

Data Table and Data Form Enhancements in Release 6.12

Scott Wilkins



Scott L. Wilkins is an applications developer in the SAS/EIS® group in the Business Solutions Division at SAS Institute. Prior to this, Scott worked in the Quality Assurance department where he tested SAS/FSP® and SAS/AF® FRAME software. Scott has a BS degree in computer science from North Carolina State University. He has been a SAS® software user for seven years.

Abstract

This article outlines several new and enhanced features of the Data Form and Data Table classes in SAS/AF FRAME Release 6.12. Contained here is a discussion of the new consolidated online documentation about Data Form and Data Table that documents all classes that make up the Data Form and Data Table, additional run mode pop-up menu items for the Data Form and Data Table, new multiple selections support in the Data Table, new viewer and column attribute methods, and the new `_COLUMN_NAME_` SCL system variable. Also discussed are the new Data Form earmarks used for page navigation, support in the Data Form for one page layout creation, the movement of widgets from page-to-page in a Data Form, the support for creating master-detail applications with no SCL code, and the ability to use drag-and-drop to create computed columns and link them to widgets in a Data Form.

Introduction

In Release 6.11 of the SAS System, SAS Institute introduced two new data entry classes in SAS/AF FRAME, Data Form (experimental) and Data Table.

The Data Table class allows browsing or editing of SAS data files in a tabular format where multiple rows and columns are displayed in the viewer. (This is similar to the FSVIEW window in SAS/FSP.)

Data Table: SASUSER.CLASS

	NAME	SEX	AGE	HEIGHT	WEIGHT
1	Alice	F	13	72	72
2	Becka	F	13	65.3	98
3	Gail	F	14	64.3	90
4	Karen	F	12	56.3	77
5	Kathy	F	12	59.8	84.5
6	Mary	F	15	66.5	112
7	Sandy	F	11	51.3	50.5
8	Sharon	F	15	62.5	112.5
9	Tammy	F	14	62.8	102.5
10	Alfred	M	14	69	112.5

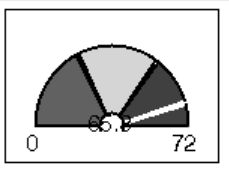
Display 1 A Data Table

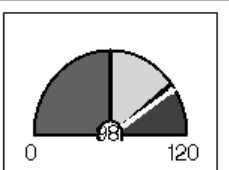
The Data Form class allows browsing or editing of SAS data files in a viewer that provides a form or a “one row at a time” view of the data. (This is similar to the FSEDIT window in SAS/FSP.) Additionally, any displayable SAS/AF FRAME object such as radio boxes, graphic text entry, sliders, and critical success factor objects can be used to display the data values within the Data Form.

Data Form: SASUSER.CLASS

Name:

AGE:

HEIGHT: 

WEIGHT: 

Display 2 A Data Form

In Release 6.11, the Data Form was classified as an experimental object, and the Data Table was classified as a production object. In Release 6.12 of the SAS System, the Data Form will also be classified as production.

In response to user needs, many new features and enhancements have been made to both the Data Form and Data Table objects in Release 6.12. This article discusses these new or enhanced features.

Documentation

The Data Form and Data Table classes are made up of two different classes each. The Data Table is created by automatically linking the Table Editor class and the Data Set Data Model classes together. The Data Form class is created by automatically linking the Form Editor and Data Set Data Model class together. Additionally, the Data Set Data Model delegates some of its methods to the Data Set Model Class.

In the Release 6.11 documentation, *SAS/AF Software: FRAME Class Dictionary, Version 6, First Edition*, each class that makes up the Data Form and Data Table were documented in separate chapters. In the Release 6.12 online documentation, all these supporting classes are documented separately but also are available consolidated. In Release 6.12, this consolidation replaces the Data Set Entry Classes chapter in the 6.11 documentation. This change was made to enhance the usability of the Data Form and Data Table classes and provide more in-depth usability information.

In this consolidation, you can obtain information on any valid method that can be used with the Data Form or Data Table classes. Also, major usage issues such as controlling execution in the SCL, what is model SCL, creating default and customized layouts in a Data Form, moving widgets in a Data Form, making selections in a Data Table, and much more are covered.

