Paper 20

An Application Developer's Roadmap for Moving to Version 7 in SAS/AF

Steven A. Wilson, MAJARO InfoSystems, Inc., Santa Clara CA

Abstract

Development of SAS/AF® applications in Version 7 will be much different than in the current Version 6 of the SAS System®. Even the name of the AF coding language has changed, from Screen Control Language to SAS Component Language, representing a more robust realization of an object oriented development environment. This change will impact all SAS/AF development groups as Versions 7, 8, and 9 are installed at sites throughout the globe.

MAJARO InfoSystems, as a developer of SAS-based clinical software that is installed at many biotech and pharmaceutical companies, is acutely affected by these changes. Will our existing products still work? How does a new V7 component-based frame look in an application comprised of V6 widget-based frames and program entries? How much of our software will need to be rewritten to implement the new version of SAS?

This paper will attempt to summarize the changes that will be encountered as you move from SAS V6 Frame development to the new V7 development environment.

Introduction

This paper begins with an overview of the most important enhancements to grasp with V7 SAS/AF. The remaining sections of this paper discuss the author's first experiences with executing and upgrading SAS/AF applications under V7.

The most important changes in SAS/AF are these:

- Environment Changes
  The user interface for applications development is much different – and greatly improved!
- Object Properties and Communications
  Object properties have greatly streamlined the applications development process, which makes working with your objects much easier. Communication between objects on your frame is also accomplished much more easily via object attributes. These include:
  - Drag and Drop
  - Attribute Linking
  - Model / View
- SCL Coding Improvements
  Use of Object Oriented "Dot Syntax" makes coding easier in your applications, and at the same time makes the code is easier to read.
- Class Editor
  Beyond the scope of this paper, the Class Editor allows developers to more easily create and maintain SAS/AF classes.

Objects Have Become Components

In V7, we are presented with an entire new set of objects, both visual and non-visual, with which to work. These objects are called *components*.

The set of V6 objects with which you are familiar is still available in V7. However, there are many advantages to using the new V7 components in your application.

There are two types of V7 components:

1) Controls
   These components are visible on the frame and are the objects formerly known as widgets. The good news is that these components are now pixel-based and use host system fonts. Thus, these new components have a much cleaner, crisp, and host-specific look on your application. For this reason, you especially want to avoid using V6 text-based objects in your V7 applications.

2) Models
   These components are not visible on the frame, but instead are used to describe data, usually for display by a Control component. These components enable the Model-Viewer paradigm and are discussed later in this paper. Let me just say here that the use of these objects has been greatly simplified in V7 and I strongly encourage you to utilize these objects in your applications.

Most of the V7 components are simply upgraded V6 objects. In some cases V7 components make life simpler; for example, replacing the Input Field and Text Entry objects with a single Text Entry component.

Common V6 composite objects built by many developers combined an Input Field widget with a Control widget, providing an arrow to click to obtain a selection list of acceptable values for the Input Field. These constructs exist as native V7 controls:

Combo Box control

This control produces a drop-down list of acceptable choices for entry into the field.

Spin Box control

This control allows the user to scroll through either a range of numeric values or a set of character values.

These tables are my attempt to correlate V6 objects with the V7 components. Most correspond one-to-one, but others did not evolve so simply. Also, there are a few components that are simply new constructs, while others are still only available as V6 objects.

V6 Objects NOT replaced
Catalog Entry Viewer
Data Form (can be populated with V7 controls)
Data Table (V7 control may be assigned to a column)
Extended Table
External File Viewer
Hotspot
Image
Image Icon
OLE – Insert Object / Paste Special / Read Object
Process Flow Diagram
Slider
Tab Layout Object
Toolbar
Video Player
Work Area

V6 Objects Replaced with V7 Components	
Version 6 Widget	**Version 7 Component**
Block	Push Button Control
Check Box	Check Box Control
Command Push Button	Push Button Control
Container Box	Container Box Control
Control Object	Combo Box Control Spin Box Control
Critical Success Factor	Critical Success Factor Control
Extended Input Field	Text Entry Control Text Pad Control
Extended Text Entry	Text Entry Control Text Pad Control
Graphic Text	Graphic Text Control
Graphics	Chart Control Histogram Control Pie Control Scatter Control
Icon	Push Button Control
Input Field	Text Entry Control
Input Field Label	Text Label Control
List Box	List Box Control
Map	Map Control
Organizational Chart	Tree View Control *
Push Button	Push Button Control
Radio Box	Radio Box Control
SAS/GIS Map	Map Control
SAS/GRAPH Output	Graph Output Control
SAS Libname Listbox*	Library Selector*
SAS Members Listbox*	Member Selector*
Scrollbar	Scrollbar Control
Text Entry	Text Entry Control Text Pad Control
Text Label	Text Label Control

