Paper 037-31

SAS/AF® – Building a Tree View Hierarchy with Drag and Drop

Darrell Massengill, SAS Institute Inc., Cary, NC
Scott Jackson, SAS Institute Inc., Cary, NC

ABSTRACT
Drag-and-drop is a powerful tool in SAS/AF software; the SAS 9.2 Tree View Control is also a powerful and flexible SAS/AF tool. In combination, these tools enable you to build exciting SAS/AF applications such as the one presented here. This example application enables the user to create a Tree View hierarchy simply by dragging and dropping items from list boxes onto the Tree View.

INTRODUCTION
The Tree View Control enables you to create a hierarchical list of nodes that are similar to the Windows Explorer folder list. The hierarchy can be created programmatically, but it can also be easily created by dragging the information from a list to the tree. The example that's given here enables you to create the tree by dragging information from two list boxes and dropping that information onto the tree. One list box contains a list of employee names, and the second list box contains a list of the products that the employees are responsible for selling. (See Figure 1.) Employees can have other employees and products listed under their names, but products cannot have anything listed under them. This example describes how to create a tree with the drag-and-drop feature but does nothing with the tree that is built.
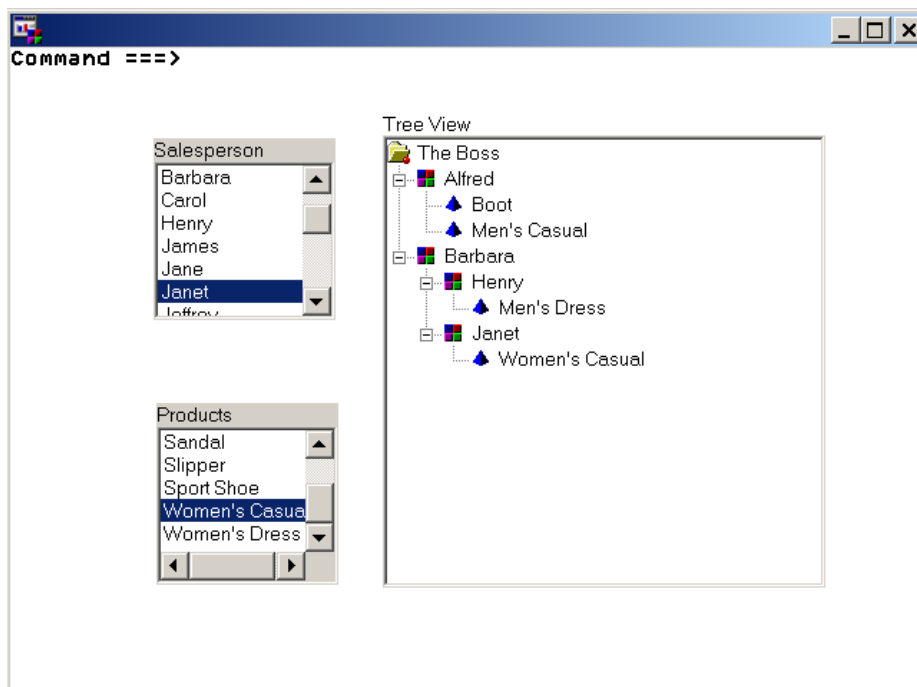

Figure 1. Tree Created by Dragging Information from List Boxes to Tree View

BUILDING THE EXAMPLE
This example is easy to create. Each step is explained in detail in the sections that follow. The first two steps create two new classes that process the information; one class drags and the other class drops the information. These new classes are then combined with standard classes on a Frame to enable the drag-and-drop feature between the list boxes and the tree view. The details presented in the following sections assume that everything is being built in the SASUSER library. If you use another library, you must modify the example accordingly.

CREATING THE DRAGLIST CLASS

The first class that you need to create is a sub-class of the List Box class. This sub-class must contain a new attribute to handle the drag information. In addition, you need to add code to override the _startDrag method and manipulate the drag information for the new attribute. Figure 2 shows an overview of the resulting class and code. Notice that the catalog that's used is named TREEDND. Here are the steps for creating **Draglist.class**:

1. Issue the command build `sasuser.treednd.draglist.class`. Because this class is a sub-class of the List Box class, specify the parent as `sashelp.classes.listbox_c.class`.

