Microsoft Windows

Designing RAID Adapters to Work with Windows

November 9, 2006

Abstract

Much confusion exists about how Microsoft® Windows® operating systems use storage devices such as RAID adapters. This paper discusses how to design RAID hardware and drivers so that they work well with Windows and meet the requirements of the Windows Logo Program.

The intended audience for this paper is hardware and driver developers. This paper does not provide all the necessary information for developing miniport drivers. For complete details, see the Windows Driver Kit (WDK) documentation.

This information applies for the following operating systems:

- Microsoft Windows Server® 2008
- Windows Vista®
- Microsoft Windows Server® 2003

The current version of this paper is maintained on the Web at:

http://www.microsoft.com/whdc/device/storage/RAID_design.mspx

References and resources discussed here are listed at the end of this paper.

Contents

Disclaimer

This is a preliminary document and may be changed substantially prior to final commercial release of the software described herein.

The information contained in this document represents the current view of Microsoft Corporation on the issues discussed as of the date of publication. Because Microsoft must respond to changing market conditions, it should not be interpreted to be a commitment on the part of Microsoft, and Microsoft cannot guarantee the accuracy of any information presented after the date of publication.

This White Paper is for informational purposes only. MICROSOFT MAKES NO WARRANTIES, EXPRESS, IMPLIED OR STATUTORY, AS TO THE INFORMATION IN THIS DOCUMENT.

Introduction

This paper discusses the proper design of RAID adapters—hardware and drivers—so that they work well with Microsoft® Windows® and meet the requirements of the Windows Logo Program. This paper does not address external storage subsystems, only adapters on an internal bus such as PCI that provide some level of RAID functionality.

In this paper, a hardware RAID implementation is defined as one that has a dedicated processor on the adapter that performs RAID functionality at any RAID level. Software RAID performs many of the same functions by using the system CPU and keeps its RAID logic in the device driver. Unless otherwise noted, this paper treats both hardware and software implementations the same as to how the Windows operating system sees the exposed devices. Therefore, both implementations have the same requirements.

Much confusion exists about how Windows uses storage devices such as RAID adapters. The information in this paper will help you to design products that work properly, avoiding loss of user data.

The intended audience for this paper is hardware and driver developers. This paper does not provide all the necessary information for developing miniport drivers. For complete details, see the Windows Driver Kit (WDK) documentation.

Hardware and BIOS Requirements

Three key considerations must be made for hardware and firmware: PCI Class Code, Subsystem ID, and write caching, as discussed in this section.

PCI Class Code

The most important specific hardware requirement for both hardware and software RAID implementation is proper setting of the PCI Class Code. Windows requires that this be set to the RAID class code (010400—see the Conventional PCI Specification 2.3 or later), regardless of the interconnect technology.

When the incorrect class code is used, Windows can load the wrong driver, which often results in data corruption such as destruction of the on-disk data structures that define the RAID layout. In a RAID set of two or more disks, all of the data is destroyed in such cases. Using the RAID class code ensures that Windows does not load a generic driver but instead always selects a specific driver as the better match. If a driver for your product is not included with the Windows product, the user must provide this whenever an upgrade or clean installation is performed.

If your product can operate in a non-RAID mode and operate with an industry-standard controller definition such as IDE or AHCI, then you must provide a mechanism for the user to select this mode when the system starts. This mechanism can be provided either through the system BIOS or through the extension ROM BIOS. When a user selects a non-RAID mode, a standard class driver can load on the product and provide out-of-box support for the attached storage. If your adapter requires a unique driver regardless of the mode, you must use the RAID class code. Otherwise, data corruption occurs.

Do not use either the IDE class code (0101) or AHCI (010601) unless your product can run with the standard in-box class drivers and has no data areas that must be protected, such as RAID format.

Why this is important: Using the wrong class code is a frequent cause of data corruption because generic class drivers might load, resulting in corruption of partitions and user data.

