Models for Integrating UDI and I_2O

As Figure 3 shows, I_2O OSMs can be implemented as part of the UDI stack. Performance becomes a concern whenever a new layer is added to a software stack, but if the performance impact of running UDI as an extra layer is negligible, then running I_2O under UDI adds an extra dimension of portability. Along with allowing non-I_2O adapters and drivers to coexist with I_2O, it also ensures that older operating system module types can be upgraded across all operating system varieties, and that new module types can be developed and introduced by third parties.

Over the long run, UDI may supplant the native I/O system, as shown in Figure 4. This eliminates any "layers of performance" issues as well as any translation overhead. It gives driver developers the kind of universal driver support in the UNIX environment that they now experience in the Windows NT environment; that is, the assurance that a single driver will run across all UNIX platforms and operating system variants.

This provides maximum portability for devices written to either the UDI or I_2O specifications.

The availability of both UDI and I_2O creates a win-win situation for hardware and operating system vendors as well as for businesses and users.

Operating system vendors can reduce their development costs and ensure a wide range of device support for the operating systems without having to undertake the expense of developing the actual drivers.

Platform and hardware device vendors can create portable, easily maintainable drivers when they choose I_2O and UDI, while at the same time reducing their development and validation costs.

IT benefits by having improved interoperability and fewer and shorter qualification cycles for new devices and hardware. In addition, the availability of industry-wide driver and I/O specifications should work to increase the number, range and variety of hardware devices.

Through its participation in the I_2O SIG and Project UDI, Intel will work with other industry leaders to create maximum synergy between I_2O and UDI and help realize the benefits of these two different yet complementary approaches to device portability and high-performance I/O.

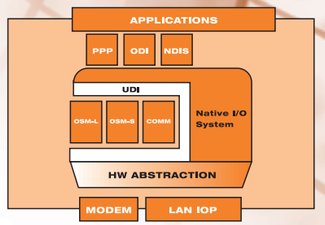


Figure 3
Greater portability: Incorporating the I_2O OSM into the UDI stack.

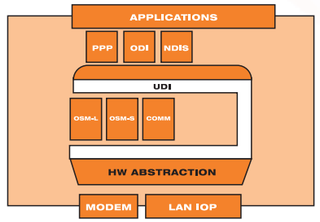
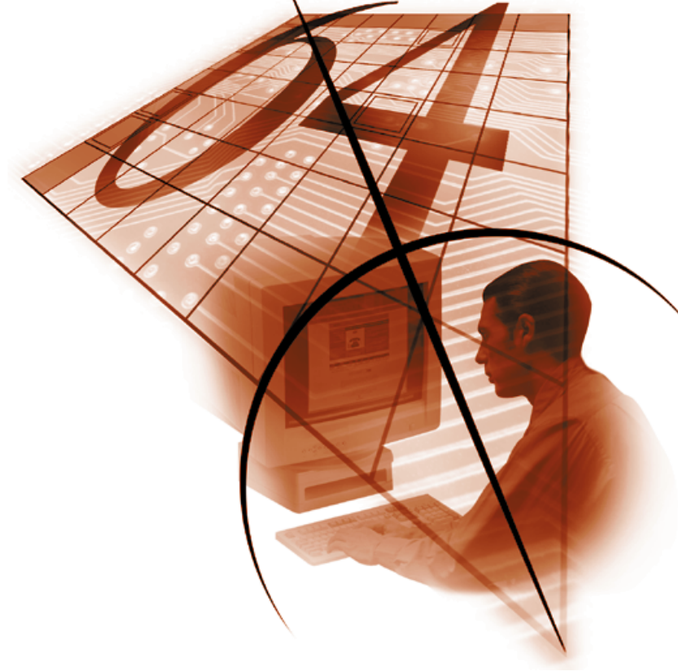


Figure 4
Complete I_2O/UDI integration: UDI to replace the native I/O system.

Part # 284095-001



UDI and I_2O™:
Complementary Approaches To Portable, High-Performance I/O

An Intel White Paper

Input and output devices on computers, such as storage or network adapters, are accessed and controlled by the operating system through device-specific software known as device drivers. Recently, these I/O devices have become more intelligent and sophisticated and, consequently, their drivers have evolved from simple routines into highly complex, multilayered programs. Because each operating system has its own unique I/O system and device interface, hardware vendors have been faced with the challenge of creating a different device driver for each operating system (including each variant of the UNIX* operating system) that they want their devices to support.

