How To Use the SAS/AF® Frame Orgchart Object

Thomas Miron
Miron InfoTec, Inc., Madison, WI

Tutorial Topics

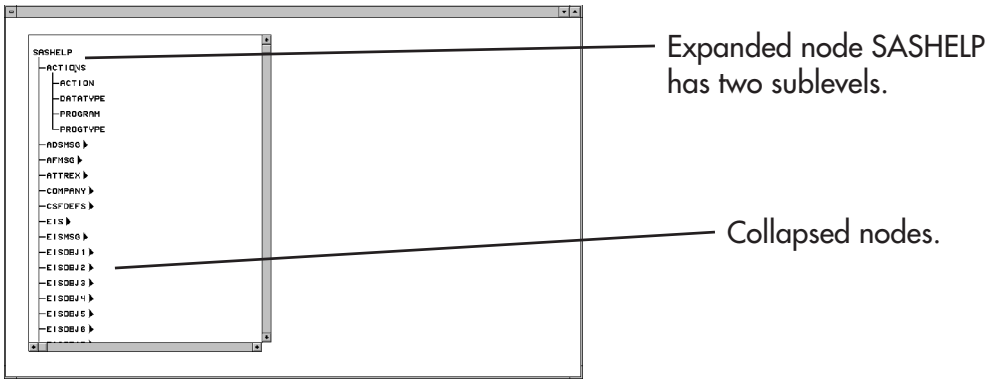
- Overview
- DEMO1 - Basic procedure for creating a chart with dynamic data set assignment
- DEMO2 - How to control the initial color of nodes
- DEMO3 - How to run a process when a node is selected by the user

Overview

What is an Orgchart Object?

An orgchart shows the hierarchical relationships among data items. The SAS/AF Frame organizational chart class is available with SAS release 6.11. The orgchart class entry is SASHELP.FSP.ORGCHART.CLASS. Orgchart is in the default BUILD.RESOURCE entry so it is selectable from the default Make List when building a frame entry. Orgcharts can display data from SAS data sets or SCL lists that conform to the specific orgchart data structure. In this tutorial we will be using a SAS data set as the orgchart source data.

Fig. 1
A directory style orgchart with three levels. Some levels are collapsed as indicated by the solid right-pointing arrow.



Why use the orgchart?

Many databases are structured as a series of tables, but a series of two-dimensional tables related by key columns may not correspond to the way users visualize or interact with the data. An orgchart may be a more natural representation of data that are visualized as a hierarchy rather than series of tables.

Creating a SAS/AF Orgchart Application

The orgchart class must be used within the context of a SAS/AF Frame application. You must be running the SAS system in an environment that supports the development of Frame applications.

Assumptions About Your Knowledge

This tutorial assumes that you are familiar with the following concepts and techniques:

- Starting and using BUILD to create SAS/AF Frame applications
- Creating objects on a frame and accessing attribute windows
- Basic DATA step and SQL programs
- Basic SCL statements and functions
- SCL lists

DEMO1: The LIB/MEM/VAR Application with Dynamic Data Set Assignment

This tutorial is a step-by-step look at the tasks required to create the LIB/MEM/VAR application shown in Fig. 1. The first version of this application is DEMO1. DEMO1 displays a library/member/variable hierarchy for the SASHELP and SASUSER libraries. In addition, this application demonstrates how to dynamically assign a data set to an orgchart at run time. Documentation in the SAS/AF Frame Dictionary covers the tasks required to assign a data set to a chart at build time via the orgchart attributes window.

Task List

1. Create frame
2. Create orgchart object
3. Set object attributes
4. Create frame SCL

1. Create a Frame

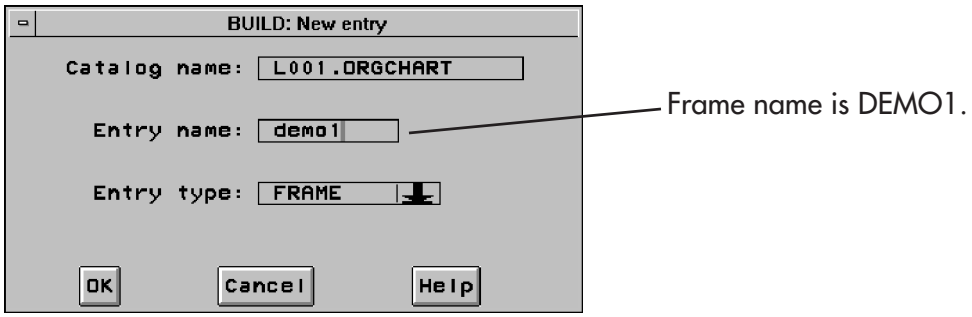


Fig. 2
Create a new frame from the Build window by selecting File, New, and Entry from the Build pmenu.

2. Place the Orgchart Object

Right click in the new frame and select Make to display the Make list.

Fig. 3
Make list. Select the Organizational chart class.

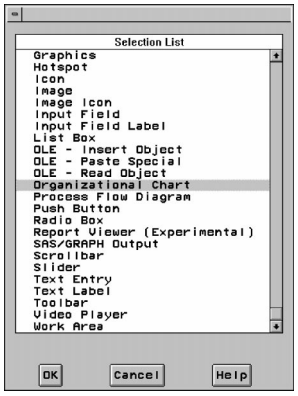
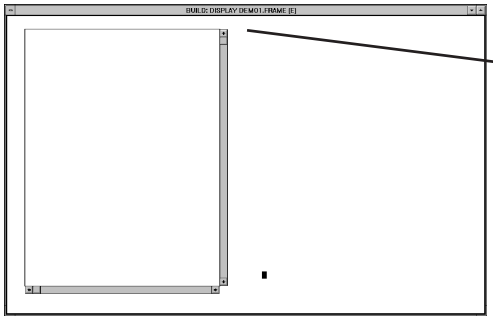


Fig. 4
A new frame with a placed orgchart object.



You can resize the orgchart by grabbing a corner with the left mouse button held down and dragging to size.

3. Orgchart Attributes

Orgchart attributes are set in a series of windows, beginning with the main attribute window shown in Fig. 5. Right click the orgchart and select Object Attributes to display the attributes window.

Fig. 5
Main orgchart attributes window. Select the Help button for a description of attributes not covered in this tutorial.