Subsystem ID

The second most important consideration is the use of unique PCI subsystem device identifiers (SIDs). A common problem is the use of a standard chipset but with different modes of operation implemented—RAID or conventional controller support. If your solution can switch modes, you must implement a unique SID for each mode and also adjust the PCI class code accordingly. The use of a unique SID prevents unexpected driver ranking problems.

If a user selects a different mode through the BIOS setup and changes class code or subsystem ID, problems will result if the system has already been installed. If a different driver is required (as would be expected), the system will likely fail to boot. In this case, the user must restore the original BIOS setting.

Why this is important: If a unique SID is not implement, an incorrect driver might be loaded, which can lead to data corruption.

Write Caching

Write caches can be implemented in Windows by using several policies. The two most common are:

- Writeback caching, where disk writes are saved in memory and committed to media at a later time (allowing optimization).
- Writethrough caching, where the writes can be saved in cache memory but write operations are not acknowledged until the disk has committed the data to media.

If your adapter provides onboard write cache memory, you must ensure the following:

- Provide battery backup if the cache implements a writeback policy.
- Set the CachesData member of PORT_CONFIGURATION_INFORMATION to TRUE.
- Respond to the SCSI MODE SENSE (6) command for Page 8 with the WCE bit set or cleared depending on whether the cache is enabled.

Otherwise, in the event of power loss, data will not be consistent in the on-disk data structures that the file system or applications use and the user's data will be lost. If the solution does not have battery backup, the writethrough cache policy can be implemented. Failing to set the CachesData value to TRUE prevents flush requests from being passed down when the system changes power states, which can lead to data corruption (SCSI SYNCHRONIZE CACHE commands will still be sent at appropriate times if WCE is set).

If RAID is implemented in a software component such as a miniport driver, write caching can be implemented only by using a writethrough policy. Using writeback caching in this situation puts users' data at risk. In the event of power failure or system crash, a cache that is maintained in system memory will be lost and both user data and file system metadata can become corrupted.

Disks attached to a controller or driver that implements write caching must also be set not to cache the data. This avoids problems with caches on different physical devices becoming out of synchronization.

Why this is important: Using system memory or other unprotected memory such as RAM on a RAID adapter leads to data corruption on disk. NVRAM or battery-backed RAM is the best alternative for implementing write caching unless your design can guarantee writethrough to media.

Additional requirements for caching adapters are listed in "Specific SCSI Command Requirements" later in this paper.

Driver Requirements

Windows supports the use of SCSIport or Storport miniports for storage adapters, with exceptions only for industry-standard ATA or SATA interfaces. Storport was introduced with Microsoft Windows Server® 2003. Storport has many advantages for storage vendors, including the ability to support much higher I/O rates and to control the device queue from the miniport.

For information about Storport, see the white paper *Storport in Windows Server 2003: Improving Manageability and Performance in Hardware RAID and Storage Area Networks*. (References are provided at the end of this paper.)

Regardless of which port driver your miniport links against, the requirements in this paper are the same and your driver must follow the requirements in this paper.

Miniport Considerations

All commands are sent to miniports in the form of SCSI Request Blocks (SRBs). Within the SRB are a number of important fields that the miniport must correctly implement, including:

- The SCSI command descriptor blocks (CDBs)
- Sense information, which contains details about errors
- Status codes for both SCSI and the SRB itself
- Data buffer pointers (if a request is mapped)
- Transfer lengths

Because all commands are sent in the form of SCSI requests, RAID implementations must correctly emulate the SCSI protocol requirements, regardless of the actual interconnect to the disk devices. Current Windows versions use the SCSI Command Sets defined by the INCITS T10 committee (previously referred to as SCSI-3).

Important: RAID implementations must follow the SCSI Command Sets prescriptively because failure to do so will result in system problems including crashing. Specifications cited in this paper include SAM-3, SBC-2, and SPC-3.

Although the following section describes some SCSI commands in detail, you should follow general considerations in the SCSI Command Set for all possible SCSI commands. Because a service, driver, or user application can create any type of SCSI request and pass it to the miniport driver, that driver must decode the command and execute or fail it with appropriate status. Because a RAID adapter is both a host bus adapter and a SCSI target, its driver is responsible for ensuring that it behaves correctly as a target.

