Date: Jan. 8, 2002

To: T10 Committee (SCSI)

From: Lee Jesionowski (IBM)

Subject: ADI - IBM Library/Drive Interface Specification (T10/02-022r0)

The attached IBM specification is being made available for review and consideration in the development of an Automation/Drive Interface standard.

IBM Library/Drive Interface Specification

Revision 4

January 8, 2002

Table of Contents

ABOUT THIS DOCUMENT

This document is intended to provide a functional specification for the *library/drive interface (LDI)* of IBM LTO tape drives.

Organization

The information in this book is presented as follows:

- Introduction on page 6 provides an overview of the interface.
- Physical Attachment on page 8 describes the *LDI* physical characteristics.
- Packet Protocol on page 10 describes the low-level packet protocol utilized for *LDI* message passing.
- Message Types and Subtypes on page 15 summarizes the messages and responses supported in the IBM LTO Tape Drive *LDI*.
- Interface Scenarios on page 30 provides examples of the sequence of messages required for specific scenarios.

Summary of Changes

Revision 0

First draft version.

Revision 1

Added configuration option to operate in polled mode.
Clarified drive method for setting SCSI address during power-on reset.
Added several new maintenance sub-commands for MAINT_COMMAND including support for the attaching library to report drive device identifiers in response to the SCSI-3 Read Element Status command with DVCID=1.

Revision 2

Added support for 9600 baud (currently only selectable via feature switch)
Added support for reading tape, uploading drive firmware, downloading drive firmware, and reading any other drive buffer or log that is available via SCSI including Cartridge Memory contents, Vital Product data, drive dumps, TapeAlert flags, and error logs.
Simplified SCSI wrap test message to run the test without drive resets.
Added several fields to the DRIVE_STATUS message.
Miscellaneous corrections and clarifications.

Revision 3

Added a feature switch option for a default setting of polled mode such that a Config_Request will never be sent.
Added a feature switch option to use 1 or 2 stop bits
Added support for 2 primitives, Drive Type Request and Drive Type, which do not follow the packet protocol.
Added support for disabling softload, disabling eject on SCSI Unload, disabling eject on *LDI* Unload, immediate response to *LDI* Unload, and new *LDI* Load message. The Load message includes options for disabling tape threading and for immediate response.

Added a new field in Drive Status Flags to report cartridge presence and clarified the Drive Status Flag settings for all possible cartridge positions.
Miscellaneous corrections and clarifications.

Revision 3.1

New Maint_Command subtype Library Utility Write Tape added.
New Maint_Command subtypes Device Online/Offline added.
Maint_Command subtypes Update Drive FMR Cartridge and Erase Drive FMR Cartridge command timeout checks added.
Miscellaneous corrections and clarifications.
Set_Config command, Configuration flags byte, bit 4 now defined as Disable Automatic Eject of FMR or Cleaner tape.
Read Log Sense Data response data format changed to match the SCSI specification.

Revision 3.2

New Drive_Status Drive Status flags added.
Miscellaneous corrections and clarifications.
New Set_Config fields to support Fibre Channel

Revision 3.3

New online control flag to Set_Config.
Config_Request timing information.

Revision 3.4

Restore the Primitives Command description.
Updates to the Device Online and Device Offline Maint_Commands

Revision 3.5

Updates to SCSI or Fibre channel addressing following power-on reset
Cleanup in the handshaking, startup process table
Miscellaneous textual cleanup throughout the document

Revision 4

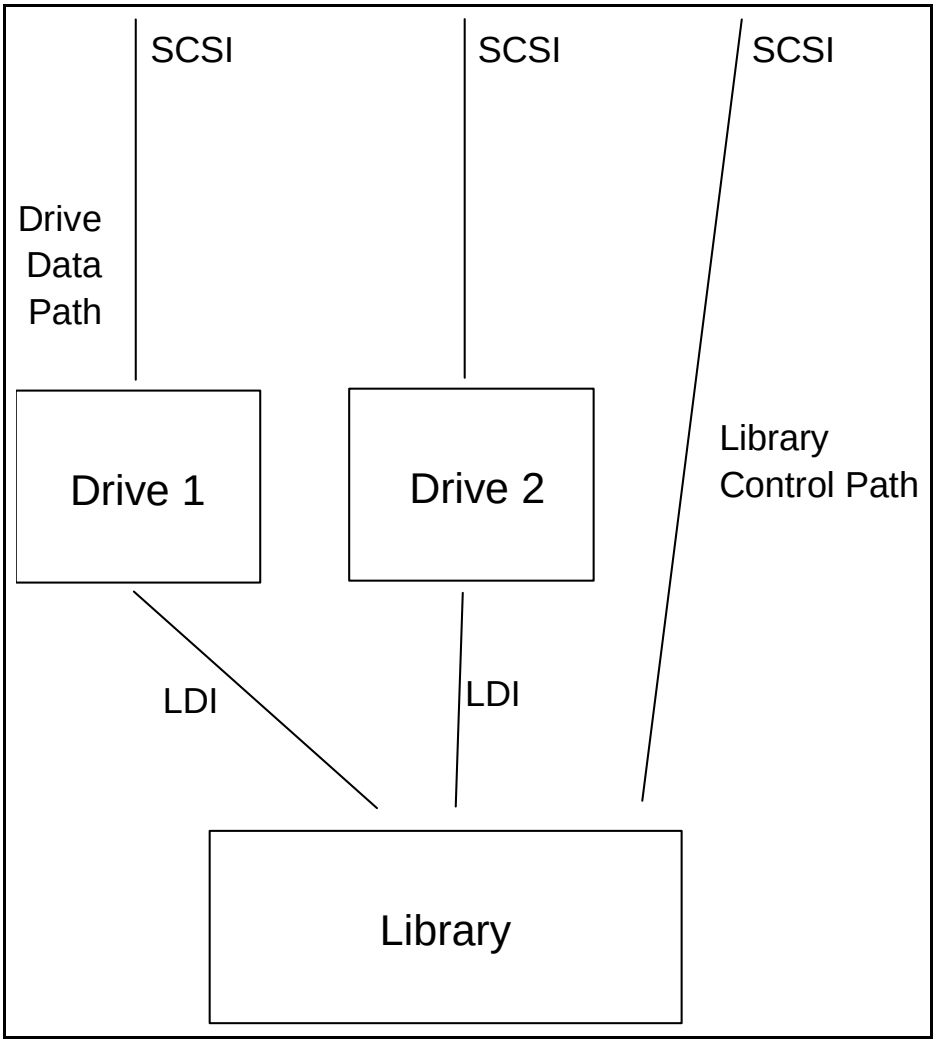
Changed document name to IBM Library/Drive Interface Specification
Removed classification of IBM Confidential
Removed specific references to other IBM documents
Changed format of message tables to be more T10 standard-compatible
Miscellaneous corrections and clarifications

NOTE: Changes from the previous revision are denoted by change bars | in the left margin of the page.

