Imagine COMMUNICATIONS™

Developer's Guide

Nexio AMP® Protocol

Version 7.5

May 2014

Publication Information

This publication is designed to assist in the use of the product as it exists on the date of publication of this manual, and may not reflect the product at the current time or an unknown time in the future. This publication does not in any way warrant description accuracy or guarantee the use for the product to which it refers. Imagine Communications reserves the right, without notice to make such changes in equipment, design, specifications, components, or documentation as progress may warrant to improve the performance of the product.

Trademarks

Contact Information

Imagine Communications has office locations around the world. For locations and contact information see:
http://www.imaginecommunications.com/company/contact-us.aspx

Support Contact Information

For support contact information see:

- Support Contacts: http://www.imaginecommunications.com/services/technical-support.aspx
- eCustomer Portal: http://support.imaginecommunications.com

Revision History

Revision	Date	Description of Changes
7.5	May 2014	– Rebrand as Imagine Communications –
7.0	August 2013	NOTE: With the addition of auto-input resolution detection and the ability to separately record all four major video resolutions on the same input, the legacy Get and Set Record Parameter commands (C0 C0, C0 C1, C8 C1, C9 C1) are no longer valid when auto-input detection is enabled (the default as of the Nexio 7.0 Software Release). Instead use the new Get and Set Auto-Detect Record Parameters commands (C5 CF and CF CF). In addition, the Get and Set Special Channel Properties (C4 B1 and CF B1) commands for bit 20 to get/set the audio routing input mask of a channel is replaced by the Get and Set Auto-Detect Record Parameters command. – B0 07 VDCP command supports new VdcpActiveIdRequest registry to indicate a channel is active if a clip is in the STILL state. – 30 05 VDCP Status command includes new extended status to indicate a stacked clip is ready to play (PVW-READY) – New Channel Properties in the C4 B1 command to indicate a channel's current AFD mode for both cases when a clip requires an aspect ratio conversion on play out or not. – New mode for Get Disk Information (C1 C7) to handle storage systems up to 1,000,000 hours – New protocol commands to manage audio track routing, their profiles, and the dynamic selection of these profiles: C2 56, CF 57, C1 16, and C1 17. – The Audio Track Routing introductory paragraph has been expanded to include the new commands available.
6.5	July 2012	– Removed references to old VR series of servers, non-Unicode systems and systems with 8-byte clip IDs – refer to earlier SDK releases to support those platforms and features – Additions to Device Type Request (00 11) command to support detection of Film Rate – Added MinFrame field to C8 4A command to report the frame offset from the first video packet in a clip to its actual start of message frame – New channel status indicators where the serve r experiences excessive drift or media playback is lacking video or audio. – Extended field 9 now called "Title"

Contents

Chapter 1: Introduction

About This Document

This developers' guide is intended to be used as a reference source for software development companies wishing to manage the contents and control the recording and playback of Imagine Communication's Nexio™ Server product line. The information in this document is proprietary and confidential and requires a signed and approved non-disclosure agreement.

This document defines and describes the protocols that are used to control the **Low Level Module (LLM),** the application that serves as the backbone of the Nexio Server system. The LLM is the clustering software that enables our video servers and the shared storage to communicate, share data, and tasks. It is also to the LLM that controllers make a socket or serial connection in order to control the servers.

Chapter One of this document provides general information about the protocols and communications with the Nexio servers. It covers the basic methods for connecting to a video channel through the LLM, whether through an RS-422 connection or over Ethernet.

Chapters Two and Three cover the specific protocols supported by Nexio Servers: our own native protocol in Chapter Two and the VDCP protocol in Chapter Three. Our native protocol implements subsets of the following manufacturer's specifications:

- Sony 9-Pin Protocol Document 9-966-979-04
- Odetics Command and Control Specification Document 4136201-A
- Nexio Control Protocol
- VDCP Video Disk Communications Protocol

The Nexio implementation of the VDCP protocol is described based on the most recent VDCP manual available to the public in February 2003. Additions to and omissions from the standard are noted where appropriate.

The implementation of these protocol interfaces are constantly changing, being enhanced to better serve our customers. Every effort is made to keep the protocol backward compatible as changes are made. However there is no guarantee that all future enhancements and fixes will not affect a controller's current interface to the Nexio server. Check back regularly for updates to this document and we strongly recommend gaining access to our new software releases as soon as they are available.

General Information

This document reflects the protocol implementation in the LLM as of May 2014. It includes the latest protocol updates released with the Nexio AMP 7.5 Software Release.

While this manual is targeted toward the Nexio AMP and Volt Servers, most of the protocol commands detailed in this manual are backward compatible to previous Nexio and VR products. For the most part, there are no major differences in how the different servers respond to automation commands. They make use of the same basic command sets, structures, and timeouts. The only real difference is in feature development and bug fixes.

Where possible, this document will indicate when commands were first implemented and make note of the applicable software release where they were introduced.

Control Communication

There are three pathways for a controller to communicate with the Nexio Servers: serially through RS-422, or via Ethernet through TCP and UDP. And while the native Nexio Server protocol supports all three of these, the VDCP specification only supports control over RS-422.

RS-422 Control

RS-422 communication with the Nexio servers is configured at **38400 baud, 1 start bit, 8 data bits, ODD parity, and 1 stop bit.** This method requires that the content of server/client conversations bears the format described in the sections in Chapters Two and Three about Command and Response structures.

Nexio AMP Servers and later versions of the Nexio XS Servers use RJ-12 connectors for their RS-422 connections. All previous Nexio and VR Servers use 9-pin connectors.

Please consult the appropriate server's User Manual for a description of pinouts and adapters available. Operators should also consult the manual to learn how to set up each RS-422 connection to a particular channel.

It's important to understand that COM port 1 does not necessarily control Channel 1 on the server. There are LLM settings that control the relationship between the channels under control and the COM ports through which they can be controlled.

You can use the Nexio Config utility to determine which COM port controls which Nexio channel. There is a special tab in Nexio Config called Automation/Serial for preparing the port control. For each serial port there will be three protocol options: Sony, Harris, or Louth, plus the appropriate server channel to assign to the port. "Sony" represents the Nexio Server native protocol. Both "Harris" and "Louth" represent slightly different versions of VDCP. The difference is based on the interpretation of changes to the VDCP manual. There was a change in the August 2000 manual in how a server should report its current timecode position. The pre-2000 definition requires the protocol setting of "Louth". The 2000 and beyond definition requires a setting of "Harris".

Once set and the LLM is launched, the settings for each serial port and each server channel are hard-wired. Even though there are protocol commands to select a different channel than the one assigned to the port, the LLM will only control those ports as specified.

For example, in order to control a channel in the Nexio native protocol, a controller must select the channel (Codec) that it wants to control. The **Set Logical Channel (C1 AC)** command is used for this purpose. All subsequent commands on the same socket will be routed to the selected channel. However, when controlling a Nexio Server using RS-422, a controller can only actually communicate with the one specific channel per COM port setting. The controller must make a connection through a different COM port in order to communicate with a different channel.

Similarly in VDCP protocol, a controller is required to first open a port (channel) with the **Open Port (30 01)** command and then select it with the **Select Port (20 22)** command. Even though each channel is hard-wired to a COM port, the controller must follow the protocol specification.

UDP Control

UDP communication to the server is conducted on **port 331** unless otherwise specified and configured. This method requires that the content of server/client conversations bear the following format:

Table 1-1: UDP Datagram Format

Field	Format	Description
ID NUMBER	4 bytes unsigned integer, MSB first	This is the identification number of a command or response. This field should have different values for adjacent commands (for example incremented mod 2^{32} at every new command), unless the command is a retry of the previous command (i.e., resulting from a response time out). In the latter case, the retried command will bear the same identification number as the original (failed) command.
CHANNEL	1 byte unsigned integer	This field indicates the Codec to which the command or response should be routed. Valid channels are 0x00 (Channel 1), 0x01 (Channel 2), 0x02 (Channel 3), 0x03 (Channel 4).
DATA0 – DATA(N-1)	N bytes, size depends on command	This field contains either a command or a response. It includes all parts except the checksum. See the section on Command and Response Structures for details on the structure of commands and responses.

The controller is responsible for managing its own timeouts and retries. It should ignore any responses that don't match the last command's identification. If the controller does not receive a response that matches the last command's identification within the protocol's timeout, it may either retry the failed command using the same identification number or proceed to the next command using a new identification number.

If the server receives two or more consecutive commands bearing the same identification numbers, it will only execute and respond to the first instance. After which it will simply return duplicates of the first response for every subsequent instance without actually executing them. This is to give the controller the option of retrying unacknowledged commands without the danger of having them executed repeatedly by the server.

TCP/IP Control

TCP/IP communication is conducted on **port 557** of the server unless otherwise specified and configured. This method requires that the content of server/client conversations bear the following format:

Table 1-2: TCP/IP Message Format

Field	Format	Description

Field	Format	Description
CMD-1	1 byte unsigned integer	Contains Basic Command Group and Byte Count
CMD-2	1 byte unsigned integer	Command
DATA0 – DATA(N-1)	N bytes, size depends on command or response	Optional. One or more bytes containing data required to act upon or further interpret the specific command.

The checksum is not included.

For most command and control operations, the controller must select a channel against which it wants to perform the task. This is done with the **Set Logical Channel (C1 AC)** command.

The controller is responsible for managing its own timeouts and retries. If for any reason a controller crashes or cables become disconnected, the server will close the current connection and wait for a new connection, at which time, the controller can reconnect.

Dual LLM Control

As of the Nexio 5.5.0 software release (LLM version 6.07.97.130), new support was added for servers configured to operate with two instances of the LLM. In this special case, one LLM provides control over the high resolution channels on the server (Nexio Server System) and the other LLM provides control over the low resolution channels (Nexio Global Proxy System). This is the typical configuration for a server recording new clips simultaneously into both high- and low-res systems.

Most controllers need not be concerned with the Nexio Global Proxy system or the information that follows. It is listed here just in case the situation is encountered and questions are raised.

In a dual LLM control system there will be two sets of registry settings for configuring each individual system: the existing registry branch under [HKCU\Software\ASC Audio Video\LLM] for the high-res LLM and a new registry branch under [HKCU\Software\ASC Audio Video\LLM1] for the low-res LLM. In this way specific configurations related to IP addresses, node numbers, domain letters, and such can be clearly delineated between the two systems.

In addition there is a registry setting for providing an alternate port number for controlling the second LLM: a DWord registry setting called “TcpPortOffset”. The value entered here is an offset value added to the above stated port numbers for interfacing to the server. And even though the phrase “Tcp” is in the registry name, it also applies to the UDP control port number.

Typically this registry would only be set for the low-res LLM1 registry, leaving the high-res control set to the documented defaults. The TcpPortOffset value must be set to at least a value of 100 so that the ports for both LLMs do not conflict with each other. For example, a setting of 100 applied to the LLM1 registry branch would indicate high-res control would remain on TCP port 557 and low-res control would be available on TCP port 657.

By default the Nexio Config utility presets this LLM1 registry value to 100, even for servers that are not configured for the Nexio Browse System. It is safe to leave this LLM1 registry branch as long as the TcpPortOffset setting remains with its default value.

Chapter 2: Nexio Native Protocol

Command and Response

What follows is an overview of the Nexio Server Protocol command and response interaction between the controllers and servers. Please see the VDCP section for more detail on that protocol's command and response definition.

First, an overview of the basic components:

Table 2-3: Command Structure

Field	Format	Description
CMD-1	1-byte unsigned integer	Contains the basic command group and byte count
CMD-2	1-byte unsigned integer	Command
Byte Count (BC)	1-byte unsigned integer	Contains byte count if CMD-1 bits 0 to 3 are set to 1 **"0xF"**
DATA0 – DATA(N-1)	N bytes, size depends on command or response	Optional. One or more bytes containing data is required to act upon or further interpret the specific command.
CHECKSUM	1-byte unsigned integer	The LSB of the sum of all the bytes in the data stream from CMD-1 to DATA(N-1)

In the two-byte command structure, the first byte defines the command group and the second byte is the command within that group. Specifically bits 4-7 of the first byte identify the basic purpose of the command.

Table 2-4: Basic Command Groups

	Source	Description
0	Controller	High level control & ID query
1	Server	ACK/NAK & device ID
2	Controller	Mode and transport control
4	Controller	Editing and random access play setup
6	Controller	Sense requests
7	Server	Sense return
8	Server	ID data return
A	Controller	ID record/play
C	Controller	Extended commands
D	Server	Extended sense return
E	Controller	Sequence play commands

F	Server	Sequence sense return

Nexio Protocol Overview

There are certain functions almost every controller using the Nexio Server Native Protocol must perform in order to effectively manage and control the servers. The following pages attempt to detail some of the recommended methods for connecting to and controlling multiple server channels in a Nexio Server system.

There are a number of steps every controller should take before ever attempting to load a piece of media into a channel and sending the play or record command.

1. Identifying the capability and purpose of every channel of every server under the controller's command
2. Retrieving the list of clips in the Nexio storage system and all their special properties
3. Manipulating the media, their metadata, and their special properties
4. Loading and playing media in a channel
5. Recording media in a channel

Identifying System Configurations

It's the controller's responsibility to learn everything it can about every server and every channel in the system that is under its control. Depending on a controller's role in a system, it may need to learn about every channel in the system.

Information to retrieve should include:

- The basic characteristics of the server system, the metadata available, any general information about the media to be stored on the system, and the current and maximum storage limits of the system.
- Extra detail about the server channels available in the system, including the number of channels per server available and any restrictions on those channels.
- What the current status of each channel is, including if there is a clip loaded, the channel's play or record status, and whether anything is cued to play next.

There are a few ways to manage this system inspection:

- One is manually. Inspect the equipment and software visually. Check with Imagine Commubnciations technical support. Get a full report on the equipment installed and maintained.
- The second is programmatically. Below is a checklist of some of the various configuration options to be aware of and the Nexio protocol commands that can be used to return the information. The information is broken down according to what is applicable system-wide, what is applicable on a server-by-server basis, and what is applicable within each server on a channel-by-channel basis. It should be noted that much of this is not possible within the VDCP protocol.

The following options are configurable on a server-by-server basis. Each individual server must be queried for the following information.

Table 2-5: Server Configuration Options

Options	Variables	Command	Additional Notes
LLM Version	ASCII String	C4 B1 00 00 00 02	The LLM version number can give the controller a clue as to the most recent features implemented.
Number of Channels	2, 3, 4, 6, or 8	C1 AC 07 and C0 AC	First set the selected channel to the highest possible number (0x07, 0-based) and then check to see what the resulting channel reports it as
Media IDs playable	On or off	C4 B1 00 00 00 08	Nexio Timeline Playback Service must be installed and started if Media IDs are part of a system's workflow
Unicode Interface	On or off	C4 B1 00 00 00 20	Indicates if the server is communicating with the controller using Unicode characters for metadata fields that support them

Once a controller has determined the number of video I/O channels a server is equipped with, it then needs to determine what each channel is capable of doing. Some channels can only play clips, some can only record, some can do both.

Most have limitations as to what type of clips may be played or recorded in those channels. For example, one channel may be limited to support only SD clips while another might support all resolutions including 1080p.

The following options are configurable on a channel-by-channel basis. Each individual channel under automation control must be queried for the following information.

Table 2-6: Channel Configuration Options

Options	Variables	Command	Additional Notes
Channel Status	A series of returned bytes based on status modes queried	61 20	Current status of the channel, whether it's cued, playing, in record, etc.
Server Capabilities	1 byte variable	C4 B1 00 00 00 04	As of the 6.5 Software Release, determines the server hardware capabilities to support 1080p
Codec Capabilities	1 byte for the first 3 commands and 4 bytes for the last command	C4 B1 00 00 00 40 and C4 B1 00 00 00 80 and C4 B1 00 00 01 00 and C4 B1 00 00 80 00	Determines each channel's capabilities and what types of clips can be played or recorded in the channel
Video Formats and Resolutions	Variable	C4 B1 00 02 00 00	Indicates a list of all video formats and resolutions

Options	Variables	Command	Additional Notes
Supported			supported in the channel
Audio Routing and Tags	32 bytes for all but the last command which is 16 bytes	C4 B1 00 08 00 00 C4 B1 00 10 00 00 C4 B1 00 20 00 00 C4 B1 00 40 00 00 C4 B1 00 80 00 00 C4 B1 01 00 00 00	Gets the audio routing capabilities and tags for record and play out channels
Other Special Channel Settings	Variable	C4 B1 00 00 02 00 C4 B1 00 00 08 00 C4 B1 00 01 00 00 C4 B1 00 04 00 00 C4 B1 02 00 00 00	Various special configurations available on a channel-by-channel basis
Synced Channel Configuration	Bitwise display	C0 C8	Indicates if one channel controls any other channels. The controlling application should only direct transport commands to the master channel and not to the slave channel

Building a Database of Media

Once the controller has established the capabilities of the system, the servers, and the channels it is controlling, it will need to retrieve a list of the media on the Nexio Server system.

For every controller connection the server maintains:

- A database containing a list of all existing IDs available to the controller.
- An Added IDs List containing a list of IDs that have been created or modified since the last time the controller queried this list. This list behaves as a FIFO stack with items dropping from the list as they are read by the controller.
- A Deleted IDs List containing a list of IDs that have been deleted since the last time the controller queried this list. This list behaves as a FIFO stack with items dropping from the list as they are read by the controller.

In this way a controller can first retrieve a list of all media on the server system and then on a regular basis retrieve a list of the updates and deletions so as to keep up to date on all changes.

For every piece of media on the system, the LLM retains two separate IDs: an 8-byte ID handle and the more user-friendly 32-character ID (also described as 64-byte ID). The 8-byte ID handle is used by most protocol commands that affect the media. As a result, controllers must be prepared to associate every 32-character ID with its 8-byte handle. When the 32-character ID is used in this protocol, it is referred to in the LLM command listing as the "Extended ID".

Controllers are able to keep the two IDs in sync by the use of two LLM protocol commands:

- Get Extended ID from ID Handle (C8 C3)
- Get ID Handle from Extended ID (CF C4)

The process of retrieving these two sets of identifiers for each piece of media in the system requires a series of **List First/List Next** commands. Access to the database list of all clips is typically gained through use of the **List First ID Handle (A0 14)** and **List Next ID Handle (A0 15)** commands, which return the 8-byte ID handle for each ID on the system. Access to all of the lists is gained through the **List First ID List (C1 4C)** and **List Next ID List (C0 4D)** commands. The latter set of commands can return the 8-byte ID handles, the normal extended ID, or both.

The logical use of these commands is to first use the appropriate List First command followed by a series of List Next commands until all IDs have been returned. Only one clip at a time is retrieved with every use of these commands.

As a controller builds its list of IDs, it can also collect the metadata associated with every ID. The LLM maintains two distinct storage areas for metadata: the normal set and an extended set. In addition, the LLM maintains a list of special properties, or attributes, associated with each piece of media.

The **Get ID Metadata (C8 4A)** command is used to retrieve the normal set of metadata. The **Set ID Metadata (CF 44)** command is used to change the normal set of metadata associated with an ID. The following field descriptions are just an example of some of the fields available in the normal set of metadata.

Table 2-7: Normal Set of Metadata

Field	Description	Source of Metadata
Video Format	Indicates an ID's video format (MPEG2, DV, etc.) and aspect ratio	Based on record parameters of media
Video Bit Rate	Value in Mb/s for only the video portion of the media	Based on record parameters of media
Duration	Number of frames in the media	Based on record parameters of media, changeable to smaller values by controllers after the fact
Procamp Values	The Video Gain, Setup, Chroma Gain, and Hue values associated with an ID	Based on record parameters of media, changeable by controllers after the fact
Children	Number of subclips made from this parent clip	Incremented as subclips are created from the ID
Audio Tracks	Number of discreet audio tracks	Based on record parameters of media
Record Date	16-bit number to indicate month, day, and year the media was recorded. This field continues to exist even though it has been superseded by an extended field	Set at record time of media by LLM
Kill Date	16-bit number that pre-determines the day the media is scheduled to be labeled as "expired". The media does not actually get deleted or become disabled on the indicated day	User-entry field, settable at record time and changeable any time later

For more detail, including a complete listing of the normal set of metadata, review the **Get ID Metadata (C8 4A)** command in the command listing below in this chapter.