Note: The SCSI Compliance Test in the WDK must be run against any RAID adapter, SCSI, serial attached SCSI (SAS), iSCSI, or Fibre Channel block storage target.

Queuing Considerations

If the RAID device supports multiple commands, the miniport must set the TaggedQueuing field of the PORT_CONFIGURATION_INFORMATION to TRUE. Also, the CMDQUE bit of the standard INQUIRY data must be set to 1. If these bits are not set, then the disk class driver will not set the SRB_FLAGS_QUEUE_ACTION_ENABLE flag. This will result in all requests being marked as untagged.

In addition, if a RAID miniport supports more than one outstanding request per logical unit (LU) or virtual disk, the miniport must control the queue depth.

- For SCSIport miniports, the NumberOfRequests registry setting controls the queue depth for the entire adapter.
- With Storport, the queue is per LU and the miniport must set the depth to an appropriate value by using StorPortSetDeviceQueueDepth. Storport miniports also can freeze and thaw the individual device queues or do so at the adapter level. (Storport does not use the NumberOfRequests setting.)

Why this is important: Failing to set the device queue depth and attempting to use SCSI Busy status will result in poor performance and increased CPU usage. Changes to Storport's handling of Busy and Task Set Full will further impact performance.

General SCSI Command Considerations

Because all RAID adapters receive SCSI commands, they must respond in the same way as a SCSI target such as a hard drive would. Depending on the design of the product, these SCSI commands can be executed in firmware, hardware, the miniport driver, or some combination. When a command is completed, all the following fields must be updated:

SrbStatus
ScsiStatus
SenseInfoBuffer
SenseInfoBufferLength
DataTransferLength

This section addresses behavior of SRB_EXECUTE_SCSI; other SRBs are described later in this paper.

SrbStatus

The miniport must set this status based on its ability to complete a particular SRB. This paper does not address all possible values or conditions, but if a SCSI command is not implemented or cannot be executed by the addressed device, then:

- The SrbStatus must be SRB_STATUS_ERROR.
- The ScsiStatus field must indicate a correct SCSI status value (Check Condition).
- The Sense data must match the documented values described in SPC-3.

SRB_STATUS_BUSY should not be used to control the queue depth for RAID adapters. For pausing and resuming I/O on a per-logical unit number (LUN) or per-adapter basis, see the earlier discussion of StorPortSetDeviceQueueDepth and the Storport topics in the WDK documentation.

Note: Storport initializes SrbStatus to SRB_STATUS_SUCCESS (the documentation states otherwise). The miniport must modify this value in its

HwStartIo or HwBuildIo routine unless the command is actually completed on return from that routine.

ScsiStatus

This one-byte field has a limited set of return values, not all of which can be processed by the Windows storage stack. The most important ones for this discussion are the following:

BUSY (8h)
CHECK CONDITION (2h)
GOOD (0h)
RESERVATION CONFLICT (18h)

CHECK CONDITION is an expected value when a command cannot be executed, either because of a syntax error in the command itself or because the device cannot perform the command. RAID adapters must not arbitrarily return a CHECK CONDITION to avoid implementing a required command per the logo requirements. In a CHECK CONDITION, valid Sense Data must be returned. For details, see Section 4.5 in SPC-3.

Sense Data

Sense data should be filled in only when a command completes with a SCSI error, as described in later sections of this paper. Otherwise, the driver should not touch this buffer and it should zero out the SenseInfoBufferLength value. There are exceptions to this based on the SCSI Architecture Model.

The Sense Data buffer is a minimum of 18 bytes in length. The specific fields within this buffer are described in Section 4.5 of SPC-3. For all devices, it is now assumed that sense data is returned automatically on an error condition (autosense); therefore, set SRB_STATUS_AUTOSENSE_VALID and fill in the sense buffer properly.

If the RAID adapter does not implement a particular optional command, the correct response is INVALID COMMAND OPERATION CODE. The SENSE KEY must be 05h, with an Additional Sense Code (ASC) of 20h and Additional Sense Code Qualifier (ASCQ) of 0—often written as 05/20/00. If the command code itself is supported but an optional parameter would prevent the execution of the data, the correct response is INVALID FIELD IN CDB, or 05/24/00.