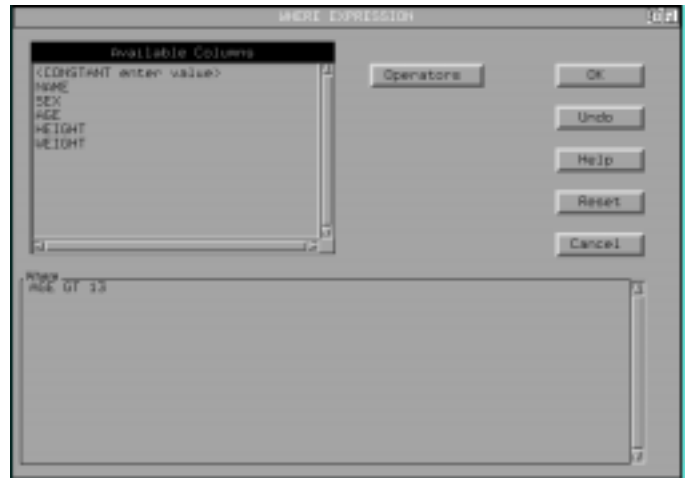
Additional Run Mode Pop-up Menu Items

Several actions have been added to the default pop-up menu available in run mode for the Data Form and Data Table objects. Here is a list of the new items you will see in addition to the other items already available in Release 6.11:

- Help
- Where
- Where Clear
- Browse/Edit Mode
- Record/Member Level Locking

The Help item was added to give the ability to retrieve information about a particular column at run time. To use this item, place the cursor over the column (Data Table) or widget representing the column (Data Form), and use the mouse's pop-up to select **Help**. This results in the column's name, its type, length, and format and/or informat being displayed on the message line of the frame. This information is useful when the user wishes to issue a WHERE clause, for example, and does not know a column's actual name because its label is displayed in the viewer.

The Where item was added to give the ability of quickly applying a WHERE clause to the Data Table or Data Form at run time. Selecting this item on the pop-up menu causes a where expression dialog window to display, which gives a point-and-click interface for building WHERE clauses.



Display 3 Interactive Where

The Where Clear item was added to give the ability of releasing any active WHERE clauses on the Data Table or Data Form. When this item is selected, all temporary WHERE clauses are cleared immediately. If no WHERE clause is in effect, this item is grayed on the pop-up menu.

The Browse/Edit Mode selection was added to give the ability to quickly toggle between browsing and editing the data displayed in the Data Form or Data Table. If the Data Form or Data Table is in edit mode, then the pop-up menu selection is Browse Mode; if the Data Form or Data Table is in browse mode, then the pop-up menu selection is Edit Mode. If the table is opened with the BRONLY data set option as a parameter to the `_SET_DATASET_` method, then you will not be able to toggle to edit mode.

continued on next page

The Record/Member Level Locking selection was added to give the ability to quickly toggle between Record Level Locking, where each row in the table is locked separately, to Member Level Locking, where the entire table is locked by the Data Table or Data Form. If the table is open with record level locking, then the pop-up menu selection is Member Level Locking; if the table is open with member level lock, then the pop-up menu selection is Record Level Locking.

Data Table Enhancements

The selection features in the Release 6.12 Data Table object have been greatly enhanced. A selection in a Data Table refers to highlighting rows, columns, or a range of cells that could span multiple rows and columns. In Release 6.11, the Data Table supported only single contiguous selections. Now in 6.12, Data Table supports two types of selections: Extended and Multiple.

Extended Selections

Extended selection refers to selecting or highlighting a range of contiguous cells, extending a selection from the active cell (starting anchor point) to the cell last selected with the mouse (ending anchor point).

In Release 6.12, the Data Table user can make extended selections the standard (6.11) way by clicking on the first cell in the range and dragging to the last cell in the range or by using single mouse clicks and the SHIFT key (new in 6.12). To use the SHIFT key to help make selections, click and hold the select mouse button down on the start cell until it is highlighted. Then hold down the SHIFT key and click in the last cell in the range to be selected. This is very useful when the anchor points are far apart in the table because the user can select the first cell, scroll to the last cell, and use the SHIFT-Click to select the entire range.

