Chapter 5: Driver I/O Control (IOCTL) Functions

The NetWare operating system or other NetWare Loadable Modules (NLMs) can make IOCTL requests to a driver to access the driver's standard system or custom IOCTL functions. IOCTL requests reference individual adapter cards, as opposed to the standard I/O routines that deal with devices attached to the cards. IOCTLs are designed as special calls to the driver that do not fit in the context of normal read and write I/O requests.

IOCTLPoll

When the OS issues an IOCTL request, the OS calls the driver's IOCTL notification routine (*IOCTLPoll*) to indicate to the driver that an IOCTL request has been placed on the adapter's IOCTL queue. **(NOTE: This call is made once and only once for each IOCTL.)** Entry to *IOCTLPoll* occurs at a non-blocking process level (identical to IOPoll). The driver's *IOCTLPoll* routine must do one of the following three actions:

1) Elect to postpone processing the IOCTL request. This is done by setting an indicator in the driver's control structures that an IOCTL is pending (if required), then returning to the caller.

   Later, when the OS calls an interrupt service routine (ISR) or a timer exit, the request can be initiated. This is done by calling *GetIOCTL,* validating the IOCTL request obtained, starting the requested operation (if any), then returning to the caller. Upon completion of the function, the driver places the completion status in the *IOCTLRequestStructure*, calls *PutIOCTL* to notify the OS of request completion, then returns to the caller or point of interrupt. (The driver must not wait in a sense-loop for the function to complete. This would halt all other processes, including the servicing LAN packets.)

2) Notify the caller that the IOCTL requested is not supported by the driver and was not completed. This is done by calling *GetIOCTL* to acquire the IOCTL request, placing "not-supported" status in the completion status field of the request (see *IOCTLRequestStructure* and IOCTL completion status below), calling *PutIOCTL* to notify the OS, then returning to the caller.

3) Accept and initiate the IOCTL request. Call *GetIOCTL* to obtain the request, then decode the requested function and sub-function codes. Drivers will normally use the subfunction code as an index into a jump table containing the standard IOCTL routines that are required to be supported.

   If the IOCTL function can be completed immediately, the routine must complete the required action, place the completion status in the *IOCTLRequestStructure*, then call *PutIOCTL* to notify the OS that the request is completed. Following this, the driver will return to the caller.

   If the IOCTL function initiated cannot be completed immediately, but must wait for an interrupt, the driver must now initiate the action, set indicators in the necessary structures to indicate that the current function is an IOCTL, and save the *IOCTLRequestStructure* handle. The driver should then return program control to the caller. Upon entry to the driver ISR, at the completion of the function, the driver must fill in the completion status in the *IOCTLRequestStructure*, then make a *PutIOCTL* call to notify the IOCTL caller that the requested function is now complete. The driver may then check to determine if any further I/O requests or IOCTLs can be issued, and start one, if possible. Finally, the driver must return back to the caller of the driver ISR.

   The *IOCTLRequestStructure* is outlined below:

```
typedef  struct  IOCTLRequestStructure
        {
            LONG         DriverLink;
            CardStruct   *CardHandle;
            WORD         CompletionCode;
            BYTE         Function;
            BYTE         SubFunction;
            LONG         IOCTLParameter;
            LONG         *IOCTLBuffer;
        }   IOCTLRequestStruct;
```

Figure 5-1 The IOCTL Request Structure

Each field in the IOCTL Request structure is defined below:

DriverLink — This field is used only by the driver. It can be used to link the outstanding IOCTL requests at the driver level. This field has no effect in NetWare v3.xx, v4.xx.

CardHandle — This field contains a card handle. This is the same value that AddDiskSystem returned during initialization. The field is not valid until after the IOCTL has been acquired using a *GetIOCTL*.

CompletionCode — The driver fills in this field before returning the IOCTL request structure to the application or NetWare. The possible completion codes are defined in Figure 5-2 below.

Function — This field specifies the target IOCTL routine. These routines are explained in the next section.

SubFunction — This field specifies the target subfunction relative to the target IOCTL function. Subfunctions are explained in the next section.

IOCTLParameter — This field is often used to specify the target device or disk. This is the same value as that returned by AddDiskDevice. Other values could also be passed in this field as needed. When not used, this parameter should be zero.

IOCTLBuffer — This field has variable usage. It sometimes contains request information and other times a pointer to a buffer containing request information. When not used, this parameter should be zero. See the specific IOCTL calls listed below for more information.

IOCTL Request Return Status

Drivers use the CompletionCode field in the IOCTLRequestStructure to return a completion or device status to the calling application. The status returned is a two byte code. The general set of status codes and their definitions are listed below. Valid codes for individual IOCTLs are listed in their specific definitions. In general, all IOCTLs should be completed and returned with a "No Error" status unless 1) the hardware has malfunctioned, 2) the IOCTL is state dependent and the driver/device is in an erroneous state, or 3) the IOCTL requires a status code to be returned to an application. (In the last case, the IOCTL should be completed and returned with the status code.)

Completion/Device Status returned to the calling application

No Error	0000h
Non-Media Error	0003h
Device Not Active	0004h
Adapter Card Error	0005h
Device Parameter Error	0006h
System Parameter Error	0007h
Not Supported By Device	0008h
Device Fault	0103h
No Media Present	0703h
Media Write Protected	0803h
Magazine Not Present	0F09h
Changer Error	1009h
Changer Source Empty	1109h
Changer Destination Full	1209h
Changer Jammed	1303h
Magazine Error	1409h
Magazine Source Empty	1509h
Magazine Destination Full	1609h
Magazine Jammed	1703h
Driver Custom Status	E0xxh - FExxh
Not Supported By Driver	FFF9h

Figure 5-2 IOCTL Request Return Status

Request Completion Status Codes:

