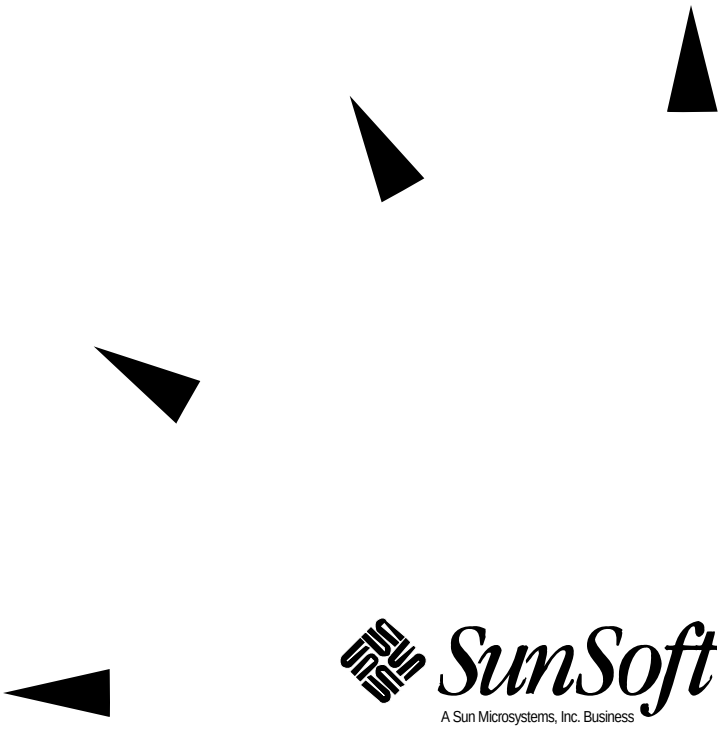
Solaris 2.5.1: Driver Developer Kit Introduction



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

SunSoft
A Sun Microsystems, Inc. Business

Please
Recycle

Contents

Preface

The *Solaris 2.5.1 Driver Developer Kit Introduction* gives an overview to the Solaris™ 2.5.1 Driver Developer Kit (DDK). It also:

- Tells you how the DDK fits into a Solaris development environment
- Lists new DDK features
- Tells you how to obtain hard copy documents, technical support, and training
- Describes each component of the DDK
- Lists and gives a brief description of DDK documentation
- Tells you where to find DDK documentation

Who Should Use This Book

If you are a driver developer interested in providing driver software for Solaris, you should read this book. Typical driver developers are independent hardware vendors (IHVs) or original equipment manufacturers (OEMs) who want their hardware products to operate in a Solaris environment.

DDK users include:

- IHVs and OEMs interested in writing DDI/DKI-compliant device drivers for hardware devices
- IHVs whose products include device drivers
- IHVs interested in writing device handlers for the OpenWindows™ server
- IHVs writing device pipelines for the XGL™ graphics library

- IHVs writing device handlers to port hardware devices to the XIL™ imaging library, and technology providers writing additional device-independent acceleration code for XIL operators
- IHVs writing FCode PROM programs for SBus cards
- IHVs writing color calibration modules to support new devices.

This manual assumes that you are familiar with the Solaris 2.5.1 distributed computing environment, and general UNIX™ device driver principles. If you are new to writing device drivers, see the first three chapters of the *Writing Device Drivers* manual.

Related Reading

For related information you may want to read:

- *Solaris 2.5 Introduction*
- *Writing Device Drivers*
- *x86: Installing Solaris Software*
- *SPARC: Installing Solaris Software*
- *Solaris 2.5.1 x86: Installation Notes*
- *Solaris 2.5.1 SPARC: Installation Notes*
- *Solaris 2.5.1 Driver Developer Kit Installation Guide*
- *Application Packaging Developer's Guide*
- *Solaris 2.5.1 Server Release Notes*

Ordering Hardcopy Documentation

The SunDocs^{SM} program makes available for individual sale product documentation from Sun Microsystems™ Computer Company and SunSoft™. For a list of documents and order information, see the catalog section of the SunExpress On The Internet site at http://www.sun.com/sunexpress.

