Legal Notices

The information in this document is subject to change without notice.

Hewlett-Packard makes no warranty of any kind with regard to this manual, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose. Hewlett-Packard shall not be held liable for errors contained herein or direct, indirect, special, incidental or consequential damages in connection with the furnishing, performance, or use of this material.

Warranty

A copy of the specific warranty terms applicable to your Hewlett-Packard product and replacement parts can be obtained from your local Sales and Service Office.

U.S. Government License

Proprietary computer software. Valid license from HP required for possession, use or copying. Consistent with FAR 12.211 and 12.212, Commercial Computer Software, Computer Software Documentation, and Technical Data for Commercial Items are licensed to the U.S. Government under vendor’s standard commercial license.

Trademark Notices

UNIX® is a registered trademark in the United States and other countries, licensed exclusively through The Open Group.
<table>
<thead>
<tr>
<th>Contents</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>HP-UX DDK FAQ</td>
<td></td>
</tr>
<tr>
<td>1 About the FAQ</td>
<td>6</td>
</tr>
<tr>
<td>2 General Information</td>
<td>7</td>
</tr>
<tr>
<td>3 HP-UX 11.0</td>
<td>8</td>
</tr>
<tr>
<td>4 HP-UX 11i v1</td>
<td>8</td>
</tr>
<tr>
<td>5 HP-UX 11i v2</td>
<td>9</td>
</tr>
<tr>
<td>6 Kernel Configuration</td>
<td>9</td>
</tr>
<tr>
<td>7 Networking Drivers</td>
<td>12</td>
</tr>
<tr>
<td>8 Mass Storage Drivers</td>
<td>15</td>
</tr>
<tr>
<td>9 DLKM</td>
<td>15</td>
</tr>
<tr>
<td>10 WSIO</td>
<td>16</td>
</tr>
<tr>
<td>11 Kernel Services</td>
<td>17</td>
</tr>
<tr>
<td>12 Software Depot</td>
<td>18</td>
</tr>
<tr>
<td>13 The DDICT Tool</td>
<td>20</td>
</tr>
<tr>
<td>14 KWDB</td>
<td>20</td>
</tr>
<tr>
<td>15 Kernel Tunables</td>
<td>21</td>
</tr>
</tbody>
</table>
1 About the FAQ

1.1 What is this FAQ about?
Driver developers are faced with a lot of questions when they start developing drivers in an HP-UX environment. This FAQ attempts to address some of the problems commonly faced by third-party driver developers.

1.2 Is there a glossary where I can look up terms and acronyms?
Yes. Refer to the glossary in the Driver Development Guide (DDG) for the HP-UX operating environment you are writing a driver for. The driver development guides for different HP-UX versions are available at the Developer & Solution Partner Program (DSPP) portal at:
www.hp.com/go/hpux_ddk

1.3 Can I start developing an HP-UX driver after reading this FAQ?
No. This FAQ is not a replacement for the DDG and the Device Driver Reference (DDR) that HP provides. We highly recommend that you refer to the DDG and the DDR for a conceptual understanding of the HP-UX I/O subsystem and a detailed discussion of the issues involved in driver development.

1.4 What HP-UX versions does this FAQ cover?
This FAQ covers some of the driver development issues on HP-UX 11.0, HP-UX 11i v1 and HP-UX 11i v2.

1.5 How can I contribute to this FAQ?
We encourage HP-UX driver developers to contribute to this FAQ and enhance its usability by sending us your Feedback via the Web or send an email to:
hphp.support@hp.com

1.6 Where can I get this FAQ?
This FAQ is available at the Developer & Solution Partner Program (DSPP) portal at:
www.hp.com/go/hpux_ddk
2 General Information

2.1 What does the HP-UX DDK include?
HP releases a Driver Development Kit (DDK) with major HP-UX releases. Currently, DDKs for the following HP-UX versions are supported: HP-UX 11.0, HP-UX 11i v1, and HP-UX 11i v2. The HP-UX DDK provides:

- **Documents** — DDG, DDR, Porting Guides, Migration Guides and other White Papers.
- **Sample Drivers** — Sample implementations of fully functional networking drivers, a SCSI interface driver, and other simple DLKM drivers.
- **Tools** — Debuggers, Performance Utilities, Driver Test Tools, Driver Compliance Tools etc.
- **Build Environment** — Makefiles, Build Tools, etc.

