Effective Utilization of Region Attachments in a FRAME Application
Bud Whitmeyer, SAS Institute, Inc., Cary, NC

ABSTRACT

This paper discussed the utilization of region attachments within a FRAME entry. The intended audience is FRAME application developers. The following topics will be covered:
- Various attachment types and options.
- Using region size restrictions.
- Attachment interpretation.
- Avoiding and handling errors.
- General tips.
- Real life examples from the SQL Query Window.

INTRODUCTION

Regions

A region defines a viewable area within a window being used by the FRAME application. Regions are normally outlined so that the location and size of the region is visible. Regions can optionally contain an object. In this sense a region is something like a box. Only one object can occupy a region, however a region can also contain other regions. A region containing other regions is considered the parent of the regions within it. The contained regions are considered nested within the parent and are called child regions. Child regions that have the same parent are considered siblings of each other. Each of these regions can also contain an object, so you can get many objects in a region if you use nested regions.

An object is what makes a region interesting. It will usually cause the region interior to be filled with something visible such as a graphic created with SAS/GRAPH® or a pushbutton widget for example.

The main area of your application is called the master region. It has numerous special properties (e.g. it cannot be removed and it will try to stay the same size as the containing window).

Region manager is a term used to describe the portion of FRAME software that is responsible for basic management of regions and objects. Managing attachments between regions and handling region move and resize events are part of the set of services provided by the region manager.

What is an attachment?

An attachment is a connection between two regions that controls the placement or size of one or both regions. A set of these attachments thereby serve to define the spatial relationships between the set of regions in a particular FRAME entry.

Resize events

A region resize event is any attempt to change the size or move a region. A region resize event can be initiated by a user grabbing a handle of a region, by an application invoking a region manager service function or by the region manager itself. A resize of the master region in the FRAME application can be initiated by resizing the containing window. When a FRAME entry is initialized an automatic resize of the master region is initiated by the region manager if the containing window is a different size than it was when the FRAME entry was saved.

Purpose of attachments

Attachments allow you to create regions that are sensitive to the movement and resizing of other regions. This permits you to create a FRAME entry that is sensitive to user window size preferences or environment window size restrictions. These features can be particularly important if you plan on porting your application.

The particular event, therefore, that is normally most important is a resize event on the FRAME entry's master region that is automatically generated when initializing with a window size different from the one that is saved with the FRAME entry. Attachments are usually set up to react appropriately to this event.

Less common, although still important, is setting up attachments for the purpose of reacting to user generated resize and move events or changes generated by a contained object itself.

Define attachment mode

Define attachment mode is the mode of operation that is used to create, modify or remove attachments while editing a FRAME entry. Define attachment mode is initiated from the region attributes pop-up menu.

Attachments are normally defined while in define attachment mode by performing a rubberband line operation. A rubberband line operation is initiated by clicking the mouse in one region and dragging it to another region or close to another region.

Please refer to the FRAME reference book (SAS/AF® Software: FRAME Entry) pages 47-58 for more details on how to interact while in define attachment mode.

ATTACHMENT OPTIONS

An attachment always serves to maintain a desired distance between connect points of two regions. Exactly how the desired distance is determined and maintained depends upon the characteristics of the attachment itself.

Attachment type

There are two basic types of attachments: absolute and relative. The desired distance is interpreted differently for different types of attachments.

An absolute attachment is used to preserve the actual distance between the connection points of two attached regions. That is, the number of pixels is kept constant. An absolute attachment is normally represented visually by a solid line (O—O).

A relative attachment will maintain the distance between the connect points as a certain percentage of the appropriate parent region dimension. The parent region may be involved in the attachment itself. Note that in the absence of a change in parent size a relative attachment is equivalent to an absolute attachment. A relative attachment is normally represented visually by a spring (O—T—O).

Each attachment has an associated attachment value, if the attachment is absolute then the value is the number of pixels to keep constant. If the attachment is relative the value is the desired percentage to maintain.
To demonstrate the difference between absolute and relative attachments, the following example shows two regions attached to their common parent. The top region is attached absolutely and the bottom is attached relatively.

