SAS/AF® Software - A Grand Tour

Tammy L. Gagliano, SAS Institute Inc., Chicago, IL

ABSTRACT

The class library in Release 6.11 of SAS/AF software has expanded to include many new visual and non-visual classes. This paper takes a closer look at some of the visual classes and what features they will add to your FRAME applications.

INTRODUCTION

Some of the new classes you will see listed when you go to fill a region on a FRAME are the following:

- Catalog Entry Viewer
- Command Push Button
- Data Form (experimental)
- Data Table
- Extended Input Field
- Extended Text Entry
- External File Viewer
- Input Field
- Input Field Label
- Image Icon
- Image Viewer
- OLE
- Organizational Chart
- Process Flow Diagram
- Toolbar
- Video Player
- Work Area

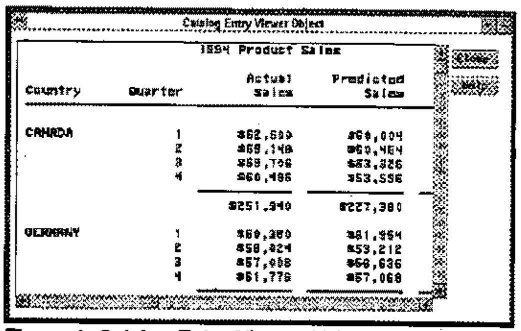
In addition to the new classes listed above, the Region Attribute window that opens for all objects in a FRAME has been significantly enhanced. New outline, title and color attributes have been added, allowing you even greater control over the appearance of each object on the window.

All of the above are described in more detail throughout the remainder of the paper. It should be noted, however, that the list of features and supported methods given for each class is *not* an exhaustive one. It serves only as an overview as to what each class has to offer. For more on each class, refer to *SAS/AF Software: FRAME Class Dictionary, Version 6, First Edition.*

Catalog Entry Viewer

No more extended tables! At least not for applications where all you need to do is browse or edit output that's stored in a catalog entry. In previous releases, to display output in a FRAME, the extended table object was used for this purpose.

The catalog entry viewer automatically displays output from SAS® catalog entry sources such as SCL, LOG, OUTPUT, HELP or SOURCE entries in a region on a FRAME. Scrolling is handled automatically with no code having to be written by the developer.



```
                     1994 Product Sales

                                  Actual      Predicted
Country        Quarter             Sales          Sales
--------------------------------------------------------
CANADA               1           $62,600        $64,004
                     2           $65,148        $60,464
                     3           $59,706        $53,326
                     4           $60,486        $53,556
                               ---------      ---------
                                $251,940       $227,380

GERMANY              1           $60,280        $61,964
                     2           $58,424        $53,212
                     3           $57,058        $58,636
                     4           $61,776        $57,068
                               ---------      ---------
```

Figure 1: Catalog Entry Viewer Object

- Options such as the display of a column ruler across the top of the output, line numbers, and control over text capitalization are available.
- Custom colors can be defined to control components of the output area such as the text, column ruler, and line numbers.
- A font can be specified to display the text.
- Scrollbars, both vertical and horizontal, can be turned on enabling the user to scroll through output that spans beyond the region.

Methods can be used to further control the object's display and behavior. For example,

- Most, if not all, of the attributes assigned in the object attribute window can be queried and modified using the appropriate methods, such as _GET_TEXT_COLOR_ and _SET_TEXT_COLOR_.
- You can specify a new catalog entry to be displayed using _SET_ENTRY_.
- Marking text in the viewer can be controlled using _ENABLE_MARKS_. Row and column information about the area marked can then be retrieved and further processed using _GET_MARK_.
- Hotspots can be assigned to a particular area on the report using _REGISTER_HOTSPOTS_. _GET_REGISTERED_HOTSPOTS_, in return, supplies information about hotspots currently registered on the viewer.
- For information about where and what the user clicks on in the viewer (e.g. row and column, actual text or word, name of hotspot), _GET_CLICK_INFO_ and _GET_CLICK_AREA_ can be used.
- To change the cursor shape as the user passes over specific areas in the viewer you can use the following Widget class methods, _CURSOR_TRACKING_ON_ and _SET_CURSOR_SHAPE_.

