

# HP-UX 11i v3 Mass Storage Device Naming



Abstract.....	2
Terms and Definitions .....	2
Introduction.....	3
Persistent Device Special File (DSF) Naming Scheme.....	4
Disk device file naming convention.....	4
Tape device special file naming convention.....	5
Auto-changer and Library robotic device special file naming convention.....	5
Array controller and other device file naming convention.....	6
Creating device special files manually.....	6
Displaying options associated with a device special file .....	8
The agile addressing hardware path format.....	9
LUN hardware path format .....	9
Lunpath hardware path format.....	9
Mapping between agile addressing scheme and legacy scheme.....	10
Mapping between legacy DSFs and persistent DSFs.....	10
Mapping between legacy hardware paths and agile addressing hardware paths.....	11
Mapping between LUNs and LUN paths.....	12
Additional Resources.....	13

## Abstract

This paper describes the HP-UX 11i v3 mass storage device naming. It is intended for system administrators and other users of the HP-UX mass storage subsystem. The reader is assumed to have a basic knowledge of the mass storage device naming on HP-UX releases prior to HP-UX 11i v3.

HP-UX 11i v3 introduces a new agile naming scheme for mass storage devices, which addresses the limitations of the device naming scheme used in previous releases of HP-UX 11i. The document provides an overview of the benefits of the agile naming scheme, and presents the new format of device special file names and hardware paths.

## Terms and Definitions

<a href="#">Agile addressing</a>	<p>The Device File Name of the logical unit is independent of the paths leading to the LUN.</p> <p>The ability to address a logical unit with the same device file regardless of the location of the LUN, i.e. The device file for a LUN remains the same even if the LUN is moved from one HBA to another, from one switch/hub port to another or presented via a different target port to the host.</p>
<a href="#">Agile View of the I/O tree</a>	<p>View of the I/O Tree using agile addressing to represent elements of the I/O tree.</p>
<a href="#">(Legacy) H/W Path</a>	<p>H/W Path in the legacy format. Used for all components, mass storage or not.</p>
<a href="#">Legacy or agile I/O Tree</a>	<p>The I/O Tree is the HP-UX representation of physical and virtual devices discovered by the Operating System. The legacy I/O tree is limited to the legacy view. The agile I/O tree is limited to the agile view.</p>
<a href="#">LUN</a>	<p>Logical unit that refers to an end storage device such as disk, tape, floppy, cdrom, changer, etc. This is the logical unit itself and does not represent the path to the logical unit.</p>
<a href="#">lunpath</a>	<p>I/O path to a LUN in agile format.</p>
<a href="#">LUN id</a>	<p>This is the SCSI address of the LUN within the target as defined in the SCSI standard.</p>
<a href="#">Agile naming model</a>	<p>DSF naming model introduced in HP-UX 11i v3 to support agile addressing.</p>
<a href="#">Persistent DSF</a>	<p>DSF following the persistent naming model conventions. Does not contain any encoding of bus, target identifier or device specific option information.</p>

legacy naming model / legacy format	Device special file format convention used prior to HP-UX 11i v3. This model is maintained in HP-UX 11i v3 for backward compatibility purpose.
(legacy) DSF	A path dependent device file, following the legacy naming model conventions, wherein the DSF embeds the bus/target/lun/option for a specific path to a mass storage device (Ex. /dev/dsk/c##d##).
SCSI class driver	A HP-UX device driver which manages one or more specific classes of mass storage devices. Ex : Disk Driver, Tape Driver, Changer Driver, Pass-thru Driver.
Target Id	Corresponds to the target port identifier as defined in SCSI transport protocols

## Introduction

HP-UX 11i v3 introduces a new agile addressing scheme for mass storage devices, with opaque minor numbers, persistent device special files (DSFs), and new hardware path types and formats. The addressing scheme used in previous HP-UX releases, will co-exist with this new scheme to ensure backward compatibility and is referred to as legacy addressing. The legacy addressing will be deprecated in a future HP-UX release.