The extended set of metadata is retrieved using the **Get Extended Field (C9 C3)** command. However this command retrieves the extended fields one field at a time. A separate command must be sent to

retrieve each field the controller is interested in retrieving. The **Set Extended Field (CF CC)** command is used to change the metadata in one of the fields in the extended set.

Unlike the normal set of metadata, many fields in the extended set support Unicode characters. The following table includes some of the extended fields used most often.

Table 2-8: Extended Set of Metadata

Field	Description	Source of Metadata
Record Date/Time	64-bit field storing the time and date stamp of when the media was recorded	Automatically set at record time of media by LLM
Video Info	128-bit field storing extended video information indicating such things as video resolution, format and frame rate	Automatically generated by LLM
User Name	UNICODE field indicating the individual logged in to the Nexio system at record time	Automatically recorded by Nexio user applications such as NXOS and Velocity at record time
Description	UNICODE user entry text field for storing detail about media (up to 120 characters)	User entry field
Agency	UNICODE user entry text field for storing detail about media (up to 15 characters)	User entry field
User-definable Field #1	UNICODE user entry field (up to 25 characters)	User entry field
User-definable Field #2	UNICODE user entry field (up to 25 characters)	User entry field
User-definable Field #3	UNICODE user entry field (up to 25 characters)	User entry field
User-definable Field #4	UNICODE user entry field (up to 25 characters)	User entry field

For more detail including a complete listing of the extended set of metadata, review the **Get Extended Field (C9 C3)** command in the command listing below in this chapter.

In addition to metadata, there are also some special attributes available for every clip in the system. These are useful for a controller to detect special properties associated with a clip. It's also a good indicator to know which IDs the controller can ignore.

Use the **Get Special ID Attributes (C8 84)** command to retrieve an ID's special properties. The following table includes some of them:

Table 2-9: Special Attributes

Value	Attribute	Description
0x0001	Delete protected	The ID is protected from deletion until this attribute is unset
0x0002	Sliding ID	Indicates a circular recording in which the length is preset and material is automatically deleted from the front as it is recorded on the back
0x0004	Macro ID	Indicates the media is a timeline made up of pointers to other pieces of media, such as a Media ID

0x0010	Project File	Indicates a special file used by editing applications to represent a container of timelines, always with the Invisible attribute set
0x0020	Invisible ID	Indicates the file is hidden from the normal media database. Most controllers should not need to keep track of invisible files.
0x0200	Not Ready to Play	Indicates the first audio buffer of the media has not yet been written to disk. A controller should wait to load and play any clips when this attribute is set.
0x0400	Not Ready to Transfer	Indicates the ID is currently in record or being written to disk

The special attributes are cumulative, meaning media may have more than one attribute. For example, an ID with an attribute value of 0x0031 is an invisible (0x0020) project file (0x0010) that is delete protected (0x0001).

A controller can use the **Set Special ID Attributes (CC 84)** command to set some of these properties.

What follows is an example using the above Nexio protocol commands for building a database of media.

Table 2-10: Building a Database of Media

Protocol	Description
C8 B1 00 00 00 02 00 00 00 02	If not already established, sets up the interchange of data to be in Unicode, where appropriate
C1 4C 21	Retrieves the first ID in the system. Mode 21 retrieves both the 8-byte ID handle and the extended ID at the same time. More traditional methods of ID retrieval use A0 14 to get the first 8-byte ID handle followed by C8 C3 to get its matching extended ID.
C8 4A 25 30 30 30 30 31 32 33	Retrieves the normal set of metadata associated with the specified 8-byte ID handle retrieved in the previous command
C9 C3 25 30 30 30 30 31 32 33 00	Retrieves the metadata for the specified clip in the first extended field
C9 C3 25 30 30 30 30 31 32 33 *nn*	Retrieves the metadata for the specified clip in each of the remaining extended fields
C8 84 25 30 30 30 30 31 32 33	Retrieves the special attributes associated with the specified clip
C0 4D	Retrieves the next ID in the system including its 8-byte ID handle and extended ID
C8 4A 25 30 30 30 30 31 32 34	Retrieves the normal set of metadata associated with the next ID in the list
…	Continue this cycle of retrieving the next ID in the list and its associated metadata until the server returns no more IDs

Once the database of media is built, the controller then needs to keep up-to-date on any changes made to the list of clips. For each controller connected, the Nexio server keeps in memory a list of all additions, modifications, and deletions to the database.

It is up to the controller to request those changes on a regular basis. This is done via the **List First ID List (C1 4C)** and **List Next ID List (C0 4D)** commands, selecting one of the Added List or Deleted List modes. The Added List includes both new clip additions and existing clip modifications. It is up to the controller to determine if the ID listed in the Added List is a new clip or a modified clip.

How often the controller checks for changes to the database can be handled in any of three different ways:

- It can check every x number of seconds by sending the **C1 4C** command on a regular basis. The frequency should be no more often than every 2.5 seconds. This is how most Nexio applications operate.
- It can watch the Added and Deleted bits provided by the **Get Channel Status (61 20)** command. If a controller is regularly checking channel status, retrieval of the tenth status byte will let it know if there are new clips in the Added or Deleted Lists. Once notified, the controller then uses the **C1 4C** command to retrieve which clip (or clips) was added to the indicated list.
- It can set up an automatic notification from the Nexio server whenever a clip has been added to the Added or Deleted Lists. This is done using the **Change Notification (C4 A5)** command with masks 1 and 2 set. Once notified, the controller then uses the **C1 4C** command to retrieve which clip (or clips) was added to the indicated list.

As the controller receives the name of an ID that was added or modified, it should use the same set of metadata retrieval commands it used to build the original database of media.

Managing Media Assets

Once a controller has collected all the relevant information about each piece of media on the system, it will need to organize those assets such that any future actions involving those assets are appropriate. For example, it is up to the controller to know the configuration of every server channel. If the controller loads an HD clip into an SD channel that is not prepared to down-convert the HD clip, there could be a server failure.

Therefore, the first step in organizing the database of media (MediaBase in NXOS) is identifying the different types of clips on a system based on the three commands used in the previous section, the **Get ID Metadata (C8 4A), Get Extended Field (C9 C3), and Get Special ID Attributes (C8 84)** commands.

The following table reviews some of the different characteristics and types of media on the Nexio SAN, how a controller can determine the information, and why it's important.

Table 2-11: Identifying Media

Media Type	Detecting Via Protocol	Importance
Parent ID	**C8 4A command** Response data bytes 28-29	Determines if the ID has any subclips (child IDs) which protect it from deletion
Subclip	**C8 4A command** Response data bytes 00-07	If the data response is different from the ID's the 8-byte ID handle, a controller can determine the ID is a subclip which is merely a pointer to a piece of a Parent ID
Audio-Only Clip	**C8 4A command** Response data byte 8	A value of 0x15 indicates a clip that has no video stored in the file, only audio

Media Type	Detecting Via Protocol	Importance
Empty ID	**C8 4A command** Response data bytes 16-19	If the data response is 1, indicating an ID with duration of 1 frame, and all the rest of the ID's attributes indicate a normal clip, a controller can determine that the file is an Empty ID, which is a placeholder file created to reserve the ID for future use
Media ID	**C8 84** Response Mask of 0x0004	Identifies an ID that is a Media ID, essentially an EDL that includes pointers to other Parent IDs. The Timeline Playback service is required on servers that load and play Media IDs.
Consolidated ID	**C8 84** Response Mask of 0x0040 or 0x0044	Identifies a Parent ID that has had its entire media except for those areas protected by subclips erased from the storage. If a controller loads and plays a consolidated ID, those erased areas will not play out. Instead depending on how the clip was consolidated it may play black in those areas or skip them.
Looping Record (Sliding ID)	**C8 84** Response Mask of 0x0002	Identifies an ID that is a circular recording in which the length is preset and material is automatically deleted from the front as it is recorded on the back. These IDs are typically used for delay-to-air programming or for creating a series of subclips.
Looping ID	**C8 84** Response Mask of 0x0100	Identifies an ID that will play in a loop if loaded and played in a channel. Once it reaches its last frame, it will seamlessly transition back to the beginning of the clip.
Hidden Files	**C8 84** Response Mask of 0x0020	Identifies IDs that will normally not appear in the List First/Next lists unless specially requested by a controller with the C8 B1 command. Hidden files are usually system files used by editing applications.

It is essentially up to the individual controller to know what type of system it is controlling, the configuration of the server it is controlling, and the types of media on that system.

It's also up to the controller to manage the amount of material on the Nexio storage system and when some of that media should be removed to make room for more. The Nexio system includes a metadata field called the "Kill Date". This is purely an advisory field and does not actually delete the material on the indicated date. However, a controller can use it to help indicate when media is no longer required.

The Nexio protocol actually includes three different Erase ID commands. The most useful to a controller is the **Erase ID with Transaction (C8 4F)** command. It provides feedback to the controller as to the status of the media deletion, whether it failed, and if it didn't, why it didn't.

There are four primary reasons why an attempt to delete media fails:

- The clip doesn't exist on the system.
- The clip is protected from deletion by a subclip. Sometimes these subclips can be invisible to the controller.
- The clip has the Delete Protect special attribute set. This requires that a controlling application has set the attribute.
- The clip is currently loaded in a server channel and the clip protection logic is enabled on the server system. This requires specific LLM registries be set.

Based on the controller's collection of the database of media, it should know if the first three conditions exist. And if not, the **C8 4F** command will provide that information.

In the case of a clip protected by one or more subclips, the controller would first have to delete all the parent clip's subclips before it can delete the parent clip. Keep in mind, some of those subclips may be invisible. These hidden subclips are typically a result of the parent clip being active in a Velocity or other editing project. In that case, the media management of the system must take into account the removal of outdated editor projects before the outdated parent clips can be removed.

In the case of a clip protected by the Delete Protect attribute, the controller has the ability to unset that attribute before attempting to delete it. Just keep in mind that there was probably a good reason why the Delete Protect attribute was set. Typical uses of this attribute are by the Nexio MOS Playlist Manager application for all clips currently in a loaded rundown, by the Nexio FTP Server application while it is currently transferring the clip from one server system to another, and by an application user who has reason to want to protect the clip from deletion.

Last is the case where the clip is protected because it is currently loaded in a server channel somewhere on the Nexio system. This is an automatic feature to prevent the deletion of a currently loaded ID. However, by default this feature is not enabled on most Nexio systems. But if it is enabled, the **C8 4F** command will indicate which server currently has the clip loaded and which controller connection loaded it. Other than going to that server and unloading the clip, there is no direct method for clearing the connection so that the clip may be deleted.

Cueing and Playing Media

There are three different commands for cueing an ID for playback, each of which has four different sets of usage parameters:

- Cue Up with Data (2x 31)
- Fast Cue Up with Data (Cx 86)
- Preset In Point (4x 14)

The **Cue Up with Data** and **Preset In Point** commands behave exactly the same. The fastest way to cue an ID in the server is by using the **Fast Cue Up with Data** command. As long as the channel is prepared for playback, this will cause an instantaneous cue to field 1 of the requested frame. The drawback for this speed is that the latency of a subsequent play command is increased, meaning that after the **Play (20 01)** command, the video might start rolling a few milliseconds later than the controller is expecting. This command is typically only used for checking the contents of an ID by allowing an operator to load and look at the first frame. It was created to save processor time in such instances.

Once cued by any of these methods, the controller should check the channel's status bits via the **Get Channel Status (61 20)** command until the server returns a CUED status for the channel. The PST IN and

PST OUT bits also alert the controller if the currently loaded clip has a specified In and Out Point. After that, the controller need only send a **Play (20 01)** command to get the ID rolling (after the pre-determined latency has passed).

There is also a mechanism in the server protocol that allows a series of IDs to be played back-to-back in succession, producing a seamless transition between each clip.

The commands responsible for this are:

- Preset Preview In Point (Ax 04)
- Preset Preview Out Point(Ax 05)

This sequenced, or stacked, playback works in conjunction with the **Cue Up** commands where one clip is cued into the "Preset" slot and the other clip is loaded into the "Preview" slot. After the preset clip has completely played out, the preview clip becomes the new preset and the preview slot is free to accept another ID. This mechanism ensures that there will be an ID cued and ready for immediate play out at the moment that the current clip has completed.

If the preview slot is empty, the currently playing clip will stop at its Out Point, less one frame. The contents of the preset and preview slots can be replaced by sending new preset or preview commands. However, a controller should be careful not to change the clip in the preview slot too close to the preset Out Point of the clip currently in the preset slot.

A controller should regularly check the channel's status bits using the **Get Channel Status (61 20)** command to determine when the transition from one clip to another has occurred. The controller need only look for the PVW IN and PVW OUT bits to drop to know that the preview slot is available.

Sequenced playback is initiated with the **Play (20 01)** command. As mentioned, when the preset clip finishes playing, there is an automatic transition to the preview clip if one exists. A controller can trigger a premature transition to the next clip with the **Next Cue (C0 02)** command. Sequenced playback is terminated with a **Stop (20 00)** command or no more clips loaded in the preview slot.

What follows is an example using Nexio protocol for cueing and playing media in a stacked playlist. For more detail on the commands and their proper usage, please consult their individual descriptions later in this chapter.

Table 2-12: Cueing and Playing Media

Protocol	Description
C1 AC 00	Selects the desired channel to control. This assumes the controller has verified that the channel is capable of playing the desired media.
40 41	Sets the channel into auto mode, clearing the preset bits of In and Out Points
61 20 0F	Checks the current status of the channel to ensure that it is in a state to load and play a clip
28 31 25 30 30 30 30 31 32 33	Cues up the specified clip to its first frame
61 20 0F	Checks the channel status until the cued bit is set. A controller should use this command regularly to ensure the channel is doing what it expects.
20 01	Plays the cued clip
40 15	Sets the Out Point of the currently playing clip to its last frame

A8 04 25 30 30 30 30 31 32 34	Stacks the specified clip in the channel in preparation for play out once the currently playing clip finishes
A0 05	Sets the Out Point of the stacked clip
61 20 0F	Checks and verifies the status of the channel. When the first playing clip reaches its end, the stacked clip will automatically start playing and the preset preview status bits will drop.
A8 04 25 30 30 30 30 31 32 35	With the preset preview channel clear, another clip can be stacked and ready to play next

Recording New Media

In a normal Nexio system, it is not possible to simply start recording a new clip with just its ID. That's because most transport control commands on the server, including Record, require the 8-byte ID handle associated with the clip in order to cause an action. What the controller must do instead is first create what is called an "empty ID," a media file empty of any video or audio.

The process of creating the empty ID auto-generates an 8-byte ID handle, after which the media file can be filled with material through one of the **Cue Up To Record (Ax 02)** commands.

There are two sets of commands used to create these empty IDs: one based on creating an 8-byte ID handle **(CF Bx)** and one based on creating the extended ID **(CF C2)**. After creating the empty ID, the controller must follow up with a series of transaction commands. The first transaction command, **Get Transaction Status (C4 B3)**, should be sent repeatedly at greater than 50 millisecond intervals until it returns TRANSACTION_SUCCESS or an error. When the transaction is successful, the controller uses the **Get Transaction ID Handle (C4 B4)** command to retrieve the ID handle that was auto-generated during the transaction.

Once the empty ID is created, the controller can use the **Cue Up to Record (Ax 02)** command to load the file in an appropriate channel in preparation for recording. After using the **Get Channel Status (61 20)** command to verify that the **REC** bit is set, the controller can issue the **Record (20 02)** command to start the recording.

A controller can record the clip indefinitely by cueing the ID for record without setting a preset out value. The recording will not stop until it is explicitly stopped. If the controller knows the length of the recording, it can issue the **Preset Out Point (4x 15)** command any time before or during the recording. The recording will automatically stop at the specified preset Out Point.

The server is also capable of recording "Sliding IDs," also known as "Looping Records". These are media that record without end but only use up a specific amount of disk space. When the specified duration is reached, the record algorithm will loop back around to the front and begin recording over the front of the ID with new material. In this way, a controller can record a clip that never grows beyond its specified length.

Sliding IDs are useful for delay recordings in which a satellite feed is recorded into the server at a predetermined length and played back out on a separate channel an hour or more later. Another use allows subclips to be created from the sliding IDs as it records in a loop. Even as the sliding ID records back over itself, any material from which a subclip has been made is protected from being overwritten. In this way only the material desired from the sliding ID is preserved and the rest is released during the looping record.

To create a sliding ID, the controller must first **Cue Up to Record (Ax 02)** a newly created empty ID, set the **Preset Out Point (4x 14)** to indicate the desired loop length, then **Set Special ID Attributes (CC 84)**

for the sliding bit *before* executing the **Record (20 02)** command. The recording will not terminate unless it is explicitly stopped.

What follows is an example using Nexio protocol for creating a new clip and putting it into record. For more detail on the commands and their proper usage, please consult their individual descriptions later in this chapter.

Table 2-13: Recording New Media

Protocol	Description
C1 AC 00	Selects the desired channel to control. This assumes the controller has verified that the channel is capable of recording the desired media.
40 41	Sets the channel into auto mode, clearing the preset bits of In and Out Points.
61 20 0F	Checks the current status of the channel to ensure that it is in a state to load and play a clip.
C0 C0	Retrieves the channel's record parameters to determine if it matches the controller's needs.
C8 C1 03 01 01 32 00 00 04 01	If a change is needed, this sets up the channel for the record parameters the controller desires. This example specifies an MPEG2 I-Frame 50 Mb/s clip with 4 audio tracks and VBI enabled.
C4 B1 00 00 01 00	Retrieves the channel's record video format code and HD pixel aspect ratio- an important additional step to verify the channel is in the correct video resolution and other HD-related parameters.
CF B1 05 00 00 01 00 53	If a change is needed, this sets the channel's record video format code and HD pixel aspect ratio. This example specifies a 1080i NTSC clip.
CF C2 14 00 00 00 00 00 00 00 00 00 00 41 00 42 00 43 00 44 00 45 00	Creates an empty ID with the specified ID, in this case, "ABCDE" sent in Unicode. By sending 8 null bytes for the ID handle, the server will automatically create a unique ID handle. The server will return a transaction ID to the controller: "00 00 00 01" in this example.
C4 B3 00 00 00 01	Requests the status of the empty ID creation. When the status changes from pending to success, the controller can request the new ID handle automatically created.
C4 B4 00 00 00 01	Retrieves the new 8-byte ID handle created during the CF C2 command: %000012 in this example.
AC 02 40 00 00 00 25 30 30 30 30 31 32	Cues up the new ID for recording with a start time code of 00:00:00:00 in drop-frame NTSC timecode.
61 20 0E	Checks the channel's status until the cued bit is set.
44 15 40 15 00 00	Sets the out point (and therefore eventual duration) of the clip about to be recorded to 00:00:15:00 in the NTSC drop-frame timecode format. This command can be sent before or after the recording has actually started.
61 20 0E	Checks the channel's status to verify the clip's preset In and Out Points are set.
20 02	Puts the channel into record.

It's a good idea after the recording has started to keep verifying both the channel status using 61 20 and the clip's status to ensure that its duration continues to grow until the expected out point, if any. The server will provide metadata updates about the clip as often as every 2.5 seconds.

Special Protocol Information

This next section details some of the special features and more complex protocol implementations that a controller should be prepared to support. These include the Nexio Unicode solution, how to interpret the special video format code used to identify clip properties and the channels that can play them, the ability to route individual audio tracks within clips, and a new special playlist control method that supports clips as short in duration as 10 frames.

Unicode Implementation

The Nexio server system is capable of handling clip IDs and other clip metadata information in Unicode. This is a 2-byte per character system of entering strings of data in a system so that it is more compatible with languages from around the world. The minimum requirement is that the Nexio system be configured for 64-byte IDs (32 Unicode) and the extended fields feature, which is the mandatory set up for all Nexio servers shipping today.

By default, the Nexio servers do *not* interact with controllers using Unicode. Instead the server treats the Unicode IDs and metadata fields capable of Unicode as though they are ANSI characters. This preserves legacy operation for those controllers that have not made the transition to Unicode. This is particularly true of VDCP controllers because the VDCP protocol does not specify support for Unicode.