What Typographic Changes Mean

The following table describes the typographic changes used in this book.

Table P-1 Typographic Conventions

Typeface or Symbol	Meaning	Example
`AaBbCc123`	The names of commands, files, and directories; on-screen computer output	Edit your `.login` file. Use `ls -a` to list all files. `machine_name% You have mail.`
`AaBbCc123`	What you type, contrasted with on-screen computer output	`machine_name% su` `Password:`
AaBbCc123	Command-line placeholder: replace with a real name or value	To delete a file, type `rm` *filename.*
AaBbCc123	Book titles, new words or terms, or words to be emphasized	Read Chapter 6 in *User's Guide.* These are called *class* options. You *must* be root to do this.

Shell Prompts in Command Examples

The following table shows the default system prompt and superuser prompt for the C shell, Bourne shell, and Korn shell.

Table P-2 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	`#`

Introduction 1

This chapter introduces you to the Solaris 2.5.1 Driver Developer Kit (DDK) and tells you how it fits into a Solaris development environment. It also lists features that are new to the DDK in the Solaris 2.5 and Solaris 2.5.1 release.

Driver Developer Kit Overview

The DDK helps you develop dynamically loadable device drivers and graphics device handlers for Solaris 2.5.1 by providing you with the necessary software tools, technical assistance, documentation, and technical training information. The DDK runs on all Solaris-supported platforms: SPARC™, x86, and PowerPC™.

Device drivers present the kernel with a consistent interface to diverse devices. Solaris supports a set of source-level interfaces between drivers and the kernel called the *device driver interface/driver-kernel interface* (DDI / DKI). Device drivers are dynamically loaded by the SunOS™ kernel. Device-driver code runs as kernel-level code.

Graphics device handlers (or *device handlers*) are software modules that add device-specific support for a Solaris VISUAL™ graphics foundation library. Each Solaris VISUAL foundation library defines a device porting interface, called a *graphics porting interface* (GPI). With the help of the DDK, you can write a device handler for a specific foundation library that is compliant with the GPI for that foundation library and is dynamically loaded by that foundation library. Device-handler code runs as user-level code.

1

For graphics devices, you generally need to write both a device driver and a graphics device handler for one or more VISUAL libraries.

The DDK also includes the FCode development tools you need to help you write OpenBoot™ PROM code for SBus cards.

Note – FCode development tools are not available for x86 systems.

DDK components are the software tools, libraries, server, and online documentation that make up the DDK. Except for the Solaris X Server, which is delivered on the Solaris CD-ROM disc, the following DDK components are provided on the DDK CD-ROM disc:

- Sample driver source code and driver development tools
- Device driver handler support for VISUAL for Solaris, which includes:
  - Solaris X Server
  - XGL graphics library
  - XIL imaging library
- Kodak Color Management System (KCMS)
- FCode development tools
- Online and hardcopy documentation

These DDK components are explained further in Chapter 2, “Components.”

How the DDK Fits Into a Solaris Development Environment

Solaris developers produce applications, drivers, and graphics handlers that are ready for the end-user Solaris runtime environments. A Solaris development environment may be constructed using the:

- Solaris runtime environments (available with any version of Solaris 2.5.1)
- Developer kits (the Solaris 2.5.1 Driver Developer Kit and Software Developer Kit), and
- Compilers (the ProCompilers and SPARCompilers C and C++).

The DDK contributes to this environment by providing the background information, requirements, and testing tools that you need to create software support for specific hardware devices in the Solaris runtime environments. The

DDK provides the information you need to create a wide array of hardware drivers. In some cases, the DDK reduces direct coding efforts by providing sample driver code as a starting point for driver development.

For more information on the Solaris 2.5.1 release, see the *Solaris 2.5 Introduction*.