In the legacy addressing scheme, each device path is represented by a minor number, a legacy DSF and a legacy hardware path. The minor number directly encodes SCSI-2 addressing information (SCSI bus, target address and logical unit number), and device options (disk partition number, tape density, etc.). The legacy DSF name also embeds SCSI-2 addressing information. This has the following limitations:

- A device with multiple paths is represented by several legacy device special files. Applications and upper layer modules (File system, volume manager, etc.) have to use independent methods such as writing meta data on the device to be able to determine that different legacy DSFs correspond to the same device.
- Limited scalability: legacy limits of 256 SCSI bus instances, 16 target identifiers per bus instance and 8 logical unit numbers per target identifier. The use of 15 bits to encode the minor number limits the overall number of lunpaths to 32768.
- Dependency on the physical connection of the device. If for instance the device is connected to the host through a different host-based adapter (HBA) or a different target port, the device special file and the minor number may be affected requiring a re-configuration of applications or upper layer modules such as volume managers and file systems.

The agile addressing scheme addresses each of these limitations:

- It defines a persistent device special file and a virtualized hardware path, which represent the device independently of its paths. So a device with multiple paths is represented by a single device special file.
- It increases the scalability by using 24 bit opaque minor numbers and storing the device options separately. The binding between a minor number and certain device options is maintained in a system registry.

- It enables certain re-configurations of the host system and the storage area network (SAN), without affecting the representation of the device on the host system. This includes operations such as connecting the device through a different HBA on the host, and moving the device to a different switch or hub in the SAN.

## Persistent Device Special File (DSF) Naming Scheme

The persistent DSF is a path independent and persistent<sup>1</sup> representation of a Logical Unit (LUN). It is bound to the LUN using the LUN's unique World Wide Identifier (WWID). If the device is moved the persistent DSF is not affected.

The format of a persistent DSF name is as follows: `/dev/<subdir>/<class><instance>[options]`

- *subdir*: A subdirectory for the device class (for example, *rdisk* for raw device special files for disks, *disk* for block device special files for disks, *rtape* for raw device special files for tape devices, etc.).
- *class*: In general this corresponds to the class of the device: *tape*, *disk*, etc.
- *instance*: instance number assigned by the operating system to the device within its class.
- *options*: device options depending on the class of the device. For instance disk partition, tape density selection.

*Note: To determine class and instance number of devices, run the `ioscan` command (see `ioscan(1M)`<sup>2</sup>).*

If the device does not support a LUN WWID (inquiry page 0x83), the system can not uniquely identify it. In this case the system creates a device special file in the format described above, but the device special file will be associated with the path through which the device is discovered. If the device has multiple paths, multiple DSFs will be created. If the device is moved or the SAN is re-configured, the device special file may change as a unique LUN WWID is not available to fully and uniquely identify it.

The system automatically creates both legacy and persistent device special files for all devices discovered on the system<sup>3</sup>, and for certain predefined device options depending on the driver the device is bound to. However, legacy device special files are created only up to the architectural limit which allows up to 32768 legacy DSFs and 256 `ext_bus`. Above the legacy addressing scheme limits, only persistent device special files are created.

The following sections describe the agile addressing naming scheme for different device classes.

### Disk device file naming convention

Disk devices correspond to block devices including the following types: random access, storage array, CD/DVD, Write-Once Read-Multiple (WORM), and Optical Memory (OM). They are typically claimed by the `esdisk` class driver. The format of the device special file depends on the method used to access the device:

- Raw access: `/dev/rdisk/disk<instance>[_p<disk_partition_number>]`
- Block access: `/dev/disk/disk<instance>[_p<disk_partitionnumber>]`

<sup>1</sup> Persistency only applies to the current installation and may not be maintained across re-installations

<sup>2</sup> On 11i v3 the default view of `ioscan` is the legacy view, to see new agile information with `ioscan` use the `'-N'` option.

<sup>3</sup> Legacy devices are not created when legacy mode has been disabled with the `rmsf -L` command.

Where `disk_partition_number` is the number assigned to the disk partition by the system if the disk is formatted into several logical partitions.

The following table shows examples of disk device special files created for two disks with instance numbers 0 and 100 respectively. The disk with the instance number 0 has two partitions.