**Display 1 Absolute and relative attachment types**

If the common parent is grown from either the right or left side the attachments will be honored and following will result.

**Display 2 Honoring different attachment types**

Notice that the absolute attachment preserved the actual distance between the top region and the left edge of the parent region, while the relative attachment preserved the percentage distance between the bottom region and the left edge of the parent region.

**Attachment direction**

Attachment direction, an integral part of an attachment, determines whether an attachment will propagate a region resize and move event. Attachments can be single directional or bidirectional. Single directional attachments can further be divided into "in" attachments and "out" attachments.

A resize or move event is considered to be propagated by an attachment if the original resize or move event, when combined with the attachment, results in the generation of a subsequent resize or move event. An attachment is considered to be honored if a resize or move event is propagated because of the attachment.

The attachment direction controls whether a resize or move event results in another resize or move event (honoring the attachment) or serves to redefine the value associated with the attachment.

The differences are best shown by example. In the following window, regions a, b and c have a common parent region and are all attached to the right side of their parent with absolute attachments. a's attachment is bidirectional, b's attachment is single directional going out of b and c's attachment is single directional going into c.
parent region to be pulled to the left since b's attachment is honored only from b. The parent was then resized resulting in the honoring of the attachments to a and c. If a had been moved the parent would also have moved since a's attachment is bidirectional. However, if c is moved the following results.

Display 6 Attachment direction example after c is moved

Only c moved. Since the attachment associated with c is only honored coming into c the resize event is not propagated (i.e. there are no additional move or resize events generated).

Attachment points

Each region can be attached at one of five different points. Each side plus the center of a region are all possible attachment points. Each attachment connects one of these points of one region to one of these points of another region.

An attachment to the side of a region typically causes the associated region to be resized when the attachment is honored, whereas an attachment to the center of a region always causes the region to be moved when the attachment is honored. All examples to this point have shown the child region's center attached to its parent's side.

Let's see how some side attachments can be used to control region size. In the following window a and b both have their right side attached to their common parent's right side and b also has its bottom attached to its parent's bottom.

Display 7 Attachment side example initial setup

If the parent is grown by moving the right side out we get the following result.

Display 8 Attachment side example after parent right side grow

Notice that the right side of both a and b moved along with the parent's right side, but the left side of each did not. If the parent is now grown by moving the bottom down the following results.

Display 9 Attachment side example after parent bottom side grow

Notice that the bottom of b is moved down along with the parent's bottom side.

An attachment is either considered vertical or horizontal depending upon the attachment points. An attachment to the top or bottom is vertical and an attachment to the left or right is horizontal. A center attachment point does not determine the vertical or horizontal nature of the associated attachment. In that case the other attachment point is used. If both attachment points associated with an attachment are to the center of the respective regions then the region manager will make the attachment horizontal if the two center points are closer to each other in the horizontal direction, otherwise the attachment will be vertical.

Resize events also have vertical and horizontal components as evidenced in the previous example. Resize events that are only horizontal in nature ignore all vertical attachments and vice versa.

When in define attachment mode, each region for which you are defining attachments is divided into sections. Each section represents a possible attachment point for that region. Also, each attachment is considered to be defined by a particular section and each section is only allowed to define one attachment. The defining section is the one in which the rubberband line is initiated.
You can still define many attachments to a single region section, however, by defining the attachments at the other attachment point. This is very useful sometimes as shown in the next example.

Sibling attachments

Attachments can further be classified by whether they attach the defining region to its parent or to a sibling region. If an attachment connects a region to one of its siblings it is referred to as a sibling attachment. All of the examples up to this point have shown parent attachments.

Sibling attachments can often simplify a set of attachments by making it visually easier to interpret the attachment logic, and can sometimes result in requiring less attachments. In the following window a, b and c all want to follow the right side of their common parent. This could have been done by attaching all regions to the parent directly, but using sibling attachments makes the visual easier to understand.