Introduction

The IBM LTO Tape Drive *library/drive* interface *(LDI)* provides a means for establishing a direct communication of commands and status between an IBM LTO tape drive and the subsystem in which it is installed, hereafter referred to as a library. The interface provides a means for the library to configure a drive for operation and for the drive to report status to the library. It can also serve as a service communication path when offline service functions are performed under the control of the operator panel of the tape library. The physical communication occurs across an *LDI* with one cable attachment for each tape drive installed in the library. A simple logical protocol is provided which supports messages that may be passed over the interface. **Figure 1, Tape Library and Tape Drive Interfaces** shows the basic configuration of the SCSI-attached tape drives and the tape library.

Figure 1, Tape Library and Tape Drive Interfaces



Each time the drive executes a reset for any reason, the drive requires a handshaking startup process (see Interface Scenarios on page 30). During this process, the *LDI* is configured to operate in non-polled or polled mode (the default setting is selectable via a feature switch on the tape *drive*). If non-polled mode is selected, then the drive may initiate communications on the *LDI* at any time. If polled mode is selected, then the drive will not initiate *LDI* communication again until the next reset (if the default is

non-polled mode) or until the mode is changed - only the library may initiate communications in polled mode. Polled mode is intended for use in designs with a multiplexed or shared *LDI* line, or when library implementations do not support queueing of requests.

Physical Attachment

LDI Electrical Characteristics

The RS-422 interface supports:

- Full duplex operation, asynchronous (transmit/receive simultaneously)
- Baud rate of 9600 or 38400 (selectable via a feature switch on the tape drive)
- Data Bits: 8
- Parity: None
- Stop Bits: 1 or 2 (selectable via a feature switch on the tape drive)

LDI Connector and Signals

The *LDI* connector and signals are *specified in the applicable drive product specification.*

Primitives

All communications follow RS-422 standards with 8 data bits, no parity bits and 1 or 2 stop bits. Most of the communications utilize a packet protocol as described in *Packet Protocol* on page *10*. The only exceptions are two primitives which can occur outside of the packet protocol, which means there are no requirements for start-text, length, checksum, end-text, acknowledgements, or byte stuffing. The two primitives are Drive Type Request and Drive Type.

Drive Type Request is a 1-byte value of 0x00 sent from the library to the drive, intended for detecting the drive type in a library capable of supporting multiple drive types.

Drive Type is a 9-byte sequence sent from the drive to the library in response to a Drive Type Request.

Byte	Description
0	0xFA
1	0x49 (ASCII 'I')
2	0x42 (ASCII 'B')
3	0x4D (ASCII 'M')
4	0x80
5 ... 8	(MSB) Firmware Revision (4 bytes) (LSB)

Packet Protocol

This section describes the packet protocol used in the IBM LTO Tape Drive *Library/Drive Interface*. All communications follow RS-422 standards with 8 data bits, no parity bits and 1 or 2 stop bits. The protocol utilizes a special set of control characters as shown in **Table 1, ASCII Control Characters**.

Table 1, ASCII Control Characters

Control Char	Hex Value	Meaning	Description
STX	0x02	Start of Text	Beginning of packet text
ETX	0x03	End of Text	End of packet text
ACK	0x06	Positive Acknowledgement	Packet received and ready to receive next packet NOTE: Only the packet checksum and message length are validated before sending an ACK, not the message contents.
NAK	0x15	Negative Acknowledgement	Packet received incorrectly; packet will be retransmitted immediately with up to 3 retries. Sent after checksum error or message length error (or on the library side, possibly on a receive time-out). NOTE:When the drive powers up or is reset, the drive will return a NAK for most messages received from the library until the handshaking startup process is completed. It is recommended that the library perform the handshaking startup process (described later) if retries fail in response to a NAK.
SNAK	0x1A	Special Negative Acknowledgement	Packet received incorrectly; packet should be retransmitted after 10 second wait and retry up to 10 times before sending error message. Sent when the entity receiving the packet is currently too busy to handle more *LDI* traffic.

Following tape drive power-on-reset completion, messages and responses are sent between the tape drive and the library using a packet structure, except for the primitives Drive Type Request and Drive Type.

NOTE: Prior to tape drive power-on-reset completion, a byte sequence ending with ETX (0x03) may be transmitted as part of the tape drive self-test procedure. These will be generated both at power up and when the drive is reset. This byte sequence does not conform to the packing protocol described in this section; however, the protocol will discard it as invalid. Therefore, higher level message handling application code will never see these bytes. It is recommended that any library or drive implementation which performs a self-test or diagnostic sequence or transmits information from the serial port include the ETX (0x03) character prior to the first meaningful message sent.

Subsequent to tape drive power-on-reset, all packets (except ACK, NAK, and SNAK) are formatted as follows:

Figure 2, Packet Format

The five fields of the packet are:

STX

Start of text character

L1 L2

A two byte field containing the number of message data bytes. A 2-byte hexadecimal integer with the high-order byte (L1) sent first. **Note:** The count does not include STX, the two length bytes, BCC, or ETX.

Message

Actual message to be sent (see Message Types and Subtypes on page 15).

BCC

Block-check character

ETX

End-of-text character

Note that the ETX is always required as a final byte of any packet, even of ACK, NAK, and SNAK responses. ACK, NAK, and SNAK responses therefore are precisely two bytes long consisting of the ACK, NAK, or SNAK, followed by the ETX character.

The block-check character (BCC), also referred to as a checksum, is derived as follows:

- Treat each byte as an integer between 0 and 255
- Add the bytes, beginning with (and including) the length bytes L1 and L2 and ending with (and including) the last byte of the Message. (The STX, ETX, and BCC are **not** included in the checksum.)
- EITHER: Divide by 256 and use the remainder as a single byte checksum -OR- Use the low order byte (SUM & 0xFF) as the single byte checksum.

Byte Stuffing

Byte stuffing is used to ensure the STX (0x02) character and the ETX (0x03) character are not confused with another 0x02 or 0x03 byte that happens to be in the length bytes, message bytes, or block check character.

It is a procedure which is performed on a packet that has already been processed and is completely ready to be sent. This means that:

1. The length bytes **L1** and **L2** have already been calculated and placed in the packet
2. The message has been added to the packet
3. The **BCC** has been calculated and placed in the packet
4. The **ETX** character and the **STX** characters may be added before or after byte stuffing, but their values are never affected by the byte stuffing operation.