Raw device special file	Block device special file	Comments
<code>/dev/rdisk/disk0</code>	<code>/dev/disk/disk0</code>	Represents the whole disk.
<code>/dev/rdisk/disk0_p1</code>	<code>/dev/disk/disk0_p1</code>	First partition of the disk
<code>/dev/rdisk/disk0_p2</code>	<code>/dev/disk/disk0_p2</code>	Second partition of the disk
<code>/dev/rdisk/disk100</code>	<code>/dev/rdisk/disk100</code>	disk device with instance 100

## Tape device special file naming convention

Tape devices typically correspond to sequential access devices claimed by the `estape` driver. The format of the device special file name is as follow:

- `/dev/rtape/tape<instance>_[w]<density>[C][n][b]`

Where:

- `w`: immediate report disabled. A write request waits until the data are written on the medium.
- `density`: is the density selection.
- `C`: Data compression mode
- `n`: no rewind on close
- `b`: Berkeley style. The tape is not repositioned following a read. By default the AT&T style is used.

*Note: density values for supported tape devices are defined in the header file: `<sys/mtio.h>`. `BEST` is the recommended density. It allows the system to automatically select the best density for the device.*

For each tape device the system automatically creates device special files corresponding to the options shown in the table below.

Options	Example for a tape device with instance number 0
• Best density and AT&T style	<code>/dev/rtape/tape0_BEST</code>
• Best density and Berkeley style	<code>/dev/rtape/tape0_BESTb</code>
• Best density, AT&T style and no rewind on close	<code>/dev/rtape/tape0_BESTn</code>
• Best density, no rewind on close, and Berkeley style	<code>/dev/rtape/tape0_BESTnb</code>

## Auto-changer and Library robotic device special file naming convention

Auto-changer and library robotic SCSI devices are claimed by the `eschgr` driver. Their device special files have the following format:

- `/dev/rchgr/autoch<instance>`

For example if the system discovers 2 auto-changer devices and assigns them instance numbers 0 and 1 respectively, it will automatically create the following device files: `/dev/rchgr/autoch0` and `/dev/rchgr/autoch1`.

## Array controller and other device file naming convention

Any device which is not claimed by a class driver is bound by default to the pass-through driver: `esctl`. This is the case for array controller devices. The device special name format for these devices is:

- `/dev/pt/pt<instance>`

For example if the system discovers 3 array controllers and assigned them instance numbers: 0, 10 and 12 respectively, it will automatically create the following device special files for these devices: `/dev/pt/pt0`, `/dev/pt/pt10` and `/dev/pt/pt12`.

## Creating device special files manually

The user may need in some situations, to manually create device special files. Examples of such situations include: restoring device files deleted accidentally, creating a pass-through device file for a device, or creating a custom device special file corresponding to certain device options.

The `insf`, `mksf` and `mknod` commands can be used to manually create DSFs:

- `insf`: can be invoked to re-create device special files automatically created by the system for devices of a particular class, devices bound to a specific driver, or for all devices on the system. If new devices are present on the system, the system allocates new minor numbers for these devices and stores the binding between the minor numbers, device options and the device in a registry.
- `mksf`: can be used to create: (1) all device special files created by default by the system for a particular device; (2) a custom device special file corresponding to certain device options. The user can override the standard naming convention by specifying his own device special file name. If a minor number was not already allocated for the device options specified, the system allocates a new minor number, and stores the binding between it and the device options in a registry.
- `mknod`: can be used to create a device special file by explicitly specifying the major number, the minor number, and the device special file name. However, unlike the legacy minor numbers which can be constructed from the hardware addressing components, the new persistent minor numbers are opaque values assigned by the system which cannot be constructed by the user. Using `mknod` to create a persistent DSF requires the use of a minor number already assigned to the device by the system.

*Note: See the `mknod(1M)`, `mkfs(1M)` and `insf(1M)` man pages for more details.*

### **Using `insf` to Create device files created by default by the system**

The following examples show how you can use the `insf` command to create or re-install device files that the system creates by default.

- To create default device files for all new devices on the system. New devices are those for which no device file has been created before.  
# `insf`
- To create default device files for all disk devices:

```
# insf -C disk
```

- To re-install all default device files for all devices bound to the estape driver:

```
# insf -d estape -e
insf: Installing special files for estape instance 2 address 64000/0xfa00/0x5
insf: Installing special files for estape instance 3 address 64000/0xfa00/0x10
```

*Note: When no DSF is created, insf does not display any message. That is the case for instance if insf is run without the -e option and there is no new device on the system.*

### Creating device files using mksf

The following examples show how you can use the *mksf* command to manually create device files with default names, your own device file name, and custom device file corresponding to certain device options.