No Error	The request was completed successfully.
Non-Media Error	The request was not completed successfully because an unspecified error has occurred.
Device Not Active	The device has been de-activated and is no longer functional.
Adapter Error	The driver has detected a host bus adapter failure.
Device Parameter Error	The device has detected an error in a parameter supplied by the caller.
System Parameter Error	The OS or driver has detected an error in a parameter supplied by the caller.
Not Supported By Device	The device does not support the requested function.
Device Fault	The device has failed and is no longer functional.
No Media Present	No media is present in the device.
Media Write Protected	The media is present but is write-protected.
No Magazine Present	No magazine is present in the device.
Changer Error	An unspecified media changer error has occurred.
Changer Source Empty	There is no media present in the changer source location specified in the IOCTL.
Changer Destination Full	There is media present in the changer destination location specified in the IOCTL.
Changer Jammed	The media changer mechanism is jammed.
Not Supported by Driver	The driver does not support this function, and the request has been ignored.
Magazine Error	An unspecified media magazine error has occurred.
Magazine Source Empty	There is no media present in the magazine source location specified in the IOCTL.
Magazine Destination Full	There is media present in the magazine destination location specified in the IOCTL.
Magazine Jammed	The media magazine mechanism is jammed.
Driver Custom Status	These codes are available for drivers to use to return special or custom status to associated NLMs. Use of these codes will prevent the driver

	from working with other NLMs which are not aware of the custom codes designated by the driver.
Not Supported by Driver	The driver does not support this function, and the request has been ignored.

Using GetIOCTL

The driver must "acquire" all (queued) IOCTL requests using the NetWare routine *GetIOCTL*. *GetIOCTL* is passed the *CardHandle* and a pointer to an IOCTLRequestStructure (or zero) and returns a pointer to an *IOCTLRequestStructure* that the driver may service. If the driver needs to retrieve a particular request, it can pass a pointer to that request in the nextRequest parameter (see Figure 5-3). *GetIOCTL* will return the same pointer, and the driver can then proceed with servicing the request. If the driver simply wants whichever request is next, it passes a 0 in the nextRequest parameter. *GetIOCTL* then returns a pointer to the next request. If *GetIOCTL* returns a pointer value of zero, no IOCTL request was "acquired" or available.

```
IOCTLRequestStruct  *GetIOCTL(
       CardStruct  *CardHandle,
       IOCTLRequestStruct  *IOCTLRequest);
```

Figure 5-3 GetIOCTL Syntax

To service an IOCTL request after having acquired it, the IOCTLPoll routine examines the IOCTL request structure's function field and responds by calling the appropriate function and passing it the IOCTL request.

After completing the function requested, all IOCTL requests must be returned to the OS. The OS has provided the PutIOCTL routine for this purpose.

IOCTLPoll

The IOCTLPoll routine services IOCTL requests from NetWare or other loadable modules.

Syntax

```
void IOCTLPoll(
    CardStruct  *CardHandle,
    IOCTLRequestStruct  *IOCTLRequest)
```

Return Values

None

Parameters

CardHandle Passes a pointer to the adapter card's card structure.

IOCTLRequest Passes a pointer to an IOCTLRequestStructure.

Remarks

The name of the IOCTLPoll routine is arbitrary. When the driver's InitializeDriver routine calls the NetWare routine AddDiskSystem, the InitializeDriver routine passes the address of the IOCTL Poll routine to NetWare.

IOCTLPoll Routine

```
IOCTLPoll(
          CardStruct            *CardHandle,
          IOCTLRequestStruct  *IOCTLRequest)
{
    IOCTLRequestStruct    *NewRequest;

    if (CardHandle -> status == BUSY)
    {
        ++IOCTLRequestCount;  /*Reminder that this request */
        return;                  /*needs to be serviced later. */
    }

    while (NewRequest = GetIOCTL(CardHandle, 0))
    {

        /* Check the structure's IOCTLParameter field */
        switch  (NewRequest->SubFunction)
        {
            case   ACTIVATEDEVICE:
                       .
                       .
                       .
                   break;

            case   DEACTIVATEDEVICE:
                       .
                       .
                       .
                   break;

            default:
                NewRequest->CompletionCode  =  NOT_SUPPORTED;

        }
            /*  Note:  PutIOCTL() must be called at some point following a call to
                       GetIOCTL(), but it may be called from another function (for
                       instance, while waiting for completion of an IOCTL). In this
                       case, PutIOCTL must NOT be called at this point.
            */
            if (CardHandle -> status == BUSY)
                return;


            PutIOCTL(CardHandle,  NewRequest);
    }   /*  end of while (...)    */


}   /* end of IOCTLPoll */
```

Standard IOCTL Functions

As explained in the previous section, individual IOCTL requests are specified by the function and sub-function fields in the IOCTLRequestStructure.

(Note: All IOCTLs supported in the NetWare v3.11 specification are still supported in NetWare v3.1x/v4.xx; however, some IOCTLs that were defined but never implemented have been removed or reassigned to other function and subfunction numbers. When writing new drivers or updating previous drivers, Novell recommends using IOCTLs as now defined in the NetWare v3.1x/v4.xx specification.)

Function	*Sub-Function*	
0	*0*	*Activate Device*
	1	*Deactivate Device*
	2	*Format*
	3	*Device Verify Mode*
	4	*Identify Device*
	5	*Return Bad-Block Info*
	6	*Return Device Status*
	7	*Logical Device Mount*
	8	*Logical Device Dismount*
	9	*Lock Device Media*
	10	*Unlock Device Media*
	11	*Eject Media*
1	*0*	*ReturnDeviceInfo (see old v3.11 func.0, subfunc.17)*
	1	*ReturnMediaInfo (see old v3.11 func.0, subfunc.18)*
	2	*SetDeviceParameters (see old v3.11 func.0, subfunc.19)*
	3	*ReturnTapeDeviceInfo*
2	*0*	*ReturnMagazineInfo*
	1	*(not assigned)*
	2	*ReturnMagazineMediaMapping*
	3	*MagazineSelectCommand*
	4	*MagazineDeselectCommand*
	5	*MagazineLoad*
	6	*MagazineUnload*
	7	*MagazineEject*
3	*0*	*ReturnChangerInfo*
	1	*ReturnChangerDeviceMapping*
	2	*ReturnChangerMediaMapping*
	3	*ChangerCommand*
4-63		*Reserved by Novell*
64-255		*IOCTLs for third party use. Assigned by Novell*