Multiple Selections

Multiple selection refers to the ability to select or highlight more than one noncontiguous range of cells. This attribute is not enabled by default for Data Table so you have to first set the MULTIPLE_SELECTIONS attribute using the `_SET_ATTRIBUTES_` method of the Table Editor Class. You could use the following code fragment in the INIT label of the SCL entry associated to the FRAME entry containing a Data Table named TABLE to enable multiple selections as well as the `select_columns` mode, which is discussed later.

```

init:
  dtattrl = makelist();
  /* enable multiple selections */
  rc=setntemc(dtattrl, 'y', 'multiple_selections');
  /* enable multiple column selections */
  rc=setntemc(dtattrl, 'y', 'select_columns');
  call notify('table', '_set_attributes_', dtattrl);

```

```

dtattrl = delist(dtattrl);
return;

```

After multiple selections are enabled, you can use them by making your first selection then holding down the CTRL key while making the rest of your selections. This means you would click and drag out the first area and release the mouse key. Then hold down the CNTL key and drag out one or several more areas.

The current group of selected cells is registered as a drag site by default. However, the drag operation is disabled when there is more than one selection active.

Data Table: Row, Column, and Label Selection Mode

In addition to the support for multiple selections, several other table attributes are provided to further customize how selections work in your Data Table.

Row selection mode, which is enabled using the `SELECT_ROWS` attribute, expands a selection of any cell, except column labels, into a selection of the entire row that contains the selected cell. If you wish to be able to select multiple, noncontiguous rows by this means, you also must enable the `MULTIPLE_SELECTIONS` attribute. The following code fragment shows an example of this:

```

init:
  dtattrl = makelist();
  rc=setntemc(dtattrl, 'y', 'multiple_selections');
  rc=setntemc(dtattrl, 'y', 'select_rows');
  call notify('table', '_set_attributes_', dtattrl);
  dtattrl = delist(dtattrl);
return;

```

Column selection mode, when it is enabled using the `SELECT_COLUMNS` attribute, expands a selection of any cell, except row labels, into a selection of the entire column. If you wish to be able to select multiple, noncontiguous columns by this means, you also must enable the `MULTIPLE_SELECTIONS` attribute. The following code fragment shows an example of this:

```

init:
  dtattrl = makelist();
  rc=setntemc(dtattrl, 'y', 'multiple_selections');
  rc=setntemc(dtattrl, 'y', 'select_columns');
  call notify('table', '_set_attributes_', dtattrl);
  dtattrl = delist(dtattrl);
return;

```

Label selection mode, when it is enabled using the `SELECT_LABELS` attribute, expands a selection of any row or column label into a selection of the entire row or column, respectively. If you wish to be able to select multiple, noncontiguous

rows and columns by this means, you also must enable the MULTIPLE_SELECTIONS attribute. The following code fragment shows an example of this:

```

init:
    dtattrl = makelist();
    rc=setntemc(dtattrl, 'y', 'multiple-selections');
    rc=setntemc(dtattrl, 'y', 'select-label');
    call notify('table', '-set-attributes-', dtattrl);
    dtattrl = delist(dtattrl);
return;

```

Two SCL methods can be used to return the coordinates of selected areas in a Data Table. `_GET_SELECT_` returns the coordinates of the last selected area, and `_GET_SELECTIONS_` returns a list of the current selections. These methods are useful as part of a process of obtaining the values of the selected cells so that some further action may be performed with the data values in those selected cells. The following code fragment shows an example of the SCL needed to create the list to contain the selected regions' coordinates and put out the coordinate list:

```

selections: /* the name of a pushbutton obj */
    sel-list=makelist();
    call notify('table', '-get-selections-', sel-list);
    call putlist(sel-list, 'Selections:', 5);
    sel-list=delist(sel-list);
return;

```

The following is an example of this code fragment's output list after the user has selected two separate areas, the first beginning at row2-column1 and ending at row3-column2 and the second beginning at row2-column4 and ending at row3-column4:

```

Selections: (
    1=(
        START-ROW=(
            1=2
        ) [. ]
        START-COLUMN=(
            1=1
        ) [. ]
        END-ROW=(
            1=3
        ) [. ]
        END-COLUMN=(
            1=2
        ) [. ]
    ) [. ]
    2=(

```

```

        START-ROW=(
            1=2
        ) [. ]
        START-COLUMN=(
            1=4
        ) [. ]
        END-ROW=(
            1=3
        ) [. ]
        END-COLUMN=(
            1=4
        ) [. ]
    ) [. ]
) [. ]

```

Additional Convenience Methods

Three new sets of methods make it easier for the application developer to control and query visual settings of the Data Form and Data Table as well as the individual attributes of each column.

