3.3 Release Notes: Driver Kit

This file contains release notes for the Driver Kit. Items specific to each of the minor releases (Release 3.3, Release 3.2, and so on) are listed in separate sections, starting with the latest one.

Notes Specific to Release 3.3

New Features

The following new features have been added to the Driver Kit since Release 3.2:

- The reservation of resources (such as I/O port ranges, interrupts, and so on) is now enforced by the kernel. This means that if your driver requests a resource in its **Default.table** that has been previously reserved, your driver will not be probed. A message will appear in the console reporting that the resource could not be reserved.

- The **-setInterruptList:num:**, **-setChannelList:num:**, **-setPortRangeList:num:** and **-setMemoryRangeList:num:** methods can be used to change the resources reserved for your driver. The return type of these methods has changed from **void** to IOReturn, and the methods will return an error if the requested resources are not available.

- Writing PCI and PCMCIA drivers is now supported. See the documentation for the IOPCIDirectDevice and IOPCMCIADirectDevice categories for more information.

- The location for device drivers has changed. Drivers are now located in **/private/Drivers/***arch*, where *arch* is the architecture type for your computer (such as i386).

- Shared interrupts are now supported. See the ªShared Interruptsº section in Chapter 2, ªDesigning a Driver,º for more information.

- IOSCSIController supports indirect as well as direct device subclasses.

- **driverLoader** now executes the post-load binary for boot drivers.

- The Driver Kit documentation has been updated to reflect the latest API and document new features, such as shared interrupts and device auto detection. Chapters 1 through 4 have been revised and expanded to provide a fuller explanation of Driver Kit concepts, and Chapter 3 now contains information about writing specific types of drivers. A new appendix contains configuration file information and a reading list.

Known Problems

The Configure application has this restriction:

Reference: 46250

Problem: Drivers cannot use C-style comments in **Default.table**.

Description: The kernel ignores C-style comment delimiters (e.g. "/*" and "*/") in driver string tables.

Workaround: Don't use C-style comment delimiters in your driver's **Default.table**.

There are some problems with adding help to a driver bundle that make it different from adding help to other kinds of projects.

1. To add help to a driver, put a valid help TOC file and the valid help files referenced in the TOC inside the driver bundle's ***Drivername*.config/Language.lproj/Help** folder.

2. The help files referenced in the TOC inside the driver bundle's ***Drivername*.config/Language.lproj/Help** folder must have unique names. No two drivers present on the system can have the same helpfile names. Helpfile names should contain some string that is unique to that driver, such as a particular brand name and model number. Next's convention for naming these files is to use the Long Name from the associated strings file with the spaces replaced with underbars. (i.e., Intel_824X0_PCI_Host_Bridge.rtfd)

3. Don't use **compresshelp** or Project Builder's other help-building tools on the help files: If you index or compress the help files in the config bundle, they won't work.

4. You can't include links to other help files (such as an application's own help files) in a driver help file.

5. You can't use the help panel to find text in a driver's help file, unless you're already viewing the driver's help file. If you're viewing a driver help file, help-clicking doesn't

work.

6. If you install a driver while **Configure** is running, the driver's TOC entries appear multiply and unreliably in Configure's TOC. If you quit and restart, it works OK.

Incompatible Changes

The following incompatible changes have appeared since Release 3.2.

- You cannot initialize your driver with the **new** method, as in this statement:

```
[<object> new];
```

You must use the **alloc** and **init** methods:

```
[[<object> alloc] initWithDeviceDescription:<descr>];
```

Notes Specific to Release 3.2

New Features

The following new features have been added to the Driver Kit since Release 3.1.

- Writing SVGA drivers is now supported. See the IOSVGADisplay class specification for information.

- Writing Token Ring drivers is now supported. See the IOTokenRing class specification for information.

- Writing sound drivers is now supported. See the IOAudio class specification for information.

- Loading drivers at boot time is now supported.

