

KCMS Calibrator Tool Loadable Interface Guide

2550 Garcia Avenue
Mountain View, CA 94043
U.S.A.



SunSoft
A Sun Microsystems, Inc. Business

© 1995 Sun Microsystems, Inc. 2550 Garcia Avenue, Mountain View, California 94043-1100 U.S.A.

All rights reserved. This product or document is protected by copyright and distributed under licenses restricting its use, copying, distribution, and decompilation. No part of this product or document may be reproduced in any form by any means without prior written authorization of Sun and its licensors, if any.

Portions of this product may be derived from the UNIX[®] system, licensed from UNIX System Laboratories, Inc., a wholly owned subsidiary of Novell, Inc., and from the Berkeley 4.3 BSD system, licensed from the University of California. Third-party software, including font technology in this product, is protected by copyright and licensed from Sun's suppliers.

RESTRICTED RIGHTS LEGEND: Use, duplication, or disclosure by the government is subject to restrictions as set forth in subparagraph (c)(1)(ii) of the Rights in Technical Data and Computer Software clause at DFARS 252.227-7013 and FAR 52.227-19. The product described in this manual may be protected by one or more U.S. patents, foreign patents, or pending applications.

TRADEMARKS

Sun, Sun Microsystems, the Sun logo, SunSoft, the SunSoft logo, Solaris, SunOS, OpenWindows, DeskSet, ONC, ONC+, and NFS are trademarks or registered trademarks of Sun Microsystems, Inc. in the United States and other countries. UNIX is a registered trademark in the United States and other countries, exclusively licensed through X/Open Company, Ltd. OPEN LOOK is a registered trademark of Novell, Inc. PostScript and Display PostScript are trademarks of Adobe Systems, Inc.

All SPARC trademarks are trademarks or registered trademarks of SPARC International, Inc. in the United States and other countries. SPARCcenter, SPARCcluster, SPARCcompiler, SPARCdesign, SPARC811, SPARCengine, SPARCprinter, SPARCserver, SPARCstation, SPARCstorage, SPARCworks, microSPARC, microSPARC-II, and UltraSPARC are licensed exclusively to Sun Microsystems, Inc. Products bearing SPARC trademarks are based upon an architecture developed by Sun Microsystems, Inc.

The OPEN LOOK[®] and Sun[™] Graphical User Interfaces were developed by Sun Microsystems, Inc. for its users and licensees. Sun acknowledges the pioneering efforts of Xerox in researching and developing the concept of visual or graphical user interfaces for the computer industry. Sun holds a non-exclusive license from Xerox to the Xerox Graphical User Interface, which license also covers Sun's licensees who implement OPEN LOOK GUIs and otherwise comply with Sun's written license agreements.

X Window System is a trademark of X Consortium, Inc. Kodak is a trademark of Eastman Kodak Company.

THIS PUBLICATION IS PROVIDED "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR NON-INFRINGEMENT.

THIS PUBLICATION COULD INCLUDE TECHNICAL INACCURACIES OR TYPOGRAPHICAL ERRORS. CHANGES ARE PERIODICALLY ADDED TO THE INFORMATION HEREIN. THESE CHANGES WILL BE INCORPORATED IN NEW EDITIONS OF THE PUBLICATION. SUN MICROSYSTEMS, INC. MAY MAKE IMPROVEMENTS AND/OR CHANGES IN THE PRODUCT(S) AND/OR THE PROGRAMS(S) DESCRIBED IN THIS PUBLICATION AT ANY TIME.



Contents

1. Overview	1
In This Chapter	1
Color Calibration	1
About KCMS Calibrator Tool	2
Purpose of Calibrator Tool	2
Colorimeter Software	2
Platforms and Support	3
Features	3
Installation	3
How Calibrator Tool Works With A Loadable Module	4
Software Components	4
Calibrator Tool Responsibilities	4
Loadable Module Responsibilities	4
Calibrator Tool and Loadable Module Interaction	5
2. Calibrating A Monitor	7

In This Chapter.	7
Before You Read Further.	8
Revisiting the Calibrator Tool User Interface.	8
Updating Profiles.	12
Error and Status Messages.	12
3. OWconfig Database	13
In This Chapter.	13
OWconfig Entry	14
Formatting Guidelines	14
Using Script Files	16
Appending an Entry	16
Removing An Entry.	19
Locating and Opening the Module	21
4. Measuring Monitor Response	23
In This Chapter.	23
Determining Displayable Visuals	23
Visuals Not Measured	24
Slow and Fast Mode Measurements	24
Precautions While Taking Measurements.	25
Recommended Number of Measurements	25
Allocating and Deallocating Structures For Measurement Data	25
Handling the Measurement Data.	25
5. Data Structures	27
KCMSCData.	28

KCMSCVisuals	29
XVisualInfo	29
6. Functions	31
KCMSCMonClose()	32
Purpose	32
Arguments	32
Return Value	32
KCMSCMonInit()	33
Purpose	33
Arguments	33
Return Value	33
KCMSCMonMeasure()	34
Purpose	34
Arguments	34
Return Value	34

Figures

Figure 2-1	Calibrator Tool Setup Window	8
Figure 2-2	Selecting a Loadable Module	10
Figure 2-3	Interface to a Loadable Module	11

Tables

Table 2-1	Events Leading Up To Loading Module	9
Table 3-1	Calibration Module Attributes and Meanings	16

Code Samples

Code Example 3-1	OWconfig File Entry	14
Code Example 3-2	Using the OWconfig File Entry as a Template.....	17
Code Example 3-3	Script for Appending an Entry	18
Code Example 3-4	Appending An Entry	19
Code Example 3-5	Script For Removing an Entry.....	20
Code Example 3-6	Removing an Entry.....	20

Preface

The *KCMS Calibrator Tool Loadable Interface Guide* describes how to create a dynamically loadable device handler to be used in calibrating devices. Your colorimeter software interacts with the Kodak Color Management System (KCMS) Calibrator Tool to gather the color measurements needed to correct ICC format profiles. The measurement data is then used by the KCMS framework to achieve the device independent display of color images. Currently calibration of color monitor devices only is supported.

This guide is part of the software development kit (SDK) portion of the KCMS software product. It is located in the Solaris™ 2.5 KCMS AnswerBook.

Who Should Use This Book

This guide is intended to be used by programmers who are writing dynamically installable modules (for example, colorimeter device handlers) that provide KCMS Calibrator Tool with color measurement data. This guide assumes its readers are familiar with the KCMS color management software and the colorimeter hardware.

Before You Read This Book

Before reading this guide, you should

- Read the *KCMS Application Developer's Guide*, which is also located in the KCMS AnswerBook. The *KCMS Calibrator Tool Loadable Interface Guide* assumes you understand the color concepts (such as characterization, profiles, and calibration) described in the Developer's Guide. The Developer's Guide also describes the KCMS API library and provides background information on the interface between Calibrator Tool, the KCMS library, and color profiles.
- Check the following manuals for any corrections or updates to the information in this guide:
 - *Solaris 2.5 Software Developer Kit Introduction*
 - *Solaris 2.5 Software Developer Kit Installation Guide*
- See the on-line SUNWrdm packages for information on bugs and issues, engineering news, and patches. For Solaris™ installation bugs and for late-breaking bugs, news, and patch information, see the *Solaris Installation Notes* (SPARC™ or X86).
- (SPARC systems) Consult the updates your hardware manufacturer may have provided.

