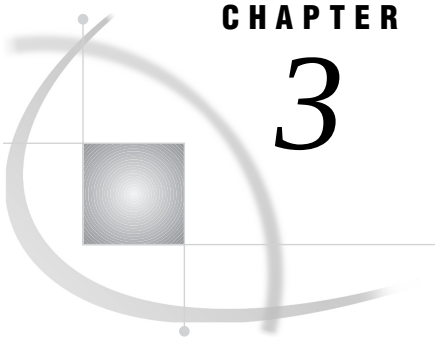


CHAPTER

3

Customizing the SAS Windowing Environment

Customizing Your Windowing Environment **36**
Understanding SAS in the X Environment **36**
X Window Managers **36**
SAS Window Session ID **37**
Workspace and Gravity in a SAS Session **37**
Window Types **37**
Using X Resources to Customize the Motif Interface **38**
Specifying X Resources **38**
Customizing SAS Resources **39**
Setting X Resources through the Preferences Dialog Box **40**
General Options **40**
DMS Options **41**
Editing Options **42**
Results Options **42**
Toolbox and Command Window Options **43**
Setting X Resources with the Resource Helper **44**
Starting Resource Helper from a SAS Session **44**
Starting Resource Helper from a Shell Prompt **45**
Changing Keys with Resource Helper **45**
Changing Colors with Resource Helper **47**
How Resource Helper Searches For X Resources **48**
Customizing Toolboxes and Toolsets **50**
Specifying Toolbox Resources **50**
Using the Tool Editor **51**
Changing the Appearance of the Entire Toolbox **52**
Changing an Existing Tool **52**
Adding Tools to the Toolbox **53**
Changing the Order of the Tools in the Toolbox **53**
Deleting Tools from the Toolbox **54**
Returning to the Default Settings **54**
Saving Changes to the Toolbox or Toolset **54**
Creating a New Toolbox **54**
Creating or Customizing an Application- or Window-Specific Toolbox **55**
Creating or Customizing an Application- or Window-Specific Toolset **55**
Customizing Key Definitions **55**
Defining Key Translations **56**
Determining Keysyms **57**
Modifying the SAS.keyboardTranslations Resource **58**
Modifying the keysWindowLabels Resource **59**
SAS Keyboard Action Names **59**
Examples **62**

Customizing Fonts **63**
Using the Host Fonts Dialog Box **63**
Specifying Font Resources **64**
How SAS Determines Which Font To Use **65**
Specifying Font Aliases **65**
Customizing Colors **66**
Using the SASCOLOR Window **66**
Using the COLOR Command **67**
Defining Color Resources **68**
Specifying RGB Values or Color Names for Foreground and Background Resources **68**
Defining Colors and Attributes for Window Elements (CPARMS) **69**
Controlling Contrast **71**
Controlling ODS Results **72**
Controlling Pull-down Menus **72**
Customizing Cut-and-Paste **73**
Customizing Session Workspace, Session Gravity, and Window Sizes **74**
Specifying User-defined Icons **76**
Miscellaneous Resources **77**
Configuring the SAS System for Host Editor Support **78**
Summary of SAS Resources **79**

Customizing Your Windowing Environment

The SAS windowing environment supports the use of X-based graphical user interfaces (GUIs). In UNIX environments, the SAS System provides an X Window System interface that is based on the Motif style.

Understanding SAS in the X Environment

The X Window System is a networked windowing system. If several machines are linked together in a network, you can run an X application program, or *client*, on one machine in the network and display it on any other machine in the network that is running an X *server*.

X Window Managers

Window managers are X clients that enable you to manage the windows on a display. The Motif interface to the SAS System can be used with any window manager that is compliant with the *Inter-Client Communication Conventions Manual* (ICCCM). Vendors provide at least one window manager with the X Window System environment. A common window manager is the Common Desktop Environment (CDE). If you are using another window manager such as the Tab Window Manager (twm), you should also read the documentation that is supplied by the vendor for that window manager.

All window managers perform the same basic functions, but they differ in their style and in their advanced functions. The appearance and function of the interface to SAS depends to some extent on your X window manager. Most window managers provide some kind of frame around a window. The window manager also governs the placement, sizing, stacking, and appearance of windows, as well as their interaction with the keyboard.

SAS Window Session ID

When you run the SAS System on an X workstation, the SAS System shares the display with other X applications, including other SAS sessions. To enable you to distinguish between different applications and SAS sessions, the SAS System generates a SAS window session ID for each session by appending a number to the application name, which by default is **SAS**. This session ID appears in the window title bar for each SAS window and in the window icon title. The SAS sessions are assigned sequentially. Your first SAS session is not assigned a number, so the session ID is **SAS**; your second SAS session is assigned the session ID **SAS2**, and so on. Although the default application name is **SAS**, you can use the **-name** X option to change the instance name. The instance name can be up to six characters long.

Workspace and Gravity in a SAS Session

When you use the SAS System on an X workstation, the display may be shared by many concurrent applications. When SAS windows from several different sessions and windows from other applications appear on the display, the display can become cluttered. To help alleviate this problem, the windows for a SAS session first appear within an *application workspace* (AWS). The AWS defines a rectangular region that represents a virtual display in which SAS windows are initially created. SAS attempts to position the AWS in relation to the upper-left corner of your display. In other words, the workspace gravitates toward a certain direction (*session gravity*) on the display. Some window manager configurations may override the placement that the SAS System has chosen for a window.

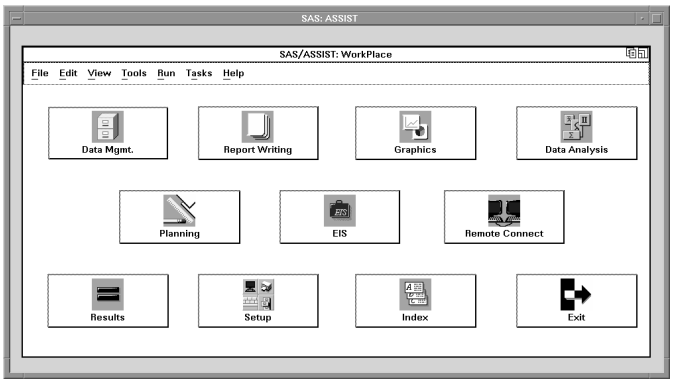
If you issue windowing commands or execute SAS System procedures that create new SAS windows, the same rules of initial position and size apply to these windows: they are initially placed in the SAS AWS. You can use the WSAVE command to save the current window positions (or geometry). See “Customizing Session Workspace, Session Gravity, and Window Sizes” on page 74 for details.

Window Types

The SAS System uses primary and interior windows. Some SAS applications consist of one or more primary windows controlled by the X window manager in addition to the interior windows controlled by the SAS System. The SAS windowing environment primary windows, as well as most SAS application windows, initially appear as *top-level windows*. Top-level windows interact directly with the X window manager. They have a full title bar along with other window manager decorations. You can manipulate them individually once they appear on the display. For details on the top-level windows, refer to the online help.

Interior windows behave differently than primary windows. SAS/ASSIST software is an application with interior windows. Interior windows are contained within *container windows*, which may or may not be primary windows. Display 3.1 on page 38 shows an interior window in SAS/ASSIST software.

Display 3.1 Sample Interior Window



The SAS System provides some degree of window management for interior windows. Specifically, interior windows have the following sizing and movement capabilities:

- You can move interior windows by clicking the left mouse button on the interior window title bar and dragging the window to the desired location. If the destination of the interior window is outside the bounds of the container window, the container window changes according to the value of the **`SAS.awsResizePolicy`** resource. (The space within the container window is the application workspace, which is described in "Workspace and Gravity in a SAS Session" on page 37.) See "Using X Resources to Customize the Motif Interface" on page 38 for more information.
- Interior windows cannot be iconified individually. Clicking on the container window icon button iconifies the container window and its interior windows.
- A *push-to-back button* (the small overlapping squares in the upper right corner) is also available with interior windows. However, you cannot push an active window behind an inactive window.
- You can raise a window by clicking on its title bar.

Using X Resources to Customize the Motif Interface

X clients usually have characteristics that can be customized; these properties are known as *X resources*. For example, X resources can be used to define a font, a background color, or a window size. The resources for an application, such as SAS, are placed in a *resource database*.

The SAS System functions correctly without any modifications to the resource database. However, you may want to change the default behavior or appearance of the interface. There are several ways to specify your customizations. Some methods modify all SAS sessions displayed on a particular X server. Some methods affect all SAS sessions run on a particular host. Other methods affect only a single SAS session.

The following sections describe how to customize the resource database. If you need more information on X Window System clients and X resources, refer to the documentation provided by your vendor.

Specifying X Resources

A resource specification has the following format:

resource-string: *value*

The *resource string* usually contains two identifiers and a separator. The first identifier is the client or application name (**SAS**), the separator is a period (.) or asterisk (*) character, and the second identifer is the name of the specific resource. The *value* given may be a Boolean value (**True** or **False**), a number, or a character string, depending on the resource type.

The application name and resource name can both specify an *instance value* or a *class value*. A specification for a class applies to a larger scope than a single instance.

The following are sample resource specifications:

```
SAS.startSessionManager: True
SAS.maxWindowHeight: 100
SAS.awsResizePolicy: grow
```

Refer to your X Window System documentation for more information on resource specifications.

Customizing SAS Resources

The following list describes the methods that you can use to customize X resources.

- Use the Font dialog box, the Preferences dialog box, or the Resource Helper to customize your SAS session. All of these tools write X resource definitions out to a location that SAS will read the next time you start a SAS session. See "Setting X Resources through the Preferences Dialog Box" on page 40, "Setting X Resources with the Resource Helper" on page 44, and "Customizing Fonts" on page 63 for more information on these tools.
- Specify session-specific resources by using the -xrm option on the command line for each invocation of the SAS System. For example, the following command specifies that SAS will not display the Confirm dialog box when you exit your SAS session:

  ```
  sas -xrm 'SAS.confirmSASExit: False'
  ```

  You can specify the -xrm option as many times as needed. You must specify the -xrm option for each resource.

  Note: If you normally invoke the SAS System with a shell script, you should protect the quote characters from the shell with the backslash (\) character:

  ```
  sasscript -xrm \'SAS.confirmSASExit: False\'
  ```

  △
- Add resource definitions to a file in your home directory. If you place resources in a file that X Toolkit normally searches for when applications are invoked, these resources will be loaded when you invoke SAS. For information about where the X Toolkit searches for resources, refer to the documentation for the X Windows System.

  You can also add resources to the resource database after SAS has initialized by running xrdb. For example, the following command merges the definitions in the **myresources** file into the resource database:

  ```
  xrdb -merge myresources
  ```

- Create a subdirectory for storing resource definitions. (This subdirectory is usually named **app-defaults**.) Set the environment variable XAPPLRESDIR to the pathname of this subdirectory. Specify the definition for the XAPPLRESDIR environment variable in the initialization file for your shell, for example, the **$HOME/.login**, **$HOME/.cshrc** or **$HOME/.profile** files, to ensure that the XAPPLRESDIR variable is defined for each shell that is started.

Create a file called **SAS** in the subdirectory identified by XAPPLRESDIR. Include your resource definitions in this file.

- If you want the customized resource definitions to be used for all users on a particular host, create a file called **SAS** to contain your resource definitions, and store this file in the system **app-defaults** directory.

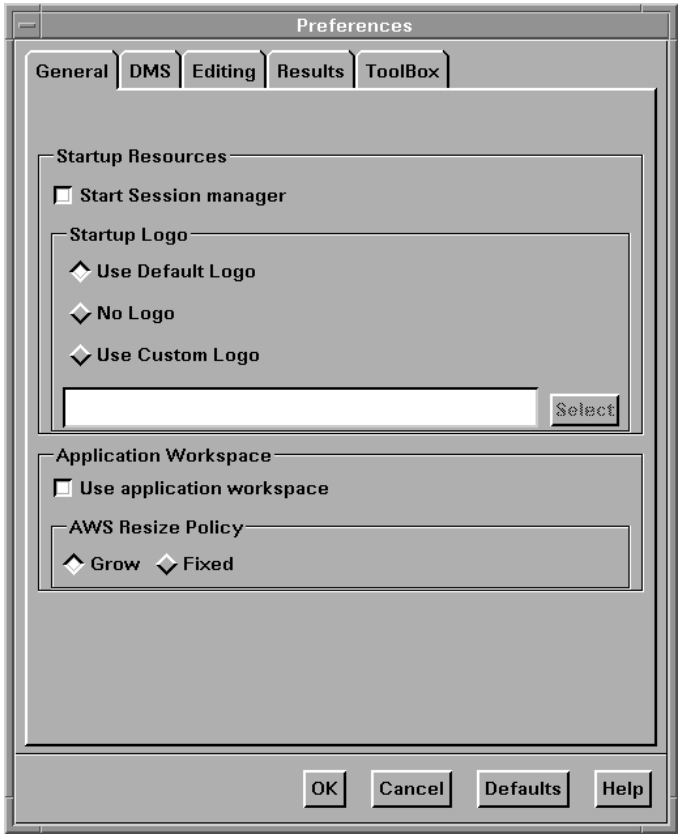
For more information on X resources, refer to the X Window System documentation supplied by your vendor or to other books on the X Window System.