In order for a controller to actually work with the Unicode characters, it must first notify the LLM that all future communications will enable the Unicode feature. This is done by setting the Unicode mask in the **Set Machine Control Options (C8 B1)** command. When enabled, the following text fields will be treated as Unicode.

Table 2-14: Unicode-Supported Fields

Field	Storage Size	Max Characters
ID	64 Bytes	32 Characters
User Name	64 Bytes	32 Characters
Department	64 Bytes	32 Characters
Link	64 Bytes	32 Characters
Description	240 Bytes	120 Characters
Agency	30 Bytes	15 Characters
User-definable Field #1	50 Bytes	25 Characters
User-definable Field #2	50 Bytes	25 Characters
User-definable Field #3	50 Bytes	25 Characters
User-definable Field #4	50 Bytes	25 Characters

As you will see, only the ID and some of the extended fields support Unicode. The 8-byte ID handle and the normal set of metadata fields do not support Unicode.

It is important for controllers that make multiple connections to the LLM that they set the same Unicode mask on every connection to the server. The danger is if there is a mix of controllers, one using Unicode characters from the non-Basic Latin group and one using the ANSI character set. There is no good way for the server to translate complex Unicode characters to ANSI. The data retrieved by the ANSI controller will appear to be corrupt.

Underneath the hood, the LLM translates the data between Unicode and ANSI by simply removing every other byte of data when going to ANSI and padding every byte with a 0x00 when going to Unicode. The 0x00 byte is effectively the byte that represents the Basic Latin Unicode character set. As a result, as long as a Unicode controller only stores metadata in the Basic Latin set, it is possible for a non-Unicode controller to operate on the same Nexio server system without corruption of data.

Video Format Codes

With the latest Nexio servers supporting so many different video resolutions, formats, standards, and more, it is very important that controllers know which channels record what type of clips and what clips can be played out of which channels. The LLM is able to manage these different channel configurations and clip properties by the use of “Video Format Codes”.

Several protocol commands make use of these codes for getting and setting these properties. As a result, it is imperative that controllers learn how to parse this data so as to better prevent the wrong clip being loaded to the wrong channel.

The format code is represented by the following equation:

```
formatCode = (verticalSizeCode  * SCAN_TYPES + scanType)
                  * FRAME_RATE_CODES + frameRateCode
```

The values in this equation are represented in the enum statements below:

```
enum VERTICAL_SIZE_CODE
{
      VERTICAL_SIZE_CODE_480=0,
      VERTICAL_SIZE_CODE_512,
      VERTICAL_SIZE_CODE_576,
      VERTICAL_SIZE_CODE_608,
      VERTICAL_SIZE_CODE_720,
      VERTICAL_SIZE_CODE_1080,
      VERTICAL_SIZE_CODES
};

enum SCAN_TYPE
{
      SCAN_TYPE_INTERLACED=0,
      SCAN_TYPE_PROGRESSIVE=1,
      SCAN_TYPES
};

enum FRAME_RATE_CODE
{
      FRAME_RATE_CODE_24M=0,
      FRAME_RATE_CODE_24,
      FRAME_RATE_CODE_25,
      FRAME_RATE_CODE_30M,
      FRAME_RATE_CODE_30,
      FRAME_RATE_CODE_50,
      FRAME_RATE_CODE_60M,
      FRAME_RATE_CODE_60,
      FRAME_RATE_CODES
};
```

While the above format code provides up to 96 different combinations, in real use of a Nexio system, most controllers will encounter only a few different format codes in a single system. The most typical format codes encountered are identified in the following table.

Table 2-15: Format Codes

Format Code	Video Standard	Video Resolution	Frame Rate
0x03 (3)	NTSC	480i SD	30 drop frame
0x22 (34)	PAL	576i SD	25
0x4D (77)	PAL	720p HD	50
0x4E (78)	NTSC	720p HD	60 drop frame
0x52 (82)	PAL	1080i HD	25
0x53 (83)	NTSC	1080i HD	30 drop frame
0x5D (93)	PAL	1080p HD	50
0x5E (94)	NTSC	1080p HD	60 drop frame

As an example using the equation from above, consider a 720p PAL clip:

```
formatCode = (verticalSizeCode  * SCAN_TYPES + scanType)
                  * FRAME_RATE_CODES + frameRateCode

formatCode = (4 * 2 + 1) * 8 + 5

formatCode = (8 + 1) * 8 + 5 = 9 * 8 + 5 = 72 + 5 = 70
```

Even though the format code Enum allows for extended height scan lines for SD, VERTICAL_SIZE_CODE_512 and VERTICAL_SIZE_CODE_608, the server does not really take these variables into account when calculating the format code. Controllers should treat any clips or channels configured for extended height capabilities in SD using the normal SD scan line values.

The following protocol commands make use of the Video Format Code

- C4 B1: Get Special Machine Properties, bit masks 0x0040, 0x0080, and 0x0100
- CF B1: Set Special Channel Properties, bit mask 0x0100
- C9 C3: Get Extended Field, Field 4

In the case of **Get Special Machine Properties**, bit mask 0x100, and **Set Special Machine Properties**, bit mask 0x100, an additional bit of information is included. If the record video format requires storing the pixels in the 4:3 aspect ratio, such as with XDCAM HD, the last bit is set high. For example, instead of a value of 0x53 (83 decimal) for NTSC 1080i, the value would be 0xD3 (211). Controllers would set or unset this variable mainly when changing the record parameters to or from support for XDCAM HD.

Meanwhile, the Extended Field commands go even further with additional bits for field number 4. These commands embed the Video Format Code into the first 96 bits of the 128-bit field. The rest of the bits allows the server to store several more properties for each individual clips. The additional bits are defined in the **Get Extended Field** command.

Audio Track Router

Nexio platforms include support for audio track routing, the dynamic routing of audio tracks on play out. The typical use of this feature is to manage the airing of different language tracks when played out of different server channels. The idea is that a single clip would have multiple languages recorded across its many possible audio tracks. And based on properties within the clip itself and the channels from which they air, the server will dynamically know which audio tracks play on which audio track channels.

This feature was introduced with the Nexio 6.0 Software Release and continues to be enhanced with additional functions as of the Nexio 7.0 Software Release. The information below regarding setting up Audio Routing Profiles and dynamically assigning them to IDs cued in channels was introduced with the 7.0 release.

What follows is the basic workflow items a controller might have to enable within protocol in order to make the audio track routing feature work.

1. Set up audio tags
2. Manage clips to ensure their audio tags and other properties are correct
3. Set up channel input mask routing for recordings
4. Set up channel output mask routing for play out
5. Set up audio profiles for play out manipulation
6. Apply audio profiles to cued or playing clips to override default settings

The Nexio Audio Track Router interface also supports the FTP import and export of discreet and paired audio track files. This information is covered in the FTP User Guide and will not be covered here, but many of the concepts of clip properties associated with imported clips are the same.

There are two sets of information to keep track of with the audio track routing logic: audio types and audio tags. The audio types define the types of audio stored on each audio track including, if compressed, the number of audio tracks stored in the compressed data. The audio tags are a simple way of managing the audio tracks and grouping them together so as to better plan what audio tracks get routed on play out of a specific server channel.

The audio types are defined on a per audio track basis according to the following 8-bit breakdown.

Table 2-16: Audio Track Types

Bit	Description
0	Indicates a track containing compressed audio which requires being stored within a pair of tracks (Neural/DTS, Dolby Digital/AC3 and Dolby-E)
1 – 3	Indicates the number of actual audio tracks stored within the compressed audio track. This information is mandatory for all compressed audio. PCM = 0
4 – 7	Indicates the audio format of the track: 0 = PCM 1 = Neural/DTS 2 = Dolby Digital/AC3 3 = Dolby E 4 = MPEG-1 Layer I and Layer II 5 = reserved 6 = AAC/HE-AAC/HE-AAC v2

What follows are some basic examples in Hex (and bits) of possible audio track types:

- 0x00 (00000000) = a typical uncompressed audio track, usually identified as PCM, assigned to one audio track
- 0x3D (00111101) = Dolby-E compressed from 5.1 audio surround sound (6 tracks) and typically stored on the server as a linked pair of audio tracks
- 0x44 (01000100) = MPEG audio compressed from a pair of tracks and typically stored on only one of the server's audio tracks

Audio track tags are additional descriptors for audio tracks which allow an operator to identify and separate audio tracks according to language or any other definition. The most basic operation is to assign one track or pair of tracks to one language and another track or tracks to another language.

The server allows for up to 256 defined audio tags that can be applied to a specific audio track. Of these, 34 tags are pre-set and can only be set for pre-determined purposes. Another 64 are pre-defined by Nexio controlling applications (NXOS, FTP Client, and Velocity) according to their assigned languages and audio types.

Table 2-17: Audio Track Tags

Tag	Description
0	The default tag for all tracks. Indicates full pass-through with no need to dynamically route the track to which this track is assigned.
1 – 32	Tags set aside for each possible discreet audio track. These are the LXF audio tracks as stored internally by the server.
33 – 175	Tag numbers available to be defined
176 – 191	Tags preset for PCM audio and assigned to some of the more common specific languages
192 – 207	Tags preset for Dolby-E compressed audio and assigned to some of the more common specific languages
208 – 223	Tags preset for Dolby Digital/AC3 compressed audio and assigned to some of the more common specific languages
224 – 239	Tags preset for MPEG Layer I/II compressed audio and assigned to some of the more common specific languages
240 – 254	Tag numbers available to be defined
255	A tag to indicate a track to be made mute

Set Up Audio Tags: The exact list of tags as programmed into the server can be found in the NXOS interface using the menu option for View\Audio Tag Definition. There you will see each audio tag defined according to tag number, ISO code to indicate the language, the defined audio type, and a description. Remember that the tag number displayed in NXOS is in Decimal whereas the protocol commands used to get or set the tag is in Hex.

It is also important not to change the defined tags once they are set up in use. For example, if you set up audio tag number 101 to be PCM audio in the Tibetan language and later change it to be MPEG 1 compressed audio in Turkish, any IDs already associated with that tag number as Tibetan PCM will be mislabeled after the change.

Manage Clips: With the information about Audio Types and Audio Tags, a controller can read or set the audio properties for every audio track inside a clip. This is done using the **Get Extended Field (C9 C3)** and **Set Extended Field (CF CC)** commands. Extended field 23 represents a clip's audio types. Extended Field 24 represents a clip's audio tags. For each field, the controller will get or set 32 bytes of data, one byte for each possible audio track that can be stored with a clip.

Even though a clip may have only been recorded with 4 or 8 audio tracks, even though the maximum number of audio tracks that can be recorded on any Nexio server channel is 16 tracks, and even thought the maximum number of audio tracks that can play out of any Nexio server channel is 16 tracks, the audio track router interface supports up to 32 tracks. This is to provide ultimate flexibility for storing multiple languages or multiple audio types with a single clip and then picking and choosing which of those audio tracks to route out of the server at play back out time.

The controller must remember to pair up any compressed audio tracks that require being stored on 2 tracks (Dolby-E, Dolby Digital, and Neural). For example, a clip with Dolby-E stored on a clip's first and second tracks and its seventh and eighth tracks, with PCM audio or nothing on all the rest of the tracks would have a series of 32 bytes in extended field 23 as follows:

```
3D 3D 00 00 00 00 3D 3D 00 00 00 00 00 00 00 00 … and so on
```

It's important to note that such pairs of audio tracks always need to start on the odd-numbered track (if operating in a 0-based numbering system such as the LLM, this would be considered the even numbered track).

Using the same example and the audio tag table available in NXOS, we might then label the first set of Dolby-E audio as being in Portuguese and the second set in Spanish, while audio tracks three and four are PCM audio in Portuguese and tracks five and six are PCM in Spanish. The resulting 32 bytes stored in extended field 24 would be displayed as follows:

```
C3 C3 B3 B3 B1 B1 C1 C1 00 00 00 00 00 00 00 00 … and so on
```

For example, the first audio track pair is set to C3. C3 in Hex is 195 in Decimal. As seen in the NXOS Audio Tag Definition table, tag 195 is Dolby-E Portuguese, which is now assigned to audio tracks 1 and 2.

The reference to C3 appears on both tracks 1 and 2 because Dolby-E audio is stored in and plays out of server on a pair of tracks. Therefore any audio tag that is paired, even uncompressed stereo audio, would appear twice in the audio tag string.

However, it is important to not set the same audio tag to separated tracks in extended field 24. Using the above example, if both sets of Dolby-E tracks were in Portuguese, you cannot set the audio associated with tracks 1-2 and 7-8 to the same tag. If you did, the audio on tracks 7-8 would be muted. Instead, you should set up a new audio tag number for the second set of audio and apply it to tracks 7-8.

Any assigned track with no actual audio should be set to nulls for both fields 23 and 24.

Set Up Channel Input: In addition to setting the properties on an individual clip basis, if the audio input to specific channels is always the same source type, the controller can prepare the channel so that all clips recorded on that channel are automatically assigned the correct values in extended field 24. In this way, you don't need to modify every clip's metadata to assign the appropriate audio tags.

This is done using the **Set Auto-Detect Record Parameters (CF CF)** command for bit mask 18 when a Nexio server makes use of the auto-input resolution detection feature (enabled by default with the Nexio 7.0 Software Release). For Nexios which have not yet enabled auto-input detection, the command to use is **Set Audio Routing Parameters (CF 55)**.

The format of the bytes used in these commands is the same as defined for extended field 24. Basically it's 32 bytes of data matching up an audio tag for every possible existing and virtual audio track.

Keep in mind if a value that is set up to get written to a clip is wrong that wrong value will get stored with the clip. It is up to the controller to correct the bad data or risk having a possible corruption in the play out of the audio.

Set Up Channel Output: Once the controller has prepared all the clips in the Nexio system with the appropriate audio properties, it is now possible to dynamically route the audio stored within clips out on different tracks than what they were actually stored on.

The typical workflow is that one server channel might be set up for playing to air audio tracks from one particular language and another server channel might be set up to play a different language. And because not every clip will have the exact same arrangement of audio tracks and tags, additional

flexibility has been added such that if one type of audio is not available for play out, it is possible to select alternate audio track types as backups.

In all, there are three levels of audio router planning: a primary, secondary, and tertiary setting. If a clip has no matching audio tags in the play out channel's primary routing table, the secondary routing table will be used. If there are no matches there, the tertiary table is used. If no matches are found there, then the server passes through the audio in the tracks as the clip was recorded.

The output audio routing is set up using the **Set Audio Routing Parameters (CF 55)** command, which prepares the primary, secondary and tertiary track routing table, respectively. Each table is made of 32 bytes, 2 bytes per audio track output. The first byte represents the audio tag for the individual track. The second byte is currently reserved for future use.

So if the desire is to prepare a channel to play the first two tracks as the Dolby-E Spanish language track and the next two as the PCM Spanish language stereo pair, and as a backup to play the Dolby-E Portuguese and PCM Portuguese tracks, and as a backup to the backup to play the Dolby-E English and PCM English tracks, then the controller would send the following commands in sequence:

```
CF 55 21 02 C1 00 C1 00 B1 00 B1 00 00 00 … and so on
CF 55 21 03 C3 00 C3 00 B3 00 B3 00 00 00 … and so on
CF 55 21 04 C0 00 C0 00 B0 00 B0 00 00 00 … and so on
```

Then when the clip referenced earlier is loaded in this channel, its seventh and eight tracks (tag = C1), which contain the Dolby-E Spanish, would be re-routed to play out of audio tracks one and two. The PCM Spanish found on tracks five and six would be re-routed to tracks three and four (see table below). The Portuguese tracks on that clip would not be routed anywhere.

Table 2-18: Sample Track Routing

Track Number	Source Audio	Primary Mask	Secondary Mask	Tertiary Mask	Final Result
1	C3	C1	C3	C0	C1 (7)
2	C3	C1	C3	C0	C1 (8)
3	B3	B1	B3	B0	B1 (5)
4	B3	B1	B3	B0	B1 (6)
5	B1	00	00	00	00
6	B1	00	00	00	00
7	C1	00	00	00	00
8	C1	00	00	00	00

If instead, the clip had no defined Spanish tracks, all four of its Portuguese tracks would have played out of the channel's first four tracks, based on the secondary routing table created above. And if none of the clip's tracks had any tags defined, the routing tables simply get ignored and the audio tracks as stored with the clip will play out the tracks as they were originally recorded.

The pass-through setting is always the default unless a channel's audio tracks are set otherwise.

Set Up Audio Profiles: As an extra level of flexibility in managing the routing of audio tracks, controllers can also set up predefined audio profiles which can be associated with individual clips as they are loaded in a channel for play out.

The main purpose of audio profiles is to support the playback of the same ID multiple times per day but with each playback following a different set of audio routing masks.

Up to 32 different audio profiles can be set up on a per server basis using the **Set Audio Routing Profile (CF 57)** command. Each profile would include settings for each of the primary, secondary, and tertiary output masks available.

Apply Audio Profiles: Audio profiles are applied to an individual ID as it is loaded in a channel for play back. Any selected audio profile overrides the channel's current output audio routing settings for the duration of the clip's playback. To continue applying audio profiles, the controller must override every cued (or re-cued) clip at cue or play time.

The **Select Preset Audio Routing Profile (C1 16)** command is used to affect the currently cued or playing ID in a channel. The **Select Preview Audio Routing Profile (C1 17)** command is used to affect the currently stacked clip waiting to play. These commands work in conjunction with the **Cue Up with Data (2x 31** and **4x 14)** and **Preset Preview In Point (Ax 04)** commands.

The command work flow is as follows:

7. Cue Up with Data (2x 31 or 4x 14) to load the first ID
8. Get Channel Status (61 20) until the CUED status bit is set
9. Select Preset Audio Routing Profile (C1 16)
10. Get Channel Status (61 20) until the CUED status bit is set again
11. Play (20 01) ID
12. Preset Preview In Point (Ax 04) to stack next ID
13. Get Channel Status (61 20) until the PVW RDY status bit is set
14. Select Preview Audio Routing Profile (C1 17)
15. Get Channel Status (61 20) until the PVW RDY status bit is set again

The selected audio profiles only apply as long as the clip is in the channel. If the clip is re-cued, the audio profile needs to be reapplied.

Short Clips Stacked Play Back

This section describes a feature introduced in the Nexio 5.7.1 Software Release. It provides new logic to allow ultra-short duration clips to play back to back out of a single Nexio channel.

During normal stacked play back out of a Nexio channel, it is strongly recommended to use clips no shorter than three seconds. Because of the time it takes to retrieve the data from storage and prepare a channel to stack the next clip so that it's ready to play, it's been determined that the only reliable mechanism for playing out a list of clips back-to-back is to limit just how short a clip can be in such a playlist. Any controller attempting shorter duration clips in this stacked play back mode will likely have undesirable play out results.

To be sure, any size clip can be loaded in a channel and played out by itself. Similarly any clip stored as a Media ID is capable of playing out a series of short cuts which are really just pieces of media from other clips. However, like ultra-short duration clips, multiple Media IDs are not supported in stacked play back mode.

To work around this, a new set of protocol commands was created to allow a controller to generate a dynamic stacked list of clips, short or long, that the Nexio server can successfully play to air. Using the **Local Timeline Control (CF 88)** command, a controller builds and maintains the list of clips while giving the Nexio server enough time to plan ahead so that it can retrieve the media's data from storage and load it into the channel for play out.

This has been successfully tested down to a list of 10-frame clips. However it must be understood that this is not the same as stacked play back mode. The standard cue up and stack commands are not used here. This functionality is not compatible with those commands. Up front, a controller must decide if it is going to control a channel in normal stacked play back mode with its known time limitations or if it needs the flexibility to generate and manage the list of clips dynamically through the server.

The entire definition for the **CF 88** command is explained later in the list of protocol commands. What follows is a basic workflow for how a controller might use the local timeline control.