Command Push Button

The push button object, first available in Release 6.08, is an object that you commonly see on many user interfaces. It is designed to look like a rectangular button, labeled with text and can initiate an action when the user selects it. It is fixed in size and the background color of the button cannot be controlled. On some systems, the text color can be changed but the back-ground color is always inherited from the FRAME entry's background color.

In contrast, a command push button object does have resizing capability. Its shape can be rectangular or square, thin or thick and on some operating systems you have complete control over the background color of the button.

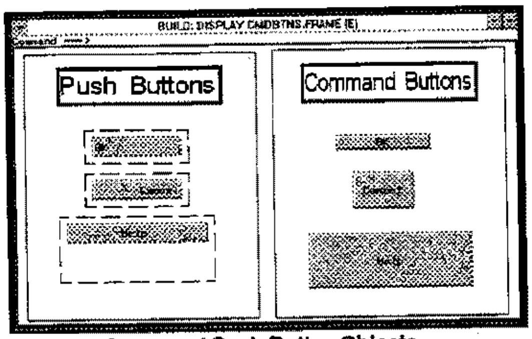


Figure 2: Command Push Button Objects

Other differences are

- The text within a command push button object will always remain centered as you grow and shrink the object.
- The color of the text within the command push button itself inherits the SASCOLOR foreground setting. Altering the text color for a push button object is an option in the object's attribute window.
- The value of a command push button object when selected takes on the value of the text that is displayed on the button. This differs slightly from the functionality provided by a push button object which can take on a value different than the text displayed on the button when selected. That value can be assigned in the object's attribute window and can be either a numeric or character value.

Data Form (Experimental)

One of the new data tools, a data form object, graphically displays data one row at a time and uses SAS/AF objects to display the columns in a SAS data set. When designing a data form, each column in the table can be displayed using different objects that best represent the data.

For example, in a data entry application, instead of typing the name of a department into a data entry field, a list box object can be used allowing the user to simply select the department name. Similarly, sales numbers can be displayed using critical success factor (CSF) objects which use color ranges to indicate where the numbers are with respect to the sales goals.

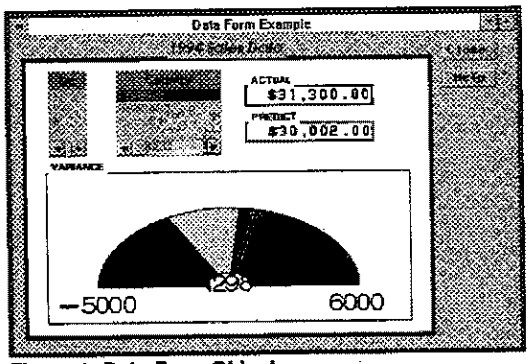


Figure 3: Data Form Object

- The tool allows browsing or editing of SAS data sets a single row at a time.
- Computed columns can be added.
- Scrolling through the data is automatically handled by the tool using pop-up menu selections at execution time.
- To display only a portion or subset of the data, where clause conditions can be specified in the object's attribute window.
- Editing options also allow control over the addition of new rows as well as allow duplication or deletion of existing rows.
- Each column can be displayed using a different object. The class used to represent the data can be specified through the attribute window or by working interactively with the object on the FRAME entry.
- At build time, the order, size and region attributes in which the columns are displayed can be changed by using the mouse to move or resize as well as by opening the object or region attribute windows to change the object's attributes or appearance.
- General Data Form options include surrounding both the data and its corresponding label with a container object and specifying whether to use the column name or label as the label for the widget on the frame.
- Other individual column customizations include control of text capitalization, specifying whether or not the column is required or protected. Justification of the data in the column can be specified as well as maximum, minimum and initial values can be supplied.
- Different fonts can be used to display both the data in each column and its column label.
- Colors for each column's data, label and error conditions can be specified. The foreground and background color of each can be customized.
- All customizations can be saved to a common location, a DATAFORM entry, for use by other data tables or data forms.
- The Data Form class can be subclassed to override specific method behavior. Its behavior can also be overridden for a particular instance of

the class using the new _SET_INSTANCE_METHOD_ method.