_GET_VIEWER_ATTRIBUTE_

_SET_VIEWER_ATTRIBUTE_

These methods were added to the Data Set Data Model Class to provide the ability to set viewer attributes on a cell-by-cell basis depending on the value of each cell. Cell here also refers to the widget used to represent the column's data value in Data Form. These methods are only valid in the Model SCL entry because they are used in the context of the data values of the columns displayed either in the Data Form or the Data Table.

These new methods will greatly simplify such tasks as "traffic lighting" in a Data Form or Data Table. *Traffic lighting* refers to the use of colors, usually green, yellow, and red, to highlight values that fall into defined ranges. The following example shows the use of these methods to set the background color of a data cell of the column AGE based on its value:

```

init:
    age:
        /* if age is modified change the background */
        /* color for values greater than 16 to red */
        if (age gt 16) then
            call send(-viewer-, '-set-viewer-attribute-', 'age', 'bcolor',
                'red');
        else
            call send(-viewer-, '-set-viewer-attribute-', 'age', 'bcolor',
                'white');
return;

```

continued on next page

Table 1 shows the viewer attributes you can retrieve or change.

Attribute Name	Affects	For Data Form	For Data Table
BCOLOR	Background Color	X	X
FCOLOR	Foreground Color	X	X
BPATTERN	Background Pattern	X	
BDRCOLOR	Border Color	X	
FONT	Font	X	X
HJUST	Horizontal Justification	X	X
VJUST	Vertical Justification	X	
LTSOURCE	Light Source	X	
MARGIN	Margin		X
REVERSE	Reverse Video		X
PROTECT	Protection	X	X

Table 1

_GET _COLUMN_ATTRIBUTE_ _SET_COLUMN_ATTRIBUTE_

These methods were added to the Data Set Data Model Class to complement the original `_GET/SET_COLUMN_ATTRIBUTES_`. These new methods are very useful when you are only concerned with a few, specific column attributes. The methods take only the column name, the attribute to get or set a value for, and the value of the attribute to set, or in the case of the get, a return parameter to contain the value of the specified attribute. This greatly simplifies the attribute setting process because it does away with the need to create an SCL list and manipulate the list items as is necessary to use the original `_GET/SET_COLUMN_ATTRIBUTES_` methods.

The following example illustrates how to determine a column's type and then assign a maximum value limit to it:

```
init: /* of the Frame SCL */
call notify('table', '_get-column-attribute-',
           'age', 'type', var-type);
if (var-type = 'N') then
call notify('table', '_set-column-attribute-',
           'age', 'maxvalue', 99);
return;
```

_GET _DISPLAYED_COLUMNS_ _SET_DISPLAYED_COLUMNS_

These methods were added to give the developer of a Data Table application an easy way to control which columns are displayed and in what order they will appear. While the `_GET_DISPLAYED_COLUMNS_` method can be used in a Data Form to query what columns are visible, the `_SET_DISPLAYED_COLUMNS_` method has no effect on the Data Form.

The following example sets the order that columns will display in a Data Table and drops two columns, COL3 and COL4, from the display:

```
init: /*of the frame SCL */
call notify('table', '_set-displayed-columns-',
           'col 1', 'col 5', 'col 6', 'col 2');
return;
```

The New _COLUMN_NAME_ SCL System Variable

In Release 6.12, the Data Form and Data Table application developer has a new SCL system variable. The variable name is `_COLUMN_NAME_`, and it contains the name of the column for which the SCL column label is being run. In the DFINIT, DFTERM, INIT, MAIN, and TERM labels, the value for the `_COLUMN_NAME_` is `_BLANK_`.