How This Book Is Organized

This guide is organized into the following chapters:

Chapter 1, "Overview," is an overview of color calibration. Read this chapter to understand the main software components involved in calibration. The chapter summarizes how Calibrator Tool (which is executed with the `kcms_calibrate(1)` command) interacts with the dynamically loadable module to obtain calibration data. (For details on `kcms_calibrate(1)`, see the man page description.)

Chapter 2, "Calibrating A Monitor," describes calibration from the end-user perspective and the action of the underlying sample software.

Chapter 3, "OWconfig Database," details how you edit the `OWconfig` file to add correctly formatted entries for dynamically loadable modules involved in calibration.

Chapter 4, "Measuring Monitor Response," touches on the rationale Calibrator Tool uses to gather measurement data.

Chapter 5, “Data Structures,” describes the structures Calibrator Tool uses for measurement data.

Chapter 6, “Functions,” alphabetically presents and describes the interfaces a loadable module uses to interact with Calibrator Tool.

Glossary defines words and phrases used in this guide.

Related Books

Sun Publications

For information using KCMS Calibrator Tool see the *Solaris Advanced User’s Guide*. The section entitled “Calibrating Your Monitor,” in Chapter 10, “Customizing Your Environment,” contains information on how to calibrate your monitor with Calibrator Tool. The manual is located in the Solaris 2.5 User AnswerBook.

For basic information on color concepts and KCMS, see the first two white papers listed in Table P-1. The other two white papers provide background information on your viewing environment. These files are located online in the `/usr/openwin/demo/kcms/docs/` directory.

Table P-1 KCMS White Papers

File Name	Title
<code>kcms-wp.ps</code>	<i>An Introduction to the Kodak Color Management System</i>
<code>kcms-wp-solaris.ps</code>	<i>Kodak Color Management System</i>
<code>reducing_eyestrain.ps</code>	<i>Reducing Eyestrain From Computer Monitors</i>
<code>video_mon_adj.ps</code>	<i>Video Monitor Adjustments: “BlackLevel” and “Picture”</i>

X Window System Information

The following X Window System™ manuals are available through SunExpress or your local bookstore. Contact your SunSoft™ representative for information on ordering.

- *Xlib Programming Manual*, O’Reilly & Associates, Inc.
- *Xlib Reference Manual*, O’Reilly & Associates, Inc.

- *X Window System, Third Edition*, Digital Press

What Typographic Changes Mean

Table P-2 describes the typographic changes used in this book.

Table P-2 Typographic Conventions

Typeface or Symbol	Meaning	Example
AaBbCc123	The names of commands, files, and directories; on-screen computer output	Edit your <code>.login</code> file. Use <code>ls -a</code> to list all files. machine_name% You have mail.
AaBbCc123	What you type, contrasted with on-screen computer output	<div style="border: 1px solid black; padding: 5px;"> machine_name% su Password: </div>
<i>AaBbCc123</i>	Command-line placeholder: replace with a real name or value	To delete a file, type <code>rm filename</code> .
<i>AaBbCc123</i>	Book titles, new words or terms, or words to be emphasized	Read Chapter 10 in <i>User's Guide</i> . These are called <i>class</i> options. You <i>must</i> be root to do this.

Shell Prompts in Command Examples

Table P-3 shows the default system prompt and superuser prompt for the C shell, Bourne shell, and Korn shell.

Table P-3 Shell Prompts

Shell	Prompt
C shell prompt	machine_name%
C shell superuser prompt	machine_name#
Bourne shell and Korn shell prompt	\$
Bourne shell and Korn shell superuser prompt	#

Naming Conventions

In this guide, *KCMS* refers to the Kodak Color Management System. The names of calibration functions and data structures consist of the prefix string 'KCMS' plus other significant words that suggest what the function or data structure does. To illustrate, `KCMSMonInit()` is a function name consisting of four significant words:

- KCMS
- C
- Mon
- Init

Table P-4 describes each significant word.

Table P-4 Naming Conventions

Significant Word	Meaning
KCMS	Kodak Color Management System
C	Calibrator
Mon	Monitor device
Init	Initialization function

Note – The prefix strings 'KCS' and 'kcs' also are used to refer to the Kodak Color Management System product in related documentation.

Overview



In This Chapter

This guide describes how to create a dynamically loadable module that interfaces with Kodak Color Management System (KCMS) Calibrator Tool to calibrate color devices.

This chapter provides an overview of calibration. The chapter

- Introduces color calibration and KCMS Calibrator Tool
- Identifies the platforms that support calibration
- Summarizes the interaction between your dynamically loadable module and Calibrator Tool

Color Calibration

KCMS color calibration uses software to adjust the output of a color device for accurate color reproduction. Currently KCMS Calibrator Tool supports calibrating color monitor devices only. Monitor calibration is accomplished by displaying a programmed sequence of test colors and measuring the output of the display. The KCMS library then computes color correction factors necessary to compensate for discrepancies between the programmed colors and those actually displayed, and it updates profiles with the resulting data.

About KCMS Calibrator Tool

Purpose of Calibrator Tool

KCMS Calibrator Tool is an interactive program designed to allow users to calibrate their own monitors to later display color-corrected data. The Calibrator Tool program is executed by the `kcms_calibrate(1)` command. (For details on `kcms_calibrate(1)`, see the man page description.)

KCMS Calibrator Tool obtains color measurements from the measurement gathering device handler. It uses the data it collects to update the monitor's profile. A *profile* describes a device, including its type, chromaticities, and generic color response values.

KCMS software uses profiles to achieve the device-independent display of color data. The values KCMS Calibrator Tool writes to profiles represent the color response of the specific monitor from which the measurements were taken. The result is data whose color values are adjusted for that monitor to provide accurate color that is more consistent across different device types and under different viewing conditions.

(For details on the C-function interface between Calibrator Tool, the KCMS color management library, and the profiles, see the *KCMS Application Developer's Guide*.)

When one of the updated profiles is later used to display an image on the monitor, updated color data are used in place of the monitor profile generic data. The new profile more accurately represents the color image viewing conditions, monitor settings, and the state of the monitor (such as brightness and contrast). As such, the image rendered is a significant improvement over the one created by the generic profile.

This color correction process is similar to gamma correction. It can be more accurate, however, because it represents color correction at a finer level of detail than a single gamma value.

Colorimeter Software

You provide the device handling software for your colorimeter to measure the color monitor response. Using software functions, Calibrator Tool loads your module dynamically as a shared library. This guide refers to the user-written

colorimeter device handler sample software as the *loadable module* to distinguish it from the KCMS Calibrator Tool (*main program*) with which it interacts.