New DDK Features

The following features are new to the Solaris 2.5 or 2.5.1 DDK:

- Driver Development Tools, including PCI Bus sample drivers
- Kodak Color Management System (KCMS)
- 3.x version of the Fcode Development Tools
- Sample XGL graphics

The sample XGL graphics handle code in the form of a template, which you can copy and easily modify for your graphics device.

New features for each component are described in Chapter 2, “Components.”

Technical Support

If you need help with the installation or use of the DDK, and you’re calling from the United States or Canada, call your Authorized Service Provider. Also refer to your Support Addendum card.

The DDK is also supported through the SunSoft Catalyst[SM] Developer’s Program. The Catalyst program offers a variety of technical support services to assist you in bringing your Solaris-ready software applications to market. For information about the Catalyst program, contact the Catalyst Information Center.

Sun Educational Services

In partnership with Sun Educational Services, SunSoft Authorized Education Centers provide training on Solaris Developer products. These courses are offered at many locations in the United States and throughout the world. For a current class and class-location list, call Sun Educational Services.

1

Components 2

The DDK CD-ROM disk contains software and online documentation. These are called the components of the DDK. This chapter describes each DDK component and tells you about the documentation associated with that component. If a component has features that are new in the Solaris 2.5 or 2.5.1 release, they are listed.

At the end of this chapter, documentation that supports software development, but is not closely associated with a software component of the DDK, is listed and described.

Sample Drivers and Driver Development Tools

The Sample Drivers and Driver Development Tools component of the DDK provides materials to assist you in developing device drivers for the Solaris environment. These materials include:

- Sample device-driver source code
- Driver development tools
- Documentation

These materials cover:

- How to use Solaris DDI/DKI interfaces to ensure forward compatibility with future Solaris releases
- How to implement drivers for different types of devices

2

New Features

The following new features are in this release of the DDK Sample Drivers and Driver Development Tools component:

- *Writing Device Drivers* manual—This book includes information on the new set of DDI/DKI common data access interfaces that allow device drivers to be written independent of *endianness* (byte order) and data ordering concerns. In addition, the manual includes descriptions of asynchronous read-and-write entry points for character driver I/O. There is extensive information and code examples for writing SCSI HBA drivers. Hardware and software issues relating to PCI bus architecture are also explained.
- Sample drivers for PCI-capable devices are included along with a white paper, entitled *Portable DDI-Compliant PCI Drivers*, that describes the new DDI/DKI data access interfaces in the context of specific PCI bus architecture issues. These issues include device and register mapping, bus address spaces, and data access attributes.
- *Solaris 2.4 x86 PCI Driver Writer's Supplement*—This feature enables PCI drivers, written with a subset of the Solaris 2.5 DDI/DKI (including the new portable driver interfaces), to be source-code compatible with the Solaris 2.4 release on an x86 hardware platform.
- x86 realmode driver development tools

Sample Device Drivers

The DDK includes a variety of sample device drivers. In most cases, these device drivers are complete functioning modules; in other cases, hardware details are omitted or generalized to provide a template from which you can generate a functioning driver.

You can use these sample device drivers, together with hardware-specific documentation and the *Writing Device Drivers* manual, as a starting point for developing DDI/DKI-compliant drivers.

The DDK provides sample device drivers for the following:

- Simple SCSI character target driver
- Simple SCSI block target driver
- Graphics (frame buffer) device driver for SPARC (cg6)
- Graphics (frame buffer) device driver for x86 (p9000)

- Graphics (frame buffer) device driver for PowerPC edition (p9100)
- Generic data link provider interface (DLPI) network device driver template
- Simple programmed I/O driver template
- Simple DMA character device driver template
- Simple RAM disk driver
- Portable PCI and SBus host bus adapter driver (QLogics ISP 1000, 1020)
- Portable PCI DLPI network device (AMD PCnet ethernet driver)
- Portable PCI frame buffer (Diamond Viper/Weitek P9000)
- PCI frame buffer (Weitek P9100) for PowerPC edition
- STREAMS input device driver (WACOM graphics tablet)

Driver Development Tools

The DDI compliance tool (DDICT) checks device driver C source code for non-DDI/DKI compliance. Non-DDI/DKI compliance is the use of features that are not part of the Solaris 2.x DDI/DKI. DDICT issues error and warning messages when it finds noncompliant code.