Display 10  Sibling attachment example initial setup

If the parent right side is moved to the right the following results (as you would expect).

Display 11  Sibling attachment example after parent resize

External (border) attachments (6.10 and above only)

By default, when an attachment is attached to the side of a region the actual attachment point is considered to be the edge of the region itself. This point is "inside" any special outline the region may have. It is also "inside" any attached scroll bars that may be associated with the region.

External attachments enable you to create attachments with attachment points on the outer edge of the region "dressing". This is often necessary when the region dressing is a fixed number of pixels as it is with region outlines and region attached scroll bars. To create an external attachment you should initialize rubberband line mode with an extended drag operation instead of a normal drag. Under the X environment extended drag is initiated by holding down the shift key while initiating the drag.

External attachments can be differentiated from normal attachments when in define attachment mode or when just viewing attachments by the thicker line used to draw the graphic representation of the attachment and the fact the terminating point of the directed arrow is the region border instead of the region itself.

The reason you might need to use this type of attachment when you have a fixed pixel dressing is due to the nature of how distance information is saved for absolute attachments in the FRAME application. When a FRAME entry is saved, the distance value is converted from pixels to an appropriate number of font units (either rows or columns). Later, when the FRAME entry is loaded (read from the catalog and initialized) this conversion is reversed. If the SAS® system DMS font cell size is significantly smaller when the FRAME is reloaded the distance between the attached side and the region it is attached to may be much closer from a "number of pixels" point of view than it was formerly. If the region has a large outline or scroll bars these may overlap the attached region since they are a fixed number of pixels (i.e. are not sensitive to the change in font cell size).

Region size restrictions

You can place restrictions on region resizing by displaying a pop-up menu while over a region in define attachment mode. You can make the region size fixed absolute, fixed relative or you can reset the fixed status of the region (remove any restrictions on resizing). There are separate horizontal and vertical settings so you can set a size restriction for a region in only one direction if you so desire. By default regions are not size restricted.

The resize restrictions absolute and relative are similar in nature to absolute and relative attachments and you may think of region size restrictions as being "internal" attachments although they are not actually attachments since they do not connect two regions. They are, however, represented visually when in define attachment mode as if they are "attachments" internal to the region.

If a region is fixed absolute then its size may not change in the direction (either horizontal or vertical) that it is fixed. If the region is fixed relative then its size will be maintained as a certain percentage of its parent region’s size in the appropriate direction.

The window below illustrates a common use of fixed absolute and fixed relative region size restrictions. Region d is fixed absolute and side attached absolutely to the right side of its parent. This will cause d to move with the right side of its parent and is equivalent to attaching to d’s center. Use of this method instead of center attachment is mostly a matter of personal preference. Regions a, b and c will share the expansion of the parent equally. Notice in this case that the fixed relative restrictions are required to achieve the desired result. If they were left off then the right hand side of c would resize, and the distance between c and b would be set correctly but the actual size of c would be undefined.
If I now grow the right side of the parent region the attachments and region size restrictions are honored to produce the following result.

Notice that regions a, b and c have grown in proportion with the parent and their respective positions are relatively preserved. Region d has been moved along with the right hand side of the parent. For contrast, here is the result of growing the parent region without any region size restrictions.

In this case, the parent resize event causes the right hand side of both c and d to move to the right, preserving the relative and absolute distances respectively. However, since there are no size restrictions the left hand side of each region is not relocated. The distance between b and c will be adjusted but the right hand side of b will be moved since the direction of the attachment is into b.

Similarly for a. So we see that the distances between the regions have been maintained as required due to the attachments, but the resultant region sizes are not clearly defined.

ATTACHMENT INTERPRETATION

Although it is not absolutely necessary to have an understanding of how the region manager honors attachments it can help. Knowledge of what is going on can be used to sidestep potential problem situations.