Creating High-Performance Drivers for all I/O Devices

Two industry efforts are focused on standardizing platform and operating system I/O interfaces with the aim of increasing device driver portability, efficiency and performance while reducing the maintenance burden for device driver developers. Project UDI is developing the Uniform Driver Interface primarily focused on the UNIX* operating system vendors, and the Intelligent I/O (I_2O) Special Interest Group (SIG) is creating a specification for intelligent I/O solutions. More information about UDI can be found on the Internet at http://www.sco.com/udi. For further information about I_2O, visit the SIG Web site http://www.i2osig.org.

These two projects are complementary efforts that together will benefit the industry, business Information Technology (IT) departments and users by promoting portable, high-performance drivers for the full range of I/O devices.

UDI: Portability Across Operating Systems

Figure 1 shows how UDI is oriented toward a broad array of devices and has the goal of enabling the driver for any given device to work across different platforms and operating environments, particularly across various implementations of the UNIX* operating system. This effort has yielded an architecture for device drivers that provides standard interfaces to which they can be written. In effect, OS-to-driver communications have been abstracted to facilitate portability. While the driver source remains the same, recompilation for platform- and OS-specific considerations are sufficient to make that driver portable from one environment to the next.

UDI is developed through the cooperation of major industry forces such as SCO, Hewlett-Packard, Compaq (Digital), Adaptec, Interphase, IBM, Sun Microsystems and other participating companies. Intel has recently joined this effort. The UDI specification is freely available to the industry and provides standards for device driver developers on a wide range of system services, as well as interfaces for intra-driver communication. As shown in Figure 1, it facilitates driver encapsulation for portability across various hardware and operating system environments.

UDI uses meta-languages to describe the support requirements for specific device types. For example, meta-languages to support SCSI and network adapter devices have been defined. Other meta-languages will be defined to support additional device types.

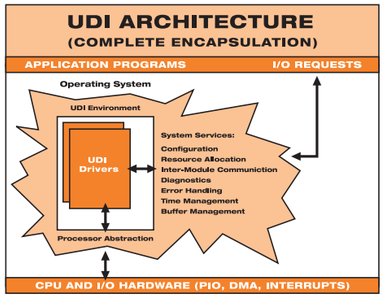


Figure 1
UDI architecture

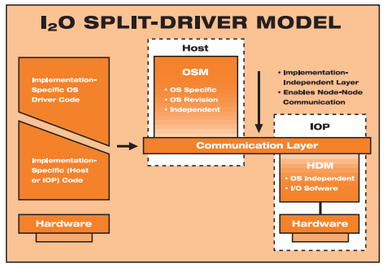


Figure 2
I_2O split-driver model

I_2O: Intelligent, Distributed I/O

I_2O technology, developed by the I_2O SIG, is a cornerstone of intelligent, high-performance distributed I/O for Intel architecture, including Intel's forthcoming IA-64 processors. The objective of I_2O is to provide a standards-based approach that complements existing drivers and offers a portable framework for the rapid development of a new generation of portable, I/O solutions.

I_2O is focused on intelligent, high-performance I/O subsystems, in which an I/O processor (IOP) offloads much of the work of controlling an I/O device. IOPs improve throughput by handling tasks such as buffering and transferring data and handling interrupts. I_2O supports message passing between multiple intelligent I/O processors.

Figure 2 shows the difference between a traditional monolithic device driver and the I_2O model with a split-driver architecture that logically divides a driver into two parts:

- The operating system services module (OSM), which interfaces to the host operating system I/O infrastructure.
- The hardware device module (HDM), which generally runs on the IOP and interfaces with the device itself. The two modules communicate through a message-passing communications layer that provides a set of application programming interfaces (APIs) for delivering messages and a set of routines that manage and process them. I_2O defines a consistent API interface downwards to the IOP and I/O adapters and a communication architecture that spans multiple operating environments and native I/O systems, including Windows® NT, NetWare® and UNIX operating systems. For any given operating system, just one operating system module is needed for each class of I/O devices (one for SCSI devices, one for network adapters, etc.) and this module is generally written by the operating system vendor. As a result, if the OSM already exists for that class of devices, the hardware vendor need only create a hardware device module, and the vendor's device can run on any operating system that includes an OSM for that class of device.

[1]The Computing Appliance Market, 1998-2002. Published 1998.

Complementary Approaches

I_2O and UDI can easily coexist, giving hardware developers maximum portability for their devices and a chance to choose whichever interface best matches the characteristics of their device:

- UDI is the driver API specification of choice for non-intelligent devices (that is, those that don't have an IOP), as well as for intelligent devices. It is also preferred for devices that stream their data.
- I_2O is the driver architecture of choice for intelligent storage and network devices with an IOP. It is also the logical choice for devices that would benefit from an IOP and for which the target operating system has an available OSM.