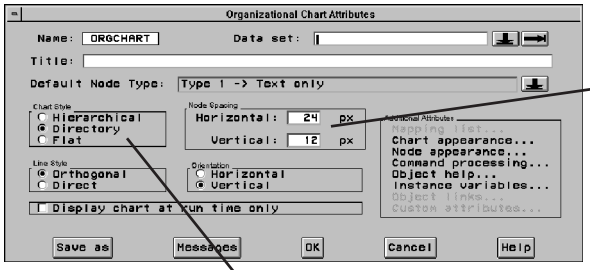
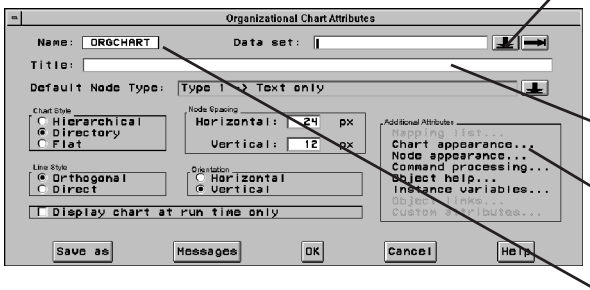


Chart style is set to Directory.

When an orgchart will display many nodes, you will probably want to explicitly set the horizontal and vertical node spacing for best appearance and efficient use of display real estate. Node spacing units are pixels so the visual effect of a setting varies with display resolution.

Data Set Name and Title

Fig. 6
Attributes window.



We will assign the data set name at runtime so select "Determine value at runtime" from the down-arrow drop-down list. The Data set field remains blank.

Title remains blank.

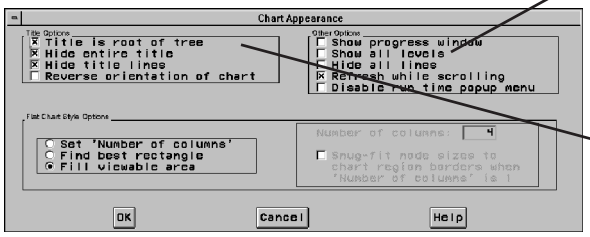
Chart appearance item. See Fig. 7.

Key a widget (object) name in the Name field. The name of this orgchart object is "ORGCHART." The name is arbitrary.

Chart Appearance Attributes

Select the Chart appearance item from the main attribute window to display the Chart Appearance window.

Fig. 7
Chart appearance window



The show all levels checkbox determines if all or just the first two levels are shown on the initial display. For DEMO1 it is left unchecked.

All charts must have a root level. Here, we are using the title as the root. The title is blank and Hide Entire Title is checked so nothing appears on the chart, but the root requirement is satisfied.

4. Frame SCL

Following is the complete frame SCL for the DEMO1 application.

```
/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
/*                                                                   */
/* HOW TO USE THE SAS/AF FRAME ORGCHART OBJECT                       */
/*                                                                   */
/* TOM MIRON                                                         */
/*                                                                   */
/* MIRON INFOTEC, INC., MADISON, WI   (608) 255-3531                 */
/*                                                                   */
/*                                                                   */
/* DEMO1 - LIBRARY/MEMBER/COLUMN CHART                               */
/*                                                                   */
/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

INIT:

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
/*                                                                   */
/*  CREATE THE TABLE DISPLAYED IN THE ORGCHART                       */
/*                                                                   */
/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

   /* INTERMEDIATE TABLE: COLINFO */
   submit continue sql;
      create table work.colinfo as
      select
          libname
         ,memname
         ,name
```

All the following code is in the frame's INIT section.

The SQL creates a table with the libname, member name (data set name), and variable name of all variables in all data sets in the SASHELP and SASUSER libraries. See the Help system or SAS Technical Report P-222 page 286 for more on the DICTIONARY views.

```
      from
         dictionary.columns

      where
         lowcase( libname ) in( 'sashelp', 'sasuser' )
         and lowcase( memtype ) = 'data'

      order by
          libname
         ,memname
         ,name

      ;
   endsubmit;

   /* CHART TABLE:  COLUMNS */
   submit continue;
      data columns;
      attrib

         objname
            length=$8
            label="Object Name"

         level
            length=8
            label="Org Chart Level"

      ;
      set colinfo;
         by libname memname name;

      /* THE LIBREF IS CHART LEVEL 1 */
      if first.libname then do;
         objname = libname;
         level = 1;
         output;
      end;

      /* THE MEMBER (DATA SET) NAME IS CHART LEVEL 2 */
      if first.memname then do;
         objname = memname;
         level = 2;
         output;
      end;

      /* THE VARIABLE NAME IS CHART LEVEL 3 */
      if first.name then do;
         objname = name;
         level = 3;
         output;
      end;
   run;
   endsubmit;

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
/*                                                                 */
/* IN THE ATTRIBUTES WINDOW NO DATA SOURCE IS SPECIFIED FOR THE    */
/* ORGCHART BECAUSE THE DATA SET WE WANT TO USE IS CREATED AT      */
/* RUNTIME.                                                        */
/* THERE IS NO METHOD TO EXPLICITLY ASSIGN A DATA SOURCE TO AN     */
/* ORGCHART OBJECT, BUT WE CAN DO IT BY SETTING INSTANCE           */
/* VARIABLES "DATASET" AND “MAPLIST”.                              */
/*                                                                 */
/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

   /* GET THE OBJECT ID OF THE ORGCHART. THIS IS THE LIST ID OF  */
   /* OF THE  OBJECT.                                           */
   call notify( '.', '_get_widget_', 'orgchart', chartid );

   /* SET THE INSTANCE VARIABLE "DATASET" TO THE TABLE CREATED   */
   /* ABOVE.                                                    */
   chartid = setnitemc( chartid, 'work.columns', 'dataset' );
```

End of SQL step

The COLUMNS data set will be displayed in the orgchart.

Read the table created in the preceding SQL step.

The DATA step uses BY variable processing to create a table with two variables: OBJNAME is the name of the libref, data set, or variable to be represented by a chart node, LEVEL indicates the hierarchical level of the object. Librefs are level 1, data sets level 2, and variables level 3.

End of DATA step that creates the table displayed in the orgchart.

Get the object id of the chart. This is also the list id of the object.