The basic algorithm

As stated earlier the purpose of attachments is to maintain a desired distance between connect points of regions. Attachments are normally in a synchronized state. An attachment is considered to be synchronized if the distance between the connect points is correct given the value associated with the attachment and the attachment type. Honoring an attachment can now also be viewed as an attempt to synchronize the attachment (without resetting the attachment value).

When a region resize or move event occurs all attachments to that region and any size restrictions that region may have are checked to see if they have been rendered out of sync by the event. If a size restriction needs to be enforced, then an appropriate change to the region is made. If an attachment is found to be out of sync then a resize or move event is generated for the "attached to" region. The entire process is then repeated for the "attached to" region. The following diagram depicts this logic.

Region resize or move event initiated.

Does Region have size restrictions that need to be honored?

No

Yes

Modify original events to reflect size restrictions.

Check all attachments to the region.

Are all attachments synchronized?

No

Synchronize attachments that are not already in sync.

Yes

Generate a resize or move event for each region attached to a synchronized attachment.

Display 15 Basic resize/move event handling algorithm

An important point to notice is that the algorithm relies upon the initial resize event propagating to other regions in order to reach a completely synchronized state. Therefore, if you define attachments that are unconnected to other attachments they may not be honored. For example, the attachment between the 2 sibling regions in the following window will not be synchronized when the parent is resized because there is not an attachment defined between either of the regions and their common parent.
Region manager shortcuts

In order to make the resize event handling process as efficient as possible and to avoid the possibility of an infinite loop while the region manager is in the process of attachment synchronization a number of "shortcuts" are used. It might be helpful to be aware of these when trying to figure out why an attachment you have created is not being honored as you expect it should. The following situations will cause an attachment to be ignored.

- The resize event is only horizontal in nature and the attachment is vertical or vice versa.
- The attachment is a side attachment to a sibling and the location of the side has not changed.
- The attachment is back to the region that originally propagated the event to the region currently being resized.
- The attachment is single directional coming into the region instead of going out of the region.
- The actual distance in pixels between the two attached regions is within \( \frac{1}{16} \) character cell (one character cell in version 6.08) of the desired distance. This applies to both absolute and relative attachments.

The last item is especially bothersome since it frequently causes attachments that have very small desired distances to be ignored. In many cases you'll never notice this.

Move versus resize

A region move event is interpreted a little differently than a resize event. When a region is moved all of its child regions move along with it and no attachment honoring occurs within the moved region. This is very different than a resize event in which (usually) correct alignment of subordinate regions requires some kind of attachment synchronization.

Errors and error handling

Even if you are careful about how you set up attachments you are bound to make some errors and create attachments that the region manager finds unsatisfiable, or that simply do not result in the desired layout. Whenever the region manager detects an error during resize event handling the entire process is "unrolled" and the regions are all restored to the location and size that they had before the attempted resize. The following list itemizes most reasons that a resize event may fail.

- The object contained within the region returned an error condition to the region manager when informed of the new size. This usually implies that the region has become too small to hold the object.
- An attachment is out of sync that has previously been synchronized during the processing of this resize event. This usually implies that you have defined attachments that cannot be satisfied. Check all your attachments for conflicts.
- Satisfying an attachment requires resizing the master region. This is not allowed. The master region can only initiate a resize event, not be resized due to attachment synchronization (propagation of the event). Check all your attachments to the master region.
- The event will result in a region being placed outside of the boundaries of its parent (unless its parent is a viewport region, like used by the workarea object, in which case this is okay). This typically implies an attachment is needed that is missing.
- The event results in a region being smaller than allowed by the region manager. This is a very small value (eight pixels). This does not necessarily imply that the attachments are in error. It could simply be the case that you attempted to shrink the original region too much.
- Overlap of regions containing textual based objects will occur if the attachments are satisfied. This usually implies an error in setting up your attachments.