2.2 Where can I get the HP-UX DDK?
The HP-UX DDK can be downloaded from the Web at:
http://www.hp.com/go/hpux_ddk

2.3 What compiler can I use to build a driver?
An HP ANSI C compiler is required to build your driver. Product numbers for HP ANSI C compilers are:

- B3901BA (for HP 9000 Servers)
- B3899BA (for HP 9000 Workstations)
- B6237AA (Documentation set)

For additional information, see the HP C/ANSI C Developers bundle page at:
http://www.hp.com/go/hpc

2.4 I hear people discussing HP-UX 11i and HP-UX 11.11 in the same context. Are they the same thing? What is the OS version mapping?
- HP-UX 11i is same as HP-UX 11.11, and HP-UX 11i v1
- HP-UX 11i v1.6 is same as HP-UX 11.22
- HP-UX 11i v2 is same as HP-UX 11.23

2.5 How do I know if an I/O card is installed correctly and recognized by the system?
Issuing the `ioscan(1M)` command results in a list of usable I/O system devices. By default, for each hardware module on the system, `ioscan` displays the hardware path to the hardware module, the class of the hardware module, and a brief description. If your device is not found by `ioscan`, it is not identified by the system. The `ioscan` command also indicates whether a card is claimed or not. Use the `lanscan(1M)` command to identify LAN devices.
3 HP-UX 11.0

3.1 How do I compile and build drivers on HP-UX?

Detailed instructions on building and compiling a driver are given in Chapter 6, “Installing Your Driver” of the HP-UX 11.0 DDG and the HP-UX 11.11 DDG.

3.2 Can I use the lanadmin, linkloop, and lanscan commands with my driver (a third party driver)?

In HP-UX 11.0, HP-UX 11i v1, and HP-UX 11i v1.6 the LAN commands, namely lanadmin and linkloop, work only with native HP drivers. Independent Hardware Vendors (IHV) are required to write their own shared libraries to provide similar functionality. lanscan works with IHV drivers.

3.3 The lanscan command does not list any NICs, but I can see them in the ioscan output. What’s wrong?

When no argument is specified to lanscan, it assumes that the kernel is located at /stand/vmunix. If your system is booted with a test kernel to see a listing of NIC cards, for example /stand/vmunix_test, pass the kernel path to lanscan. Run kmpath to identify the kernel the host is booted with.

3.4 I passed the boot kernel path to lanscan, but I still do not see my NIC listed in the lanscan output. Why?

Make sure certain fields of the hwift structure are initialized in your driver’s init() routine. Also, make sure the hwift structure is attached to the hwift list. Do this by calling hw_ift_attach using a pointer to the hw_ift_t structure associated with the device instance as an argument.

3.5 Can I use SAM to configure my driver (a third party driver)?

In HP-UX 11.0, SAM does not support configuring IHV network drivers.

4 HP-UX 11i v1

4.1 Can I use SAM to configure my driver (a third party driver)?

From HP-UX 11i v1 onward, SAM supports configuration of IHV network drivers. Refer to Chapter 7, “Creating Networking Device Drivers” in the 11i v1 DDG.

4.2 Can I use deprecated WSIO services in an HP-UX 11i v1 driver?

WSIO has deprecated a set of services in HP-UX 11i v1, for example: isrlink() and wsio_map(). New services are provided with enhanced functionality. Refer to the HP-UX 11i v1 Driver Development Reference Manual for a list of deprecated services along with the new services that offer similar functionality. HP strongly recommends that the new functionality be used in any drivers you develop, because the deprecated services will be obsolete in future releases.
5 HP-UX 11i v2

5.1 How do I compile and build drivers on HP-UX 11i v2?
The driver build process changed for HP-UX 11i v2. Archived driver libraries are not supported with HP-UX 11i v2. Refer to HP-UX 11i v2 DDG Chapter 6, “Kernel Module Configuration” for further details.

5.2 Can I use lanadmin, linkloop, and lanscan to work with my driver (a third-party driver)?
Starting with HP-UX 11i v2, the lanadmin and linkloop, and lanscan commands work with both Native DLPI and non-Native IHV network drivers.