Assign the data set name to the instance variable (list item) “dataset.”

```
   /* CREATE A MAPPING LIST */
   maplist = makelist();
   maplist = setnitemc( maplist, 'objname', 'text' );
   maplist = setnitemc( maplist, 'level', 'level' );
   maplist = setnitemc( maplist, 'level', 'current_node' );

   /* ASSIGN THE MAPLIST AS THE INSTANCE VAR "MAPLIST" */
   chartid = setniteml( chartid, maplist, 'maplist' );

   /* NOW TELL THE CHART TO REPOPULATE WITH ITS NEW DATA SOURCE    */
   call send( chartid, '_repopulate_' );

return;
```

Create and assign items to a list ("maplist" in this case) that will be assigned as the chart node mapping list. This list tells the chart how data set variables correspond to chart node variables. See Orgchart documentation for more on node variables.

Assign the map list id as the chart instance variable "maplist."

Send the chart the _REPOPULATE_ method to populate the chart from the data set.

End of INIT section.

DEMO1 Display

By default, two levels are displayed: the root (hidden title) and the first data level (the libref). See Fig. 7.

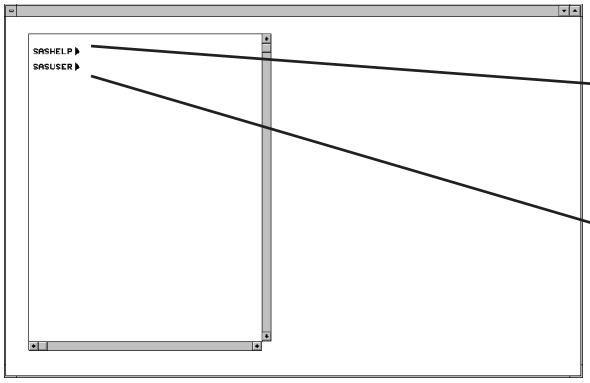


Fig. 8
The initial display of the DEMO1 application.

The title is the chart root but does not appear because it's hidden. See Fig. 7.

The right arrow indicates that the node is collapsed.

Double Click on a Collapsed Node Expands the Node

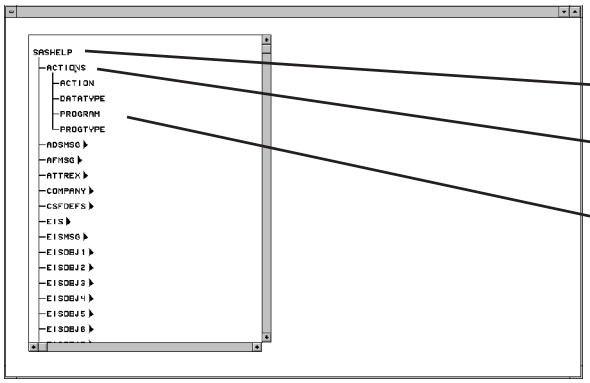


Fig. 9
Expanded SASHELP/ ACTIONS node shows data sets and variables.

Library (level=1)

Data set (level=2)

Variables (level=3)

DEMO2 - Add Node Color

The DEMO2 application is a modification of DEMO1. DEMO2 shows you how to add color to nodes. The color is static, i.e., it is assigned once, via a variable in the chart data set. When assigned in this manner, the color does not change based on runtime actions. The orgchart class does provide a method (_SET_COLOR_) that you can use to assign node color at runtime. Assigning a static color is useful when you want the color to correspond to a node's level in the chart hierarchy or some other static node attribute.

Task List

1. Add color variable to the chart data set (modify frame SCL).

2. Add the foreground color item to the mapping list (modify frame SCL).

1. Add Color Variable (partial frame INIT section)

```
/* CHART TABLE:  COLUMNS */
  submit continue;
     data columns;
     attrib
        objname
           length=$8
           label="Object Name"
        level
           length=8
           label="Org Chart Level"

        color
           length=$24
           label='Foreground Color'
     ;
     set colinfo;
        by libname memname name;

     /* THE LIBREF IS CHART LEVEL 1 */
     if first.libname then do;
        objname = libname;
        level = 1;
        color = 'red';
        output;
     end;

     /* THE MEMBER (CATALOG) NAME IS CHART LEVEL 2 */
     if first.memname then do;
        objname = memname;
        level = 2;
        color = 'blue';
        output;
     end;

     /* THE OBJECT (ENTRY) NAME IS CHART LEVEL 3 */
     if first.name then do;
        objname = name;
        level = 3;
        color = 'green';
        output;
     end;
  run;
  endsubmit;
```

Create another variable in the chart data set, COLUMNS, to hold the color of each node. The variable name is arbitrary, here its's called COLOR.

Assign a value to COLOR for each level.

2. Add Color Control Node Variable to the Mapping List

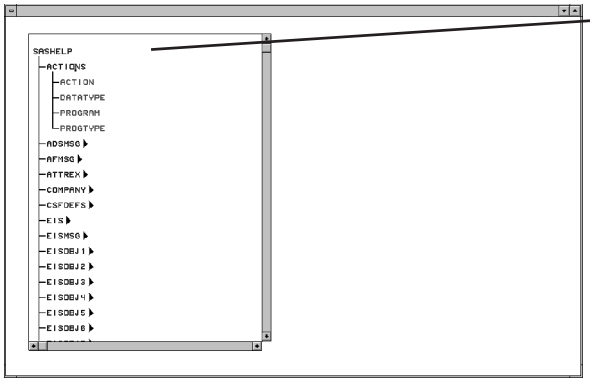
The map list assigns the value of data set variable COLOR as the node variable FOREGROUND_COLOR.

```
   /* CREATE A MAPPING LIST */
   maplist = makelist();
   maplist = setnitemc( maplist, 'objname', 'text' );
   maplist = setnitemc( maplist, 'level', 'level' );
   maplist = setnitemc( maplist, 'level', 'current_node' );
   maplist = setnitemc( maplist, 'color', 'foreground_color' );

   /* ASSIGN MAPLIST AS THE INSTANCE VAR "MAPLIST" */
   chartid = setniteml( chartid, maplist, 'maplist' );

   /* NOW TELL THE CHART TO REPOPULATE WITH ITS NEW DATA SOURCE    */
   call send( chartid, '_repopulate_' );
return;
```

The maplist item "foreground_color" determines the foreground color of a node. There is also an optional "background_color" item.