Customizing the Object Using Methods

When you want to change the data set being displayed, subset or sort the data, or search for specific values in the table, you can use methods in the FRAME's SCL to query the object and control its behavior at run time. For example, you can use

_SET_DATASET_
sets the data set being accessed by the object.

_FIND_ROW_, _REPEAT_FIND_ROW_
enables you to pass in the string you are searching for and the method returns the row number(s) of the next observation that meets the find request.

_SET_WHERE_
sets where clauses on the table.

SORT
sorts the table by one or more columns.

_GET_COLUMN_TEXT_,
_GET_COLUMN_VALUE_
returns the actual character text or numeric value for that particular column. This is useful when you want to perform specific actions based on the value of a specific column (for example, traffic lighting).

_SET_COLUMN_TEXT_,
_SET_COLUMN_VALUE_
sets the character text or numeric value for a particular column.

_GET_CURRENT_WIDGET_INFO_
returns information about the currently active widget.

_GET_LINKED_COLUMN_
returns the name of the column linked to the specified widget.

_GET_CURRENT_ROW_NUMBER_
retrieves the row number of the current row.

_GOTO_COLUMN_, _GOTO_ROW_
HSCROLL
all control scrolling of the data form either to a specific column, row, or page.

_SET_KEY_, _KEY_COUNT_
sets a key for quick retrieval of rows in the table and returns the number of rows that meets the current key.

The Model SCL

Rules for performing data validation or defining computed columns are stored separately from the FRAME SCL. These rules typically go in what's referred to as the Model SCL. The Model SCL is a SCL entry stored separately from the FRAME SCL and is assigned to the DATAFORM entry in the object's attribute window as the DATAFORM SCL.

Storing this logic separate from the FRAME's SCL enables other data forms or data tables to use these same specifications simply by referencing the same SCL entry.

Some of the methods that are commonly used in the Model SCL are

_GET_DATA_BACKGROUND_COLOR_,
_GET_LABEL_BACKGROUND_COLOR_
return the default colors that are used to display the background for *all* data widgets or labels.

_SET_DATA_BACKGROUND_COLOR_,
_SET_LABEL_BACKGROUND_COLOR_
sets the default color used to display the back-ground for *all* data widgets or labels.

_SET_DATA_FONT_,
_SET_DATA_COLOR_,
_SET_LABEL_FONT_,
_SET_LABEL_COLOR_
sets the default font and foreground color used for *all* of the data or label widgets on the data form.

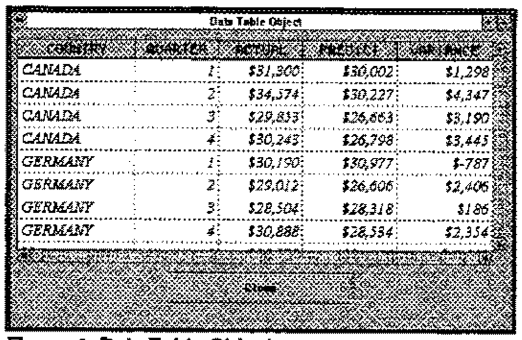
_ERROROFF_COLUMN,
_ERRORON_COLUMN_
controls error handling on a column.

_PROTECT_COLUMN_,
_UNPROTECT_COLUMN_
controls the protection mode of a column.

The above methods can be used in the FRAME's SCL also; however, by separating the logic that is specific to the data itself, such as error handling or protecting and unprotecting of columns based on values entered in other columns, these same data rules can then be applied to other data table or data form objects through the use of the same Model SCL.

Data Table

A data table object can be added to a FRAME application and displays data in a row and column format, much like a spreadsheet. Multiple rows of data are displayed at a time.