Table 2-19: Local Timeline Workflow

Protocol	Command	Description
40 40	Set Auto Mode Off	Disengages local timeline control (in case a timeline was already loaded)
20 00	Stop	Unloads anything that might be loaded in the channel
40 41	Set Auto Mode On	Engages local timeline control
CF 88…	Local Timeline Control	A series of these commands are sent to set up the list of clips to be played in order
20 30	Preroll	This performs the function of cueing the timeline. Do not use the normal cue up commands
20 01	Play	Plays the timeline
CF 88…	Local Timeline Control	During playback, monitor status of the timeline. It is also possible to add more clips to or remove clips from the timeline
20 00	Stop	Send this command when the timeline is complete or when you want to stop the playlist
40 40	Set Auto Mode Off	Disengage timeline control to turn off the feature

The CF 88 command has multiple functions for building and managing the local timeline. The controller has the option of building the list using the clips' regular ID or its 8-byte ID handle. They can add and remove individual elements in the list based on the element's position number. They can even do this adding and removing of clips while the playlist is on the air. However it is important that any clip within a 3-second window of the current position of the timeline not be disturbed. Neither should the clips in that window be deleted nor have anything inserted ahead of them.

The structure of the CF 88 command includes a series of parameters: a byte count, a sub-code, a data bitmap, and finally the actual data. The sub-code indicates the actual function being performed by the command, whether that be adding or deleting an item in the list, requesting the current position of the timeline, or making sure what's in the list.

Depending on the sub-code selected, the data bitmap might be nothing, a single entry, or a series of variables. Adding clips to the timeline makes use of most of these variables. What follows is an example of a timeline built from scratch using sub-code 00 and three clip IDs called "abc", "def", and "ghi".

```
CF 88 0F 00 8E 00 00 40 00 01 00 40 30 00 00 61 62 63
CF 88 0F 00 8E 00 01 40 00 01 00 40 30 00 00 64 65 66
CF 88 0F 00 8E 00 02 40 00 01 00 40 30 00 00 67 68 69
```

In this example, all three clips are prepped to cue to the 00:01:00;00 mark and play for 30 seconds before automatically transitioning to the next. **0F** (16 decimal) represents the number of bytes to follow in the command. The following **00** indicates sub-code 0. **8E** represents the data bit mask, which

represents new entry ID (0x02), starting timecode value (0x04), duration in timecode value (0x08), and extended ID name (0x80). The server will accept up to 256 entries in the timeline at any given time.

After then sending the **Preroll (20 30)** command and then the **Play (20 01)** command, the controller should continue to monitor the status of the timeline. Sub-code 3 (CF 88 02 03 00) provides this information, indicating both which element in the timeline is currently playing and at what timecode (based on the entry's starting timecode value not the actual timecode value itself). Here is an example of a series of the server responses to status queries separated by 10 frames each:

```
DF 88 08 00 09 00 00 10 00 00 00
DF 88 08 00 09 00 00 20 00 00 00
DF 88 08 00 09 00 00 00 01 00 00
```

In this case, the first clip in the timeline is playing and the timecode is incrementing as normal. **08** represents the number of bytes to follow in the command. **00** represents an error code of success (no error). **09** indicates the data bit mask for entry ID (0x01) and duration in timecode value (0x08). Most likely a controller would send the status queries as frequently as one per frame just to be sure the timeline is playing as expected.

At this point the controller might want to modify an item in the list, maybe insert something up higher in the list. Again, as long as the affected item is not within three seconds of air, it should be safe to modify or delete.

For example, the controller might decide to change the second item in the list so that it starts playing 15 seconds later and only plays for 5 seconds.

```
CF 88 0C 01 0D 00 01 40 15 01 00 40 05 00 00
```

01 is the sub-code for modifying an existing entry. **0D** is the data bit mask for entry ID (0x01), starting timecode value (0x04), and duration in timecode value (0x08).

It's important for any controller playing clips to air using this Local Timeline Control set of commands to keep a close watch on the activity of the channels they control. The timing of changes is critical and it is strictly under the control of the controller.

The controller should also not expect full transport control of the timeline while in this local timeline control mode. It is intended as a play out only control. However there are two possible configurations for this feature. In the default configuration, only the following transport commands are allowed in local timeline mode: Play, Shuttle Forward in the linear speed range, Stop, and Eject. There is no reverse play back. There is no ability to cue to a timecode position within the timeline. And once a timeline is fully played out, the timeline list is auto-deleted. Any subsequent **Get First Entry ID** sub-code commands sent will only return the currently playing or loaded entry.

In the more advanced mode, it is possible to use reverse play back commands, it is possible to cue to an absolute timecode position within the timeline, and the events in the timeline are not auto-deleted when the end of the timeline is reached. To set this advanced mode, the controller should use the **Set Channel Properties (CF B1)** command for mask 0x00040000 and set the channel to a value of 1.

Standard Server Responses

When a controller directs a protocol command to the LLM that does not specify a return of data, the LLM will return an acknowledgement that it received the command or an error to indicate that it could not carry out the instruction.

The format of these standard responses by the LLM to the controllers is defined below:

```
10 01                                                    ACK
```

Syntax: **10 01**

Description: When receiving a command from the controller requiring no additional data to be returned, the server will send back this response as an acknowledgement.

It is up to the controller to ensure that it sends validly formatted commands to the LLM. Any invalid commands will return the ACK response.

```
11 12                                                    NAK
```

Syntax: **11 12 ERROR**
ERROR: 1-byte unsigned integer, as indicated in the table below

Description: When a transmission error occurs, the server will send back this response with an ERROR bitmap as defined below:

Table 2-20: Error Bitmap

BIT7	BIT6	BIT5	BIT4	BIT3	BIT2	BIT1	BIT0
Timeout error	Framing error	Overrun error	Parity error	---	Checksum error	---	Undefined command

System Information Commands

00 11 Device Type Request

Syntax: **00 11**

Response: **12 11 CODE**
CODE: 2-byte unsigned integer, MSB first, as seen in table below

Description: This is a legacy command that is more relevant to the old VR and original Nexio server platforms. It is recommended that controllers retrieve the below information through other protocol commands for current shipping products.

If sent to the server, however, it will return one of the following codes:

Table 2-21: Server Types

Code	Description
0xAA11	2-channel server, FILM
0xAA12	2-channel server, NTSC
0xAA13	2-channel server, PAL
0xAA15	2-channel server, FILM, VBI
0xAA16	2-channel server, NTSC, VBI
0xAA17	2-channel server, PAL, VBI
0xAA31	4-channel server, FILM
0xAA32	4-channel server, NTSC
0xAA33	4-channel server, PAL
0xAA35	4-channel server, FILM, VBI
0xAA36	4-channel server, NTSC, VBI
0xAA37	4-channel server, PAL, VBI

The information regarding number of channels returned only indicates the number of video channels capable of playing back material. Nexio servers can include one or two “Record-Only” channels in addition to the number indicated in the command response.

To accurately determine the total number of channels in a Nexio AMP, Volt, or XS server, a controller should instead use the **Set Logical Channel (C1 AC)** command to attempt to control the highest number channel possible. If the server does not have that number of channels, the response to the **Get Logical Channel (C0 AC)** command will indicate the actual highest channel number.

Example: As of this writing, the maximum number of channels possible is eight. The controller should first try to change the current channel under control to channel eight.

```
C1 AC 07
```

The LLM will set the current channel under control to the highest numbered channel available. To learn what channel that is, the controller can then ask the LLM what channel it just changed to.

```
C0 AC
```


In response, the LLM will return the actual number of channels.

```
D1 AC 01    //2-channel server
D1 AC 02    //3-channel server
D1 AC 03    //4-channel server
D1 AC 05    //6-channel server
D1 AC 07    //8-channel server
```

Also note that as a result of this command sequence, the controller will now be controlling the returned channel number. All future commands will continue to apply against this channel until another **Set Logical Channel (C1 AC)** command is sent.

C4 B1 — Get Special Machine Properties

Syntax: **C4 B1 MASK**
MASK: 4-byte unsigned integer, MSB first

Response: **DF B1 BC DATA** or
D0 B1 if no valid properties are requested
BC: 1-byte unsigned integer, indicating total amount of data to follow
DATA: String or bitmap depending on **MASK** selected

Description: This command allows a controller to retrieve a server's special machine or channel properties. Depending on the bit mask selected, a controller can determine several different properties as detailed in the table below.

Please note that some of the properties below are specific to the VR and the original Nexio servers and do not apply to the current line of server products. Others are specific to the Nexio AMP, Volt, and XS servers, as noted.

Table 2-22: Special Machine Properties

Bit	Mask	DATA Response in Bytes	Description	Software Release Introduced
0	0x00000001	1	Determines whether hidden IDs are included in media database queries. If response is non-0, the hidden IDs are included in the ID lists.	All servers
1	0x00000002	Variable	Provides the LLM Version returned as an ASCII string not including the zero termination	All servers
2	0x00000004	1	Identifies the video card or cards installed in the server.	Nexio 6.5 Software Release
3	0x00000008	1	Indicates if Timeline Service is active. If the response is non-0, the timeline service is currently connected to the channel and Media ID type clips are available for play out (but not for stacked play out)	All servers
4	0x00000010		Not supported	

5	0x00000020	1	Indicates if the connected controller is making use of Unicode text for clip IDs and their extended metadata. If the server returns 0, the server accepts protocol commands with the Extended IDs and fields in ANSI format and converts them into Unicode for storage on the disk. If 1, the server accepts the Extended IDs and fields in Unicode format.	All servers
6	0x00000040	1	Indicates the input video format code for the selected channel. 0xFF indicates a play-only channel. See notes below.	All servers
7	0x00000080	1	Indicates the output video format code for the selected channel. 0xFF indicates a record-only channel. See notes below.	All servers
8	0x00000100	1	Indicates the record video format code and HD pixel aspect ratio for the selected channel. 0xFF indicates a play-only channel. See notes below.	All servers
9	0x00000200	14	Indicates the default output Video ARC assigned to a channel. Response is the 14-byte USER_ARC structure as defined in the **C9 C3** command for field 19.	All servers
10	0x00000400	1	Indicates the output aspect ratio for SD channels or 0xFF for record only channels. 0 = 4:3; 1= 16:9. HD channels are always 16:9.	All servers
11	0x00000800	2	Indicates if the channel is configured for insertion of timecode on output. See table in **CF B1** command for definition of bits. MSB first.	Nexio 4.5 Software Release
12	0x00001000	1	Indicates the channel's setting for what to do at the end of a playing and unstacked clip: 0 = last frame freeze (legacy), 1 = auto-blank (black output), 2 = auto-blank on all stops including initial load	Nexio 4.5 Software Release. Option 2 as of Nexio 6.0 Software Release
13	0x00002000	1	Identifies the current LLM's node number	Nexio 4.5 Software Release
14	0x00004000	1	Indicates the number of CPU processors in the Nexio computer	Nexio 5.0 Software Release

15	0x00008000	4	Returns an output channel's maximum video resolution for support of up-, down-, and cross-conversion: 0 = SD-only media (no HD allowed), 1 = 1080i or SD media allowed, 2 = 1080i or 720p or SD media allowed, 3 = all of the above plus 1080p, >3 all resolutions except 1080p.	Nexio 5.0 Software Release
16	0x00010000	4	Indicates if an output channel has E-E routed to it from one of the input channels: 0 = E-E from input channel 1, 1 = E-E from input channel 2, etc. until maximum input channel. FFFFFFFF = invalid value	Nexio 5.0 Software Release
17	0x00020000	Variable	Byte 1: Byte Count of data to follow Byte 2 + : A series of bytes indicating the video resolution and video formats the selected channel is capable of loading. See notes below.	Nexio 5.7.0 Software Release
18	0x00040000	1	Identifies the current local timeline control mode to indicate if the local timeline control includes extra features. See **CF 88** command for more details.	Nexio 5.7.1 Software Release
19	0x00080000	32	Audio routing input mask for extended field 23. See **Get Extended Field (C9 C3)** command, field 23, for syntax	Nexio 6.0 Software Release
20	0x00100000	32	Audio routing input mask for extended field 24. See **Get Extended Field (C9 C3)** command, field 24, for syntax. As of Nexio 7.0, if auto-input resolution detection is enabled, this MASK will not function correctly. Instead use the **Set Auto-Detect Record Parameters (CF CF)** command.	Nexio 6.0 Software Release
21	0x00200000	32	Audio routing output primary mask. See notes below.	Nexio 6.0 Software Release
22	0x00400000	32	Audio routing output secondary mask. See notes below.	Nexio 6.0 Software Release
23	0x00800000	32	Audio routing output tertiary mask. See notes below.	Nexio 6.0 Software Release

24	0x01000000	16	Audio routing output audio type. One byte per output track indicating the audio type being output. See **Get Extended Field (C9 C3)** command, field 23, for syntax.	Not currently implemented as of Nexio 6.5.1 Software Release
25	0x02000000	1	Indicates if the channel is encoding or decoding video using 10-bits-per-component. Possible values are 8 (0x08) or 10 (0x0A) to indicate the number of bits per component.	Nexio 6.0 Software Release
26	0x04000000	1	Indicates the video standard that the server is configured for. NTSC = 0, PAL = 1, FILM (23.98 frame rate) = 2, FILM (24 frame rate) = 3.	Nexio 6.5 Software Release
27	0x08000000	14	Indicates the channel's current configuration for the output of AFD data based on the channel's aspect ratio. Data returned is in the 14-byte Video ARC structure. See **Get Extended Field (C9 C3)** command, field 19, for syntax.	Nexio 7.0 Software Release
28	0x10000000	14	Indicates the channel's default setting for handling AFD data on output when a clip of the same aspect ratio as the channel is loaded for play back. See **Get Extended Field (C9 C3)** command, field 19, for syntax.	Nexio 7.0 Software Release
29	0x20000000	14	Indicates the channel's *current* setting for handling AFD data on output when a clip of the same aspect ratio as the channel is loaded for play back. See **Get Extended Field (C9 C3)** command, field 19, for syntax.	Nexio 7.0 Software Release

Bits 21 - 23 Note: The 32 bytes returned for Bit Masks 0x200000, 0x400000, and 0x800000 represent the audio track routing configuration in place for each channel. They provide a tri-level hierarchy for routing audio tracks based on specified audio track tags.

The 32 bytes represent the routing for up to 16 output audio tracks with two bytes assigned per track. The first byte of each represents an audio track tag. For now, the second byte of each is reserved for future use and should be set to 0x00.

Bit 17 Note: The series of bytes returned for Bit Mask 0x020000 represent a concatenated list of all the video formats and video resolutions supported on a specific channel. The first byte returned is a byte count identifying the number of bytes to follow. A return value of 0 indicates the feature is not currently enabled or implemented. The remaining bytes are a series of one byte values indicating which formats and

resolutions are supported. The format and resolution indicators are split into the two nibbles of each byte.

The LSB nibble is used to identify one and only one video format supported. If a server channel supports multiple formats, there will be one bye of data per supported format.

Table 2-23: Video Format Nibble

Data	Video Format
0xn0	Default: Assume all files may be played on any port
0xn1	JPEG
0xn2	MPEG2 4:2:0
0xn3	MPEG2 4:2:2
0xn4	DV, all types
0xn5	JPEG2000
0xn6	DNxHD
0xn7	H.263, MPEG4 Part 2 (Long GOP, ASF)
0xn8	H.264, MPEG4 Part 10 (Long GOP, AVC)
0xn9	AVC-Intra, MPEG4 Part 10 (I-Frame, AVC)
0xnA	Apple Pro-Res
0XnB	XDCAM HD (1440x1080)
0xnC	REDCODE
0xnD	Reserved
0xnE	Reserved
0xnF	Reserved

The MSB nibble is used to identify all of the video resolutions supported in that channel. Channels configured for the HD/Mixed option are capable of up-, down-, or cross-converting clips from different video resolutions. As a result, for each resolution supported in the channel, the data is added together in the MSB nibble.

Table 2-24: Video Resolution Nibble

Data	Video Resolution
0x0n	Default: Assume all files may be played on any port
0x1n	SD
0x2n	720p
0x4n	1080i
0x8n	1080p

For example, all current Nexio channels on a server configured for 6 channels of SD would return the following information:

```
03 12 13 14
```

In this example, there are 3 bytes of data to follow, each of which indicates the channel is capable of loading only SD clips which can be MPEG2 4:2:0, MPEG2 4:2:2, or DV.

Meanwhile, a typical 3-channel server configured for 1-in/2-out based on a 1080i output might have the following for each channel:

```
04 72 73 74 6B
```


In this example, there are 4 bytes of data to follow, the first three of which indicate the channel is capable of loading SD, 720p, or 1080i clips which can be MPEG2 4:2:0, MPEG2 4:2:2, or DV and the last which indicates XDCAM HD clips are supported in 720p and 1080i.

This channel property only functions as of the Nexio 5.7.0 Software Release and later.

Bits 6 - 8 Note: The values returned for Bit Mask 0x0040, 0x0080, and 0x0100 are based on the video format code formula:

```
formatCode = (verticalSizeCode  * SCAN_TYPES + scanType)
                * FRAME_RATE_CODES + frameRateCode
```

In addition, Bit Mask 0x0100 includes an extra bit to help identify an alternate pixel aspect ratio in the channel. All of these variables are defined with examples above in the **Video Format Codes** section.

A return value of 0xFF in any of these three properties indicates that the particular channel does not support the associated property.

Bit 2 Note: The one byte value returned in Bit Mask 0x0004 indicates the type of video card or cards in the server. Bit 7 works with the other bits to indicate the two cards in the platform, one of which is the accelerator card which supports AVC-Intra play back.

Table 2-25: Video Card Type

Bits	Card Type
0	None
1	MA200 (Volt and NX3601)
2	MA400 (NX3801)
3	MA410 (Volt 2 and Versio)
4 – 6	Reserved
7	Accelerator (NXCP)

C8 B1 — Set Special Machine Properties

Syntax: **C8 B1 MASK VALUE**
MASK: 4-byte unsigned integer, MSB first
VALUE: 4-byte unsigned integer, MSB first

Response: 10 01

Description: This command allows a controller to set a few of the server's special properties that are important in determining how the controller interacts with the server.

This command will assign the values from **VALUE** only to those bit fields that are set to 1 in **MASK**. The following table details two values that affect a controller's connection to the entire server.

Table 2-26: Set Special Machine Properties

Bit	Mask	Description
0	0x00000001	Prepares "promiscuous" mode for ID lists: If set to 1, all hidden IDs are included in the ID lists. This is usually only important for editor-based controllers.
1	0x00000002	Prepares the Unicode interchange with the server. If enabled, the server expects all Unicode-enabled fields to be read and written as Unicode strings in the protocol. If set to 0, the server expects Unicode-enabled fields to be read and written in the ANSI format. The LLM will convert the fields to Unicode for storage on the disk.

If a controller has several different control connections to the same server, it should set these values the same on every connection.

Example: To show all hidden media in the server media lists:

```
C8 B1 00 00 00 01 00 00 00 01
```

To exit promiscuous mode:

```
C8 B1 00 00 00 01 00 00 00 00
```

To enter Unicode mode send:

```
C8 B1 00 00 00 02 00 00 00 02
```

To exit Unicode mode send:

```
C8 B1 00 00 00 02 00 00 00 00
```

CF B1 — Set Special Channel Properties

Syntax: **CF B1 BC MASK PROPERTY**
BC: 1-byte unsigned integer, indicating total amount of data to follow
MASK: 4-byte unsigned integer, MSB first
PROPERTY: Data representing the new **PROPERTY** value for the indicated **MASK**

Response: **10 01**

Description: Only supported in the Nexio AMP, Volt, and XS platforms, this command allows a controller to set a special channel property that is crucial in preparing the correct video record format and resolution for each channel under control.