DEMO2 Display

Fig. 10
DEMO2 display with some nodes expanded.



Each level is displayed in the color assigned via the maplist.

DEMO3 - Add Node Select Processing

DEMO3 is a modification of DEMO2. DEMO3 shows how to add node selection processing. We will add the ability to view a data set when the user single-clicks on a data set node. In this example, the Hide/unhide children upon double click attribute is turned off in the Select Action attributes window (see Fig. 11) and Show all levels is checked in the Chart Appearance window (See Fig. 7). The net result is the entire chart is always displayed to the user. Expand/collapse is disabled because we want to capture a click event and distinguishing between a single-click (select node) and double-click (expand/collaspe) involves techniques beyond the scope of this tutorial.

Task List

1. Add SCL LENGTH statement to declare new character variables and a non-executable LINK statement to avoid compile-time warning message (modify frame SCL).
2. Add a node select routine to capture the selected data set name and its parent library, then call the FSVIEW function to view the data set (modify frame SCL).
3. Name the select action routine in the Select Action attribute window.

1. LENGTH and LINK statements

The node select routine is not executed via a LINK statement in the SCL. When the code is compiled the compiler will complain that a routine exists that is never linked to. Place a LINK statement outside of any executable section to eliminate the message. The following statements are placed before the frame's INIT section.

```
length
   name $8  lib $8
;

link click1;
```

LENGTH statement to declare NAME and LIB as character.

Non-executable LINK statement. (See explanation above.)

2. Node Select Routine

This routine is executed when a node is selected. See Fig. 11.

```
/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
/*                                                                   */
/*  CLICK1                                                           */
/*                                                                   */
/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

CLICK1:

   /* FIRST GET THE NODE ID OF THE SELECTED NODE */
   call send( chartid, '_get_selected_', widgetid, nodeid );

   /* MAKE SURE A NODE WAS SELECTED. IF NOT THEN RETURN NOW. */
   if not nodeid then return;

   /* GET INFO FOR THE SELECTED NODE: TEXT (OBJECT NAME) AND LEVEL */
   call send( chartid, '_get_current_', nodeid, 'text level', node_list );

   /* CHECK THE NODE LEVEL. WE'RE ONLY INTERESTED IN MEMBERS      */
   /* (SAS DATA SETS), LEVEL = 2.                                 */
   level = getnitemn( node_list, 'level' );
   if level ne 2 then return;

   /* GET THE NAME (TEXT) ASSOCIATED WITH THE NODE */
   name = getnitemc( node_list, 'text' );

   /* WE NEED THE PARENT OF THE NODE, I.E. THE LIBRARY. THE TEXT ITEM */
   /* IN THE NODE LIST HOLDS THE NAME OF THE LIBRARY.                 */
   call send( chartid, '_get_parent_', nodeid, 'text', parent_list, 'n' );
   lib = getnitemc( parent_list, 'text' );

   /* GOT THE LIBRARY AND MEMBER NAME SO BROWSE THE DATA SET */
   call fsview( lib || '.' || name );

return;
```

CLICK1 is placed after the frame's INIT section as a separate routine.

Call method to get the nodeid of the selected node in SCL variable NODEID.

NODEID could be 0 if a node was not directly selected, in that case return immediately.

Call method to set the TEXT and LEVEL node variables in list NODE_LIST.

If level is not 2, i.e., the node does not name a SAS data set, then return now.

The TEXT item is the data set name.

The parent of this node will be the libref for the data set.

Form libref.memname and call FSVIEW to display the data set.

3. Name the Select Routine in the Select Action Window

The node selection routine is named in the Select Attribute window. To get to the window bring up the object attributes window, select Node appearance, Select action.

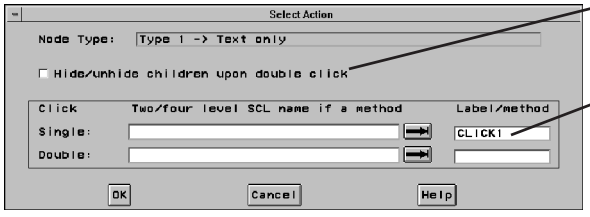


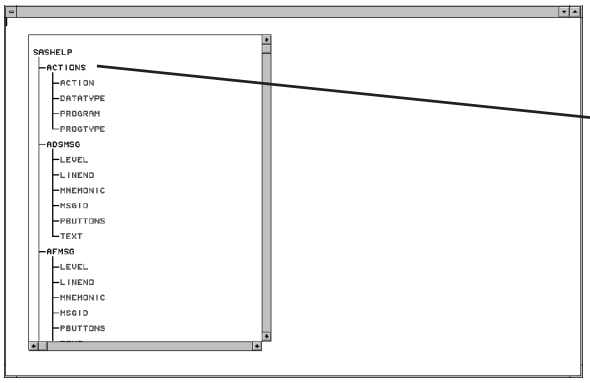
Fig. 11
Node Select Action attribute window.

Expand/collapse behavior check box.

CLICK1 is named in the Select Action attribute window as the routine to run when the user clicks on a node.

DEMO3 Display

Fig. 12
For the DEMO3 application all orgchart levels are displayed on initial display because Show all levels has been checked. See Fig. 7.

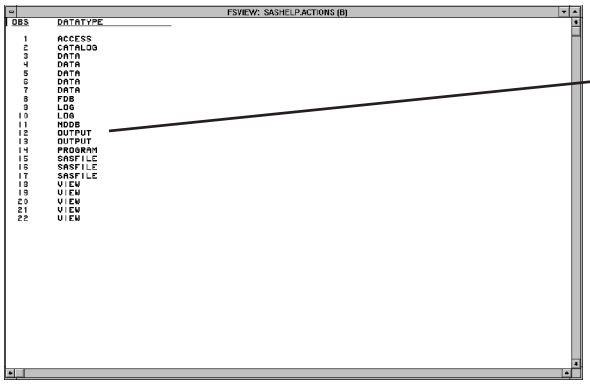


If the user selects a SAS data set name (level=2) that data set is displayed by FSVIEW. See below.

Data Set Node Select

When the user clicks on a data set level node, the data set is displayed via the FSVIEW function.

Fig. 13
FSVIEW display of ACTIONS selected above.



The selected data set (ACTIONS in this case) is displayed.