If examination of the attachments together with the results of attempted resizes does not make an attachment problem clear, you can get the region manager to trace the handling of the resize event by issuing the command RM GROW DEBUG. The next time you try to resize the region diagnostic messages will be sent to the appropriate window (message or log). As an example of the type of output generated in an error situation, consider a window with a region containing another region with a pushbutton in it as shown in the following.

```
File View Actions Locals Globals Help

Display 16  Attachment that will not be honored when parent resizes
```

```
File View Actions Locals Globals Help

Display 17  Region with pushbutton in another region
```

An absolute attachment is defined from the bottom of the pushbutton region to the bottom of its parent as shown below. This is done thinking (erroneously) that this is all that is required to keep the pushbutton region a fixed distance from the bottom of its parent and preserve the pushbutton region's size.
An attempt to move the bottom of the parent up results in an error and the following messages are sent to the log by default:

ERROR: Region too small for object (OBJ1).
ERROR: Unable to perform resize request (attachment/size).

This may be enough information to determine what is required (either attach to the center of the pushbutton region or for the size of the pushbutton region vertically). If it is not you can issue the RM GROW DEBUG command and the following information will be sent to the log.

NOTE: Resize beginning for 'Empty' newbox=15,30:333,180 oldbox=15,30:333,228 dir=r.
NOTE: Processing Absolute attachment (in) of bottom to bottom of child.
NOTE: Attachment reinitialized. Adjusted box=45,75:180,75.
NOTE: Resize beginning for 'OBJ1' newbox=45,75:180,75 oldbox=45,75:180,115 dir=r.
NOTE: Object resize exit called for 'OBJ1'. rc=0.
NOTE: Resize error detected for 'OBJ1'.
ERROR: Region too small for object (OBJ1).
ERROR: Unable to perform resize request (attachment/size).

The bounding box information is presented in the form of upper-left-x, upper-left-y, lower-right-x, lower-right-y. In this case the resize for 'OBJ1' has attempted to set the region size in the vertical direction to two pixels (upper position at 75, lower position at 76). Presumably this makes it clear that we have forgotten about the top of the region as it is certainly not our intent to try to hit a pushbutton into a two pixel high region.

If you have defined many attachments turning on resize diagnostics may cause many messages to be issued and wading through all of them may be tedious. Sometimes, however, you can go down to the bottom of the log and see what the error is and work back from that to see why the error occurred.

Using Attachments

Overall analysis

By far the most important aspect of defining attachments for a FRAME entry is to carefully think about how you would like various regions and contained objects to react to the appropriate resize event. For simplicity I will assume that the reason you are defining attachments is to handle master region (i.e. window) resize events appropriately. The points made will also apply to other situations also.

Carefully examine the regions and determine what regions should absorb the gain or loss of space. Examine the horizontal and vertical dimensions separately. It is fairly common for a resize to be handled in a different manner in each dimension.

Tips

There are many interesting situations when dealing with attachments. Some of these have already been touched on in the preceding sections. The following list can help you avoid more problems and be more efficient in your creation of attachments.

1. Use only the attachments you need.

Do not define more attachments than are needed to achieve the desired result. When a resize event is generated all attachments must be checked, thus a performance penalty is paid for unnecessary attachments. Also, attachment information must be stored with the FRAME entry so unnecessary attachments increase the size of the stored entry.

The same is not true for region size restrictions. There is virtually no overhead paid for these settings. Sometimes it is nice to turn on a fixed size restriction even if only for "documentation" purposes so that if the attachments are viewed by another person at a later date they will be more understandable.

2. The master region always resizes from the lower right.

Resizes of the master region are always interpreted as a resize of the lower right even if they were generated by a window resize of the upper left of the window. Thus absolute attachments to the top or the left of the master region are unnecessary. Note that this is not true of regions in general, only of the master region.

3. Take advantage of move only situations.

If you know that a container or group region will only be moved and not resized due to attachments to its siblings and parent then you do not need to define any attachments to the child regions of the container or group.

4. Keep the attachment logic simple.

Keep the attachments at each region nesting level as simple as possible. If it seems that the attachments are getting too complicated consider creating a container (or group or empty) region just to hold a set of related regions. Attach the container to its parent and/or siblings and set up attachments (if necessary) to the container's children.