The x86 realmode development tools include documentation and realmode binaries that help you create bootable realmode drivers for x86 machines (see the *Realmode Drivers* white paper).

Documentation

The following driver development documentation is available in the DDK. For the locations of these documents, see Chapter 3, “Documentation.”

- *man Pages(9): DDI and DKI Overview*—Section 9 of the *Solaris 2.5 Reference Manual AnswerBook*. These manual pages document DDI/DKI source-level interfaces. Subsections cover:
  - 9E—required driver entry points
  - 9F—kernel functions that drivers may call
  - 9S—kernel structures used by drivers
- *Writing Device Drivers*—This manual describes device driver development for character-oriented devices, block-oriented devices, and SCSI host and target devices. It also covers general device driver topics.
- *STREAMS Programming Guide*—This manual discusses STREAMS-specific driver development topics.

- *FAQ*—The FAQ is an ASCII file containing frequently asked questions and the answers to those questions for DDI/DKI and device driver development topics.
- `ddict.1`—This is the `man` page for DDICT.
- *Data Link Provider Interface Specification*—This white paper specifies a STREAMS kernel-level instantiation of the ISO Data Link Service Definition (DIS 8886) and Logical Link Control (DIS 8802/2).
- *Multithreading and Real-Time in Solaris*—This white paper describes requirements, design, and implementation of real-time processes and multithreading in the Solaris 2.x environment.
- *Realmode Drivers*—This white paper describes the approach, interfaces, and requirements for providing realmode support in IHV x86 device drivers.
- *USCSI Ioctl: Application-level Access to SCSI Device Capabilities*—This white paper describes an unsupported, undocumented feature of Sun SCSI device drivers that allows users to issue SCSI requests directly to SCSI devices via an ioctl.
- *Portable DDI-Compliant PCI Drivers*—This white paper describes DDI/DKI interfaces that ensure kernel driver source-level portability across instruction set architectures and their supporting platforms. PCI drivers written to these interfaces can be recompiled and run on all PCI machines without any source code modification.
- *Solaris 2.4 x86 PCI Driver Writer's Supplement V 3.0*—This white paper describes how to use 2.5 interfaces in conjunction with a special compatibility module to create PCI driver source code that can be rebuilt to run on Solaris 2.5 machines as well as on x86 machines running the Solaris 2.4 release.
- *Hints for Device Driver Writers*—This white paper briefly describes the steps involved in writing a device driver. It also provides hints for debugging. This is a good starting document for all driver writers.

Solaris VISUAL

The Solaris VISUAL environment is the windows and graphics hardware developer environment.

It includes support for the:

- XGL graphics library
- XIL imaging library
- X11R5-based window-system server (Solaris X server)

This environment includes application-programming interfaces (APIs) for a wide variety of graphics functionality, including:

- 2-D and 3-D geometric graphics (the XGL graphics library)
- Imaging and digital video (the XIL imaging library)
- Stencil-paint style graphics (Display PostScript™ library)
- Basic pixel graphics (the X11 library)

You can build applications using application libraries from SunSoft™ and from third parties. These application libraries are built on a set of foundation libraries that are part of the Solaris SDK development environment—one for each major area of graphics functionality. Each foundation library defines a graphics porting interface, which is the interface for porting the library to hardware devices.

You can port your device to the Solaris VISUAL environment by using one or more of the Solaris GPIs. The DDK provides information enabling you to do this. A device might be a:

- Frame buffer
- Graphics accelerator
- Input device
- Frame grabber
- Image compression device

A device might also be a software component, for example, an optimized version of a compression algorithm or rendering pipeline. You can use the Solaris GPIs to support such “software devices.”