Table 2-27: Set Special Channel Properties

Bit	Mask	Property Size	Description	Server Platforms
8	0x00000100	1	Sets the record video format code and the pixel aspect ratio for the selected channel. See the section above on Video Format Codes and pixel aspect for syntax. An LLM restart is required for video format code changes. Changing the pixel aspect value does not require a restart.	Nexio AMP, Volt, and XS only
9	0x00000200	14	Sets the default output Video ARC assigned to a channel. The Property uses the 14-byte USER_ARC structure as defined in the **C9 C3** command for field 19.	Nexio AMP, Volt, and XS only
11	0x00000800	2	Configures the channel for insertion of timecode on output. See table below for definition of bits. MSB first.	Nexio AMP, Volt, and XS only as of Nexio 4.5 Software Release
12	0x00001000	1	Configures the channel for what to do at the end of a playing and unstacked clip: 0 = last frame freeze (legacy), 1 = auto-blank (black output), 2 = auto-blank on all stops including initial load	Nexio AMP, Volt, and XS only as of Nexio 4.5 Software Release. Option 2 as of Nexio 6.0 Software Release
18	0x00040000	1	Sets the current local timeline control mode to indicate if the local timeline control includes extra features. See **CF 88** command for more details.	Nexio AMP and Volt only as of Nexio 5.7.1 Software Release
19	0x00080000	32	Sets the channel's audio routing input mask for extended field 23. See **Get Extended Field (C9 C3)** command, field 23, for syntax and the Audio Track Router section above for details.	Nexio AMP and Volt only as of Nexio 6.0 Software Release

20	0x00100000	32	Sets the channel's audio routing input mask for extended field 24. See **Get Extended Field (C9 C3)** command, field 24, for syntax and the Audio Track Router section above for details. As of Nexio 7.0, if auto-input resolution detection is enabled, this MASK will not function correctly. Instead use the **Set Auto-Detect Record Parameters (CF CF)** command.	Nexio AMP and Volt only as of Nexio 6.0 Software Release
21	0x00200000	32	Sets the channel's audio routing output primary mask. See the Audio Track Router section above for details.	Nexio AMP and Volt only as of Nexio 6.0 Software Release
22	0x00400000	32	Sets the channel's audio routing output secondary mask. . See the Audio Track Router section above for details..	Nexio AMP and Volt only as of Nexio 6.0 Software Release
23	0x00800000	32	Sets the channel's audio routing output tertiary mask. See the Audio Track Router section above for details.	Nexio AMP and Volt only as of Nexio 6.0 Software Release
24	0x01000000	16	Sets the channel's audio routing output audio type. One byte per output track indicating the audio type being output. See **Get Extended Field (C9 C3)** command, field 23, for syntax.	Nexio AMP and Volt only as of Nexio 6.0 Software Release
25	0x02000000	14	Sets the channel's default setting for handling AFD data on output when a clip of the same aspect ratio as the channel is loaded for play back. Data sent is in the 14-byte Video ARC structure. See **Get Extended Field (C9 C3)** command, field 19, for syntax.	Nexio 7.0 Software Release

Output Timecode: The value set for Bit Mask 0x0800 is a bit-wise field indicating the output timecode control desired for the channel. If Bit 8 is set high, the **Get Current Time Sense (61 0C)** command will make use of the selected timecode implementation instead of the legacy SOM-based timecode reporting.

Table 2-28: Output Timecode Control

Bit	Description
8	1 = Provides the full timecode implementation for LTC/VITC/UB in the **61 0C** command 0 = Legacy support in the **61 0C** command, always returning SOM-based timecode and claim it's Sony LTC
7	insert continuous VITC based on SOM
6	insert continuous LTC based on SOM
5	VITC line duplication (see RP188 standard)
4 - 0	VITC line select (see RP188 standard)

C0 C3 Get Video Restriction

Syntax: **C0 C2**

Response: **D1 C2 RESTRICTION**
RESTRICTION: 1-byte unsigned integer

Description: This is a legacy command that retrieves the current channel's video format restrictions. This is required only for the VR and original Nexio servers where DV and MPEG material could not be played back-to-back on the same channel. As a result, every channel in a VR or Nexio system was locked down to either DV-capable or MPEG-capable.

Table 2-29: Get Video Restrictions

Restriction	Video Format
0	Unrestricted
1	MPEG
2	DV

The Nexio AMP and Volt servers are able to play DV and MPEG clips back-to-back on the same channel. As a result, the response in those systems will always return a value of "0".

The "MPEG" value includes all forms of MPEG2 including long GOP, I-Frame, 4:2:0, 4:2:2, and IMX.

The "DV" value includes DVCAM, DVCPRO, and DVCPRO50.

C0 C3 Get Max Extended ID Size

Syntax: **C0 C3**

Response: **D1 C3 SIZE***
SIZE: 1-byte unsigned integer indicating size of extended IDs in bytes

Description: This is a legacy command that retrieves the maximum extended ID size that was set for the server system in the **Initialize Disk** window of the LLM. Nexio AMP and Volt systems are always 64 byte ID systems.

* Response format violates the convention of data count embedded in LSN of CMD-1.

CX A4 — Get Current Time

Syntax: **C0 A4** or
C1 A4 TYPE
TYPE: 1-byte unsigned integer

Response: **D4 A4 TIME** or
D5 A4 TIME FRACTIONAL or
D0 A4 if timer TYPE is not available
TIME: 4-bytes indicating the time in the format of FF.SS.MM.HH
FRACTIONAL: 1-byte unsigned integer indicating fractional frame

Description: This command returns the current time on the server. It should be used by the controller to synchronize its own clock in order to correctly interpret subsequent time stamped notifications.

Refer to the following table if a **TYPE** is included with the command for definition of the source of the time.

Table 2-30: Source Time Types

Type	Time Value	Server Response
0x00	Current time on server (same as C0 A4)	D4 A4 TIME
0x01	Current time on server in fractional frame units	D5 A4 TIME FRAC
0xFF	Timecode from attached RS-232 timecode reader	D4 A4 TIME

NTSC times are always returned in drop frame. The **FRACTIONAL** variable returned from the 0x01 **TYPE** is represented with 8 bits of precision (using a denominator = 256).

Drop Frame Timecode Note: In the timecode syntax of FF.SS.MM.HH, the frames unit has a base value of 40 to indicate drop frame timecode in NTSC. PAL timecode and non-drop frame NTSC are 0-based.

61 0C — Get Current Time Sense

Syntax: **61 0C CODE**
CODE: 1 byte unsigned integer

Response: **74 04 TC or**
TC: 4-bytes indicating a timecode point in the format of FF.SS.MM.HH

Description: This command is used to retrieve the server's current timecode. The timecode is reported in four bytes presented in reverse order: frames, seconds, minutes, hours. The most significant bit of the tens nibble of the seconds byte will be set high if the server is displaying field 2 video. If the specified **CODE** is missing or invalid, the server will return all nulls in place of timecode: 74 04 00 00 00 00.

The following table lists the available codes.

Table 2-31: Source Timecode Codes

Code	Type
0x01	LTC
0x02	
0x03	VITC / LTC
0x04	TIMER-1
0x05	TIMER-1 / LTC
0x08	TIMER-2
0x10	LTC UB
0x20	

As of the Nexio AMP 4.5 release, this command can provide embedded timecode information in the data returned, including discontinuities in the timecode on play out. However the channel's output timecode property must first be set using the 11th Bit Mask of the **Set Special Channel Properties (CF B1)** command. Discontinuities in timecode during recordings are not reported dynamically. Only on play out.

If the full embedded timecode implementation is not enabled, the server operates in legacy fashion returning the SOM-based timecode as though it were the Sony LTC type.

Drop Frame Timecode Note: In the timecode syntax of FF.SS.MM.HH, the frames unit has a base value of 40 to indicate drop frame timecode in NTSC. PAL timecode and non-drop frame NTSC are 0-based.

C8 B0 — Set Controller ID

Syntax: **C8 B0 CONTROL**

CONTROL 8-byte unsigned integer, MSB first

Response: **10 01**

Description: This command allows a controller to assign an ID for the current controller connection. The controller need only select a unique name for each connection it makes to an LLM.

This is useful for Nexio systems using the auto-clip protection feature. The self-assigned identifier in this command allows other controllers to be aware of which controller is currently protecting a clip when using the **Erase ID with Transaction (C8 4F)** command.

The auto-clip protection feature is turned off by default. It prevents the deletion of clips that are currently loaded in a channel anywhere on the Nexio system. It must be enabled on every server in the system to function correctly.

Channel Status Commands

C0 AC — Get Logical Channel

Syntax: **C0 AC**

Response: **D1 AC CHANNEL**
CHANNEL: 1-byte unsigned integer, indicating channel number

Description: This command retrieves the currently selected channel, where 0=Channel 1, 1=Channel 2, 2=Channel 3, and so on up to 7=Channel 8.

C1 AC — Set Logical Channel

Syntax: **C1 AC CHANNEL**
CHANNEL: 1-byte unsigned integer, indicating channel number selected

Response: **10 01**

Description: This command sets the currently selected channel for control. Subsequent commands will be routed to the specified CHANNEL, where 0=Channel 1, 1=Channel 2, 2=Channel 3, and so on up to 7=Channel 8.

If a controller sets the channel to a number more than the actual channels available, the channel will be set to the highest number available. This is a good method for a controller to determine the total number of channels available on a server.

A0 21 — Get Device ID

Syntax: **A0 21**

Response: **88 21 DID**
DID: Unterminated 8-byte string

Description: This command returns the value set by the **Set Device ID (A8 20)** command. This command is useful in distinguishing machines within a large system

A8 20 — Set Device ID

Syntax: **A8 20 DID**
DID: Unterminated 8-byte string

Response: **10 01**

Description: This command sets the server's Device ID for the current channel to the specified **DID**. This command is useful in distinguishing machines within a large system. Use the **Select Logical Channel (C1 AC)** command to select the desired channel.

C0 03 — Get Loaded ID

Syntax: **C0 03**

Response: **D8 03 ID** or
D0 03 if ID is not found
ID: 8-byte ID handle

Description: This command retrieves the currently loaded ID handle in the channel. If no ID is loaded, the server returns **D0 03**.

A8 18 — ID Status Request

Syntax: **A8 18 ID**
ID: 8-byte ID handle

Response: **81 18 STATUS**
STATUS: 1-byte unsigned integer

Description: This command returns the status of the specified ID handle. **STATUS** is defined by the following table:

Table 2-32: ID Status Report

Status	Description
0x00	ID does not exist
0x01	ID exists
0x02	ID is loaded
0x03	ID exists and is loaded

61 20 — Get Channel Status

Syntax: **61 20 DATA**
DATA: 1-byte unsigned integer

Response: **7x 20 BYTES, where x is the LSN of DATA**
BYTES: 1 or more bytes as defined by DATA

Description: This command retrieves the current channel's status. The data bytes returned are dependent on the **DATA** parameter. The most significant nibble of **DATA** defines the starting byte to be returned, and the least significant nibble of **DATA** defines the number of bytes after the starting byte to be returned.

See the following table for the interpretation of the return data against the command parameters.

Table 2-33: Channel Status Bytes

Byte	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0x00	0	0	0	0	0	0	0	**LOCAL**
0x01	**READY**	0	**STOP**	0	**REW**	**FF**	**REC**	**PLAY**
0x02	**SRV**	0	**SHTL**	**JOG**	**VAR**	**DIR**	**STILL**	**CUED**

0x03	**AUTO**	0	0	0	0	0	**PST OUT**	**PST IN**
0x04	0	**EE ON**	0	**EDIT**	**REVIEW**	0	**PREVIEW**	**PRRL**
0x05	0	**INSERT**	**ASMBL**	**VIDEO**	0	**TC**	**A2**	**A1**
0x06	0	0	0	0	0	0	0	0
0x07	0	0	0	0	0	0	0	0
0x08	0	0	0	0	0	0	0	**REC INHIBIT**
0x09	0	0	0	0	0	**PVW RDY**	**PVW OUT**	**PVW IN**
0x0A	0	0	0	0	0	0	**IDs DELETED**	**IDs ADDED**
0x0B	0	0	0	0	0	0	**EXCESS DRIFT**	**NO AUDIO OR VIDEO**
0x0C	0	0	0	0	0	0	0	0
0x0D	0	0	0	0	0	0	0	0
0x0E	0	0	0	0	0	0	0	0
0x0F	0	0	0	0	0	0	0	0

The **PVW RDY** bit is available only after the Nexio 5.5 Software Release for the Nexio AMP Server platform and Nexio XS platform after the 64-bit upgrade. It is an indication that the clip currently stacked in a play out channel is fully cued and ready for the transition to the active channel when needed.

The **Excess Drift** and **No Audio or Video** indicators were added as part of the Nexio 6.5 Software Release.

C0 C0 — Get Record Parameters

Syntax: **C0 C0**

Response: **D8 C0 VIDF VIDN VIDM VIDBR RES AUDIO VBI**
VIDF: 1-byte unsigned integer, representing the video format
VIDN: 1-byte unsigned integer, representing the video N value
VIDM: 1-byte unsigned integer, representing the video M value
VIDBR: 1-byte unsigned integer, representing the video bit rate in Mb/s
RES: 2-byte unsigned integer, always returned as zeroes
AUDIO: 1-byte unsigned integer, representing the number of audio tracks
VBI: 1-byte unsigned integer, representing whether VBI is present

Description: This command gets the current record parameters for the channel. Please refer to the **Get ID Metadata (C8 4A)** command for further definition of each of the above parameters.

Video bite rate **(VIDBR)** values greater than 50 make use of the progressive slope algorithm as defined in the **Get ID Metadata (C8 4A)** command. However because the progressive slope reduces the granularity of the larger bit rates stored to multiples of 5 or 10, a new **Get Extended Record Parameters (C0 C1)** command was added as part of the Nexio 6.0 Software Release.

Caution: This command should not be used if the auto-input resolution detection option is enabled on a Nexio server. If enabled, this command may provide incorrect information about the channel's record parameters. Instead you should use the **Get Auto-Detect Record Parameters (C5 CF)** command.

C0 C1 Get Extended Record Parameters

Syntax: **C0 C1**

Response: **D9 C1 VIDF VIDN VIDM VIDBR16 RES AUDIO VBI**
VIDF: 1-byte unsigned integer, representing the video format
VIDN: 1-byte unsigned integer, representing the video N value
VIDM: 1-byte unsigned integer, representing the video M value
VIDBR16: 2-byte unsigned integer, MSB first, representing the video bit rate in Mb/s
RES: 2-byte unsigned integer, always returned as zeroes
AUDIO: 1-byte unsigned integer, representing the number of audio tracks
VBI: 1-byte unsigned integer, representing whether VBI is present

Description: This command gets the current record parameters for the channel including the exact bit rate set for recordings over 50 Mb/s. The video bit rate parameter returns two bytes of data instead of the one byte provided with the original **C0 C0** command, which requires the progressive slope algorithm for bit rates greater than 50. This new command provides a better granularity of data in those cases.

Please refer to the **Get ID Metadata (C8 4A)** command for further definition of each of the above parameters.

This command was added as part of the Nexio 6.0 Software Release.

Caution: This command should not be used if the auto-input resolution detection option is enabled on a Nexio server. If enabled, this command may provide incorrect information about the channel's record parameters. Instead you should use the **Get Auto-Detect Record Parameters (C5 CF)** command.

C5 CF Get Auto-Detect Record Parameters

Syntax: **C5 CF RESOLUTION MASK**
RESOLUTION: 1-byte unsigned integer, representing the video resolution parameters desired
MASK: 4-byte unsigned integer, representing the record parameters desired, MSB first

Response: **DF CF BC PARAMETERS** or
D0 CF if an invalid resolution is specified
BC: 1-byte unsigned integer, indicating total amount of data to follow
PARAMETERS: The data requested as indicated by the **MASK**

Description: Available with the Nexio 7.0 Software Release, this command returns the requested record parameters for the current channel based on a specified video resolution. With the addition of auto-input resolution detection on record channels, record parameters are expected to be different based on the video resolution of the input to the record channel.

Each resolution must be queried separately and the **MASK** defines which of the record parameters are desired.

Table 2-34: Record Resolution

Data	Resolution
0x00	SD
0x01	720p

0x02	1080i
0x03	1080p

The values specified in MASK are returned in ascending bit position order and is represented MSB first.

Table 2-35: Record MASK

Bit	MASK	Parameter Size	Description
0	0x00000001	1	Resolution detection enabled for specified resolution. 0 = legacy settings and commands. 1 = input resolution detection logic.
1	0x00000002	1	Video Format
2	0x00000004	1	Video N value
3	0x00000008	1	Video M value
4	0x00000010	4	Exact Video Bit Rate
5	0x00000020	2	Metabytes Per Frame
6	0x00000040	1	Aspect Ratio
7	0x00000080	1	Audio Tracks
8	0x00000100	1	Video Format Code
9	0x00000200	1	VBI Format
10	0x00000400	4	VBI Start Line
11	0x00000800	4	VBI Stop Line
12	0x00001000	4	VBI Max Stored Lines
13	0x00002000	4	VBI Threshold Level
14	0x00004000	4	VBI Threshold Count
15	0x00008000	1	VBI Compressed
16	0x00010000	4	Audio Bits
17	0x00020000	4	Video Pixel Aspect Ratio
18	0x00040000	32	Input Audio Routing Mask

Please refer to the **Get ID Metadata (C8 4A)** and **Get Extended Field (C9 C3)** commands for further definition of many of the above fields.

To learn how to change the record parameters for each resolution see the **Set Auto-Detect Record Parameters (CF CF)** command.

C8 C1 — Set Record Parameters

Syntax: **C8 C1 VIDF VIDN VIDM VIDBR RES AUDIO VBI**
VIDF: 1-byte unsigned integer, representing the video format
VIDN: 1-byte unsigned integer, representing the video N value
VIDM: 1-byte unsigned integer, representing the video M value
VIDBR: 1-byte unsigned integer, representing the video bit rate in Mb/s
RES: 2-byte unsigned integer, always returned as zeroes
AUDIO: 1-byte unsigned integer, representing the number of audio tracks
VBI: 1-byte unsigned integer, representing whether VBI is present

Response: **10 01**

Description: This command sets record parameters for a channel. Please refer to the **Get ID Metadata (C8 4A)** command for further definition of each of the above field

definitions. The **VIDF** parameter includes support for setting an SD clip to an aspect ratio of 16:9 as defined in the **C8 4A** command.

The parameters will be applied to all the new clips recorded on that channel. It is the responsibility of the controller to determine the capabilities of the server channel which it is controlling. The server will allow a controller to set up a channel to record using parameters which may be invalid for the specific channel.

It is also the responsibility of the controller to know the limits of the parameters based on video format, video resolution, and video standard. For example, no DV video format should have a VIDM or VIDN value greater than 1.

VIDBR: Video bite rate **(VIDBR)** values greater than 50 make use of the progressive slope algorithm as defined in the **Get ID Metadata (C8 4A)** command. However because the progressive slope reduces the granularity of the larger bit rates stored to multiples of 5 or 10, a new **Set Extended Record Parameters (C9 C1)** command was added.

Caution: This command should not be used if the auto-input resolution detection option is enabled on a Nexio server. If enabled, this command may provide incorrect information about the channel's record parameters. Instead you should use the **Set Auto-Detect Record Parameters (CF CF)** command.

C9 C1 — Set Extended Record Parameters

Syntax: **C9 C1 VIDF VIDN VIDM VIDBR16 RES AUDIO VBI**
VIDF: 1-byte unsigned integer, representing the video format
VIDN: 1-byte unsigned integer, representing the video N value
VIDM: 1-byte unsigned integer, representing the video M value
VIDBR16: 2-byte unsigned integer, MSB first, representing the video bit rate in Mb/s
RES: 2-byte unsigned integer, always returned as zeroes
AUDIO: 1-byte unsigned integer, representing the number of audio tracks
VBI: 1-byte unsigned integer, representing whether VBI is present

Response: **10 01**

Description: Available with the Nexio 6.0 Software Release, this command sets record parameters for a channel including the exact bit rate set for recordings over 50 Mb/s. The video bit rate parameter returns two bytes of data instead of the one byte provided with the original **C8 C1** command, which requires the progressive slope algorithm for bit rates greater than 50. This new command provides a better granularity of data in those cases.

Please refer to the **Get ID Metadata (C8 4A)** command for further definition of each of the above field definitions. The **VIDF** parameter includes support for setting an SD clip to an aspect ratio of 16:9 as defined in the **C8 4A** command.

The parameters will be applied to all the new clips recorded on that channel. It is the responsibility of the controller to determine the capabilities of the server channel which it is controlling. The server will allow a controller to set up a channel to record using parameters which may be invalid for the specific channel.

It is also the responsibility of the controller to know the limits of the parameters based on video format, video resolution, and video standard. For example, no DV video format should have a VIDM or VIDN value greater than 1.

Caution: This command should not be used if the auto-input resolution detection option is enabled on a Nexio server. If enabled, this command may provide incorrect information about the channel's record parameters. Instead you should use the **Set Auto-Detect Record Parameters (CF CF)** command.