DataTransferLength

For all commands—including INQUIRY and MODE SENSE—the driver must update the DataTransferLength field to match the actual number of bytes of *valid* data copied to the DataBuffer. Except for READ commands, this is almost always less than the value that was passed in as the maximum buffer size. The SCSI Compliance Test validates this field based on information in the data buffer, such as the ADDITONAL LENGTH field in the returned INQUIRY data structure.

Why this is important: Windows components execute error paths based on information returned in the SRB. A driver's failure to set correct status is a leading cause of incorrect error recovery. Incorrect status also leads to upper-level storage components and adapters to misinterpret the information. This can also easily lead to buffer overruns.

Specific SCSI Command Requirements

In addition to the general SCSI requirements for the common SRB fields, SCSI commands must be executed properly. The Windows Logo Program document lists many other commands and their requirements.

The SCSI Compliance Test was created to test the behavior of various SCSI commands that are sent to peripheral devices. The test can also be used as a standalone development tool. Each test assertion has a description of the standard and a pointer to the section that is being tested. When you use this as a standalone test, you can enable higher levels of verbose output by using the **–v** command-line option. Levels 2 and 3 are most useful. Because this test relies on proper setting of the BusType parameter, make sure that your INF sets it correctly to RAID.

To use the SCSI Compliance Test as a standalone test

- Switch to the WDK binary directory and run the following at the command prompt:

  ```
  Scsicompliancetest –v 3 \\.\PhysicalDrive n
  ```

  where *n* is a physical disk number of a LU that your RAID adapter exposes.

INQUIRY

Windows uses the INQUIRY command extensively for device discovery, so this command is extremely important to implement properly. There are two forms: Standard Inquiry Data and Vital Product Pages.

Standard Inquiry Data is a minimum of 36 bytes, as described in Section 6.4.2 of SPC-3. RAID adapters need only implement the minimum number of bytes, but they must emulate SPC-2 or later formats correctly (Version >= 4). RAID adapters must expose direct access storage devices as type 0 and can also expose management LUs as Processor (3) or Array Controllers (0ch, preferred). For details, see “INFs for Storage Miniports” later in this paper.

Reading SPC-3 carefully will help you to avoid common problems with standard Inquiry data. Fields that are used include the following:

HISUP (optional for RAID)
PERIPHERAL DEVICE TYPE
PERIPHERAL QUALIFIER
PRODUCT ID
PRODUCT REVISION LEVEL
RESPONSE DATA FORMAT (must be 2)
VENDOR ID
VERSION (must be 4, 5 or 6)

For Vital Product Data—when an INQUIRY command is sent with the EVPD bit set to 1—the data must be returned in the correct format. Do not simply return Standard Inquiry Data; examine the requested page code and respond appropriately. Required formats are:

Device Identifiers Page (83h)
Supported VPD Pages (0)
Unit Serial Number Page (80h)

Page 80h is straightforward and can return either a serial number for the adapter or a LU-specific serial number in ASCII text only. For details, see Section 7.6.10 of SPC-3.

The Device Identification VPD page is much more complicated because it can contain one or more identifiers. These identifiers absolutely must be correctly formatted to avoid buffer overruns, and at least one LU-specific identifier of type 2 or type 3 must be returned. The SCSI Compliance Test will help you to determine the validity of this page. You can enable more information by using the **–v 2** or **–v 3** option when running the test as a standalone development tool.

Note: Failure to return the same serial number or descriptor values on subsequent system boots will result in Windows license check failures. These values must remain constant.

MODE SENSE (6)

Windows makes extensive use of the 6 byte MODE SENSE command. For correct formation of the mode parameters, see Section 7.4 of SPC 3. It is very common to see malformed data. Also, remember to set DataTransferLength correctly and obey the Disable Block Descriptors bit.