COUNTRY	QUARTER	ACTUAL	PREDICT	VARIANCE
CANADA	1	$31,300	$30,002	$1,298
CANADA	2	$34,574	$30,227	$4,347
CANADA	3	$29,833	$26,663	$3,190
CANADA	4	$30,243	$26,798	$3,445
GERMANY	1	$30,190	$30,977	$-787
GERMANY	2	$29,012	$26,606	$2,406
GERMANY	3	$28,504	$28,318	$186
GERMANY	4	$30,888	$28,534	$2,354

Figure 4: Data Table Object

- The tool allows browsing or editing of SAS data sets in a tabular format.
- Computed columns can be added.
- Scrolling through the data, both vertically and horizontally, is automatically handled by the tool.
- To display only a portion or subset of the table, where clause conditions can be specified in the object's attribute window.
- Editing options also allow control over the addition of new rows to the table as well as allowing duplication or deletion of existing rows.
- Different fonts can be used to display both the data in each column and its column heading or label.
- Colors for each column's data, label and error conditions can be specified. The foreground and background color of each can be customized.

- Individual column customizations include control of text capitalization, specifying whether or not the column is required or protected. Justification of the data in the column can be specified as well as maximum, minimum and initial values can be supplied.
- Extensive customizing of the table layout can be done through table options such as controlling the border, margins, grid style, and text justification.
- Other general table options include displaying row and column labels, growing the rows or columns to fit the region, allowing the user to resize columns at execution time, support for drag and drop and more.
- The order in which the columns are displayed can be changed by dragging a column to a new location in the table and dropping it during both build time and run time.
- Column resizing can also be done by grabbing the column's border and growing or shrinking it to the desired size.
- Printing is supported for the current display or for the entire table.
- All column customizations can be saved to a common location, a DATAFORM entry, for use by other data tables or data forms.
- The Data Table class can be subclassed to override specific method behavior. Its behavior can also be overridden for a particular instance of the class using the new _SET_INSTANCE_METHOD_ method.

Customizing the Object Using Methods

Like the Data Form class, the Data Table's behavior can be controlled through the use of methods in the FRAME's SCL. Some of the methods available and that are typically placed in the FRAME's SCL are

_SET_DATASET_
which enables you to change the data set that is being displayed in the data table dynamically.

_SET_WHERE_
sets where clauses on the table.

SORT
sorts the table by one or more columns.

_GET_COLUMN_TEXT_,
_GET_COLUMN_VALUE_
returns the actual character text or numeric value for that particular column. This is useful when you want to perform specific actions based on the value of a specific column (for example, traffic lighting).

_SET_COLUMN_TEXT_,
_SET_COLUMN_VALUE_
sets the character text or numeric value for a particular column.

_PRINT_SETUP_
brings up a dialogue to set printing options.

PRINT
prints the table.

_GET_ATTRIBUTES_, _SET_ATTRIBUTES_
returns and alternately sets attributes for the data table object such as the display of column labels versus column names, the use of a grid in the table, the resizing of columns and more.

_FIND_ROW_, _REPEAT_FIND_ROW_
enables you to pass in the string you are searching for and the method returns the row number(s) of the next observation that meets the find request.

_SELECT_ROW_, _CLEAR_SELECT_
useful when using a data table as a selection list where you want to highlight an entire row as if selected or clear (deselect) a row.

_HOLD_COLUMN_, _RELEASE_COLUMN_
holds or releases a given column (and all visible columns to its left) when left and right scrolling occurs.

_GET_TOPROW_, _SET_TOPROW_
returns or sets the topmost row in the table.

_GOTO_ROW_, _GOTO_COLUMN_
goes to a specified row or column.

There are many other methods which control table attributes such as the default margins, vertical and horizontal adjustment within the cells, grid color, style and width and more.

The Model SCL

The Data Table class shares the same rules and methods for its Model SCL as the Data Form class. Refer to the section under **Data Form** labeled *The Model SCL* for more detail.

