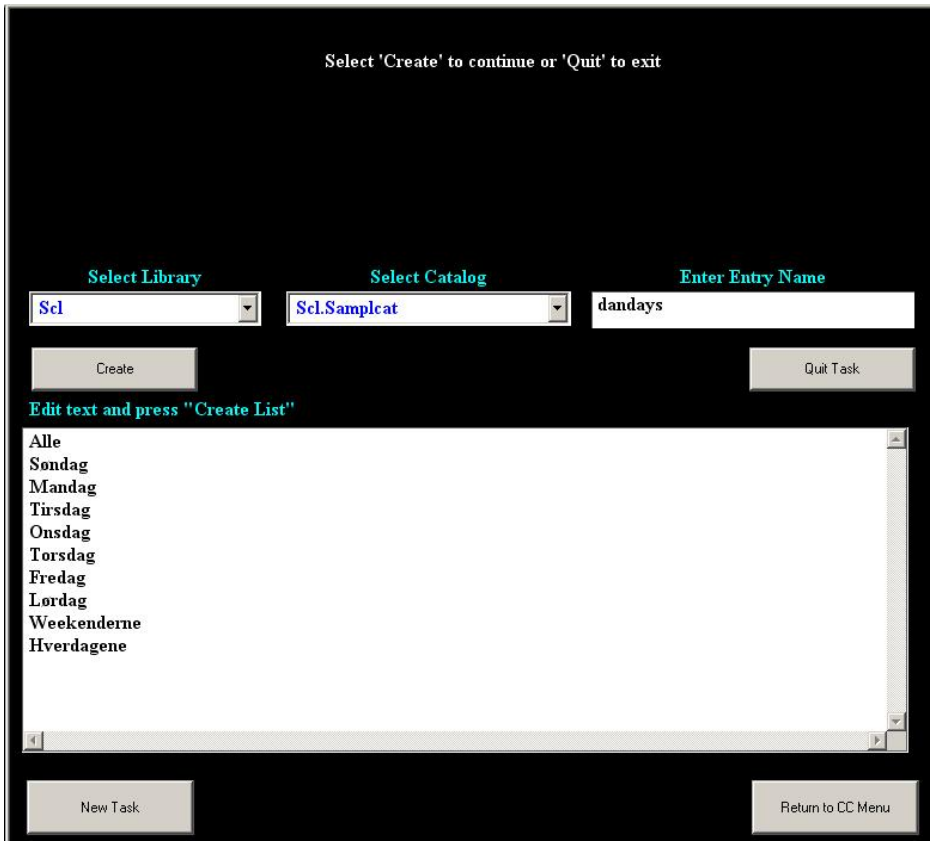
Using the editor is straightforward. To create an ordered SCL list entry, select the first option, which is "Create List from External File or by Typing". A file dialog box will pop-up requesting the external file. If you hit "Cancel" from the file dialog, the application assumes that you're going to type the list items into the text pad area. You cannot type items directly into a named list; this is a program limitation, and can be added in the future. If you are loading an

ASCII file, the ASCII file record is considered to end at the line terminator. Under Windows, this is a hexadecimal 0D. The terminator is important because text pad objects can wrap long lines of text and scroll them if necessary at run-time, so that you do not have to be responsible for determining where the line breaks should be. Therefore, the only line breaks necessary are the ones between list items.

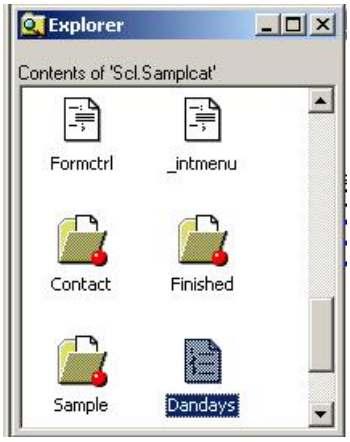
Here, we are going to create an ordered SCL list from the file “dandays.txt”.



Once the file has been selected, the contents are read into the text pad object. Now you select the library and catalog where you want the SLIST entry stored, and give it a name.