New Version 7 Objects
Desktop Icon Control
Catalog Entry Selector Control*
Dual Selector Control*
List View Control*

* Experimental, unsupported objects that do not appear in the default resource catalogs. To access these controls, you must add the appropriate resource from SASHELP.FSP to the Components window:
  - AFCOMPONENTSPLUS.RESOURCE
  - EXPERIMENTALAFCOMPONENTS.RESOURCE

Resource catalogs contain a set of objects. The default Resources are: 1) V7 components and 2) V6 objects.

Component Attributes

In V7 the behavior of an object is more likely to be controlled by attributes rather than methods. These attributes are stored with the component and essentially replace instance variables. Attributes contain more information than instance variables.

V6 Object Attribute screens have been replaced by a consistent Properties window that displays V7 Component Attributes in a data table. Of all the new windows available in the V7 build environment, the Properties window will be the most used. This window has two parts:

1) A tree diagram displaying both visual and non-visual components on the frame, as well as the frame component itself. Right-click on the component or component property to open a popmenu of actions, including access to the class help for the component.

2) A data table displaying the active component's properties. By default the displayed properties are the component attributes.

The Properties window provides the following:

- Access to the object attributes, region attributes, and other properties in a consistent format.

- Ability to assign values to most attributes via a selection list or push button. This ensures that valid values are entered for the attribute.

- Attribute categorization allows you to sort the attributes by category (Appearance, Behavior, Help, etc.) or subset the list of attributes to see only a desired category of attributes.

- Attribute linking is performed in the Linked To attribute column. This allows one component to change an attribute value when the attribute on another component is changed. For example, this can link the name of a graphic in a text entry control with the graph to display in the graph output control. No SCL is needed!

- Drag and Drop is easily implemented by setting the DragEnabled attribute of the component to drag from and the DropEnabled attribute of the component to drop on. The DragInfo and DropInfo attributes determine the value to take when dragging and where it is placed when dropped, but the defaults are usually acceptable for these attributes. Again, no SCL needed!

- Model/View communication is also specified via the attributes. This is discussed later.

- Ability to manipulate the shared attributes of multiple selected components. For example, selecting both a Text Entry control and a CheckBox control would allows you to specify common attributes (BorderColor etc.) for both of the controls at the same time.

- When a V6 object is selected, the Properties window will allow developers access to the old V6 object and region attribute windows.

- All attributes of a component can be accessed via SCL. This is another improvement over V6, where not all object attributes could be accessed programmatically.

The use of component attributes will significantly reduce the amount of SCL code you need to write for your applications. However, SCL will still be necessary. The next section details some improvements available with SCL coding.

SCL Coding Changes

The most important coding change is the availability of "Dot Notation" to set and get attribute values and to invoke methods. By providing a coding shortcut for invoking methods and for setting or querying attribute values, dot notation reduces typing and makes SCL programs easier to read. With dot syntax, there is no need to ever again write a call notify or a call send.

- You can reference attribute values directly:

```
ObjName.Text = 'cvalue' ;
field_value = ObjName.Text ;
```

instead of

```
call notify ( 'ObjName' , '_set_text_' , 'cvalue' );
call notify ( 'ObjName' , '_get_text_' , field_value);
```

You can also use attribute references as values in SCL statements. For example:

```
value = 5 + ObjectName.attribute ;
if ObjName.BorderColor = 'blue'  then do;
```

- Dot notation can also be used to invoke methods:

```
ObjectName._method('parameter') ;
```

For example, the following statement uses dot notation to send the _gray_ method to the control named field:

```
field._gray () ;
```

instead of

```
call notify ( 'field' , '_gray_' ) ;
```

Notice too that all the underscores in SAS method names are now optional. Only the first underscore is now required. Thus, a method may be written as _set_border_color_ or _setBorderColor.

Although attributes and method names are not case sensitive, the use of mixed cases is a coding convention that is quite useful for increasing readability of your programs. Please adopt this convention of using initial caps on words when coding your V7 SCL.