- Standard makefiles now exist. Converting existing drivers to use the standard makefiles is described in the **README.rtf** file under **/NextLibrary/Documentation/NextDev/Examples/DriverKit/TestDriver**.

- Many classes have changed slightly. Some of these changes are described under ªIncompatible Changes,º below.

Known Problems

The Configure application has this restriction:

Reference: 37886

Problem: Accessory views for custom inspectors must not exceed 80 pixels in height.

Description: If you create an accessory view for a custom driver inspector used in Configure, your view must not exceed 80 pixels in height.

Workaround: You can check the size using Interface Builder's Size inspector. Also, make sure the view is not resizable, that is, none of the springs are sprung in the inspector in Interface Builder.

Incompatible Changes

The following incompatible changes have appeared since Release 3.1.

- You must recompile all Driver Kit drivers under 3.2Ð3.1 drivers will not load into 3.2 systems.

- IOEthernet drivers need several changes before you can recompile them. These changes are described below.

IOEthernet Changes

To convert a 3.1 Ethernet driver to a 3.2 Ethernet driver:

1. Add 2 instance variables to the subclass of IOEthernet

```
#import <bsd/net/etherdefs.h>

@interface MyEthernetDriver:IOEthernet
{
    enet_addr_t  myAddress;
    IONetwork   *network;

    /* Old
     * instance
     * variable
     * list */
}
```

2. Change the driver's **initFromDeviceDescription:** implementation to match the following:

```
- initFromDeviceDescription:(IODeviceDescription *)devDesc
{
    /* After this call, you should use [self name] to identify
     * the driver in any IOLog calls.*/
    if ([super initFromDeviceDescription:devDesc] == nil)
     return nil;

    /* Initialize instance variables */

    /* Perform any cold-start driver initialization, which should
     * include setting myAddress to the proper value.*/

    IOLog("MyEthernetAdaptor at port %x irq %d\n",ioBase,irq);

    /* Perform warm-start driver initialization. */
    [self resetAndEnable:NO];

    /* Inform my superclass of my address and cache the id of the
     * IONetwork object it creates for me. myAddress must be set
     * to the hardware address before this call is made. */
    network = [super attachToNetworkWithAddress:myAddress];

    return self;
}
```

3. Remove all references to IOEthernet's **netif** instance variable, which is no longer accessible. You should now use your own network instance variable and set it to the value returned by the **attachToNetworkWithAddress:** method.

4. Remove all references to IOEthernet's **ethernetAddress** instance variable, which is no longer accessible. You should now use your own **myAddress** instance variable instead.

5. Implement **resetAndEnable:** to perform warm-start initialization for your driver. For 3.2, you'll need to call **setRunning:** inside your **resetAndEnable:** method.

6. See the IOEthernet documentation for a description of the new API for implementing multicast addresses and promiscuous mode.

Notes Specific to Release 3.1

New Features

Release 3.1 is the first release to contain the Driver Kit. Using the Driver Kit, you can write drivers for Intel-based computers running NeXTSTEP.

Note: Driver Kit isn't supported for 680x0-based systems.

Documentation for the Driver Kit is available in **/NextLibrary/Documentation/NextDev/OperatingSystem/Part3_DriverKit**. A simple example of building, loading, and debugging a Driver Kit driver is under **/NextLibrary/Documentation/NextDev/Examples/DriverKit**.

The **Concepts** directory under **OperatingSystem/Part3_DriverKit** contains information about the Driver Kit as a whole (Chapter 1), as well as some preliminary details on driver writing issues (Chapter 2). It also contains some information on building and configuring drivers (Chapter 4). Chapter 3 is missing; in the future, it'll contain information about writing specific types of drivers, such as drivers for ethernet cards and drivers for graphics cards.

The **Reference** directory has documentation for most of the core Driver Kit classes, protocols, and functions. The classes and protocols that aren't documented (many of which currently have header files under **bsd/dev** and **bsd/if**) are likely to have API changes (for example, class and method names are likely to change).