Note: Byte stuffing does not affect the calculations for the length of the message or the block check character. It is only used to insure that the receiver can accurately determine the STX and ETX characters, since only a valid STX and ETX are sent as a 0x02 or a 0x03. Byte stuffing is done only **after** the calculations for the message length and the checksum have already been done. ***Note that STX, ETX, ACK, NAK, and SNAK are the only characters not subject to possible change due to byte stuffing.***

Byte stuffing is defined as follows:

- The byte value 0x02 is changed to the bytes 0xFF 0xF2 by the sender before the byte is sent.
- The bytes 0xFF 0xF2 are changed back to 0x02 by the receiver as the message is received.
- The byte value 0x03 is changed to the bytes 0xFF 0xF3 by the sender before the byte is sent.
- The bytes 0xFF 0xF3 are changed back to 0x03 by the receiver as the message is received.
- The byte value 0xFF is changed to the bytes 0xFF 0xFF by the sender before the byte is sent.
- The bytes 0xFF 0xFF are changed back to 0xFF by the receiver as the message is received.
- While unstuffing, any 0xFF byte followed by a value other than 0xF2, 0xF3 or 0xFF is undefined and may be treated as an error condition, and a NAK should be issued in response.

To summarize:

When transmitting, convert:	
...this byte...	...to these bytes...
0x02	0xFF 0xF2
0x03	0xFF 0xF3
0xFF	0xFF 0xFF

When receiving, convert	
...these bytes...	...to this byte..
0xFF 0xF2	0x02
0xFF 0xF3	0x03
0xFF 0xFF	0xFF

The following example shows how this process works.

Figure 3, Byte Stuffing Example

Assume that the following packet must be sent:

0x02	0x00	0x02	0x23 0xFF	0x24	0x03
STX	L1	L2	Message bytes	BCC	ETX

Instead the following would be sent:

0x02	0x00	0xFF 0xF2	0x23 0xFF 0xFF	0x24	0x03
STX	L1	Modified L2	Modified Message bytes	BCC	ETX

Note that the block check character (BCC) and the length bytes L1 and L2 are all computed before the packet is modified.

Note: In the modified packet, it is **guaranteed** that there will never be a 0x02 or 0x03 in the packet except for an **STX** and **ETX**.

Sending Packets

The rules for transmitting a packet follow. Note that this is dominantly written from the tape drive perspective, with notes on recommendations for the tape library.

1. The sending station must format each packet as shown in **Figure 2, Packet Format** on page 11 by adding STX, the length, the checksum (BCC), and ETX to the message. Only ETX is added to

ACK, NAK, or SNAK. The message length and message bytes are then modified using byte stuffing.

2. A sending station must initiate a 5-second time-out after the last character of a packet is sent. It must receive an ACK, NAK, or SNAK prior to the end of the time-out. If it does not, it retries the send up to 3 times. The send is also retried up to 3 times if NAK is received and up to 10 times if SNAK is received (but after a 10 second delay each time in the case of SNAK). Note that the SNAK is an indication that the receiving entity was too busy to deal with message handling and asked for a delayed retry. If all retries fail, the send ends in error.
3. A sending station must not send a second packet until either an ACK, NAK, or SNAK has been received in response to the previous packet, or until the time-out has expired.
4. The possibility of receiving duplicate responses must be provided for, as when an ACK/NAK/SNAK for a packet is received after the response timer has expired. The Msg ID field can be used for this purpose in single packet cases, and the Msg ID and Section Number can be used in multiple packet messages.
5. The tape library and drive have discretion in the choice of retry counts and in the choice of timeout values. The ones described in this list characterize the values for the IBM tape drive.

Receiving Packets

The rules for receiving a packet follow. Note that this is dominantly written from the tape drive perspective, with notes on recommendations for the tape library.

1. A buffer size of 1024 bytes is required to receive the largest single-packet transmission, due to byte stuffing. The byte unstuffing can be done “on the fly” as the packet is received, lowering buffer requirements and more evenly distributing unstuffing processing time. If “on the fly” unstuffing is used, the receiver may need to maintain additional packet delineation information as the 0x02 and 0x03 may appear within the packet and are not STX and ETX characters.
2. All characters preceding STX are ignored,except for ACKs, NAKs, and SNAKs. Partial packets pre-pended onto full packets will result in a NAK, however, due to an apparent mismatch of the L1, L2 length indication with the total (combined) message length. Note that the set of bytes sent by the IBM tape drive prior to power-on-reset completion will be discarded by an attaching library which follows this rule, since none of the self-test associated bytes are STX (0x02). It is permissible to only delineate packets by searching for ETX characters. In this case, the packet validation differs. ACKs, NAKs and SNAKs need to be immediately followed by an ETX, and all other packets must start with an STX. Length and BCC checks will ensure validity.
3. After receiving the ending ETX, the receiver un-stuffs the length and message bytes and looks for the checksum (BCC). If the length or checksum is incorrect, the receiving station responds to this packet by sending a NAK (followed by ETX). If the receiver is too busy to handle the packet at all, it generates a SNAK (followed by ETX). The IBM tape drive implementation will not return SNAKs, but will process those generated by a library.
4. If an ETX has been received, and the length and checksum bytes are correct, the receiving station responds to this packet by sending an ACK (followed by ETX). Note that if the attaching library receives any message type that is not listed in **Table 2, Message Types** on page 15 it should simply return an ACK and take no further action.
5. The IBM tape drive does not currently implement a receive time-out. If a partial packet is received, no response is returned until a subsequent ETX is received (from a subsequent packet). Due to probable L1 L2 mismatch with the aggregate packet a NAK will then be returned to the attaching library. The tape library or drive may implement a receive time-out if desired. A possible timeout value for packet completion following the receipt of the first character is 10 seconds. Failure to receive a complete packet would result in the tape library sending a NAK. In the event of a such a time-out, the tape library should discard any partial packet received. The receive time-out should be less than the send time-out. NAKs should be sent immediately following reception of an ETX byte (when NAK is appropriate). NAKs sent at other times should be ignored.

6. Messages are sent one at a time. For messages that require a response, once a message has been started, no additional message can be started until a response is received for that message. For messages that do not require a response, no additional message can be started until sending of that message is complete and ACK received. The one exception is that the drive will receive and respond to a Drive_Status_Request command even while working on another command that it owes a response to.

General Implementation Issues

1. Timeouts and retry counts may vary. The afore-mentioned values are typical and have been used in implementations, however, there may be instances where other values are more appropriate.
2. The *LDI* receive and transmit logic must be implemented such that race conditions on near/concurrent received and transmitted packets do not result in deadlock. To avoid this, consider the following: Once a packet is received, an ACK/NAK/SNAK must be sent. This cannot interrupt a packet which is currently being sent, but should immediately follow it (prior to receipt of the ACK/NAK/SNAK for the received packet).
3. No received characters should be dropped. It is possible for a packet to be received immediately followed by an ACK/NAK/SNAK for a previously/concurrently sent packet. This packet response must be routed to the transmit logic in a timely fashion to prevent the timeout and retransmit of the command. Even with no character loss, there is an implicit race here which is handled by logic in the receiver to filter duplicate received valid packets.