All devices must support page 3fh (all pages), but some devices can have more than 255 bytes of data if all pages were to be returned. In this case, you can set a data underrun condition and support MODE SENSE (10) for this page.

The most important page for RAID adapters to return is the Caching Mode Page (8h). Except for page 3fh, all others are optional. Drivers should return a Check Condition, 05/24/00, if an unsupported page is requested. For details, see Section 6.3.3 of SBC-2.

Within the Caching page, only the WCE and CACHE SEGMENT SIZE fields are used, but setting them correctly allows Windows to optimize data transfers and implement proper cache behavior because Windows interprets WCE to mean a writeback cache. Windows uses Page Codes 0 and 1 (current and changeable), so make sure that these are both accurate.

The mode page header is also important because Windows uses the DPOFUA to determine whether writethrough commands can be supported and it uses WP to determine whether a device is writeable.

MODE SENSE (10)

See the preceding entry. This is mandatory only if the request for all pages (3fh) fails because the combined data in all pages exceeds 255 bytes.

MODE SELECT (6)

Caching Page is the only page that is used, and this should be implemented only if your device allows modifying write cache enable. RAID adapters rarely implement this capability because finer control over cache behavior is typically enabled by vendor-unique utilities. You can fail this command with 5/20/00 or other appropriate sense values if your device does not support the command or changing the parameter that is being modified.

READ (10)

Most data transfer in Windows uses the 10 byte READ command.

READ (6)

This command is no longer used by Windows code, but it can be sent by using a SCSI Passthrough command. If implemented, it must perform as expected. If not implemented, the RAID adapter must return 05/20/00 as the status (see the preceding entry).

READ CAPACITY

This command must be implemented correctly. Complete support for sector sizes other than 512 bytes (specifically 1024, 2048, and 4096) is limited to Windows Vista and later.

REQUEST SENSE

All devices must implement this command. The most recent sense data is returned in the data buffer and the DataTransferLength set accordingly. It is not permitted to fail this command or return Sense Data except in the data buffer. If the Sense Data contains error information that was previously returned, the Valid bit should be cleared.

REPORT LUNS

If a RAID adapter does not support multiple LUs per target, this command is optional. If the adapter does support multiple LUs, REPORT LUNS must be supported and the data returned must consist of single-level LU numbers less than 256. No other format is supported. For details, see Table 2 in Section 4.9.4 of SAM-3.

If the adapter does not support multiple LUs, fail this command with 05/20/00.

WRITE (10)

Most data transfers to a LU of less than 2 terabytes (TB) use the WRITE (10) command. If the LU contains a writeback cache, support of the FUA bit is mandatory and must cause data to be committed to media before the command completes.

WRITE (6)

This command is no longer used by Windows code for fixed disks, but it can be sent by using a SCSI Passthrough command. If implemented, it must perform as expected. If the command is not implemented, the RAID adapter must return 05/20/00 as the status (see the preceding entry).

Support for Large LUNs (>2TBs) (optional)

If a RAID adapter supports LUs larger than 2 TBs, 16-byte (instead of 10 byte) READ and WRITE commands are used.

For details and additional requirements, see the white paper *Large Logical Unit Support and Windows Server 2003 SP1*.

SRBs

In addition to SRB_EXECUTE_SCSI, the following are mandatory:

SRB_FUNCTION_RESET_BUS
SRB_FUNCTION_RESET_DEVICE (Storport only)
SRB_FUNCTION_RESET_LOGICAL_UNIT (Storport only)

Important: The RESET functions must be properly completed. For information on handling these commands, see the WDK documentation.

You can also choose to implement the following:

SRB_FUNCTION_FLUSH
SRB_FUNCTION_SHUTDOWN
SRB_FUNCTION_IO_CONTROL

Power and Plug and Play SRBs

The Storport driver now forwards a device power I/O request packet (IRP) to the miniport as SRB_FUNCTION_POWER requests and forwards Plug and Play IRPs to the miniport as SRB_FUNCTION_PNP requests. Miniports can optionally handle these requests for the adapter and take appropriate action.