Setting X Resources through the Preferences Dialog Box

The Preferences dialog box allows you to control the settings of certain X resources. Changes made through the Preferences dialog box (with the exception of those resources on the General tab) become effective immediately, and the settings are saved in SASUSERPREFS file in your SASUSER directory. You can invoke the Preferences dialog box by issuing the DLGPREF command or by selecting

Tools ► Options ► Preferences

Display 3.2 Preferences Dialog Box



General Options

Start Session manager
specifies whether you want the session manager to be started automatically when you start your SAS session. If you want to use your host editor in your SAS session, the session manager must be running. The session manager allows you to interrupt or terminate your SAS session and minimize and restore all of the windows in a SAS session. See "Using the SAS Session Manager (motifxsassm)" on page 33and "Configuring the SAS System for Host Editor Support" on page 78 for more information. This check box sets the **SAS.startSessionManager** resource.

Startup Logo
specifies whether you want SAS to display an XPM file while your SAS session is being initialized and, if so, which file.

If you select **Use Default Logo**, SAS uses the default file for your site. If you select **No Logo**, then no file is displayed. If you select **Use Custom Logo**, then you can either enter the XPM filename directly in the text field or press Select to open the File Selection dialog box. This check box sets the **SAS.startupLogo** resource.

Use application workspace
confines all windows displayed by an application to a single Application Work Space. This check box sets the **SAS.noAWS** resource. You must exit and reopen the windows for changes to this resource to take effect.

AWS Resize Policy
controls the policy for resizing AWS windows as interior windows are added and removed. (See "Workspace and Gravity in a SAS Session" on page 37 and "Window Types" on page 37.)

Grow
The AWS window will attempt to grow any time an interior window is grown or moved (to make all of its interior windows visible), but it will not shrink to remove unused areas.

Fixed
The AWS window will attempt to size itself to the size of the first interior window and will not attempt any further size changes.

This check box sets the **SAS.awsResizePolicy** resource.

DMS Options

Use menu access keys
activates pull-down menu mnemonics. When mnemonics are turned on, you can select menu items by typing the single, underlined letter in the item. This check box sets the **SAS.usePmenuMnemonics** resource.

Confirm exit
displays the Exit dialog box when you exit your SAS session. This check box sets the **SAS.confirmSASExit** resource.

Save settings on exit
tells SAS to issue the WSAVE ALL command when you exit your SAS session. This command saves the global settings, such as window color and window position, that are in effect for all windows that are currently open.These settings are saved in your SASUSER.PROFILE catalog. This check box sets the **SAS.wsaveAllExit** resource.

Use host printing
turns on host printing. This check box sets the **SAS.hasXPrinter** resource.

Backup Documents
allows you to specify whether you want SAS to automatically save (at the interval specified by the **SAS.autoSaveInterval** resource) the documents that you currently have open. This check box sets the **SAS.autoSaveOn** resource.

Help & Documentation Browser -- Netscape Path
specifies the path name for the web browser that you want to use to view online help and the CD-ROM documentation. This field sets the **SAS.helpBrowser** resource.

Editing Options

Default paste buffer
defines an alias for the default SAS buffer. The following list describes the paste buffer alias names and the X buffer with which each name is associated.

XPRIMARY
X primary selection (**PRIMARY**)

XSCNDARY
X secondary selection (**SECONDARY**)

XCLIPBRD
X clipboard (**CLIPBOARD**)

XTERM
exchange protocol used by the xterm client

XCUT*n*
X cut buffer where *n* is between 0 and 7, inclusive

This check box sets the **SAS.defaultPasteBuffer** resource. See “Controlling Pull-down Menus” on page 72 for more information about cut-and-paste butters.

Automatically store selection
generates a STORE command every time you mark a region of text with the mouse. This check box sets the **SAS.markPasteBuffer** resource.

Cursor
controls the editing mode in SAS text editor windows. The **Insert** and **Overtype** check boxes set the **SAS.insertModeOn** resource to **True** and **False**, respectively.

Results Options

The items on the Results tab affect only output that is produced through the Output Delivery System (ODS). For a complete description of ODS, refer to *The Complete Guide to the SAS Output Delivery System*.

Create Listing
opens the ODS Listing destination, which produces monospace output. Selecting this check box is equivalent to entering the ODS LISTING SELECT ALL statement. This check box sets the **SAS.resultsListing** resource.

Create HTML
opens the ODS HTML destination, which produces output that is formatted in Hypertext Markup Language. Selecting this check box is equivalent to entering the ODS HTML SELECT ALL statement. This check box sets the **SAS.resultsHTML** resource.

Folder
specifies a destination for HTML files. Specifying a directory in this field is equivalent to specifying a directory with the PATH option in the ODS HTML statement. This field sets the **SAS.resultsTmpDir** resource.

Use WORK Folder
tells the ODS to send all HTML files to your WORK directory. Selecting this check box is equivalent to specifying the pathname of your WORK directory with the PATH option in the ODS HTML statement. This check box sets the **SAS.resultsUseWork** resource.

Style
specifies the style definition to use for HTML output. The style definition controls such aspects as color, font name, and font size. Specifying a style in this field is equivalent to specifying a style with the STYLE option in the ODS HTML statement. You can specify any style that is defined in the \ODS\PREFERENCES\STYLES key in the SAS Registry. You can open the SAS Registry by issuing the REGEDIT command or by selecting

Solutions ▶ Accessories ▶ Registry Editor

This field sets the **SAS.resultsHTMLStyle** resource.

View results as they are generated
tells SAS to automatically display results files when they are generated. If you select this check box, make sure that **Password protect HTML file browsing** is deselected. This check box sets the **SAS.resultsAutoNavigate** resource.

Password protect HTML file browsing
tells SAS to prompt you for your password before sending HTML files to your browser. If you select this check box, make sure that **View results as they are generated** is deselected. This check box sets the **SAS.htmlUsePassword** resource.

Toolbox and Command Window Options

The items in the ToolBox Window area of the Preferences dialog box affect both the toolbox and the command window.

Display tools window
determines whether to display the default toolbox. This check box sets the **SAS.defaultToolBox** resource.

Display command window
determines whether to display the command window. This check box sets the **SAS.defaultCommandWindow** resource.

Auto Complete Commands
specifies whether SAS automatically fills in the remaining letters of a command as you type a command in the command window that begins with the same letter as a command that you have entered previously. If both this check box and **Save Commands** are selected, then SAS can automatically fill in commands that were entered in previous sessions. This check box sets the **SAS.autoComplete** resource.

Save Commands
specifies whether SAS saves the commands that you enter in the command window and how many commands are saved. You can specify a number from 0 to 50. If you specify 0, no commands will be saved. If you specify 1 or more, that number of commands is saved in the file **commands.hist** in your SASUSER

directory. If this check box is selected, then SAS will be able to automatically fill in (see **Auto Complete Commands**) commands that were entered in previous sessions. This field sets the **SAS.commandsSaved** resource.

Combine windows
combines the toolbox and command window into one window. The toolbox and command window are combined by default. This check box sets the **SAS.useCommandToolBoxCombo** resource.

Use arrow decorations
adds arrows to both end of the combined toolbox/command window. This check box sets the **SAS.useShowHideDecorations** resource.

Always on top
keeps the toolbox or the combined toolbox/command window on top of the window stack. This check box is selected by default, which may cause problems with window managers and other applications that want to be on top of the window stack. If you have such a situation, turn off this feature. This check box sets the **SAS.toolBoxAlwaysOnTop** resource.

Toolbox Persistent
specifies whether the toolbox that is associated with the Program Editor stays open when you close the Program Editor. By default, the Program Editor toolbox stays open whenever you close the Program Editor. If you deselect this check box, then the toolbox will close if you close the Program Editor. This check box sets the **SAS.isToolBoxPersistent** resource.

The items in the Tools area affect the individual tools in the toolbox.

Use large tools
controls whether tool icons are displayed as 24x24 or 48x48 pixels. The default is 24x24. This check box sets the **SAS.useLargeToolBox** resource.

Use tip text
specifies whether tool tip text is displayed when you position your cursor over a tool in the toolbox. Some window managers may place the toolbox tip behind the toolbox. If this happens in your environment, deselect this check box. This check box sets the **SAS.useToolBoxTips** resource.

delay
controls the delay in milliseconds before popping up the toolbox tip. This check box sets the **SAS.toolBoxTipDelay** resource. You can enter a value directly into the field or use the arrows to the right of the field to change the value.

Setting X Resources with the Resource Helper

With Resource Helper, you can customize the key definitions and colors of SAS's interactive interface. Resource Helper creates SAS resource definitions and stores them in a location where the Resource Manager can find them. See "How Resource Helper Searches For X Resources" on page 48 for a list of the locations that Resource Helper searches for resource definitions. Resource settings that are saved with Resource Helper will take effect the next time you start a SAS session.

You can start Resource Helper from within a SAS session or from your shell prompt.

Starting Resource Helper from a SAS Session

Start the SAS Resource Helper from a SAS window by entering

```
reshelper
```

on the command line.

Display 3.3 Main Window for Resource Helper

Starting Resource Helper from a Shell Prompt

Resource Helper is installed into the **/utilities/bin** subdirectory in the directory where SAS is installed (**!SASROOT**). The name of the executable module is **reshelper**. For example, if the SAS System is installed in **/usr/local/sas8**, you start Resource Helper by typing the following command:

```
/usr/local/sas8/utilities/bin/reshelper &
```

If you run Resource Helper from a shell script, you might need to set the XKEYSYMDB and XLOCALEDIR environment variables in order for Resource Helper to work. XKEYSYMDB should contain the path to **/X11/resource_files/XKeysymDB** in the **!SASROOT** directory, and XLOCALEDIR should contain the path to **/X11/resource_files/locale/** in that same directory. If SAS is installed in **/usr/local/sas8**, then in the Bourne or Korn shells, you can assign these environment variables as follows:

```
export XKEYSYMDB=\
/usr/local/sas8/X11/resource_files/XKeysymDB
export XLOCALEDIR=\
/usr/local/sas8/X11/resource_files/locale/
```

In the C shell, you can enter

```
setenv XKEYSYMDB \
/usr/local/sas8/X11/resource_files/XKeysymDB
setenv XLOCALEDIR \
/usr/local/sas8/X11/resource_files/locale/
```

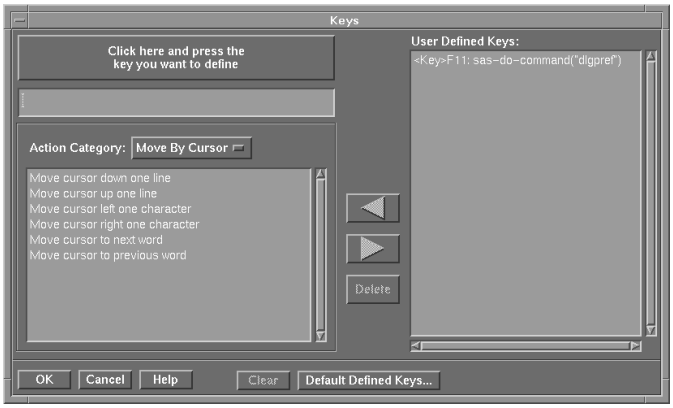
Resource Helper accepts the standard X Toolkit options. Refer to the documentation provided by your X Window System vendor for a list of these options.

Changing Keys with Resource Helper

Note: In most cases, using Resource Helper is much easier and faster than defining the resources yourself. However, the X Window System searches for resources in several places, so it is possible for Resource Helper to pick up the wrong key symbol for the key you are trying to define. If you get unexpected results while using Resource Helper, you might need to define your key resources yourself. See "Defining Key Translations" on page 56 for more information. △

Start the Resource Helper (see "Setting X Resources with the Resource Helper" on page 44) and select the Keys icon.

Display 3.4 Keys Window for Resource Helper



Key definitions are divided into several **Action Categories**:

- **Move By Cursor**
- **Move By Field**
- **Edit**
- **Miscellaneous**
- **All Actions**

To define a key, follow these steps:

1. Select Click here and press the keys you want to define.
2. Press the key or combination of keys that you want to assign an action to. For example, press **F12**. If a default SAS translation has already been assigned to the key combination, Resource Helper displays the default translation.
3. Select the action category menu button to open a list of action categories. Select the action category that you want. For example, if you want to define a key to delete the current field, select **Edit** as your **Action Category**. Resource Helper will display a list of actions in that category.
4. Select an action from the list. For example, **delete current field**. Resource Helper can assign only one action to a translation. If the action that you select requires an argument (such as **sas-action-routine**), Resource Helper prompts you for the argument.

   Resource Helper displays the key combination and its new definition:

   ```
   None<Key>F12: sas-delete()
   ```

   Note: If you select the sas-function-key action routine, then the key definition is automatically displayed in the KEYS window. If you choose another action routine and if you want the definition to appear in the KEYS window, you will need to define a window label for the key. See "Modifying the keysWindowLabels Resource" on page 59 for information on defining labels in the KEYS window. △
5. Select the right arrow to add this key translation to the list of **User-Defined Keys**.
6. Select OK to exit the Keys window after you have finished defining key translations.
7. To save your translations permanently, from the Resource Helper pull-down menus, select

   File ▶ Save Resources

To modify a key definition that is already in the **User-Defined Keys** list, select the definition, select the left arrow to remove the definition from the list, and edit the definition.