5.3 Can I use SAM to configure my driver (a third-party driver)?
From HP-UX 11i v1 onward, SAM supports configuration of IHV network drivers. Refer to HP-UX 11i v2 DDG Chapter 14, “SAM Support for LAN Drivers” for additional details.

5.4 How can I write an EFI driver for IPF platforms running HP-UX 11i v2?
Refer to the EFI Driver Writing Guide available at Intel’s developer website:
http://developer.intel.com/technology/efi/dg.htm

6 Kernel Configuration

6.1 Why do I receive the object file format error message when a DLKM module fails to load?
The driver install function in the modmeta file of the DLKM driver does not specify static. This is a common problem.
If a DLKM driver contains a modmeta file similar to the following:
initfunc driver_install mydrivername_install
It should be modified to include static, as follows:
initfunc driver_install mydrivername_install static
With this construct in the driver modmeta file, the driver install routine will not be called unless the driver is statically linked. When a driver is dynamically loaded, the driver load routine is called instead.
6.2 How do I completely remove a module from a system?

You cannot completely remove a module from a system by simply removing it from the /usr/conf/mod directory. First unload the module, then uninstall the module, and finally remove the module from the /usr/conf/mod directory as follows:

1. Unload the module:
   
   ```
   # kcmodule -s <mydrivername>=unused
   ```

2. Uninstall the module:
   
   ```
   # kcmodule -s <mydrivername>=uninstall
   ```

3. Remove the module from the /usr/conf/mod file:
   
   ```
   # rm -f /usr/conf/mod/<mydrivername>
   ```

6.3 What are the different methods for linking a static module to the kernel?

**Method #1**

1. Copy the module into the /usr/conf/mod directory
2. Run the `# kcmodule -s <mydriver_name>=static` command
3. reboot

**Method #2**

1. Copy the module into /usr/conf/mod directory
2. Add the `<mydriver_name>` module static construct to the end of module listings in /stand/system file
3. Run the `# kconfig -i /stand/system` command
4. reboot

**Method #3**

1. Copy the module into /usr/conf/mod directory
2. Add the `<mydriver_name>` module static construct to the end of module listings in /stand/system file
3. Run the `# mk_kernel -i /stand/system` command
4. reboot

6.4 What's the difference between the unused state and the uninstall state?

To unload the module but still leave the module in the system configuration, run the following command:

```
# kcmodule -s <mydrivernname>=unused
```

To completely remove the module from the system configuration, run the following command:

```
# kcmodule -s <mydrivernname>=uninstall
```

Note: The file will remain in the /usr/conf/mod directory.

6.5 What should the version of a new module be?

The initial release of a new module should be version 1.0.0
6.6 Can I have multiple versions of a module configured on a system simultaneously?
No. At any time, only one version of a module can be configured on a system.

6.7 What are the build tools for KC and where can I get them?
The following build tools are required to develop drivers for HP-UX 11i v2:
- modmeta
- convert_master
- kmsecgen
- modlink
Refer to /opt/ddk/11.23/BE for these tools and their manpages.

6.8 What are the guidelines for changing the version number of a module?
A module version number is represented as: major_num . minor_num . revision_num
- major_num should be changed when a major change is made to the module
- minor_num should be changed when a new feature is added to the module
- revision_num should be changed when a bug is fixed in the module

6.9 Do I have to change the module version number every time there is a change in the module?
Yes. KC tools look at the module version when a module is picked up. Even if you copy the version number to the /usr/conf/mod file, KC will not pick up the most recent version of the module unless the most recent version of the module has a different version number.

6.10 Why does a DLKM module fail to load with the ld: Unsatisfied symbol foo_conf_data in the /stand/mod/foo file?
DLKM modules marked for either loaded or auto states must declare an extern symbol for the mod_conf_data data structure. For example, the module foo must declare extern struct mod_conf_data foo_conf_data in its source file. If a module does not declare the extern symbol in its object file, DLKM load returns with the error message ld: Unsatisfied symbol foo_conf_data in the /stand/mod/foo file.

6.11 How can I declare module dependencies in the modmeta structure?
Use the dependency attribute in the module metadata file to list the module dependencies.