Platforms and Support

KCMS Calibrator Tool currently calibrates color monitors only and is supported on workstations running the Solaris environment. The tool requires a color frame buffer and a color monitor. The code for one of the two sample loadable modules provided describes an interface to a colorimeter.

The ability to add loadable modules to Calibrator Tool is part of the KCMS portion of the Solaris SDK.

Features

Calibrator Tool provides the following features:

- It supports different measurement methods (with and without a colorimeter)
- It allows the end user to select from among several specific monitor devices to calibrate
- It automatically configures a system to use X Window System Version 11 visuals
- It provides error and status handling information
- It is internationalized using standard SunSoft utilities

Note that, although Calibrator Tool and the sample loadable modules provided with the KCMS portion of the SDK implement their GUI using Motif library routines, this is not a requirement for your loadable module. It can use alternate GUIs.

Installation

The `kcms_calibrate(1)` command is installed in the `/usr/openwin/bin` directory and the manual page, in the `/usr/openwin/man/man1` directory.

How Calibrator Tool Works With A Loadable Module

Software Components

There are two main software components to calibration: Calibrator Tool (main program) and one or more OEM-written loadable modules. Calibrator Tool is started with the `kcms_calibrate(1)` command. Then it works together with loadable modules to obtain color data and to update color profiles.

Calibrator Tool Responsibilities

Calibrator Tool obtains measurements to update a monitor's color profile. It obtains the data it needs from the loadable software module.

Calibrator Tool obtains the measurement data through two data structures that it passes to the loadable module. It allocates and deallocates the memory for the structures and sets up their formats based on the number and types of X11 visuals that the X Window System supports. Using standard X11 routines, it determines which X visuals are supported on the system. Then it creates a profile copy for each visual. Finally it hands off this information formally packaged in the data structures to the loadable module.

Once it retrieves the measurement data from the loadable module, Calibrator Tool updates the X visual profile copies on the local machine.

Loadable Module Responsibilities

The loadable module doesn't need to know anything about the color profiles or where they are stored. It simply performs the required measurements and fills out the data structures passed in by Calibrator Tool. Then it returns the information to the main program.

This is a simplification, of course. The module uses some mechanism for obtaining the calibration data from the monitor device. The KCMS portion of the Solaris SDK provides two sample dynamically loadable modules. One module requires the use of a hardware colorimeter connected to a serial port on the local system to obtain color data measurements. The other obtains measurement data without a colorimeter. In either case, the module provides a GUI for interaction with an end user.

Calibrator Tool and Loadable Module Interaction

Calibrator Tool and the loadable module interact with each other using three application programming interface (API) functions. These are

- `KCMSCMonInit()`
- `KCMSCMonMeasure()`
- `KCMSCMonClose()`

Each programming interface identifies a phase (initialization, data collection, or closure) in the interaction between Calibrator Tool and the loadable module. To indicate the success or failure of a phase, each module returns status information. A zero status value indicates success; any nonzero value indicates that an error has occurred. If a nonzero status value is returned at any point in the calibration process, the loadable module application is terminated and Calibrator Tool displays a status message. (For details on Calibrator Tool status messages, see the section entitled “Customizing Your Monitor” in Chapter 10, “Customizing Your Environment,” in the *Solaris Advanced User’s Guide*.) The loadable module may have its own custom error and status messages.

The following summarizes the three phases in the interaction between Calibrator Tool and the module.

Initialization

During initialization, Calibrator Tool takes a handle to the dynamically opened module and accesses the `KCMSCMonInit` symbolic name using `dlsym(3X)`. If it finds the name, it initializes and allocates space for the visual data.

The tool calls `KCMSCMonInit()` to allow the loadable module to initialize the serial port for the hardware colorimeter (if it uses one to take measurements), to start its own GUI-based application, and to make any additional preparations necessary to measure monitor data.

Note – The sample modules provided with the SDK set up the application user interface during initialization using included source code from TeleUSE (Motif GUI builder) and the X Toolkit library (available with Solaris 2.5). (Refer to the `colorsense.c` and `XSolarisVisualGamma.c` source files.) Be aware that you are not required to use this code in the design of your module. The examples are an implementation only. All or parts of them can be used as a template, depending on your needs.

Data Collection

Calibrator Tool initializes and sets up the data structure to gather data. It makes standard X11 Window System calls to establish the number and types of X visuals for which it needs measurement data. The function's interface structure includes the array of visuals supported by the system, which is provided by the main program, and the matching array of measurements, which is supplied by the loadable module.

Calibrator Tool calls `KCMSCMonMeasure()` to “fill in” the measurement data.

Closure

If the loadable module successfully supplies the data measurements, it returns status value 0 (status OK) to Calibrator Tool. Calibrator Tool then calls `KCMSCMonClose()` to allow the loadable module to free any resources it is using, for example, to unlock the serial port and to close the associated file descriptor.

If, however, the status value returned is any other nonzero value, Calibrator Tool disregards all previously obtained data and informs the end user that calibration has not properly completed. In this case, the calibration process must be restarted from the very beginning with another call to the loadable module.

Calibrating A Monitor



In This Chapter

Chapter 1 introduced the major software components involved in calibration. This chapter describes calibration from the end-user perspective. The end user starts calibration and provides the necessary parameters (such as the device type, what module to load, and various other options for performing the calibration).

Note - The SDK provides two “loadable” source modules that work with Calibrator Tool to gather measurement data: `colorsense` and `XSolarisVisualGamma`. To use the `colorsense` module, the end user must purchase the `colorsense` colorimeter device. `XSolarisVisualGamma` presents the same GUI but does not require a colorimeter device.

The calibration process this chapter describes is how the sample loadable module works with Calibrator Tool. This method of obtaining measurement data may or may not apply to the design of your loadable module.

The chapter is meant to provide you with a better understanding of the sample loadable module. To accomplish this, it refers back and forth between the end user actions and those performed by the underlying software.

Before You Read Further

Before you read this chapter, you should review the section entitled “Calibrating Your Monitor” in Chapter 10, “Customizing Your Work Environment,” in the *Solaris Advanced User’s Guide*. That section provides additional information on calibration not repeated in this chapter, including

- General calibration concepts
- How to adjust the viewing environment before calibrating a monitor
- How to run Calibrator Tool

Revisiting the Calibrator Tool User Interface

Recall that when the end user starts Calibrator Tool, a setup window is displayed. This setup window is shown in Figure 2-1.

