11 Mass Storage Stack Architecture

This chapter provides an overview of the Mass Storage I/O Stack in HP-UX. The Mass Storage I/O Stack includes SCSI Device Drivers (also known as SCSI Class Drivers), SCSI Services, and SCSI Transport Drivers (also known as SCSI Interface Drivers). The SCSI Transport Drivers are responsible for controlling the hardware that transports the SCSI protocol from the host system to the end SCSI device (LUN). Today in HP-UX there are parallel SCSI (pSCSI) and Fibre Channel (FC) transport drivers, which control pSCSI and FC Host Bus Adapters (HBAs). In the future there will be iSCSI transport drivers which control iSCSI HBAs or virtual interfaces.

The following sections are provided in this chapter:

- “Overview of HP-UX Mass Storage I/O Stack”
- “SCSI Addressing Paradigm”

For details on writing a SCSI Class Driver, see Chapter 13, “Writing SCSI Device Drivers.” For details on writing a SCSI Transport Driver, see Chapter 12, “Writing a SCSI Interface Driver.”
Overview of HP-UX Mass Storage I/O Stack

The following Figure 11-1, “Mass Storage Stack,” illustrates the Mass Storage I/O Stack in HP-UX, its key components, and the key modules that it interacts with.

Figure 11-1 Mass Storage Stack

The SCSI subsystem can be broadly divided into three layers:

1. SCSI class drivers (peripheral device drivers)
2. SCSI Services
3. SCSI transport drivers (HBA drivers)

Above the SCSI Subsystem are upper level modules such as volume managers and other pseudo drivers, file systems, and raw I/O access code, which can be used to access the SCSI subsystem. Further up the stack, user applications and I/O commands access the SCSI subsystem via one of these upper level modules.

Off to the side of the SCSI subsystem are WSIO and kernel services which the drivers can call to get themselves configured into the system and attached to their hardware, obtain resources, start and stop timers, map their card’s memory for PIO and setup system memory for DMA, etc. WSIO is the Device Driver Environment (DDE) CDIO which defines and supports the entry points and configuration services used by the SCSI Class and Transport Drivers to get configured and connected to each other and to their hardware. WSIO also provides various I/O tree and configuration database services and access to platform-dependent services such as PIO and DMA services. Currently, drivers also need a number of kernel services such as timer and memory allocation services. Refer to Chapter 1, “Overview of the Driver Environment,” Chapter 2, “HP-UX I/O Subsystem Features,” Chapter 3, “Multiprocessing,” Chapter 4, “Writing a Driver,” and Chapter 5, “Installing Your Driver,” for detailed discussions on WSIO and how a driver fits into the WSIO installation and configuration sequence. Refer to Chapter 1, “Overview of the Driver Environment,” Chapter 2, “HP-UX I/O Subsystem Features,” Chapter 4, “Writing a Driver,” and Chapter 5, “Installing Your Driver,” for detailed discussions on kernel services, WSIO services and SCSI services.
At the bottom of the SCSI subsystem are the transport drivers which typically directly control hardware (HBAs). These drivers directly access their hardware via Programmed I/O (PIO) read and write registers on the HBA card, and via the initiation of Direct Memory Access (DMA) to cause the HBA to transfer data to and from host memory. Refer to Chapter 12, “Writing a SCSI Interface Driver,” for additional details.

Within the SCSI subsystem, above the transport drivers, is a layer called the SCSI Services, which provides connection interface services between the SCSI Class Drivers and Transport Drivers. The SCSI Services layer defines the entry points and services that are used at the top side of the Transport Drivers to communicate up the stack (to receive requests or send completions) and at the bottom side of the Class Drivers (to send requests or receive completion notifications). In addition, the SCSI Services provides configuration and other services that are SCSI-specific, such as allocation of qtags, handling of SCSI queue depth, and probing of LUNs.