- Use of this new dot notation for method calls offers the huge advantage of compile-time error checking, guaranteeing that you will never see the following message at run time:

ERROR: This method is not defined for the object class

- Methods may now return values, which is not possible in V6. Notice the new return= option on the method statement and the ability to return a parameter value via the return statement:

```
MYMETHOD: method  return=num ;
  … code……
return ( 5 ) ;
endmethod ;
```

```
value = ObjectName.MyMethod() ;
```

will set value equal to the value returned by the method call, 5.

You can also use method returns as values in SCL statements:

```
value = 5 + ObjectName.MyMethod() ;
if ObjectName.MyMethod() = 5  then..;
```

Model Components

This brief discussion of non-visual objects is included because the use of these objects should be very important to you. To develop effective SAS/AF applications, you must learn to use these objects.

Model Components provide a way to access some type of data. The table below lists the model components available in the default V7 Resource catalog:

Model Components	Type of Data
Catalog Entry List Model	Catalog Entries metadata
Catalog List Model	Catalog metadata
Color List Model	System colors
Data Set List Model	Data Set metadata
External File List Model	Allocated External File metadata
Library List Model	Allocated Libraries metadata
LIST Entry Model	AF Program List entry
Range Model	Numeric range
SAS File List Model	Data Set metadata
SLIST Entry List Model	AF Saved SCL List entry
Variable List Model	Variables metadata

Typically these model components are used in conjunction with a Viewer Component to provide a visual representation of the data being accessed by the model. The Viewer Component is the user interface that presents the data to the user. Since the viewer is data independent it may be used with different sources (models) of data.

One of the great enhancements in V7 is the ability to instantiate (create) a Model Component in your frame in the same manner as visual components. In other words, you can place models in your frame just like any other object. When associating a model with a viewer, you can drag the Model Component onto the frame and drop it on a Viewer Control.

Although they are not visible on the frame display, the instantiated Model Component appears in the Properties window, where it may be selected, deleted, or similarly manipulated.

Coding is significantly reduced because there is no longer the need to initialize and setup non-visual objects via SCL. This eliminates the V6 requirement of using the loadclass and instance SCL functions to create a non-visual object as well as the need for calls to _setup_ and _get_members_ methods.

Running Existing V6 Applications in Version 7

At an absolute minimum, you must re-certify your V6 application under V7. This means executing the Qualification Protocol or QA Validation Test that was originally used to certify your application under V6. (This assumes that you have validated and documented the validation of your V6 application.) The results obtained from the execution of the validation under V7 must be the same as those obtained under V6 or adequate documentation of the differences must be provided.

As a cursory test of how a V6 application would run in V7, I executed ClinAccess™, a sophisticated clinical information system developed by MAJARO InfoSystems, using V7. Being a large V6 SAS/AF application, ClinAccess seemed like it would be a challenging test of how well a V6 SAS/AF application will run under V7. I have not yet executed the full validation test of ClinAccess under V7.

It is a simple matter to change the .exe used by an icon on the Windows desktop so that it points to SAS V7. In most cases with simple applications, this is all that you will need to do to run your existing SAS applications in V7.

However, after changing the .exe for the ClinAccess icon on my desktop, clicking the modified icon resulted in nothing!

What I discovered is that the customized config file that I was using with my application in V6 was causing V7 to crash before SAS initialized. Some V6 config file options are no longer available.

These changes are not documented and required significant trial-and-error to diagnose. I found these differences in the custom ClinAccess config file:

-formchar	Character set must now be quoted
-wincharset	No longer supported
-helpenv	No longer supported
-$*option*	Likely replaced with a V7 option

After appropriately adjusting my config file, ClinAccess initialized just fine. However, I noticed a number of changes in my existing application. These are some common items that may require developer intervention to run a V6 application in V7:

1) When running an application in V7, you may wish to close the Results window to allow use of the entire SAS Application Workspace by your application. Use the following options in your SAS invocation in the order shown:
   -nodmsexp -dms

2) Selection lists are significantly different in appearance. These lists, generated by the liblist, dirlist, catlist, etc. functions, do not look anything like the SAS generated selection lists of V6. These have changed to be more similar to host-system windows of the same type.