2. In the Class Editor, select Attributes from the tree view on the left. Right-click in the window to create a new attribute named *dragDetail* of type LIST. Let the other settings retain their default values. Dragging and dropping can extend between Frames, so this list will be created internally as a global list. While you cannot directly pass an object on this attribute, you can pass numeric list identifiers (listID). In this example, two items are added to this list: the numeric listID of the Draglist object and the name of the Draglist object. This is done in the code in **Draglist.scl** (Figure 2).

   Note: The custom access methods (CAMs) will not be run during the drag operation from a list box, so you cannot see the currently selected item. The tree-node code will check it.

3. Override the *dragInfo* attribute, and change the list that's specified for Initial Value to the *dragDetail* attribute that was created in step 2.

4. Override the *dragEnabled* attribute and set Initial Value to YES.

5. Override the _*startDrag* method with the signature `(O:SASHELP.CLASSES.DRAGANDDROP.CLASS;)V`. Specify the Source Entry as `sasuser.treednd.draglist.scl`.

6. Save **Draglist.class**.

7. Create **Draglist.scl** and include the override code for the *startDrag* method as shown in Figure 2. In the *startDrag* method, you will place the numeric object ID and object name into the *dragDetail* list that you created earlier. Remember that you must not insert an object or list type.

   Note: If you allow dragging between Frame entries, then you cannot pass identifiers of any object or list and expect to use them in the other Frame. This example does not support dragging across Frame entries.
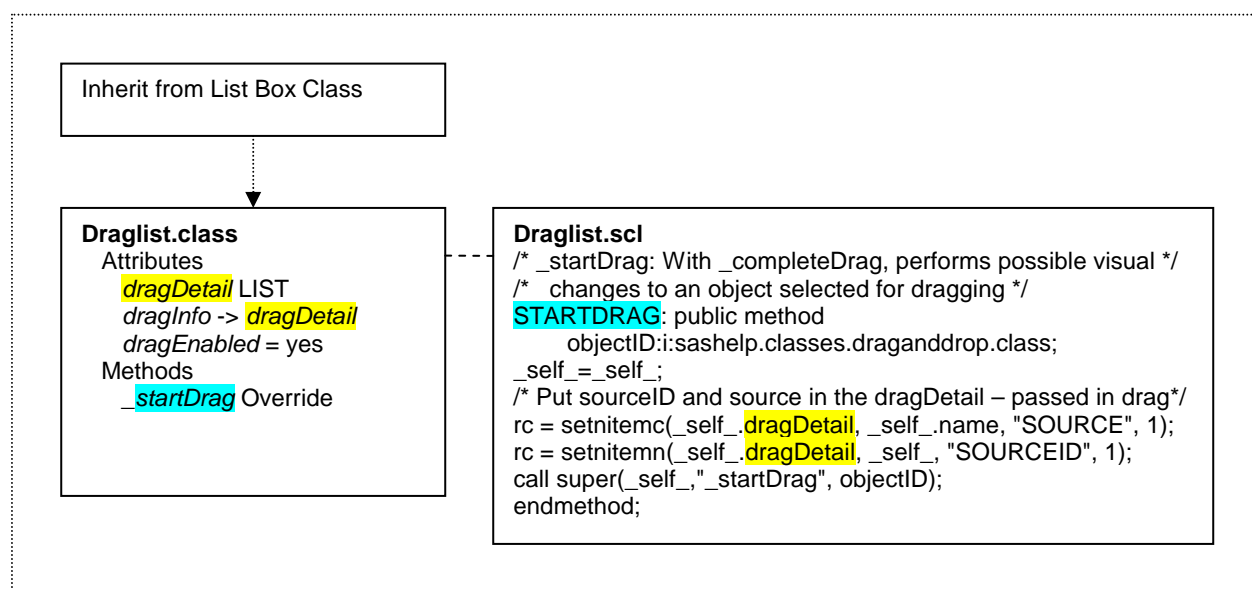
8. Save and compile **Draglist.scl**.

Figure 2. Draglist Class and SCL Code

CREATING THE DROPNODE CLASS

The next class you need to create is a sub-class of the Tree Node class. You must create a new attribute for handling the dropped information. In addition, a new setCAM method must be created for the new attribute. Figure 3 shows an overview of this class and code. Here are the steps for creating **Dropnode.class**:

1. Issue the command `build sasuser.treednd.dropnode.class`. Because this is a sub-class of the Tree Node class, specify the parent as `sashelp.classes.treenode_c.class`.