- To create the default block device file for disk device with instance 98 and hardware path 64000/0xfa00/0x13 using the instance number to identify the device:

```
# mksf -C disk -I 98
# ls -l /dev/disk/disk98*
brw-r----- 1 bin      sys          1 0x000013 Jan 16 22:27 /dev/disk/disk98
```

- To create the default raw device file for disk device with instance 98 with hardware path 64000/0xfa00/0x13 using the hardware path to identify the device:

```
# mksf -r -H 64000/0xfa00/0x13
# ls -l /dev/rdisk/disk98*
crw-r----- 1 bin      sys          11 0x000013 Jan 16 22:28 /dev/rdisk/disk98
```

- To create the device file for the first disk partition of the disk with hardware path 64000/0xfa00/0x13. Note that the system allocates a new minor number (0x000027) to be associated with the partition.

```
# mksf -H 64000/0xfa00/0x13 -s 1
# ls -l /dev/disk/disk98*
brw-r----- 1 bin      sys          1 0x000013 Jan 16 22:27 /dev/disk/disk98
brw-r----- 1 bin      sys          1 0x000027 Jan 16 22:35 /dev/disk/disk98_p1

# mksf -r -H 64000/0xfa00/0x13 -s 1
crw-r----- 1 bin      sys          11 0x000013 Jan 16 22:28 /dev/rdisk/disk98
crw-r----- 1 bin      sys          11 0x000027 Jan 16 22:46 /dev/rdisk/disk98_p1
```

- To create a device file name /mydev/disk19\_partition2 corresponding to the second partition of disk19. Note that the directory /mydev should already exist. Also note that the system allocates a new minor number (0x000028) for the second disk partition.

```
# mksf -C disk -I 19 -s 2 /mydev/disk19_partition2
# ls -l /mydev
total 0
brw-r----- 1 bin      sys          1 0x000028 Jan 16 22:58 disk19_partition2
# lssf /mydev/disk19_partition2
esdisk section 2 at address 64000/0xfa00/0x13 /mydev/disk19_partition2
```

*Note: It is not recommended to create device files outside /dev unless for test purposes and in a temporary manner. Not doing so may affect certain commands or subsystems, which assume that their device files are under /dev.*

## Creating pass-through device files using *mksf*

In some situations the user may need to send SCSI commands directly to the device via the pass-through driver: *esctl*. For that purpose the user should create a device file attached to the pass-through driver. In HP-UX releases prior to 11i v3, *mknod* was used to create pass-through device files. In HP-UX 11i v3 the *-P* option of *mksf* must be used to create pass-through persistent DSFs. The *-P* option can be used with any of the device identifiers accepted by *mksf*, including the combination of driver/class and instance number, and the hardware path. By default the system uses the following convention for the pass-through device file name:

- `/dev/pt/pt_<class><instance>`

Where:

- *class*: is the device class such as *disk*, *tape* or *autoch*.
- *instance*: the instance number assigned to the device within its class by the system.

The following examples show how to create pass-through device files using *mksf*.

- To create the default pass-through device file for disk with instance 98: `/dev/pt/pt_disk98`  

```
# mksf -P -C disk -I 98
```
- To create the default pass-through device file for tape device with instance number 2: `/dev/pt/pt_tape2`  

```
# mksf -P -C tape -I 2
```
- To create a pass-through device file for the tape device with instance number 3, with the name of the device special file specified as: `/dev/pt/tape3_passthrough`  

```
# mksf -P -C tape -I 3 /dev/pt/tape3_passthrough
```

## Displaying options associated with a device special file

The *lssf* command displays the options associated with a device special file: `lssf <device_file>`.

