Rapid Applications Development using the Data Form and Data Table Objects in SAS/AF® Software for Release 6.11

Scott L. Wilkins
SAS Institute Inc., Cary, NC

ABSTRACT

There are two new data entry objects available in Release 6.11 of the SAS® System in the SAS/AF® FRAME entry. These objects are called the Data Form (Experimental for Release 6.11) and the Data Table. The goal of this paper is to examine how these data entry objects contribute to Rapid Application Development (RAD) when creating a data entry application using the SAS/AF FRAME entry.

REQUIREMENTS

In order to build and use the Data Form and Data Table data entry objects described here, it is necessary to have Release 6.11 of SAS/FSP® and SAS/AF® Software installed and licensed.

INTRODUCTION

SAS users writing data entry-type applications with the SAS/AF FRAME entry continue to have to use the same amount of Screen Control Language (SCL) as in the PROGRAM entry to link the screen variables (or objects representing screen variables) to the data set variables and perform the subsequent data input/output (Data I/O) operations.

In some applications where the screen variables are closely matched to the data set variables, the SCL code can make use of the SCL Function SET to handle the linking. This method has several limitations, not least of which is the need for logic to determine which observation to fetch as well as when to perform an update to save changes made to the data values.

Another method of performing the Data I/O for a SAS/AF data entry application deals with the use of the SCL Functions OPEN, FETCHOBS, VARNUM, GETVARC, GETVARN, PUTVARN, and PUTVARC. But, this method becomes very cumbersome as the number of variables to keep track of increases.

Figure 1 illustrates the amount of code needed to update a Text Entry Field (FRNAME) with a value from a data set variable (NAME) and then to update with the new value if the Text Entry Field is modified.

```
INIT:
dsid = open('sasuser.class','i');
rc = fetchobs(dsid,10); /* fetch obs. 10 */
vnum = varnum(dsid,'name');
fname = getvarc(dsid,vnum);
rc = close(dsid);
RETURN;
FRNAME:
dsid = open('sasuser.class','u');
vnum = varnum(dsid,'name');
rc = fetchobs(dsid,10);
call putvarc(dsid,vnum,fname);
rc = update(dsid);
rc = close(dsid);
RETURN;
```

Figure 1: SCL for Single Variable Data I/O

In Release 6.11 of the SAS System, two new data entry objects have been added to the SAS/AF FRAME Entry that allow viewing and editing of SAS data sets: The Data Form (Experimental for Release 6.11) and the Data Table objects. These objects greatly simplify the task of creating SAS/AF FRAME based data entry applications.

Data Form

The Data Form object enables viewing and editing of a SAS data set within a FRAME application. Data is presented one observation at a time. This object is similar in some ways to the SAS/FSP FEDIT procedure.

To create this object in build mode, select the ACTIONS menu and select MAKE. Then select Data Form (experimental) from the selection list. The Data Form object places a
special selection (FORM->) on its BUILD popup menu. This allows for Data Form specific BUILD popup menu selections.

<table>
<thead>
<tr>
<th>NAME:</th>
<th>Alice</th>
</tr>
</thead>
<tbody>
<tr>
<td>SEX:</td>
<td>F</td>
</tr>
<tr>
<td>AGE:</td>
<td>13</td>
</tr>
<tr>
<td>HEIGHT:</td>
<td>56.5</td>
</tr>
<tr>
<td>WEIGHT:</td>
<td>84</td>
</tr>
</tbody>
</table>

Figure 2: Example of the Data Form Object

The Data Form object allows the ability to incorporate SAS/AF FRAME display objects into the Data Form. By default, the data values and data labels are displayed using Input Field objects and Input Field Label objects respectively. But any display object such as a Slider or Critical Success Factor object can easily be used to display the data in a Data Form object.

Data Table
The Data Table object enables viewing and editing of a SAS data set within a FRAME application. The observations in the data set are presented in a tabular format. Each row represents an observation in the data set and each column represents a variable. This object is similar in some ways to the SAS/FSP FSVIEW procedure.

To create this object in build mode, select the ACTIONS pmenu and select MAKE. Then select Data Table from the selection list.