Extended Input Field

There are several objects that can be used as data entry fields on a FRAME entry such as

- Text Entry Field (supported in previous releases)
- Extended Text Entry Field (new)
- Input Field (new)
- Extended Input Field (new).

Each widget offers some unique features over the others. The main difference all three new additions have over the text entry field object previously available is that these new objects are graphical components. This brings greater control over the placement of the object on the FRAME entry. The text entry field can only be aligned on a window to the nearest column or row specification. The graphical objects can be placed to the nearest pixel which allows you more flexibility when dealing with limited real estate of a window. Fields can be placed closer together; more objects can be added to a window.

The extended input field supports automatic scrolling, allowing the user to enter values longer than the display length of the field. The widget also allows the use of host fonts for displaying the data in the field. Marking of text is supported as well as paragraph-style word wrap. In addition to support for the standard SASCOLORs, the extended input field widget also supports user-defined colors.

If you want the object to automatically handle field validation either by specifying it as a required field or by providing a list or range of valid values, you will want to continue using the text entry object. That functionality is automatically built into the object and can be specified in the object's attribute window.

Extended Text Entry

A data entry field created using the Extended Text Entry class can contain only character data. This differs from the other three data entry widgets (refer to **Extended Input Field** section for a list of other data entry field objects available). No control over format or informat can be specified; however, initial values can be set for each row. Using any of the other data entry objects which allow multiple rows, only an initial value for the first row can be specified.

Additional editing and display features are available which offer greater font control with the ability to resize the font to fit the region either by closest font size or a scaleable font and the ability to resize the region to fit the font. The field also supports marking of text, scrollable text and paragraph-style word wrap.

External File Viewer

Both the External File Viewer and Catalog Entry Viewer are subclasses of the same class, the Text Viewer. As such, many of their methods and options are the same. The main difference between the two is that an external file viewer object displays external files within a region on a FRAME entry instead of catalog entries. An external file is just an ordinary text file, such as an ASCII file.

The options you can specify in the Object Attribute window when building an external file viewer object are almost identical to those supported by the catalog entry viewer. You can control the display of line numbers, a column ruler, scrollbars and whether or not to display the text in all capital letters.

You can also change the color of items on the viewer and specify a different font to be used. The differences between the two viewers are:

- A fileref or actual filename of the external file is given as the data source to be displayed in the external file viewer.
- Carriage control is also supported in the external file viewer.

The object methods are also similar to those listed for the Catalog Entry Viewer although some differences are apparent due to the different data sources each class supports.

For example when viewing external files, carriage control settings for the file are important to recognize. Using the _GET_CARRIAGE_CONTROL_ and _SET_CARRIAGE_CONTROL_ methods, the carriage control attribute can be queried and altered dynamically. An example of when you might need to change this attribute is when using the _SET_FILE_ method to dynamically change the name of the file being displayed at run time. At that point, you might also need to change the type of carriage control handling to be used for the new file.

Input Field

This object can be used to display one-line text fields that can accept user input and display text. Character or numeric data is allowed. Control over format and informat is also provided. It is a graphical object; however, it will automatically provide an alternate, non-graphical definition when displayed on a non-graphics terminal.

Other functionality this object provides is the ability to mark text entered in the field and setting its field length to be longer than the actual window display length. This optimizes the limited real estate of a window by allowing the user to type values longer than the display length of the field. The field will automatically handle scrolling.

Using the input field object, the font cannot be changed. If this functionality is desired, you'll want to use either the extended text entry or extended input field object. The background color for all of the new data entry objects can be specified in the object's attribute window. Unlike the other data entry objects, the input field object does not provide customization of the text color through its attribute window but is inherited from the SASCOLOR foreground setting instead.

Input Field Label

This object is typically used to display text next to another object as a description or field label. It displays noneditable text, just like the text label object. However, the input field label is a graphical (pixel-based) object. This provides greater control over the placement of the object on the FRAME entry compared to the text label object which is character-based and can only be placed to the nearest character cell on the window. If an application is built using this object and is then displayed on a non-graphical device, the object will automatically switch to using the text label object as its alternate definition.

