Worldwide Developers Conference

Understanding Rhapsody Drivers

John Signa

Rhapsody Core OS Evangelist

Understanding Rhapsody Drivers

Dean Reece

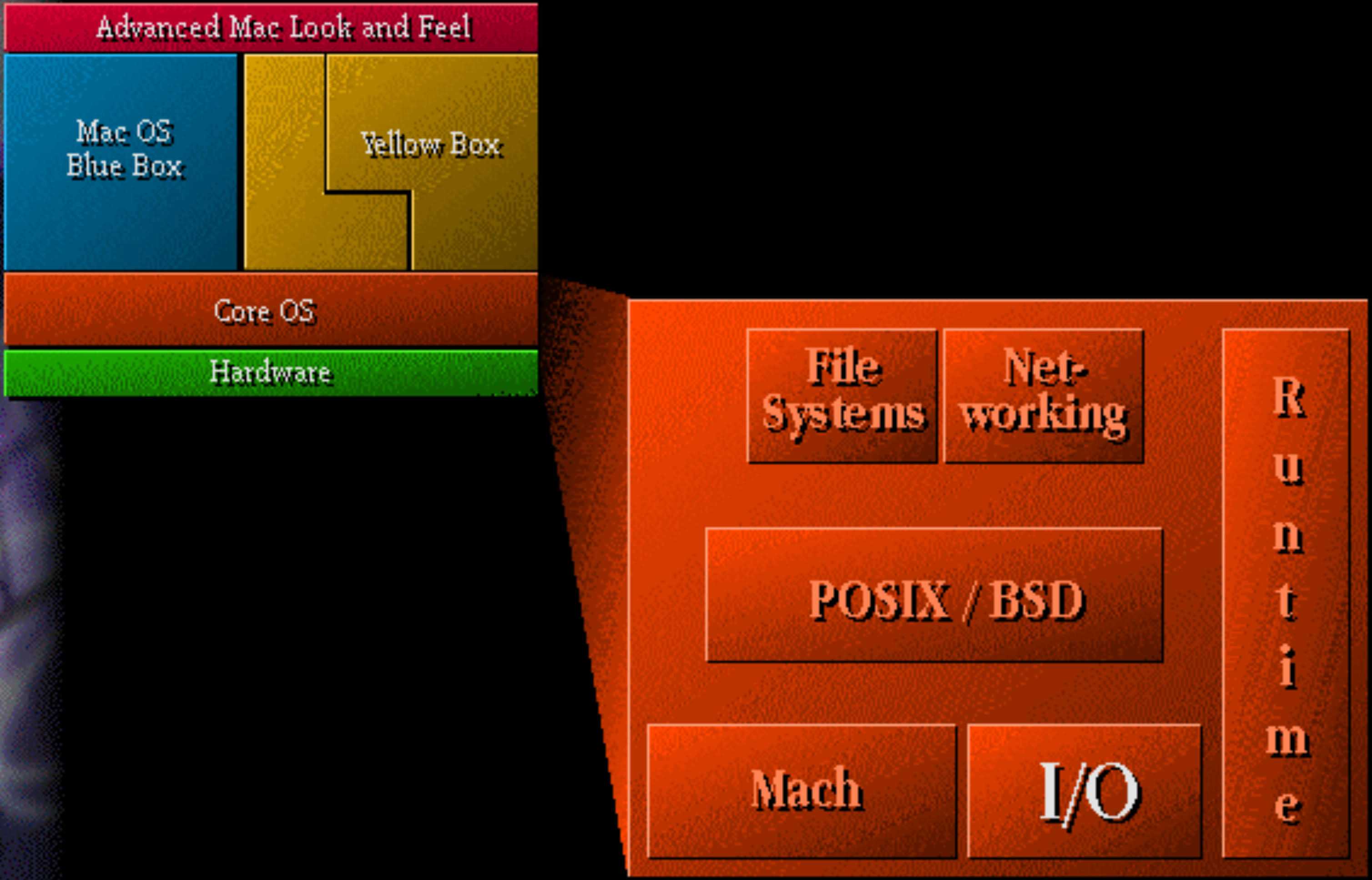
I/O Kit Manager

Introduction

Topics will include:

- Rhapsody I/O Overview
- I/O Kit Interactions
- Driver Model
- Wrap-Up

Rhapsody Core OS

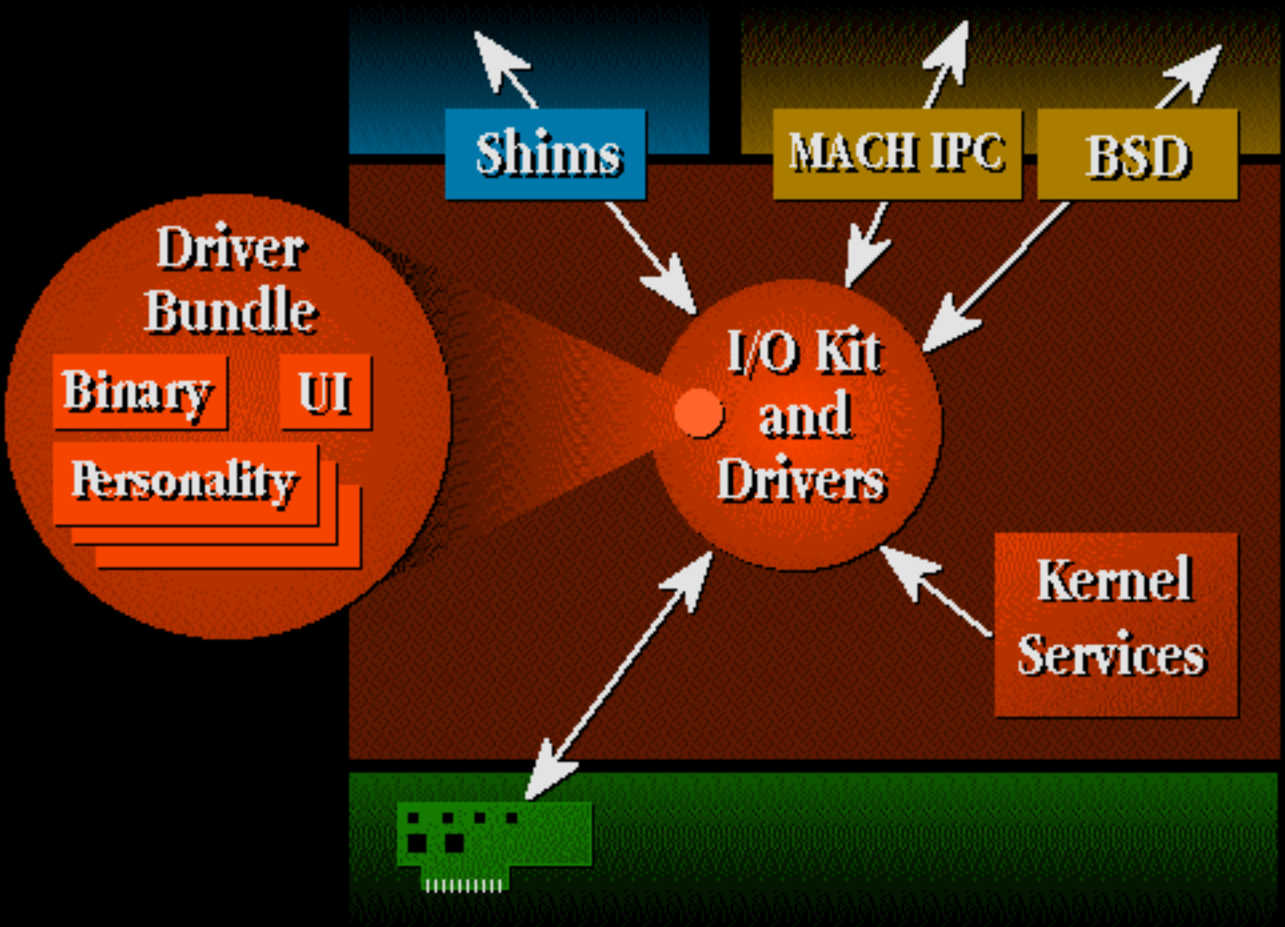


Drivers Run Under I/O Kit

I/O Kit is a framework designed to make writing full-featured drivers as easy as possible