How To Use the SAS/AF® Frame Orgchart Object

Thomas Miron
Miron InfoTec, Inc., Madison, WI

Tutorial Topics

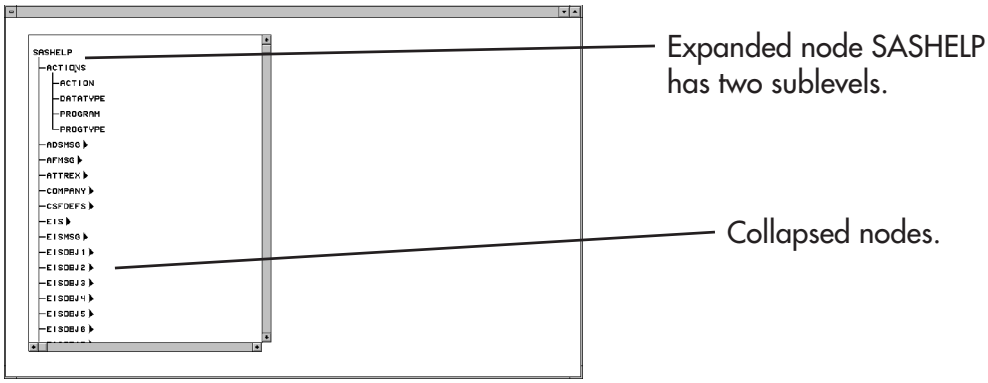
- Overview
- DEMO1 - Basic procedure for creating a chart with dynamic data set assignment
- DEMO2 - How to control the initial color of nodes
- DEMO3 - How to run a process when a node is selected by the user

Overview

What is an Orgchart Object?

An orgchart shows the hierarchical relationships among data items. The SAS/AF Frame organizational chart class is available with SAS release 6.11. The orgchart class entry is SASHELP.FSP.ORGCHART.CLASS. Orgchart is in the default BUILD.RESOURCE entry so it is selectable from the default Make List when building a frame entry. Orgcharts can display data from SAS data sets or SCL lists that conform to the specific orgchart data structure. In this tutorial we will be using a SAS data set as the orgchart source data.

Fig. 1
A directory style orgchart with three levels. Some levels are collapsed as indicated by the solid right-pointing arrow.



Why use the orgchart?

Many databases are structured as a series of tables, but a series of two-dimensional tables related by key columns may not correspond to the way users visualize or interact with the data. An orgchart may be a more natural representation of data that are visualized as a hierarchy rather than series of tables.

Creating a SAS/AF Orgchart Application

The orgchart class must be used within the context of a SAS/AF Frame application. You must be running the SAS system in an environment that supports the development of Frame applications.

Assumptions About Your Knowledge

This tutorial assumes that you are familiar with the following concepts and techniques:

- Starting and using BUILD to create SAS/AF Frame applications
- Creating objects on a frame and accessing attribute windows
- Basic DATA step and SQL programs
- Basic SCL statements and functions
- SCL lists

DEMO1: The LIB/MEM/VAR Application with Dynamic Data Set Assignment

This tutorial is a step-by-step look at the tasks required to create the LIB/MEM/VAR application shown in Fig. 1. The first version of this application is DEMO1. DEMO1 displays a library/member/variable hierarchy for the SASHELP and SASUSER libraries. In addition, this application demonstrates how to dynamically assign a data set to an orgchart at run time. Documentation in the SAS/AF Frame Dictionary covers the tasks required to assign a data set to a chart at build time via the orgchart attributes window.

Task List

1. Create frame
2. Create orgchart object
3. Set object attributes
4. Create frame SCL

1. Create a Frame

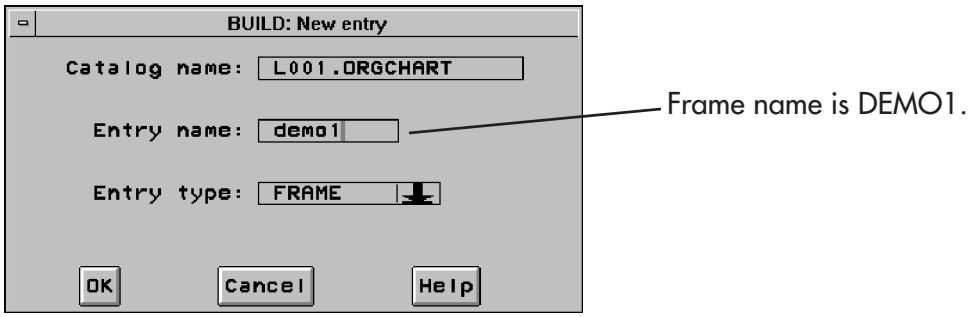


Fig. 2
Create a new frame from the Build window by selecting File, New, and Entry from the Build pmenu.

2. Place the Orgchart Object

Right click in the new frame and select Make to display the Make list.

Fig. 3
Make list. Select the Organizational chart class.

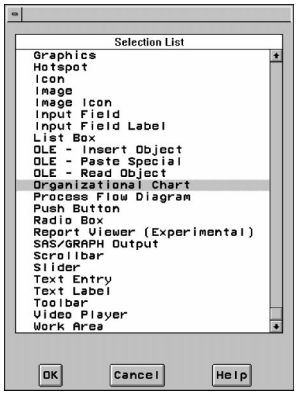
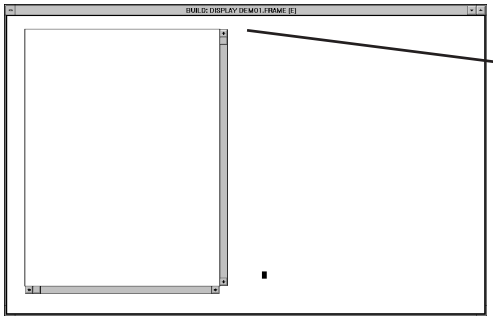


Fig. 4
A new frame with a placed orgchart object.



You can resize the orgchart by grabbing a corner with the left mouse button held down and dragging to size.

3. Orgchart Attributes

Orgchart attributes are set in a series of windows, beginning with the main attribute window shown in Fig. 5. Right click the orgchart and select Object Attributes to display the attributes window.

Fig. 5
Main orgchart attributes window. Select the Help button for a description of attributes not covered in this tutorial.