2. Create the new attribute *dropDetail* of type LIST. Edit the attribute and define set CAM to be *setcamDropDetail*. This attribute will be used to obtain the information dropped on the node. The *_startDrag* method adds the information to the list in **Dragnode.scl** and *setcamDropDetail* will retrieve the information from the list.

3. When prompted to create the method *setcamDropDetail*, click YES. In the New Method window, make sure Scope is set to "Protected" and Source entry is set to `sasuser.treednd.dropnode.scl`.

4. Override the *dropInfo* attribute. Click the Initial Value field and open the editor. Add *dropDetail* as the drop attribute.

5. Save **Dropnode.class**.

6. Create **Dropnode.scl** and include the *setcamDropDetail code* (see Figure 3). This code extracts the dropped information from the list and uses it. The name of the list box is used to decide which icon is used and if expanding is allowed. The selected item information is added to the node for the tree.

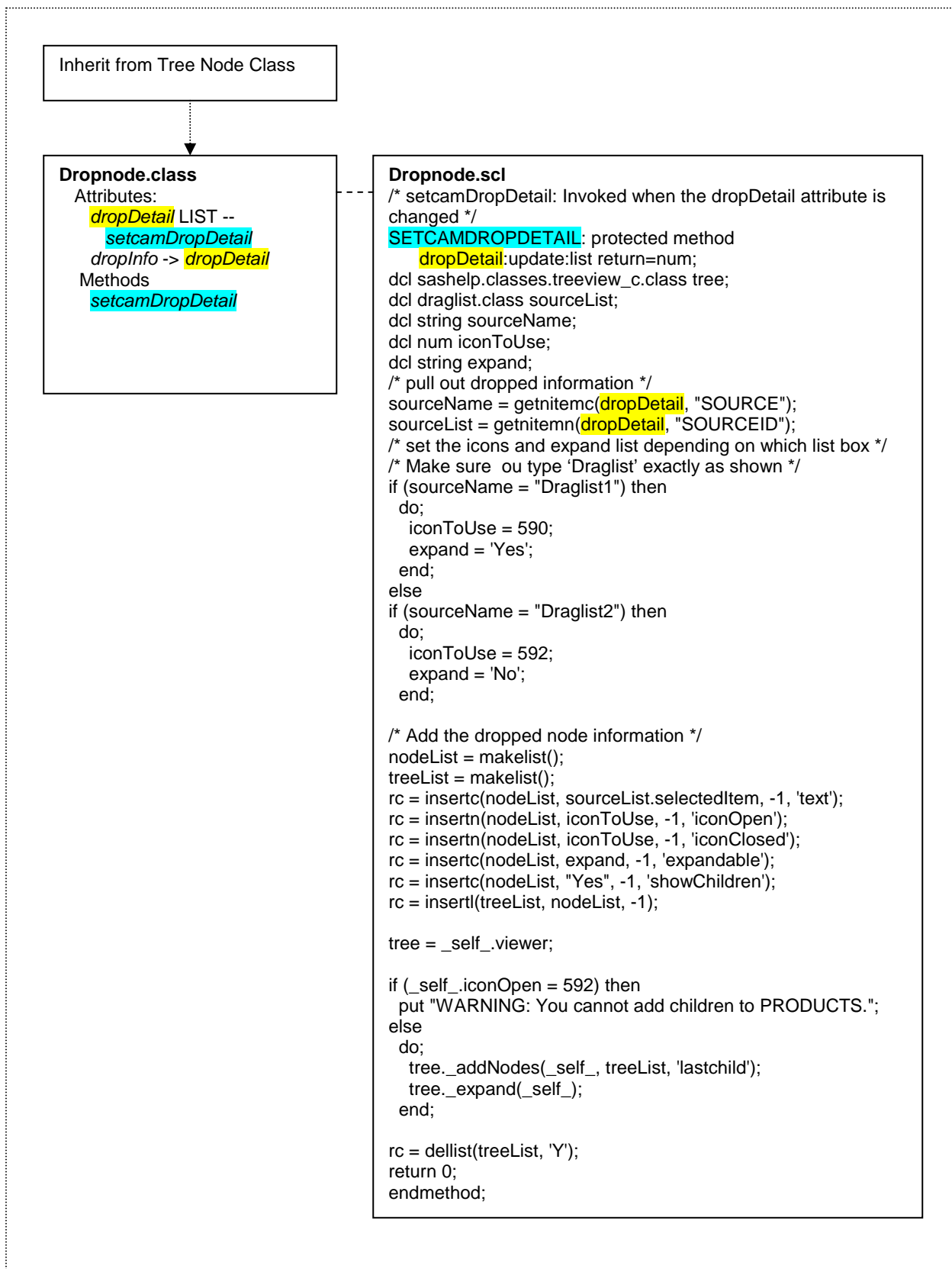7. Save and compile **Dropnode.scl**.