CF CF — Set Auto-Detect Record Parameters

Syntax: **CF CF BC RESOLUTION MASK PARAMETERS**
RESOLUTION: 1-byte unsigned integer, representing the video resolution parameters desired
MASK: 4-byte unsigned integer, representing the record parameters desired, MSB first
PARAMETERS: The record parameter data as indicated by the **MASK**

Response: **10 01** or
D0 CF if an invalid resolution is specified

Description: Available with the Nexio 7.0 Software Release, this command sets up the record parameters for the current channel based on a specified video resolution. With the addition of auto-input resolution detection on record channels, record parameters are expected to be different based on the video resolution of the input to the record channel.

Each resolution must be set separately and the **MASK** defines which of the record parameters are included with the command.

Table 2-36: Record Resolution

Data	Resolution
0x00	SD
0x01	720p
0x02	1080i
0x03	1080p

The values specified in MASK are returned in ascending bit position order and is represented MSB first.

Table 2-37: Record MASK

Bit	MASK	Parameter Size	Description
0	0x00000001	1	Resolution detection enabled for specified resolution. 0 = legacy settings and commands. 1 = input resolution detection logic.
1	0x00000002	1	Video Format
2	0x00000004	1	Video N value
3	0x00000008	1	Video M value
4	0x00000010	4	Exact Video Bit Rate
5	0x00000020	2	Metabytes Per Frame
6	0x00000040	1	Aspect Ratio
7	0x00000080	1	Audio Tracks
8	0x00000100	1	Video Format Code
9	0x00000200	1	VBI Format

10	0x00000400	4	VBI Start Line
11	0x00000800	4	VBI Stop Line
12	0x00001000	4	VBI Max Stored Lines
13	0x00002000	4	VBI Threshold Level
14	0x00004000	4	VBI Threshold Count
15	0x00008000	1	VBI Compressed
16	0x00010000	4	Audio Bits
17	0x00020000	4	Video Pixel Aspect Ratio
18	0x00040000	32	Audio Routing Input Mask

Please refer to the **Get ID Metadata (C8 4A)** and **Get Extended Field (C9 C3)** commands for further definition of many of the above fields.

Bit 18, Audio Routing Input Mask, refers to the values written to extended field 24 for all new clips recorded once these values are set for any particular resolution.

C1 B7 Record Timecode Source

Syntax: **C1 B7 SOURCE**
SOURCE: 1-byte indicating the source of the timecode

Response: **10 01**

Description: This command allows a controller to select a specific timecode source(s) for recording. It will affect all subsequent record commands. As of the Nexio 6.5.2 Software Release, the LLM will retain the last timecode source set using this command. Prior to that release, the controller would have to resend the instruction with each server connection.

The record timecode **SOURCE** is a bit field with the following structure:

Table 2-38: Record Timecode Source

BIT7	BIT6	BIT5	BIT4	BIT3	BIT2	BIT1	BIT0
0	0	0	0	Time of Day	VITC2	VITC1	LTC

A bit is set to request a specific timecode source. If multiple sources are requested, the recorder will use the first source that receives valid timecode information, starting from low (LTC) to high (Time of Day). If no source is selected or if neither of the selected sources receives valid timecode information, then the timecode is determined by the current location of the channel (this is the default).

To support the input of LTC, the optional NXUSBLTC device must be connected to the server feeding timecode to the desired input channels.

Example: To ignore all external timecode:

```
C1 B7 00
```

To record from either VITC1 or VITC2 with VITC1 having priority:

```
C1 B7 06
```


C0 50 — Get Audio Status

Syntax: **C0 50**

Response: **D4 50 NUMBER RES SAMPLES**
NUMBER: 1-byte unsigned integer, indicating the number of audio channels enabled
RES: 1-byte unsigned integer, reserved and currently set to zero
SAMPLES: 2-byte unsigned integer, MSB first, indicating the sample rate of the audio

Description: This command retrieves a channel's current audio information. The return data indicates the number of active audio channels and their current audio sample rate.

C1 51 — Get Audio Levels

Syntax: **C1 51 CODE**
CODE: 1-byte unsigned integer, as indicated in table below

Response: **D4 51 CH1 CH2** or
D8 51 CH1 CH2 CH3 CH4 or
DF 51 BC CH1 CH2 … CH16
CH*n*: 2-byte unsigned integer, MSB first, as defined in tables below
BC: 1-byte unsigned integer, indicating total amount of data to follow

Description: This command returns the current audio levels and gain settings for a channel. Values returned are two-byte values per channel between 0 and 65534.

The **CODE** parameter selects which level or setting to return.

Table 2-39: Audio Level Code

Code	Description
0x00	Input Meter Level
0x01	Output Meter Level
0x02	Input Level Setting
0x03	Output Level Setting

Cx 52 — Set Audio Input Volume(s)

Syntax: **C3 52 TRACK VOLUME** or
C4 52 VOL1 VOL2 or
C8 52 VOL1 VOL2 VOL3 VOL4 or
CF 52 BC VOL1 VOL2 … VOL16 *
TRACK: 1-byte unsigned integer to indicate the specific audio track to be affected
VOLUME: 2-byte unsigned integer, MSB first, to indicate volume level
VOL*n*: 2-byte unsigned integer, MSB first, to indicate volume level for specified audio channel *n*.
BC: 1-byte unsigned integer, indicating total amount of data to follow.

Response: **10 01**

Description: This command sets the audio input gain level for one individual track or all tracks, depending on which syntax is used. Volume levels specified for tracks that do not exist will be ignored.

Volume values sent must be between 0 and 65534.

* Command format violates the convention of data count embedded in LSN of CMD-1.

Cx 53 Set Audio Output Volume(s)

Syntax: **C3 53 TRACK VOLUME** or
C4 53 VOL1 VOL2 or
C8 53 VOL1 VOL2 VOL3 VOL4 or
CF 53 BC VOL1 VOL2 … VOL16 *

TRACK: 1-byte unsigned integer to indicate the specific audio track to be affected
VOLUME: 2-byte unsigned integer, MSB first, to indicate volume level
VOL*n*: 2-byte unsigned integer, MSB first, to indicate volume level for specified audio channel *n*.
BC: 1-byte unsigned integer, indicating total amount of data to follow.

Response: **10 01**

Description: This command sets the audio output gain level for one individual track or all tracks, depending on which syntax is used. Volume levels specified for tracks that do not exist will be ignored.

Volume values sent must be between 0 and 65534.

* Command format violates the convention of data count embedded in LSN of CMD-1.

C1 54 Get Audio Routing Parameters

Syntax: **C1 54 PARAMETER**
PARAMETER: 1-byte unsigned integers

Response: **DF 54 BC DATA** or
D0 54 if specified parameter is invalid
BC: 1-byte unsigned integer, indicating total amount of data to follow
DATA: 32-bytes of data

Description: This command is used to retrieve the audio routing configuration of the current channel. There are five different parameters possible, two for setting up the input configuration values and three for the output.

Table 2-40: Video Format Nibble

Parameter	Parameter Name	Description
0x00	Input Mask HD	32-byte audio routing mask for HD ingest (see caution below)
0x01	Input Mask SD	32-byte audio routing mask for SD ingest (see caution below)
0x02	Output Mask Primary	32-byte audio routing mask
0x03	Output Mask Secondary	32-byte audio routing mask
0x04	Output Mask Tertiary	32-byte audio routing mask

The input mask contains a default value for the output mask that is to be automatically written to Extended Field 24 of the clip upon record cue up. The channel determines whether to use the SD input mask or HD input mask based on its record configuration.

The output mask defines the audio routing for playout. Each byte in the output mask represents an audio track and can be assigned to any of the 256 track tags defined by the Audio Track Editor. Each channel supports 3 possible output masks applied in a hierarchical order.

If no output mask is specified, then audio routing is pass-thru.

Caution: This command should not be used for the input mask values if the auto-input resolution detection option is enabled on a Nexio server. If enabled, this command may provide incorrect information about the channel's record parameters. Instead you should use the **Get Auto-Detect Record Parameters (C5 CF)** command to set Bit 18 for each video resolution.

CF 55 — Set Audio Routing Parameters

Syntax: **CF 55 BC PARAMETER DATA**
BC: 1-byte unsigned integer, indicating total amount of data to follow
PARAMETER: 1-byte unsigned integer
DATA: 32-bytes of data

Response: **10 01**

Description: This command is used to set the audio routing configuration of the current channel. There are five different parameters possible, two for setting up the input configuration values and three for the output.

The allowed values for **PARAMETER** can be found in the **Get Audio Routing Parameters (C1 54)** command. The format of the **DATA** values is defined in the section above describing the Audio Track Router.

Caution: This command should not be used for the two input mask values if the auto-input resolution detection option is enabled on a Nexio server. If enabled, this command may provide incorrect information about the channel's record parameters. Instead you should use the **Set Auto-Detect Record Parameters (C5 CF)** command to set Bit 18 for each video resolution.

C2 56 — Get Audio Routing Profile

Syntax: **C2 56 PROFILE PARAMETER**
PROFILE: 1-byte unsigned integer, valid values between 00 and 31
PARAMETER: 1-byte unsigned integer

Response: **DF 56 BC VALUE** or
D0 56 if invalid PROFILE or PARAMETER requested
BC: 1-byte unsigned integer, indicating total amount of data to follow
VALUE: 32-bytes representing audio routing mask

Description: This command is used to retrieve the data values of an audio routing profile. The profile defines the output audio routing masks associated with a Nexio platform so that a controller can dynamically change the audio routing on the fly using the **Preset Audio Routing Profile (C1 16)** and **Preview Audio Routing Profile (C1 17)** commands.

Audio Routing Profiles are built using the **Set Audio Routing Profile (CF 57)** command. The maximum number of different profiles saved per Nexio is 32. Each profile includes values for the three different audio routing masks. PARAMETER defines which mask is requested.

Table 2-41: Audio Routing Parameter

Parameter	Parameter Name
0x00	Output Mask Primary
0x01	Output Mask Secondary
0x02	Output Mask Tertiary

The **VALUE** returned will always be 32 bytes, one byte representing each routable audio track associated with a clip. The structure of these 32 bytes is based on the audio track tags, as defined at the top of this chapter in the section on the Audio Track Router.

Note: This command was introduced as part of the Nexio 7.0 Software Release and does not function in prior released software.

CF 57 Set Audio Routing Profile

Syntax: **CF 57 BC PROFILE PARAMETER VALUE**
BC: 1-byte unsigned integer, indicating total amount of data to follow
PROFILE: 1-byte unsigned integer, valid values between 00 and 31
PARAMETER: 1-byte unsigned integer
VALUE: 32-bytes representing audio routing mask

Response: **10 01** or
D0 57 if invalid values set

Description: This command is used to set the data values of an audio routing profile. The profile defines the output audio routing masks associated with a Nexio platform so that a controller can dynamically change the audio routing on the fly using the **Preset Audio Routing Profile (C1 16)** and **Preview Audio Routing Profile (C1 17)** commands.

The maximum number of different profiles saved per Nexio is 32. Each profile includes values for the three different audio routing masks. PARAMETER defines which mask is requested.

Table 2-42: Audio Routing Parameter

Parameter	Parameter Name
0x00	Output Mask Primary
0x01	Output Mask Secondary
0x02	Output Mask Tertiary

The **VALUE** set for each **PARAMETER** of each **PROFILE** will always be 32 bytes, one byte representing each routable audio track associated with a clip. The structure of these 32 bytes is based on the audio track tags, as defined at the top of this chapter in the section on the Audio Track Router.

Note: This command was introduced as part of the Nexio 7.0 Software Release and does not function in prior released software.

CE 67 — Set Input Video ARC

Syntax: **CE 67 MODE HSIZE HPOS HCROP VSIZE VPOS VCROP**
MODE: 2-byte unsigned integer, MSB first, representing the ARC_MODE
HSIZE: 2-byte unsigned integer, MSB first
HPOS: 2-byte unsigned integer, MSB first
HCROP: 2 1-byte unsigned integers
VSIZE: 2-byte unsigned integer, MSB first
VPOS: 2-byte unsigned integer, MSB first
VCROP: 2 1-byte unsigned integers

Response: **10 01**

Description: This command allows a controller to adjust the video ARC settings of a selected input channel and also controls the E-E ARC, if applicable. These parameters are defined in the **Set Manual Video ARC (CF 66)** command.

C4 6E — Get Genlock Properties

Syntax: **C4 6E MASK**
MASK: 4-byte unsigned integer, MSB first

Response: **DF 6E BC PROP_1 PROP_2 … PROP_N** or
D0 6E if no valid properties
BC: 1-byte unsigned integer, indicating total amount of data to follow.
PROP_N: Variable based on properties requested according to tables below

Description: This command returns the values of all the Genlock-related properties specified by the MASK, in ascending bit position order. The bit mask and all the returned property values are represented MSB first.

The possible values retrieved depend on the Nexio server. The following two tables represent first the Nexio AMP and then the Nexio XS servers. Property values supported only on the XS server will return null values when requested on an AMP server.

Table 2-43: Nexio AMP Genlock Properties

Bit	Mask	PROP_N Size	PROP_N Values Returned
0	0x00000001	1	reference source, where: 0 = internal 1 = video in 2 = reference in
5	0x00000020	2	vertical delay, in lines, 16-bit integer
6	0x00000040	2	horizontal delay, in pixels, 16-bit integer

7	0x00000080	1	Genlock status, where: 0 = locked 1 = unlocked
10	0x00000400	6	vertical delay range, in lines as 16-bit integer values for min, max, and unity
11	0x00000800	6	horizontal delay range, in pixels as 16-bit integer values for min, max, and unity

Table 2-44: Nexio XS Genlock Properties

Bit	Mask	PROP_N Size	PROP_N Values Returned
0	0x00000001	1	reference source, where: 0 = internal 1 = video in 2 = reference in 3 = reference in poor quality
1	0x00000002	1	reference resolution, where: 0 = SD 1 = 1080i 2 = 720p
2	0x00000004	1	flywheel enable, where: 0 = disable 1 = enable
3	0x00000008	4	flywheel unlock time, in milliseconds, 32-bit int
4	0x00000010	4	flywheel recovery time, in milliseconds, 32-bit int
5	0x00000020	2	vertical delay, in lines, 16-bit int
6	0x00000040	2	horizontal delay, in pixels, 16-bit int
7	0x00000080	1	Genlock status, where: 0 = locked 1 = unlocked 2 = free-running 3 = recovering 4 = relocking
8	0x00000100	12	flywheel unlock time range, in millisecond values for min, max, and unity
9	0x00000200	12	flywheel recovery time range min/max/unity, in milliseconds
10	0X00000400	6	vertical delay range, in lines as 16-bit int values for min, max, and unity
11	0X00000080	6	horizontal delay range, in pixels as 16-bit int values for min, max, and unity
12	0x00001000	2	Genlock vertical delay (in lines, 16-bit int)
13	0x00002000	6	Genlock vertical delay range: min/max/unity (in lines, 16-bit int's)

CF 6E — Set Genlock Properties

Syntax: **CF 6E BC MASK PROP_1 PROP_2 … PROP_N**
BC: 1-byte unsigned integer, indicating total amount of data to follow.
MASK: 4-byte unsigned integer, MSB first
PROP_N: Variable based on properties to be set according to tables below

Response: **D1 6E STATUS**
STATUS: 1-byte unsigned integer

Description: This command allows a controller to set some of the Genlock-related properties available on the Nexio AMP and XS servers. Which properties are set depends on the **MASK** passed to the server and the capabilities of the server.

The server returns the **STATUS** byte to indicate success – a value of "0" – or failure – a value of "1." Failure is typically a result of the amount of PROP_N sent not being consistent with the **MASK** value.

The following two tables represent the support for this command in first the Nexio AMP and then the Nexio XS servers.

Table 2-45: Nexio AMP Genlock Properties

Bit	Mask	PROP_N Size	PROP_N Values Sent
0	0x0001	1	reference source, where: 0 = internal 1 = video in 2 = reference in
5	0x0020	2	vertical delay, in lines, 16-bit int
6	0x0040	2	horizontal delay, in pixels, 16-bit int

Table 2-46: Nexio XS Genlock Properties

Bit	Mask	PROP_N Size	PROP_N Values Sent
0	0x0001	1	reference source, where: 0 = internal 1 = video in 2 = reference in
1	0x0002	1	reference resolution, where: 0 = SD 1 = 1080i 2 = 720p
2	0x0004	1	flywheel enable, where: 0 = disable 1 = enable
3	0x0008	4	flywheel unlock time, in milliseconds, 32-bit int
4	0x0010	4	flywheel recovery time, in milliseconds, 32-bit int
5	0x0020	2	vertical delay, in lines, 16-bit int
6	0x0040	2	horizontal delay, in pixels, 16-bit int

C0 C8 — Get Channel Sync Mask

Syntax: **C0 C8**

Response: **D4 C8 MASK**
MASK: 4-byte unsigned integer, MSB first

Description: This command retrieves the current **MASK** setting for channel synchronization, a feature where play out channels can be synchronized in their response to transport commands. It is important that the controller query every channel it controls to know if any are currently synchronized.

See the **Set Channel Sync Mask (C4 C8)** command for the MASK definition.

C4 C8 — Set Channel Sync Mask

Syntax: **C4 C8 MASK**
MASK: 4-byte unsigned integer, MSB first

Response: **10 01**

Description: This command sets and unsets the ability to synchronize the play out of multiple channels in their response to transport commands. A bit value of "1" in the **MASK** parameter indicates that the corresponding channel will become subject to all subsequent transport commands that are issued to its synchronous channel.

Typically for a 4-channel play out server, channel 1 is linked to channel 2, while channel 3 is linked to channel 4, where channels 2 and 4 are the master channels which accept the transport commands.

The controller must first select the channel it wants to be master by using the **Set Logical Channel (C1 AC)** command. The **MASK** then identifies itself and the channel or channels it is controlling via a bitwise definition.

Example: The command as sent to channel 4 so that it is master over channel 3 (bits = 00001100):

```
C4 C8 00 00 00 0C
```

To reverse to normal operation so channel 4 only controls channel 4 (bits = 00001000):

```
C4 C8 00 00 00 08
```

There is an independent sync transport mask in each channel so care must be taken to send the transport commands to the same logical channel that received the mask via this command.

Channel synchronization is supported for the following commands:

Table 2-47: Channel Sync Support

Nexio Protocol	
20 00 Stop	2x 1x Var. Play Forward
20 01 Play	2x 2x Var. Play Reverse
20 02 Record	2x 31 Cue Up With Data
20 10 Fast Fwd	Cx 86 Fast Cue Up With Data
20 20 Rewind	Ax 04 Preset Preview In
20 0F Eject	C0 02 Next Cue
VDCP Protocol	
20.24/A0.24 Play Cue 10 05 Step	20.25/A0.25 Cue With Data

As part of the cue up process for channel synchronization, when a controller loads a clip into the master channel, the server looks at the specified clip's **Link** field to know which clip to load into the slave channel. The **Link** field is a metadata field retrieved using the **Get Extended Field (C9 C3)** command which stores the name of a clip that is linked to the master clip. When the master clip is loaded into a sync-

enabled channel, its linked clip gets loaded in the slave channel. If there is no ID in the **Link** field, no clip gets loaded in the slave channel.

There is an optional registry feature in which, if an ID does not have a linked clip, the channel sync logic will load the same clip into both the master and the slave channels. This is useful for controllers that wish to play the same clip out of two channels at the same time. The main reason for doing so would be if one channel is pre-configured as an SD channel and the other is an HD channel. In this way, any unlinked clips will play out simultaneously in both SD and HD (assuming up- and down-conversion is enabled).

The registry setting to enable this "one clip loads into two channels" logic is a DWORD setting called ***ClipSyncMode***, which resides in the LLM's ***Control*** registry branch. Set to "1," the feature is enabled. Set to "0," the default, it is disabled.

Field accurate synchronous operation between the selected channels is guaranteed, except as follows:

- **20 0F** when issued while in playback. It can be used however to synchronously terminate recording
- Variable play commands when going at more than 150 % speed
- All decompression commands above when issued in E-E state They will however synchronously terminate an ongoing recording
- In stacked play back when the clip loaded in the slave channel is shorter than the clip loaded in the master channel

The channel sync logic works best in stacked playback mode when both the master and slave clips are of the same length. Also the **Preset Out** commands, **4x 15** and **Ax 05**, are not recommended for stacked and synced play back. Use of these commands may result in unexpected preset/preview behavior.