To delete a definition from **User-Defined Keys**, select it and select **Delete**.

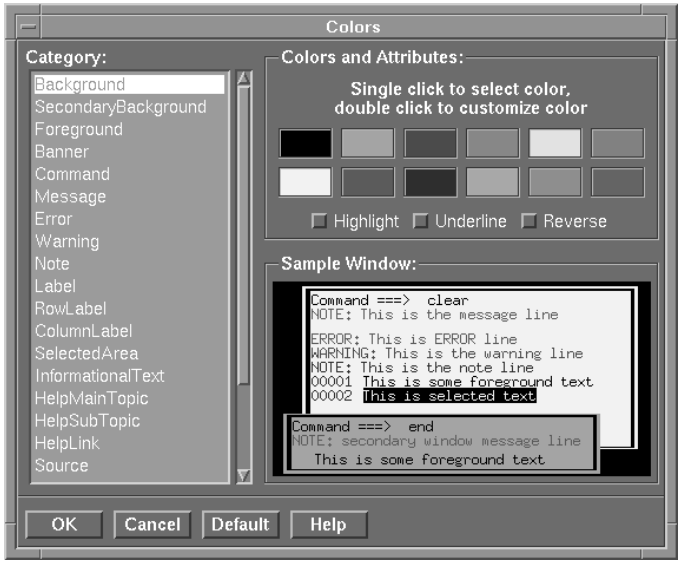
Clear clears the key definition edit window.

Default Defined Keys... displays the default key definitions for your system.

Changing Colors with Resource Helper

Start Resource Helper and select the Colors icon.

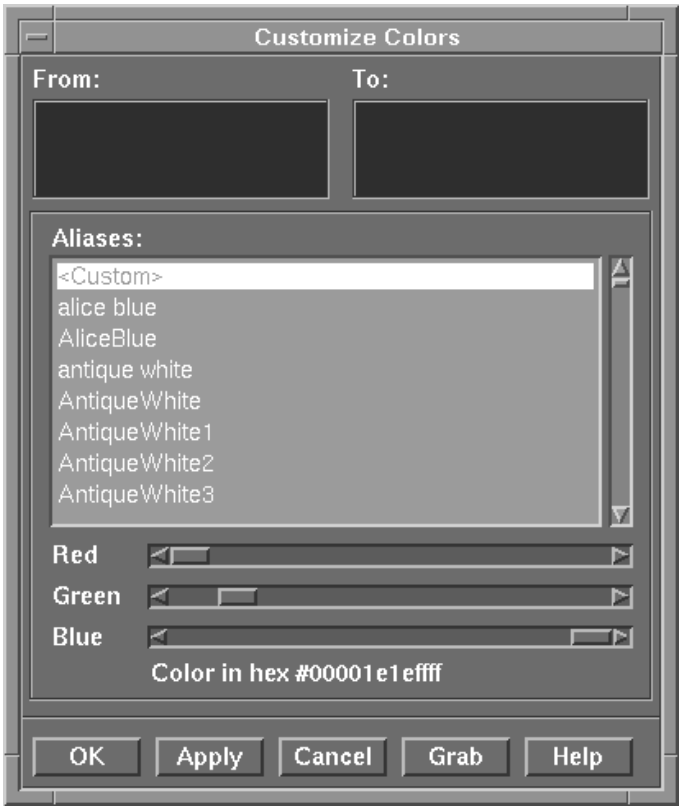
Display 3.5 Colors Window for Resource Helper



You can modify the color of part of a SAS window as follows:

1. Select a category from the **Category** window.
2. Single-click on a color and/or attribute in the Colors and Attributes window, or double-click on a color to open the Customize Color window, shown in Display 3.6 on page 48.

Display 3.6 Customize Colors Window for Resource Helper



You can customize a color by

- selecting a new **Alias**
- moving the **Red, Blue,** or **Green** sliders
- selecting Grab and clicking on a color anywhere on your screen.

The result is displayed in the **Sample Window**. The hex value of the color is displayed at the bottom of the window.

For example, double-click on **Red** in the **Colors** window. **Change Red from:** displays the **Red** currently used by SAS windowing environment. Click on **Aquamarine** under **Aliases** and observe the change in the **To:** display. Move the **Red**, **Green**, and **Blue** sliders with your mouse button and note the changes in the color of the **To:** display. Click on **Apply** and note the difference in the color displayed as **Red** in the **Colors** window. Select OK to save your changes.

You can also change the attributes of some categories of SAS windows. **Attributes** allows you to select **Highlight**, **Underline**, or **Reverse**.

Select Defaults to restore your color settings to their default values. Select OK to exit the Colors window after you have finished defining your color settings. To save your color settings permanently, from the Resource Helper pull-down menus, select

File ▶ Save Resources

How Resource Helper Searches For X Resources

The following list describes the locations where the Resource Helper searches for resource definitions and the order in which it searches these locations.

1 Resource Helper loads the resources in the file pointed to by the XENVIRONMENT environment variable. If XENVIRONMENT is not set, Resource Helper loads the resources in the **~/.Xdefaults-***hostname* file, where *hostname* is the name of the machine on which Resource Helper is running.

2 Resource Helper loads the resources defined in the RESOURCE_MANAGER property. If the RESOURCE_MANAGER property is the first location in which Resource Helper finds resources, the RESOURCE_MANAGER property will override any resources that you generate with Resource Helper.

   To determine if any resources have been defined in your RESOURCE_MANAGER property, issue the following command:

   ```
   xrdb -q | more
   ```

   If no listing is returned, the RESOURCE_MANAGER property does not exist. In this case, Resource Helper loads the resources defined in the **~/.Xdefaults** file.

3 Resource Helper loads the resources in the file pointed to by the XUSERFILESEARCHPATH environment variable. (On RS6000 systems, you must set the XUSERFILESEARCHPATH variable.)

   You can use **%N** to substitute an application class name for a file when specifying the XUSERFILESEARCHPATH environment variable. For example, to point to **/usr/local/resources** as the location of all the resources for any application, issue the following command in the Bourne or Korn shells:

   ```
   export XUSERFILESEARCHPATH=\
   /usr/local/resources/%N
   ```

   In the C shell, the command is

   ```
   setenv XUSERFILESEARCHPATH \
   /usr/local/resources/%N
   ```

   As a result, when SAS is invoked, the file pointed to by XUSERFILESEARCHPATH is

   ```
   /usr/local/resources/SAS
   ```

   SAS is the application class name for SAS.

4 Resource Helper loads the resources in the file specified by the the XAPPLRESDIR environment variable. The application's class name is appended to the XAPPLRESDIR environment variable and the resulting string is used to search for resources. For example, if you issue the following command in the Bourne or Korn shells:

   ```
   export XAPPLRESDIR=/usr/local/app-defaults
   ```

   at the next invocation of SAS, the application's class name is appended to the path:

   ```
   /usr/local/app-defaults/SAS
   ```

   In the C shell, the command is

   ```
   setenv XAPPLRESDIR /usr/local/app-defaults
   ```

5 Resource Helper loads the resources in the file named **~/SAS**.

6 Resource Helper loads the resources in the file or substitution specified by the XFILESEARCHPATH environment variable.

   Note: To determine if an environment variable has been set, you can issue the following command:

   ```
   env|grep <environment_variable>
   ```

△

7 Resource Helper loads the resources defined in **/usr/lib/X11/app-defaults**. Resource Helper does not need to have write access to this file, but it must be able to read the file and add the SAS resources to a writable resource file. Resource Helper does not generate a warning message if the file is not present or if it cannot read the file.

8 Resource Helper loads the fallback resources that are defined in the SAS code.

Except for the **/usr/lib/X11/app-defaults** file, Resource Helper trys to write the new resources to the same directory and file where it first found SAS resources. This location must be a writable file in a writable directory. If Resource Helper cannot write to the file, the SAS resources in that file will remain in effect and any new or modified resources generated by Resource Helper will not take effect. If this happens, Resource Helper displays an error dialog box that contains the file or directory and suggests a way to fix the problem.

Customizing Toolboxes and Toolsets

You can customize toolboxes

- through the Preferences dialog box. The Preferences dialog box allows you to customize the appearance and behavior of toolboxes. See "Setting X Resources through the Preferences Dialog Box" on page 40 and "Toolbox and Command Window Options" on page 43 for information on using the Preferences dialog box.
- by specifying SAS resources in your resource file. "Specifying Toolbox Resources" on page 50 describes the SAS resources that affect toolboxes.
- through the Tool Editor. The Tool Editor allows you to customize the individual tools in a toolbox. See "Using the Tool Editor" on page 51 for more information.

The Tool Editor also allows you to create custom toolsets for your SAS applications. A *toolset* is a set of predefined tools that is associated with an application. Toolsets make it easier for individual users to customize their application toolboxes. If you create a toolset for an application, users can press the Actions button in the Tool Editor and choose the tools that they want to appear in their toolboxes. Users do not have to define the icons, commands, tip text, and IDs for those tools. For example, you can define a default toolbox for your application that includes tools for opening files, cutting, copying, and pasting text, and saving files. You can define a toolset that includes those tools and tools for opening the Preferences dialog, opening the Replace dialog, and entering the RECALL command. These additional tools will not appear in the users' toolbox unless a user adds them to their toolbox with the Tool Editor. See "Changing an Existing Tool" on page 52 and "Creating or Customizing an Application- or Window-Specific Toolset" on page 55 for more information.

Specifying Toolbox Resources

You can control the behavior of toolboxes with the following SAS resources:

SAS.autoComplete: True | False
specifies whether SAS automatically fills in the remaining letters of a command as you type a command in the command window that begins with the same letter as a command that you have entered previously. The default value is True.

SAS.commandsSaved : close up | *n*
specifies whether SAS saves the commands that you enter in the command window and how many commands are saved. You can specify a number from 0 to

50. If you specify 0, no commands will be saved. If you specify 1 or more, that number of commands is saved in the file **commands.hist** in your SASUSER directory. If you specify 1 or more for this resource and **SAS.autoComplete** is True, then SAS will be able to automatically fill in commands that were entered in previous sessions. The default value is 25.

SAS.defaultToolBox: True | False
controls opening the default toolbox when the SAS System is invoked. The default is True.

SAS.isToolBoxPersistent: True | False
controls whether the toolbox that is associated with the Program Editor stays open when you close the Program Editor. The default value is True.

SAS.toolBoxAlwaysOnTop: True | False
controls whether the toolbox is always on top of the window stack. The default value of True may cause problems with window managers that are not Motif interface window managers or other applications that want to be on top of the window stack. If you have such a situation, set this resource to False.

SAS.toolBoxTipDelay: *delay-in-milliseconds*
sets the delay in milliseconds before displaying the toolbox tip. The default is 750.

SAS.useCommandToolBoxCombo: True | False
controls whether the command window and toolbox are joined or separated. The **SAS.defaultToolBox** and **SAS.defaultCommandWindow** resources control whether the toolbox and command window are displayed. If both are displayed, this resource controls whether they are joined or separated. The default value is True.

SAS.useLargeToolBox: True|False
controls whether tool icons in the toolbox are displayed as 24x24 pixels or 48x48 pixels. The default is False (24x24 pixels).

SAS.useShowHideDecorations: True | False
controls whether the combined command/toolbox window has arrows at the left and right. You can use these arrows to hide or show portions of the window as they are needed. The default value is False.

SAS.useToolBoxTips: True | False
determines if toolbox tip text is displayed. Some window managers may place the toolbox tip behind the toolbox. If this happens in your environment, set this resource to False. The default is True.

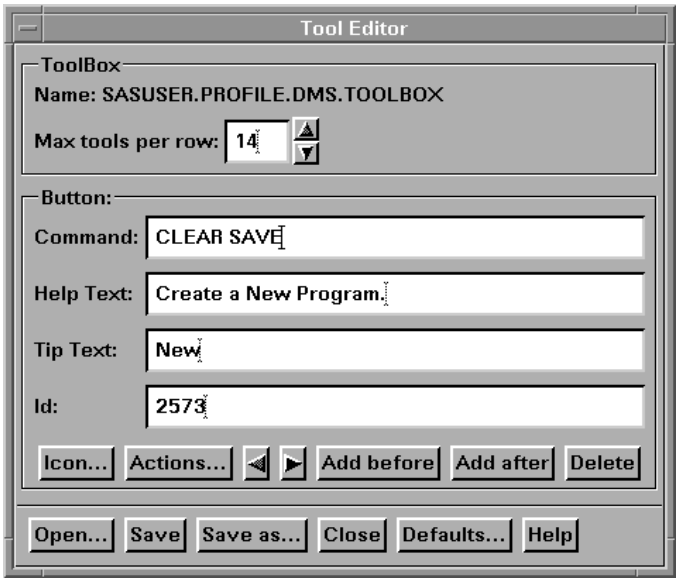
Using the Tool Editor

You can change the appearance and contents of a toolbox using the Tool Editor. To invoke the Tool Editor, (shown in Display 3.7 on page 52) select

Tools ► Options ► Edit Toolbox...