- True Plug-n-Play
- Integrated Power Management
- Dynamic Device Management
- Very Modular and Extensible

How Drivers Fit In



I/O Kit Is Cross Platform

Write once / Run anywhere

- Available on Rhapsody and Rhapsody for Intel
- Tools allow “fat” drivers
- 99% code commonality

I/O Kit's Origins

Based heavily on DriverKit…
…but extended with key Apple technology

- I/O Kit Consists of
  - Runtime Environment
  - Documentation
  - Headers and Makefiles
  - Example Source

I/O Kit Is Object Oriented

- Brings OO advantages to drivers
  - Code reuse, time to market, testability
  - Proven capabilities
- C++ or ObjC syntax may be used
- Model is not object heavy
  - Used mostly for partitioning
  - 90% of a typical driver is C code
  - Typical drivers encounter few objects

Driver Compatibility

- New OS = New driver model
  - Not Compatible with DRVR
  - No support for DLPI drivers
  - UNIX/BSD drivers not fully supported
- Limited NDRV Support
  - Display Only
  - Unaccelerated
  - Single Display Adapter

I/O Kit and Booter

- Boot Sequence:
  - Booter collects config info via OF/BIOS
  - Booter uses primative I/O to load kernel and boot drivers
  - Booter passes control to kernel
  - Kernel can load additional drivers
- I/O Kit drivers are not used during boot
  - Boot via Open Firmware on Power Mac
  - Boot via BIOS on Intel

I/O Kit and Mach

- I/O Kit is rooted in Mach services
- Drivers are Mach Loadable Kernel Servers
- Mach messaging is used behind the scenes
- Mach memory and thread primitives visible
- I/O Kit provides wrappers for common behavior

I/O Kit and BSD

- Drivers may present BSD device nodes
- Some subsystems use this for User to Kernel communication
- I/O Kit classes generally hide this interface from driver writers