6.12 How do I declare the states attribute in a module metadata structure so a module will be automatically loaded?
Declare auto in the states attribute in the module metadata structure, then enter the following command:
# kcmodule -s <mydrivernname>=auto
6.13 I receive the ld: (Warning) Ignoring IA_64.unwind_hdr section in file /usr/conf/mod/<module name> warning message when I try to statically link a DLKM module. Is this a problem?
No. This is a warning that occurs when a module that supports dynamic loading is statically linked. Ignore this warning message.

6.14 Where is the mod_systemfile located?
The mod_systemfile is a Software Distributor utility. It is located in the /usr/lbin/sw/control_utils directory.

6.15 Is the <modname>_conf_data structure automatically generated in a DLKM module?
Yes. KC automatically generates the <modname>_conf_data structure in a DLKM module. However, the module must also declare the extern struct mod_conf_data foo_conf_data symbol in its source file to avoid module linking errors.

6.16 How can a DLKM module export symbols to other DLKM modules?
Declare the symbols (interfaces) to be exported as globals in the module source, then define the export attribute in the module metadata file. List the symbols you want to export under the export attribute. To access these symbols (interfaces), other modules will need to define a dependency attribute in their respective module metadata structures and list the source module under the dependency attribute.

7 Networking Drivers

7.1 What is DLPI?
DLPI is an acronym for Data Link Provider Interface. It is an OpenGroup standard.

7.2 Where can I get the DLPI standard on the Web?
http://www.opengroup.org/onlinepubs/9638599/toc.htm

7.3 What is a DLS user?
A DLS user is any entity that sends messages to a DLPI provider. It can be an IP entity, or any layer 3 protocol entity in the OSI model. It can also be a custom-made user level program doing a series of putmsg and getmsg system calls.

7.4 What is PPA?
PPA is an acronym for Physical Point of Attachment. The PPA is the instance number of a NIC that is sending and receiving data. When you enter the lanscan command, each instance of the NIC appears with its name followed by its PPA (no space between them), in the Net-Interface column of the lanscan output.
7.5 Are there any HP-UX specific DLPI extensions?

Yes. HP-UX has some OS specific extensions to the DLPI interface that enhance functionality. Some of these extensions are for enlisting LAN devices, IP fast path, enabling or disabling loopback, and so on. The dlpi_ext.h header file is exported with the DLPI extensions.

7.6 What is IP fastpath?

Fastpath was designed as a low overhead alternative to the unitdata path for DLPI users that do not need to transmit data to the driver through the DL_UNITDATA_REQ and DL_UNITDATA_IND primitives. Currently, fastpath is only supported by the TCP/IP protocol stack.

An overview of how fastpath works follows:

1. A DLPI user sends the driver an IOCTL request setting up a fastpath on the stream.
2. The driver builds a link header and returns it to the DLPI user.
3. The DLPI user keeps the link header as a template and uses it to prepend a copy of the link header bits to the outbound data, before sending the packet to the driver.
4. The driver passes the packet to the device without going through the process of building a link header.
5. Similarly, for an inbound packet on the fastpath, the driver strips off the link header and passes the packet to the DLPI user without building and prepending the DL_UNITDATA_IND primitive to the data.

7.7 How do I set up fastpath? What does the driver expect from the DLPI user and what should the DLPI user return?

A DLPI user sends the driver an M_IOCTL message with the IOCTL command set to DLPI_IOC_HDR_INFO. The M_IOCTL mblk is linked with an M_PROTO mblk containing a DL_UNITDATA_REQ primitive. The driver will build an M_DATA mblk containing the link header for the specific interface. Then the driver links the M_PROTO mblk with the M_DATA mblk and sends them back to the DLPI user. For an example of this implementation, refer to the enet sample driver provided with the DDK.

7.8 My driver receives an M_IOCTL message with the IOCTL command set to DLPI_SET_NOLOOPBACK during ifconfig. What is this message and how can I handle it in the driver?

During the IP/DLPI driver stream setup stage, on a per stream basis, IP sends a DLPI_SET_NOLOOPBACK IOCTL and the driver marks the stream by turning on the noloopback_flg in the per-stream control structure. From this point forward, the driver checks each outbound packet in this stream to see if the noloopback_flg is on. If the noloopback_flg is on, the b_flag in an mblk is marked with MSGNOLOOP.