Alternatively, you can issue the TOOLEDIT command as described in “TOOLEDIT” on page 180 or open the window menu for the command window/toolbox and select **Edit toolbox...**.

Display 3.7 Tool Editor Dialog Box



By default, the Tool Editor edits the current toolbox. To edit a different toolbox, select Open... and specify the libref, catalog, and entry name for the toolbox you want to edit.

Once you invoke the Tool Editor, the toolbox goes into preview mode. In preview mode, clicking on a tool icon makes that icon the current icon and displays its associated commands in the **Command** field. The current icon always appears pressed.

Changing the Appearance of the Entire Toolbox

The items in the area of the Tool Editor labeled "ToolBox" affect the entire toolbox:

Name:
displays the catalog entry that you are editing. The default toolbox is named SASUSER.PROFILE.DMS.TOOLBOX.

Max tools per row:
specifies how the icons in the toolbox are arranged. The default value creates a horizontal toolbox. One tool per row creates a vertical toolbox.

Changing an Existing Tool

When you open the Tool Editor, the first icon is the current icon, and information about the current icon appears in the Button area of the dialog box. To change an existing tool, you can select a tool from the toolset displayed by the Actions button or you can modify the fields individually.

Note: The Actions button will display a toolset only if a toolset is associated with (has the same entry name as) the toolbox that you are editing. See "Saving Changes to the Toolbox or Toolset" on page 54 for more information. △

To use Actions, select the tool that you want to change, and then select Actions. The Tool Editor displays the toolset associated with toolbox. If you select a tool from this toolset, the Tool Editor enters the appropriate information into the button fields for you.

To modify the fields individually:

1 Select the icon you want to change.

2 Click in and change the Button fields as appropriate.

Command:
specifies the command or commands that you want executed when you click on the icon. You can use any windowing environment command available under UNIX. For information on portable commands, see *SAS Language Reference: Dictionary*. Separate commands with a semi-colon (;). For example, you could create an icon to open the Change Working Directory dialog box by using the DLGCDIR command.

Help Text:
is used for applications that are designed to be run under Windows or OS/2. The help text is displayed in the AWS status line on Windows and OS/2 when a toolbox is ported to and loaded on those platforms.

Tip Text:
specifies the text that is displayed when you position the cursor over the icon.

Id:
is useful if you are creating toolboxes for SAS/AF applications. The ID is the identifier of the corresponding pull-down menu item in the application. This number is the value assigned to the item in the ID option of the ITEM statement in PROC PMENU. If you specify an ID, then the application can set the state of the PMENU item to match the state of the tool in the toolbox, and it can make the PMENU item active or inactive to match whether the PMENU item is active or inactive. If you do not specify an ID, the ID defaults to 0.

3 Change the icon if necessary.

   a Select Icon... or double-click on an icon in the preview toolbox. The Tool Editor opens the Select A Pixmap dialog box, which displays the icons provided with the SAS System. These icons are divided into several categories such as SAS windows, data, analysis, numbers and symbols, files, folders, and reports, and so on. To change categories, select the arrow to the right of the **Icon Category** field and select a new category.
   b Select the icon you want to use and then select OK.

4 Save your changes as described in "Saving Changes to the Toolbox or Toolset" on page 54.

Adding Tools to the Toolbox

To add a tool to the toolbox:

1 Select the icon next to where you want to add the new tool.

2 Select Add before or Add after. The Tool Editor adds a new icon to the toolbox and clears the Button fields.

3 Enter the appropriate information in the Button fields as described in "Changing an Existing Tool" on page 52.

4 Change the icon, if necessary, as described in "Changing an Existing Tool" on page 52.

5 Save your changes as described in "Saving Changes to the Toolbox or Toolset" on page 54.

Changing the Order of the Tools in the Toolbox

To change the position of a tool in the toolbox, select the tool icon, and then click on the left or right arrows to move the tool.

Deleting Tools from the Toolbox

To delete a tool from the toolbox:

1. Select the tool you want to delete.
2. Select Delete.
3. Save your changes as described in "Saving Changes to the Toolbox or Toolset" on page 54.

Returning to the Default Settings

To return all tools in the current Toolbox to their default settings, select Defaults... The Tool Editor asks you to verify your request. Select Yes, No, or Cancel.

Saving Changes to the Toolbox or Toolset

You can save the changes to the catalog entry shown in the **Name** field or create a new toolbox with a different name.

If you are customizing a window- or application-specific toolbox or toolset for your own personal use, you should save the customized toolbox or toolset in your SASUSER.PROFILE catalog using the same entry name as the PMENU entry for the window or application. The SAS System searches for toolboxes and toolsets first in SASUSER.PROFILE, and then in the application catalog.

If you are a SAS/AF application developer or site administrator and you are editing a window- or application-specific toolbox that you want to be accessible to all users, you must save the TOOLBOX entry with the same library, catalog, and entry name as the PMENU entry for the window or application. To associate a toolset with a specific toolbox, save to TOOLSET entry with the same library, catalog, and entry name as the TOOLBOX entry. You will need write permissions to the appropriate location. For example, to store a customized toolbox for the graphics editor, the site administrator needs to store the toolbox in SASHELP.GI.GEDIT.TOOLBOX.

Save saves the toolbox information to the catalog entry shown in the **Name** field. Save As... prompts you to enter a different libref, catalog, and entry name. You can also choose to save the toolbox as a toolset. If you save the toolbox as a toolset, the entry type will be TOOLSET; otherwise, the entry type is always TOOLBOX. (Saving a set of tools as a TOOLSET does not change your TOOLBOX entry. See "Customizing Toolboxes and Toolsets" on page 50 and "Creating or Customizing an Application- or Window-Specific Toolset" on page 55 for information about toolsets.)

If you select Close or Open... without first saving your changes, the Tool Editor prompts to save the changes to the current toolbox or toolset before continuing.

After you save the toolbox or toolset, the Tool Editor remains open for additional editing, and the **Name** field changes to the name of the new entry (if you entered a new name).

Creating a New Toolbox

To create an entirely new toolbox, you can:

- edit an existing toolbox using the Tool Editor and save it using Save as... as described in "Saving Changes to the Toolbox or Toolset" on page 54.
- open the SASUSER.PROFILE catalog in the Explorer and add a new toolbox by selecting

  File ▶ New... ▶ Toolbox

Creating or Customizing an Application- or Window-Specific Toolbox

If you are an application developer and want to create or edit an existing application toolbox, you must:

1. Delete any existing TOOLBOX entry in your SASUSER.PROFILE for the window or application that you want to customize. Deleting the copy of the toolbox in your SASUSER.PROFILE allows you to pick up a copy of the toolbox supplied with the SAS System when you invoke the Tool Editor.
2. Create or edit the application toolbox as described in "Creating a New Toolbox" on page 54 or "Using the Tool Editor" on page 51.
3. Save the edited toolbox as described in "Saving Changes to the Toolbox or Toolset" on page 54.
4. Inform your users that you have changed the window or application toolbox. If they want to use the new toolbox, they must delete the corresponding TOOLBOX entry from their SASUSER.PROFILE. The new toolbox will then be automatically loaded when the window or application is invoked. If a user does not delete the corresponding TOOLBOX entry from their SASUSER.PROFILE, that copy of the toolbox will be loaded instead of the new toolbox.

The TOOLLOAD and TOOLCLOSE commands are most useful when you are developing SAS/AF applications. You can use the EXECMDI routine with these commands to enable your application to open and close the toolbox and to give users of your applications access to several toolboxes during the course of their work. See *SAS Component Language: Reference* for a description of the EXECMDI routine.

Creating or Customizing an Application- or Window-Specific Toolset

You define application- or window-specific toolsets in the same way that you create an application- or window-specific toolbox. There are only two differences:

- To create a new toolset, start by defining a toolbox as described in "Creating a New Toolbox" on page 54.
- After you have defined the toolbox, save it as a TOOLSET entry, not as a TOOLBOX entry.

Note: If you are an application developer, make sure that you delete any existing TOOLSET entry for your application as described in "Creating or Customizing an Application- or Window-Specific Toolbox" on page 55 before you modify your application's toolset. △

Customizing Key Definitions

There are four ways to customize your key definitions:

- through the KEYS window. To open the KEYS window, issue the **keys** command or select

  Tools ▶ Options ▶ Keys

  If you change any key definitions through the KEYS window for the primary SAS windowing environment windows, the definitions are stored in the catalog SASUSER.PROFILE in the entry DMKEYS.KEYS. Key definitions for other SAS windows are stored in catalog entries named BUILD.KEYS, FSEDIT.KEYS, and so on.

Refer to the online help for more information on the **keys** command and the KEYS window.

- with the KEYDEF command. The KEYDEF command allows you to redefine individual function keys:

  keydef *keyname <command|~text-string>*

  For example, if you specify **keydef F8 dlgpref**, then the F8 key will open the Preferences dialog box.

  Refer to the online help for more information about the KEYDEF command.
- through the Resource Helper (reshelper). Resource Helper generates SAS resource specifications based on keys and functions that you select. You can use Resource Helper to change the function of any key listed in the KEYS window. See "Setting X Resources with the Resource Helper" on page 44 and "Changing Keys with Resource Helper" on page 45 for more information on Resource Helper.

  In most cases, Resource Helper is much easier and faster than defining the resources yourself. However, because the X Window System searches for resources in several places, it is possible for Resource Helper to pick up the wrong key symbol for the key you are trying to define. Also, unless the action routine that you assign to your keys is the **sas-function-key** routine, then Resource Helper does not provide a way to change the key labels in the KEYS window. In both of these cases, you will need to define your key resources yourself.
- by defining the **SAS.keyboardTranslations** and **SAS.keysWindowLabels** resources in your resources file as described in "Defining Key Translations" on page 56.

You can define most of the keys on your keyboard. However, a few keys have dedicated functions that are associated with them. For example, the mouse buttons are dedicated to the cursor and cut-and-paste operations and are not available for user customization.

Defining Key Translations

Key customization for the X Window System consists of defining a key sequence and an action to be executed when that key sequence is typed on the keyboard. This is known as *binding* keys to *actions*; together they are referred to as a *translation*.

The **SAS.keyboardTranslations** resource specifies the set of key bindings that the SAS System uses in all SAS windows. The default value for the **SAS.keyboardTranslations** resource is determined at run time based on the vendor identification string reported by the X server that you are using as the display. These defaults are listed in the files contained in **!sasroot/X11/resource_files**. To modify the default bindings supplied by the SAS System, you must modify the keyboardTranslations resource.

Note: The X Toolkit Intrinsics translations specified in this resource apply to both the user area and the command line of all SAS windows that are affected by this resource. This resource does not affect windows that are controlled by Motif interface resources, such as the Command window, the Open or Import dialog boxes, and some other pull-down menu dialog boxes. △

To create a key definition, follow these steps:

1. Determine the keysyms for the keys that you want to define. *Keysyms* are the symbols recognized by the X Window System for each key on a keyboard. See "Determining Keysyms" on page 57 for more information.

2 Modify/add the **SAS.keyboardTranslations** resource in your resource file to include the definitions of the keys that you want to define. Use a keyboard action routine to define which action you want the key to perform. The definition in the right column in the KEYS window will no longer control the function of any keys that are defined with a keyboard action routine other than **sas-function-key**. The definitions of those keys in the KEYS window become labels that have no effect. See "Modifying the SAS.keyboardTranslations Resource" on page 58 for more information.

3 Modify/add the **SAS.keysWindowLabels** resource in your resource file. The **SAS.keysWindowLabels** resource specifies the set of valid labels that will appear in the SAS KEYS window. Modify this resource only if you want to add new labels or modify existing labels in the left column in the KEYS window.

   The **SAS.keysWindowLabels** resource defines only the mnemonics used in the KEYS window. For a specific key to perform an action, you must specify a **SAS.keyboardTranslations** definition for the key. See "Modifying the keysWindowLabels Resource" on page 59 for more information.

4 Start a SAS session and open the KEYS window.

5 In the right-hand column in the KEYS window, type a command name or other description of each key that you have defined.

See "Examples" on page 62 for examples of key definitions.

Determining Keysyms

You can use the **xev** utility to determine the keysyms associated with the keys on your keyboard. **xev** is distributed with most UNIX operating systems, but if **xev** is not installed on your operating system, contact the Technical Support Department at SAS Institute for assistance.

xev prints a message for each X event that occurs. The **KeyPress** event specifies the keysym for each key that is pressed.

1 Start **xev** on the X server for which you want to define keys. The **xev** client displays a small Event Tester window that lists the X events that occur. (The **xev** client generates a large amount of output, so you may want to save the output to a file for later review. You can issue the UNIX **script** command to save the output to a file.)

2 Give keyboard focus to the Event Tester window by clicking the mouse pointer on the window, if necessary.

3 Press the key that you want to define, and watch for the **KeyPress** event to be listed. The listing has a number of items that are separated by commas. One of the fields in the **KeyPress** event lists the keysym name that is associated with the key that was pressed.