At the top of the SCSI subsystem are the class drivers, which control the individual SCSI LUNs for a given device class (disk, tape, autochanger, etc.). A generic class driver, called the SCSI pass-thru driver, can be used by SCSI-specific applications, commands, or upper level modules to build SCSI commands directly and send them to a device for vendor-specific or special control of a device, or for application control of a device class that isn’t supported by a specific class driver. Refer to Chapter 13, “Writing SCSI Device Drivers,” for additional details.

**SCSI Addressing Paradigm**

In the current HP-UX Mass Storage Stack, the SCSI Services and Class Drivers live in a SCSI-2 addressing paradigm (with up to 8 LUNs/target and up to 16 targets/bus). On transports that support larger addressing than this, the transport driver needs to create virtual buses, targets, and LUNs for use in this SCSI-2 paradigm, and translate these addresses to the larger physical target and LUN values for use on the transport. A description of how this can be done is provided.

**Virtual Bus, Target, and LUN Mapping**

Transport drivers can work around the SCSI-2 addressing restriction mentioned previously by creating virtual buses and targets which are attached to physical target I/O Tree nodes. In Fibre Channel, for example, the physical target I/O Tree node can be represented by a series of three I/O Tree nodes containing the 24-bit N-Port address of the actual FC target port (each I/O Tree node can represent up to a maximum of 8 bits of hw_path addressing). In support of this approach, the transport driver must register itself as an interface driver which claims each associated virtual bus (collection of 16 virtual targets) as a separate interface driver instance with its own isc, etc. This “virtual bus driver” must then claim the virtual bus nodes and register an appropriate probe routine such as parallel_scsi_probe, refer to the “Probing Functions” section to probe and create the necessary virtual targets and LUNs, and map them to their corresponding physical targets and LUNs internally in the “bus driver”.

For example, given a storage device attached to a Fibre Channel link containing 1024 LUNs (from 0–1023), the bus driver would create 8 virtual buses. These 8 virtual buses (which need to contain SCSI-2 style target and LUN addressing) would each contain 16 virtual targets, with 8 virtual LUNs per virtual target, for a total of 1024 virtual LUNs across the 8 virtual buses. The mapping can be simple; e.g., virtual bus 0, target 0, LUNs 0–7 maps to physical LUNs 0–7 in the storage device, virtual bus 0 target 0, LUN 1 maps to physical LUNs 8–15; virtual bus 1 maps to physical LUNS 128 to 255, and so on. If the storage device doesn’t have a contiguous LUN range the virtual LUNs corresponding to holes in the range (non-existent LUN addresses) are not created and virtual LUN range is similarly non-contiguous.
WSIO I/O Tree Management Functions

In HP-UX 11i v1 WSIO provides functions for transport drivers to use to, create and manage driver-specific I/O Tree nodes. These functions can be used by the transport driver to create child nodes of the HBA node such as the physical target nodes and virtual bus nodes. The virtual bus nodes need to get created by the transport driver as WSIO interface nodes so that they can be subsequently claimed by the transport driver’s virtual bus driver. To facilitate this, the virtual bus node would normally need to be in a subtree rooted at the HBA node, with the physical target nodes between the two as parents of the virtual bus node and children of the HBA node.

These child nodes can be created by calling the `wsio_create_interface` function, passing the HBA’s `isc` as the parent parameter, and the relative hw path of the corresponding physical target or virtual bus node. The type parameter can be `WSIO_TRANS` (transparent node) for the physical target nodes, and would need to be `WSIO_INTERFACE` (interface node) for the virtual bus nodes.

Probing Functions

To instantiate the SCSI-2 style buses, targets, and LUNs, the transport driver registers a probe routine with WSIO via a call to `wsio_register_addr_probe()` from the driver’s install routine, refer to “Writing a `driver_install()` Routine” and “Writing Driver Probe Routines” in Chapter 4, “Writing a Driver.” This probe function does probing of LUNs at the SCSI Services level (i.e., using SCSI-2 style bus, target, and LUN values), which are then translated by the transport driver to its physical values, as described.

The SCSI Services routine `parallel_scsi_probe` provides a generic LUN probing routine for use in this paradigm, and therefore currently most of the various SCSI transport drivers, whether pSCSI or Fibre Channel, register this routine for this purpose.