IOCTL Functions deleted from the new specification

0	*12*	*Return Changer Element count*
	13	*Return Changer Element Info*
	14	*Changer command*
	15	*Select Media*
	16	*Unselect Media*

Figure 5-4 v3.1x/v4.xx IOCTL (I/O Control) Routine Assignments

Function	*Sub-Function*	
0	*0*	*Activate Device*
	1	*Deactivate Device*
	2	*Format*
	3	*Device Verify Mode*
	4	*Identify Device*
	5	*Return Bad-Block Info*
	6	*Return Device Status*
	7	*Logical Device Mount*
	8	*Logical Device Dismount*
	9	*Lock Device Media*
	10	*Unlock Device Media*
	11	*Eject Media*
	12	*Return Changer Element count **
	13	*Return Changer Element Info **
	14	*Changer command **
	15	*Select Media **
	16	*Unselect Media **
	17	*ReturnDeviceInfo (see 3.1x/v4.xx func.1, subfunc.0) **
	18	*ReturnMediaInfo (see 3.1x/v4.xx func.1, subfunc.1) **
	19	*SetDeviceParameters (see 3.1x/v4.xx func.1, subfunc.2) **
1-63		*Reserved by Novell*
64-255		*IOCTLs for third party use. Assigned by Novell*

** These IOCTLs are defined in later versions of the 3.11 specification but are never issued by the NetWare 3.11 OS.*

Figure 5-5 Old v3.11 IOCTL (I/O Control) Routine Assignments

Novell has reserved IOCTL functions 4 through 63. IOCTLs 64 and up will be assigned by Novell to developers (Novell assigns certified drivers a Driver ID number. If another loadable module needs the driver to perform a special IOCTL service, this value could be used as a function number in the IOCTLPoll procedure).

Activate Device (Mandatory)

Function: 0
Sub-function: 0

This IOCTL directs the driver to activate a mounted device. The driver should return a "No Error" CompletionCode if the device is powered up and fully operational. This call may cause a previously deactivated device to be re-activated, provided that the device is now functional (the driver must recognize this call for previously deactivated drives). The driver must verify that media is present for removable devices.

Allowed CompletionCode values (in IOCTLRequestStructure):

0000h	No Error
0003h	Non-Media Error
0005h	Adapter Card Error
0006h	Device Parameter Error
0007h	System Parameter Error
0103h	Device Fault
0703h	No Media Present

IOCTL Request Structure Fields

LONG DriverLink	Driver specific usage.
CardStruct *CardHandle	Contains the card handle returned by *AddDiskSystem*.
WORD CompletionCode	The driver fills this field with a completion status.
BYTE Function	Contains a value of 0.
BYTE SubFunction	Contains a value of 0.
LONG IOCTLParameter	Contains the device handle returned by *AddDiskDevice*.
LONG *IOCTLBuffer	Not used.

Deactivate Device (Mandatory)

Function: 0
Sub-function: 1

The *Deactivate Device* IOCTL notifies the driver that a device has been deactivated by NetWare, resulting from a hardware error, HotFix error, or user request. When a device is deactivated, the driver must return all requests previously obtained (using a *GetRequest*) but not completed. This is done using *PutRequest* and a "Device Not Active" completion code.

In **NetWare v4.xx** and **NetWare v3.12** all pending requests are removed from the queue and returned with a "Device Not Active" status code by the OS. (These are requests that were not obtained by the driver using *GetRequest*.)
In **NetWare v3.11**, the driver must initiate the removal of the requests from the queue. This is done in one of two ways:

1) by putting a request back using *PutRequest* and a "Non-Media Error" completion code. This may require the driver to first obtain a request using *GetRequest*. (This method has the undesirable side effect of issuing a drive failure deactivation message to the console.)

2) by getting all the request from the queue using *GetRequest* and putting them back using *PutRequest* and a "Device Not Active" return code. (*GetRequest* will return a NULL request handle when the queue is empty.)

The driver should respond to any subsequent calls to the IOPoll routine by obtaining the request (using *GetRequest*) and returning it (using *PutRequest*) with the "Device Not Active" completion code. This procedure should be continued until an *Activate Device* IOCTL is received.

The device structure is not removed from the OS. The driver may make *AlertDevice* calls to the OS to indicate further status changes to the device. An "Activate" IOCTL may change the device to "Active" status provided that the device can resume normal function (the driver must return "No Error" status).

Possible Completion Codes (in IOCTLRequestStructure):

0000h	No Error
0006h	Device Parameter Error
0007h	System Parameter Error

IOCTL Request Structure Fields

LONG DriverLink	Driver specific usage.
CardStruct *CardHandle	Contains the card handle returned by *AddDiskSystem*.
WORD CompletionCode	The driver fills this field with a completion status.
BYTE Function	Contains a value of 0.
BYTE SubFunction	Contains a value of 1.
LONG IOCTLParameter	Contains the device handle returned by *AddDiskDevice*.
LONG *IOCTLBuffer	Not used.

The OS makes a "Deactivate" call to all registered devices prior to completing a "down" command.

Format Device (Optional)

Function: 0
Sub-function: 2

This IOCTL directs the driver to format the device. The format IOCTL should have been preceded by a "Return Device Status" IOCTL (allows removables to redefine drive geometry, access flags, and other related indicators). The driver must keep a format-busy indicator for each device, and either return "Device Not Active" status or postpone servicing all IOCTL and I/O requests until the format operation is completed.