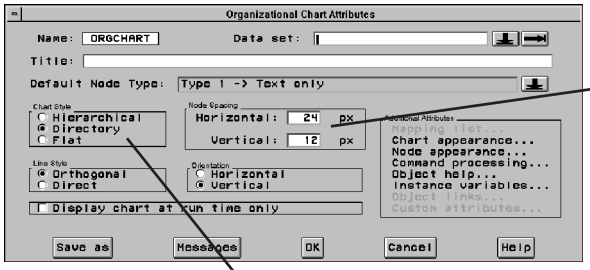
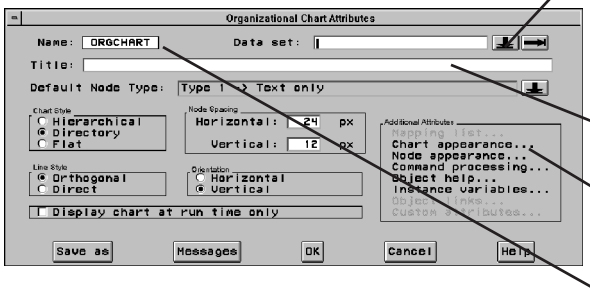


Chart style is set to Directory.

When an orgchart will display many nodes, you will probably want to explicitly set the horizontal and vertical node spacing for best appearance and efficient use of display real estate. Node spacing units are pixels so the visual effect of a setting varies with display resolution.

Data Set Name and Title

Fig. 6
Attributes window.



We will assign the data set name at runtime so select "Determine value at runtime" from the down-arrow drop-down list. The Data set field remains blank.

Title remains blank.

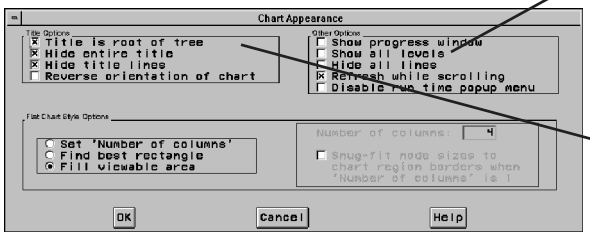
Chart appearance item. See Fig. 7.

Key a widget (object) name in the Name field. The name of this orgchart object is "ORGCHART." The name is arbitrary.

Chart Appearance Attributes

Select the Chart appearance item from the main attribute window to display the Chart Appearance window.

Fig. 7
Chart appearance window



The show all levels checkbox determines if all or just the first two levels are shown on the initial display. For DEMO1 it is left unchecked.

All charts must have a root level. Here, we are using the title as the root. The title is blank and Hide Entire Title is checked so nothing appears on the chart, but the root requirement is satisfied.

4. Frame SCL

Following is the complete frame SCL for the DEMO1 application.

```
/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
/*                                                                   */
/* HOW TO USE THE SAS/AF FRAME ORGCHART OBJECT                       */
/*                                                                   */
/* TOM MIRON                                                         */
/*                                                                   */
/* MIRON INFOTEC, INC., MADISON, WI   (608) 255-3531                 */
/*                                                                   */
/*                                                                   */
/* DEMO1 - LIBRARY/MEMBER/COLUMN CHART                               */
/*                                                                   */
/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

INIT:

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
/*                                                                   */
/*  CREATE THE TABLE DISPLAYED IN THE ORGCHART                       */
/*                                                                   */
/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

   /* INTERMEDIATE TABLE: COLINFO */
   submit continue sql;
      create table work.colinfo as
      select
          libname
         ,memname
         ,name
```

All the following code is in the frame's INIT section.

The SQL creates a table with the libname, member name (data set name), and variable name of all variables in all data sets in the SASHELP and SASUSER libraries. See the Help system or SAS Technical Report P-222 page 286 for more on the DICTIONARY views.

```
      from
         dictionary.columns

      where
         lowcase( libname ) in( 'sashelp', 'sasuser' )
         and lowcase( memtype ) = 'data'

      order by
          libname
         ,memname
         ,name

      ;
   endsubmit;

   /* CHART TABLE:  COLUMNS */
   submit continue;
      data columns;
      attrib

         objname
            length=$8
            label="Object Name"

         level
            length=8
            label="Org Chart Level"

      ;
      set colinfo;
         by libname memname name;

      /* THE LIBREF IS CHART LEVEL 1 */
      if first.libname then do;
         objname = libname;
         level = 1;
         output;
      end;

      /* THE MEMBER (DATA SET) NAME IS CHART LEVEL 2 */
      if first.memname then do;
         objname = memname;
         level = 2;
         output;
      end;

      /* THE VARIABLE NAME IS CHART LEVEL 3 */
      if first.name then do;
         objname = name;
         level = 3;
         output;
      end;
   run;
   endsubmit;

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
/*                                                                 */
/* IN THE ATTRIBUTES WINDOW NO DATA SOURCE IS SPECIFIED FOR THE    */
/* ORGCHART BECAUSE THE DATA SET WE WANT TO USE IS CREATED AT      */
/* RUNTIME.                                                        */
/* THERE IS NO METHOD TO EXPLICITLY ASSIGN A DATA SOURCE TO AN     */
/* ORGCHART OBJECT, BUT WE CAN DO IT BY SETTING INSTANCE           */
/* VARIABLES "DATASET" AND “MAPLIST”.                              */
/*                                                                 */
/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

   /* GET THE OBJECT ID OF THE ORGCHART. THIS IS THE LIST ID OF  */
   /* OF THE  OBJECT.                                           */
   call notify( '.', '_get_widget_', 'orgchart', chartid );

   /* SET THE INSTANCE VARIABLE "DATASET" TO THE TABLE CREATED   */
   /* ABOVE.                                                    */
   chartid = setnitemc( chartid, 'work.columns', 'dataset' );
```

End of SQL step

The COLUMNS data set will be displayed in the orgchart.

Read the table created in the preceding SQL step.

The DATA step uses BY variable processing to create a table with two variables: OBJNAME is the name of the libref, data set, or variable to be represented by a chart node, LEVEL indicates the hierarchical level of the object. Librefs are level 1, data sets level 2, and variables level 3.

End of DATA step that creates the table displayed in the orgchart.

Get the object id of the chart. This is also the list id of the object.