Message Types and Subtypes

The message sent in the packet has the following general format:

Figure 4, Message Format

Byte	Description
0	Msg Type
1	Tgt Addr
2 ... 5	Msg ID
6 ... n	Message Data (variable length)

LEGEND:

Msg Type (Message Type)

One byte that defines the format (contents) of the message data. *In order to co-exist with legacy IBM library/drive interfaces, there are only* two message types defined for this interface as shown in **Table 2, Message Types**. The Two_Way message type is further sub-divided into message subtypes. In this document, the term 'message' is used generically in reference to specific message types as well as message subtypes. Additional details on all message types and subtypes follow later in this section.

Table 2, Message Types

Value	Message Type	Direction	Description	Details
0xAB	Two_Way	Drive-to-Library Library-to-Drive	The first byte of the Message Data is a message subtype. See **Table 3, Two_Way Message Subtypes** on page 16.	Page 16
0xAC	Set_Config	Library-to-Drive	Library configuration information, asynchronous or response to Config_Request message subtype.	Page 16

Tgt Addr (Target *LDI* Address)

One byte that indicates for which target the message is intended. When the tape library is the target, the *LDI* address is always set to 0xFF. For a target drive, the *LDI* address should be a value in the range of 0x01 to the maximum drive number (up to 0xFE). It is recommended that this address be unique for each drive connected to a common library controller (it is set by the library during the Set_Config message). Keeping the address unique may aid in debugging and allow more rigorous packet integrity checking.

Msg ID (Message ID)

Four bytes that identify every particular packet. The first byte is the *LDI* address of the original message source. The next three bytes should be used to provide a unique key for the source of two or more messages to associate their respective responses. Any response type of message (e.g. Maint_Status_Good) should be sent with these 4 bytes unaltered from the original message.

Message Data

Variable data whose contents are determined by the Message Type. For the Message Type of Two_Way, the first byte of Message Data is the Message Subtype as summarized in **Table 3, Two_Way Message Subtypes** on page 16.

Table 3, Two_Way Message Subtypes

Value	Message Subtype	Direction	Description	Details
0x01	Config_Request	Drive-to-Library	Request for library configuration, library responds with Set_Config	Page 16
0x30	Maint_Command	Library-to-Drive	*Request for drive operation, identity, status, logs, or firmware.*	Page 18
0x3C	Maint_Status_Good	Drive-to-Library	Response to Maint_Command w/optional data	Page 25
0x3F	Maint_Status_Error	Drive-to-Library	Response to Maint_Command w/error sense data	Page 26
0x40	Drive_Status	Drive-to-Library	Drive status, asynchronous or Drive_Status_Request response	Page 26
0x41	Drive_Status_Request	Library-to-Drive	Request for drive status, drive responds with Drive_Status	Page 29

Additional details for all Message Types and Subtypes are provided below:

Config_Request (Drive-to-Library, Type=0xAB, Subtype=0x01)

When the drive powers up or is reset, the drive will return a NAK for any Two_Way message received from the library until the handshaking startup process is completed. If the interface is operating with a default of non-polled mode (settable via the drive feature switches), the drive will initiate the handshaking startup process by sending the Config_Request message. This message will be sent within 10 seconds after the drive powers up or is reset. *For polled mode of operation, see Scenario for Handshaking Startup Process (Library-*Initiated) *on page 30.*

Byte	Description
0	Msg Type (0xAB)
1	Tgt Addr (0xFF)
2 ... 5	Msg ID
6	Msg Subtype (0x01)

Set_Config (Library-to-Drive, Type=0xAC)

The Set_Config message may be either asynchronously sent from the library to the drive, or can be a response to a Config_Request message.

Note: If this is a response to a Config_Request, it is critical that all four Msg ID bytes are identical to those in the Config_Request (even though the drive *LDI* address may no longer be valid for the case of a drive that has just been moved from one position to another). A change to the drive *LDI* address is communicated to the drive using the Tgt Addr field of the Set_Config message.

Byte	Description
0	Msg Type (0xAC)
1	Tgt Addr
2 ... 5	Msg ID
6	Interface Version Major (0x02)
7	Interface Version Minor (0x00)

8 ... 25	Reserved (18 bytes)
26	Drive FC/SCSI Address
27 ... 52	Reserved (26 bytes)
53	Configuration Flags
54	Library Information Field
55 ... 62	(MSB) Drive Fibre Channel World Wide Node Name (LSB)
63	Reserved

LEGEND:

Drive FC/SCSI Address

This field is the method for the library to provide the I/O address to the drive. For SCSI, valid addresses range from 0 to 15. For Fibre Channel, the address is a Loop ID (not an AL_PA address) and valid addresses range from 0 to 127. Addresses 126 or 127 specify that the drive uses soft addressing mode (this mode is not recommended except in the highly unlikely event that it is explicitly specified by host device driver/host bus adapter installations).

Configuration Flags

- Bit 7: Non-Polled Mode
  - 0: Configure *LDI* to operate in polled mode
  - 1: Configure *LDI* to operate in non-polled mode
- Bit 6: Disable Softload
  - 0: Enable drive softload capability (SCSI Mode Page settings may apply).
  - 1: Disable drive softload capability. This setting will take precedence over any SCSI Mode Page setting to enable the drive softload capability but no SCSI mode pages will be changed (reference T10/99-347r1, Autoload Mode field).
- Bit 5: Disable SCSI Eject
  - 0: Enable eject on SCSI Unload command (SCSI Unload CDB options may apply).
  - 1: Disable eject on SCSI Unload command. This setting will take precedence over any SCSI Unload CDB option to perform the eject (reference T10/99-347r1).
- Bit 4: Disable automatic eject of FMR or Cleaner tape after command
  - 0: Automatic eject enabled
  - 1: Automatic eject disabled
- Bit 3: Disable automatic drive online
  - 0: Drive goes online to SCSI or Fibre Channel automatically without needing Set_Config
  - 1: Requires the drive to receive a Set_Config to go online
- Bit 2-0: Reserved

Library Information Field
This information is relayed to the SCSI host in every Request Sense (error and non-error) response in parameter data, byte 35.

Drive Fibre Channel World Wide *Node* Name
This is used by the drive to set its Fibre Channel port name(s) and node name. This allows the library to own these values. A value of zero is an instruction for the drive to use its own world wide names.