   Since these windows are different sizes than their V6 counterparts, modifications will be required if you define the region size for these windows in V6. Also, the titles defined in the function call are not always displayed as the window title.

3) The selection lists generated by datalistc and datalistn do not use an active WHERE clause that is applied to the data set being displayed. There are also problems with making selections from these windows as well as using the Find button. These are *significant* problems that have been fixed in Version 8.

   A change for the better with these windows is that they now no longer highlight the first displayed row. This was an irritation in V6 because users often assumed that this first item was selected because it was highlighted.

4) Because In-cell editing within Data Tables is now permitted, you cannot simply double-click in a cell to execute the table object's labeled section of SCL (e.g. if a Data Table is used as a selection list). You must either double-click on the row number, or place the cursor in a cell before double clicking in the cell to run the label section. This can be turned off to regain V6 functionality.

5) Generated output is no longer black & white. Titles and fonts are controlled by the new Output Delivery System (ODS). This actually makes your output easier to read and I encourage you to take advantage of ODS in your applications.

6) Custom pmenus created using an ID=1 on the ITEM statement of PROC PMENU will automatically gray the pmenu item. ID values in PROC PMENU must be greater than 3000.

7) Since V7 cannot update a V6 catalog entry, any V6 application that allows a user to edit or save a catalog entry (Source, Screen, Keys, Slist, etc.) must be modified to run under V7. In these cases, you may wish to convert the catalog to V7.

8) By default, when browsing a data set via CALL FSEDIT or PROC FSBROWSE, the cursor is not visible on the screen. This will cause problems if your application relies upon cursor positioning when using FSP screens. In V7, you may obtain V6 cursor visibility in browse via:

–hidecursor off	SAS invocation option
whidecursor off	command

9) SAS/SHARE will not initialize a server using your V6 code. A new parameter on the PROC SERVER statement is required to start a server:
AUTHENTICATE= OPT | REQ
See the online SAS/SHARE documentation.

10) Field-specific help entries specified in the List field attribute of Program entries do not work under V7 because an unreadable character is appended to the name of the help entry. You must convert to V7 and re-enter the help specification.

11) Line drawing characters used on Program entry display windows using the OEM character set do not display properly because the OEM character set is no longer supported. Line drawing characters created in V6 using the Sasfont font and the ANSI character set were not tested.

This list of items is only a starting point. Rest assured that there are other items that you will find that require modification to get your application to adequately run under V7. Some obscure issue (such as the loss of the OEM character set) may seem minor, but depending on the extent used, could require substantial rework.

Considering the complex nature of the V6 ClinAccess product, I felt that this application ran well under V7.

Upgrading Existing Applications to Version 7

As discussed in the previous section, most V6 applications will require only minor enhancements to run under V7. Because of this, you may not want to devote significant resources to upgrading a V6 application to V7. The scope and life span of the application, as well as planned enhancements, should be assessed when making this decision.

If you plan to further enhance the application, you may wish to convert the entire application to V7 frames. This permits use of the new development environment, Properties window, and long object names, as well as avoiding the unseemly appearance of an application that has both V6 and V7 objects.

Upgrading a V6 application to V7 is not as simple as replacing the V6 objects on the frame with V7 components. In addition to the items detailed in the previous section, there are more changes that are required when upgrading a V6 frame to a V7 frame:

1) Your V6 catalogs must first be copied into V7. This step is critical because once you begin making enhancements in the V7 catalog, there is no turning back without losing your enhancements. You cannot modify a V6 SAS/AF entry from within V7, nor can V6 modify a V7 entry.

2) When a Program entry is opened and closed in V7, the date of modification is updated, regardless of whether the entry was modified. This has been resolved in V8, but in the meantime you should be sure to use the Cancel command to close Program entries that are not modified to prevent this misleading indication. You will need to issue the Cancel command via your toolbar or a function key since it is not provided on the build environment pmenu.

3) The issue of replacing V6 widgets with V7 components is optional, since V6 objects work in a V7 frame. However, aesthetics are also a consideration with your applications.

   All text based V6 widgets (text fields, Push button, List box, etc.) will look odd in comparison to their V7 counterparts. You may want to replace V6 text based objects with V7 components. This may be very time consuming.

   However, since the appearance of V6 graphical objects (Graphics object, Container box, etc.) do not stand out in a V7 application, you will only want to replace these objects with their V7 counterparts when specific V7 functionality is required or if you want to use object attributes.