Image Icon

The Image Icon class is a graphical version of the Icon class. It uses catalog entries of type IMAGE to display the pictures. It provides the same functionality as the icon object, plus some additional capabilities such as changing the image size and label font.

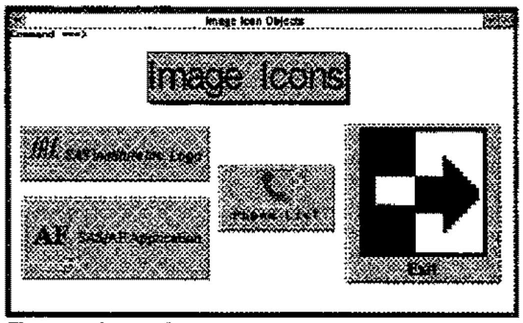


Figure 5: Image Icon Objects

Image Viewer

This class enables you to read bitmapped images into a region on a FRAME entry. The image source can be an external file, a fileref that points to an external file, or a four-level name of an IMAGE catalog entry. You can

- scale the image size of the region.
- preserve the aspect ratio for the image. If the aspect ratio is not preserved, the scaled image will be contorted to fit the region's size.
- adjust the region size by specifying a width and height in pixels.
- use scrollbars to scroll both vertically and horizontally to view an image that is scaled larger than its region size.

- set an auto-correct image option which then automatically attempts to color-correct an image so that it can always display on the current device.
- use methods such as _READ_CATALOG_, _READ_CLIPBOARD_, and _READ_FILEPATH_ to ad an image from another source and dynamically change the image currently being displayed.

OLE

The OLE class enables you to include an Object Linking and Embedding (OLE) object in a FRAME entry. There are three types of OLE classes: the OLE Insert Class, the OLE Paste Class, and the OLE Read Class.

Release 6.11 now fully exploits OLE 2.0. New functionality includes support for

- drag and drop of OLE objects within the same SAS/AF FRAME or across windows.
- visual editing which allows you to edit an OLE object in the container application versus having to open a separate window to edit its contents.
- automation of objects which provides a mechanism to control or script other applications using new SCL methods.
- OLE control (OCX) support which enables the use of third party OCXs within a FRAME application.

Organizational Chart

The Organizational Chart class creates hierarchical charts from data stored in SAS data sets or SCL lists. A chart can be created from any kind of hierarchical data such as an employee database, a file or library directory, project management data, or class hierarchies.

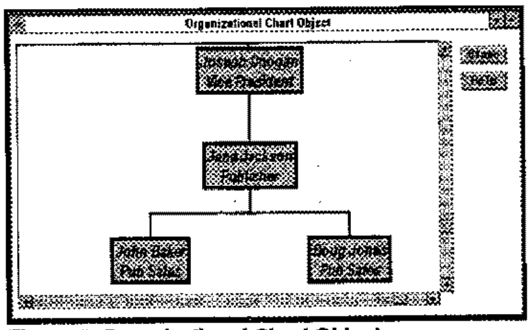


Figure 6: Organizational Chart Object

Using an organizational chart object, you can

- display hierarchical trees symmetrically or asymmetrically.
- change the size, color, font and borders of the nodes. Each node on the tree is a FRAME object.
- display catalog or host file images inside the nodes.
- display text horizontally or vertically.
- change tree orientation from left-to-right or top-to-bottom.
- change visual aspects of the tree at execution time using the pop-up menu or by opening the object's attribute window.
- print text nodes.
- change the attributes of the lines used to connect the nodes such as their color and width or hide them completely.

Process Flow Diagram

Building a process flow diagram is easy using this new object's interactive editing environment. You can design a diagram composed of graphics objects, such as images or SAS/GRAPH® output objects, along with other standard symbols and drawing tools by dragging and dropping them where you want them in your diagram. For example,

- to add a symbol onto the drawing area, simply drag a symbol from the Symbol Palette on the toolbar and drop it where you want it to go.
- to connect the nodes with arrows, drag from the inside edge of one node to the inside edge of another node.
- to add text to an item on the diagram, simply select it and begin typing.
- if you want to move a node, simply drag it to its new location. All connecting arrows will move along with the node as well.