Below are examples of use of the *lssf* command to display device options and other information.

```
# lssf /dev/disk/disk15 /dev/disk/disk15_p1 /dev/disk/disk15_p3
esdisk section 0 at address 64000/0xfa00/0x0 /dev/disk/disk15
esdisk section 1 at address 64000/0xfa00/0x0 /dev/disk/disk15_p1
esdisk section 3 at address 64000/0xfa00/0x0 /dev/disk/disk15_p3

# lssf /dev/rtape/tape2_BEST
estape AT&T BEST density at address 64000/0xfa00/0x5 /dev/rtape/tape2_BEST

# lssf /dev/rtape/tape2_BESTn
estape AT&T No-Rewind BEST density at address 64000/0xfa00/0x5 /dev/rtape/tape2_BESTn

# lssf /dev/rtape/tape3_BESTb
estape Berkeley BEST density at address 64000/0xfa00/0x10 /dev/rtape/tape3_BESTb

# lssf /dev/rtape/tape2_BESTnb
estape Berkeley No-Rewind BEST density at address 64000/0xfa00/0x5 /dev/rtape/tape2_BESTnb
```

# The Agile Addressing Hardware Path Format

The agile addressing scheme defines two new sets of hardware paths for each device:

- The LUN hardware path: is a virtualized path, which represents the LUN independently of any of its I/O paths.
- A set of lunpath hardware paths: each represents a specific I/O path to the LUN and maps to a single legacy hardware path. The format of the lunpath hardware path is different from the legacy hardware path and depends on the SCSI transport.

## LUN hardware path format

The virtualized LUN hardware path is built using the following elements:

- The root node of all virtual devices: by convention this has the constant address 64000. Virtual devices in the context of the I/O tree are devices for which the hardware path does not represent any physical location of the device.
- The virtual bus address under which all LUN nodes are attached: by convention this has the constant address 0xfa00.
- A virtual LUN identifier corresponding to the hexadecimal representation of the default minor number assigned to the device in the agile view.

So the hardware path of a LUN will always have the following format: *64000/0xfa00/0xh*, where *h* is the hexadecimal representation of the default minor number assigned to the device.

The following *ioscan* output shows examples of LUN hardware paths and persistent DSFs for disk devices.

```
# ioscan -kfNnC disk
Class      I  H/W Path  Driver S/W State  H/W Type  Description
=====
disk      15  64000/0xfa00/0x0  esdisk CLAIMED    DEVICE    HP 36.4GST336706LC
          /dev/disk/disk15  /dev/rdisk/disk15
          /dev/disk/disk15_p1  /dev/rdisk/disk15_p1
          /dev/disk/disk15_p2  /dev/rdisk/disk15_p2
          /dev/disk/disk15_p3  /dev/rdisk/disk15_p3
disk      16  64000/0xfa00/0x1  esdisk CLAIMED    DEVICE    HP 36.4GST336706LC
          /dev/disk/disk16  /dev/rdisk/disk16
          /dev/disk/disk16_p1  /dev/rdisk/disk16_p1
          /dev/disk/disk16_p2  /dev/rdisk/disk16_p2
          /dev/disk/disk16_p3  /dev/rdisk/disk16_p3
disk      17  64000/0xfa00/0x2  esdisk CLAIMED    DEVICE    TEAC      DV-28E-B
          /dev/disk/disk17  /dev/rdisk/disk17
disk      18  64000/0xfa00/0x3  esdisk CLAIMED    DEVICE    SEAGATE  ST39103FC
          /dev/disk/disk18  /dev/rdisk/disk18
```

## Lunpath hardware path format

The format of the lunpath hardware path is as follows:

- *<HBA hw path>.<target port WWN or port id>.<SCSI-3 LUNid>*

Where:

- *HBA hw path*: legacy HBA hardware path (example: 0/3/1/0).
- *target port WWN or port id*: hexadecimal representation of the target port Worldwide name (WWN), or the target port identifier if the SCSI transport does not support WWN (example: 0x21000020370fe8c8 for WWN, and 0x0 for port id).
- *SCSI-3 LUNid*: hexadecimal representation of the SCSI-3 64 bit LUN identifier.

Notes:

- The *lunpath* represents an I/O path to a LUN. In the agile view of the I/O tree, it is an independent SCSI object of the class 'lunpath', and bound to the pseudo-driver *eslpt*.
- Likewise the *target path* is an independent object in the agile view of the I/O tree. It is of class 'tgtpath' and is bound to the pseudo-driver *estp*.