Assign the data set name to the instance variable (list item) “dataset.”

```
   /* CREATE A MAPPING LIST */
   maplist = makelist();
   maplist = setnitemc( maplist, 'objname', 'text' );
   maplist = setnitemc( maplist, 'level', 'level' );
   maplist = setnitemc( maplist, 'level', 'current_node' );

   /* ASSIGN THE MAPLIST AS THE INSTANCE VAR "MAPLIST" */
   chartid = setniteml( chartid, maplist, 'maplist' );

   /* NOW TELL THE CHART TO REPOPULATE WITH ITS NEW DATA SOURCE    */
   call send( chartid, '_repopulate_' );

return;
```

Create and assign items to a list ("maplist" in this case) that will be assigned as the chart node mapping list. This list tells the chart how data set variables correspond to chart node variables. See Orgchart documentation for more on node variables.

Assign the map list id as the chart instance variable "maplist."

Send the chart the _REPOPULATE_ method to populate the chart from the data set.

End of INIT section.

DEMO1 Display

By default, two levels are displayed: the root (hidden title) and the first data level (the libref). See Fig. 7.

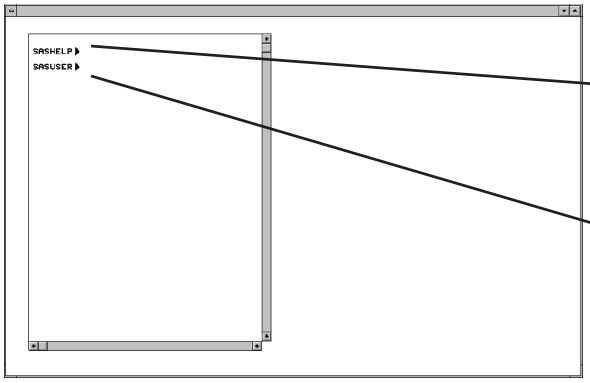


Fig. 8
The initial display of the DEMO1 application.

The title is the chart root but does not appear because it's hidden. See Fig. 7.

The right arrow indicates that the node is collapsed.

Double Click on a Collapsed Node Expands the Node

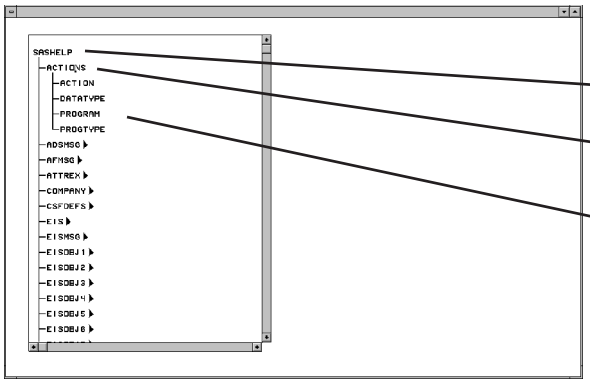


Fig. 9
Expanded SASHELP/ ACTIONS node shows data sets and variables.

Library (level=1)

Data set (level=2)

Variables (level=3)

DEMO2 - Add Node Color

The DEMO2 application is a modification of DEMO1. DEMO2 shows you how to add color to nodes. The color is static, i.e., it is assigned once, via a variable in the chart data set. When assigned in this manner, the color does not change based on runtime actions. The orgchart class does provide a method (_SET_COLOR_) that you can use to assign node color at runtime. Assigning a static color is useful when you want the color to correspond to a node's level in the chart hierarchy or some other static node attribute.

Task List

1. Add color variable to the chart data set (modify frame SCL).

2. Add the foreground color item to the mapping list (modify frame SCL).

1. Add Color Variable (partial frame INIT section)

```
/* CHART TABLE:  COLUMNS */
  submit continue;
     data columns;
     attrib
        objname
           length=$8
           label="Object Name"
        level
           length=8
           label="Org Chart Level"

        color
           length=$24
           label='Foreground Color'
     ;
     set colinfo;
        by libname memname name;

     /* THE LIBREF IS CHART LEVEL 1 */
     if first.libname then do;
        objname = libname;
        level = 1;
        color = 'red';
        output;
     end;

     /* THE MEMBER (CATALOG) NAME IS CHART LEVEL 2 */
     if first.memname then do;
        objname = memname;
        level = 2;
        color = 'blue';
        output;
     end;

     /* THE OBJECT (ENTRY) NAME IS CHART LEVEL 3 */
     if first.name then do;
        objname = name;
        level = 3;
        color = 'green';
        output;
     end;
  run;
  endsubmit;
```

Create another variable in the chart data set, COLUMNS, to hold the color of each node. The variable name is arbitrary, here its's called COLOR.

Assign a value to COLOR for each level.

2. Add Color Control Node Variable to the Mapping List

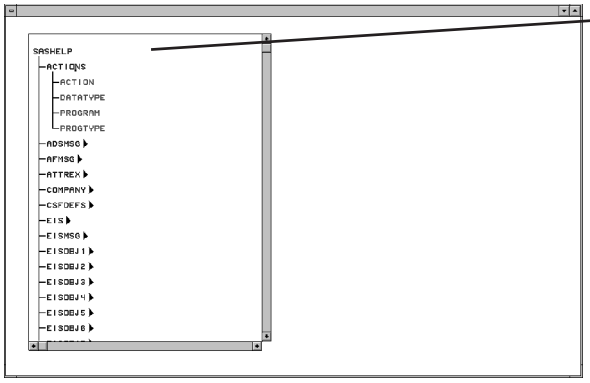
The map list assigns the value of data set variable COLOR as the node variable FOREGROUND_COLOR.

```
   /* CREATE A MAPPING LIST */
   maplist = makelist();
   maplist = setnitemc( maplist, 'objname', 'text' );
   maplist = setnitemc( maplist, 'level', 'level' );
   maplist = setnitemc( maplist, 'level', 'current_node' );
   maplist = setnitemc( maplist, 'color', 'foreground_color' );

   /* ASSIGN MAPLIST AS THE INSTANCE VAR "MAPLIST" */
   chartid = setniteml( chartid, maplist, 'maplist' );

   /* NOW TELL THE CHART TO REPOPULATE WITH ITS NEW DATA SOURCE    */
   call send( chartid, '_repopulate_' );
return;
```

The maplist item "foreground_color" determines the foreground color of a node. There is also an optional "background_color" item.