Inherit from Tree Node Class

Dropnode.class
  Attributes:
    dropDetail LIST --
      setcamDropDetail
    dropInfo -> dropDetail
  Methods
    setcamDropDetail

```
Dropnode.scl
/* setcamDropDetail: Invoked when the dropDetail attribute is
changed */
SETCAMDROPDETAIL: protected method
    dropDetail:update:list return=num;
dcl sashelp.classes.treeview_c.class tree;
dcl draglist.class sourceList;
dcl string sourceName;
dcl num iconToUse;
dcl string expand;
/* pull out dropped information */
sourceName = getnitemc(dropDetail, "SOURCE");
sourceList = getnitemn(dropDetail, "SOURCEID");
/* set the icons and expand list depending on which list box */
/* Make sure  ou type 'Draglist' exactly as shown */
if (sourceName = "Draglist1") then
  do;
    iconToUse = 590;
    expand = 'Yes';
  end;
else
if (sourceName = "Draglist2") then
  do;
    iconToUse = 592;
    expand = 'No';
  end;

/* Add the dropped node information */
nodeList = makelist();
treeList = makelist();
rc = insertc(nodeList, sourceList.selectedItem, -1, 'text');
rc = insertn(nodeList, iconToUse, -1, 'iconOpen');
rc = insertn(nodeList, iconToUse, -1, 'iconClosed');
rc = insertc(nodeList, expand, -1, 'expandable');
rc = insertc(nodeList, "Yes", -1, 'showChildren');
rc = insertl(treeList, nodeList, -1);

tree = _self_.viewer;

if (_self_.iconOpen = 592) then
  put "WARNING: You cannot add children to PRODUCTS.";
else
  do;
    tree._addNodes(_self_, treeList, 'lastchild');
    tree._expand(_self_);
  end;

rc = dellist(treeList, 'Y');
return 0;
endmethod;
```

Figure 3. Dropnode Class and SCL Code

CREATING THE TREEDEMO FRAME

Now that you have the new classes, you can start building the Frame for the application. The Frame will have two list boxes: one list contains the names of the sales people, and one list box contains the types of shoes that the company sells. These list boxes are created with the new **Draglist.class**. Each list box will get the data from the Variable Values List Model. The Frame will also contain a Tree View Control, and this control will be connected to the new **Dropnode class**. The SCL for the Frame entry, **Treedemo.scl**, initializes the tree. Figure 4 shows how these pieces fit together. Here are the steps for creating **Treedemo.frame**:

1. Issue the command `build sasuser.treednd.treedemo.frame`.

2. Drag the **Draglist.class** from the SAS Explorer window onto the Frame and place it on the top-left side. This will be named Draglist1 and will contain the names of the sales people.

3. Again, drag the **Draglist.class** from the SAS Explorer window onto the Frame and place the second list box on the bottom-left side. This will be named Draglist2.

4. From the Components window, drag a VariableValueslist model and drop it on the top list box (Draglist1). Edit the properties of the VariableValueslist1 model, and change the value of the *dataSet* attribute to `sashelp.class`. Change the value of the *variable* attribute to `name`.

5. Drag a second VariableValueslist model, and drop it on the bottom list box (Draglist2). Edit the properties of the VariableValueslist2 model, and change the value of *dataset* to `sashelp.shoes`, and change the value of the *variable* attribute to `products`.

6. From the Components window, drag a Tree View Control onto the Frame and place it on the right side. Drag the Tree View and make it a larger size so that more of the tree will be visible.

7. Edit the properties of the Treeview1 control and change the *nodeClass* attribute to the **Dropnode.class** that you created (`sasuser.treednd.dropnode.class`).

8. Modify the *dropInfo* attribute by clicking the ellipsis in the Value column, and changing the drop attribute to *dropDetail*.

9. Change the d*ropEnabled* attribute to YES.

10. Save these changes.

11. Include the code that's shown in Figure 5 as the Source Code for the Frame entry. This code initializes the tree and creates the top-level node.

