

Appendix C : Device Interface Classes

Device interface classes are the means by which drivers make devices available to applications and other drivers.

I: Introduction to Device Interfaces

Any driver of a physical, logical, or virtual device to which user-mode code can direct I/O requests must supply some sort of name for its user-mode clients. Using the name, a user-mode application (or other system component) identifies the device from which it is requesting I/O.

In Windows NT® 4.0 and earlier versions of the NT-based operating system, drivers named their device objects and then set up symbolic links in the registry between these names and a user-visible Win32® logical name.

For Windows® 2000 and later, drivers do not name device objects. Instead, they make use of device interface classes. A device interface class is a way of exporting device and driver functionality to other system components, including other drivers, as well as user-mode applications. A driver can register a device interface class, then enable an instance of the class for each device object to which user-mode I/O requests might be sent.

Each device interface class is associated with a GUID. The system defines GUIDs for common device interface classes in device-specific header files. Vendors can create additional device interface classes.

For example, three different types of mice could be members of the same device interface class, even if one connects through a USB port, a second through a serial port, and the third through an infrared port. Each driver registers its device as a member of the interface class GUID_DEVINTERFACE_MOUSE. This GUID is defined in the header file ntddmou.h.

Typically, drivers register for only one interface class. However, drivers for devices that have specialized functionality beyond that defined for their standard interface class might also register for an additional class. For example, a driver for a disk that can be mounted should register for both its disk interface class (GUID_DEVINTERFACE_DISK) and the mountable device class (MOUNT_DEV_MOUNTED_DEVICE_GUID).

When a driver registers an instance of a device interface class, the I/O Manager associates the device and the device interface class GUID with a symbolic link name. The link name is stored in the registry and persists across system boots. An application that uses the interface can query for instances of the interface and receive a symbolic link name representing a device that supports the interface. The application can then use the symbolic link name as a target for I/O requests.

II: Register Device Interfaces in a Driver

IoRegisterDeviceInterface registers a device interface class, if it has not been previously registered, and creates a new instance of the interface class. A driver can call this routine several times for a given device to register several interface classes and create instances of the classes. A function or filter driver typically registers device interfaces in its *AddDevice* routine.

If the device interface class has not been registered previously, the I/O Manager creates a registry key for it, along with instance-specific persistent storage under the key.

A driver registers an interface instance once and then calls *IoSetDeviceInterfaceState* to enable and disable the interface.

Registered interfaces persist across operating system reboots. If the specified interface is already registered, the I/O Manager passes its name in `SymbolicLinkName` and returns the informational success status `STATUS_OBJECT_NAME_EXISTS`.

Most drivers use a NULL reference string for a device interface instance. If a driver uses a non-NULL reference string, it must do additional work, including

possibly managing its own namespace and security.

Callers of this routine are not required to remove the registration for a device interface when it is no longer needed. Device interface registrations can be removed from user mode, if necessary.

Callers of *IoRegisterDeviceInterface* must be running at IRQL = PASSIVE_LEVEL in the context of a system thread.