The drive may optionally reset itself for the following configuration changes:

W First-ever receipt of this command for any drive model

- Drive *LDI* Address (Tgt Addr)
- Drive FC/SCSI Address
- Drive Fibre Channel World Wide *Node* Name
- *Configuration Flag Bit 3, Disable automatic drive online*

The drive will store the following fields in non-volatile storage:

- Drive *LDI* Address (Tgt Addr)
- Drive FC/SCSI address
- Configurations Flag bit 3, Disable automatic drive online
- Library Information Field
- Drive Fibre Channel World Wide *Node* Name

Drive *FC/SCSI* Address Handling During Power-On Reset

Control of address handling during power-on reset is managed using Set-Config message, configurations flag bit 3, Disable automatic drive online. When the drive is configured such that it expects its SCSI or Fibre Channel address from the *LDI* (see *the applicable drive product specification*), the drive performs one of the following actions at power-on reset:

- If Disable automatic drive online bit is set in non-volatile storage, the drive remains offline (does not respond to Selection on the SCSI bus and does not participate in the loop in Fibre Channel) until it receives a Set_Config message with a valid address. This greatly reduces the possibility of a replacement (unconfigured) drive causing address conflict errors during concurrent drive maintenance ('hot swap') scenarios.
- If Disable automatic drive online bit is not set in non-volatile storage, the drive will set its address to that of the prior Set_Config message and go online.

Maint_Command (Library-to-Drive, Type=0xAB, Subtype=0x30)

The Maint_Command message is used to request one of a set of pre-defined operations (referred to as sub-commands) to be executed by the drive. The drive will respond to this message with a Maint_Status_Good or Maint_Status_Error message and data, as appropriate. Note that the Maint_Command message provides implicit permission for the drive to use the *LDI* when the Maint_Status_Good or Maint_Status_Error message is ready to be sent, even if the drive *LDI* is operating in polled mode.

This message was originally intended as a catch-all for maintenance/service operations (hence, the name) but the intended usage now extends beyond maintenance/service activities. Any of these commands may be issued at any time, though the library design *may* dictate that *some of* these maintenance commands are used while the library is offline (not ready).

Byte	Description
0	Msg Type (0xAB)
1	Tgt Addr
2 ... 5	Msg ID
6	Msg Subtype (0x30)
7	Maintenance Sub-command
8...n	Parameter data (if specified below)

The following maintenance sub-commands are provided:

Byte 7	Description	Additional Details
0x00	Device Offline	See page 20.
0x01	Device Online	See page 20.
0x10	Read Standard Inquiry Data	Requests SCSI Standard Inquiry data.
0x11	Read Request Sense Data	Requests SCSI Request Sense data - this does not clear SCSI error sense for any initiator. The drive queues multiple errors, so in some cases the library may want to request sense until sense data indicating no additional errors are present is returned.
0x12	Read Device Identification Data	Requests SCSI Inquiry Page 0x83 in order for a SCSI library to support SCSI-3 Read Element Status with DVCID=1.
0x20	Set Drive Serial Number	See page 20.
0x30	Unload Drive	See page 20.
0x31	Load Drive	See page 21.
0x50	Perform Read/Write Test	Destructive test: this command must be issued after a scratch cartridge is loaded - test will take 1 to 5 minutes
0x51	Perform Head Test	Destructive test: this command must be issued after a scratch cartridge is loaded - test will take 5 to 10 minutes
0x52	Perform Media Test	Destructive test: this command must be issued after a scratch cartridge is loaded - test will take 5 to 10 minutes
0x53	Short Read/Write Diagnostic	This is a quick diagnostic to test out reading and writing. A cartridge is required.
0x55	Perform Self-Test	No cartridge required.
0x60	Firmware Update From FMR Cartridge	Must be issued immediately before the FMR cartridge is loaded. The drive allows 60 seconds from the time that command is issued until the cartridge is successfully loaded and comes ready.
0x61	Create Drive FMR Cartridge	Must be issued immediately after the scratch cartridge is loaded.
0x62	Erase Drive FMR Cartridge	Must be issued immediately before the FMR cartridge is loaded. The drive allows 60 seconds from the time that command is issued until the cartridge is successfully loaded and comes ready.
0x63	Read Buffer	See page 21.
0x64	Write Buffer	See page 22.
0x65	Read Tape	See page 23.
0x66	Write Tape	See page 23.
0x70	Write Drive Dump to Tape	Must be issued immediately after a scratch cartridge is loaded.
0x71	Force Drive Dump	
0x80	Reset Drive	
0x81	Force Check One Dump and Reset Drive	
0x90	SCSI Wrap Test	See the applicable drive service guide Setup, Operator, & Service Guide for information on when to insert and remove the wrap tool.
0xB0	Setup Forced Error Logging	
0xB5	Read Log Sense Data	See page 24.
0xC0	Set Traps	See page 24.
0xC1	Remove Traps	See page 24.

Maint_Command for Device Offline

The Device Offline sub-command is used to take the drive offline on the SCSI bus. Any attempt to select a drive while it is offline results in a selection timeout. If a SCSI command is in progress or if the drive does not go offline within 60 seconds after receiving this command, the drive will respond with an error.

Byte	Description
0	Msg Type (0xAB)
1	Tgt Addr
2 ... 5	Msg ID
6	Msg Subtype (0x30)
7	Maintenance Sub-command (0x00)

Maint_Command for Device Online

The Device Online sub-command is used to make the drive online on the SCSI bus. Any attempt to select a drive while it is offline results in a selection timeout. Placing the drive back online will allow the drive to respond to selection again. The drive is “guaranteed” to go online.

Byte	Description
0	Msg Type (0xAB)
1	Tgt Addr
2 ... 5	Msg ID
6	Msg Subtype (0x30)
7	Maintenance Sub-command (0x01)

Maint_Command for Set Drive Serial Number

The Set Drive Serial Number sub-command is immediately followed by 13 bytes including a 12-byte serial number which is to be written into drive VPD and a 1-byte checksum of the 12-byte serial number.

Byte	Description
0	Msg Type (0xAB)
1	Tgt Addr
2 ... 5	Msg ID
6	Msg Subtype (0x30)
7	Maintenance Sub-command (0x20)
8 ... 19	(MSB) Drive Serial Number (12 bytes, ASCII) (LSB)
20	Checksum for Drive Serial Number

Maint_Command for Unload Drive

The Unload Drive sub-command is immediately followed by 1 optional parameter byte (if omiitted, it will be treated as a value of 0x00).

Byte	Description
0	Msg Type (0xAB)
1	Tgt Addr
2 ...	Msg ID

5	
6	Msg Subtype (0x30)
7	Maintenance Sub-command (0x30)
8	Unload Options

LEGEND:

Unload Options

- Bits 7-2: Reserved
- Bit 1: Disable Eject
  - 0: Unthread and eject
  - 1: Unthread only - do not eject.
- Bit 0: Immediate Response
  - 0: The drive is to respond with Maint_Status_Good or Maint_Status_Error when the requested unload operation is complete.
  - 1: The drive is to respond with Maint_Status_Good immediately and the library may poll the completion of the unload operation using Drive_Status_Request.