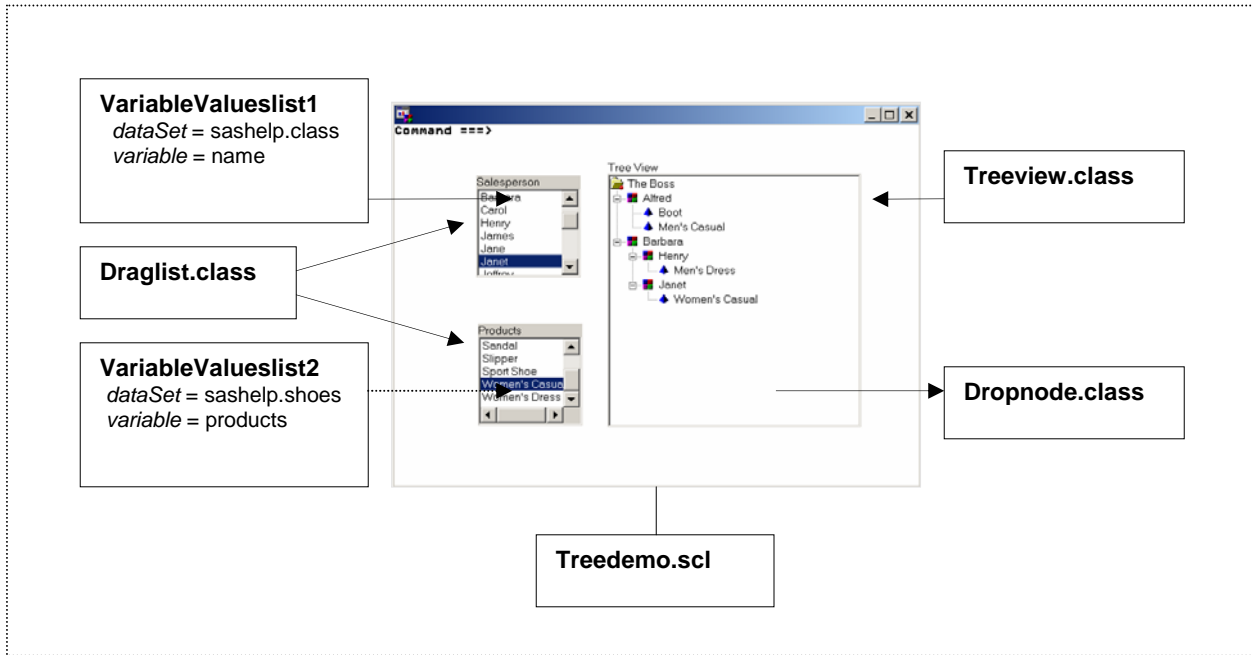12. Save **Treedemo.frame** and **Treedemo.scl**.

VariableValueslist1
  dataSet = sashelp.class
  variable = name

Draglist.class

VariableValueslist2
  dataSet = sashelp.shoes
  variable = products

Treeview.class

Dropnode.class

Treedemo.scl

Figure 4. Treedemo.Frame

Treedemo.frame
  Treeview1
  Draglist1
  Draglist2
  VariableValueslist1
  VariableValueslist2