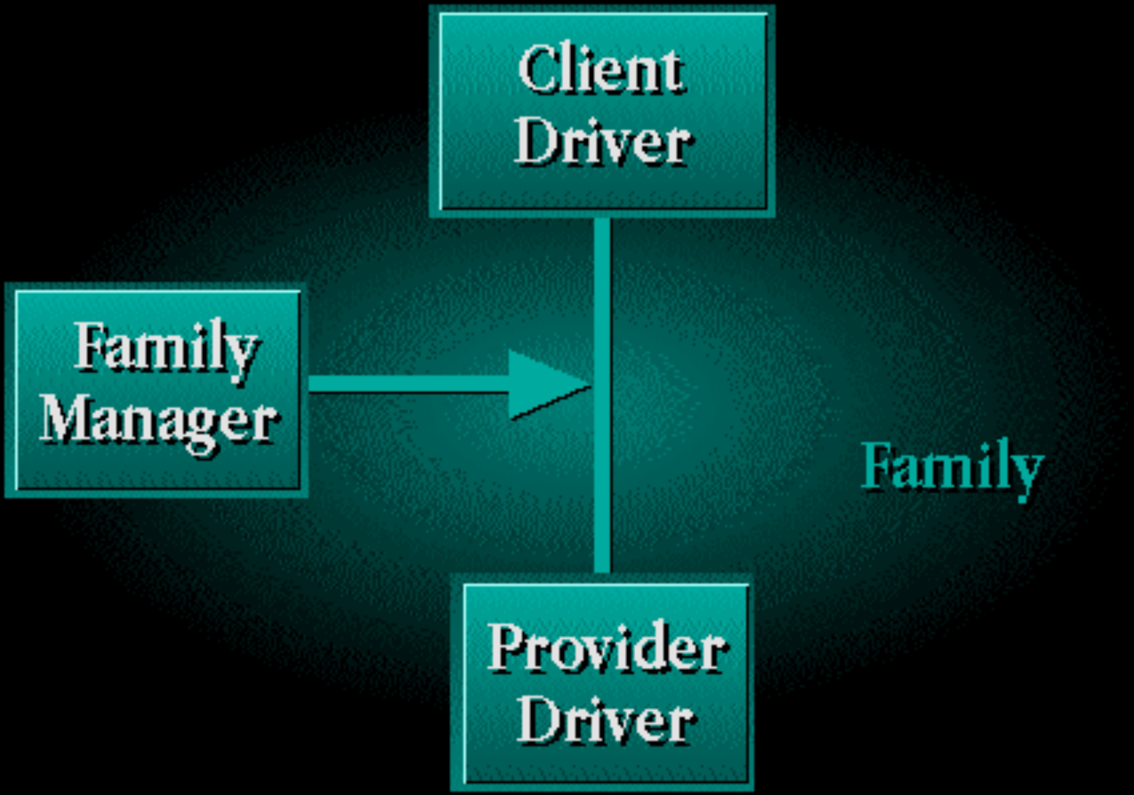
Basic Driver Environment

- Drivers and kernel share address space
- Drivers can be multi-threaded
- I/O Kit uses object runtime in kernel
- No protection between kernel and drivers