4) If you have V6 objects that do not have counterparts in V7, you will need to find a work-around in order to upgrade these objects to V7. It is easy to change a V6 Toolbar object to a set of V7 push button controls. Other V6 objects may not be so easy to upgrade to V7.

   For example, you cannot upgrade a V6 Graphics object that permits the type of graph to be changed via the _set_type_ method. This is because in V7 there is no single Graphics control, but instead there are separate graphics controls for each type of chart (chart, histogram, pie chart, scatter plot). To upgrade to a V7 control, you must select which type of graphic to allow and use the appropriate control.

5) Establish a template frame (or classes) that contain a set of common standardized components that can be quickly copied into your new V7 frames. Objects can now be copied from one frame to another, but beware that in V7 the object name is not copied. Resolved in V8.

6) It is common in V6 to use an assignment statement to assign the default attribute to an object. For example, your V6 frame SCL may have statements like this:

```
widgetx = 'value' ;
```

where widgetx is the name of a Text Entry, Text Label, or some other widget that accepts character data as its default attribute.

Such statements that reference a V7 component will fail with the following message:

```
ERROR: [Line 99]  widgetx is an object handle in
V7.  The value of the object handle for frame
widgets cannot be reassigned.
```

When upgrading your V6 objects to V7 components, you must modify all such SCL statements that are used to assign a value to an object. Thus, the previous statement would be re-coded for use with V7 components as:

```
widgetx.Text = 'character value' ;
        for a Text Entry component, or
widgetx.SelectedItem = 'value' ;
        for a Combo Box component.
```

7) Although dot notation is not required in a V7 frame, it may be desirable to re-code the call send and call notify statements with dot notation and to use component attributes. For this investment of time, you gain more maintainable code and compile-time error checking of methods. Dot notation may be used with V6 objects in V7 by clicking the "Use object name as ID in SCL" option in the object properties.

8) You must decide whether to implement the use of long variable and table names into your application. Under V7, variable names and data set names may be up to 32 characters, making the 8-character limit in your V6 application obsolete.

9) A number of concerns, mostly appearance related, are associated with the V6 Data Form object running in V7. I refer you to the SAS Notes page at www.sas.com for details.

10) Upgrading a V6 Program entry to a V7 frame is more difficult. You should be aware of the issues encountered with this type of upgrade if you have previous experience with converting Program entries to Frame entries in V6. For simple Program entries, you may only need to replace the Program fields with appropriate V7 controls and then follow the previously described steps for conversion.

Conclusion

As with all software upgrades, there will be difficulties encountered and patches that will need to be made to your applications. Although very few complex V6 applications will run seamlessly under V7, most applications will likely experience only minor irritations when running under V7.

While some of these irritations will require you to re-work some SAS/AF entries, you should be able to certify your V6 application to run under V7 without much grief. However, upgrading your V6 application to V7 will be more difficult.

As noted herein, there are a number of known bugs in V7. Fortunately, the SAS Institute has a strong record of working closely with users and making quick fixes to identified deficiencies in new software releases. You may want to start to qualify, upgrade, and develop applications in V7, with intent to release them under the soon-to-be-released SAS System Version 8.

References

(1) Chen Yao (1998), "SCL Reborn – The New SAS Component Language (SCL) in Version 7", unpublished paper presented at SUGI 23.
(2) Walker, Glen and Gagliano, Tammy (1998), "Dressing Up Your Version 6 Objects to be Version 7 Components" unpublished paper presented at SUGI 23.
(3) Walker, Glen and Gagliano, Tammy (1998), "Version 7 SAS/AF Software – The New Component Technology", unpublished paper presented at SUGI 23.

Acknowledgements

I would like to extend sincere thanks to Glen Walker at the SAS Institute for the time, effort, and patience he extended helping with the preparation and review of this paper. Also, much thanks to Ann Rockett at the SAS/AF technical support desk for the time spent helping to debug my applications.

About the Author

Steve Wilson is the Director of Clinical Applications Development at MAJARO InfoSystems, Inc., a consulting and software development company for the pharmaceutical and biotech industries. MAJARO develops and markets ClinAccess ™, an integrated clinical trials system developed in SAS/AF software.

Steve has been developing applications in SAS for 15 years and has given numerous presentations at SUGI as well as regional and local conferences.
Steve can be contacted at: Swilson@majaro.com