The Data Table object places a special selection (TABLE->) on its BUILD popup menu. This allows for Data Table specific BUILD popup menu selections.

Figure 3: Example of the Data Table object

THE DATA ENTRY OBJECTS AND RAPID APPLICATIONS DEVELOPMENT (RAD)

These objects are designed for anyone who wants to create powerful data entry applications using the FRAME entry. There are several features, however, of the Data Form and Data Table data entry objects that make them important for application developers who need to be able to prototype an application quickly and get it into production in a timely manner.

Some of these features are as follows:

- Model-View-Controller architecture performs Data access and updates behind the scenes.
- Extensive attribute windows allows for application customization without having to code any SCL.
- Attribute windows allow multiple attributes to be set for multiple columns at once.
- Runtime popup menu for Scrolling and Data I/O operations.
- The DATAFORM entry allows the Data Table and Data Form objects to share customizations.
- Availability of the Model SCL entry in addition to the FRAME SCL entry.
- Ability to use drag and drop to link columns to display objects in the Data Form viewer.
MODEL-VIEW-CONTROLLER

These two objects rely on data model classes in order to access the SAS data set associated with them. This is what is known as the MVC (Model-View-Controller) paradigm. In the case of the Data Form and the Data Table, the model, viewer, and controller are bundled together so that the application developer does not have to be concerned with attaching them.

Using MVC, the viewer class controls the graphical presentation of the data and the model/controller class handles the interface between the data source and the viewer. The Data Form and Data Table objects use the SAS Data Set Data Model Class, which is able to interpret valid SAS data files, and pass that information on to the table or form viewer. Likewise, any modification such as updating values, adding rows, or deleting rows the user makes via the viewer is passed along to the model so that the data source can be updated.

What this means, is that the application developer has two ready-to-use data viewing and editing objects that are prepared to operate on data contained in any SAS data file and can do so without the need of SCL code.

These objects can be build and in use, very quickly.

ATTRIBUTE WINDOWS

The Data Form and Data Table objects share attribute windows. Customizations made in these attribute windows are saved in a DATAFORM catalog entry.

These attribute windows allow for a large amount of customization and deal with issues such as fonts, colors, justification, data validation, setup, and many other areas as described in the sections below.

Data Table and Data Form Attributes Window

Figure 4 is the primary attribute window for the Data Form and/or Data Table objects. All other attribute windows for these objects are reached through this window.

This screen is where the Table (SAS data set) name to associate with the object is specified. Once a table name is entered, a Where clause may be entered that will be applied to the data set each time it is accessed by the object.

A new DATAFORM Entry may be specified to save any customizations made for the Data Form or Data Table. Alternatively, an existing DATAFORM Entry may be specified to use customizations that already exist.

In the SCL Entry field, an SCL Entry that is separate from the frame’s SCL entry may be specified. This SCL Entry is used by the data model for actions such as data validation and any computations made using the columns in the data set.

Browse or Edit Data Mode is specified here as well as Record or Member Lock Mode, and whether or not vertical and horizontal Scrollbars will appear on the viewer. Further attributes allow specifying other Table Characteristics such as whether or not deletion is allowed, addition/duplication is allowed, and whether or not to allow overrides of conditions where a field is in error or a field is required. And finally, the Autosave value is set in this screen. The autosave value determines how many row modifications must occur before an automatic save of the data is performed.

There are several Additional attribute screens accessed from the main attribute screen. The ones that are specific to these objects include Customize columns..., Customize form..., and Customize table...
Customize Columns Window

If a Table name and a DATAFORM entry have been specified, this set of attribute windows can be used to set customizations on a column-by-column basis. The Customize Columns Window has five buttons that allow setting information about Columns, Fonts, Classes, Options, and Colors.

Each of these windows provide a list of the columns contained in the data set that is linked to the object. These column lists allow the application developer to signal which individual column or group of columns to apply a specific attribute setting or group of attribute settings to. Everywhere a column selection list exists, a popup menu is available to help with the selection of desired columns.

For example, in the OPTIONS window the following popup menu selections are available:

- Select All
- Select All Numeric
- Select All Character
- Reset All To Defaults
- Reset Selected To Defaults

Consider the case where the application developer wants to set the CAPS option for all character columns in a data set that has 100 character columns mixed in with 100 numeric columns. In this case, they would simply use the Select All Character popup menu selection and then set the CAPS option which would apply to all the selected character columns.