When these outbound packets are processed, the driver will not loopback packets that have the MSGNOLOOP flag turned on. Since IP will not send any unicast packets with a self-destination address, the noloopback_flag is mainly for multicast and broadcast packets. For an example of this implementation, refer to the enet sample driver provided with the DDK.

7.9 Can a STREAMS/UX driver ever sleep?

No, except for open() and close() routines. STREAMS/UX provides protection to ensure that all resources are properly protected from upper layer access during open() and close() operations.
7.10 What is the difference between a Native STREAMS DLPI network driver and a non-native HP DLPI based network driver?

A Native STREAMS DLPI network driver implements its own DLPI layer. A Native STREAMS DLPI network driver is really two drivers in one, a STREAMS driver and a WSIO interface driver. In contrast, a non-native HP-DLPI based network driver is also a WSIO interface driver, but it relies on the DLPI layer implementation provided by the HP-DLPI. It does not implement its own DLPI layer.

7.11 Do I need to write a DLPI layer or does the HP-UX I/O subsystem provide a standard DLPI layer?

Starting with HP-UX 11i v2, HP exports HP-DLPI interfaces to IHV drivers. HP-UX 11i v2 IHV drivers do not need to write their own DLPI layers; instead they can leverage the HP-DLPI implementation. HP-UX 11i v2 also supports network drivers that have their own DLPI layer. Both of these approaches are documented in Chapters 9, 10, and 11 of the DDG. For HP-UX 11.0 and HP-UX 11i v1, the HP-DLPI is not exported to IHV drivers. IHV network drivers must provide their own DLPI layer in HP-UX 11.0 and 11i v1.

7.12 What should the mod_operations be for a DLKM network driver?

It depends. For a Native STREAMS DLPI network driver, you should use str_drv_ops as mod_operations in the DLKM version. For a non-native Network driver, you should use gio_drv_ops in the DLKM version.

7.13 I successfully loaded my driver and it claims the NICs, but when I try to configure an IP address on a NIC instance using ifconfig, it fails. Why?

Make sure that a device special file is created for your driver. Run the lsdev command to get the driver major number assigned to your driver. Run the mknod command to create a device special file. An example of the sequence of commands for the sample enet driver follows:

```
$ lsdev | grep enet
 76  -1  enet  unknown
$ mknod /dev/enet c 72 0x00004c
```

7.14 I’m writing a driver for HP-UX 11i v2. What should I include for the type attribute in the modmeta file?

If you are writing a Native STREAMS DLPI network driver, you must include the following line:
```
type streams_drv
```

If you are writing a non-Native WSIO network interface driver, you must include the following line:
```
type wsio_intfc
```

7.15 I’m porting my HP-UX 11i v1 network driver to HP-UX 11i v2. Where do I start?

Start with the Network Driver Migration Guide available with the HP-UX 11i v2 DDK. It provides an overview of the network driver architecture alternatives available in HP-UX 11i v2. Refer to the HP-UX 11i v2 DDG for more specific details.
7.16 If I have multiple instances of the same NIC, how do I specify which instance to configure?
WSIO assigns an instance number to each device on the host. The instance numbers are unique across each
driver class. For LAN devices, when a lanscan is performed each instance of the device is listed with its name
followed by its PPA (no space) in the Net-Interface column of the output. The PPA number is the instance
number of the device.

8 Mass Storage Drivers

8.1 How do I incorporate SCSI-3 addressing for more than 16 targets and 8 LUNs?
HP-UX 11i v2 provides new services that let a driver create interface nodes that can be used to access greater
than 8 LUNs. Refer to Chapter 16, “Mass Storage Stack Architecture” and Chapter 18, “Writing a SCSI
Interface Driver” in the HP-UX 11i v2 DDG.

8.2 Are there instructions for porting a Solaris interface (transport) driver to HP-UX?
There is a driver-porting guide that explains the specific steps you need to follow to port a Solaris interface
(transport) driver to HP-UX 11i v1. It is available at:
http://www.hp.com/go/hpux_ddk

8.3 Is there a restriction on which driver class a SCSI interface driver can belong to?
Yes. SCSI Interface drivers must belong to the ext_bus driver class. The SCSI subsystem relies on a unique
instance number in the ext_bus driver class for the ISC of each NIC.