The following output of the *ioscan* command shows examples of *lunpath* hardware paths. The first three correspond to *lunpaths* of devices attached to system via parallel SCSI, and they include the port identifier. The last three correspond to *lunpaths* of devices attached to system via Fibre Channel. They include the target port WWN.

```
# ioscan -kfnClunpath
Class      I  H/W Path  Driver S/W State  H/W Type  Description
=====
lunpath    2  0/0/2/0.0x0.0x0          eslpt CLAIMED  LUN_PATH  LUN path for
disk17
lunpath    1  0/1/1/0.0x0.0x0          eslpt CLAIMED  LUN_PATH  LUN path for
disk16
lunpath    0  0/1/1/0.0x1.0x0          eslpt CLAIMED  LUN_PATH  LUN path for
disk15
lunpath    7  0/3/1/0.0x21000020370fe8c8.0x0  eslpt CLAIMED  LUN_PATH  LUN path for
disk22
lunpath    4  0/3/1/0.0x21000020371972e3.0x0  eslpt CLAIMED  LUN_PATH  LUN path for
disk19
lunpath    5  0/3/1/0.0x21000020371972eb.0x0  eslpt CLAIMED  LUN_PATH  LUN path for
disk20
```

Notes:

- The elements of the *lunpath* hardware path, up to the HBA hardware path, are in decimal, but the target port WWN or port id and the SCSI-3 LUN id are displayed in hexadecimal for ease of reading and correlation with the hardware components.
- For parallel SCSI, the elements of the legacy hardware path and of the *lunpath* hardware path are the same, with the only difference that the port id and SCSI-3 LUN id are displayed respectively in decimal and in hexadecimal.
- For Fibre Channel, the LUN id can be decoded in a LUN number and addressing method using the command '*scsimgr get\_attr -a lunid -H <lunpath\_hw path>*'

## Mapping Between Agile Addressing Scheme and Legacy Scheme

In HP-UX 11i v3 the *ioscan* command has new options *-N* and *-m*. The *-N* option displays the agile addressing view of the I/O tree. The *-m* option displays mappings between elements of the agile view and elements of the legacy view.

### Mapping between legacy DSFs and persistent DSFs

- To view the legacy DSFs corresponding to a persistent DSF.
  - *ioscan -m dsf <persistent\_dsf>*

In the example below *disk19* has 2 I/O paths. The persistent DSF */dev/disk/disk19* represents the disk independently of its paths. The 2 legacy DSFs */dev/dsk/c4t1d0* and */dev/dsk/c5t1d0* are associated to each path.

```
# ioscan -m dsf /dev/disk/disk19
Persistent DSF          Legacy DSF(s)
=====
/dev/disk/disk19       /dev/dsk/c4t1d0
                       /dev/dsk/c5t1d0
```

- To view the persistent DSF corresponding to a legacy DSF (Note: A legacy DSF always maps to a single persistent DSF)

- `ioscan -m dsf <legacy_dsf>`

```
# ioscan -m dsf /dev/dsk/c4t1d0
Persistent DSF          Legacy DSF(s)
=====
/dev/disk/disk19       /dev/dsk/c4t1d0
```

- To view mappings of all existing mass storage DSFs:

- `ioscan -m dsf`

```
# ioscan -m dsf
Persistent DSF          Legacy DSF(s)
=====
/dev/pt/pt2            /dev/rscsi/c4t10d0
/dev/pt/pt3            /dev/rscsi/c5t10d0
/dev/rdisk/disk15     /dev/rdisk/c2t1d0
/dev/rdisk/disk15_p1  /dev/rdisk/c2t1d0s1
/dev/rdisk/disk15_p2  /dev/rdisk/c2t1d0s2
/dev/rdisk/disk15_p3  /dev/rdisk/c2t1d0s3
/dev/rdisk/disk16     /dev/rdisk/c2t0d0
/dev/rdisk/disk16_p1  /dev/rdisk/c2t0d0s1
/dev/rdisk/disk16_p2  /dev/rdisk/c2t0d0s2
/dev/rdisk/disk16_p3  /dev/rdisk/c2t0d0s3
/dev/rdisk/disk17     /dev/rdisk/c0t0d0
/dev/rdisk/disk18     /dev/rdisk/c4t0d0
                       /dev/rdisk/c5t0d0
/dev/rdisk/disk19     /dev/rdisk/c4t1d0
                       /dev/rdisk/c5t1d0
```