The Computed Columns Window allows defining of the Name, Length, Label, Format, Informat, and Type of a computed column. The value for the computed column has to be provided through computations made by the model's SCL entry which is defined in the main attribute window in the SCL Entry field.

Fonts Window
Allows setting of the label and data fonts used by the viewer.

Classes Window
Allows setting of the classes to be used in the Data Form viewer's display of the data values. This does not affect the Data Table.

Options Window
Allows setting of options for capitalization, required, protected, maximum, minimum, initial value, and justification on a column-by-column basis.

Colors Window
Allows setting of colors to be used by the column's data and labels as well as colors to use for error conditions.

Customize Form Window

This window allows setting of several attributes that are specific to the Data Form object. First it controls whether the Column Window is displayed. This window is available in the frame's build mode and allows the developer to link specific data set or computed columns to display objects that have been created inside the Data Form object.

For example, if the data set contains a column named GENDER, that column can be linked to a Radio Box object, placed in the Data Form, with the selections Male and Female. The value of the column will be reflected in the Radio Box.
changes the setting the new value in the data set. This is discussed in the section, LINKING COLUMNS USING DRAG AND DROP.

The Label/Data Widgets in the diagram that will control whether the
In order for a command to be applied to the Data Form or the Data Table object, the region that they are in has to be the active region when the command is executed. In order to issue a command to one of the data entry objects follow these steps:

1. Click in the master region so that no region within it is considered active.
2. Click on the frame’s command line.
3. Type the command, then ARROW down to the data entry object that the command will be applied to.
4. Once the cursor is inside the data entry object press ENTER and the command will be applied to the object.

As with the runtime popup menu, these commands are part of the data entry objects. No extra coding is needed on the part of the application developer to provide this functionality.

THE DATAFORM CATALOG ENTRY

A DATAFORM catalog entry is used to save column customizations as well as widget information for the Data Form and Data Table objects. Since both the Data Form and Data Table use the same entry type to store their customizations, attributes set for one can be used by the other.

Consider an application requiring a form and table view of a data set using blue as the data’s background color in both viewers. The background color for the data in the Data Form object could be set to blue and saved in a DATAFORM entry. That same DATAFORM entry name could then be specified for the Data Table object. Then, whenever either the Data Form or Data Table displayed, the data’s background color would be blue.

To specify a DATAFORM entry, enter its name in the DATAFORM Entry field of the Main Attribute Window of either the Data Form or Data Table object.

The only way to create a DATAFORM entry is by entering the name of the entry to create in the DATAFORM Entry field. If the entry does not exist then it will be created and any customizations will be saved in it when the attributes window is ended. If the DATAFORM Entry is blank when the attribute screen is ended, any customizations made while in the attribute windows will be lost.

THE DATA MODEL SCL ENTRY

When building a FRAME entry SCL code is usually written only in the SCL source entry associated with the FRAME itself. The Data Form and Data Table, however, support a second SCL source entry. Since this: SCL entry in not linked to the FRAME but is linked to the data model, which interprets the data source, it is refereed to as the Model SCL entry.

To assign a Model SCL entry specify the Model SCL entry’s name in the Main Attribute Window for the Data Form or Data Table object in the SCL Entry field.

After ending the attribute screen, the Model SCL entry can be accessed through the BUILD popup menu under FORM-> or TABLE, Edit SCL... If the entry exists it will be displayed and if it does not, it will be created and then displayed.

One advantage to using the model SCL entry is the ability to create sections of code which correspond to the columns in the SAS data set. This is done by using the column’s name as the label of the SCL code section. Then, whenever that column’s value is modified the labeled section of the SCL code will run. This is useful for performing extensive data validation or certain computations and/or actions based on given values or modified columns.

As with other SCL entries the INIT: MAIN: and TERM: labeled sections are valid. Two additional labeled sections: DFINIT and DFTERM, are available. DFINIT executes once when the object is displayed and DFTERM executes once when the object is terminated.