The controller should always verify and wait for both channels to cue before sending any play command. It is possible for the channels to fall out of sync because one channel took longer than the other to cue up.

CF B0 — Lock Channel Control

Syntax: **CF B0 BC MODE LABEL ***
BC: 1-byte unsigned integer, indicating total amount of data to follow.
MODE: 1-byte unsigned integer
LABEL: String of text to indicate name of lock

Response:

D1 B0 STATUS	or
DF B0 BC LABEL	or
DF B0 BC EXCL LABEL	or
10 01	depending on MODE

STATUS: 1 byte unsigned integer to indicate lock status
LABEL: String of text to indicate name of lock, non-null terminated
EXCL: 1 byte indicating exclusive nature of the lock

Description: This command allows controllers to name channels and lock their use so other controllers who support this command do not conflict with their control of the

channel. This command will have no effect on controllers that do not check for lock control when they first take control of a channel.

This command, based on the **MODE** parameter defined below, allows a controller to lock a channel for shared or exclusive use.

Table 2-48: Channel Lock Modes

Mode	Description	Server Returns
	Server Network Locks	
0x00	Status of lock	D1 B0 STATUS
0x01	Acquire lock for shared use	10 01
0x02	Acquire lock for exclusive use	10 01
0x03	Release lock	10 01
0x04	Release all locks with prefix of LABEL	10 01
	Per Connection Local Locks	
0x05	Set the connection's owner label	10 01
0x06	Acquire local lock for shared use	D1 B0 STATUS or DF B0 BC LABEL
0x07	Acquire local lock for exclusive use	D1 B0 STATUS or DF B0 BC LABEL
0x08	Release local lock	10 01
0x09	Force release of local lock by all owners	10 01
0x0A	Query a local lock's owner	D0 B0 DF B0 BC EXCL LABEL

There are two sets of lock commands, depending on a controller's requirements. The first five **MODEs** are limited in use to controllers that wish to set up channel locks but not necessarily know who owns which locks. Because the lock ownership is handled per machine, there is no real association between the channels and the locks.

Meanwhile, the second group of **MODEs** provides per-controller lock ownership. They provide a direct correlation between the channels locked and the controller doing the locking. This set of commands includes the ability to query as to which connection has a lock on any specific channel.

Per Server: To acquire a lock in the "Server Network Locks", a controller would use **MODE** 1 or 2, whichever is appropriate for its use, and then poll using **MODE** 0 with the same **LABEL** until the returned status changes from "Request Pending". If the channel has been successfully acquired, the status will change to "Owned".

Table 2-49: Lock Status Bitmap - Mode 0

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	0	0	0	0	Request Pending	Exclusive	Owned

Requests for several independent locks can be pending at the same time because **MODE** 0 looks for the specified **LABEL**. So the various threads of the controlling application are not restricted to poll inside a critical section.

In order to avoid name conflicts between different controlling applications a possible strategy would be to precede all the names by the application's name (except for the locks that eventually will need to be shared by all applications).

Per Connection: Acquiring a lock in the "Per Connection Local Locks" is very similar in command structure to the "Server Network Locks," except instead of appending the controller's label to the lock's name, the controller begins the lock process by storing its owner's label using **MODE** 5.

The real difference is seen in how the LLM responds to the commands. Because the "Per Connection" locks are local, the result can be returned in the same command. No polling is necessary.

Name conflicts are not as big of a problem either. A controller can use the same lock when controlling channels on different servers because the lock names themselves are defined locally and not across the network.

In addition, the "Per Connection" locks provide an emergency mode (9) that allows a controller to force the release of a lock by all connections and another mode (10) that allows a controller to query who owns a lock.

Table 2-50: Lock Status Bitmap - Mode 6 & 7

LLM Response	Description
D1 B0 00	Success
D1 B0 01	Failed due to lock table being full
DF B0 BC LABEL	Failed due to at least one other owner, as specified in the LABEL returned parameter

The Nexio applications NXOS and Nexio Remote currently support the "Per Connection" locks modes.

* Command format violates the convention of data count embedded in LSN of CMD-1.

Mediabase Commands

A0 14 List First ID Handle

Syntax: **A0 14**

Response: **88 14 ID** or
80 14 if none found
ID: 8-byte ID handle

Description: This command returns the first existing ID handle in the server's disk database. This command should be followed by a series of **List Next ID Handle (A0 15)** commands to retrieve the rest of the list. If no IDs exist in the database, **80 14** is returned.

After receiving an ID Handle, a controller should use the **Get Extended ID from ID Handle (C8 C3)** command to retrieve the ID's proper name.

A0 15 List Next ID Handle

Syntax: **A0 15**

Response: **88 14 ID** or
80 14 if at end of list
ID: **8-byte ID handle**

Description: This command will follow a **List First ID Handle (A0 14)** command or a previous **List Next ID Handle** command. It returns the next existing ID handle in the server's disk database. If no more IDs exist in the database, **80 14** is returned.

After receiving each ID Handle, a controller should use the **Get Extended ID from ID Handle (C8 C3)** command to retrieve the ID's proper name.

C1 4C List First ID List

Syntax: **C1 4C LIST**
LIST: 1-byte unsigned integer, based on table below

Response: See table below or
D0 4C if none found
BC: 1-byte unsigned integer, indicating total byte count of the data to follow
ID: 8-byte ID handle
XID: Extended ID, in Unicode format when enabled

Description: This command returns the first existing ID of the specified list. To retrieve the remaining IDs in the list, a controller should follow up this command with the **List Next ID List (C0 4D)** command.

See the table below for the interpretation of the **LIST** data and the response syntax. If no IDs are in the specified list, **D0 4C** is returned.

Table 2-51: List First ID List Table

List	Description	Return Data
0x01	Main ID Handle List	D8 4C ID
0x02	ID Handles Added List	D8 4C ID
0x03	ID Handles Deleted List	D8 4C ID
0x11	Main Extended ID List	DF 4C BC XID *
0x12	Extended IDs Added List	DF 4C BC XID *
0x13	Extended IDs Deleted List	DF 4C BC XID *
0x21	Main ID Handle/Extended ID List	DF 4C BC ID XID *
0x22	ID Handles/Extended IDs Added List	DF 4C BC ID XID *
0x23	ID Handles/Extended IDs Deleted List	DF 4C BC ID XID *

The "Main" lists, 0x01, 0x11, and 0x21, provide the same functionality as the **List First ID Handle (A0 14)** command except the controller has the option to retrieve just the Extended ID or both the ID handle and the extended ID of the first clip in the list.

The "Added" and "Deleted" lists can be monitored via the **Get Channel Status (61 20)** command's 0x0A byte. If the lists are empty, the IDs ADDED and IDs DELETED flags will be set to 0.

When using the ID handle lists, 0x01 – 0x03, a controller should use the **Get Extended ID from ID Handle (C8 C3)** command to retrieve the ID's proper name.

Example: Below is an example of a response to **LIST** modes 0x21-0x23 where the byte count is 0x0E (14), the ID handle is "%0000003" and the Extended ID is "abc" presented in Unicode. In these three modes, the ID handle portion of the response is always 8 bytes.

DF 4C 0E 25 30 30 30 30 30 30 33 61 00 62 00 63 00

Unicode Note: To make use of Unicode text in this command, the controller must have set the Unicode bit in the **Set Machine Control Options (C8 B1)** command.

* Response format violates the convention of data count embedded in LSN of CMD-1.

C0 4D — List Next ID List

Syntax: **C0 4D**

Response: See table below or
D0 4D if at end of list

BC: 1-byte unsigned integer, total byte count of the data to follow

ID: 8-byte ID handle

XID: Extended ID, in Unicode format when enabled

Description: This command returns the next existing ID from the ID list selected with the **List First ID List (C1 4C)**. It is the responsibility of the controller to know which **LIST** mode was initiated with **C1 4C** when processing responses. If no IDs remain in the specified list, **D0 4D** is returned.

Table 2-52: List Next ID List Table

List	Description	Return Data
0x01	Main ID Handle List	D8.4D ID
0x02	ID Handles Added List	D8.4D ID
0x03	ID Handles Deleted List	D8.4D ID
0x11	Main Extended ID List	DF.4D BC XID *
0x12	Extended IDs Added List	DF.4D BC XID *
0x13	Extended IDs Deleted List	DF.4D BC XID *
0x21	Main ID Handle/Extended ID List	DF.4D BC ID XID *
0x22	ID Handles/Extended IDs Added List	DF.4D BC ID XID *
0x23	ID Handles/Extended IDs Deleted List	DF.4D BC ID XID *

The "Main" lists, 0x01, 0x11, and 0x21, provide the same functionality as the **List First ID Handle (A0 14)** command except the controller has the option to retrieve just the Extended ID or both the ID handle and the extended ID of the first clip in the list.

Unicode Note: To make use of Unicode text in this command, the controller must have set the Unicode bit in the **Set Machine Control Options (C8 B1)** command.

* Response format violates the convention of data count embedded in LSN of CMD-1.

C8 C3 — Get Extended ID from ID Handle

Syntax: **C8 C3 ID**
ID: 8-byte ID handle

Response: **DF C3 BC XID*** or
D0 C3 if ID handle not found
BC: 1-byte unsigned integer, byte count of Extended ID (**XID**)
XID: Extended ID, in Unicode format when enabled

Description: This command returns the Extended ID based on a specified ID handle. If the ID handle is not found, the server returns **D0 C3**.

Unicode Note: To make use of Unicode text in this command, the controller must have set the Unicode bit in the **Set Machine Control Options (C8 B1)** command.

* Response format violates the convention of data count embedded in LSN of CMD-1

CF C4 — Get ID Handle from Extended ID

Syntax: **CF C4 BC XID***
BC: 1-byte unsigned integer, byte count of Extended ID (**XID**)
XID: Extended ID, in Unicode format when enabled

Response: **D8 C4 ID**
ID: 8-byte ID handle

Description: This command returns an ID handle based on a specified Extended ID. If the Extended ID is not found, the server returns **D0 C4**.

Unicode Note: To make use of Unicode text in this command, the controller must have set the Unicode bit in the **Set Machine Control Options (C8 B1)** command.

* Command format violates the convention of data count embedded in LSN of CMD-1

CF CA — Rename Extended ID

Syntax: **CF CA BC ID NEW_XID***
BC: 1-byte unsigned integer, total byte count of data to follow
ID: 8-byte ID handle
NEW_XID: New Extended ID name, in Unicode format when enabled

Response: **D4 CA TID**
TID: 4-byte unsigned integer, MSB first, identifying a Transaction ID associated with the command.

Description: This command changes the Extended ID of the specified ID handle to the newly specified Extended ID. It is the responsibility of the controller to monitor the transaction status using the **Get Transaction Status (C4 B3)** command at greater than 50 ms intervals until the transaction returns a success or failure.

If the rename succeeds, the ID handle/old Extended ID will be placed in the IDs Added List and the IDs ADDED status bit will be set. In addition, the old Extended ID will be placed in the IDs Deleted List along with a placeholder ID handle. It is the responsibility of the controller to detect the IDs Added bit and update its database with the new Extended ID.

If another clip with the same Extended ID already exists, the transaction will fail with TRANSACTION_NAME_CONFLICT (0x0B).

Example: Rename the Extended Name of ID Handle “%0001234” to “123”, as entered in a Unicode string.

```
CF CA 0E 25 30 30 30 31 32 33 34 31 00 32 00 33 00
                      // Send Rename command
                      // Server returns TID 0x00000009
C4 B3 00 00 00 09     // Monitor progress until it
                      returns
                      // TRANSACTION_SUCCESS
C1 4C 22              // Check IDs Added List…
C0 4D                 // …until renamed ID is included in
                      the response

Response: DF 4C 0E 25 30 30 30 31 32 33 34 31 00 32 00 33 00
```

The last two commands demonstrate a method of detecting a renamed ID using Mode=0x22 of the **List First ID List (C1 4C)** command. The controller should check the documentation for the **LIST** mode most appropriate for its needs.

Unicode Note: To make use of Unicode text in this command, the controller must have set the Unicode bit in the **Set Machine Control Options (C8 B1)** command.

*Command format violates the convention of data count embedded in LSN of CMD-1.

C8 4A — Get ID Metadata

Syntax: **C8 4A ID**
ID: 8-byte ID handle

Response: **DF 5F DATA0-94 *** or
D0 00 if ID handle not found
DATA: 95 bytes, see tables below for breakout

Description: This command returns the normal table of metadata associated with an ID handle. See the following tables for the interpretation of the return data. If the ID handle is not found on disk, the server returns **D0 00**.

Fields below shaded in gray are modifiable using the **Set ID Metadata (CF 44)** command. None of these fields is enabled for Unicode text.

Table 2-53: ID Metadata Structure

Data#	Size	Field	Description
0-7	8	Code	Base ID reference. If Code equals ID handle, then the clip is a parent clip. Otherwise, the Code field is the ID handle of the ID's parent clip.
8	1	Video Format	See table below- it also includes an indication of aspect ratio
9	1	Video N Value	Total size of GOP, usually from 1-to-16, but can be larger if the clip is H.264.
10	1	Video M Value	Distance between reference picture frames in GOP, possible values of 1-3, must be an equal divisor of Video N value
11	1	Video Bit Rate	Use the information in this field only if the Exact Video Bit Rate = 0 (**Data** #20-21). It reports the compressed bit rate of the video portion of the clip. See note below for interpreting values >50 Mb/s.
12-15	4	Start	Start timecode of ID in format of "FF.SS.MM.HH". If the timecode is NTSC drop-frame, the frames unit is padded by 40 frames where "40 00 00 01" represents a start time of exactly one hour.
16-19	4	Duration	Length of recorded material in frames in format of LSB first
20-21	2	Exact Video Bit Rate	Exact report of the compressed bit rate of the video portion of the clip. The 2 bytes are returned LSB first. This information replaces information provided in **Data** #11 as of the Nexio 6.0 Software Release
22	1	---	Reserved; returned with zero
23	1	Video Gain	Video procamp value, MSB first, see Set ID Metadata command fior more detail on the values
24	1	Video Setup	Video procamp value, MSB first, see Set ID Metadata command for more detail on the values
25	1	Chroma Gain	Video procamp value, MSB first, see Set ID Metadata command for more detail on the values
26-27	2	Hue	Video procamp value, MSB first, see Set ID Metadata command for more detail on the values
28-29	2	Children	Total number of subclips and Reference clips associated with the ID, in format of MSB first
30-31	2	References	Total number of Reference clips associated with the ID, in format of MSB first

32-33	2	---	Reserved; returned with zeroes
34	1	Audio tracks (deprecated)	This information is only valid for media with up to 8 audio tracks. Controllers should instead use the **Get Audio Track Information (C8 CA)** command to retrieve the most accurate report of audio tracks.
35	1	VBI Present	Indicates whether VBI is present, 0=no, 1=yes
36-39	4	Min Frame	Frame offset indicating the number of frames from the clip's first video packet to its actual start of message
40-41	2	Record Date (deprecated)	Old method for retrieving the date an ID was recorded, in the format of LSB first based on the bitmap table below. Controllers should use the Get Extended Field (C9 C3) command to retrieve field number 0 which contains the more accurate date and time the ID was recorded.
42-43	2	Kill Date	Date ID is flagged to expire; in the format of LSB first based on the bitmap table below
44	1	TC Type	Timecode type: 0 = NDF, 1 = DF, 2 = PAL
45	1	---	Reserved; returned with zeroes
46	1	Disk	Disk where ID is located
47-72	25 + 1 null	Old Description	Old text description field; null padded to 26 bytes, no longer used
73-88	15 + 1 null	Old Agency	Old text agency field; null padded to 16 bytes, no longer used
89-94	5 + 1 null	Type	Text field; null padded to 6 bytes

Video Format Field: This field provides information about both the video format and the aspect ratio of the indicated clip.

Table 2-54: Video Format Table

Value	Video Format
0x00	Not supported
0x01	Not supported
0x02	MPEG2 4:2:0
0x03	MPEG2 4:2:2
0x04	DVCAM version of DV25
0x05	DVCPRO version of DV25
0x06	DV50, DVCPRO HD, or DNxHD
0x07	Uncompressed KRGB 8 bits
0x08	Uncompressed K 16 bits
0x09	IMX (MPEG 4:2:2), CBG (constrained bytes per GOP)
0x0A	H.264 4:2:0
0x0B	H.264 4:2:2
0x0C	H.264 4:2:0 CBG (e.g. AVC-Intra class 50)
0x0D	H.264 4:2:2 CBG (e.g. AVC-Intra class 100)
0x0E	Reserved
0x0F	Audio-only (no video)

For value 0x06, what determines the actual format of the three listed is the video bit rate of the clip. If the clip is 50 Mb/s, it's DV50. If 100 Mb/s, it's DVCPRO HD. If it's more than 100 Mb/s, it's DNxHD.

The Video Format field also includes an indication of the clip's aspect ratio. A value of 0x10 is added to the above values to indicate an SD clip is 16:9. All HD clips are assumed to be 16:9 even if the added value is not included in the returned data.

It is possible for a controller to override the aspect ratio defined here by setting a flag in Field 19 of the extended fields. See the description for Field 19 using the **Get Extended Field (C9 C3)** command.

Video Bit Rate Fields: As of the Nexio 6.0 Software Release, the Exact Video Bit Rate field was added in data position 20 - 21 to more accurately define the exact bit rate of the video portion of a clip. If this field is currently null, then the controller should use the original Video Bit Rate field at data position 11.

Because the original Video Bit Rate field only stores one byte of data, it was not possible to report all values of bit rates possible up to 300 Mb/s. As a result video bit rate values above 50 are expressed in a progressive slope, as seen in the following table.

Table 2-55: Video Bit Rate Table

Byte Value	Calculation
≤ 50	Bit rate = Value
> 50 ≤ 60	Bit rate = (5 * Value) – 200
> 60	Bit rate = (10 * Value) – 500

Date Fields: The data in the **Record Date** and **Kill Date** fields include 16-bits, which break down as follows:

Table 2-56: Date Format Bitmap

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FIELD	Day					Month				Year						

The maximum value of these date fields is the last day of the year 2027.

* Response format violates the convention of data count embedded in LSN of CMD-1.

CF 44 — Set ID Metadata

Syntax: **CF 44 ID BITMAP DATA0-58***
ID: 8-byte ID handle
BITMAP: 1-byte unsigned integer, based on bitmap/data table below
DATA: 59 bytes, based on bitmap/data table below

Response: **10 01** or
D0 00 if ID handle not found

Description: This command allows the controller to update fields in the normal set of metadata associated with a specified ID handle. The **BITMAP** byte indicates which field or fields to update. While any combination of bits is allowed, all fields in DATA0-58 must be sent with the command. Unused fields should be set to zero. None of these fields is enabled for Unicode text. If the ID is not found on disk, the server returns **D0 00**.

Table 2-57: ID Metadata Update Bitmap

Data#	Size	Bitmap	Field	Description
0-3	4	0x01	Start	Start Timecode of ID in format of "FF.SS.MM.HH". If the timecode is NTSC drop-frame, the frames unit is padded by 40 frames where "40 00 00 01" represents a start time of exactly one hour.
4-6	3	---	---	Reserved, set all bytes to zero
7-11	5	0x80	Procamp Values	Procamp values for video gain, video setup, chroma gain, and hue as shown in table below
12-13	2	0x08	Kill Date	The date the ID is flagged to expire, in format of LSB first, based on Table 19 in Get ID Metadata (C8 4A) command above
14-38	25	0x10	Old Description1	Old text description field, null padded to 25 bytes, no longer used
39-53	15	0x20	Old Agency[1]	Old text agency field, null padded to 15 bytes, no longer used
54-58	5	0x40	Type	Text type field, null padded to 5 bytes

1 – The Old Description and Old Agency fields are only available in legacy systems configured for 8- and 32-byte ID systems and when the system does not have the extended fields feature enabled.