Possible Completion Codes (in IOCTLRequestStructure):

0000h	No Error
0003h	Non-Media Error
0005h	Adapter Card Error
0006h	Device Parameter Error
0007h	System Parameter Error
0008h	Not Supported By Device
0103h	Device Fault
0703h	No Media Present
0803h	Media Write Protected
FFF9h	Not Supported By Driver

IOCTL Request Structure Fields

LONG DriverLink	Driver specific usage.
CardStruct *CardHandle	Contains the card handle returned by *AddDiskSystem*.
WORD CompletionCode	The driver fills this field with a completion status.
BYTE Function	Contains a value of 0.
BYTE SubFunction	Contains a value of 2.
LONG IOCTLParameter	Contains the device handle returned by *AddDiskDevice*.
LONG IOCTLBuffer	This field is initially supplied by the caller, and indicates the **interleave factor** to be used in formatting the drive (0=default, 1=1:1, 2=1:2, 3=1:3, etc). The driver will return the actual interleave factor used in this field. This field is used only for interleave factors. Drivers for devices which require special interleave tables to be given to the controller or drive must generate the corresponding tables internally.

Device Verify Mode (Mandatory for Read/Write Devices)

Function: 0
Sub-function: 3

This IOCTL directs the driver to set or return the status of the device's current read-after-write-verify mode. During initialization the driver must use the *GetReadAfterWriteVerifyStatus* call to obtain the default system Read-After-Write Verify mode (On or Off) and then set the mode of all supported devices accordingly. The use of software in place of hardware verification and vise versa is discretionary.

Possible Completion Codes (in IOCTLRequestStructure):

0000h	No Error
0006h	Device Parameter Error
0007h	System Parameter Error
0008h	Not Supported By Device
FFF9h	Not Supported By Driver

IOCTL Request Structure Fields

LONG DriverLink	Driver specific usage.
CardStruct *CardHandle	Contains the card handle returned by *AddDiskSystem*.
WORD CompletionCode	The driver fills this field with a completion status.
BYTE Function	Contains a value of 0.
BYTE SubFunction	Contains a value of 3.
LONG IOCTLParameter	Contains the device handle returned by *AddDiskDevice*.
LONG *IOCTLBuffer	Points to a buffer (size LONG) containing one of the following values:

0 Do not verify writes on this device
1 Perform write verification with hardware
2 Perform write verification with software
3 Return current device verify mode (no change)

Note: The driver must return the actual mode set for the device in this buffer location, which may differ from the mode initially supplied by the IOCTL caller.

Identify Device (Optional)

Function: 0
Sub-function: 4

This IOCTL directs the driver to identify a device by causing it to beep or flash its select light.

Possible Completion Codes (in IOCTLRequestStructure):

0000h	No Error
0003h	Non-Media Error
0006h	Device Parameter Error
0007h	System Parameter Error
0008h	Not Supported By Device
FFF9h	Not Supported By Driver

IOCTL Request Structure Fields

LONG DriverLink	Driver specific usage.
CardStruct *CardHandle	Contains the card handle returned by *AddDiskSystem*.
WORD CompletionCode	The driver fills this field with a completion status.
BYTE Function	Contains a value of 0.
BYTE SubFunction	Contains a value of 4.
LONG IOCTLParameter	Contains the device handle returned by *AddDiskDevice*.
LONG *IOCTLBuffer	Points to a buffer containing one of the following values:

0 Start Identifying
1 Stop Identifying
2 Identify once
3 Return current identification mode in buffer (mode set by previous identify status call in this location)

Note: The driver must return the identification mode set for the device in this same buffer location for option 3. Drivers supporting this IOCTL should retain the current identify mode in a dedicated field, and initialize the field to a value of 1.

Return Bad Block Info (Optional)

Function: 0
Sub-function: 5

This IOCTL directs the driver to returns bad block information used by HotFix when initializing a partition. Parameters are listed below.

Possible Completion Codes (in IOCTLRequestStructure):

0000h	No Error
0003h	Non-Media Error
0006h	Device Parameter Error
0007h	System Parameter Error
0008h	Not Supported By Device
0703h	No Media Present
FFF9h	Not Supported By Driver

IOCTL Request Structure Fields

LONG DriverLink	Driver specific usage.
CardStruct *CardHandle	Contains the card handle returned by *AddDiskSystem*.
WORD CompletionCode	The driver fills this field with a completion status.
BYTE Function	Contains a value of 0.
BYTE SubFunction	Contains a value of 5.
LONG IOCTLParameter	Contains the device handle returned by *AddDiskDevice*.
LONG *IOCTLBuffer	Points to a buffer containing the information as defined in the structure bbinfostruct (shown below):

```
struct bbinfostruct
  {
    LONG badblockcount              /*set by caller*/
    LONG beginningsequencenumber          /*set by caller*/
    LONG endingsequencenumber       /*set by driver*/
    struct bbstruct badblocks[badblockcount];
  }

struct bbstruct
  {
   LONG physicalsectornumber;
   LONG numberofsectors;
  };
```

Since the calling application cannot make any assumptions about the amount of bad block information returned by the driver, this IOCTL should be called in a recursive manner with updated "beginningsequencenumber" to obtain all bad block information.

Prior to making this IOCTL call, the calling application allocates memory for a bbinfostruct that will receive bad block information, sets the "badblockcount" field to indicate the size of the "badblocks" field, and initializes the beginning sequence number to zero.

The driver uses the "badblockcount" field to determine the number of bad blocks it can return. The "beginningsequencenumber" field tells the driver where to begin indexing into its bad block table. The driver fills the "badblocks" array with bad block information and updates the "endingsequencenumber" with the index for the next set of bad block information.

If the IOCTL returns with a "No Error" completion code, the calling application can determine the amount of bad block information returned by subtracting the "beginningsequencenumber" from the "endingsequencenumber". It can then process the valid entries in the "badblocks" field. If the "badblocks" array is full, the driver can obtain additional bad block information by copying the contents of the "endingsequencenumber" field into the "beginningsequencenumber" field and reissuing the IOCTL.