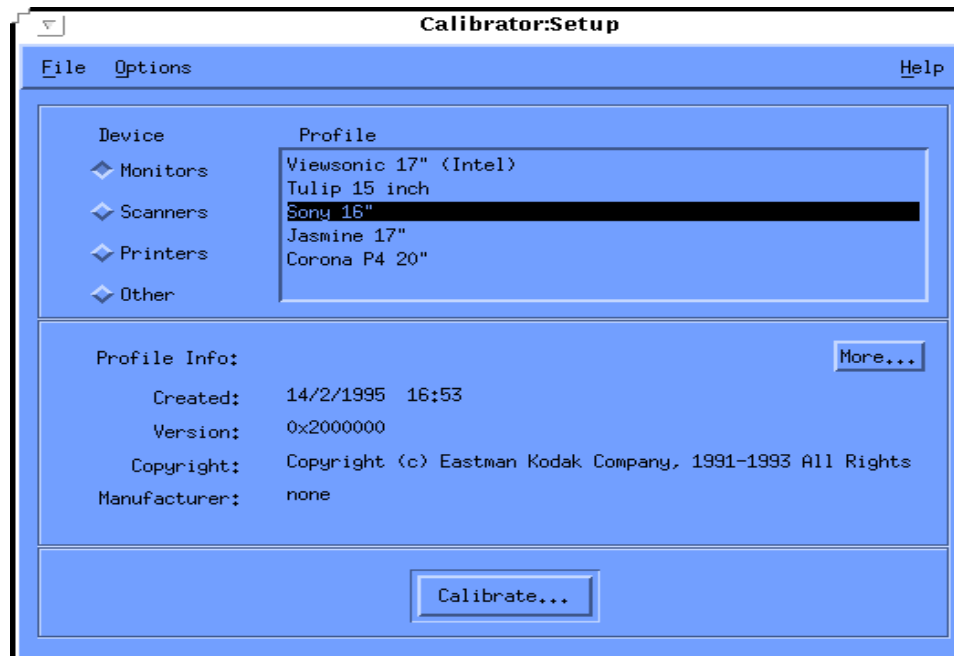


Figure 2-1 Calibrator Tool Setup Window

Table 2-1 summarizes the end-user actions and the actions of the underlying software that lead up to loading the `XSolarisVisualGamma` or `colorsense` module into memory.

Table 2-1 Events Leading Up To Loading Module

End-User Actions	Underlying Software
Starts Calibrator Tool by typing <code>kcms_calibrate</code> at the command-line prompt and pressing Return.	Calibrator Tool displays the window in <code>/usr/openwin/bin</code> . (See Figure 2-1.)
Selects Monitors as the Device type. (See Figure 2-1.)	By default, Calibrator Tool searches the following predefined libraries for monitor profiles: <code>/usr/openwin/etc/devdata/profiles</code> <code>/etc/openwin/devdata/profiles</code>
Selects the monitor profile (for example Sony 16). (See Figure 2-1.)	KCMS library functions open and obtain the generic device description of the user-selected monitor. For each X11 Window System visual supported by the frame buffer, KCMS library functions make a copy of the generic profile from <code>/usr/openwin/etc/devdata/profiles</code> or <code>/etc/openwin/devdata/profiles</code> .
Clicks on Calibrate. (See Figure 2-1.)	Calibrator Tool displays the Calibrator:Devices popup window. (See Figure 2-2.)
Selects the <code>XSolarisVisualGamma</code> device, for example, and clicks on Load.	The server dynamically loads the <code>XSolarisVisualGamma</code> module into memory shared with the Calibrator Tool main module. (The module is expected to be installed in the directory <code>/usr/openwin/etc/devhandlers</code> .) Then the loadable module displays its own GUI. (See Figure 2-3.)

The `/usr/openwin/etc/devdata/profiles` directory contains the names of files describing the color characteristics of the generic devices. Currently only monitors are supported by Calibrator Tool. Monitor profile names typically have the extension `.mon`. Profiles exist for other device types as well,

such as scanners (profile names with the extension `.inp`) and printers (profile names with the extension `.out`). The Calibrator Tool window in Figure 2-1 on page 8 also lists another category called Other for additional device types.

Figure 2-2 shows the Calibrator Tool popup window that lists the names of the loadable modules. The SDK includes the source code for sample modules shown in the figure. As previously mentioned, the `colorsense` module listed in the figure uses a colorimeter (hardware puck connected to an RS-232C serial port on the monitor) that the end user would need to purchase to obtain measurement values. `XSolarisVisualGamma` doesn't require this hardware. In either case, the measurement values obtained are used to build a table of color correction data for the particular visual.

The source code for the sample modules is provided in the `/opt/SUNWsdk/kcms` directory. You can use either of these modules as a template to create your own customized version of a loadable module.

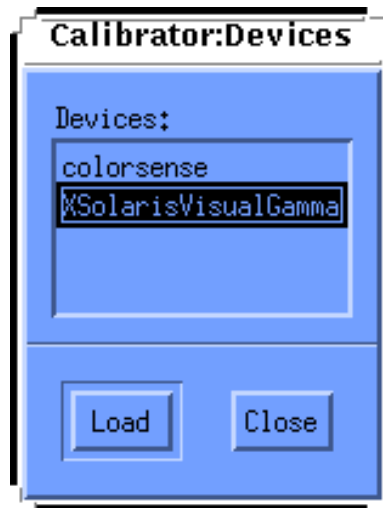


Figure 2-2 Selecting a Loadable Module

When the end user clicks on Load, Calibrator Tool uses the `OWconfig` database file to extract the names of the available loadable modules. You must edit `OWconfig` to add an entry for each loadable module you want to use with

Calibrator Tool. OWconfig is the database from which a module is located and later dynamically loaded. For details on OWconfig and the format of calibration module entries, see Chapter 3, “OWconfig Database.”

Figure 2-3 shows the GUI that XSolarisVisualGamma (or colorsense) displays when the end user selects the module from Calibrator Tool’s Calibrator:Devices window.

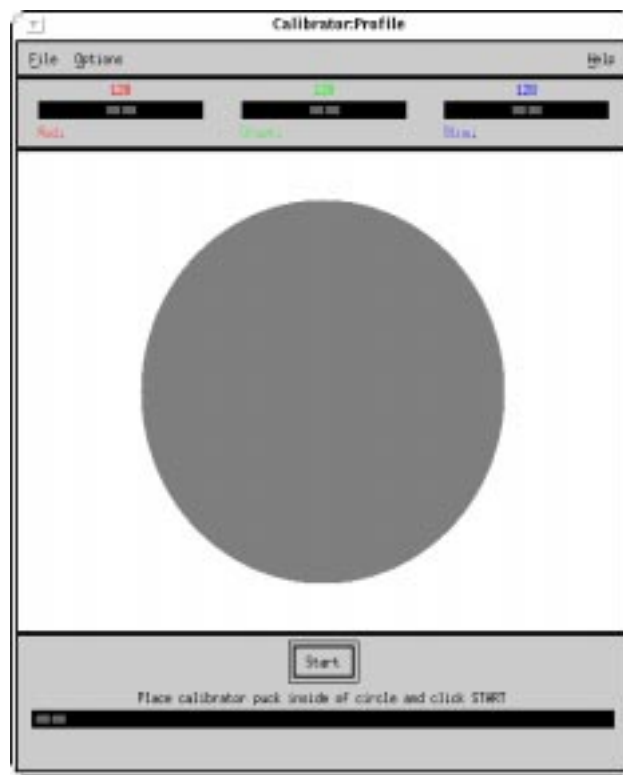


Figure 2-3 Interface to a Loadable Module

The significant point here is that, if your module makes use of a colorimeter, you need to consider a user interface design. The sample modules implement a color patch to obtain data from the monitor and, at the same time, to provide feedback to the user that calibration is actually taking place. Visibly each

module provides feedback on its data gathering activity by displaying the color patch, which changes from cyan to magenta to yellow as the luminance values of each color are being read.