9 DLKM

9.1 My interface driver loaded successfully, but it does not claim the NIC. Why?
Loading a DLKM driver does not automatically invoke device probing. By running ioscan to force a device
scan, the driver attach routine will be called.

9.2 My DLKM driver registers a probe routine with WSIO, but the driver probe routine is not
called during the device scan. Why?
DLKM drivers must call wsio_activate_probe() to activate the driver probe routine. Refer to the DDR
manual for additional details on the WSIO interface.

9.3 How does my DLKM driver synchronize the unload entry point with the driver ISR?
In its unload entry point, the DLKM driver calls wsio_intr_deactivate() to disable an interrupt object. A
call to this service blocks until the driver returns from the ISR (if one is currently running).
10 WSIO

10.1 How can a driver create an interface node in the I/O tree?
HP-UX 11i v2 provides the `wsio_create_interface()` WSIO interface for registering new I/O interfaces.

10.2 What is a TRANSPARENT node?
A TRANSPARENT node has no hardware associated with it. It is used to create hardware path elements. A request to create a TRANSPARENT node is identified by passing `WSIO_TRANS` as a `wsio_mod_type_t` argument to `wsio_create_interface()`. Refer to the DDR for additional information.

10.3 I have a pointer to an ISC structure. How can I find out the hardware path to the I/O tree node identified by the ISC?
The WSIO service `wsio_isc_to_hwpath()` is used to get the hardware path associated with the ISC.

10.4 How can a driver get or set an interface attribute?
WSIO provides services to create, modify, query, and destroy attributes associated with an interface.

10.5 Do drivers on Itanium®-based platforms need to call `dma_sync()` or `dma_sync_IO()`?
Itanium®-based platforms are fully I/O cache coherent. It is not necessary for drivers to call `dma_sync()` or `dma_sync_IO()`, but to keep the same source code across both PA-RISC and Itanium®-based platforms it is advisable to call these services and let the platform code determine what is needed.

10.6 Do DMA mapping services always succeed?
DMA mapping services may or may not succeed in mapping the supplied host address range. The driver is responsible for handling both partially succeeded and failed calls to the WSIO DMA mapping services. This issue is further explained in the man pages for `wsio_map_dma_buffer()` and other DMA mapping services.

10.7 My NIC supports 64-bit DMAs. How can I enable this functionality in my driver?
The driver should call `wsio_dma_set_device_attributes()` to set the `WSIO_DMA_ATTR_ADDR_WIDTH` to 64. The default value is 32. The driver should call `wsio_dma_set_device_attributes()` before allocating any DMA objects.

10.8 When I try to retrieve the ISC pointer using `wsio_get_isc()`, I get a NULL pointer even though the call to `wsio_get_isc()` returns success. What’s wrong?
The device instance number is supposed to be encoded in bits 16-23 (in the third byte). Look at the definition of `m_instance` in `<sys/io.h>` to get a clear picture of the instance number encoding. If the `dev_t` that is passed down has this encoding, `wsio_get_isc()` will return a valid ISC pointer. If there are problems with the `wsio_drv_info_t` argument, `wsio_get_isc()` returns a 0 (failure). Otherwise, `wsio_get_isc()` returns success, even though it could not find a valid ISC at the instance pointed to by the `dev_t`.
11 Kernel Services

11.1 What is KRS?
Beginning with HP-UX 11i v2, Kernel Registry Services (KRS) provides a common mechanism to support the maintenance of persistent structured data.

11.2 Why do I need KRS?
To maintain your kernel module’s persistent data across reboots and to make it available earlier in the boot process than traditional UNIX system files.

11.3 How can I use KRS?
KRS has a set of kernel interfaces that you can use to store and to recover persistent data specific to a module or driver.

11.4 Where can I find information about the KRS APIs?
Refer to the Driver Development Reference (DDR) for the KRS API manpages and the KRS chapter in the Driver Development Guide (DDG) for additional KRS information. Both manuals are located on the DSPP Web site.

11.5 What types of data can I store in KRS?
KRS supports various types of data, including text, integers, references and addresses. In addition, KRS supports user-defined data.