2

Documentation

For further information on the Solaris VISUAL environment, see the following:

- *Solaris VISUAL White Paper*—This white paper gives the overview and philosophy of the Solaris VISUAL environment which includes foundation libraries, X graphics libraries, and graphics porting interfaces. Includes detailed discussion on SGL (geometry), XIL (imaging and video), Display PostScript (stencil/paint), and X (Pixel-based).
- *Solaris VISUAL Overview for Driver Developers*—This white paper gives an introduction to the Solaris graphics porting interfaces (GPIs), the low-level interfaces for porting the Solaris graphics environment (Solaris VISUAL) to frame buffers, graphics accelerators, and other graphics devices. Components and porting options are covered, and a guide to other documents is provided.

Solaris XGL 3.2 Graphics Library

The XGL graphics library is a foundation graphics library that provides geometry graphics support for Solaris-based applications. The XGL library includes a device-level interface that defines the mapping of XGL code to the underlying hardware. If you write XGL loadable device pipelines (device handlers), which provide this mapping, you can build graphics devices that support any binary XGL application.

Because the Solaris environment provides mechanisms to dynamically load kernel device drivers and user process shared libraries, you can incorporate a new graphics accelerator into the Solaris environment by providing a dynamically loadable kernel device driver and an XGL device pipeline.

The XGL architecture provides open, well-defined interfaces that facilitate the task of implementing loadable device pipelines. The geometry-rendering interfaces are organized into a set of porting layers, which enable you to map underlying hardware capabilities to the XGL application-programming API. You can choose a layer for the device pipeline based on the functionality of the device and let XGL handle the rendering of functionality not accelerated by the device.

Documentation

For further information on the XGL graphics library, see the following documents:

- *XGL Device Pipeline Porting Guide*—This guide describes how to write an XGL graphics handler. It provides information on XGL internal interfaces and utilities, and on the mechanisms that enable the device code to work with the XGL device-independent code.
- *Getting Started Writing XGL Device Handlers*—This guide explains how to use the XGL skeleton pipeline to create an XGL graphics handler. The skeleton pipeline is a template for an XGL graphics handler from which hardware details have been omitted.
- *XGL Architecture Guide*—This guide provides information on the XGL architecture and presents details on the implementation of key aspects of that architecture. It also provides information on the design of the XGL loadable pipelines and describes XGL object-oriented internal design and coding conventions.
- *XGL Test Suite User's Guide*—This guide describes the installation and use of a set of graphics verification programs used to test the accuracy of a particular XGL implementation.
- *The XGL White Paper*—This white paper describes the purpose, structure, and features of functions in the XGL graphics library.

Solaris XIL 1.2 Imaging Library

The Solaris XIL imaging library is a foundation library for image processing and digital video applications. The XIL imaging library provides a single interface to the hardware with which it interacts. The XIL library has two public interfaces:

- ISV interface—documented in the Solaris 2.5.1 Software Developer Kit (SDK)
- Technology provider interface—provided with the Solaris 2.5.1 DDK

  The technology provider interface enables you to:
  - Port new hardware devices to the XIL imaging library.
  - Accelerate existing XIL functions.
  - Add video compressors and decompressors to the XIL imaging library.

Documentation

For further information on the XIL imaging library see the following documents:

- *XIL Device Porting and Extensibility Guide*—This guide describes the architecture and internal interfaces of the XIL library. It also describes the XIL library C++ classes and the mechanism for acceleration and porting of new hardware. If you are porting hardware to use the XIL imaging library or if you are writing device-independent acceleration code for XIL operations, you should read this guide.
- *XIL Test Suite User's Guide*—This guide describes how to run the Xilch test suite to verify the XIL imaging library. It also describes how to create new Xilch tests and benchmarking.
- *The XIL White Paper*—This white paper describes the purpose, structure, and features of functions in the XIL imaging library.

Solaris X Server 3.5

The Solaris X Server is based on the MIT X Consortium X11R5 server. The Solaris X Server provides basic window system and pixel graphics support. It also provides stencil-paint style graphics through the Display PostScript (DPS) extension.

The Solaris X Server provides a device-level GPI based on the standard Device Dependent X (DDX) interface and the XInput Extension for input devices. You can incorporate support for new graphics accelerators or input devices by providing dynamically loadable device handlers that implement the GPI for the Solaris X Server.