When a column has been modified and the label for that column is driven for the model, the value contained in this SCL variable is the name of the column. The `_COLUMN_NAME_` variable must be specified as a character. The following example code fragment shows how this variable could be used to consolidate operations that should be performed when any of several columns are modified:

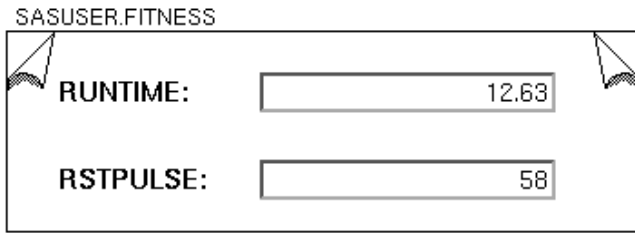
```
length -column-name- $8;

/* column1-5 are all column names */
column1: column2: column3: column4: column5:
/* set columns in error if any have the value 9 */
call send(-self-, '_get-column-value-', -column-name-,
value);
if (value > 9) then
call send(-self-, '-erroron-column-', -column-name-);
else
call send(-self-, '-erroroff-column-', -column-name-);
return;
```

Data Form Enhancements

In Release 6.12, Data Form uses earmarks to indicate that there are multiple Data Form pages. These earmarks are positioned

at the upper right and/or left of the page and are used to scroll from page to page within a Data Form.



Display 4 Data Form with Right and Left Earmarks

The earmark on the top right corner of the Data Form scrolls to the next page, and the earmark on the top left corner scrolls to the previous page. Additionally, if you double-click on an earmark, you will be taken to the last page (top right corner earmark) or the first page (top left corner earmark).

The attribute windows have also been modified so that the appearance of these earmarks can be customized. From the Data Form's Customize Form attribute window, you can set attributes such as whether the earmarks are displayed, the background color for the earmarks, and the outline color for the earmarks.

If you wish to control the color of the earmarks using frame SCL, four new form editor methods are now provided:

```
-GET-EARMARK-COLOR-
-SET-EARMARK-COLOR-
-GET-EARMARK-OUTLINE-COLOR-
-SET-EARMARK-OUTLINE-COLOR-
```

Moving Objects Between Pages and Data Forms

A powerful feature added to the Data Form in Release 6.12 is the ability to copy or move objects within a Data Form from page to page or from one Data Form to another. This is accomplished by the addition of three new build mode Data Form pop-up menu selections:

```
Cut to Form buffer
Copy to Form buffer
Paste from Form buffer
```

To use the cut or copy selections, position the mouse cursor over the object you wish to move or copy, and click the mouse pop-up button. Select the desired cut or copy selection from the pop-up menu.

Cut to Form buffer is used to delete the object from the Data Form and store it in a special buffer referred to as the Form Buffer. After the object is cut from the Data Form, you can paste it back into the same Data Form or another Data Form using Paste from Form buffer. Simply scroll to the page you wish to move the object to, and select Paste from Form buffer from the Data Form pop-up menu.

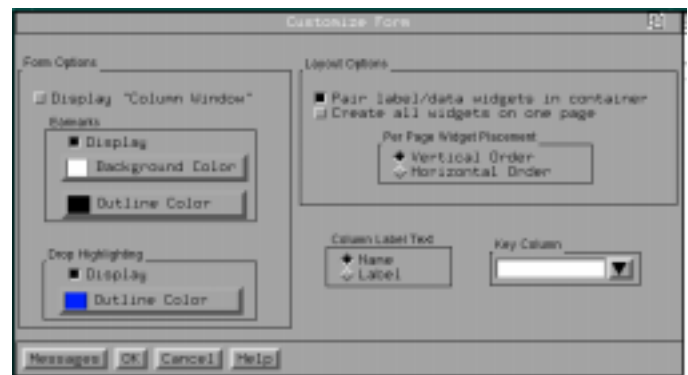
Copy to Form buffer does the same as Cut to Form buffer, but the original object is not deleted from the Data Form. This is useful to make a copy of an object in the same Data Form or another Data Form. This is one method you can use to quickly create repeated columns within a Data Form.

Creation of a One Page Default Layout

The default behavior of the Data Form when creating the initial layout of the form is to place as many objects that will fit on one page, using the region size, until the page is full. Then create subsequent pages to display the rest of the columns.

In Release 6.12, you have the ability to request that the Data Form place all columns on one page. If you request that all widgets be placed on one page in the Data Form and there are too many columns to fit in the region, the Data Form will provide scroll bars so that the viewer can be scrolled to see the columns outside the Data Form's region size.