Updating Profiles

Calibrator Tool uses the data it collects from the loadable module to update the X Window System profiles.

Error and Status Messages

See Chapter 10, “Customizing Your Work Environment,” in the *Solaris Advanced User’s Guide*, for error and status messages that Calibrator Tool displays. For example, a message would be displayed if the loadable module did not finish collecting data. This situation would occur if, for example, the hardware failed to complete measurement or the end user cancelled the measurement. In such a case, the loadable module returns a nonzero status to Calibrator Tool and the entire measurement process must be restarted.

Note – It is up to the writer of the loadable module to provide the end user with any special status or error messages.

In This Chapter

To provide for maximum flexibility in adding software modules, your module is dynamically loaded as a shared object at run time. The list of loadable objects (modules) is maintained in a system configuration database file called `OWconfig`. `OWconfig` must be edited using scripts to add entries for modules that will be used with Calibrator Tool.

By default, Calibrator Tool reads `OWconfig` from the `/usr/openwin/server/etc` directory. If it does not find the `OWconfig` file or the modules in it that it is looking for, it reads the `OWconfig` file from `/etc/openwin/server/etc`.

This chapter describes `OWConfig` and its format. The chapter explains how you can create entries for the calibration modules you want to add to this file. The chapter also describes how the server uses the information in `OWconfig` entries to locate a dynamically loadable module.

Note – `OWconfig` is a database for various dynamically loadable objects. It not only can contain entries for calibration but also can include entries for other extensions and X Window System modules. The guidelines in this chapter pertain specifically to modules that interact with Calibrator Tool.

OWconfig Entry

Code Example 3-1 shows an OWconfig text file entry for two dynamically loadable calibration modules.

You create a text file that contains an OWconfig entry for your dynamically loadable module(s). Then you execute a script to append your entry to the system OWconfig file. You also can remove your entry with a script. This chapter describes both of these procedures and provides templates you can use for adding and removing entries.

Code Example 3-1 OWconfig File Entry

```
# Start SUNWkcsrt
#####
# This is database for the calibrator tool
#####

# Kcs calibrate class and the name of the class.

# Module that contains handlers for Kodak ColorSense

class="KCMS_CALIBRATE" name="colorsense"
    kcmsCalDeviceType="monitor"
    kcmsCalLoadableModule="kcmscsUNWcolorsense.so.1";

# Module that contains handlers for the XSolarisVisualGamma

class="KCMS_CALIBRATE" name="XSolarisVisualGamma"
    kcmsCalDeviceType="monitor"
    kcmsCalLoadableModule="kcmscsUNWgamma.so.1";

# End SUNWkcsrt
```

Formatting Guidelines

OWconfig has some basic formatting guidelines you should be aware of. These are described below.

Any characters following the pound character (#) through the end of a line are treated as a comment and are disregarded when the file is read.

Quotation marks around value strings are required only if the string contains delimiters such as white space or a semicolon (;).

The back slash character (\) can be used as an escape character. For example, \" is used to include the double quotation mark character as part of a string value.

Parsing routines strip the quotation marks surrounding string values and pass the string only to the underlying software. The parsing software treats all values as strings; interpretation of the string value is up to the module.

The strings 'Start' and 'End' indicate the beginning and end of the database definitions for a set of loadable modules. In Code Example 3-1, for example, the string 'SUNWkcsrt' identifies the Kodak Color Management System (kcs) loadable modules. The contents of the string you use for your entry are up to you. You will use the string in a script file to append or delete your entry from the `OWconfig` file. (See "Using Script Files" on page 16, for details.)

Between the 'Start' and 'End' strings are one or more loadable module 'resource' entries for calibration. Each calibration module description consists of a set of four attribute=value pairs separated by white space (including tabs, spaces, and new line characters) and terminated by a semicolon.

Table 3-1 describes the meaning of each attribute=value pair.

Table 3-1 Calibration Module Attributes and Meanings

Attribute=	Meaning
class=	This value string is always KCMS_CALIBRATE, which uniquely identifies the class of loadable modules that interact with Calibrator Tool.
name=	This value is the name of the loadable module. It uniquely identifies the instance of the class object in the <code>OWconfig</code> file and must be different for each loadable module. This name string will appear in the Calibrator Tool Calibrator:Devices window. (See Figure 2-2 on page 10 in Chapter 2, “Calibrating A Monitor.”)
kcmsCalDeviceType=	This value is accessed when the end user chooses a particular device type to calibrate. It identifies the type of the device and is used to create a list of loadable modules for this device. The names of all the modules of the specified device type are displayed in the Calibrator:Devices window. (See Figure 2-1 on page 8 in Chapter 2.) There can be several monitor device type entries. Each is uniquely identified by the name= value.
kcmsCalLoadableModule=	This value is the name of the shared object that will be loaded when the end user clicks on Load. It has all the information necessary to start its own application to measure the monitor data. The module is expected to be installed in the <code>/usr/openwin/etc/devhandlers/</code> directory.

Using Script Files

Before your module can be loaded, you need to add an entry for it in `OWconfig`. To add (or delete) `OWconfig` entries, you create an executable script file.

Appending an Entry

You add an entry to `OWconfig` by appending the entry to the file. The following procedure illustrates this process.

1. Create an entry.

Use Code Example 3-1 on page 14 as a template to create your entry, filling in values for your module. Replace the string associated with 'Start' and 'End' ('SUNWkcsrt') with a string of your choice. Code Example 3-2 is exactly the same as Code Example 3-1 except 'Start' and 'End' are associated with the string 'Morrismodule'.

2. Save your entry.

Code Example 3-2 is saved locally as OWconfig.

Code Example 3-2 Using the OWconfig File Entry as a Template

```
# Start Morrismodule
#####
# This is database for the calibrator tool
#####

# Kcs calibrate class and the name of the class.

# Module that contains handlers for Kodak ColorSense

class="KCMS_CALIBRATE" name="colorsense"
    kcmsCalDeviceType="monitor"
    kcmsCalLoadableModule="kcmscSUNWcolorsense.so.1";

# Module that contains handlers for the XSolarisVisualGamma

class="KCMS_CALIBRATE" name="XSolarisVisualGamma"
    kcmsCalDeviceType="monitor"
    kcmsCalLoadableModule="kcmscSUNWgamma.so.1";

# End Morrismodule
```

3. Create a script file to append your entry.

Use the script file shown in Code Example 3-3 as a template. Note the line associating 'Start' and 'End' with 'Morrismodule'. It is repeated below:

```
sed -e "/# Start Morrismodule/,/# End Morrismodule/d" \
```

You need to replace this string with the one for your own entry. Then save the script file with a name (for example `ins.OWconfig`) and use the `chmod(1)` command to make it executable.