Maint_Command for Load Drive

The Load Drive sub-command is immediately followed by 1 parameter.

Byte	Description
0	Msg Type (0xAB)
1	Tgt Addr
2 ... 5	Msg ID
6	Msg Subtype (0x30)
7	Maintenance Sub-command (0x31)
8	Load Options

LEGEND:

Load Options

- Bits 7-2: Reserved
- Bit 1: Disable Thread
  - 0: Load and thread
  - 1: Load only - do not thread.
- Bit 0: Immediate Response
  - 0: The drive is to respond with Maint_Status_Good or Maint_Status_Error when the requested load operation is complete.
  - 1: The drive is to respond with Maint_Status_Good immediately and the library may poll the completion of the load operation using Drive_Status_Request.

Maint_Command for Read Buffer

The Read Buffer sub-command is immediately followed by 7 parameter bytes. These parameter bytes indicate the buffer that is to be read, the offset within the buffer, and the maximum amount of buffer data to be sent in the response to this message. For additional details, see *the applicable tape drive SCSI reference*.

Byte	Description
0	Msg Type (0xAB)
1	Tgt Addr
2 ... 5	Msg ID
6	Msg Subtype (0x30)
7	Maintenance Sub-command (0x63)

8	Buffer ID
9 ... 11	(MSB) Buffer Offset (LSB)
12 ... 14	(MSB) Allocation Length (LSB)

Below is a table describing Buffer ID support for the Read Buffer command:

Buffer ID	**Buffer Definition**
0 and 4 (please use buffer 4--buffer 0 may change)	Firmware image from non-volatile storage.
1	Firmware trace (dump) from DRAM.
2	Test buffer.
3	Entire VPD image from DRAM.
5	Entire CM. Information is provided even if tape is not loaded--whatever is stored in DRAM.
6	Drive error log.

Maint_Command for Write Buffer

The Write Buffer sub-command is immediately followed by 6 parameter bytes and the data to be written. These parameter bytes indicate the buffer that is to be written, the offset within the buffer, and the amount of data included in this message. For additional details, see *the applicable tape drive SCSI reference*.

Byte	**Description**
0	Msg Type (0xAB)
1	Tgt Addr
2 ... 5	Msg ID
6	Msg Subtype (0x30)
7	Maintenance Sub-command (0x64)
8	Buffer ID
9 ... 11	(MSB) Buffer Offset (LSB)
12 13	(MSB) Parameter List Length (n-13) (LSB)
14 ... n	Parameter List Data

Below is a table showing Buffer ID support for the Write Buffer command:

Buffer ID	**Buffer Definition**
0 and 4 (please use buffer 4--buffer 0 may change)	Firmware image to non-volatile storage. Currently does not reset after write is complete.
2	Test buffer.
6	Clears drive error log. Any data sent is ignored. Size of data sent is also ignored.

Maint_Command for Read Tape

The Read Tape sub-command is immediately followed by 3 parameter bytes. These parameter bytes indicate the maximum amount of tape data to be sent in the response to this message. The drive will read from beginning-of-tape until one of the following occurs: a filemark is detected, end-of-data is detected, or the maximum amount of data is sent (Allocation Length).

Byte	Description
0	Msg Type (0xAB)
1	Tgt Addr
2 ... 5	Msg ID
6	Msg Subtype (0x30)
7	Maintenance Sub-command (0x65)
8 ... 10	(MSB) Allocation Length (LSB)

Maint_Command for Write Tape

The Write Tape sub-command is used to write to a scratch cartridge in the tape drive that has already been loaded. It is the responsibility of the library to ensure this is a cartridge that may be written. Writing begins at Beginning Of Tape. There really isn't any concept of a logical block in this command. Data is just pushed down to tape. The data is sent from the library to the drive in sections. If at any point an error occurs, a Maint Status_Error message is sent to the library. At this point the library Write Tape command series of messages is considered ended. A retry will require a re-issue of the entire series of messages starting back with a section number of 0.

The drive returns a Maint_Status_Good message each time about 4096 bytes have been sent within the same series of messages. This is for feedback that writes are in progress. A Maint_Status_Good message is also sent after the very last message in the series, indicated by Section Flag of 0x00.

Byte	Description
0	Msg Type (0xAB)
1	Tgt Addr
2 ... 5	Msg ID
6	Msg Subtype (0x30)
7	Maintenance Sub-command (0x66)
8	Section Number
9	Section Flag
10 11	(MSB) Section Size (n-11) (LSB)
12 ... n	Section Data

LEGEND:

Section Number

This unsigned value must be 0 for the first section. Subsequent sections must be incremented by a value of one. If 0XFF is reached, the next section number wraps to 0x00.

Section Flag

The section flag must be 0x00 for the last section. This causes all data in the buffer to be flushed followed by a rewind.
The section flag must be 0x80 for all other sections.

Section Size

The maximum section size is 494.

Maint_Command for Read Log Sense Data

This sub-command requests SCSI Log Sense Data - this does not clear any flags including TapeAlert.
The Read Log Sense Data sub-command is immediately followed by 3 parameter bytes. These parameter bytes indicate the Log Page that is to be returned and the maximum amount of log data to be sent in the response to this message. Note that the granularity is only at the Log page code level. The parameter code cannot be specified. The format of the data returned is exactly like that of log parameter data for the SCSI command. For additional details, see *the applicable tape drive SCSI reference*.

Byte	Description
0	Msg Type (0xAB)
1	Tgt Addr
2 ... 5	Msg ID
6	Msg Subtype (0x30)
7	Maintenance Sub-command (0xB5)
8	Page Code
9 10	(MSB) Allocation Length (LSB)

Maint_Command for Set Traps

The Set Traps sub-command is immediately followed by 2 bytes. The 2 bytes indicate the Fault Symptom Code to trap on. This sub-command works the same as the SCSI Send Diagnostic routine of the same name.

Byte	Description
0	Msg Type (0xAB)
1	Tgt Addr
2 ... 5	Msg ID
6	Msg Subtype (0x30)
7	Maintenance Sub-command (0xC0)
8 9	(MSB) Fault Symptom Code (LSB)

Maint_Command for Remove Traps

The Remove Traps sub-command is immediately followed by 2 bytes. The 2 bytes indicate the Fault Symptom Code to no longer be trapped on. This sub-command works the same as the SCSI Send Diagnostic routine of the same name.