Return Device Status (Mandatory for Removables)

Function: 0
Sub-function: 6

This IOCTL directs the driver to return general status of a device. It is issued after a "Logical Device Mount" IOCTL to update NetWare device information (for removable media). Please note that it is not valid to change some restricted items, such as the drive type, or the access flags with the following exception: If the media is no longer writable, (i.e. the write-protect tab has been set on media in a MO unit, or the media of a WORM device has run out of space) the ReadOnlyDevice AccessFlag should be set.

Possible Completion Codes (in IOCTLRequestStructure):

0000h	No Error
0003h	Non-Media Error
0006h	Device Parameter Error
0007h	System Parameter Error
0008h	Not Supported By Device
0703h	No Media Present
FFF9h	Not Supported By Driver

IOCTL Request Structure Fields

LONG DriverLink	Driver specific usage.
CardStruct *CardHandle	Contains the card handle returned by *AddDiskSystem*.
WORD CompletionCode	The driver fills this field with a completion status.
BYTE Function	Contains a value of 0.
BYTE SubFunction	Contains a value of 6.
LONG IOCTLParameter	Contains the device handle returned by *AddDiskDevice*.
LONG *IOCTLBuffer	Points to a structure the driver must fill with information in the following format:

Buffer Structure

```
LONG  Reserved0
LONG  Reserved1
LONG  DriveTotalSize
LONG  DriveParameters
LONG  DriveSizes
LONG  EstFormatTime
LONG  Reserved2[6]
```

The fields in the buffer structure are defined as follows:

DriveTotalSize — The useable sector capacity of the physical device or media (if removeable). The sector size is as reported in the **SectorSize** field. For writeable media this value should be rounded down to a cylinder boundary (using the device geometry as reported below), since all partitions must begin and end on cylinder boundaries. For read-only media (CDROM) this value should be reported with no modifications. For sequencial access devices, if the capacity is unknown, this field should be set to a -2.

DriveParameters — For sequencial access devices, this field should be set to a -1. For all other devices, this field includes the following drive parameter fields:

```
db  SectorCount          (lsb)
db  HeadCount
dw  CylinderCount   (msw)
```

SectorCount is the device's sectors per track.
HeadCount is the device's number of heads.
CylinderCount is the number of cylinders on the device. For writeable media the SectorCount and HeadCount parameters are used by the partition editor to determine the partition boundaries and are required to match the geometry of other partitions on the drive. For read-only media, if the device capacity does not fall on a cylinder boundary, the count should be incremented to include the partial cylinder. (See DriveTotalSize.)

DriveSizes — Information about the drive size. It includes the following bytes:

```
db  AccessFlags (lsb)
db  DriveType
db  BlockSize
db  SectorSize        (msb)
```

AccessFlags indicates special device or access characteristics to be used with the device:

RemovableDevice	01h
ReadOnlyDevice	02h
WriteSequential	04h
ChangerDevice	10h *
MagazineDevice	20h *

* v3.12 & v4.xx only

RemovableDevice indicates that the device exists even if it is not currently ready or doesn't have valid media present. It also implies that the media geometry may be re-defined when a change of media occurs (determined by this IOCTL call).

ReadOnlyDevice indicates to the OS that no write calls should be issued to this device. (NetWare volumes are only supported for Read-Only devices with v3.11 and later versions).
WriteSequential indicates to the OS that the device is sequential and that writes will be done in the sequence they are issued to the device (random reads may be simulated with sequential devices).
ChangerDevice indicates to the OS that a Read/Write device associated with an autochanger is being added to the system. If this flag is set, the NetWare 4.xx or 3.12 OS will subsequently issue the appropriate IOCTLs in order to obtain the autochanger configuration.
MagazineDevice indicates to the OS that a Read/Write device associated with a magazine is being added to the system. If this flag is set, the NetWare 4.xx or 3.12 OS will subsequently issue the appropriate IOCTLs in order to obtain the magazine configuration.

The **DriveType** is defined as follows:

0 Hard disk
1 CD-ROM Device *
2 WORM Device *
3 Tape Device *
4 Magneto-Optical (MO) Device

* Note: NetWare volumes are not currently supported on these device types

BlockSize is the maximum I/O request blocksize that may be issued to the device driver.

Example (assume actual sector size = 512):

0 - 1 sector (512) 4 - 16 sectors (8K)
1 - 2 sectors (1K) 5 - 32 sectors (16K)
2 - 4 sectors (2K) 6 - 64 sectors (32K)
3 - 8 sectors (4K) 7 - 128 sectors (64K)

SectorSize:
The value inserted for **SectorSize** is actually a shift factor. The shift factor is used as the exponent in the following formula:

$512 * 2^{(sectorSize)}$ = Actual Sector Size

Currently, for devices that support NetWare volumes this must be a value of 0 which calculates to a sector size of 512. The NetWare File System only supports 512-byte sectors, and requests generated by it will be in sectors of that size. Drivers that support devices with native sector sizes other than 512 are required to make the appropriate sector translations for these requests. This restriction may be ignored for devices that bypass the NetWare File System.

EstFormatTime Driver estimated time to format media, in minutes (zero if undetermined).

Logical Device Mount (Mandatory for Removables)

Function: 0
Sub-function: 7

This IOCTL directs the driver to confirm the presence of valid media in the device and its operability (mounts the media).

Possible Completion Codes (in IOCTLRequestStructure):

0000h	No Error
0003h	Non-Media Error
0006h	Device Parameter Error
0007h	System Parameter Error
0008h	Not Supported By Device
0703h	No Media Present
FFF9h	Not Supported By Driver

IOCTL Request Structure Fields

LONG DriverLink	Driver specific usage.
CardStruct *CardHandle	Contains the card handle returned by *AddDiskSystem*.
WORD CompletionCode	The driver fills this field with a completion status.
BYTE Function	Contains a value of 0.
BYTE SubFunction	Contains a value of 7.
LONG IOCTLParameter	Contains the device handle returned by *AddDiskDevice*.
LONG *IOCTLBuffer	Not used.