Code Example 3-3 Script for Appending an Entry

```

#!/bin/ksh
#
# Installation script for the OWconfig class
# If an OWconfig file existed, remove any entry belonging to
# this package, and append a new entry
#
echo $1
echo $2
chmod 644 $2
  if [ -r $2 ]
  then
    #An OWconfig file already exists
    if [ -w $2 ]
    then
      #It's editable by this script, edit it
      cp $2 /tmp/$$OWconfig || exit 2
      sed -e "/# Start Morrismodule/,/# End Morrismodule/d" \
      /tmp/$$OWconfig > $2 || exit 2
      cat $1 >> $2 || exit 2
      rm -f /tmp/$$OWconfig
    else
      #An OWconfig file exists that's not editable
      exit 2
    fi
  else
    #An OWconfig file was not present
    cat $1 >> $2 || exit 2
  fi
chmod 444 $2
exit 0

```

4. Execute the script file.

First you need to become a superuser. Code Example 3-4 illustrates how to do this. The executable script file `ins.OWconfig` takes two arguments: the relative path to the local text file entry (`OWconfig`) and the full path to the system `OWconfig` file (`usr/openwin/server/etc/OWconfig`).

As the script file executes, it displays the values (OWconfig and /usr/openwin/server/etc/OWconfig) of the two arguments passed to it.

Code Example 3-4 Appending An Entry

```
morris-120% su
Password:
# ./ins.OWconfig OWconfig /usr/openwin/server/etc/OWconfig
OWconfig
/usr/openwin/server/etc/OWconfig
# ^D
```

Removing An Entry

To remove the entry inserted by the above procedure, use the following procedure.

1. Use the script in Code Example 3-5 as a template.

Replace the string 'Morrismodule' in Code Example 3-5 with the string for the entry you want to remove. Save the script with a name such as `rm.Owconfig`, and make the file executable.

Code Example 3-5 Script For Removing an Entry

```
#!/bin/ksh
#
# Removal script for the OWconfig class
# Remove any extries that belong to this module
# Delete the file if it's empty
#
echo $1
chmod 644 $1
    sed -e "/# Start Morrismodule/,/# End Morrismodule/d" $1 > \
/tmp/$$OWconfig || exit 2
    if [ -s /tmp/$$OWconfig ]
    then
        mv /tmp/$$OWconfig $1 || exit 2
    else
        rm $1 || exit 2
    fi
chmod 444 $1
exit 0
```

2. Execute the script.

First you need to be a superuser. Code Example 3-6 illustrates how to so this. In this case, the script file `rm.Owconfig` takes one argument: the path to the system `OWconfig` file. As the script file executes, it displays the value (`/usr/openwin/server/etc/OWconfig`) of the single argument passed to it.

Code Example 3-6 Removing an Entry

```
morris-125% su
Password:
# ./rm.Owconfig /usr/openwin/server/etc/OWconfig
/usr/openwin/server/etc/OWconfig
# ^D
```

Locating and Opening the Module

Calibrator Tool supplies the `OWconfig` values as arguments to the `OWconfig` library routines. Using the values, the `OWconfig` routines find and dynamically open the loadable module. Then, using `dlsym(3X)`, they execute the module functions.

Measuring Monitor Response



In This Chapter

This chapter touches on the rationale Calibrator Tool uses to set up the structures for measuring data. In addition the chapter identifies precautions you can take in the design of your loadable module to be sure that it provides Calibrator Tool with accurate and meaningful measurement data.

Determining Displayable Visuals

When measuring monitor response, Calibrator Tool has to determine what X Window System visuals the frame buffer is capable of displaying. This is important because some frame buffers have their own gamma correction tables while others output directly to the monitor. Typically, however, frame buffers do not provide gamma correction, and gamma values are approximately 2.22 for SPARC systems. To identify the display capabilities of different visuals, Calibrator Tool calls the standard X Window System library routine `XGetVisualInfo(3)` and the Solaris library routine `XSolarisGetVisualGamma(3)`. This combination of routines returns a list of all the visual structures and their gamma values. (For details on `XGetVisualInfo`, see the *Xlib Reference Manual* listed in “Related Books” in the Preface to this guide. For details on `XSolarisGetVisualGamma(3)`, see the man page description.)

Visuals Not Measured

Currently Calibrator Tool does not measure GrayScale or StaticGray visuals, because they are not color visuals.

Furthermore, it does not measure StaticColor visuals (24-bit TrueColor being the exception). This is because the loadable module has to display an *exact* color on the screen when it is measuring a color. To accomplish this, the loadable module must be able to access the writable color cell to modify the color at will. The loadable module has no direct control over the RGB values for StaticColor visuals, so it does not measure them.

Slow and Fast Mode Measurements

Calibrator Tool provides two modes of measurement: slow and fast. In fast mode, the Calibrator Tool main program requests the loadable module to return the minimum number of measurements to increase the speed of acquiring the measurement data. In slow mode, it requests that the loadable module return measurements for all the visuals whose color it has writable control over. These distinct modes provide support for frame buffers that might have different color responses at different depths.

In slow mode, visuals are considered to be *equivalent* if they have the same gamma and depth. In this case, measurements are required for one representative of a class of equivalent visuals. If, for example, 24-bit DirectColor and 24-bit TrueColor have the same visual gamma, they are considered to be equivalent, and measurement is performed for TrueColor only. Since measurement is a response of the monitor to a particular depth and gamma, the measurement is not repeated for DirectColor. Instead, the DirectColor visual profile is updated with the TrueColor measurement results.

In fast mode, visuals are considered to be equivalent if they have the same gamma, regardless of depth. This mode is provided for monitors whose response is believed to be the same for varying numbers of planes. If, for example, 8-bit PseudoColor and 24-bit TrueColor have the same gamma, the PseudoColor measurement is the only one performed, with the result being used to update the TrueColor as well as the PseudoColor profiles. In this case, it doesn't matter which visual generates the color patches as long as the visuals represent the same color.

Precautions While Taking Measurements

The sample modules display a color patch that changes between cyan, magenta, and yellow as the end user takes measurements. When taking measurements from the monitor display using a hardware puck, it is important to ensure that the window displaying the patch is not obscured inadvertently by some other window tool. The sample modules prevent interference by using Motif's override shells, which are always on top of every other window. If by chance another window is moved into the color patch display region, that window passes under the override shell window and does not disturb the color patch display.

Recommended Number of Measurements

Although there is no specified number of measurements for visuals, it is recommended that the number be large enough for the KCMS API library to perform a reasonable interpolation of data for each color channel. The `XSolarisVisualGamma` (or `coloursense`) module determines the number of intensity levels based on the `bits_per_rgb` value of the visual (that is, $2^{\text{bits_per_rgb}}$). Then it measures every fourth intensity level to speed up measurement. If, for example, it is measuring a 6-bit frame buffer with 64 intensity levels, it performs 17 measurements. For an 8-bit frame buffer, it performs 65 measurements (including intensity 0). For intensity levels that are less than 8 `bits_per_rgb`, you must skip fewer intensity levels to achieve a more accurate resulting response curve.