The Solaris X Server defines a basic GPI based on DDX. It also provides several utility porting layers that help you implement the GPI on various types of graphics accelerators and input devices. Some of these utility porting layers are common to the X11R5 server, such as:

- Color frame buffer (cfb)
- Monochrome frame buffer (mfb)
- Machine independent (mi)

Other layers, such as direct graphics access (DGA) and multiple plane group (MPG), enable you to use SunSoft features that provide enhanced graphics performance or support for advanced frame buffer architectures. The porting interface enables phased portion where you can use limited acceleration, then later optimize the device port to use the full graphics accelerator functionality. The Solaris X Server includes:

- Sample device handler source code for many devices
- Sample source code for some of the utility layers to aid debugging
- Server header files required to compile device handlers
- Sample directory hierarchy, including `Imakefiles`, to help build device handlers
- Support for x86, SPARC, and PowerPC architectures
- Support for transparent overlays
- DGA drawable interface
- `OWconfig` access method
- Debug server to aid in debugging your DDX handlers
- Support for visual gamma corrections

Documentation

For more information on the Solaris X Server, including detailed information on these new features, see the *X Server Device Developer's Guide*. This guide provides detailed information for writing device handlers for the Solaris X Server.

New Features

The following new features are in this release of the DDK Solaris X server:

- Enhanced debug server functionality that includes `dix` and `ddx` object modules for memory-leak detection utility programs
- A direct pixel access (DPA) interface that allows the window server to directly manipulate pixels in drawables that you control in your DDX handler
- A client DPA interface that supports direct access to windows in overlay planes.

Kodak Color Management System 1.0

The Kodak Color Management System (KCMS) software product is a color management solution that ensures color consistency from input devices to output devices. It is technology licensed from Eastman Kodak that was developed in conjunction with SunSoft. The KCMS product is fully integrated with SunOS™, is bundled with Solaris, and has SDK and DDK components.

The KCMS provides a flexible and powerful framework for developing color management technology. The DDK component of KCMS is the color management module (CMM) interface. It is a set of C++ classes that you can extend and override. You can add attributes to the current list, incorporate new color processing technology and create your own profile formats. The DDK also provides KCMS header files and sample code.

Documentation

For further information on the KCMS CMM interface, see the following documents:

- *KCMS CMM Developer's Guide*—This guide provides information and example programs to help you write your own CMM, create your own profile format, add attributes or tags to the ICC profile format, and override various class methods.
- *KCMS CMM Reference Manual*—This manual provides detailed information on all C++ classes available.

- *KCMS: Kodak Color Management System—Overview*—This white paper describes the different ways printers, scanners and desktop computers can represent color, the color differences that occur between devices and how the Kodak Color Management System corrects these differences.
- *KCMS: Kodak Color Management System*—This guide presents fundamental knowledge of color and can help you understand how color is used by imaging systems and how it is managed on the desktop computer. It helps you to understand some of the complex color problems that confront today's desktop computer systems and the solution that the Kodak Color Management System provides.

FCode Development Tools

Note – FCode Development Tools are not available for x86 systems.

FCode is a Forth-like language used to write to OpenBoot PROM code for SBus interface cards. When used with the other standard programming tools provided with the Solaris release, the FCode tokenizer, detokenizer, and fakeboot are important FCode development tools.

The FCode tokenizer converts FCode source into FCode binary that is suitable to reside on PROM. Use the FCode tokenizer if you want to design new SBus interface cards for Sun SPARC systems. The detokenizer converts the FCode executable file into a source file. For testing, fakeboot encloses an executable in a file suitable for loading into memory with the boot program.