Byte	Description
0	Msg Type (0xAB)
1	Tgt Addr
2	

... 5	Msg ID
6	Msg Subtype (0x30)
7	Maintenance Sub-command (0xC1)
8 9	(MSB) Fault Symptom Code (LSB)

Maint_Status_Good (Drive-to-Library, Type=0xAB, Subtype=0x3C)

The Maint_Status_Good message is a non-error response to a Maint_Command. The Msg ID field will be copied from the initial Maint_Command message. Data is only returned in response to the Read Standard Inquiry Data (0x10), Read Request Sense Data (0x11), Read Device Identification Data (0x12), Read Buffer (0x63), Read Tape (0x65), and Read Log Sense Data (0xB5) sub-commands.

Maint_Status_Good with No Data

Byte	Description
0	Msg Type (0xAB)
1	Tgt Addr (0xFF)
2 ... 5	Msg ID
6	Msg Subtype (0x3C)
7	Section Number (0x00)
8	Section Flag (0x00)
9 10	(MSB) Section Size (0x0000) (LSB)

Maint_Status_Good with Single Data Section

Byte	Description
0	Msg Type (0xAB)
1	Tgt Addr (0xFF)
2 ... 5	Msg ID
6	Msg Subtype (0x3C)
7	Section Number (0x00)
8	Section Flag (0x00)
9 10	(MSB) Section Size (n-10) (LSB)
11 ... n	(MSB) Section Data (LSB)

Maint_Status_Good with Multiple Data Sections

If the data return size exceeds the packetable maximums, it must be sent in multiple sections. This may apply to the Read Buffer (0x63), Read Tape (0x65), and Read Log Sense Data (0xB5) sub-commands. The data can be broken into any convenient number of packets. The packets will be sent in order.

Byte	Description
0	Msg Type (0xAB)
1	Tgt Addr (0xFF)
2	

... 5	Msg ID
6	Msg Subtype (0x3C)
7	Section Number
8	Section Flag
9 10	(MSB) Section Size (n-10) (LSB)
11 ... n	(MSB) Section Data (LSB)

LEGEND:

Section Number

Section number of the current packet, starting at 0x00 and incrementing by 1, with 0xFF followed by 0x00.

Section Flag

- 0x00: Indicates this is the last section to be sent.
- 0x80: Indicates there are more sections to be sent
- All other values: Reserved

Maint_Status_Error (Drive-to-Library, Type=0xAB, Subtype=0x3F)

The Maint_Status_Error message is an error response to a Maint_Command. The Msg ID field will be copied from the initial Maint_Command message.

When error data is returned, the error data will be in the format of SCSI Request Sense data.

Byte	Description
0	Msg Type (0xAB)
1	Tgt Addr (0xFF)
2 ... 5	Msg ID
6	Msg Subtype (0x3F)
7	Section Number (0x00)
8	Section Flag (0x00)
9 10	(MSB) Error Data Size (n-10) (LSB)
11 ... n	(MSB) Error Data (LSB)

Drive_Status (Drive-to-Library, Type=0xAB, Subtype=0x40)

If the interface is operating in non-polled mode (Configuration Flags bit 7 set to 1), the Drive_Status message will be sent asynchronously when

- the drive receives a Set_Config message and does not require a reset,
- any cartridge is loaded in the drive, or
- any cartridge is unloaded from the drive.

The Drive_Status message is also sent in response to a Drive_Status_Request message from the library. Any transmission of this message may or may not reflect a state change, so library logic should not be "edge" or transition triggered. The library should store this information in a per-drive buffer, or issue explicit Drive_Status_Request messages, as needed.

Byte	Description
0	Msg Type (0xAB)
1	Tgt Addr (0xFF)
2 ... 5	Msg ID
6	Msg Subtype (0x40)
7	Drive Status Flags 1
8	Additional Status Length (0x17)
9	Display Character
10	Display Character Rate
11	Status LED
12	Status LED Rate
13	Tape Motion
14 ... 21	(MSB) Tape VolSer (LSB)
22 ... 29	(MSB) TapeAlert Flags (LSB)
30	Drive Status Flags 2
31	Drive Status Flags 3

LEGEND:

Drive Status Flags 1

- Bit 7: Cartridge Not Loaded
  - 0: A cartridge is loaded in the drive.
  - 1: No cartridge is loaded in the drive (cartridge could be present in ejected/loadable position).
- Bit 6: Drive Clean Required
  - 0: Drive does not need to be cleaned.
  - 1: Drive needs to be cleaned.
- Bit 5: Reserved
- Bit 4: Write-Protected Cartridge
  - 0: No cartridge is loaded in the drive or the loaded cartridge is not write protected
  - 1: A write protected cartridge is loaded in the drive
- Bit 3: Compression Enabled
  - 0: Compression is disabled in the drive
  - 1: Compression is enabled in the drive
- Bit 2: Cartridge Present
  - 0: A cartridge is not present in the drive
  - 1: A cartridge is present in the drive (could be in ejected/loadable position)
- Bit 1: Reserved
- Bit 0: LUN 0 Ready (Sequential Access Device)
  - 0: LUN 0 is in a SCSI Not Ready condition
  - 1: LUN 0 is in a SCSI Ready condition

Note: The cartridge position can be derived from bits 7, 2, and 0 of the Drive Status Flags 1, as follows:

Bit 7	Bit 2	Bit 0	Cartridge Position
1	0	0	Cartridge not present
1	1	0	Cartridge present in ejected/loadable position
0	1	0	Cartridge present in LTO-CM accessible position, but not threaded
0	1	1	Cartridge present, loaded, and threaded

Display Character

This ASCII field mirrors the single-character display on the front of the tape drive. Possible values include “0” to “9”, “A”, “b”, “C”, “d”, “E”, and “H”. For additional details, see the *applicable drive product specification*.

Display Character Rate

This field indicates the current flashing rate, in Hertz, of the single-character display on the front of the tape drive. For additional details on the display, see the *applicable drive product specification*.

- 0x00: Off, blank
- 0x01: Blinking at 1 Hz
- 0x02: Blinking at 2 Hz
- 0x04: Blinking at 4 Hz
- 0x7F: On solid, no blinking

Status LED

This field mirrors the dual-color status LED on the front of the tape drive. For additional details, see the *applicable drive product specification*.

- 0: Off
- 1: Green
- 2: Yellow

Status LED Rate

This field indicates the current flashing rate, in Hertz, of the dual-color status LED on the front of the tape drive. For additional details one the display, see the *applicable drive product specification*.

- 0x00: Off, blank
- 0x01: Blinking at 1 Hz
- 0x02: Blinking at 2 Hz
- 0x04: Blinking at 4 Hz
- 0x7F: On solid, no blinking

Tape Motion

This field (decimal) indicates the current unique tape motion being done.