Allocating and Deallocating Structures For Measurement Data

Calibrator Tool handles allocation and deallocation of `KCMSCVisuals` and `KCMSCData` structures. Once the loadable module completes the shared data structure `KCMSCData` (by filling in the output portion of the `KcsCalibrationData` structure for each of the measured X visuals) it returns, leaving the data in the shared structure.

Handling the Measurement Data

Calibrator Tool uses the data provided by the loadable module to update the X Window System profiles using calls from the KCMS API library.

Data Structures

5 

This chapter describes the data structures used by Calibrator Tool.

KCMSCData

```
typedef struct _KCMSCData {
    int                size; /* number of data sets */
    KcsCalibrationData ** data; /* pointers to data sets */
} KCMSCData;
```

KCMSCData is a structure that collects data. For monitor calibration, the size element must be equivalent in value to numVisuals in the KCMSCVisuals structure. (For more information on numVisuals, see “KCMSCVisuals” on page 29 in this chapter.)

The members in this structure are described below:

size	Is the number of sets of data. For monitors, the value must be equivalent to the value of numVisuals.
KcsCalibrationData	Is a variable array of pointers to KcsCalibrationData structures. KcsCalibrationData contains an array of input values for the profiles (supplied by Calibrator Tool) and an array of output values (results of measurements) that represent the color response of the device. (For details on the KcsCalibrationData type, see Chapter 3, “Data Structures” in the <i>KCMS Application Developer’s Guide</i> .)

KCMSCVisuals

```
typedef struct _KCMSCVisuals {
    Display * display;
    int      screen;
    int      numVisuals; /* dynamic array size */
    XVisualInfo ** visuals; /* visuals to get data for */
} KCMSCVisuals;
```

KCMSCVisuals is a structure that identifies to the loadable module the display and visuals upon which to perform measurements. The members in this structure are described below:

display	Is a Display pointer. The Display pointer is for use when the loadable module makes an X library call (such as XCreateWindow) to connect to the local windowing system. (For details on Display, see the <i>XLib Programming Manual</i> .)
screen	Is the screen number to be used in Xll calls. Calibrator Tool fills in the value of this field.
numVisuals	Is the number of X visuals for which color measurement data is needed. Calibrator Tool fills in the value of this field based on the number of X visuals to be updated. (For details, see “Slow and Fast Mode Measurements” on page 24 in Chapter 4, “Measuring Monitor Response.”)
visuals	Is an array of XVisualInfo pointers. (See “XVisualInfo” below for the format of this structure.)

XVisualInfo

The XVisualInfo structure has the format shown below. For details on the members of this structure, see the *XLib Programming Manual*.

```
typedef struct {
    Visual* visual;
    VisualID visualid;
    int screen_num;
    unsigned int depth;
    int class;
    unsigned long red_mask;
    unsigned long green_mask;
    unsigned long blue_mask;
    int colormap_size; /* Same as map_entries member of Visual */
    int bits_per_rgb;
} XVisualInfo;
```


Functions



This chapter alphabetically presents each calibration function your loadable module calls to interface with KCMS Calibrator Tool. For each function, the chapter provides its purpose, arguments, and return values.

Once a module is dynamically opened, Calibrator Tool uses `dlsym(3X)` to access the module functions by their symbolic names. Then it adds the names to the process symbol space.

≡ 6

KCMSCMonClose ()

```
int  
KCMSCMonClose (void);
```

Purpose

KCMSCMonClose () performs any cleaning necessary (for example, unlocking the serial port) for the module code.

Arguments

None.

Return Value

Returns 0 if successful; returns any other nonzero value if unsuccessful.

KCMSCMonInit()

```
int  
KCMSCMonInit (KCMSCVisuals *vis_data);
```

Purpose

KCMSCMonInit() accepts a pointer to a KCMSCVisuals structure passed to it from the Calibrator Tool main program and is used by the module for initialization. Examples of initialization tasks the module might perform include:

- Initializing the RS-232 serial port, if a hardware colorimeter is used to take color measurements
- Starting its own GUI-based application
- Setting everything necessary to measure monitor data
- Doing nothing, if the measurements were previously stored in a file by the end user

Arguments

vis_data

Points to a KCMSCVisuals structure. For details on the format of this structure, see “KCMSCVisuals” on page 29 in Chapter 5, “Data Structures.”

Return Value

Returns 0 if successful; returns any other nonzero value if the hardware fails to initialize.

KCMSCMonMeasure ()

```
int  
KCMSCMonMeasure (KCMSCData *measured_data);
```

Purpose

KCMSCMonMeasure () accepts a pointer to a KCMSCData structure passed to it from the Calibrator Tool main program and performs the following functions:

- It performs the number of measurement sets specified by the value of the `size` field in the KCMSCData structure (alternately, it can be programmed to read all the measurements previously created).
- It uses the measurement values to fill in the appropriate fields in the KCMSCData structure.

Arguments

measured_data

Points to a KCMSCData structure. For details on the format of this structure, see “KCMSCData” on page 28 in Chapter 5, “Data Structures.”

Return Value

Returns 0 if successful; returns any other nonzero value if the collection of data was interrupted either by the user or by failure of the hardware to measure the data.

Glossary

channel

Refers to a red, green, or blue intensity. See *RGB values*.

color lookup table

A color map that establishes the relationship between pixel values and visible screen colors. A pixel value is an index into this table to arrive at an RGB value defining a displayable color.

colorimeter

A hardware device used to measure luminance values for calibrating a monitor.

depth

The number of planes (the number of bits per pixel value) for a window.

display

A set of one or more screens driven by an X Window System server. The `Display` structure contains all the information about a particular display and its screens. It is filled when a program calls an X library routine to connect to a windowing system. Also see *screen*.

gamma correction

Changing RGB values to ensure that the appropriate color appears on the screen.

KCMS framework

The KCMS API library routines available with the Kodak Color Management System portion of the Solaris 2.5 SDK that are used to manipulate color profiles.

OWconfig

A system configuration database file that maintains a list of the dynamically loadable objects (modules).

profile

A file that provides information to the KCMS framework on how to convert input color data to the appropriate color-corrected output color data. See *KCMS framework*.

RGB values

Red, green, and blue intensity values used to define a color.

screen

One or more screens can be associated with a single monitor, keyboard, and pointing device. A `Screen` structure is a member of a `Display` structure. See *display*.

visual

A pixel-value translation into colors that are displayed on the screen. X Window System visuals include `PseudoColor`, `DirectColor`, `GrayScale`, `StaticColor`, `TrueColor`, and `StaticGray`.