To compile this Model SCL entry, either enter the COMPILE command on the command line of the SCL entry while editing it or use the BUILD popup menu selection FORM-> or TABLE->, Compile SCL... . It is important that the Model SCL entry be compiled in one of these two ways only. If the entry is compiled
from the BUILD Directory window, an error will result when the data entry object tries to use the SCL at runtime.

**LINKING COLUMNS TO WIDGETS USING DRAG AND DROP**

The Data Form object has a powerful, time-saving feature which allows the use of drag and drop to link specific columns with corresponding objects for display. Unlike the Data Table object, the Data Form object can use many different display objects to represent data in the columns of a data set. For example a Critical Success Factor object can be used to display the data in a numeric column.

This can be accomplished in two ways. First by specifying to use this particular object with the numeric column in the CLASSES window accessed through the CUSTOMIZE COLUMNS window.

The other way to accomplish this is by using the COLUMN WINDOW which can be activated by using the BUILD popup menu selection, FORM->, Display Column Window.

This brings up a window which has all the data set columns and computed columns listed in it. Now to link a column to a display object, simply create the object inside the Data Form, click on the column to link, hold down the DRAG mouse button, drag the column to the widget, and drop it there.

Linking columns to widgets in this manner, allows for the linking of a single column to as many display objects as needed. For example, a single numeric column could be linked to a Critical Success Factor object, a Scrollbar object, and a Text Entry Field object.

To be able to tell which columns are linked to which display objects, select the BUILD popup menu selection, FORM->, Turn Column Describe On. Then, by clicking on any display object, a message is generated reporting the column that the widget is linked to.

**CONCLUSION**

The Data Form and Data Table objects in Release 6.11 of SAS/AF Software will make the data entry application developer's job much easier. The Data Form and Data Table objects provide all the presentation and manipulation capabilities packaged together. With their extensive attribute screens, built-in popup menus and command support, the SAS/AF data entry objects allow for the creation of a powerful, robust data entry application in a very timely manner.

Although these objects are very well self contained it is important to also mention the large number of methods the classes, that make up the Data Form and Data Table, support. All these methods are available to be used in the SCL to further enhance data entry applications. Methods that can be passed to the Data Form, Data Table, and their data models are listed in APPENDIX A.

**NOTICES**

SAS, SAS/AF and SAS/FSP are registered trademarks of SAS Institute Inc. in the USA and other countries. © indicates USA registration.

**REFERENCES**


APPENDIX A

These methods are documented in the SAS/AF® Software: FRAME Class Dictionary, Version 6, First Edition in their respective chapters.

CLASS: DATA SET MODEL
PATH: SASHELP.FSP.DATA_M.CLASS
METHODS:
  _ADD_ROW_
  _ADD_COLUMN_
  _ADD_ROW_
  _DELETE_ROW_
  _DESCRIBE_COLUMNS_
  _EXEC_CMD_
  _FETCH_ROW_
  _FIND_ROW_
  _GET_AUTOSAVE_
  _GET_COLUMN_ATTRIBUTES_
  _GET_COLUMN_ID_
  _GET_COLUMN_NUMBER_
  _GET_COLUMN_TEXT_
  _GET_COLUMN_VALUE_
  _GET_CURRENT_ROW_NUMBER_
  _GET_DATASET_ATTRIBUTES_
  _GET_DATASET_NAME_
  _GET_COLUMN_ATTRIBUTES_
  _GET_ROW_
  _GET_ROW_NUMBER_
  _GET_WHERE_
  _GO_TO_ABSOLUTE_ROW_
  _KEY_COUNT_
  _REPEAT_FIND_ROW_
  _SAVE_
  _SAVE_AS_
  _SET_ATTRIBUTES_
  _SET_AUTOSAVE_
  _SET_COLUMN_ATTRIBUTES_
  _SET_COLUMN_TEXT_
  _SET_COLUMN_VALUE_
  _SET_DATASET_
  _SET_INITIAL_
  _SET_KEY_
  _SET_OPENMODE_
  _SET_ROW_
  _SET_WHERE_
  _SORT_
  _UNLOCK_ROW_
  _UPDATE_ROW_
  _INIT_
  _TERM_