   For example, when the 0 key on the keypad of an HP 9000/700 keyboard is pressed, it generates the following output:

```
KeyPress event, serial 14, synthetic NO,
  window 0x4400001,root 0x23, subw 0x4400002,
  time 507920400, (54,37),root:(67,66),
  state 0x0, keycode 30 (keysym 0xffb0, KP_0),
  same_screen YES,
  XLookupString gives 1 characters:  "0"
```

   In this example, the keysym name is **KP_0**.

Note: SAS defines a set of *virtual keysyms* with the **SAS.defaultVirtualBindings** resource. Virtual keysyms all begin with osf, such as

osfPageDown, osfClear, and osfPrimaryPaste. If you remap these virtual bindings instead of using the defaults supplied by SAS, you might get unexpected results. If you specify a key translation that does not work, you might be trying to redefine a key that is bound to a virtual keysym. In this case, you must specify the virtual keysym in the **SAS.keyboardTranslations** resource instead of the keysym displayed by **xev**. To determine the virtual keysym that is bound to a key, you can start the Resource Helper, select Keys, and press the key or key combination that you want to define. Resource Helper will display the virtual keysym name. You can also refer to the key definition files in **/X11/resource_files** in the directory where SAS is installed (**!SASROOT**) and to the man pages for VirtualBinding or xmbind. △

Modifying the SAS.keyboardTranslations Resource

Note: Most SAS documentation uses angle brackets (<>) to indicate optional syntax. However, in this topic, optional syntax is shown with square brackets ([]). The angle brackets that are shown in this topic are part of the syntax and should be entered exactly as shown. △

The syntax of the **SAS.keyboardTranslations** resource is

SAS.keyboardTranslations: #override \

[*modifier*] <Key>*keysym* : *action-routine* \n\

[*modifier*] <Key>*keysym* : *action-routine*

#override
indicates that this definition should override any existing bindings for the specific keys that you define without affecting any other keys. If you omit the **#override** directive, the new bindings replace all of the default bindings, and none of the other keys on the keyboard will be available.*

modifier
can be **Alt**, **Ctrl**, **Meta**, **Shift**, **Lock**, **Mod1**, **Mod2**, **Mod3**, **Mod4**, **Mod5**, **None**, or a blank space. The list of valid modifiers varies depending on your keyboard. To display a list of valid modifiers for your keyboard, enter the **xmodmap** UNIX command. Refer to the **man** page for **xmodmap** for more information.

<Key>
is required. It signals the beginning of the keysym.

keysym
is the key symbol recognized by X for the key that you are defining. See "Determining Keysyms" on page 57 for more information.

action-routine
is what you want the key to do. You can specify any action routine described in "SAS Keyboard Action Names" on page 59.

\n
allows the X translation manager to determine where one translation sequence ends and the next one begins. Do not enter \n after the end of the last translation.

\
prevents the newline character at the end of the line from being interpreted as part of the definition. This is a stylistic convention that allows each translation to

* For information on the **#augment** and **#replace** directives, refer to the documentation for the X Window System.

be listed on a separate line. Do not enter a backslash after the end of the last translation.

Note: The SAS System does not prevent you from specifying invalid keys in the **SAS.keyboardTranslations** resource. In some cases, invalid keys will produce warnings in the shell window. △

Modifying the keysWindowLabels Resource

Note: The square brackets ([]) in the following syntax indicate that the *(InternalKeyName)* is optional. △

The syntax of the **SAS.keysWindowLabels** resource is

SAS.keyWindowLabels: \

KeyWindowLabel [(*InternalKeyName*)] \n\

KeyWindowLabel [(*InternalKeyName*)]

KeyWindowLabel
is the label (1 to 8 characters) that you want to appear in the KEYS window.

InternalKeyName
is the character string that is passed to the **sas-function-key** action routine in the corresponding **SAS.keyboardTranslations** key binding. (*InternalKeyName* is used by the SAS System to correlate KEYS window entries to key definitions in the KEYS modules loaded from SAS catalogs or defined in the SAS KEYS window.) If the *InternalKeyName* is not specified, SAS uses the *KeyWindowLabel* as the *InternalKeyName*.

\n and \
serves the same purpose as in the **SAS.keyboardTranslations** resource. See "Modifying the SAS.keyboardTranslations Resource" on page 58 for more information.

SAS Keyboard Action Names

Note: Most SAS documentation uses angle brackets (<>) to indicate optional syntax. However, in this topic optional syntax is shown with square brackets ([]). The angle brackets that are shown in this topic are part of the syntax and should be entered exactly as shown. △

The SAS System declares a set of keyboard actions during X initialization. You can think of these keyboard actions as simple functions. When the actions are executed, they act on the window that currently has keyboard input focus.

The following list of keyboard actions represents action routines registered by the Motif interface for use with X toolkit keyboard event translations.

sas-cursor-down()
moves the cursor down one line in the SAS window. The cursor does not wrap when it reaches the bottom of the SAS window interior.

sas-cursor-left()
moves the cursor left one character in the SAS window. The cursor does not wrap when it reaches the left side of the SAS window interior.

sas-cursor-right()
moves the cursor right one character in the SAS window. The cursor does not wrap when it reaches the right side of the SAS window interior.

sas-cursor-up()
moves the cursor up one line in the SAS window. The cursor does not wrap when it reaches the top of the SAS window interior.

sas-delete()
deletes all text in the current field.

sas-delete-begin()
deletes text from the current cursor position to the beginning of the current text field.

sas-delete-char()
deletes the character under the text cursor and leaves the cursor in place.

sas-delete-end()
deletes text from the current cursor position to the end of the current text field.

sas-delete-prev-chr()
deletes the character to the left of the text cursor and moves the cursor back one space.

sas-delete-prev-word()
deletes text to the start of the previous word from the current cursor position. If the cursor is in the interior of a word when the action is invoked, the text from the cursor position to the start of the word is deleted.

sas-delete-word()
deletes text from the current cursor position to the end of the current or next word.

sas-do-command()
accepts one or more text string parameters that are interpreted as SAS commands to be executed when the action is invoked. The action may be invoked with multiple parameters. The parameters are concatenated with semicolon delimiters supplied by the sas-do-command action between the parameters. The assembled SAS command string is then submitted for execution. For example, the following translation syntax can be used to define a HOME, SUBMIT key sequence for all SAS windowing environment windows:

```
<Key>KP_F3: sas-do-command(HOME;SUBMIT)
```

sas-function-key("*InternalKeyName*")
invokes the SAS commands associated with the function key identified by the *InternalKeyName* label. *InternalKeyName* is the character string (1 to 8 characters long) that is passed to the keysWindowLabels resource. Enclose *InternalKeyName* in quotes. Refer to “Defining Key Translations” on page 56 for a description of internal key names.

sas-home-cursor()
is the equivalent of the HOME command. It is provided for convenience so that the HOME action may be defined for all SAS windowing environment windows.

sas-insert-char(["*InsertionString*"])
inserts or overwrites the character typed into the input field under the text cursor. Insert or overstrike behavior is determined by the sas-toggle-insert action, which

has a mode that is reflected by the text cursor style displayed; the block cursor indicates overstrike mode, and the underline cursor indicates insert mode. Normally, sas-insert-char translates the XKeyEvent into the appropriate character and inserts it at the SAS text cursor location. If you specify the parameter, the text string represented by this parameter is inserted at the SAS text cursor location. White space in the string is interpreted by the X Toolkit as a parameter delimiter unless you enclose the string in double quotes. Refer to your X Window System documentation for information on embedding quotes in the string parameter. To include an escaped quote, use the following syntax:

```
Shift<Key>KP_1:  sas-insert-char("One\\"1\\"")
```

This produces the text string **One"1"** at the SAS text cursor location.

sas-kp-application()
sets the workstation's numeric keypad to allow function key translations to be reinstated. This action only works for those keypad keys that are bound to sas-function-key() actions. Keypad bindings to other actions are not affected by this translation.

sas-kp-numeric()
sets the workstation's keypad to generate numeric characters instead of its previous function key assignment. This action only works for keypad keys that are bound to sas-function-key() actions. Keypad bindings to other actions are not affected by this translation.

sas-move-begin()
moves the cursor to the beginning of the current text field.

sas-move-end()
moves the cursor to the end of the current text field.

sas-new-line()
generates an end-of-line event when invoked. This is a context-sensitive action. If the action is typed on the SAS command line, the text entered will be submitted for execution. If invoked in the SAS application client area, the action depends on the attributes of the text area under the text cursor. In simplest terms, this action is the general line terminator for an input field.

sas-next-field()
advances the SAS application to the next field in the SAS window client area.

sas-next-word()
skips the text cursor forward to the beginning of the next word in the current text field. If sas-next-word does not find the beginning of a word in the current text field, it advances to the next SAS application field. If you are typing in the SAS command line area of the window, the cursor will not wrap into the SAS window client area.

sas-page-down()
scrolls the current window contents forward by one page.

sas-page-end()
moves the text cursor to the end of the current page.

sas-page-top()
moves the text cursor to the top of the current page.

sas-page-up()
scrolls the window contents backward by one page.

sas-prev-field()
returns the SAS application to the previous field in the SAS window client area.

sas-prev-word()
skips the text cursor backward to the beginning of the previous word in the current text field. If sas-prev-word does not find the beginning of a previous word in the current text field, it returns to the end of the previous SAS application field. If you are typing in the SAS command line area of the window, the cursor will not wrap into the SAS window client area.

sas-to-bottom()
Moves the text cursor to the absolute bottom of the window's text range.

sas-to-top()
Moves the text cursor to the absolute top of the window's text range.

sas-toggle-insert()
switches the associated window line-editing behavior between insert and overstrike modes. This only applies to the SAS command line and the SAS window client area. The current mode is indicated by the cursor style in use. The block cursor indicates overstrike mode, and the underline cursor indicates insert mode.

sas-xattr-key(<*KeyType*>[,<*KeyParam*>])
processes SAS extended attribute keys. The *KeyType* parameter must be one of the following values: XACOLOR, XAATTR, XACLEAR. For *KeyType* XACOLOR, the 12 DMS color names are valid parameters; for *KeyType* XAATTR, the valid values are HIGHLIGHT, REVERSE, BLINK, and UNDERLINE; for XACLEAR, no parameter is required. The BLINK attribute is not supported in the Motif interface. However, if you specify the BLINK attribute, it will be displayed when the catalog is ported to other operating environments.

Examples

Note: Most SAS documentation uses angle brackets (<>) to indicate optional syntax. However, in these examples, optional syntax is shown with square brackets ([]). The angle brackets that are shown in these examples are part of the syntax and should be entered exactly as shown. △

In the following example, the sas-do-command action routine specifies that the COMMAND command is to override any existing definition for **KP_0**.

```
SAS.keyboardTranslations: #override \n\
    None<Key>KP_0: sas-do-command(COMMAND)
```

All other keys retain their current definitions.

The following example binds the key sequence CTRL-K to the KEYS command and specifies that CTRL-D deletes the character under the cursor. Commands entered in the KEYS window for CTRL-K and CTRL-D will have no effect.

```
SAS.keyboardTranslations: #override\
    Ctrl<Key>k: sas-do-command(keys)\n\
    Ctrl<Key>d: sas-delete-char()
```

The following example specifies that the key associated with the keysym **hpClearLine** performs the command entered beside the **MyClrLn** label in the KEYS window.

```
SAS.keyboardTranslations: #override \
   <Key>hpClearLine : sas-function-key("ClearLn")
SAS.keysWindowLabels: MyClrLn(ClearLn)
```

The character string that appears inside the parentheses in the **SAS.keysWindowLabels** resource must match the string entered as the parameter to

the sas-function-key routine. The label (**MyClrLn**) can be any character string, and the keysym **hpClearLine** must be a valid keysym for your keyboard.

Customizing Fonts

SAS uses two main types of fonts:

- The system font is used in most dialog boxes and pull-down menus. SAS inherits the system font defined by the CDE ***.systemFont** resource. If this resource is not defined, SAS uses a helvetica font.
- Windowing environment fonts are used in SAS windows. You can change the SAS windowing environment font either through the Host Fonts dialog box or by specifying the resources in your resources file. The windowing environment font must be a fixed font.

Note: It is best to change fonts before invoking any applications. Changing fonts while applications are running might result in unexpected behavior. △

Using the Host Fonts Dialog Box

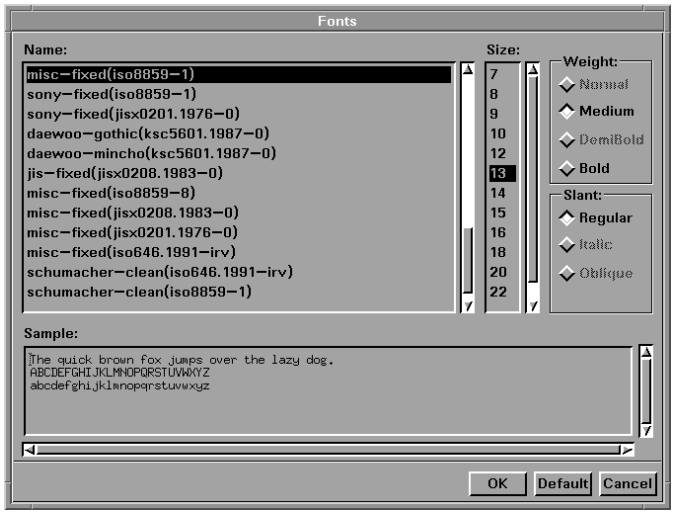
The Host Fonts dialog box allows you to change the windowing environment font and the printer font for printing text windows.