- 0: No current motion
- 1: Writing
- 2: Reading
- 3: Erasing
- 10: Locating
- 11: Rewinding
- 20: Cleaning
- 30: Unloading
- 32: Loading

Tape VolSer

This field indicates the Volume Serial Number recorded by the host software on the tape media or in the CM of the currently loaded cartridge, if this can be detected by the tape drive. This is not to be confused with the cartridge serial number that is stored in the CM by the cartridge manufacturer.

TapeAlert Flags

This bit-map indicates the current state of TapeAlert flags as they would be reported in response to a SCSI Log Sense command. Byte 22, bit 7 indicates the state of TapeAlert Flag 1 and Byte 29, bit 0 indicates the state of TapeAlert Flag 64. Any flag that has been reset by SCSI Log Sense will be indicated as such for this method of reporting. Note also that this method of reporting will not reset any TapeAlert flag.

Drive Status Flags 2

- Bit 7: Drive Communication Type
  - 0: SCSI
  - 1: Fibre Channel

- Bit 6: Reserved
- Bit 5: Reserved
- Bit 4: Offline
  - 0: Online
  - 1: Offline
- Bit 3: Hard AL_PA (Has meaning only for Fibre Channel attached drives)
  - 0: Soft AL_PA addressing, AL_PA determined by drive during LIP processing
  - 1: Hard AL_PA addressing, AL_PA set using LID provided by library or hardware switch settings
- Bit 2: AL_PA Conflict (Has meaning only for Fibre Channel attached drives)
  - 0: No conflict
  - 1: Another device has the AL_PA or no address is available
- Bit 1: Light Detected (Has meaning only for Fibre Channel attached drives)
  - 0: No light detected, cable disconnect is possible
  - 1: Light detected, cable is connected
- Bit 0: Loop Initialized (Has meaning only for Fibre Channel attached drives)
  - 0: Loop Initialization (LIP) is not complete
  - 1: Loop Initialization (LIP) is complete

Drive Status Flags 3

- Bit 7: POST Processing
  - 0: POST processing is complete
  - 1: POST is currently processing
- Bit 6: Reserved
- Bit 5: Reserved
- Bit 4: Reserved
- Bit 3-0: Tape Cartridge Type
  - 0000b: No tape present or cartridge not mounted successfully
  - 0001b: Standard LTO Generation 1 format (per U18 spec.)
  - 0010b: Cleaner cartridge
  - 0011b: Drive FMR (firmware update) tape created via the Maintenance Panel
  - 0100b: Drive FMR (firmware update) tape created via the Maint Command Create Drive FMR Cartridge (0x61)
  - 0101b: Reserved
  - 0110b: Invalid cleaner cartridge
  - 0111b: Invalid cartridge type
  - 1000b - 1111b: Reserved

Drive_Status_Request (Library-to-Drive, Type=0xAB, Subtype=0x41)

The Drive_Status_Request message may be used to request an explicit update of the drive status. The drive will respond with a Drive_Status message. This message is not normally required. This message is unique in that it may be sent even though another command is already in progress.

Byte	Description
0	Msg Type (0xAB)
1	Tgt Addr
2 ... 5	Msg ID
6	Msg Subtype (0x41)

Interface Scenarios

Scenario for Handshaking Startup Process (Drive-Initiated)

When the drive powers up or is reset, the drive will return a NAK for any Two_Way message received from the library until the handshaking startup process is completed (this process may take about five seconds).

If the interface is operating with a default of polled mode (settable via the drive feature switches), the library will be responsible for initiating the handshaking startup process as described in Scenario for Handshaking Startup Process (Library-Initiated).

If the interface is operating with a default of non-polled mode, the drive will initiate the handshaking startup process by sending the Config_Request message. This message can be expected within ten seconds from when the drive powers up. The library should respond with a Set_Config message with the current drive configuration information. The drive may need to reset itself depending on the current configuration and the values received (this process may take about a minute). The table below describes when changes will cause a reboot:

Field	Reboot
LDI Address (Tgt_Addr)	N
SCSI/FC Address	Y (N if within 10 seconds of Power On Reset)
Drive Fibre Channel WorldWide Node Name	Y
Configurations Flag bit 3, Disable automatic drive online	Y

Note: All drive firmware levels accept bytes 0-53 of the Set_Config. One way to avoid problems is to always start with a Set_Config of just these first 54 bytes. Additional Set_Config commands can be subsequently sent depending on Drive_Status (for example, indicating the drive is a Fibre Channel drive) and the need of the library to set additional fields.

After the Set_Config, the drive will send a Drive_Status message if the drive is not in the process of resetting and if the interface is operating in non-polled mode. If the drive is in the process of resetting, the entire process is repeated for that drive beginning with the drive sending the Config_Request message. When the drive accepts Set_Config, that drive is now considered available (the drive can accept Two_Way messages).

To summarize, the handshaking startup process is as follows:

Config_Request	Drive-to-Library	Request for library configuration (starts handshake)
Set_Config	Library-to-Drive	Library configuration information (completes handshake)
Drive_Status	Drive-to-Library	Initial drive status (if Set_Config of non-polled mode)

Scenario for Handshaking Startup Process (Library-Initiated)

When the drive powers up or is reset, the drive will return a NAK for any Two_Way message received from the library until the handshaking startup process is completed (this process may take about five seconds).

When the library powers up, is reset, or detects an unexpected NAK from the drive, it may optionally initiate the handshaking startup process.

To initiate the handshake startup process, the library sends an asynchronous Set_Config message with the current drive configuration information. The drive may need to reset itself (this process may take about a minute) depending on the current configuration and the values received. The table below describes when changes will cause a reboot:

Field	Reboot
LDI Address (Tgt Addr)	N
Drive FC/SCSI Address	Y (N if within 10 secs of Power On Reset)
Drive Fibre Channel World Wide Node Name	Y
Configurations Flag bit 3, Disable automatic drive online	Y

Note: All drive firmware levels accept bytes 0-53 of the Set_Config. One way to avoid problems is to always start with a Set_Config of just these first 54 bytes. Additional Set_Config commands can be subsequently sent depending on Drive_Status (for example, indicating the drive is a Fibre Channel drive) and the need of the library to set additional fields.

After the Set_Config, the drive will send a Drive_Status message if the drive is not in the process of resetting and if the interface is operating in non-polled mode. When the drive is in polled mode, the library can discover when the Set_Config is done and the drive is ready for normal operations by issuing a Drive_Status_Request. If Drive Status Byte 3 bit 7, POST Processing, is returned as 0, the drive is ready. If the drive is in the process of resetting, the handshaking startup process is repeated once for that drive. When the drive accepts Set_Config, that drive is now considered available (the drive can accept Two_Way messages).

To summarize, the handshaking startup process is as follows:

Set_Config	Library-to-Drive	Library configuration information (completes handshake)
Drive_Status	Drive-to-Library	Initial drive status (if Set_Config of non-polled mode)