New Feature

The 3.x version of the Fcode Development Tools 3.x, FCode tokenizer 3.x and Fcode detokenizer 3.x, are new in Solaris 2.5.1. They are not part of the Open Boot CPU PROM. They run under the operating system, reading input from one file and writing output to another file. The tokenizer converts FCode source code into FCode binary (byte codes) and the detokenizer does the reverse by converting FCode binary to FCode source. FCode is carried onboard devices in the form of PROM and is used at power-up to do system boot. During boot it provides the operating system with device configuration information and during development time it verifies a device’s functionality. Fcode 3.x is IEEE-1275 compliant.

2

Documentation

For more information on FCode development of OpenBoot PROM and SBus cards, see the following documentation:

- *Writing FCode 2.x Programs*—This manual describes how to write, debug, and test FCode programs for SPARC-based systems and interface card devices. It is a guide and reference for FCode application and SBus card developers on OpenBoot 2.x.
- *Writing FCode 3.x Programs*—This manual describes how to write, debug, and test FCode programs for SBus-based systems. It is a guide and reference for developers of FCode applications for SBus peripherals on OpenBoot 3.x. OpenBoot 3.x is compliant with IEEE Standard 1275-1994.

For additional information on FCode development of OpenBoot PROM and SBus cards, you may want to see the following documentation although it is not provided with the DDK:

- *OpenBoot 2.x Command Reference Manual*—This manual is available in the *Solaris 2.5 System Administrator AnswerBook* online documentation.
- *OpenBoot 3.x Command Reference Manual*—This manual is available in the *Solaris 2.5 System Administrator AnswerBook* online documentation.
- *OpenBoot 2.x Quick Reference Card*—This card provides a quick reference for OpenBoot commands. It is available in the *Solaris 2.5 System Administrator AnswerBook* online documentation.
- *OpenBoot 3.x Quick Reference Card*—This card provides a quick reference for OpenBoot commands. It is available in the *Solaris 2.5 System Administrator AnswerBook* online documentation.
- *IEEE Standard for a Chip and Module Interconnect Bus: SBus* (IEEE Standard 1496-1993)—To obtain a copy of this specification write, IEEE at:

  Institute of Electrical and Electronic Engineers, Inc.
  345 East 47th Street
  New York, NY 10017, USA

Documentation 3

This chapter provides a complete list of the DDK documentation for easy reference. It also tell you where you can find the DDK documentation.

The DDK documentation is provided in the following ways:

- Online through the AnswerBook product and in PostScript files
- In hardcopy in the DDK box
- In hardcopy through SunExpress

Documents Available Through the AnswerBook Product

The AnswerBook product provides a way for you to view the DDK documentation online. The DDK includes several sets of online documentation. These sets of documentation are called:

- *Solaris 2.5.1 Driver Developer AnswerBook*
- *Solaris 2.5 Supplemental Developer AnswerBook*
- *Solaris 2.5 Reference Manual AnswerBook*

For information on installing the DDK online documentation, see the *Solaris 2.5.1 Driver Developer Kit Installation Guide.*

3

Solaris 2.5.1 Driver Developer AnswerBook

This AnswerBook includes:

- *KCMS CMM Developer's Guide*
- *KCMS CMM Reference Manual*
- *X Server Device Developer's Guide*
- *Writing Device Drivers*
- *Writing FCode 2.x Programs*
- *Writing FCode 3.x Programs*
- *XGL Architecture Guide*
- *XGL Device Pipeline Porting Guide*
- *XGL Test Suite User's Guide*
- *Getting Started Writing XGL Device Handlers*
- *XIL Device Porting and Extensibility Guide*
- *XIL Test Suite User's Guide*
- *Solaris 2.5.1: Driver Developer Kit Introduction*

Solaris 2.5 Supplemental Developer AnswerBook

This AnswerBook includes:

- *Application Packaging Developer's Guide*
- *STREAMS Programming Guide*

Solaris 2.5 Reference Manual AnswerBook

This AnswerBook contains the *SunOS Reference Manual*. This reference manual includes:

- *OpenWindows Desktop Reference Manual*
- *Solaris X Window System Reference Manual*
- *man Pages(1): User Commands*
- *man Pages(1M): System Administration Commands*
- *man Pages(2): System Calls*
- *man Pages(3): Library Routines*
- *man Pages(4): File Formats*
- *man Pages(5): Headers, Tables and Macros*
- *man Pages(6): Demos*
- *man Pages(7): Device Network Interfaces*