Unlike the HwStorAdapterControl function, the SRB_FUNCTION_POWER and SRB_FUNCTION_PNP requests can be completed asynchronously. Therefore, any time-consuming operation that must be done during device start or stop time can be done when these requests are received.

Unknown SRBs

A miniport might be sent an unknown or malformed SRB at any time. A miniport must check the SRB function value first before trying to interpret any other field in

the SRB. If the function code is not known, the SRB must be failed immediately with SRB_STATUS_INVALID_REQUEST.

Note that if the SRB Function is SRB_FUNCTION_EXECUTE_SCSI, unsupported CDBs should be failed with appropriate ScsiStatus and sense data as noted earlier.

IRQL, DPCs, and Watchdog Timeouts

Generally, Storport calls the miniport driver at elevated interrupt request level (IRQL)—that is IRQL >= DISPATCH_LEVEL. In these calls, miniport drivers should not spend more than 30 seconds. The FindAdapter routines are called at PASSIVE IRQL except during crashdump and hibernate. Miniports might need to check whether they are the dump miniport before taking actions that must be performed at PASSIVE IRQL. For more information about crashdump support in miniports, see the WDK documentation.

The Storport driver also uses deferred procedure calls (DPCs) to perform several operations that call miniport driver routines such as the timer routine or adapter control routine. In these calls, the miniport should limit the amount of time spent in the DPC and in no case spend more than 1.5 seconds.

The WDK documentation states that the StorPortStallExecution() or ScsiPortStallExection() routines should be called to stall the process for less than 1 millisecond. However, best practice dictates that these routines—which call KeStallExecutionProcessor()—not be used for delays greater than 100 microseconds. For an example, see the white paper *Device-Driver Performance Considerations for Multimedia Platforms*.

However, frequently miniport drivers call the KeStallExecutionProcessor function in a loop to stall the execution for several seconds—or minutes in some cases. This delay causes the Windows DPC watchdog timer to be evoked when running with an attached debugger. When a system is running without a debugger, stalling the processor prevents other work from occurring and the system is unresponsive—for example, the mouse will not move.

IOCTLs

Many storage IOCTLs depend on correct behavior of the miniport and adapter. The Storioctlstest sends malformed IOCTLs down to the port driver or miniport. This test has detected many failures and serves as a reminder that error handling is as important as correctly implementing supported commands. This test has been added to the WDK as a convenience; it will become mandatory in the future.

All other miniport IOCTLs are forwarded to the driver for processing. Your driver must parse the signature field to see if the IOCTL is something that the driver supports. If your driver does not support the particular function, fail it with SRB_STATUS_ERROR. Do not assume that the IOCTL code is enough to determine that this is an IOCTL that you can process.

In addition to failing unsupported IOCTLs, you must validate the information for any IOCTL that you do support. Do not assume that the IOCTL structures are formed properly.

Why this is important: Many applications send IOCTLs to all miniports. If you do not properly handle the IOCTL, Windows creates a bug check. Also, if you do not properly check the information fields and buffer lengths, a bug check or memory corruption is highly likely. In some cases, this can open the system to attack.

INFs for Storage Miniports

The miniport write must take into consideration several INF requirements.

Setting BusType

The operating system makes many decisions based on the characteristics of the storage interface, which is obtained by using the IOCTL_STORAGE_QUERY_PROPERTY API. To determine this interface, each INF file for a storage miniport must set the correct bus type by using the BusType parameter. This is not a unique requirement for RAID, but RAID adapters must set this to the RAID bus type, not the bus that is used to connect to the disks. For the full list of supported buses, see Ntddstor.h or winioctl.h.

Why this is important: Windows has no visibility to the disk interconnect, and all implementations of RAID must use this bus type. Failure to set this prevents the correct tests from running and also allows Windows to attempt incorrect functionality.