Note the 'Create all widgets on one page' selection inside the Layout Options container on the Customize Form attribute window, shown in Display 5:



Display 5 Customize Form Window

Horizontal and Vertical Placement of Widgets When Creating a Default Layout

A new attribute setting in the Customize Form attribute window allows you to specify if you want the Data Form to create its layout with the columns placed in horizontal or vertical order. The default is vertical order.

Vertical

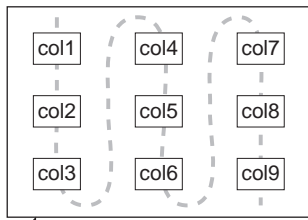
The placement of widget representations starts in the upper-left corner of the Data Form region and continues from top to bottom and left to right.

Horizontal

The placement of widget representations starts in the upper-left corner of the Data Form region and continues from left to right and top to bottom.

continued on next page

Vertical Order



Horizontal Order

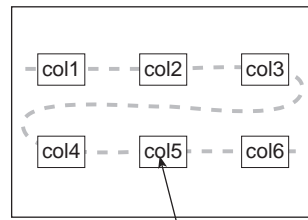
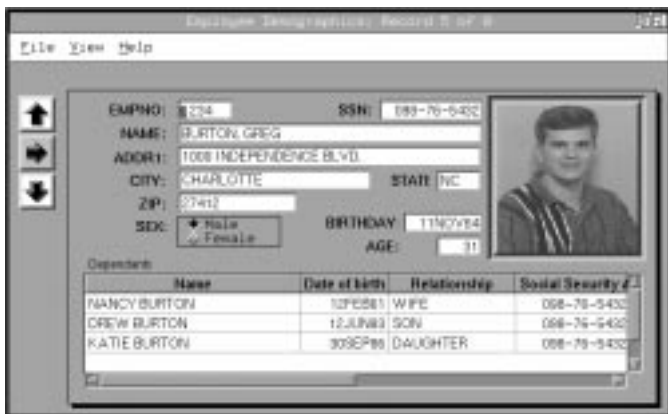


Figure 1 Vertical and Horizontal Widget Placement

For more information on horizontal and vertical placement of widgets when creating a default layout, refer to Customize Form Window in the 6.12 Data Form and Data Table Class online documentation.

Easier Creation of a Master-detail Relationship between a Data Form and Data Table

Master-detail refers to a configuration where master or key data is displayed row-by-row in a Data Form and detail records related to the master records are displayed in a Data Table. An example of this is a frame where an employee's personal data are displayed in a Data Form and their related dependents' data are displayed in a Data Table. In this case, the employee's personal data is the master data record, and the records that store data on their dependents is the detail data. The master data is linked to the detailed data with a field such as an employee ID.



Display 6 Master-detail Example Application

Applications with master-detail relationships between the Data Form and Data Table are very common. In Release 6.12, the ability to easily create this relationship with no SCL coding was added to the Data Form and Data Table. The creation of such a rela-

tionship is done through a combination of attribute window settings and the ability to link columns to objects within a Data Form.

The basic strategy for creating a master-detail relationship between a Data Form and Data Table is first to create a Data Form that uses the master table and then create inside it a Data Table that uses a related detail table. There must be a common key column between the two tables, for example, an employee ID or social security number.

This common key is defined to the Data Table through its attribute window settings. In the Data Table, the key column is defined in the Customize Table Setup window, shown in Display 7. After the Data Table is created inside the Data Form and the key column defined, the Data Form's Display Column window is used to link the key column in the Data Form to the Data Table object inside the Data Form.



Display 7 Customize Table Setup Window

When this relationship is created, the Data Form will execute a `_SET_WHERE_` or `_SET_KEY_` if the key column is an indexed column to subset the records in the Data Table at run time. Only detail records related to the current row in the Data Form are displayed in the Data Table.

Steps to link the Data Form and Data Table objects:

1. Create a Data Form using the table (data set) that contains the master data records. Also, specify a Data Form entry name to save the following customizations.
2. Create a Data Table inside the Data Form, using the table that contains the detail data records.
3. To link the Data Table to the Data Form, you will need to set the Key Column attribute in the Data Table:
 - a. From the Data Table attributes window, select `Customize table`.
 - b. Within the Customize Table window, select `Setup`.
 - c. Within `Setup attributes`, specify the common Key Column between the master and detail records.
 - d. Select `OK` twice to return to the frame.