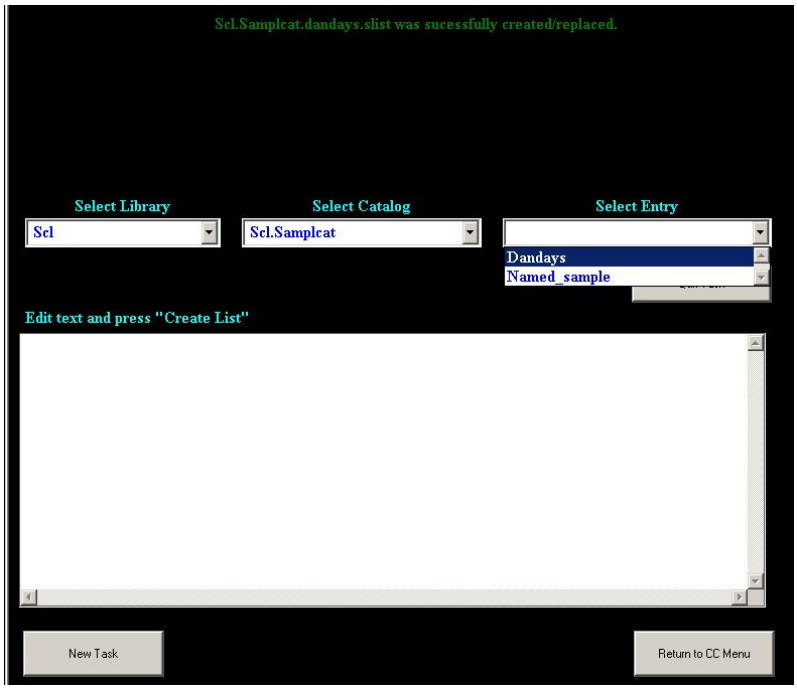
Once that is done, push the “create” button (which does not become visible unless there is text in the text pad or data in the table viewer.) Now there is an entry named “dandays.SLIST” in SCL.SAMPLCAT.



Click on it, and the log displays:

```
SLIST(
   'Alle' {C}
   'Søndag' {C}
   'Mandag' {C}
   'Tirsdag' {C}
   'Onsdag' {C}
   'Torsdag' {C}
   'Fredag' {C}
   'Lørdag' {C}
   'Weekenderne' {C}
   'Hverdagene' {C}
)[3]
```

It really is that easy. To edit the list once it has been saved, click on “Edit Existing List Entry”, select the library, catalog, and entry as shown below.



This will load the contents of “SCL.SAMPLCAT.dandays.SLIST” into the text pad, and you can edit the contents of the ordered list here, then save the changes that you made.

Once you've finished making your changes, hit the "Create" button again, and your changes will be saved to the SCL list entry. From the above example, we've removed the first and last items in the list, and saved it. Now when we click on the dandays SLIST entry in SCL.SAMPLECAT from the Explorer, the log shows:

```
SLIST(
   'Søndag' {C}
   'Mandag' {C}
   'Tirsdag' {C}
   'Onsdag' {C}
   'Torsdag' {C}
   'Fredag' {C}
   'Lørdag' {C}
   'Weekenderne' {C}
)[3]
```