11.6 Is there a kernel system call that I can invoke to cause my driver thread to sleep() without a busywait? I do not want the processor to spin while letting other kernel threads proceed. I need to have my driver thread sleep for several seconds (not milliseconds or microseconds). This is because some of my IOCTLs are driver SCSI robotic auto-changers, which can take many seconds to respond to move-medium commands.

There is no dedicated kernel service that can be called to sleep for some definite number of seconds, but this can be achieved by using sleep() and timeout(). Call timeout() on the wakeup() function with the channel set to the address of the interface specific structure. You can set the timeout value to the number of seconds the thread has to sleep, for example {number of seconds x 100 ticks}. Then call sleep() on the channel (addr of interface specific structure). After timing out, the wakeup() function is called. This wakes up the sleeping thread on the channel (this is your thread). An example of the code follows:

timeout (wakeup, iftp, timeout_value); sleep ((caddr_t) iftp, PRIBIO);

Where iftp is a pointer to interface specific structure.
12.1 How can I make a tape depot out of a directory depot?
At the command prompt run:

```
# swpackage -x media_type=tape  -s <exiting dir depot location/dir depot>  -d <tape depot location>/<depotname>.depot
```

12.2 How can I install a directory depot on a remote system?
At the command prompt run:

```
# swinstall -s <directory depot location>/<directory depot> <productname> @ <hostname>
```

Note: You need to have the correct ACL permissions set to install or copy a depot.

12.3 How can I copy a directory depot on to a remote system?
At the command prompt run:

```
# swcopy -s <depot location>/depot productname @ hostname
```

Note: You need to have the correct ACL permissions set to install or copy a depot.

12.4 How do I set Software Depot ACLs?
If you are working on Host B and you want to install or delete a depot on HostA, run the following command on HostB:

```
# swacl -l root -M user:root@HostA:crwit
```

If you want to copy and register a depot from HostB to HostA, run the following command on HostB:

```
# swacl -l host -M user:root@HostA:crwit
```

12.5 From command line, how do I list the products and filesets (and their attributes) contained in a bundle?
Run the following command:

```
# swlist -dvl <product><fileset> @ <depot location>/depot
```

12.6 How do I install a different version of the same product on a system?
During swinstall, set the allow_multiple_versions option to true:

```
allow_multiple_versions = true
```
12.7 How can I skip installing the same version of a fileset again?
During `swinstall`, set the `reinstall` option to `false`:
```
reinstall = false
```
Note: The default setting for the `reinstall` option is `false`.

12.8 How can I install an older version of the same fileset on a system?
During `swinstall`, set the `allow_downdate` option to `true`:
```
allow_downdate = true
```

12.9 How do I compress a tape depot?
In your PSF file, set the `compress_files` option to `true`:
```
compress_files = true
```
This compresses files before transferring them from a source.

12.10 How do I copy a control file to an existing directory depot?
If the depot is already created and you want to add a control file, run the following commands in the order given:
```
# cd <dir depot location>/<depotname>/catalog/<productname>/pfiles
# cp <location of postinstall> postinstall #swmodify -d -x
control_files=postinstall<productname> @ <dir depot location>/<depotname>
```

12.11 How do I copy products from an existing directory depot to different directory depot?
Run the following command:
```
# swcopy -s <existing depot location>/<depotname> <productname> @ <destination depot location>
```
For example, `# swcopy -s /var/spool/sw/depots/docs1/DOCS @ /var/spool/sw/depots/docs2/` will copy the `DOCS` product from the `docs1` depot to `docs2` depot.

12.12 How can I list the products contained in a bundle?
Run the following command:
```
# swlist -l product <bundlename>
```

12.13 How can I list the filesets contained in a bundle?
Run the following command:
```
# swlist -l fileset <bundlename>
```
12.14 How do I get the attribute information for a bundle, product, or fileset?

Run the following command:

```
# swlist -a attributename <bundlename>/<productname><fileset>
```

For example, `# swlist -a description B543A` will display a description of all B543A and description of all products under B543A.

**Note:** `swlist -a description <bundlename>` will list a description of the specified bundle and a description of all of the products contained in that bundle.

---

13 The DDICT Tool

13.1 What is DDICT?

DDICT is an acronym for Device Driver Interface Compliance Tool. DDICT verifies that an HP-UX driver is only using the kernel APIs that are exported. DDICT is available starting with HP-UX 11i v2.