CLASS: DATA SET DATA MODEL
PATH: SASHELP.FSP.DATA_M.CLASS
METHODS:
  _ADD_COMPUTED_COLUMN_
  _ADD_ROW_
  _COMMIT_NEW_ROW_
  _COPY_ROW_
  _DELETE_COMPUTED_COLUMN_
  _DELETE_ROW_
  _DISPLAY_COLUMN_LABEL_
  _DISPLAY_COLUMN_NAME_
  _ERROROFF_COLUMN_
  _ERRORON_COLUMN_
  _FETCH_ROW_
  _FIND_ROW_
  _GET_COLUMN_ATTRIBUTES_
  _GET_CURRENT_ROW_NUMBER_
  _GET_DISPLAYED_COLUMN_NAME_
  _GET_HIDDEN_COLUMNS_
  _GET_RECORD_INFO_
  _GET_ROW_
  _GET_ROW_NUMBER_
  _GO_TO_ABSOLUTE_ROW_
  _HIDE_COLUMN_
  _LOCK_ROW_
  _MOVE_COLUMN_
  _OVERRIDE_
  _OVERRIDE_ON_ERROR_
  _OVERRIDE_ON_REQUIRED_
  _PROTECT_COLUMN_
  _REPEAT_FIND_ROW_
  _REREAD_
  _REREAD_ALL_
  _SAVE_
  _SET_COLUMN_ATTRIBUTES_
  _SET_COLUMN_TEXT_
  _SET_COLUMN_VALUE_
  _SET_DATASET_
  _SET_KEY_
  _SET_MSG_
  _SET_OPENMODE_
  _SET_ROW_
  _SET_WHERE_
  _SORT_
  _UNHIDE_COLUMN_
  _UNLOCK_ROW_
  _UNPROTECT_COLUMN_
  _UPDATE_ROW_
CLASS: FORM EDITOR
PATH: SASHELP.FSP.FORM_E.CLS
METHODS:
_ATTACH_
_DETACH_
_SET_ATTRIBUTES_
_GET_ATTRIBUTES_
_GET_WIDGETS_
_GET_WIDGET_
_GET_CURRENT_WIDGET_
_SET_CURRENT_WIDGET_
_GET_CURRENT_WIDGET_INFO_
_GOTO_PAGE_
_GET_CURRENT_PAGE_
_GET_PAGE_COUNT_
_GOTO_ROW_
_GOTO_COLUMN_
_UPDATE_VALUES_
_GET_LABEL_BACKGROUND_COLOR_
_SET_LABEL_BACKGROUND_COLOR_
_GET_DATA_BACKGROUND_COLOR_
_SET_DATA_BACKGROUND_COLOR_
_GET_LABEL_COLOR_
_SET_LABEL_COLOR_
_GET_DATA_COLOR_
_SET_DATA_COLOR_
_GET_LABEL_FONT_
_SET_LABEL_FONT_
_GET_DATA_FONT_
_SET_DATA_FONT_
_GET_LINKED_COLUMN_
_SET_LINKED_COLUMN_
_UPDATE_DIM_
_GET_PROPERTIES_
_SET_PROPERTIES_
_HSCROLL_
_VSCROLL_
_SET_HSCROLL_
_SET_VSCROLL_
_GET_HSCROLL_
_GET_VSCROLL_
_SET_MSG_
_UPDATE_ROW_DIM_
_UPDATE_COLUMN_DIM_
_UPDATE_ROW_DATA_
_UPDATE_ALL_
_NEED_ROW_DATA_
_UPDATE_COLUMN_LABELS_
_SET_EDIT_
_UPDATE_COLUMN_DATA_
_UPDATE_HIDE_COLUMNS_
_UPDATE_UNHIDE_COLUMNS_
_UPDATE_PROTECT_COLUMNS_
_UPDATE_UNPROTECT_COLUMNS_
_VALIDATE_CHILD_NAME_
_SET_TOPROW_
_GET_TOPROW_
_GOTO_ROW_NUMBER_
COLUMN_WINDOW_INIT_
COLUMN_WINDOW_TERM_
REFILL_USING_ATTRIBUTES_
UPDATE_COLUMN_