- *man Pages(9): DDI and DKI Overview*
- *man Pages(9E): DDI and DKI Driver Entry Points*
- *man Pages(9F): DDI and DKI Kernel Functions*
- *man Pages(9S): DDI and DKI Data Structures*

Documents Available Through PostScript Files

These files are installed in `/opt/SUNWddk/ddk_2.5.1/doc`. They are also directly from the DDK CD-ROM disc in `/cdrom/ddk_2_5_1/SUNWsdkwp`.

Title	Filename
Multithreading and Real-Time in Solaris: Terms and Concepts	`OSMT-WP.PS`
Data Link Provider Interface Specification	`DLPISPEC.PS`
Solaris VISUAL White Paper	`SVIS-WP.PS`
Solaris VISUAL Overview for Driver Developers	`VISOVRVW.PS`
The XGL White Paper	`XGL-WP.PS`
The XIL White Paper	`XIL-WP.PS`
KCMS: Kodak Color Management System	`KCMS-WP.PS`
KCMS: Kodak Color Management System-Overview	`KCMS-WP-SOLARIS.PS`
Realmode Drivers	`REALMODE.PS`
USCSI Ioctl: Application-level Access to SCSI Device Capabilities	`USCSI.PS`
Portable DDI-Compliant PCI Drivers	`PCI.PS`
Solaris 2.4 x86 PCI Driver Writer's Guide, Version 3.0	`XPCI.PS`
Hints for Device Driver Writers	`DRIVER_HINTS.PS`

These files are on the CD-ROM disc in `/cdrom/ddk_2_5_1/Docs`. They are also in the AnswerBook online documentation.

Title	Filename
Application Packaging Developer's Guide	`APIG.PS`
Solaris 2.5.1 Driver Developer Kit Introduction	`DDK_Intro`
Writing FCode 2.x Programs	`FCOD.PS`
Writing FCode 3.x Programs	`FCODE.PS_3.x.ps`
KCMS CMM Developers Guide	`KCMSCMMDG.PS`
KCMS CMM Reference Manual	`KCMSCMMRM.PS`
X Server Device Developers Guide	`OWDDG.ps`
STREAMS Programming Guide	`STREAMS.PS`
Writing Device Drivers	`WDD.PS`
XGL Architecture Guide	`XGLARCH.PS`
Writing XGL Device Handlers	`XGLGETST.PS`
XGL Device Pipeline Porting Guide	`XGLPORTGU.ps`
XIL Device Porting and Extensibility Guide	`XILSYSPG.ps`
XIL Test Suite User's Guide	`XILTESTUG.ps`
XGL Test Suite User's Guide	`XGLTESTSTE.ps`

Documents Available in Hardcopy

- *Solaris 2.5.1 Driver Developer Kit Installation Guide* (only available in hardcopy)
- *Solaris 2.5.1: Driver Developer Kit Introduction* (available in hardcopy and through the AnswerBook product)

AnswerBook Documents Also Available in Hardcopy

To get hardcopy versions of the DDK AnswerBook online documentation, contact SunExpress or an authorized Sun reseller.

Suggested Reading Beyond the DDK

In addition to the DDK documentation, you can find related information in the following manuals:

- *OpenBoot 2.x Command Reference Manual*—This manual is available in the *Solaris 2.5 System Administrator AnswerBook* online documentation.
- *OpenBoot Quick Reference Card*—This card is available in the *Solaris 2.5 System Administrator AnswerBook* online documentation.
- *IEEE Standard for a Chip and Module Interconnect Bus: SBus* (IEEE Standard 1496-1993)—To obtain a copy of this specification, write IEEE at:

  Institute of Electrical and Electronic Engineers, Inc.
  345 East 47th Street
  New York, NY 10017, USA

- *PCI Local Bus Specification*, Revision 2.1, PCI-SIG, October 21, 1993.