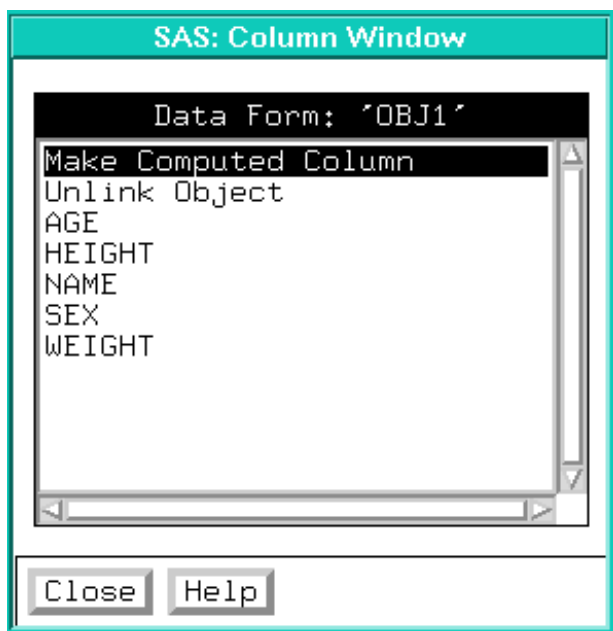
- Using the Column Window, link the key column in the Data Form to the Data Table object.

Step number 4 is accomplished as follows: Display the Column Window by using the Data Form's build mode pop-up menu and selecting Form->. Then select Display Column Window. Finally, highlight the key column in the Column Window, and drag it onto the Data Table object inside the Data Form.

Making a Computed Column Based on an Existing Widget

A new or modified widget inside a Data Form does not execute any labeled section in the model SCL unless it is linked to a column, either an existing column in the table or a computed column. For example, you might decide that you would like a push button object inside your Data Form that, when pressed, will cause some model SCL to execute. A good way to do this is to link the computed column to the push button and then create a labeled section in your model SCL by the same name as your computed column.

The Make Computed Column item was added to the Column Window in 6.12 to allow you to create quickly a computed column that is then automatically linked to the widget it is dragged and dropped onto.



Display 8 Picture of the Column Window

The name of the computed column will be the name of the widget, unless that name already exists in the Data Form. If the

name exists, a number is appended to the name of the widget to create a unique name, for example RADIO1, RADIO2, and so on. The name of the new computed column displays in the column window. When the widget associated with the computed column is modified or selected, the labeled section in the model SCL, which corresponds to the name of the computed column, executes. The type of the computed column is the same as the type of the widget.

Summary

In the 6.12 release of the SAS System, the SAS/AF FRAME Data Form and Data Table have been greatly enhanced to add power and flexibility to the creation and use of data entry and data viewing applications.

Both the Data Form and Data Table now have new run mode pop-up menu selections to help with the setting of where clauses, retrieving information about a column, and changing the open mode, as well as the locking level of the table.

Selections in the Data Table have been enhanced to allow multiple selections. Additional convenience methods have been added to aid in setting display attributes based on a column's value, setting and getting single column attributes, and setting and getting the displayed columns as well as their order.

A new SCL system variable, `_COLUMN_NAME_`, is now available to the application developer, which contains the name of the column for which its model SCL label is running.

The Data Form has been enhanced to provide earmarks for navigation in a multipage Data Form. The ability to easily copy or move widgets from page to page has been added to the Data Form as well. Also, the Data Form developer can request that the default layout be created on one page no matter what the size of the table is and, furthermore, can set horizontal or vertical default placement of the widgets on the page or pages. A very powerful enhancement to the Data Form is the ability to register master-detail relationships between a Data Form displaying the master data and a Data Table displaying related detail data. In the Column Window of the Data Form, a selection has been added so that the developer can create computed columns and link them to a widget in the Data Form by using drag-and-drop operations.

And finally, the 6.12 online documentation for these classes has been consolidated into one comprehensive chapter along with usage information on several major Data Form and Data Table topics. ●

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

Other brand and product names are registered trademarks or trademarks of their respective companies.