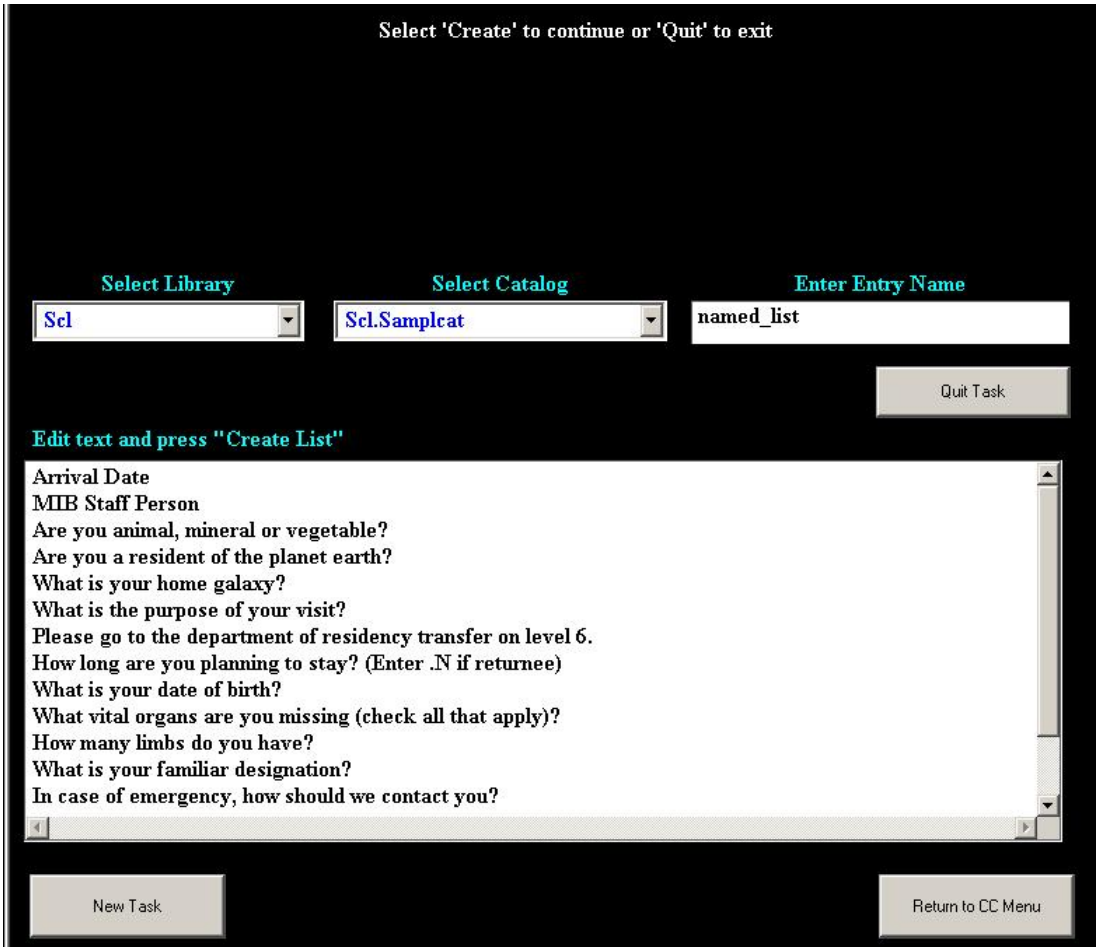
Creating an external file from an ordered SCL list works the same way as creating an ordered SCL list from an external file except in reverse. Once you've selected the SLIST entry, the list contents are displayed in the text pad. When you hit "Create" a file dialog pops up asking you to name the text file and its location. Deleting an existing entry is a matter of selecting the SLIST entry to be deleted, and confirming that this is what you indeed want to do.

Named lists work in a slightly different fashion. First, the ASCII file must be tab-delimited; the first field in each record is the item name, and the second is the item text.

Sample Named List ASCII File

```
A  →  Arrival·Date¶
B  →  MIB·Staff·Person¶
1  →  Are·you·animal,·mineral·or·vegetable?¶
2  →  Are·you·a·resident·of·the·planet·earth?¶
3  →  What·is·your·home·galaxy?¶
4  →  What·is·the·purpose·of·your·visit?¶
X1 →  Please·go·to·the·department·of·residency·transfer·on·level·6.¶
5  →  How·long·are·you·planning·to·stay?·(Enter·.N·if·returnee)¶
6  →  What·is·your·date·of·birth?¶
7  →  What·vital·organs·are·you·missing·(check·all·that·apply)?¶
8  →  How·many·limbs·do·you·have?¶
9  →  What·is·your·familiar·designation?¶
10 →  In·case·of·emergency,·how·should·we·contact·you?¶
11 →  Please·provide·your·contact·frequency.·¶
12 →  Please·confirm·the·purpose·of·your·visit.¶
fini→ Thank·you.·Please·enjoy·your·stay·on·the·third·planet·of·the·Sol·system.
```