_GET_LABEL_REPRESENTATION_
_SET_LABEL_REPRESENTATION_
_GET_LABEL_CLASS_
_SET_LABEL_CLASS_
_GET_DATACLASS_
_SET_DATACLASS_
CLASS: TABLE EDITOR
PATH: SASHELP.FSP.TABLE_E.CLASS
METHODS:
_ATTACH_
_DETACH_
_GET_SELECT_
_SET_SELECT_
_CLEAR_SELECT_
_CLEAR_ACTIVE_CELL_
_GET_ACTIVE_CELL_
_SET_ACTIVE_CELL_
_GOTO_CELL_
_HOLD_ROW_
_HOLD_COLUMN_
_RELEASE_ROW_
_RELEASE_COLUMN_
_SET_ATTRIBUTES_
_GET_ATTRIBUTES_
_HSCROLL_
_VSCROLL_
_GET_HSCROLL_
_GET_VSCROLL_
_SET_HSCROLL_
_SET_VSCROLL_
_SET_MAXROW_
_SET_MAXCOL_
_UPDATE_DATA_
_UPDATE_LABELS_
_UPDATE_COLUMN_LABELS_
_UPDATE_ROW_LABELS_
_UPDATE_TITLE_LABELS_
_UPDATE_DIM_
_UPDATE_COLUMN_DIM_
_UPDATE_ROW_DIM_
_UPDATE_TITLE_DIM_
_GET_LABEL_BACKGROUND_COLOR_
_SET_LABEL_BACKGROUND_COLOR_
_GET_DATA_BACKGROUND_COLOR_
_SET_DATA_BACKGROUND_COLOR_
_GET_LABEL_COLOR_
_SET_LABEL_COLOR_
_GET_DATA_COLOR_
_SET_DATA_COLOR_
_GET_GRID_STYLE_
_GET_GRID_WIDTH_
_GET_GRID_COLOR_
_SET_GRID_STYLE_
_SET_GRID_WIDTH_
_SET_GRID_COLOR_
_GET_LABEL_FONT_
_SET_LABEL_FONT_
_GET_LABEL_MARGIN_
_SET_LABEL_MARGIN_
_GET_LABEL_HJUST_
_SET_LABEL_HJUST_
_GET_LABEL_VJUST_
_SET_LABEL_VJUST_
_GET_LABEL_BORDER_COLOR_
_SET_LABEL_BORDER_COLOR_
_GET_LABEL_BORDER_STYLE_
_SET_LABEL_BORDER_STYLE_
_GET_LABEL_BORDER_WIDTH_
_SET_LABEL_BORDER_WIDTH_
_GET_DATA_FONT_
_SET_DATA_FONT_
_GET_DATA_MARGIN_
_SET_DATA_MARGIN_
_GET_DATA_HJUST_
_SET_DATA_HJUST_
_GET_DATA_VJUST_
_SET_DATA_VJUST_
_GET_DATA_BORDER_COLOR_
_SET_DATA_BORDER_COLOR_
_GET_DATA_BORDER_STYLE_
_SET_DATA_BORDER_STYLE_
_GET_DATA_BORDER_WIDTH_
_SET_DATA_BORDER_WIDTH_
_SET_PROPERTIES_
_GET_COLUMN_CONFORM_THRESHOLD_
_SET_COLUMN_CONFORM_THRESHOLD_
_GET_ROW_CONFORM_THRESHOLD_
_SET_ROW_CONFORM_THRESHOLD_
_GET_TABLE_HJUST_
_SET_TABLE_HJUST_
_GET_TABLE_VJUST_
_SET_TABLE_VJUST_
_GET_VSCOL_
_GET_VSROW_
_GOTO_ROW_
_GOTO_COLUMN_
_GET_MAXCOL_
_SET_MAXCOL_
_SELECT_COLUMN_
_SELECT_ROW_
_SELECT_ALL_
_GET_TOPROW_
_SET_TOPROW_
_GET_LEFTCOLUMN_
_SET_LEFTCOLUMN_
_GET_HELD_COLUMNS_
_SET_HELD_COLUMNS_
_GET_HELD_ROWS_
_SET_HELD_ROWS_