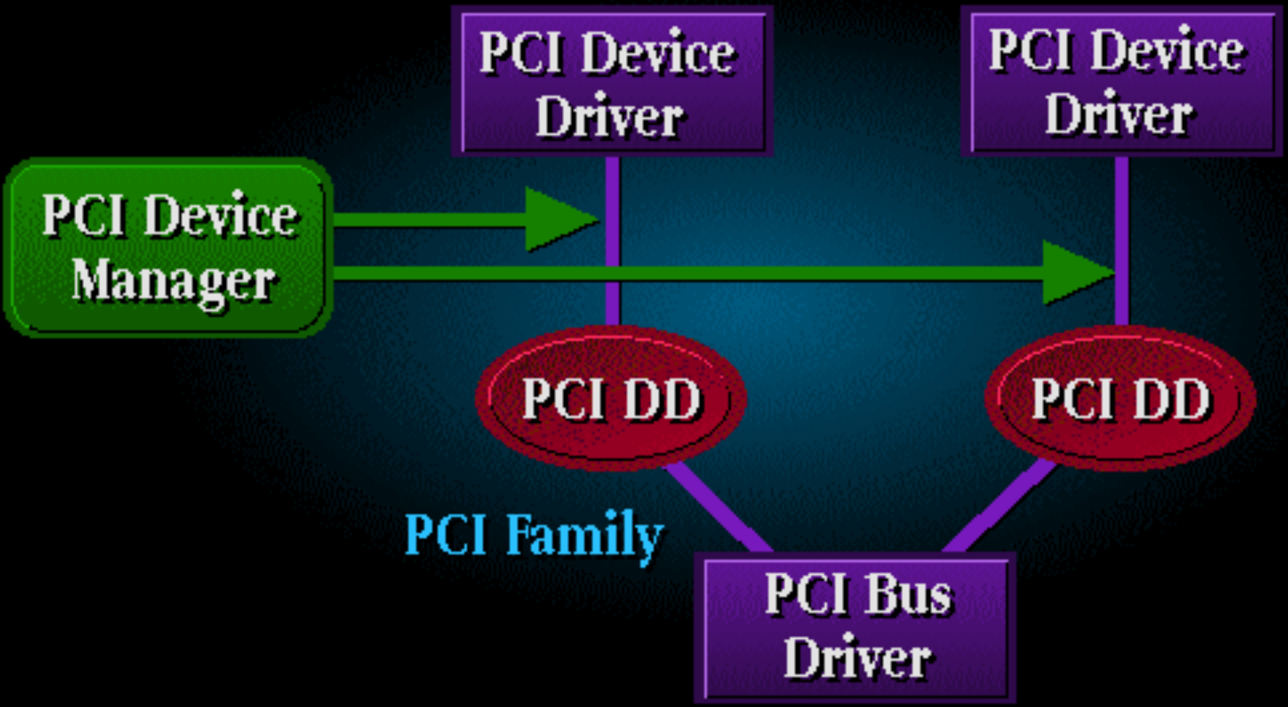
The Role of Objects in I/O Kit

- I/O Kit objects represent devices
  - One device = one object
- Objects categorized by Interfaces/protocols
- Objects communicate via
  - Method invocation
  - C function calls
  - Mach messaging