Logical Device Dismount (Mandatory for Removables)

Function: 0
Sub-function: 8

This IOCTL is a notification from the Operating System that the media mounted will no longer be referenced (logically dismounts the media on a removable device). It directs the driver to unlock the media but not eject it.

Possible Completion Codes (in IOCTLRequestStructure):

0000h	No Error
0006h	Device Parameter Error
0007h	System Parameter Error
0008h	Not Supported By Device
FFF9h	Not Supported By Driver

IOCTL Request Structure Fields

LONG DriverLink	Driver specific usage.
CardStruct *CardHandle	Contains the card handle returned by *AddDiskSystem*.
WORD CompletionCode	The driver fills this field with a completion status.
BYTE Function	Contains a value of 0.
BYTE SubFunction	Contains a value of 8.
LONG IOCTLParameter	Contains the device handle returned by *AddDiskDevice*.
LONG *IOCTLBuffer	Not used.

Lock Device Media (Optional)

Function: 0
Sub-function: 9

This IOCTL directs the driver to physically lock the media in the removable device so that it cannot be manually ejected.

Possible Completion Codes (in IOCTLRequestStructure):

0000h	No Error
0003h	Non-Media Error
0006h	Device Parameter Error
0007h	System Parameter Error
0008h	Not Supported By Device
0703h	No Media Present
FFF9h	Not Supported By Driver

IOCTL Request Structure Fields

LONG DriverLink	Driver specific usage.
CardStruct *CardHandle	Contains the card handle returned by *AddDiskSystem*.
WORD CompletionCode	The driver fills this field with a completion status.
BYTE Function	Contains a value of 0.
BYTE SubFunction	Contains a value of 9.
LONG IOCTLParameter	Contains the device handle returned by *AddDiskDevice*.
LONG *IOCTLBuffer	Not used.

Unlock Device Media (Optional)

Function: 0
Sub-function: 10

This IOCTL directs the driver to physically unlock the media on a mounted removable device so that it may be ejected or removed.

Possible Completion Codes (in IOCTLRequestStructure):

0000h	No Error
0003h	Non-Media Error
0006h	Device Parameter Error
0007h	System Parameter Error
0008h	Not Supported By Device
FFF9h	Not Supported By Driver

IOCTL Request Structure Fields

LONG DriverLink	Driver specific usage.
CardStruct *CardHandle	Contains the card handle returned by *AddDiskSystem*.
WORD CompletionCode	The driver fills this field with a completion status.
BYTE Function	Contains a value of 0.
BYTE SubFunction	Contains a value of 10.
LONG IOCTLParameter	Contains the device handle returned by *AddDiskDevice*.
LONG *IOCTLBuffer	Not used.

Eject Media (Optional)

Function: 0
Sub-function: 11

This IOCTL directs the driver to eject the media in a removable device. This function will override a lock. This call is illegal for a device embedded within a autochanger (see the "Changer Command" IOCTL).

Possible Completion Codes (in IOCTLRequestStructure):

0000h	No Error
0003h	Non-Media Error
0006h	Device Parameter Error
0007h	System Parameter Error
0008h	Not Supported By Device
0703h	No Media Present
FFF9h	Not Supported By Driver

IOCTL Request Structure Fields

LONG DriverLink	Driver specific usage.
CardStruct *CardHandle	Contains the card handle returned by *AddDiskSystem*.
WORD CompletionCode	The driver fills this field with a completion status.
BYTE Function	Contains a value of 0.
BYTE SubFunction	Contains a value of 11.
LONG IOCTLParameter	Contains the device handle returned by *AddDiskDevice*.
LONG *IOCTLBuffer	Not used.

Return Device Info (Mandatory for Removables) (Recommended for all other device types)

Function: 1
Sub-function: 0

This IOCTL directs the driver to return additional information about the device. For removable devices, this IOCTL should be filled out to the extent possible and returned with "No Error", even if there is no media present.

Possible Completion Codes (in IOCTLRequestStructure):

0000h	No Error
0003h	Non-Media Error
0008h	Not Supported By Device
FFF9h	Not Supported By Driver

IOCTL Request Structure Fields

LONG DriverLink	Driver specific usage.
CardStruct *CardHandle	Contains the card handle returned by *AddDiskSystem*.
WORD CompletionCode	The driver fills this field with a completion status.
BYTE Function	Contains a value of 1.
BYTE SubFunction	Contains a value of 0.
LONG IOCTLParameter	Contains the device handle returned by *AddDiskDevice*.
LONG *IOCTLBuffer	Points to a structure the driver must fill with information in the following format:

Bytes	Field	Description
4	device type	Indicates the device type.

disk	0x00000000
tape	0x00000001
printer	0x00000002
WORM	0x00000004
CDROM	0x00000005
magneto optical	0x00000007
changer	0x00000008
multiple	0x00000009

4	device type mask	Bit map indicating the device functionality supported by the device.

disk	0x00000001
tape	0x00000002
printer	0x00000004
WORM	0x00000010
CDROM	0x00000020
magneto optical	0x00000080
changer	0x00000100

4	media cartridge	Indicates the type of cartridge/magazine that the device can use. The definition may be expanded by Novell to accommodate new classes of media. If unknown, this field should be set to a -2

fixed media	0x00000000
5.25 in floppy	0x00000001
3.5 in floppy	0x00000002
5.25 in optical	0x00000003
3.5 in optical	0x00000004
.5 in tape	0x00000005
.25 in tape	0x00000006
8 mm tape	0x00000007
4 mm tape	0x00000008
Bernoulli disk	0x00000009
12 in optical	0x0000000A

4	function mask	Indicates the type of I/O access functions supported on the device.

random read	0x00000001
random write	0x00000002
random write once	0x00000004
sequential read	0x00000008
sequential write	0x00000010
reset end of media	0x00000020
single file marks	0x00000040
consecutive file marks	0x00000080
single set marks	0x00000100
consecutive set marks	0x00000200
relative data blocks	0x00000400
absolute data blocks	0x00000800
sequential partition operations	0x00001000
physical media operations	0x00002000
random erase	0x00004000

4	control mask	Indicates the type of I/O control functions (IOCTLs) that can be issued to this device.