Index

A

adding `OWconfig` entries, 16 to 18
attribute=value pairs, 15, 16

C

calibration
 devices supported, 1
 platforms supporting, 1
Calibrator, 11
Calibrator Tool
 setup, 8
Calibrator Tool devices window, 11
Calibrator Tool responsibilities, 4
characteristics, 9
color correction data, 10
colorimeter, 4, 10, 33
colorsense, 7, 11
colorsense.c, 6
creating `OWconfig` entries, 14

D

devices, 1
DirectColor, 24

E

equivalent visuals, 24
error messages, 12
example programs, 7, 10, 25

F

format, 13
formatting guidelines, 14
freeing resources, 6

G

gamma, 2, 23
generic devices, 9
generic profiles, 2

I

initializing, 33
intensity levels, 25

K

KCMS (Kodak Color Management System), 2, 4
KCMS library, 1, 2, 25
`kcms_calibrate(1)`, 2, 3, 4

KCMSCData, 25, 34
KCMSCMonClose(), 5, 6, 32
KCMSCMonInit(), 5, 33
KCMSCMonMeasure(), 5, 34
KCMSCVisuals, 25, 33
KcsCalibrationData, 25

L

library X Toolkit, 6
library, KCMS, 1, 2
library, X Window System, 23
loadable module responsibilities, 4
luminance, 12

M

main program, 3
measurement modes, 24
measurements, recommended number
of, 25
messages, status, 12
modules, sample, 7, 10, 25
Motif, 6, 25

N

naming profiles, 9

O

OWconfig, 10, 13 to 21

P

platforms supporting calibration, 1, 3
precautions, 23
profiles
defined, 2
generic, 2
naming, 9
updating, 1, 4, 25
X Window System, 12, 25
program examples, 7, 10, 25

PseudoColor, 24

R

reading OWconfig, 13
remove, 19
removing OWconfig entries, 14, 19, 20
resources, freeing, 6
RGB values, 24, 25
run time, 13

S

sample modules, 7, 10, 25
SDK (software development kit), 4
serial port, 10, 33
shared object, 13
source, 10
source files, 6
StaticGray, 24
status, 5
status messages, 12
status values, 6
structures, allocating and deallocating, 25

T

TeleUSE, 6
templates, using, 10, 17
TrueColor, 24

U

updating profiles, 1, 4, 25
using script files, 16 to 20
using templates, 10, 17

V

visuals, 4, 6, 9, 23, 24
visuals not measured, 24

X

X Toolkit library, 6

X visuals, 6, 9

X Window System, 12, 25

X Window System visuals, 4, 23

X Window visuals, 9

X11 Window System, 6

XGetVisualInfo(3), 23

XSolarisGetVisualGamma(3), 23

XSolarisVisualGamma, 7, 10, 11

XSolarisVisualGamma.c, 6

Copyright 1995 Sun Microsystems Inc., 2550 Garcia Avenue, Mountain View, Californie 94043-1100 U.S.A.

Tous droits réservés. Ce produit ou document est protégé par un copyright et distribué avec des licences qui en restreignent l'utilisation, la copie, et la décompilation. Aucune partie de ce produit ou de sa documentation associée ne peuvent être reproduits sous aucune forme, par quelque moyen que ce soit sans l'autorisation préalable et écrite de Sun et de ses bailleurs de licence, s'il en a.

Des parties de ce produit pourront être dérivées du système UNIX[®], licencié par UNIX System Laboratories, Inc., filiale entièrement détenue par Novell, Inc., ainsi que par le système 4.3. de Berkeley, licencié par l'Université de Californie. Le logiciel détenu par des tiers, et qui comprend la technologie relative aux polices de caractères, est protégé par un copyright et licencié par des fournisseurs de Sun.

LEGENDE RELATIVE AUX DROITS RESTREINTS: l'utilisation, la duplication ou la divulgation par l'administration américaine sont soumises aux restrictions visées à l'alinéa (c)(1)(ii) de la clause relative aux droits des données techniques et aux logiciels informatiques du DFARS 252.227-7013 et FAR 52.227-19. Le produit décrit dans ce manuel peut être protégé par un ou plusieurs brevet(s) américain(s), étranger(s) ou par des demandes en cours d'enregistrement.

MARQUES

Sun, Sun Microsystems, le logo Sun, SunSoft, le logo SunSoft, Solaris, SunOS, OpenWindows, DeskSet, ONC, ONC+ et NFS sont des marques déposées ou enregistrées par Sun Microsystems, Inc. aux Etats-Unis et dans d'autres pays. UNIX est une marque enregistrée aux Etats-Unis et dans d'autres pays, et exclusivement licenciée par X/Open Company Ltd. OPEN LOOK est une marque enregistrée de Novell, Inc. PostScript et Display PostScript sont des marques d'Adobe Systems, Inc.

Toutes les marques SPARC sont des marques déposées ou enregistrées de SPARC International, Inc. aux Etats-Unis et dans d'autres pays. SPARCcenter, SPARCcluster, SPARCcompiler, SPARCdesign, SPARC811, SPARCengine, SPARCprinter, SPARCserver, SPARCstation, SPARCstorage, SPARCworks, microSPARC, microSPARC-II, et UltraSPARC sont exclusivement licenciées à Sun Microsystems, Inc. Les produits portant les marques sont basés sur une architecture développée par Sun Microsystems, Inc.

Les utilisateurs d'interfaces graphiques OPEN LOOK[®] et Sun[™] ont été développés par Sun Microsystems, Inc. pour ses utilisateurs et licenciés. Sun reconnaît les efforts de pionniers de Xerox pour la recherche et le développement du concept des interfaces d'utilisation visuelle ou graphique pour l'industrie de l'informatique. Sun détient une licence non exclusive de Xerox sur l'interface d'utilisation graphique, cette licence couvrant aussi les licenciés de Sun qui mettent en place OPEN LOOK GUIs et qui en outre se conforment aux licences écrites de Sun.

Le système X Window est un produit du X Consortium, Inc. Kodak est une marque de Eastman Kodak Company.

CETTE PUBLICATION EST FOURNIE "EN L'ETAT" SANS GARANTIE D'AUCUNE SORTE, NI EXPRESSE NI IMPLICITE, Y COMPRIS, ET SANS QUE CETTE LISTE NE SOIT LIMITATIVE, DES GARANTIES CONCERNANT LA VALEUR MARCHANDE, L'APTITUDE DES PRODUITS A REPENDRE A UNE UTILISATION PARTICULIERE OU LE FAIT QU'ILS NE SOIENT PAS CONTREFAISANTS DE PRODUITS DE TIERS.

CETTE PUBLICATION PEUT CONTENIR DES MENTIONS TECHNIQUES ERRONEES OU DES ERREURS TYPOGRAPHIQUES. DES CHANGEMENTS SONT PERIODIQUEMENT APPORTES AUX INFORMATIONS CONTENUES AUX PRESENTES. CES CHANGEMENTS SERONT INCORPORES AUX NOUVELLES EDITIONS DE LA PUBLICATION. SUN MICROSYSTEMS INC. PEUT REALISER DES AMELIORATIONS ET/OU DES CHANGEMENTS DANS LE(S) PRODUIT(S) ET/OU LE(S) PROGRAMME(S) DECRITS DANS CETTE PUBLICATION A TOUS MOMENTS.