5. Beware of using absolute and relative together.

Avoid mixing absolute and relative attachments in the same direction. It is very easy to create a situation that is unsatisfactory when mixing absolute and relative attachments in the same direction. There are situations when this is appropriate, but it's a good idea to take a second look to make sure this is really what you need.

6. Avoid relative attachments with textual objects.

Avoid using relative attachments with regions containing textual based objects that must be aligned on a row/column boundary. Usually the desired placement of a region after honoring a relative attachment will not exactly align on a row/column boundary necessitating a shift of the region to attain the required alignment. This can create odd visual effects and significantly increases the chance of textual object overlap. If you feel you must use relative attachments with textual based objects make sure to leave space around each region to allow for shifting without creating an overlap situation.

7. You can resize in define attachment mode.

Remember that you can still resize any region without attachments being honored by entering define attachment mode and performing the desired move or resize. This will cause any associated attachments to be redefined instead of honored.

8. Create small and allow for growth.

In general, results are usually better when honoring a grow rather than a shrink of the master region. You are better off

Avoid using very short attachments—especially if it is important that the attachment be honored in order to correctly propagate the resize event. These attachments are most likely to be ignored due to the \( \frac{1}{2} \) character cell rule mentioned earlier. Short attachments are acceptable if placement or size variations within \( \frac{1}{2} \) character cell are insignificant and propagation is not an issue.

10. Test your attachments.

All attachments can be tested while in build mode. Position the regions/objects, create the attachments and save the entry so that the desired placement is not lost. In case the attachments do not work as you expect them, if they do not then cancel and re-edit the FRAME and change the attachments appropriately.

11. Use single directional attachments in most cases.

Honoring a master region resize event is essentially an “out-in” process and thus usually single directional attachments are most appropriate, especially for attachments from parent to child. Sibling attachments can also be made single directional, but be sure you have analyzed how the master region resize event will be propagated so you define sibling attachments in the correct direction.

12. You can be relatively sized easier than you think.

If you want a region to be sized relatively you can do this by attaching one side to the parent relatively and setting the region size restriction to fixed relative instead of attaching both sides relatively. This uses one attachment instead of two and can result in significantly fewer attachments if you have many regions that are being sized relatively in this manner.

13. You can find out about an attachment.

If you aren’t sure exactly what kind of attachment is defined by a region you can click on the region while in define attachment mode and a message will be displayed indicating what kind of attachment is defined there and what it is attached to.

REAL LIFE EXAMPLES

The following examples are actually used in the SOL Query Window.

LOOKUP.FRAME

Display 19 LOOKUP.FRAME

LOOKUP.FRAME is a simple FRAME entry that contains only a text entry widget, a listbox widget, and two push button widgets. The listbox is clearly the primary focus of the window and we want it to absorb most of the space from a master resize event in order to show as many “lookup values” as possible. It is reasonable for the message area to be sensitive to the horizontal window size. With these factors in mind the following attachments are set up.

![Diagram of LOOKUP.FRAME](image)

Display 20 Attachments for LOOKUP.FRAME

Let’s examine how a master region resize event will be interpreted in this case. Assume the master region is being grown. Consider what happens vertically first.

1. Each pushbutton is absolutely attached by its center to the bottom of the master region so they will both be pulled down. Since they are initially character cell aligned and the master region is always guaranteed to be an integral number of character cells there will not be any shifting required to satisfy character cell alignment needs.

2. The pushbutton on the right side will propagate the resize event to the listbox. Since the bottom of the listbox is attached it will be pulled down thus increasing the size of the listbox region. This could have also been accomplished by attaching to the parent directly, or to the pushbutton on the left side.

3. The text message area is unaffected vertically.

Now let’s look at what happens horizontally.

1. The right pushbutton is pulled along with the right hand side of the master region due to the absolute attachment. Since its size is restricted as fixed absolute the left hand side is pulled along also, resulting in a move of the region to the right.