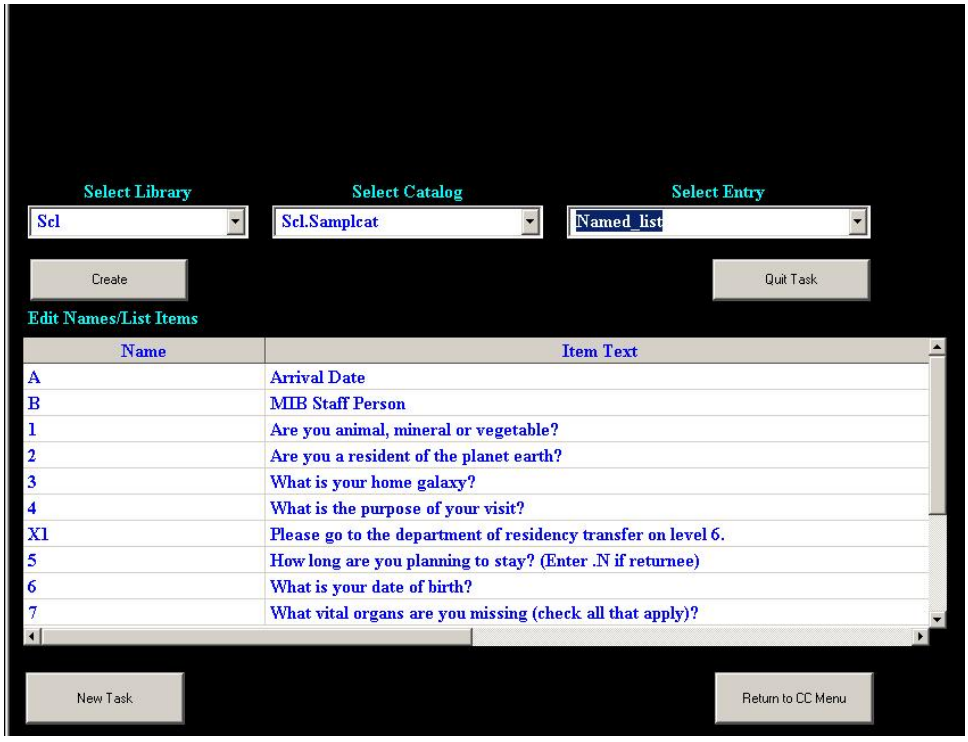
When you select the file name, the text pad is filled with the list items. You do not have access to the item names at this point, so it looks exactly like an ordered SCL list. When you press “Create”, the application will act the same as it does with ordered SCL lists.



However, the list has been saved as a named list, even though you have not yet been able to see the names in the application. If we click on the named_list SLIST entry in SCL.SAMPLECAT from the Explorer, the log shows:

```
SLIST(
   A='Arrival Date' {C}
   B='MIB Staff Person' {C}
   1='Are you animal, mineral or vegetable?' {C}
   2='Are you a resident of the planet earth?' {C}
   3='What is your home galaxy?' {C}
   4='What is the purpose of your visit?' {C}
   X1='Please go to the department of residency transfer on level 6.' {C}
   5='How long are you planning to stay? (Enter .N if returnee)' {C}
   6='What is your date of birth?' {C}
   7='What vital organs are you missing (check all that apply)?' {C}
   8='How many limbs do you have?' {C}
   9='What is your familiar designation?' {C}
   10='In case of emergency, how should we contact you?' {C}
   11='Please provide your contact frequency. ' {C}
   12='Please confirm the purpose of your visit.' {C}
   FINI='Thank you. Please enjoy your stay on the third planet of the Sol
system.' {C}
)
```

However, you can see the item names in the application when you press “Edit Existing Named List Entry”. Here, we’ve selected the “named_list” entry. When the entry is read, a temporary SAS data set is created, and the table viewer will take the place of the text pad. This allows you to edit the item name or the item text. Bear in mind that list item names can be any valid set of characters, and do not have to conform to the variable naming convention in SAS. The list item names are automatically set to uppercase, but the list item text is not.



LESSONS LEARNED

The first lesson we learned was that you cannot manipulate list items of one character in a text pad under the Windows operating system. “A (some text)” was valid, but “A” would not show. In fact, the production version of the application tests each line for a length of greater than 1, and warns you if you do. However, you can create a line with a single character from an ASCII file—you just can’t use the application to edit the SLIST entry under Windows.

The next lesson involved inter-platform compatibility with our LINUX system. If “smart quotes” were in the SLIST entry, the SLIST entry would be truncated after the first smart quote. Since the scripts for the screening application were created using cut-and-paste from Microsoft Word versions of the screening forms, the smart quotes were cut-and-pasted too. While this does not cause an issue with the Windows version of the software, we got lots of calls

from our LINUX users telling us that the scripts were truncated. This issue was solved easily enough: we just replaced the smart quotes with plain quotes through the application on the Windows side, and then re-did a PROC CPORT.

SUMMARY

SCL lists are a valuable piece of SAS applications development. The development of a tool to create and manipulate SCL lists has saved a great deal of development time and energy. The need for this development, combined with the timeline in which it was to be accomplished forced some compromises in its initial design, but it is still useful for both ordered and named SCL lists containing only character items.

ACKNOWLEDGEMENTS

This work has been partially funded by NIH grant U01 AG023746.

CONTACT INFORMATION:

Further inquiries are welcome to:

Derek Morgan
Division of Statistical Genomics
Washington University Medical School
Box 8506, 4444 Forest Park Blvd.
St. Louis, MO 63108
Phone: (314) 362-3685 FAX: (314) 362-4227
E-mail: derek@wustl.edu

This and other SAS System examples and papers can be found on the World Wide Web at:

http://www.biostat.wustl.edu/~derek/sasindex.html