To change the windowing environment font, issue the DLGFONT command or select

Tools ► Options ► Fonts...

(To change printer fonts, you must access this dialog box through the Printer Properties dialog box as discussed in “Changing Printer Fonts” on page 147 .)

Display 3.8 Host Fonts Dialog Box



To change the font, select a new font name and, if desired, a size, weight, and slant. (Not all fonts are available in all sizes, weights, or slants.) The **Sample** field shows what the selected font looks like. After you have selected a name, size, weight, and slant, select **OK**.

The windowing environment font is stored in SASUSER.PROFILE.DMSFONT.UNXPREFS and will be used in future SAS sessions.

To return to the default font, select **Default**.

To cancel any changes and leave the Host Fonts dialog box, select **Cancel**.

Specifying Font Resources

You can customize the fonts used in the SAS windowing environment with the following resources:

SAS.DMSFont: *font-name*
specifies the font that you want to be used as the default normal font. The default normal font is Courier.

SAS.DMSboldFont: *font-name*
specifies the font that you want to be used as the default bold font.

SAS.DMSDBfont: *font-name*
specifies the multibyte normal character set font used by the SAS windowing system for operating environments that support multibyte character sets.

SAS.DMSDBboldFont: *font-name*
specifies the multibyte bold character set font used by the SAS windowing system for operating environments that support multibyte character sets.

SAS.DMSfontPattern: *XLFD-pattern*
specifies an X Logical Font Description, or XLFD pattern that you want SAS to use to determine the windowing environment font. Most fonts in the X Window System are associated with an XLFD, which contains a number of different fields delimited by a dash (–) character. The fields in the XLFD indicate properties such as the font family name, weight, size, resolution, and whether the font is proportional or monospaced. Refer to your X Window documentation for more information on the XLFD and font names used with X.

The *XLFD-pattern* that you specify for **SAS.DMSfontPattern** must contain the same number of fields as an XLFD. An asterisk (*) character means that any value is acceptable for that particular field. For example, the following pattern matches any font that has a regular slant, is not bold, is monospaced, and is an iso8859 font:

```
SAS.DMSFontPattern: -*-*-*-r-*--*-*-*-*-m-*-iso8859-1
```

SAS uses the *XLFD-pattern* to choose a font as follows:

1. SAS queries the X server for the list of fonts that match the **SAS.DMSfontPattern** resource.
2. SAS excludes all fonts that have X and Y resolution values different from the current X display, all fonts that have variable character cell sizing (such as proportional fonts), and all fonts that have point sizes smaller than 8 points or larger than 15 points. If this step results in an empty list, SAS chooses a generic (and usually fixed) font.
3. The font with the largest point size is chosen from the remaining list.

SAS.fontPattern: *XLFD-pattern*
specifies an XLFD font pattern that describes the candidate fonts used to resolve SAS graphics font requests. This allows the user to optimize or control the use of X fonts within the context of various SAS graphics applications. The default value of * usually does not affect performance to a significant degree. You may want to restrict the font search if you are running SAS on a server with an excessive number of fonts or that is operating in performance-limited environment.

How SAS Determines Which Font To Use

SAS determines the normal (not bold) default windowing environment font as follows:

1. If you have saved a font in SASUSER.PROFILE.DMSFONT.UNXPREFS through the Host Font dialog box, this font is used as the default normal font.
2. If you have not saved a font through the Host Font dialog box, but you have set the **SAS.DMSFont** resource, SAS uses the font specified by this resource as the default font.
3. If you have not set the **SAS.DMSFont** resource, SAS uses any *Font resources that you have defined.
4. If you have not set the *Font resources, but you have set the **SAS.DMSFontPattern** resource, SAS uses this resource to determine which font to use. The **SAS.DMSfontPattern** resource will have no effect if a *Font resource is defined.
5. If no resources have been set, SAS chooses a font from the fonts that are available on your server.

If you have not specified a value for the **SAS.DMSboldFont** resource, SAS uses the default normal font to determine the default bold font. If the normal **SAS.DMSFont** has an XLFD name associated with it, then SAS selects the matching bold font and loads it. If SAS cannot automatically select or load a bold font, the normal font is also used for the bold font.

In many cases, font names are given aliases so that a shorter name can be used to refer to a font that has an XLFD name associated with it. The name used in determining a bold font is based on the XA_FONT font property for the normal font.

Specifying Font Aliases

If your server does not provide fonts to match all of those supplied by the SAS system, you can use font alias resources to substitute the fonts available on your system. Include a line in your resource file with the following syntax:

SAS.*supplied-font*Alias: *substitute-family*

supplied-font is the name of the font supplied by the SAS System. *substitute-family* is the family name of the font you wish to substitute. For example, if your system does not have a Palatino font, but it does have a Lucida font, you can include the following line in your resources file to substitute Lucida for Palatino:

```
SAS.palatinoAlias: lucida
```

Table 3.1 on page 65 lists SAS font alias resource names.

Table 3.1 SAS Font Alias Resources

Resource Name	Class Name
SAS.timesRomanAlias	TimesRomanAlias
SAS.helveticaAlias	HelveticaAlias
SAS.courierAlias	CourierAlias
SAS.symbolAlias	SymbolAlias
SAS.avantGardeAlias	AvantGardeAlias

Resource Name	Class Name
SAS.bookmanAlias	BookmanAlias
SAS.newCenturySchoolbookAlias	NewCenturySchoolbookAlias
SAS.palatinoAlias	PalatinoAlias
SAS.zapfChanceryAlias	ZapfChanceryAlias
SAS.zapfDingbatsAlias	ZapfDingbatsAlias

CAUTION:
Do not specify a SAS font as a font alias. There may be a conflict if you specify a font supplied by the SAS System as a font alias, as in the following example:

```
SAS.timesRomanAlias: symbol
```

Assigning this value to a font alias prevents the selection of any symbol fonts through the font selection dialog box, because they are specified as the Times Roman alias. △

Customizing Colors

The SAS System ships to all sites a default set of colors and attribute settings for the elements of all SAS windows. You can customize the colors in your SAS session

- through Resource Helper (reshelper). Resource Helper allows you to customize any color. See “Setting X Resources with the Resource Helper” on page 44 and “Changing Colors with Resource Helper” on page 47 for more information.
- through the SASCOLOR window, as described in “Using the SASCOLOR Window” on page 66. You can customize any window element for most SAS windows with the SASCOLOR window.
- with the COLOR command as described in “Using the COLOR Command” on page 67. The COLOR command affects only the specified element of the active window. Changes made with the COLOR command override changes entered through any of the other methods described here.
- by entering the color resource specifications yourself. You can enter specific RGB values or color names for any of the X resources that control color. See “Defining Color Resources” on page 68 for more information.

Using the SASCOLOR Window

You can use the SASCOLOR window to change the color and highlighting of specific elements of SAS windows. To open the SASCOLOR window, issue the SASCOLOR command or select

Tools ► Options ► Colors...

Display 3.9 SASCOLOR Window

To change a color for a window element, select the element name, and then select color and attribute that you want assigned to the element.

The BLINK attribute is not supported. The HIGHLIGHT attribute causes text to be displayed in bold font.

When you select Save, your changes are saved to the catalog entry SASUSER.PROFILE.SAS.CPARMS.

For more information on the SASCOLOR window, refer to the online help.

Using the COLOR Command

You can use the COLOR command to set the color for specific elements of the active window:

`color` *`field-type`* `<`*`color`*`|NEXT <`*`highlight`*`>>`

field-type
specifies an area of the window such as background, banner, command, border, message, and so on.

color
specifies a color such as blue (which can be abbreviated B), red (R), green (G), cyan (C), pink (P), yellow (Y), white (W), black (K), magenta (M), gray (A), brown (B), or orange (O).

NEXT
changes the color to the next available color.

highlight
can be H (which causes text to be displayed in a bold font), U (underlined), or R (reverse video). The BLINK attribute is not supported.

To save your changes, issue the WSAVE command. The changes are saved to SASUSER.PROFILE.*window*.WSAVE.

Note: The WSAVE command is not available for all SAS windows. For example, with SAS/FSP or SAS/CALC software, changes are saved either through the EDPARMS or the PARMS window. (To determine whether WSAVE is available for a SAS window, refer to the product documentation.) △

Refer to the online help for more information on the COLOR and WSAVE commands.

Defining Color Resources

Color resources fall into two categories:

foreground and background definitions
: These resources allow you to customize the RGB values that are used to define the 12 DMS colors. Since each color could be used as either a background or a foreground color, you can specify different RGB values or color names for each color for each usage. For example, you can specify that when blue is used as a foreground color, color #0046ED is used, and when blue is used as a background color, CornflowerBlue is used.

window element definitions
: These resources, which are referred to as CPARMS resources, allow you to specify which of the 12 DMS colors you want to use for each window element. For example, you can specify that message text is displayed in magenta.

These two types of resources work together. The CPARMS color values use the current foreground and background definitions. For example, the following resources specify that the background of your primary windows will be CornflowerBlue:

```
SAS.blueBackgroundColor: CornflowerBlue
SAS.cparmBackground: DmBlue
```

Specifying RGB Values or Color Names for Foreground and Background Resources

The SAS System uses **SAS.systemBackground**, **SAS.systemForeground**, and the resources listed in Table 3.2 on page 68 to determine the colors to be used in its windows.

SAS.systemForeground: *color*
: specifies the color for the foreground system color in the SASCOLOR window.

SAS.systemBackground: *color*
: specifies the color for the background system color in the SASCOLOR window.

SAS.systemSecondaryBackground: *color*
: sets the system secondary background color and specifies the color for the secondary background system color in the SASCOLOR window.

You can specify color names such as MediumVioletRed or RGB values such as #0000FF for all of the foreground and background resources. Refer to your X Window System documentation for information on RGB color values.

The following table lists all of the foreground and background color resources and their class names. All of these resources are of the type String.

Table 3.2 Foreground and Background Color Resources

Resource Name	Class Name
SAS.systemForeground	SystemForeground
SAS.systemBackground	SystemBackground
SAS.systemSecondaryBackground	Background
SAS.blackForegroundColor	BlackForegroundColor

Resource Name	Class Name
SAS.blueForegroundColor	BlueForegroundColor
SAS.brownForegroundColor	BrownForegroundColor
SAS.cyanForegroundColor	CyanForegroundColor
SAS.grayForegroundColor	GrayForegroundColor
SAS.greenForegroundColor	GreenForegroundColor
SAS.magentaForegroundColor	MagentaForegroundColor
SAS.orangeForegroundColor	OrangeForegroundColor
SAS.pinkForegroundColor	PinkForegroundColor
SAS.redForegroundColor	RedForegroundColor
SAS.whiteForegroundColor	WhiteForegroundColor
SAS.yellowForegroundColor	YellowForegroundColor
SAS.blackBackgroundColor	BlackBackgroundColor
SAS.blueBackgroundColor	BlueBackgroundColor
SAS.brownBackgroundColor	BrownBackgroundColor
SAS.cyanBackgroundColor	CyanBackgroundColor
SAS.grayBackgroundColor	GrayBackgroundColor
SAS.greenBackgroundColor	GreenBackgroundColor
SAS.magentaBackgroundColor	MagentaBackgroundColor
SAS.orangeBackgroundColor	OrangeBackgroundColor
SAS.pinkBackgroundColor	PinkBackgroundColor
SAS.redBackgroundColor	RedBackgroundColor
SAS.whiteBackgroundColor	WhiteBackgroundColor
SAS.yellowBackgroundColor	YellowBackgroundColor

Defining Colors and Attributes for Window Elements (CPARMS)

You can define the colors and attributes for specific window elements by assigning values to SAS resources known as CPARMS. Each CPARMS resource defines the color and attribute of a specific window element, such as the background in a secondary window or the border of a primary window.

You can specify multiple color and attribute names in the same resource definition, but only the final color and attribute will be used:

```
SAS.cparm Resource: DmColorName|DmAttrName\
<+DmColorName|DmAttrName>
```

Resource can be any of the CPARMS resources listed in Table 3.3 on page 70. All of these resources are of type DmColor, and their default values are dynamic—that is, the default values are determined at run time.