Null or NODRV INFs

If a RAID adapter exposes a control unit, an INF can be supplied to claim this device even though no class driver applies. The following is a sample INF.

```
; This is a prototype file for a NODRV INF.
; This installs the NULL service (i.e., nothing)for a device node.
; You must edit where indicated for your actual parameters.
; No guarantee this will pass chkinf,
; So make sure you clean up anything that chkinf finds.

[Version]
Signature="$WINDOWS NT$"
Class=System
ClassGuid={4D36E97D-E325-11CE-BFC1-08002BE10318}
Provider=%MYCOMPANY%
CatalogFile=my.cat
DriverVer=08/09/2006,1.00.00

[ControlFlags]
ExcludeFromSelect = *

[SourceDisksNames]

[SourceDisksFiles]

[Manufacturer]
%MYCOMPANY%=MYCOMPANY

; These are formed using the device ID for the actual LUN exposed
; you might have processor, array, or Other
; Subject to change in later OSes
; as more device types are added to SPC-x
[MYCOMPANY]
%NULL.DeviceDesc%     = NODRV, SCSI\Other__RAID___SESDevice____
%NULL.DeviceDesc%     = NODRV, SCSI\ProcessorRAID___SESDevice____

[NODRV.Services]
;
; this is blank to just allow the install to succeed
; this is the important bit
; you will wind up with a device string in device manager
; and no banged out devnode
; Null service of course does nothing.
```

```
; You can use your device by opening the FDO via the port \\.\ScsiN
; and passthrough commands then set the bus/target/LUN
;
AddService = , %SPSVCINST_ASSOCSERVICE%   ; null service install


[strings]
MYCOMPANY             = "My Company, Inc."
NULL.DeviceDesc       = "My SES Virtual Device"

;****************************************
;Handy macro substitutions (non-localizable)
SPSVCINST_ASSOCSERVICE = 0x00000002
```

Note: Windows Vista and Microsoft Windows Server 2008 provide a default match for these device types, so it is no longer necessary to submit the NODRV INF files for signing. The only advantage is to uniquely identify your controller object. This has no impact on your ability to send passthrough commands to this “device” by using the SCSI Adapter object.

Why this is important: The NODRV INF prevents new device popups and presents the controller device without a question mark in Device Manager.

Test Requirements

In addition to the SCSI Compliance Test and Storioctls mentioned earlier and all the other WDK tests, you must develop your code by using checked builds. Your miniport, kernel, port, and hardware abstraction layer (HAL) are sufficient.

Take advantage of the extended logging that is included in the SCSI Compliance Test during the development of your products. Use the **v 3** option and run the test outside of Driver Test Manager (DTM). Failure of the test because the adapter did not handle commands properly does not result in a contingency for your product.

Resources

SCSI specifications
http://www.t10.org

Specifications in draft form can be obtained from http://www.t10.org. In particular, block storage products—including RAID adapters regardless of interconnect—are expected to comply with SAM-3, SBC-2, and SPC-3. Use the latest drafts of these specifications or obtain the published versions from Global Engineering.

Windows Driver Kit
http://www.microsoft.com/whdc/DevTools/WDK/aboutWDK.mspx

Your best source for information about writing miniport drivers is the Windows Driver Kit documentation in the MSDN Library. Because links are subject to change, start at http://msdn.microsoft.com/library and navigate to “Win32 and COM development\Windows Driver Kit”. This information is updated frequently.

Windows Logo Program
http://www.microsoft.com/whdc/winlogo/WLP30.mspx

The Windows Logo Program details specific requirements for all device types. You must meet all of the logo requirements under the general device category—which includes PCI compliance—as well as the storage requirements. Read this document first. The goal of Windows device compatibility is not to simply pass tests, but for devices to perform properly with the Windows operating system. Often, the requirements list a superset of current test capabilities. If your product ignores the full set of requirements, an audit can result in logo failure.

Device-Driver Performance Considerations for Multimedia Platforms
http://www.microsoft.com/whdc/driver/perform/mmdrv.mspx

Large Logical Unit Support and Windows Server 2003 SP1
http://www.microsoft.com/whdc/device/storage/LUN_SP1.mspx

Storport in Windows Server 2003: Improving Manageability and Performance in Hardware RAID and Storage Area Networks
http://download.microsoft.com/download/5/6/6/5664b85a-ad06-45ec-979e-ec4887d715eb/Storport.doc