Using items from various context sensitive pop-up menus, you can

- change the font.
- specify different colors to be used for each node, arrow or text.
- control the style of line used as well as its width.
- compress the diagram in cases where the diagram is larger than the viewing area, enabling you to see more of the diagram without having to scroll.
- select multiple elements in the diagram and use the *condense* option which will automatically create a subdiagram enabling you to manage large or complex diagrams by breaking them into smaller pieces that users can access by drilling down.

For further customization, SCL programs can be written that execute when a specific node or item in the diagram is selected for drill down applications. For example, design a process flow diagram to depict a specific manufacturing process. When one of the nodes or elements in the diagram is selected, an SCL program can be executed that displays a control chart of the most recent data that the node represents.

Conversely, the diagram can be data-driven by using one or more of the object's methods. To extend the previous example, the nodes in the diagram can function as *traffic lights*. Methods can be used to dynamically change the color of a node based on values in the underlying data that the node represents.

Using the object's methods, you can

- dynamically create new nodes and arrows by using the _ADD_NODE_ and _ADD_ARROW_ methods.

- retrieve information about a specified node or arrow, or about other currently selected elements in the process flow diagram using the _GET_INFO_ method.
- change the default settings for attributes such as font, node color or line style, using the _SET_DEFAULTS_ method. To change these attributes for specific arrows or nodes, use _SET_INFO_ which expects the name of the element to be passed in as an argument.

Once created, the process flow diagram can be printed or saved in several different forms. A graphical representation of the diagram can be saved in bitmap format to a file or to the clipboard or to a SAS catalog as a GRSEG or IMAGE entry. Or, you may decide to save the structural information about the diagram to a SAS data set or SLIST catalog entry. Saving it in this format enables you to use these files as input to other process flow diagrams.

Toolbar

A toolbar consists of one or more buttons containing a label, an image, or both. Each item on the toolbar can be defined to perform a specific action when the user selects and/or deselects the item.

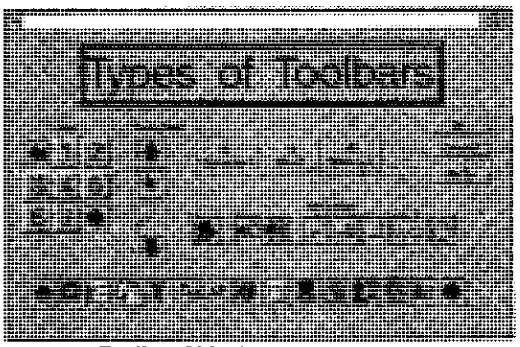


Figure 7: Toolbar Objects

In addition to selecting images, labels and actions for each button, you can

- design toolbars that are vertical or horizontal in the frame.
- control the toolbar layout and provide automatic scrolling capability when the toolbar cannot display all the buttons at once.
- select fonts for labels.
- specify status line help for each button.
- specify whether buttons act like push buttons, check boxes, or radio buttons.
- save toolbar definitions in SCL lists so they can be reused.
- design the application using drag and drop since the toolbar class automatically supports this functionality both in build mode and at run time. In build mode, you can change the order that the items appear on the toolbar by dragging a button over to its new location. At run time, the user can also drag an item from the toolbar and drop it on another object on the FRAME entry to perform certain actions.
- retrieve the index number, and optionally the name, of the button selected at run time using the _GET_LAST_SEL_ method. The index number corresponds to its position in the toolbar.
- gray or protect individual items on the toolbar or the entire toolbar itself using the _GRAY_ method. Similarly, _UNGRAY_ returns the toolbar or button to a selectable state.
- dynamically build a toolbar or change its settings at run time using methods such as _SET_DISPLAY_TYPE_, _SET_IMAGE_, _SET_LABEL_, _SET_FONT_, _SET_SPACE_AFTER_ and more.