Table 3.3 SAS CPARM Resources

Resource Name	Class Name	Specifies the color and attribute settings for...
SAS.cparmBackground	CparmBackground	backgrounds within all primary windows displayed in a SAS session
SAS.cparmForeground	CparmBackground	all editable text fields within a SAS window
SAS.cparmSecondaryBackground	CparmForeground	backgrounds in dialog boxes that prompt the user (secondary windows)
SAS.cparmBorder	CparmBackground	the border of a primary window
SAS.cparmSecondaryBorder	CparmForeground	the border of a secondary window
SAS.cparmBanner	CparmForeground	banners within window such as the command line prompt
SAS.cparmCommand	CparmForeground	the command data entry field when PMENUs are disabled
SAS.cparmMessage	CparmForeground	the message field
SAS.cparmNumber	CparmForeground	line numbers
SAS.cparmText	CparmForeground	text labels for row information. You can use this resource within the SAS editor to identify editing lines and in spreadsheets windows to label spreadsheets rows.
SAS.cparmInfo	CparmForeground	text that is displayed in a window as an aid to the user, for example: **Press Enter to continue**
SAS.cparmLabel	CparmForeground	text that precedes a widget, such as the text **Name:** in the following line is a label: **Name:** _____
SAS.cparmColumn	CparmForeground	text labels for column information. You can use this resource within the SAS editor to identify editing lines and in spreadsheet windows to label spreadsheet rows.
SAS.cparmHelpMainTopic	CparmForeground	topic words or phrases in the help system
SAS.cparmHelpSubTopic	CparmForeground	topic words or phrases in the help system
SAS.cparmHelpLink	CparmForeground	links to additional levels of HELP information
SAS.cparmError	CparmForeground	ERROR lines that are written to the Log or Output windows
SAS.cparmMark	CparmForeground	areas that have been selected for operations such as FIND, CUT, and COPY
SAS.cparmNote	CparmForeground	NOTE lines that are written to the Log or Output windows
SAS.cparmSource	CparmForeground	SAS source lines that are written to the Log window
SAS.cparmData	CparmForeground	general lines written to the Log or Output windows
SAS.cparmFootnote	CparmForeground	FOOTNOTE lines written to the Output window
SAS.cparmTitle	CparmForeground	TITLE lines written to the Output window
SAS.cparmHeader	CparmForeground	HEADER lines written to the Output window

Resource Name	Class Name	Specifies the color and attribute settings for...
SAS.cparmByline	CparmForeground	BY lines written to the Output window
SAS.cparmWarning	CparmForeground	WARNING lines written to the Log or Output windows

DmColorName can be any one of the following colors:

- DmBLUE
- DmRED
- DmPINK
- DmGREEN
- DmCYAN
- DmYELLOW
- DmWHITE
- DmORANGE
- DmBLACK
- DmMAGENTA
- DmGRAY
- DmBROWN

DmAttrName can be any one of the following attributes:

- DmHIGHLIGHT
- DmUNDERLINE
- DmREVERSE

For example, the following resources specify that all background colors are gray and all foreground colors are black:

```
SAS.cparmBackground: DmGRAY
SAS.cparmForeground: DmBLACK
```

These resources specify that errors should be displayed in red with reverse video, and warnings should be displayed in yellow with reverse video and a bold font:

```
SAS.cparmError: DmRED + DmREVERSE
SAS.cparmWarning: DmHIGHLIGHT + DmYELLOW + DmREVERSE
```

SAS looks for default CPARMS resources in two places:

- If your SAS Site Representative has entered color and attribute settings in the SASHELP.BASE.SAS.CPARMS catalog entry, then these settings become the default for your site.
- If you have saved settings in SASUSER.PROFILE.SAS.CPARMS, then these settings override the settings specified for your site.

Controlling Contrast

During interactive move/stretch operations, such as rubberbanding and dragging rectangles in SAS/INSIGHT software, you may find it hard to see the outline of the graphics primitive because of the lack of contrast between the primitive and the background. The XCONTRAST command makes the primitive visible against the background. The rendering performance and the aesthetic appearance of the primitive is compromised for the sake of visibility. You can enter XCONTRAST to act as a toggle, or you can specify XCONTRAST ON or XCONTRAST OFF.

In some color combinations, text fields, push buttons, check boxes, and other foreground categories may not be visible. The **SAS.dmsContrastCheck** resource makes these categories legible.

SAS.dmsContrastCheck: True | False
controls whether contrast mapping is applied to nongraphic foreground colors in a SAS window. The default value is False. A value of True specifies that DMS foreground colors will be remapped if necessary to produce a contrast. Some color usage based on graphic operations are not affected by this resource.

Controlling ODS Results

The Output Delivery System (ODS) is a method of delivering output in a variety of formats and of making the formatted output easy to access. ODS can produce an output data set, traditional monospace output (send output to the Listing destination), and output that is formatted in Hypertext Markup Language (send output to the HTML destination). For information about using ODS, refer to *The Complete Guide to the SAS Output Delivery System*.

SAS.htmlUsePassword: True | False
specifies whether SAS prompts you to enter your password before sending HTML files to your browser. The default value is True.

SAS.resultsAutoNavigate: True | False
specifies whether SAS automatically displays results files when they are generated. The default value is False. If you set **SAS.resultsAutoNavigate** to True, then you should set **SAS.htmlUsePassword** to False.

SAS.resultsHTML: True | False
specifies whether the ODS HTML destination is open or closed. The default value is False, which means that the HTML destination is closed.

SAS.resultsHTMLStyle: "*style*"
specifies the style definition to use for HTML output. The style definition controls such aspects as color, font name, and font size. The default style is "Default". You can specify any style that is defined in the \ODS\PREFERENCES\STYLES key in the SAS Registry. You can open the SAS Registry by issuing the REGEDIT command or by selecting

Solutions ▶ Accessories ▶ Registry Editor

SAS.resultsListing: True | False
specifies whether the ODS Listing destination is open or closed. The default value is True, which means that the Listing destination is open.

SAS.resultsTmpDir: "*directory*"
specifies the destination for HTML files. The default value is the NULL string.

SAS.resultsUseWork: True | False
specifies whether SAS should use your WORK directory as the destination for your HTML files. The default value is True.

Controlling Pull-down Menus

Pull-down menus are controlled by the following resources:

SAS.pmenuOn: True | False
forces the global PMENU state on regardless of the information stored by the WSAVE command. The WSAVE state of an individual window takes precedence over the global state. The default is True. (You can also use the PMENU ON and PMENU OFF commands to turn pull-down menus on and off.)

SAS.usePmenuMnemonics: True | False
specifies whether mnemonics are attached to the pmenus for the current SAS session. The default is True.

Customizing Cut-and-Paste

Note: For instructions on cutting and pasting text, see "Selecting (Marking) Text" on page 23 and "Copying or Cutting and Pasting Selected Text" on page 24 . △

There are four SAS paste buffers. Each SAS paste buffer is associated with a X paste buffer:

XPRIMARY
is associated with X primary selection (PRIMARY).

XSCNDARY
is associated with the X secondary selection (SECONDARY).

XCLIPBRD
is associated with the X clipboard selection (CLIPBOARD). This paste buffer allows you to use the MIT X Consortium xclipboard client with the SAS System.

XTERM
is associated with the paste buffer used by the xterm client. XTERM is the default buffer. DEFAULT is an alias for XTERM. If you copy or cut text into the XTERM buffer, the text is actually copied or cut into all four of the paste buffers. When you paste text from the XTERM buffer, the text is pasted from the XPRIMARY buffer.

XCUT*n*
is associated with X cut buffer*n* where 0 <= *n* <= 7.

If you are not sure which X data exchange protocols your other X clients are using, you should use the XTERM paste buffer. You can specify your default paste buffer with the **SAS.defaultPasteBuffer** resource:

```
SAS.defaultPasteBuffer: XTERM
```

If you know that the X clients in your workstation environment all use the X PRIMARY selections to exchange data, you should use the XPRIMARY paste buffer:

```
SAS.defaultPasteBuffer: XPRIMARY
```

This specification uses both SAS and X resources more efficiently and provides for the on-demand transfer of data between clients.

Sun OpenWindows desktop clients use the CLIPBOARD selection as the basis for their copy-and-paste operations. If you use the SAS XCLIPBRD paste buffer, you can exchange text directly with these clients.

You can also use the SAS XCLIPBRD paste buffer to interact with Motif clients that use the Motif clipboard mechanism for text exchanges. This clipboard mechanism makes it unnecessary to have a dedicated client such as xclipboard. For example, you can use XCLIPBRD to exchange text directly with the Motif xmeditor application when you select the **Cut**, **Copy**, or **Paste** items from the xmeditor **Edit** pull-down menu.

The Motif quick-copy data exchange and Motif clipboard data exchange mechanisms are specific to the Motif interface toolkit and are not currently supported as SAS paste buffers. However some dialog boxes, such as the File Selection dialog box, use Motif interface text widgets. In these dialog boxes, the Motif quick copy and clipboard data exchange mechanisms are available.

If you want SAS to automatically copy selected text into your paste buffer every time you mark a region of text with the mouse, you should also specify your paste buffer name in the **SAS.markPasteBuffer** resource:

```
SAS.markPasteBuffer: XTERM
```

Alternatively, because DEFAULT is an alias for XTERM, you could specify

```
SAS.markPasteBuffer: DEFAULT
```

The **SAS.markPasteBuffer** definition causes SAS to automatically issue a STORE command whenever you select text.

The STORE command, as well as the CUT and PASTE commands, support a BUFFER= option that specifies which buffer to use. When these commands are issued from function keys or pull-down menus whose definitions do not include the BUFFER= option, if the **SAS.markPasteBuffer** resource is not defined, these commands use BUFFER=DEFAULT. If this resource is defined, these commands use BUFFER=*buffer-name*.

You can customize your normal cut, copy, or paste keys to issue any of these commands with the BUFFER= option. For example, you can override the **SAS.keyboardTranslations** definition for the **osfCopy** and **osfPaste** keys with the following specifications:

```
SAS.keyboardTranslations: #override \
 <Key>osfCopy: sas-do-command(\"STORE BUFFER=XCLIPBRD\") \n\
 <Key>osfPaste: sas-do-command(\"PASTE BUFFER=XCLIPBRD\")
```

For more information on customizing keys, see “Customizing Key Definitions” on page 55.

When you cut or copy and paste text between SAS sessions using the XTERM, XPRIMARY, or XSCNDARYpaste buffers, the color and attribute information is preserved. However, if you copy and paste the same text into an xterm window while using the vi editor, the color and attribute information is lost. If you change the definition for **SAS.defaultPasteBuffer** and **SAS.markPasteBuffer** to XCUT0, then you will not retain the text and color attributes when you copy and paste text between two SAS sessions.

When you use the xclipboard client, SAS text attributes are not preserved in exchanges made between SAS sessions. However, when you use the XCLIPBRD paste buffer without a clipboard manager such as the xclipboard client, SAS text attributes are preserved in exchanges between SAS sessions.

Customizing Session Workspace, Session Gravity, and Window Sizes

The SAS System uses the following resources to determine the size of the session workspace, the gravity of the workspace, and the size of the windows. The default values for these resources are listed in Table 3.4 on page 80.

SAS.awsResizePolicy: grow | fixed
controls the policy for resizing AWS windows as interior windows are added and removed. Possible values include the following:

grow	the AWS window will attempt to grow any time an interior window is grown or moved, in order to show all interior windows, but it will not shrink to remove dead areas.
fixed	the AWS window will attempt to size itself to the size of the first interior window and will not attempt any further size changes.

SAS.maxWindowHeight: *units*
specifies the number of units for the maximum height of a window. The unit is specified by the **SAS.windowUnitType** resource.

SAS.maxWindowWidth: *units*
specifies the number of units for the maximum width of a window. The unit is specified by the **SAS.windowUnitType** resource.

SAS.noAWS: True | False
controls whether each of your application's windows appears in its own native window rather than in an Application Work Space (AWS). The default is False, which confines all windows displayed by an application to a single AWS window.

SAS.scrollBarSize: *pixels*
specifies the default size of the scroll bar in pixels.

SAS.sessionGravity: *value*
controls the region of the screen where the SAS System will attempt to place its windows. This resource may be ignored by some window manager configurations. Possible values include the following:

CenterGravity
EastGravity
WestGravity
SouthGravity
NorthGravity
SouthEastGravity
NorthEastGravity
SouthWestGravity
NorthWestGravity

SAS.sessionGravityXOffset: *offset*
specifies an x offset to be added when the SAS System attempts to place a window in the gravity region.

SAS.sessionGravityYOffset: *offset*
specifies a y offset to be added when the SAS System attempts to place a window in the gravity region.

SAS.windowHeight: *units*
specifies the number of units for the default height of a window. The unit is specified by the **SAS.windowUnitType** resource.

SAS.windowUnitType: character | pixel | percentage
specifies the unit type for **SAS.windowWidth**, **SAS.windowHeight**, **SAS.maxWindowWidth**, and **SAS.maxWindowHeight**. Possible values include the following:

character
units specify the number of rows and columns.

pixel
units specify the number of pixels.

percentage
units specify the percentage of the screen.

SAS.windowWidth: *units*
specifies the number of units for the default width of a window. The unit is specified by the **SAS.windowUnitType** resource.

Specifying User-defined Icons

You can add your own icons to those icons that are supplied with the SAS System. For example, if you want to use your own color icons in the toolbox, define the **SAS.colorUiconPath**, **SAS.colorUiconCount**, and **SAS.sasUiconx** resources. Then, when you are defining tools in the tool editor, the tool editor will include your icons in the display of icons that you can choose for each tool.