DEMO2 Display

Fig. 10
DEMO2 display with some nodes expanded.



Each level is displayed in the color assigned via the maplist.

DEMO3 - Add Node Select Processing

DEMO3 is a modification of DEMO2. DEMO3 shows how to add node selection processing. We will add the ability to view a data set when the user single-clicks on a data set node. In this example, the Hide/unhide children upon double click attribute is turned off in the Select Action attributes window (see Fig. 11) and Show all levels is checked in the Chart Appearance window (See Fig. 7). The net result is the entire chart is always displayed to the user. Expand/collapse is disabled because we want to capture a click event and distinguishing between a single-click (select node) and double-click (expand/collaspe) involves techniques beyond the scope of this tutorial.

Task List

1. Add SCL LENGTH statement to declare new character variables and a non-executable LINK statement to avoid compile-time warning message (modify frame SCL).
2. Add a node select routine to capture the selected data set name and its parent library, then call the FSVIEW function to view the data set (modify frame SCL).
3. Name the select action routine in the Select Action attribute window.

1. LENGTH and LINK statements

The node select routine is not executed via a LINK statement in the SCL. When the code is compiled the compiler will complain that a routine exists that is never linked to. Place a LINK statement outside of any executable section to eliminate the message. The following statements are placed before the frame's INIT section.

```
length
   name $8  lib $8
;

link click1;
```

LENGTH statement to declare NAME and LIB as character.

Non-executable LINK statement. (See explanation above.)

2. Node Select Routine

This routine is executed when a node is selected. See Fig. 11.

```
/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
/*                                                                   */
/*  CLICK1                                                           */
/*                                                                   */
/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

CLICK1:

   /* FIRST GET THE NODE ID OF THE SELECTED NODE */
   call send( chartid, '_get_selected_', widgetid, nodeid );

   /* MAKE SURE A NODE WAS SELECTED. IF NOT THEN RETURN NOW. */
   if not nodeid then return;

   /* GET INFO FOR THE SELECTED NODE: TEXT (OBJECT NAME) AND LEVEL */
   call send( chartid, '_get_current_', nodeid, 'text level', node_list );

   /* CHECK THE NODE LEVEL. WE'RE ONLY INTERESTED IN MEMBERS      */
   /* (SAS DATA SETS), LEVEL = 2.                                 */
   level = getnitemn( node_list, 'level' );
   if level ne 2 then return;

   /* GET THE NAME (TEXT) ASSOCIATED WITH THE NODE */
   name = getnitemc( node_list, 'text' );

   /* WE NEED THE PARENT OF THE NODE, I.E. THE LIBRARY. THE TEXT ITEM */
   /* IN THE NODE LIST HOLDS THE NAME OF THE LIBRARY.                 */
   call send( chartid, '_get_parent_', nodeid, 'text', parent_list, 'n' );
   lib = getnitemc( parent_list, 'text' );

   /* GOT THE LIBRARY AND MEMBER NAME SO BROWSE THE DATA SET */
   call fsview( lib || '.' || name );

return;
```

CLICK1 is placed after the frame's INIT section as a separate routine.

Call method to get the nodeid of the selected node in SCL variable NODEID.

NODEID could be 0 if a node was not directly selected, in that case return immediately.

Call method to set the TEXT and LEVEL node variables in list NODE_LIST.

If level is not 2, i.e., the node does not name a SAS data set, then return now.

The TEXT item is the data set name.

The parent of this node will be the libref for the data set.

Form libref.memname and call FSVIEW to display the data set.

3. Name the Select Routine in the Select Action Window

The node selection routine is named in the Select Attribute window. To get to the window bring up the object attributes window, select Node appearance, Select action.

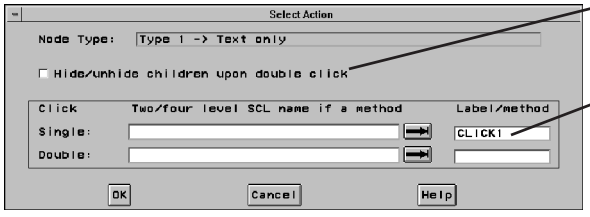


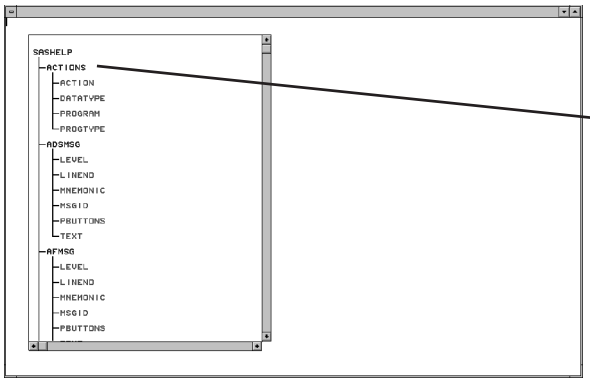
Fig. 11
Node Select Action attribute window.

Expand/collapse behavior check box.

CLICK1 is named in the Select Action attribute window as the routine to run when the user clicks on a node.

DEMO3 Display

Fig. 12
For the DEMO3 application all orgchart levels are displayed on initial display because Show all levels has been checked. See Fig. 7.

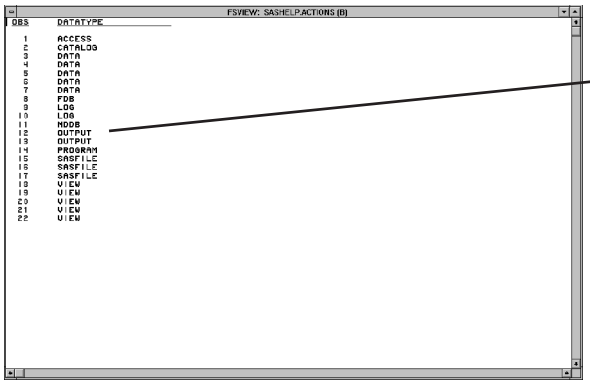


If the user selects a SAS data set name (level=2) that data set is displayed by FSVIEW. See below.

Data Set Node Select

When the user clicks on a data set level node, the data set is displayed via the FSVIEW function.

Fig. 13
FSVIEW display of ACTIONS selected above.



The selected data set (ACTIONS in this case) is displayed.