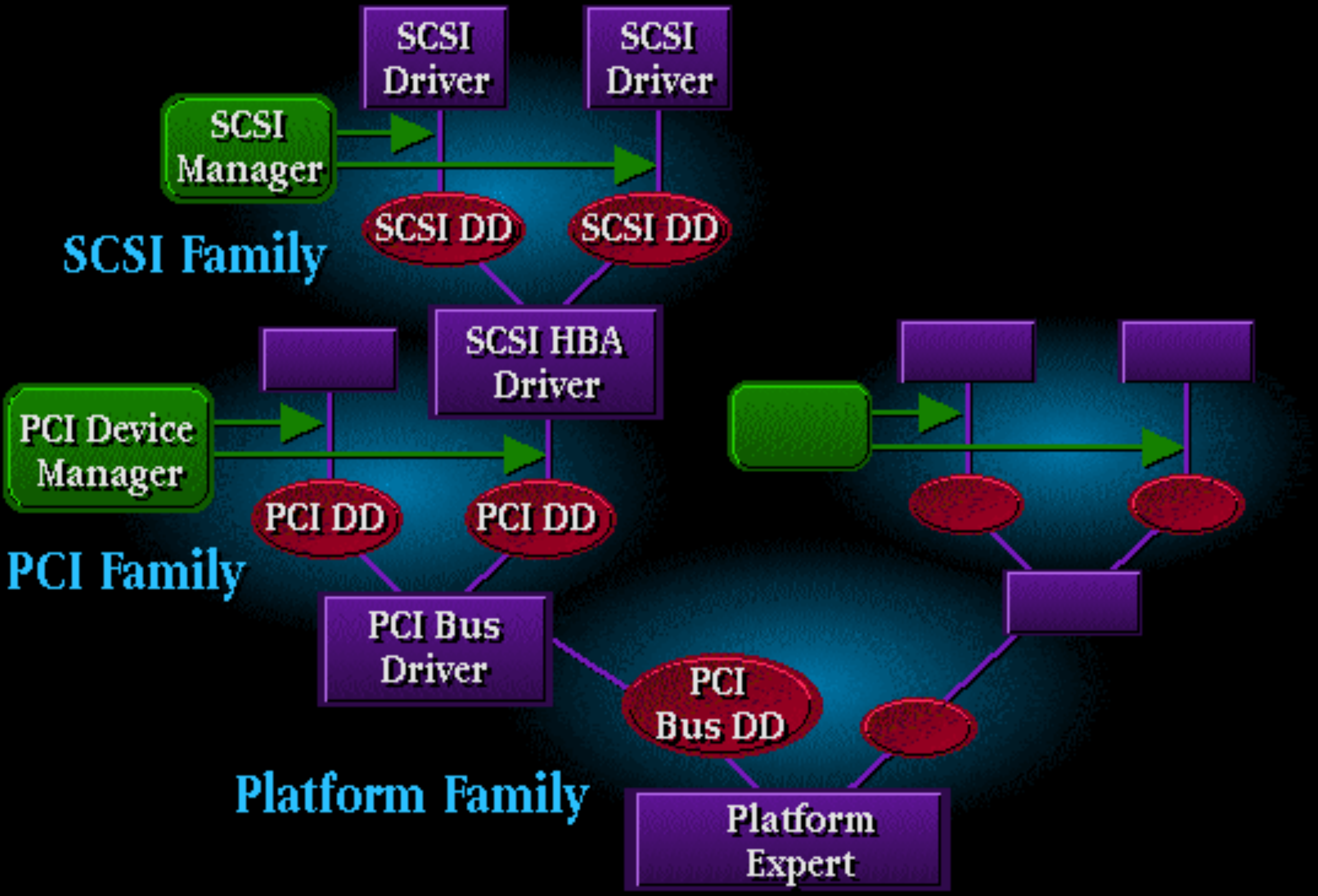
Object Relationships



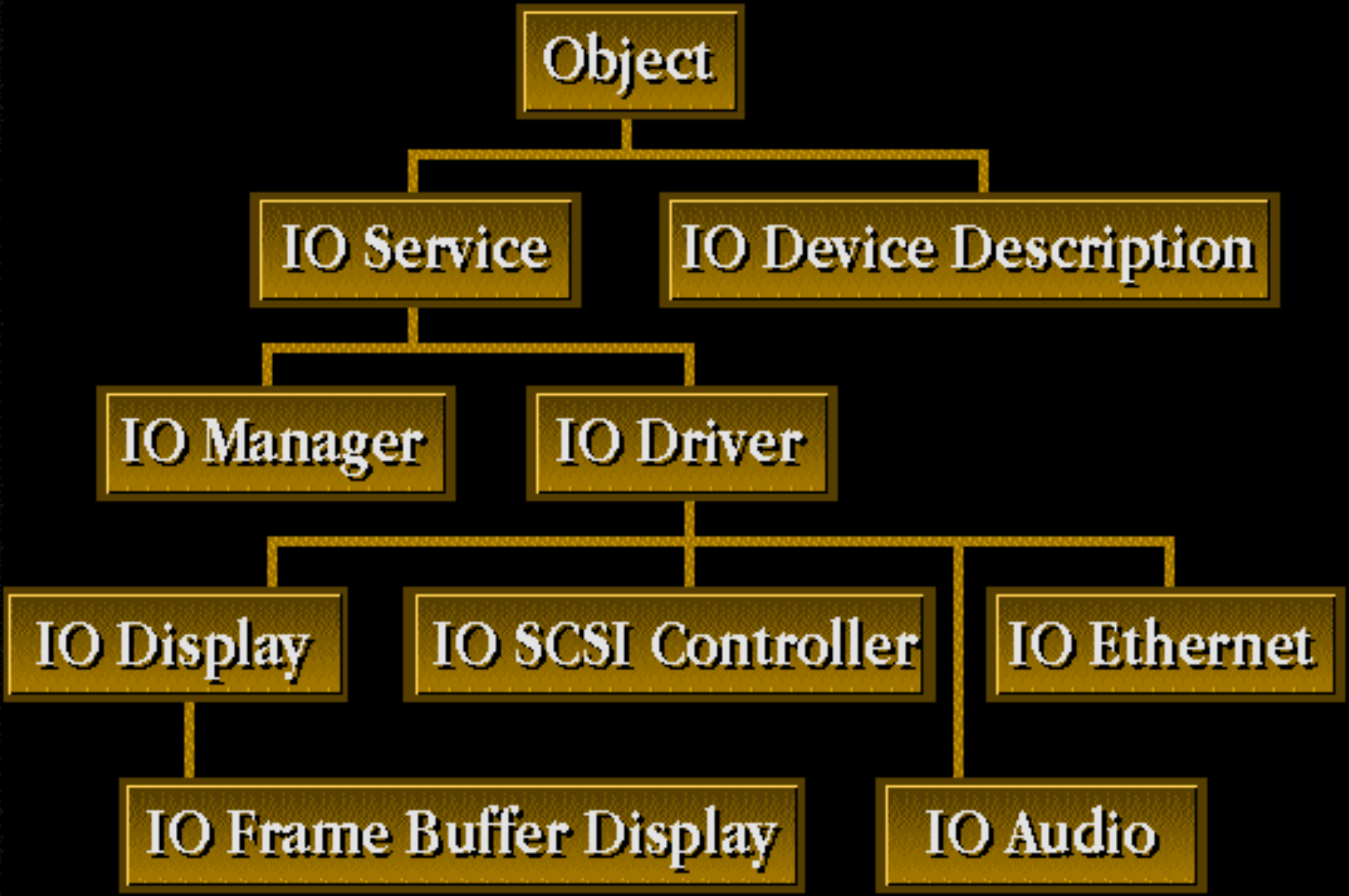
Members of the Family



The Family Tree



I/O Kit Class Hierarchy



Object

IO Service

IO Device Description

IO Manager

IO Driver

IO Display

IO SCSI Controller

IO Ethernet

IO Frame Buffer Display

IO Audio

Families in Rhapsody Premier

- SCSI, Block Storage
- Pointing Device, Keyboard
- Display, Audio
- Ethernet
- ADB, Serial
- PCI, (E)ISA, PCMCIA

Families in Rhapsody Unified

- USB
- FireWire
- Video capture
- What else?

Understanding Rhapsody Drivers

Martin Minow

Rhapsody I/O Group

Bringing Up a SCSI SIM

It's easier than you think

- About three weeks to port the Copland SCSI plug-in
  - A few days to learn Objective C and Driver Kit
  - Two weeks to port an existing, complex driver
  - Two days to install and debug the driver

Rules to Live By

Plan for Debugging

- Write a good set of logging macros
  - Log every method entrance and exit
- Write display functions for all of your variables
- “Get it right, then make it fast.”

SCSI on Rhapsody

Blue Box Compatibility

- Supported by a shim library
  - SCSI Manager 4.3
  - "Old" (Inside Mac IV) SCSI
- Reasonable-subset supported for
  - Backup utilities
  - Scanner and RIP plug-ins
- Cannot support all capabilities

SCSI on Rhapsody

Yellow Box Direct SCSI API