		Func.	SubFunc.
activate/deactivate	0x00000001	0	0, 1
mount/dismount	0x00000002	0	7, 8
select/unselect	0x00000004	2	3, 4
lock/unlock	0x00000008	0	9, 10
eject	0x00000010	0	11
move media	0x00000020	3	3
magazine support	0x00000040	2	0 - 7
changer support	0x00000080	3	0 - 3

4	data transfer unit size	Indicates the current transfer unit size (sector size) of the device in bytes. Currently, the **NetWare File System only supports 512 byte sectors**. Devices that contain mountable NetWare volumes must return a value of 512 in this field. This restriction may be ignored for devices that bypass the NetWare File System. If unknown, this field should be set to a -2
4	maximum transfer size	Indicates the maximum number of transfer units that can be specified in a single command. If unknown, this field should be set to a -2.
4	capacity in unit size	Indicates the capacity of the device in units defined in the "data transfer unit size" field. If unknown, this field should be set to a -2.
4	preferred unit size	Indicates the preferred (native) transfer unit size (sector size) of the device in in bytes. If the value in this field differs from that in the "data transfer unit size", the device driver is required to make the appropriate sector translations on all requests received. If unknown, this field should be set to a -2
64	reserved	Reserved by NetWare.

Return Media Info (Mandatory for Removables) (Recommended for all other device types)

Function: 1
Sub-function: 1

This IOCTL directs the driver to return additional information about the media in the device.

Possible Completion Codes (in IOCTLRequestStructure):

0000h	No Error
0003h	Non-Media Error
0008h	Not Supported By Device
0703h	No Media Present
FFF9h	Not Supported By Driver

IOCTL Request Structure Fields

LONG DriverLink	Driver specific usage.
CardStruct *CardHandle	Contains the card handle returned by *AddDiskSystem*.
WORD CompletionCode	The driver fills this field with a completion status.
BYTE Function	Contains a value of 1.
BYTE SubFunction	Contains a value of 1.
LONG IOCTLParameter	Contains the device handle returned by *AddDiskDevice*.
LONG *IOCTLBuffer	Points to a structure the driver must fill with information in the following format:

Bytes	Field	Description	
4	media type	This field indicates the media type.	
		disk	0x00000000
		tape	0x00000001
		printer	0x00000002
		WORM	0x00000004
		CDROM	0x00000005
		magneto optical	0x00000007
4	media type mask	This field is a bit map indicating the type media supported by the device.	
		disk	0x00000001
		tape	0x00000002
		printer	0x00000004
		WORM	0x00000010
		CDROM	0x00000020
		magneto optical	0x00000080

4 media cartridge — This field indicates the type of cartridge/magazine that the device can use. The definition may be expanded by Novell to accommodate new classes of media. If unknown, this field should be set to a -2

fixed media	0x00000000
5.25 in floppy	0x00000001
3.5 in floppy	0x00000002
5.25 in optical	0x00000003
3.5 in optical	0x00000004
.5 in tape	0x00000005
.25 in tape	0x00000006
8 mm tape	0x00000007
4 mm tape	0x00000008
Bernoulli disk	0x00000009
12 in optical	0x0000000A

4 function mask — This field indicates the type of I/O access functions supported on the device.

random read	0x00000001
random write	0x00000002
random write once	0x00000004
sequential read	0x00000008
sequential write	0x00000010
reset end of media	0x00000020
single file marks	0x00000040
consecutive file marks	0x00000080
single set marks	0x00000100
consecutive set marks	0x00000200
relative data blocks	0x00000400
absolute data blocks	0x00000800
sequential partition operations	0x00001000
physical media operations	0x00002000
random erase	0x00004000

4 control mask — This field indicates the type of I/O control functions (IOCTLs) that can be issued to this device.

		Func.	SubFunc.
activate/deactivate	0x00000001	0	0, 1
mount/dismount	0x00000002	0	7, 8
select/unselect	0x00000004	2	3, 4
lock/unlock	0x00000008	0	9, 10
eject	0x00000010	0	11
move media	0x00000020	3	3

4	data transfer unit size	This field indicates the current transfer unit size (sector size) used to record data on the "unit"-size media in bytes. Currently, the **NetWare File System only supports 512 byte sectors**. Devices that contain mountable NetWare volumes must return a value of 512 in this field. This restriction may be ignored for devices that bypass the NetWare File System.
4	maximum transfer size	This field indicates the maximum number of transfer units that can be specified in a single command.
4	capacity in unit size	Indicates the capacity of the media in units defined in the "data transfer unit size" field. If unknown, this field should be set to a -2.
4	preferred unit size	Indicates the preferred (native) transfer unit size (sector size) of the media in bytes. If the value in this field differs from that in the "data transfer unit size", the device driver is required to make the appropriate sector translations on all requests received.
4	media formatting	A value of one (1) in this field indicates that the media has not been formatted and requires formatting before data can be written to it. A value of zero (0) indicates that the media is formatted.
64	reserved	Reserved by NetWare.

Set Device Parameters

(Mandatory for Tape Drives)
(Recommended for all other device types)

Function: 1
Sub-function: 2

This IOCTL directs the driver to set the device configuration options. A single parameter is set with each call.