SAS.colorUiconPath: *search-path*
specifies the file search path for locating user-defined color icon files. This string resource specifies the directory paths to be searched for an icon file. These files should be in X Pixmap (xpm) format. Use a comma to separate individual directory pathnames. For example, the following string first searches for icon files in the **/usr/lib/X11/pixmaps** directory and then in the **/usr/lib/X11/pixmaps/SAS** directory:

```
SAS.colorUiconPath : /usr/lib/X11/pixmaps, \
/usr/lib/X11/pixmaps/SAS
```

SAS.colorUiconCount: *num-icons*
specifies the number of user-defined color icons that are available for the SAS System to use.

SAS.uiconCount: *num-icons*
specifies the number of user-defined icons that are available for use in the SAS session.

SAS.uiconPath: *search-path*
specifies the file search path for locating user-defined icon bitmap files. This string resource specifies the directory paths to be searched for an icon file. These files should be in X Bitmap (xbm) format. Use a comma to separate individual directory pathnames. For example, the following string will first search for bitmap files in the **/usr/lib/X11/bitmaps** directory and then in the **/usr/lib/X11/bitmaps/SAS** directory:

```
SAS.uiconPath : /usr/lib/X11/bitmaps,\
/usr/lib/X11/bitmaps/SAS
```

SAS.sasUiconx: *name*
associates a value with the filename of an X bitmap or pixmap file. *X* is a number assigned to the file. A file extension of **.xbm** or **.xpm** is automatically supplied.

The resource name used to locate the icon bitmap filename for user icon number *x* is **SAS.sasUiconx**.

For example, to define the filename **myicon** for the user icon **1**, you should define the resource:

```
SAS.sasUicon1: myicon
```

If the resource name is not defined, SAS generates a filename of the form **sasui*nnn*.xbm** or **sasui*nnn*.xpm**. The path elements from the **SAS.uiconPath** or **SAS.colorUiconpath** resource are searched in sequence until the icon file is found or until the search path is exhausted.

For example, the following set of X resources defines a collection of color icons.

```
SAS.colorUiconPath: /users/jackaroe/pixmaps/
SAS.colorUiconCount: 7
SAS.sasUicon1: adsetup
SAS.sasUicon2: adverse
SAS.sasUicon3: altmenu
SAS.sasUicon4: batch
SAS.sasUicon5: is
SAS.sasUicon6: patgrps
SAS.sasUicon7: pctchg
```

The Motif interface will search for icon **sasUicon1** in a file named **/users/jackaroe/pixmaps/adsetup.xpm**.

Miscellaneous Resources

You can also customize the following resources:

SAS.altVisualId: *ID*
specifies a visual type ID.

SAS.autoSaveInterval: *minutes*
specifies how often (in number of minutes) that the data from the Program Editor window should be saved.

SAS.autoSaveOn: True | False
specifies that data from the Program Editor window should be saved to a file at intervals specified by the **SAS.autoSaveInterval** resource.

SAS.confirmSASExit: True | False
controls whether SAS displays the Exit dialog box when you enter the DLGENDR command or select

File ▶ Exit...

The default is True.

SAS.defaultCommandWindow: True | False
specifies whether the command window is invoked when you start your SAS session. The default is True.

SAS.directory: *directory-pathname*
specifies the directory that you want when you first invoke the Open dialog box. By default, the Open dialog box uses the current directory.

SAS.graphicsClipboardPath: *directory-name*
specifies the directory to place the GSTORE temporary file in. The default directory is **/usr/tmp** and is usually the best place to store the temporary file. To cut and paste graphs and images between between SAS sessions, the setting of

this resource must be identical in both sessions, and the path must be accessible to the systems on which the sessions are running.

SAS.hasXPrinter: True | False
specifies whether host printing is turned on. The default is False.

SAS.helpBrowser: *pathname*
specifies the pathname of the World Wide Web browser that you want to use for viewing the online help. The default browser is Netscape.

SAS.insertModeOn: True|False
controls the editing mode in SAS editor windows. The default is False (overtype).

SAS.noDoCommandRecall: True | False
controls whether or not SAS commands submitted through the sas-do-command() action routine are recorded in the command recall buffer. The default value of True causes commands to be omitted from the command recall buffer; a value of False causes them to be recorded.

SAS.pattern: *default-pattern*
specifies the default pattern that you want to be used as the file filter when you first invoke the Open and Import Image dialog boxes. This pattern is displayed in the text field at the top of the dialog box. By default, the dialog box uses the first filter in the File type list. The pattern resource has no effect on the File type field.

SAS.selectTimeout: *seconds*
specifies the X toolkit selection conversion timeout value in units of seconds. This determines the amount of time that the SAS System will wait for a request to convert an X toolkit selection to complete. The default value should be adequate in most cases.

SAS.startupLogo: *xpm-filename* | None | ""
specifies the XPM file that you want the SAS System to display when it is initialized. If the string is empty, the SAS System uses the default logo.

SAS.startSessionManager: True|False
specifies whether SAS automatically starts the session manager when a new SAS session is started. Using your own host editor with SAS requires that the session manager be running. The default is True.

SAS.wsaveAllExit: True|False
specifies whether SAS should issue the WSAVE ALL command when you end your session. This command saves the global settings, such as window color and window position, that are in effect for all windows that are currently open. The default is False.

SAS.webBrowser: *pathname*
specifies the pathname of the World Wide Web browser that you want invoked when the WBROWSE command is issued. The default browser is Netscape.

Configuring the SAS System for Host Editor Support

The SAS System supports the use of a host text editor with the Motif interface, so you can use an editor such as vi or EMACS with your SAS session. There is no host editor set as the default host editor, so you must specify one to use this feature. Host editor support requires the use of the **motifxsassm** client. (See "Using the SAS Session Manager (motifxsassm)" on page 33 for more information.)

To use your host text editor with SAS, use the EDITCMD system option to specify the command required to invoke your editor. Then, use the HOSTEDIT command to invoke the editor as needed. The HOSTEDIT command passes data from a SAS window to the host editor. When you save in the host editor, the data is copied back into the SAS window if the window is writable. HED is an alias for the HOSTEDIT command. See "EDITCMD" on page 264 and "HOSTEDIT" on page 177 for more information.

When you issue the HOSTEDIT command from a SAS text editor window, the contents of the buffer for that window are written to a temporary file in the directory specified by the SASWORK option. A command invoking the specified host editor is passed to the SAS Session Manager. The session manager issues the command to the operating environment to invoke the editor for the temporary file.

The X display used with the HOSTEDIT command is the same one used with your SAS session.

The EDITCMD system option specifies the command that is issued to the operating environment. If you are using a terminal-based editor, such as vi, you must specify a command that runs the editor inside a terminal emulator window.

You can define the EDITCMD option using the SASV8_OPTIONS environment variable as part of a configuration file or on the command line to make the definition available automatically to the SAS System. The option must be specified as a quoted string. You can use either single or double quotes. You can change the value for the EDITCMD option during a SAS session by issuing an OPTIONS statement.

Text attributes, such as color and highlighting, are not transferred between a host editor window and a SAS text editor window. Issue the HEATTR ON command to display a dialog box that will warn you if you are editing text with highlighting and color attributes that will be removed by the host editor. This dialog box prompts you to continue or abort the HOSTEDIT command. Specify HEATTR OFF to suppress this dialog box.

After you return to the SAS text editor window, you can issue the UNDO command to undo all of the changes that you made with your host editor. You must issue the UNDO command a second time to return to the state of the window before the HOSTEDIT command was issued. If you issue the HOSTEDIT command in a read-only window, you can save your editing changes to an external file, but the SAS text editor window remains unchanged.

Some systems have an X-based editor installed that is called xedit. If you want to use xedit with the HOSTEDIT command, you can invoke SAS with the following command:

```
sas -editcmd '/usr/local/bin/xedit'
```

The vi editor is a terminal-based editor that requires a terminal window. The xterm client's **-e** option runs a program when the xterm client is invoked. To use the EDITCMD option to display an xterm client in conjunction with vi, invoke SAS as follows:

```
sas -editcmd '/usr/bin/X11/xterm -e /usr/bin/vi'
```

Summary of SAS Resources

Table 3.4 on page 80 lists the instance and class names, type, and default values for many of the SAS resources. See the following tables for additional resources of specific types:

- "SAS Font Alias Resources," Table 3.1 on page 65
- "Foreground and Background Color Resources," Table 3.2 on page 68
- "SAS CPARM Resources," Table 3.3 on page 70.

Table 3.4 SAS Resources

Resource Name	Class Name	Type	Default
SAS.altVisualId	AltVisualId	int	NULL
SAS.autoComplete	AutoComplete	Boolean	TRUE
SAS.autoSaveInterval	AutoSaveInterval	int	10
SAS.autoSaveOn	AutoSaveOn	Boolean	TRUE
SAS.awsResizePolicy	AWSResizePolicy	String	grow
SAS.colorUiconCount	UiconCount	int	0
SAS.colorUiconPath	UiconPath	String	NULL
SAS.commandsSaved	CommandsSaved	int	25
SAS.confirmSASExit	ConfirmSASExit	Boolean	TRUE
SAS.defaultCommandWindow	DefaultCommandWindow	Boolean	TRUE
SAS.defaultPasteBuffer	DefaultPasteBuffer	String	XTERM
SAS.defaultToolBox	DefaultToolBox	Boolean	TRUE
SAS.directory	Directory	String	NULL
SAS.dmsContrastCheck	DmsContrastCheck	Boolean	FALSE
SAS.DMSDBFont	Font	String	*dynamic*
SAS.DMSDBboldFont	Font	String	*dynamic*
SAS.DMSboldFont	Font	String	*dynamic*
SAS.DMSFont	Font	String	*dynamic*
SAS.DMSfontPattern	DMSFontPattern	String	-*-*-*-r-*–*-*-*-*-m-*-iso8859-1
SAS.fontPattern	FontPattern	String	*
SAS.graphicsClipboardPath	GraphicsClipboardPath	String	/usr/tmp
SAS.hasXPrinter	HasXPrinter	Boolean	FALSE
SAS.helpBrowser	HelpBrowser	String	netscape
SAS.htmlUsePassword	HtmlUsePassword	Boolean	TRUE
SAS.insertModeOn	InsertModeOn	Boolean	FALSE
SAS.isToolBoxPersistent	IsToolBoxPersistent	Boolean	TRUE
SAS.keyboardTranslations	KeyboardTranslations	Translation	*dynamic*
SAS.keysWindowLabels	KeysWindowLabels	String	*dynamic*
SAS.markPasteBuffer	MarkPasteBuffer	String	XTERM
SAS.maxWindowHeight	WindowHeight	Dimension	95
SAS.maxWindowWidth	WindowWidth	Dimension	95
SAS.noAWS	NoAWS	Boolean	FALSE

Resource Name	Class Name	Type	Default
SAS.noDoCommandRecall	NoDoCommandRecall	Boolean	TRUE
SAS.pattern	Pattern	String	NULL
SAS.pmenuOn	PmenuOn	Boolean	TRUE
SAS.resultsAutoNavigate	ResultsAutoNavigate	Boolean	FALSE
SAS.resultsHTML	ResultsHTML	Boolean	FALSE
SAS.resultsHTMLStyle	ResultsHTMLStyle	String	Default
SAS.resultsListing	ResultsListing	Boolean	TRUE
SAS.resultsTmpDir	ResultsTmpDir	String	NULL
SAS.resultsUseWork	ResultsUseWork	Boolean	True
SAS.sasUicon	SasUicon	String	NULL
SAS.scrollBarSize	ScrollBarSize	Dimension	17
SAS.selectTimeout	SelectTimeout	int	60
SAS.sessionGravity	SASGravity	String	NorthWestGravity
SAS.sessionGravityXOffset	SASGravityOffset	int	0
SAS.sessionGravityYOffset	SASGravityOffset	int	0
SAS.startSessionManager	StartSessionManager	Boolean	TRUE
SAS.startupLogo	StartUpLogo	String	NULL
SAS.toolBoxAlwaysOnTop	ToolBoxAlwaysOnTop	Boolean	TRUE
SAS.toolBoxTipDelay	ToolBoxTipDelay	int	750
SAS.uiconCount	UiconCount	int	0
SAS.uiconPath	UiconPath	String	NULL
SAS.useCommandToolBoxCombo	UseCommandToolBoxCombo	Boolean	TRUE
SAS.useLargeToolBox	UseLargeToolBox	Boolean	FALSE
SAS.usePmenuMnemonics	UsePmenuMnemonics	Boolean	TRUE
SAS.useShowHideDecorations	UseShowHideDecorations	Boolean	FALSE
SAS.useToolBoxTips	UseToolBoxTips	Boolean	TRUE
SAS.wsaveAllExit	WsaveAllExit	Boolean	FALSE
SAS.webBrowser	WebBrowser	String	Netscape
SAS.windowHeight	WindowHeight	Dimension	50
SAS.windowWidth	WindowWidth	Dimension	67
SAS.windowUnitType	WindowUnitType	String	percentage

The correct bibliographic citation for this manual is as follows: SAS Institute Inc., *SAS® Companion for UNIX Environments, Version 8*, Cary, NC: SAS Institute Inc., 1999.

SAS® Companion for UNIX Environments, Version 8

ISBN 1–58025–502–7

SAS Institute Inc., SAS Campus Drive, Cary, North Carolina 27513.

1st printing, October 1999