2. The right hand side of the listbox region is also moved due to an attachment similar to the one on the pushbutton. This results in a horizontal increase in the size of the listbox region.

3. Similarly the text message area is increased in size.

4. The left pushbutton is unaffected horizontally.

It is usually a good idea to examine the entry to see what the limiting areas are if a shrink is attempted. This serves as additional verification that you have set the attachments up in a manner that will produce the desired results. In this case the shrink will be limited by the eventual overlap of the two pushbuttons horizontally and by the eventual “shrinking away” of the listbox region vertically.
WHERE.FRAME

Display 21 WHERE.FRAME

This is a somewhat more complex FRAME entry with a set of pushbuttons and a checkbox near the bottom, an extended table in the center, a listbox on the right and two text areas near the top.

The pushbuttons and checkbox need to hug the bottom and left.

The extended table and listbox both need to grow vertically and the listbox needs to grow horizontally. The message area needs to grow horizontally. With this in mind, the following attachments were defined.

Display 22 Attachments for WHERE.FRAME

Probably the key item to notice about this FRAME entry is the use of an empty region to surround the pushbuttons and checkbox.

This allows only one attachment to be used instead of one to each of the individual regions. Also, since we have center attached the empty region it will only move, therefore no attachments are necessary within it. All other aspects of this FRAME entry are similar to LOOKUP.FRAME.

QRYCOLS.FRAME

The situation in this FRAME entry is a bit more complex. Vertically things are easy. The two listboxes on either side should grow with the master region. Horizontally, things are a bit harder. Ideally, the master region grow/shrink would be absorbed by the two listboxes equally. The location of the buttons in the middle complicates this scenario. The attachments below represent a reasonable effort given the desired layout.

Display 23 QRYCOLS.FRAME

Display 24 Attachments for QRYCOLS.FRAME

Let's only consider what's happening horizontally since the vertical case does not present anything new. Both the listboxes are setup to be sized relatively. This will allow significant growth when the master region grows. Notice the attachment of the opposite side to the parent (e.g. the "Available Columns" listbox has its right side attached to its parent's left side). This avoids the "short attachment" problem. The use of a fixed relative size restriction eliminates a need to attach the other side to the parent helping to create a simpler scenario.

The middle container, which serves the same grouping purpose as the empty region in the previous example, must be kept a fixed size. It is center attached relatively to the left hand side of the "Available Columns" listbox. Using a relative attachment here will keep the buttons centered within the two listboxes creating a nicer visual effect.

There are two main problems with this attachment setup. The first is that it is very limited shrinkage-wise horizontally. This is because the right listbox will quickly squeeze into the button area creating overlap. The second is that space isn't utilized as efficiently as it could be. That is, when a grow occurs horizontally, not as much space is given to the two listboxes as would be ideal.

Another alternative is presented below that will allocate more of the master region growth to the two listboxes.
Again, only the horizontal will be analyzed. What is somewhat different about this approach is that the master region resize event is first propagated to the middle container box via an incoming relative attachment to the center of the container. This causes a move of the container box and keeps it centered within the master region. There is then an absolute attachment to the left side of the right listbox region which pushes that side and then the event is propagated further by an absolute attachment back to the right side of the right listbox. The desired layout is completed by the attachment to the parent of the right side of the right listbox. The logic of this scheme does not increase the size of the space between the three main regions, but distributes all growth to the listboxes.

This scheme also addresses the inability to shrink. Since the size of each listbox is not restricted in any way, they can absorb more of the master shrinkage similar to how they absorb more of the growth.

CONCLUSION

We have discussed various types of attachments and attachment creation options, the use of region size restrictions, how attachments are interpreted by the region manager and covered many tips for better results when setting up attachments.

We can now see that by analyzing the interface window created with the FRAME builder and being careful about the use of attachments and region size restrictions the interface window can be made sensitive to both user preferences and host GUI environment restrictions.

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