13.2 Where can I get the DDICT?

The DDICT comes with the HP-UX DDK 11i v2 DDK depot.

13.3 What does a violation in the DDICT output mean?

The DDICT lists any driver references to un-exportable symbols (as defined in the DDK) as violations. You must fix any violations listed by the DDICT or the driver will not be qualified for the “Certified for HP-UX” logo available through the hphp program. Contact hphp.support@hp.com for assistance fixing violations identified by the DDICT.

13.4 What should I do if the DDICT lists symbols that are not directly referenced in my driver as violations?

Package the following items in a tar file and send it to hphp.support@hp.com:

- Driver module (not the source, since we do not want to see your driver sources)
- Driver makefile(s) (for example, `Makefile` and `Makefile.bld`)
- Driver `modmeta` file
- DDICT output

---

14 KWDB

14.1 How can I get the KWDB debugger for HP-UX 11i v2?

Starting with HP-UX 11i v2, the KWDB debugger is shipped with the standard OS distribution.
14.2 What are the major changes in the KWDB for HP-UX 11i v2?
The HP-UX 11i v2 KWDB includes both a kernel debugger and a crash dump analyzer (also known as “Q4”). To invoke Q4 capabilities from the KWDB, enter the `kwdb -q4` command. The KWDB on HP-UX 11i v2 also prints file names and line numbers in the debug output.

14.3 Where can I find the user documentation for the KWDB?
KWDB documentation is shipped included with the HP-UX 11i v2 DDK.

14.4 What is PXDB and why do I need it?
PXDB is a kernel pre-processor. A kernel must run through PXDB before KWDB/Q4 can extract symbol information from it.

14.5 Where can I find PXDB?
PXDB will be located on one of the following file paths:

```
/opt/langtools/bin/pxd
/usr/contrib/Q4/bin/q4pxdb (same tool with a different name)
```

14.6 Is there an IPF version of PXDB?
No. IPF kernels do not get pre-processed.

15 Kernel Tunables

15.1 How is a computed default for a kernel tunable generated?
A computed default is generated when a kernel tunable registers a handler function that includes an algorithm to calculate a kernel tunable default value based on the state of the system.

15.2 How do I develop a kernel tunable that will generate a computed default value?
To generate a computed default value, a kernel tunable must have a handler function that responds to the `KTOP_GETDEFAULT` operation code. This code is only used with a reason code of `KEN_EVENT`. When the handler function receives this request, it computes an appropriate default value then it gives that value to the tunable infrastructure by calling `ktune_savedefault()`. For more information refer to Chapter 7, “Tunable Parameters in Drivers” in the HP-UX 11i v2 DDG.

15.3 Where is a computed default for a kernel tunable used?
If you do not set a value for a kernel tunable, the kernel uses the default value computed for the tunable by the handler.
15.4 If I provide a kernel tunable, do I need to provide a computed default value?
Only kernel tunables that adjust their behavior based on the system state require a computed default.

15.5 What if I don't provide a computed default?
If a kernel tunable does not have a computed default, the default value must be specified in the module metadata file defining the kernel tunable.

15.6 What happens if a kernel tunable with a computed default does not have a handler registered with the KTOP_GETDEFAULT capability?
The kernel will panic when tunables are initialized. The kernel also pansics if a tunable handler registers with the KTOP_GETDEFAULT capability, but does not include a computed default in the modmeta file.

15.7 What kernel patches are required if I want to use the computed default functionality?
The following patches must be installed before you can use the computed default functionality:
PHCO_33023
PHKL_33024
PHKL_33025
PHKL_33028

15.8 How do I know if a kernel tunable has a computed default?
A kernel tunable has a computed default if it shows the following output at nextboot in the verbose mode:
Value at Next Boot Default (automatic)

15.9 Where do I get the modmeta compiler that supports computed defaults?
The modmeta compiler that supports computed defaults and it's manpage are provided via the Build Environment (BE) Tools depot, which is downloaded from the HP-UX 11i v2 web page at:

Note: The build environment (BE) tools that are available as part of the HP-UX 11i v2 Driver Development Kit depot do not include the modmeta compiler that supports computed defaults.