Treedemo.scl
```
dcl sashelp.classes.treeview_c.class tree;
init:
   tree = treeview1;
   link populate;
return;

populate:
 /* Make the main tree list. */
 treeList = makelist();
 /* Make the root node list. */
 rootList = makelist();
 rc = insertc(rootList, 'The Boss', -1, 'text');
 rc = insertc(rootList, 'Yes', -1, 'showChildren');
 rc = insertc(rootList, 'Yes', -1, 'expandable');

 rc = insertl(treeList, rootList, -1);

 tree._addNodes(0, treeList, 'firstchild');
 rc = dellist(treeList, 'Y');
return;
```
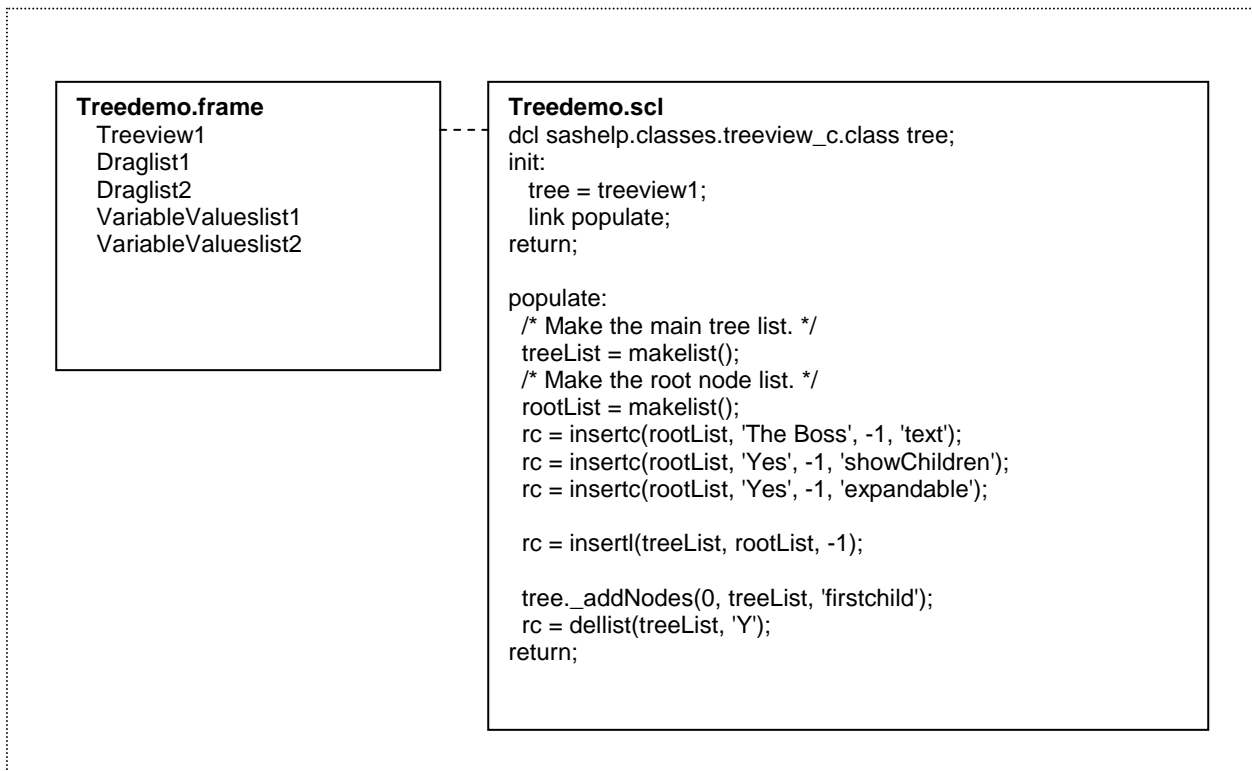
Figure 5.   Frame and SCL Code for Treedemo

COMPILE AND RUN APPLICATION

The application is now complete. Close all open entries and compile `treedemo.frame`. Run the application by issuing the command:

```
af c=sasuser.treednd.treedemo.frame
```

Now, select a name from the top list, left-click the item, drag the item to the tree, and drop it onto the node labeled 'The Boss'. If the item does not appear, double-click the icon next to 'The Boss'. Now, you can continue to drag names and products and drop them on the tree.

If you have problems with your example, carefully examine all the steps. Also, make sure that your SCL code exactly matches the code in this presentation. Literal strings such as "Draglist1" must match exactly.

CONCLUSION

Dragging and dropping is a very powerful and very easy-to-use tool. Using the drag-and-drop feature with the Tree View Control gives you an extremely powerful tool for dynamically building a tree hierarchy. The example that's provided in this paper will get you started in building your own drag-and-drop application.

RESOURCES

Massengill, Darrell. 2006. "SAS/AF$^®$ Software $--$Building a Tree View Hierarchy Using Drag and Drop". SUGI 31 "SAS Coder's Corner" Handout and Example Source Code Download. Available support.sas.com/rnd/papers.

RECOMMENDED READING

Curley, Lynn. 2006. "SAS/AF$^®$ Rises Again: Enhancements in SAS$^®$ 9.2". SUGI 31 Proceedings. Cary, NC: SAS Institute Inc.

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the authors:

> Darrell Massengill
> SAS Institute Inc.
> SAS Campus Drive
> Cary, NC 27513
> Phone: (919) 677-8000
> Darrell.Massengill@sas.com
>
> Scott Jackson
> SAS Institute Inc.
> SAS Campus Drive
> Cary, NC 27513
> Phone: (919) 677-8000
> Scott.Jackson@sas.com

SAS and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. ® indicates USA registration.

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