- Synchronous, using the existing NeXT libraries
  - Use fork for asynchrony
- Support of Blue-Box compatibility library?
  - Not precluded, but not recommended

SCSI on Rhapsody

Native Rhapsody API

- Being designed by developers
  - "Even as we speak"
- Optimized for high-performance
  - Reliability
  - RAID performance
  - Ultra, FiberChannel, and other future bus designs
- Will import some, but not all, Copland advances

SCSI on Rhapsody

Native Rhapsody API — Goals

- Reliability
- Performance
- Suitable for hardware developer needs
- Easy for Macintosh users to administer
- Suitable for driver developers
- Suitable for scanner plug-in and backup utilities

SCSI on Rhapsody

Native Rhapsody API — Non-Goals

- Highest possible Blue Box performance
- Driver and SIM compatibility with SCSI Manager 4.3

Rhapsody Driver Program

John Signa

Rhapsody Core OS
Evangelist

signa@apple.com

Rhapsody Driver Program

- Continuation of Copland Driver Program
- Events
  - Kitchens
  - DDK
    - Documentation
    - Sample code
    - Pre-Premier Release OS
- Enrollment
  - signa@apple.com

Additional Sessions

- Rhapsody Networking APIs and Services
  - Friday, 5:50 pm, Room A1
- Rhapsody Core OS Feedback Forum
  - Thursday, 11:10 am, Room J4
- Rhapsody Feedback
  - rhapsody-dev-feedback@apple.com

Q&A

Worldwide Developers Conference