Video Player

Using this class, you can add multimedia to your FRAME applications! You can play video clips and control playback features such as the portion of the video that is played, the volume and balance of the audio, playback speed and size of the picture.

As with all the objects in a FRAME application, its behavior can be controlled using methods. You can

- conditionally start and stop the video using _PLAY_ and _STOP_.
- use _PLAY_SEGMENT_ to play only a portion of the video by specifying the starting and ending frame numbers.
- change the video being played using _SET_TEXT_ to specify a different video filename.
- tell the video to play once or play repeatedly until stopped using _SET_PLAY_MODE_.

Work Area

Using this class, an area is created in a FRAME entry in which you can place other objects. A work area object is scrollable, therefore, it can be larger than the FRAME itself and provide a virtually unlimited desktop area.

The SAS/ASSIST® Desktop (Figure 8), new in Release 6.11, uses the Work Area class as its underlying technology and serves as an example of the type of user interface or application that can be designed using work areas.

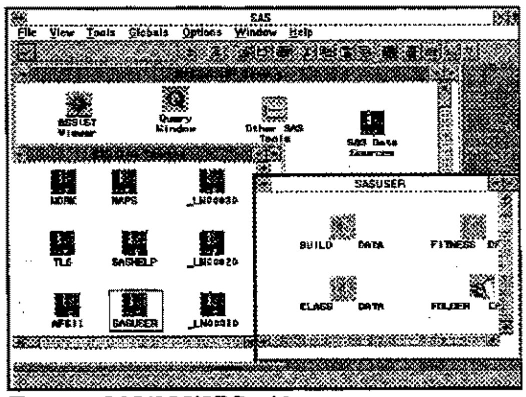


Figure 8: SAS/ASSIST Desktop

The items in a work area may represent data (like an icon that represents a SAS data set) or tasks (like printing or graphing) or applications (like producing daily reports). In a work area, you can

- perform tasks on or with the objects by selection, with drag and drop functionality, or through user-defined pop-ups.
- dynamically create and position new objects.
- save your current work area layout so you can return to it.

Enhanced Region Attributes Window

This new window provides greater consistency and more control over the appearance of any object added to a FRAME entry. Its options can be broken down into three main areas:

Outline
provides choices of outline styles, control over the light source and width of the outline. The object can also be designed to act like a button. You can control its behavior by setting its button behavior to perform like a

- push button, which when the user clicks once, will give the effect of a button being momentarily pushed in (or selected) and then pushed out (unselected).
- check box, which is similar to the push button except that the button acts like a toggle, switching from selected and unselected states each time the user clicks on the region.
- radio button, which when selected, can no longer be unselected. The button can only be unselected by the application through a method call. This feature is suited for applications where the developer wants to manage their own radio box.

If you've chosen one of the button behaviors above, you can also set an option to have it initially selected and provide a description which is used when the user selects the object for additional help on the item.

Title
enables you to enter a text string to display in the top region border. You can control its font, text justification and the vertical adjustment.

Color
controls both the outline and background color of the object's region. SASCOLOR options are available and on some host, custom colors can also be defined.

CONCLUSION

The best way to see what these new classes have to offer is to try them out! Use them to build objects and explore their object and region attribute settings. Use the SCL debugger to test the various methods that each class supports to see what information can be dynamically retrieved and set at execution time.

This paper has only skimmed the surface as to the feature set provided with Release 6.11, specifically what some of the new FRAME classes have to offer. The class library has increased significantly. And with the addition of the Composite class, which enables you to design your own objects, the control that you have over the design and functionality of your FRAME applications is virtually limitless!

REFERENCES

SAS Institute Inc., *SAS/AF Software: FRAME Class Dictionary, Version 6, First Edition*, Cary NC: SAS Institute Inc., 1995, 1155pp.

AUTHOR

Tammy L. Gagliano
SAS Institute Inc.
Two Prudential Plaza, 52nd Floor
Chicago, IL 60601
phone: (312) 819-6824
email: SASTLG@UNX.SAS.COM