## Mapping between legacy hardware paths and agile addressing hardware paths

- To view lunpaths and legacy paths of a LUN. In the example below the LUN has two lunpaths:

- `ioscan -m hwpath -H <lun_hardware_path>`

```
# ioscan -m hwpath -H 64000/0xfa00/0x3
Lun H/W Path          Lunpath H/W Path          Legacy H/W Path
=====
64000/0xfa00/0x3
                       0/3/1/0.0x21000020371972ee.0x0  0/3/1/0.8.0.255.0.0.0
                       0/4/1/0.0x21000020371972ee.0x0  0/4/1/0.8.0.255.0.0.0
```

- To view the hardware path of the LUN, and the legacy hardware path corresponding to a lunpath.

- `ioscan -m hwpath -H <lunpath_hardware_path>`

```
# ioscan -m hwpath -H 0/3/1/0.0x21000020371972ee.0x0
Lun H/W Path      Lunpath H/W Path      Legacy H/W Path
=====
64000/0xfa00/0x3
                   0/3/1/0.0x21000020371972ee.0x0    0/3/1/0.8.0.255.0.0.0
```

- To view the LUN hardware path, and the lunpath hardware path corresponding to a legacy hardware path

- `ioscan -m hwpath -H <legacy_hardware_path>`

```
# ioscan -m hwpath -H 0/3/1/0.8.0.255.0.0.0
Lun H/W Path      Lunpath H/W Path      Legacy H/W Path
=====
64000/0xfa00/0x3
                   0/3/1/0.0x21000020371972ee.0x0    0/3/1/0.8.0.255.0.0.0
```

- To view mapping between LUN hardware paths, lunpath hardware paths and legacy hardware paths of all LUNs

- `ioscan -m hwpath`

```
# ioscan -m hwpath
Lun H/W Path      Lunpath H/W Path      Legacy H/W Path
=====
64000/0xfa00/0x0
                   0/1/1/0.0x1.0x0          0/1/1/0.1.0
64000/0xfa00/0x1
                   0/1/1/0.0x0.0x0          0/1/1/0.0.0
64000/0xfa00/0x2
                   0/0/2/0.0.0x0.0x0        0/0/2/0.0.0.0
64000/0xfa00/0x3
                   0/3/1/0.0x21000020371972ee.0x0    0/3/1/0.8.0.255.0.0.0
                   0/4/1/0.0x21000020371972ee.0x0    0/4/1/0.8.0.255.0.0.0
64000/0xfa00/0x4
                   0/3/1/0.0x21000020371972e3.0x0    0/3/1/0.8.0.255.0.1.0
                   0/4/1/0.0x21000020371972e3.0x0    0/4/1/0.8.0.255.0.1.0
```

## Mapping between LUNs and LUN paths

To get information about the lunpaths of a LUN, run the following command:

- `ioscan -m lun <persistent_dsf>`

The example below shows the hardware path of disk20 (64000/0xfa00/0x5), the hardware paths of its 2 lunpaths (0/3/1/0.0x21000020371972eb.0x0, 0/4/1/0.0x21000020371972eb.0x0), and its block and raw device files (/dev/disk/disk20, /dev/rdisk/disk20).

```
# ioscan -m lun /dev/rdisk/disk20
Class      I  Lun H/W Path  Driver  S/W State  H/W Type  Health  Description
=====
disk       20  64000/0xfa00/0x5  esdisk  CLAIMED  DEVICE    online   SEAGATE ST39103FC
          0/3/1/0.0x21000020371972eb.0x0
          0/4/1/0.0x21000020371972eb.0x0
          /dev/disk/disk20  /dev/rdisk/disk20
```

## Additional Resources

### White papers

White papers can be found at:

<http://docs.hp.com/en/netsys.html#Storage%20Area%20Management>

- The Next Generation Mass Storage Stack HP-UX 11i v3
- HP-UX 11i v3 Persistent DSF Migration Guide
- HP-UX 11i v3 SCSI management and diagnostics utility

### Man pages

- `intro(7)`, `insf(1M)`, `ioscan(1M)`, `lssf(1M)`, `mksf(1M)`, `mknod(1M)`, `rmsf(1M)`, `scsimgr(1M)`