Possible Completion Codes (in IOCTLRequestStructure):

0000h	No Error
0003h	Non-Media Error
0008h	Not Supported By Device
0703h	No Media Present
FFF9h	Not Supported By Driver

IOCTL Request Structure Fields

LONG DriverLink	Driver specific usage.
CardStruct *CardHandle	Contains the card handle returned by *AddDiskSystem*.
WORD CompletionCode	The driver fills this field with a completion status.
BYTE Function	Contains a value of 1.
BYTE SubFunction	Contains a value of 2.
LONG IOCTLParameter	Contains the device handle returned by *AddDiskDevice*.
LONG *IOCTLBuffer	Points to a structure in the following format that contains the device configuration options to be set by the driver.

Buffer Structure

```
LONG  ParamSelectMask
LONG  OptionValue
LONG  Reserved0[16]
```

Where:

ParamSelectMask — This field is a bit mask that designates which parameter is to be set. The allowable parameter selections are listed below with their corresponding bit mask.

00000001h	Data Transfer Unit Size	Indicates either the block size to be used when writing to sequential access media or the default sector size of random access devices. (See the *Return Device Info* and *Return Media Info* IOCTLs.)
00000002h	Format Sector Size	Sector size used to format the media in the device. (See the *Return Media Info* IOCTL.)
00000004h	Media Write Format	Indicates the type of format this device will use while writing to the media. (See the *Return Tape Device Info* IOCTL.)
00000008h	Data Compression	The compression state and options selected for a tape device. (See the *Return Tape Device Info* IOCTL.)

OptionValue	The field contains the value to which the parameter designated in ParamSelectMask is set.
Reserved0	Reserved by NetWare.

Return Tape Device Info (Mandatory for Tape Drives)

Function: 1
Sub-function: 3

This IOCTL directs the driver return additional information about a tape device. It provides a means by which an application can discover if a particular device possesses all of the capabilities required by the application.

Possible Completion Codes (in IOCTLRequestStructure):

0000h	No Error
0003h	Non-Media Error
0008h	Not Supported By Device
0703h	No Media Present
FFF9h	Not Supported By Driver

IOCTL Request Structure Fields

LONG DriverLink	Driver specific usage.
CardStruct *CardHandle	Contains the card handle returned by *AddDiskSystem*.
WORD CompletionCode	The driver fills this field with a completion status.
BYTE Function	Contains a value of 1.
BYTE SubFunction	Contains a value of 3.
LONG IOCTLParameter	Contains the device handle returned by *AddDiskDevice*.
LONG *IOCTLBuffer	Points to a structure the driver must fill with information in the following format:

Bytes	Field	Description
4	Absolute Position Buffer Size	Indicates the size of the buffer in bytes (usually 4) needed for the absolute position information. (See the *AbsoluteDataBlock* (0Bh) I/O function.)
4	Media Cartridge	Indicated the type of media cartridge this device supports. 1/2 inch 0x00000005 1/4 inch 0x00000006 8mm 0x00000007 4mm 0x00000008

4 Media Write Format

Indicates the write formats supported by the device. The formats will be grouped below according to the specifications that apply to each tape size.

For 1/4 Inch:

QIC-24	0x00000001	
QIC-120	0x00000002	
QIC-150	0x00000004	
QIC-320	0x00000008	
QIC-525	0x00000010	
QIC-1350	0x00000020	
QIC-2100C	0x00000040	
QIC-1000	0x00000080	
QIC-3010	0x00000100	
QIC-3020	0x00000200	

For 1/2 Inch:

X3B5/87-099	0x00000001	
X3B5/86-199	0x00000002	
HI-TC1	0x00000004	
HI-TC2	0x00000008	
X3.193-1990	0x00000010	
X3B5/91-174	0x00000020	
X3B5/91-227	0x00000040	
X3.266-199x	0x00000080	
X3B5/94-354	0x00000100	

For 8mm:

EXABYTE-8200	0x00000001	(14h X3.202-1991)
EXABYTE-8500	0x00000002	(15h ECMA TC17)
EXABYTE-8500C/05	0x00000004	(8Ch)
EXABYTE-8205	0x00000008	(90h)
EXABYTE-9500	0x00000010	

For DAT:

DDS	0x00000001
Data DAT	0x00000002
DDS2	0x00000004

4	Media Read Format	Indicates the read formats supported by the device. (Use the same values defined for Media Write Formats above.)
4	Minimum Block Size	When in Fixed Block Mode, this field returns the minimum block size supported by the device.
4	Maximum Block Size	When in Fixed Block Mode, this field returns the maximum block size supported by the device.
4	Maximum Partitions	This field returns the maximum number of partitions that can be created on a single media by this device.
4	Maximum Partition Size	This field returns the maximum partition size (in megabytes) that can be created by the device. For sequencial access devices, if the maximum partition size is unknown, this field should be set to a -2.
4	Data Compression	This field returns the options possible for data compression on the device. It also returns the present state of the compression function on the device. The options defined are bit encoded as follows: DeviceSupportsCompression 0x00000001 CompressionModeSelectable 0x00000002 DocompressionIsIndependent 0x00000004 The current state bits are defined as: CompressionIsSelectedBit 0x00000100 DecompressionIsSelectedBit 0x00000200
64	Reserved	Reserved for future use.

Return Magazine Info (Mandatory for Magazines)

Function: 2
Sub-function: 0

This IOCTL directs the driver to return a structure that contains magazine configuration data. This IOCTL should not be issued until after the "Magazine Load" IOCTL is issued.

Possible Completion Codes (in IOCTLRequestStructure):

0000h	No Error
0006h	Device Parameter Error
0007h	System Parameter Error
0008h	Not Supported By Device
0F09h	Magazine Not Present
1409h	Magazine Error
1703h	Magazine Jammed
FFF9h	Not Supported By Driver

IOCTL Request Structure Fields

LONG DriverLink	Driver specific usage.
CardStruct *CardHandle	Contains the card handle returned by *AddDiskSystem*.
WORD CompletionCode	The driver fills this field with a completion status.
BYTE Function	Contains a value of 2.
BYTE SubFunction	Contains a value of 0.
LONG IOCTLParameter	Contains the device handle returned by *AddDiskDevice*.
LONG *IOCTLBuffer	