Procamp Field: The four procamp variables, each represented MSB first, must be set at the same time in a specific order as detailed in the following table:

Table 2-58: Procamp Data Order and Values

Data Byte	Procamp Video Setting	Min Value	Min Setting	Default Setting	Max Value	Max Setting
7	Video Gain (Luma)	-15.0	-99	0	+5.3	99
8	Video Setup	-200mV	-127	0	+200mV	127
9	Chroma Gain	-Inf dB	-100	0	+6.0	100
10-11	Hue, MSB first	0	360	180	360	0

The settings listed above are represented in Decimal values.

The new values set by the controller will take effect after a variable delay that depends on network activity (typically less than 100 ms). Therefore, this command is not appropriate for tracking a slider's movement. To see near-immediate results of procamp changes, use the **Set Manual Procamps (C6 CA)** command with a loaded ID.

* Command format violates the convention of data count embedded in LSN of CMD-1.

C9 C3 — Get Extended Field

Syntax: **C9 C3 ID NUMBER**
ID: 8-byte ID handle
NUMBER: 1-byte unsigned integer, indicating the number of the extended field desired, as listed in the below table

Response: **DF C3 BC DATA** or
D0 C3 if ID handle not found

BC: 1-byte unsigned integer, total amount of data to follow
DATA: Data representing contents of field, if any, in Unicode format when enabled

Description: This command returns the data present in one of the fields in the extended set of metadata associated with the specified ID handle. Only one field can be returned with each use of the command. Which field is desired is based on its field **NUMBER** as indicated in the table below.

If the specified ID handle is not found, the server returns **D0 C3**.

These extended fields are not available in legacy systems that are configured for 8-byte or 32-byte (ANSI) IDs.

Fields below shaded in gray are modifiable using the **Set Extended Field (CF CC)** command.

Table 2-59: Extended Set of Metadata Field Numbers

Number	Field	Description	Max Size
0	Record Date/Time	A FILETIME structure representing the time and date stamp of when the media was recorded	8
1	Codec Where Recorded	The video server name and channel where the media was recorded	19
2	Source 8-byte ID Handle	Used only in Nexio Browse 1.0 and older low-res systems to link a low-res clip to its matching high-res clip's 8-byte ID handle.	8
3	UMID	Unique Media Identifier	64
4	Video Info	Extended video format and frame rate in a bit-wise algorithm of 128 bits (see detail below)	16
5	Source Video Parameters	Used only in Nexio Browse 1.0 and older low-res systems to indicate the video format, bit rate, and GOP of a low-res clip's matching high-res clip	4
6	GUID	128-bit global unique identifier	16
7	User Name	UNICODE-stored field indicating the individual logged in to the Nexio system at record time	64
8	Department	UNICODE-stored field typically indicating the department of the individual logged in at record time	64
9	Title	UNICODE-stored field typically indicating a title associated with the clip	64
10	Reserved	Reserved	12
11	Link	UNICODE-stored field for use with ClipSync feature that stores the ID of media linked to the current ID for in sync dual playback	64
12	Description	UNICODE-stored 120-character user entry text field for storing detail about media	240
13	Agency	UNICODE-stored 15-character user entry text field for storing detail about media	30
14	User-definable Field #1	UNICODE-stored 25-character user entry field	50
15	User-definable Field #2	UNICODE-stored 25-character user entry field	50
16	User-definable Field #3	UNICODE-stored 25-character user entry field	50

Number	Field	Description	Max Size
17	User-definable Field #4	UNICODE-stored 25-character user entry field	50
18	External Controller UID	A field reserved for external controllers needing a place to store their own unique identifiers	16
19	Video ARC	A series of seven 2-byte values defining a clip's video aspect ratio conversion when loaded into a channel of opposite resolution. This field also sets whether to override existing AFD data and whether to override an SD clip's native aspect ratio. (see detail below)	14
20	Modified Timestamp	A FILETIME structure representing the time and date stamp when the media was last modified. See below for list of protocol commands which trigger this timestamp.	8
21	Video QA Status	Results of video quality analysis as performed by the QuiC software application (see detail below)	2
22	User Segments In Use	Used for managing the information stored in user data segments, part of the DTA modes. The 2 bytes represent a bitmask of "in use" user data slots	2
23	Audio Track Compression Info	A series of 1-byte data per audio track (up to 32 tracks) to indicate each track's compression type and channel count (see detail below)	32
24	Audio Track Tag Info	A series of 1-byte data per audio track (up to 32 tracks) to identify the content of each track using special audio tags (See detail below)	32

Unicode Note: To make full use of the Unicode text fields available in this command, the controller must have set the Unicode bit in the **Set Machine Control Options (C8 B1)** command.

Field 4: The **Video Info** field packs into 128 bits a description of a clip's special video characteristics, including its resolution, frame rate, and scan types. It's a bitwise algorithm in which the first 96 bits are made up of the video format code used in several other commands to identify these attributes. The remaining attributes provide an extra level of detail.

The LLM records this information in the Video Info field as valid video data arrives on the system.

What follows is an enum definition for parsing the 128-bit data represented LSB first:

```
#define VIDEO_INFO_FORMATS
(VERTICAL_SIZE_CODES*SCAN_TYPES*FRAME_RATE_CODES)

enum VIDEO_INFO_BIT
{
      // Format bits, bit # = formatCode, where
      // formatCode = (verticalsizeCode * SCAN_TYPES +
      scanType)
      //              * FRAME_RATE_CODES  + frameRateCode;
   VIDEO_INFO_FORMAT_BASE=0,                 // 0..95
```

```
      // Picture structure bits:
      // if (Mpeg) pictureStructure!=PICTURE_STRUCTURE_FRAME
   VIDEO_INFO_PICTURE_FIELD=VIDEO_INFO_FORMAT_BASE+VIDEO_INFO_
   FORMATS,    // 96
      // else if (Mpeg) progressiveFrame==0
   VIDEO_INFO_PICTURE_FRAME_INTERLACED,
         // 97
      // else if (Mpeg) repeatFirstField==0
   VIDEO_INFO_PICTURE_FRAME_PROGRESSIVE,
         // 98
      // else (Mpeg pull-down frame)
   VIDEO_INFO_PICTURE_FRAME_PROGRESSIVE_REPEAT,
         // 99

         // Chroma format bits (see table below):
   VIDEO_INFO_CHROMA_420,
         // 100
   VIDEO_INFO_CHROMA_422,
         // 101
   VIDEO_INFO_CHROMA_311,
         // 102

         // Pixel aspect bits, HD only (see table below):
   VIDEO_INFO_PIXEL_ASPECT_1_1,
         // 103
   VIDEO_INFO_PIXEL_ASPECT_4_3,
         // 104

         // Video encoded in bits per component
   VIDEO_INFO_8_BITS_PER_COMPONENT,
         // 105
   VIDEO_INFO_10_BITS_PER_COMPONENT,
         // 106

   VIDEO_INFO_BITS=128
};

where:

enum VERTICAL_SIZE_CODE          // unofficial
{
   VERTICAL_SIZE_CODE_480=0,
   VERTICAL_SIZE_CODE_512,
   VERTICAL_SIZE_CODE_576,
   VERTICAL_SIZE_CODE_608,
   VERTICAL_SIZE_CODE_720,
   VERTICAL_SIZE_CODE_1080,
   VERTICAL_SIZE_CODES
};

enum SCAN_TYPE             // = (Mpeg) progressive_sequence
{
   SCAN_TYPE_INTERLACED=0,
   SCAN_TYPE_PROGRESSIVE=1,
   SCAN_TYPES
};

enum FRAME_RATE_CODE       // unofficial = (Mpeg)
frame_rate_code - 1
{
   FRAME_RATE_CODE_24M=0,
   FRAME_RATE_CODE_24,
```

```
    FRAME_RATE_CODE_25,
    FRAME_RATE_CODE_30M,
    FRAME_RATE_CODE_30,
    FRAME_RATE_CODE_50,
    FRAME_RATE_CODE_60M,
    FRAME_RATE_CODE_60,
    FRAME_RATE_CODES
};
```

To help decode the video format code value retrieved from the **Video Info** field's first 96 bits (bits 0 – 95), use the following reverse formula:

```
frameRateCode     = formatCode % FRAME_RATE_CODES;
temp              = formatCode / FRAME_RATE_CODES;
scanType          = temp % SCAN_TYPES;
verticalsizeCode  = temp / SCAN_TYPES;
```

More information on the Video Format Code can be found earlier in this chapter in the section titled *Special Protocol Information*.

As for the bits beyond the first 96, they typically indicate one thing or another. For example, bits 97 and 98 indicate interlaced or progressive line scanning.

The chroma format bits, 100 – 102, are determined with the help of a clip's video format and in some cases its bit rate:

Table 2-60: Chroma Format Bits

Video Format	Bit Rate	Chroma
MPEG2 MpMl	n/a	4:2:0
MPEG2 MpPl	n/a	4:2:2
DVCPRO encoded as HDCAM	140	3:1:1
DVCPRO50 encoded as DVCPRO HD	100	4:2:2

The pixel aspect bits, 103 and 104, only apply to HD media and are based on the media's pixels. XDCAM HD is the most common format using 4:3 pixels.

Table 2-61: Pixel Aspect Bits

Bitmap Sizes	Width/Height
1920x1080	1:1
1440x1080	4:3
1280x720	1:1
960x720	4:3

For backward compatibility, if neither bit is set for HD media, then the controller should assume a pixel aspect of 1:1.

Meanwhile, bits 105 and 106 are valid only as of the Nexio 6.0 software release. Prior to that all encoded media was assumed to be 8 bits per component. Now there's the option for 10 bits per component video, such as AVC-I and some DNxHD bit rates.

Field 19: The Video ARC (aspect ratio conversion) field helps define several different characteristics associated with the clip in the area of aspect ratio and aspect ratio conversions.

The primary function of the field is to define the ARC applied to the clip when it is loaded for play out in a channel configured for the clip's opposite aspect ratio. For example, when a 4:3 clip is loaded into a channel configured for 16:9 play out, the server can convert the clip's display in the different aspect ratio according to the parameters defined in this field.

In addition, this field can determine if the clip's ARC setting should override any ARC definitions already established within the media stream of the clip. And lastly, this field can change the aspect ratio characteristics of an SD clip.

The Video ARC field is 14 bytes long, but most controllers will focus on the first two bytes, represented in little endian (LSB first), which sets the ARC mode for the clip plus a few extra properties.

Table 2-62: ARC Mode

Value	Description	Conversion
0x0000	Default	Both
0x0001	Linear or Anamorphic	Both
0x0002	Letter Box	16:9 to 4:3 only
0x0003	Pillar Box	4:3 to 16:9 only
0x0004	Center Cut	16:9 to 4:3 only
0x0005	Middle Cut	4:3 to 16:9 only
0x0006	14:9 for 4:3 source	4:3 to 16:9 only
0x0007	14:9 for 16:9 source	16:9 to 4:3 only
0x0008	21:9 for 4:3 source	4:3 to 16:9 only
0x0009	21:9 for 16:9 source	16:9 to 4:3 only
0x000A	14:9 Center Cut Alternate for 16:9 source	16:9 to 4:3 only
0x000B	14:9 Pillar Box Alternate for 4:3 source	4:3 to 16:9 only
0x000C	14:9 Letter Box Alternate for 16:9 source	16:9 to 4:3 only
0x000D	14:9 Middle Cut Alternate for 4:3 source	4:3 to 16:9 only
0x000E	Alternate Letter Box for 16:9 source	16:9 to 4:3 only
0x000F	Alternate Middle Cut for 4:3 source	4:3 to 16:9 only
0x0010	Letter Box Top Aligned	16:9 to 4:3 only
0x0011	Top Aligned 16:9 frame	4:3 to 16:9 only
0x0012	Top-Aligned 14:9 frame	4:3 to 16:9 only
Added Bits		
0x2000	Forces SD clips to be 4:3	
0x4000	Forces SD clips to be 16:9	
0x8000	Overrides any AFD data present	

The first group of values represents the aspect ratio conversion to be applied to the clip as it is played to air in a channel whose aspect ratio is opposite to that of the clip. These settings also affect the AFD data inserted into the clips on play out if the clip had no AFD data to begin with or the AFD override setting is enabled.

It's important that the ARC applied to a clip match its current aspect ratio or else the setting will have no effect and the AFD data output by the server will be wrong. For example, a 4:3 clip should not be assigned the Letter Box ARC setting.

The second group of values are additive to the first group. That is, if the extra settings are desired, the values will be added to the selected ARC mode applied to the clip.

No more than one of the first two values, 0x2000 and 0x4000, may be selected for any one clip. These only apply to SD clips and have the resulting consequence of forcing the clip to be one aspect ratio or the other.

For example, an SD clip may have been ingested into a channel mistakenly configured for the 16:9 aspect ratio. As it records, the internal metadata associated with the clip gets stamped with the 16:9 flag. It's possible to set the 0x2000 flag in the Video ARC field to change the clip back to a 4:3 aspect ratio. This flag does not apply to HD clips, which are always recorded with the 16:9 aspect ratio.

The final value, 0x8000, has the effect of overriding any AFD data that might be embedded within a clip. By default the Nexio server will apply the ARC as defined within the Active Format Description (AFD). So even if an ARC is defined by the use of this field, it will be ignored on play out as long as the clip already contains AFD data and the 0x8000 flag has not been enabled. Turn on this flag and the server will make use of the ARC as defined in this field.

The following table offers a few samples of ARC mode settings based on the first two bytes of the Video ARC field. Remember the bytes are represented LSB first.

Table 2-63: Sample ARC Settings

Value	Description
0x0200	Letter Box ARC applied to a 16:9 clip in a 4:3 channel. AFD data in the clip would override this setting.
0x0020	Default ARC applied to an SD clip that is using the aspect ratio override to change the 16:9 clip to 4:3
0x0320	Pillar Box ARC applied to an SD clip that is using the aspect ratio override to change the 16:9 clip to 4:3
0x0280	Letter Box ARC applied to a 16:9 clip in a 4:3 channel. This setting overrides any AFD data that may be present in the clip.
0x04C0	Center Cut ARC applied to an SD clip whose original 4:3 aspect ratio has been changed to 16:9 and whose ARC setting overrides any AFD data in the clip.

The last 12 bytes of the Video ARC are normally set to their default values. They are typically only used when the controller requires an ARC that is not among the ones pre-defined. In such a case, the controller should set the ARC mode to 1, or Linear, and then apply the remaining 12 bytes based on the following table:

Table 2-64: Video ARC Values

Procamp Video Setting	Bytes	Min Value	Min Setting	Default Setting	Max Value	Max Setting
Horizontal Size	2	50%	0x0040	0x0080	100%	0x0080
Horizontal Position	2	Currently not supported – default = 0x0000				
Horizontal Crop from left	1	0 pixels	0x00	0x00	255 pixels	0xFF
Horizontal Crop from right	1	0 pixels	0x00	0x00	255 pixels	0xFF
Vertical Size	2	50%	0x0040	0x0080	100%	0x0080

Procamp Video Setting	Bytes	Min Value	Min Setting	Default Setting	Max Value	Max Setting
Vertical Position	2	Currently not supported – default = 0x0000				
Vertical Crop from top	1	0 pixels	0x00	0x00	255 pixels	0xFF
Vertical Crop from bottom	1	0 pixels	0x00	0x00	255 pixels	0xFF

The values in between the minimum and maximum values are linear. And the 2-byte values are represented LSB first.

A controller can use the **Set Manual Video ARC (CF 66)** command to preview possible ARC settings for a clip loaded in a channel. The command makes use of the same set of 14 bytes as described above. The only difference is the **Set Manual Video ARC** command sends the 2-bytes values represented MSB first.

When the controller identifies the desired ARC setting, it can use the **Set Extended Field (CF CC)** command to write the desired values to the ID's field 19 represented LSB first.

The aspect ratios referenced here are the "display" aspect ratio values for clips, not necessarily how the clip is compressed. For example, an XDCAM HD clip with stored pixels of 1440x1080 display as 1920x1080 and is therefore referred to as a 16:9 clip.

Field 20: The **Modified Timestamp** field is updated when any controller uses a protocol command to update the clip. The following is a list of the commands that have an effect on this field:

- Native Protocol
  - CF 44: Set ID Metadata
  - CF CC: Set Extended Field
  - CC 84: Set Special ID Attributes
  - CC B6: Redefine ID In Point
  - CF A3: Redefine ID Timecode Bounds
  - C8 C5: Consolidate ID
  - CF CA: Rename Extended ID
  - 20 02: Record
- VDCP Protocol
  - 00 15: Delete Protect ID
  - 00 16: Undelete Protect ID
  - 10 02: Record
  - A0 1D: Rename ID

Field 21: The **Video QA Status** field is generated as a result of the Imagine Communications QuiC media analysis tools. The field is a bit-wise field indicating the ID's current state according to this table:

Table 2-65: Video QA Status

Bits	Meaning
0	None
1	Passed
2	Failed

3	Analyzing
4	Pending

All clips arrive in the system with an initial state of “0”. If a QuiC media analyzer is attached to the server system it will monitor the system for new media and analyze it and report back information to this field

Field 23: Using the Audio Track Router interface introduced with the Nexio 6.0 Software Release, it is possible define existing audio tracks and attach additional audio tracks to recorded media so that they can be routed on play out to different tracks depending on audio compression and language needs. Extended Fields 23 and 24 help manage the information associated with each track.

The Audio Track Compression Info field reports the compression information associated with every possible audio track in a clip. The structure of the field is broken down into one byte per audio track up to a maximum of 32.

Table 2-66: Audio Track Compression Info

Bit	Description
0	Indicates a track containing compressed audio which requires being stored within a pair of tracks (Neural/DTS, Dolby Digital/AC3 and Dolby-E)
1 – 3	Indicates the number of actual audio tracks stored within the compressed audio track. This information is mandatory for all compressed audio. PCM = 0
4 – 7	Indicates the audio format of the track: 0 = PCM 1 = Neural/DTS 2 = Dolby Digital/AC3 3 = Dolby E 4 = MPEG-1 Layer I and Layer II 6 = AAC/HE-AAC/HE-AAC v2

All 32 bytes are recorded and reported for every clip even if the clip is currently storing fewer audio tracks. Unused audio tracks should be left at the default value of 0x00.

For more information on the implementation of this field, please see the section above on the Audio Track Router.

Field 23 is supported only on the Nexio AMP and Volt servers as of the Nexio 6.0 Software Release.

Field 24: The Audio Track Tag Info field is the second field used to help define information about a clip’s audio tracks as part of the Audio Track Router interface introduced with the Nexio 6.0 Software Release.

The Audio Track Tag Info field can store special data that might identify the language source of the audio and its audio type. For each track there are 256 potential tags to choose from. For more information on the implementation of this field, please see the section above on the Audio Track Router.

CF CC — Set Extended Field

Syntax: **CF CC BC ID NUMBER DATA**
BC: 1-byte unsigned integer, total amount of data to follow
ID: 8-byte ID handle
NUMBER: 1-byte unsigned integer, indicating the number in hex of the extended field desired, as listed in the table shown in the **Get Extended Field from ID (C9 C3)** command
DATA: Data to replace the current contents of the indicated field

Response: **D4 CC TID**
TID: 4-byte unsigned integer, MSB first, identifying a Transaction ID associated with the command.

Description: This command allows the controller to update a field in the extended set of metadata associated with a specified ID handle. Only one field can be updated with each use of the command and only some of the field numbers are modifiable, as indicated in the shaded areas of the table shown in the **Get Extended Field from ID (C9 C3)** command. This command replaces any metadata currently in the specified field with the new data that follows.

It is the responsibility of non-Unicode controllers to allow for the automatic Unicode conversion of these fields by sending no more than half the number of bytes associated with each field.

It is also the responsibility of the controller to monitor the transaction status of the pending command using the **Get Transaction Status (C4 B3)** command at greater than 50 ms intervals until the transaction returns a success or failure.

Unicode Note: To make use of Unicode text in this command, the controller must have set the Unicode bit in the **Set Machine Control Options (C8 B1)** command.

CF CB — Find First ID Whose Extended Field Matches Criteria

Syntax: **CF CB BC NUMBER CRITERIA**
BC: 1-byte unsigned integer, total amount of data to follow
NUMBER: 1-byte unsigned integer, based on the extended field number in hex to be searched
CRITERIA: Data representing the criteria to search the extended field for a matching ID

Response: **D8 CB ID** or
D0 CB if no matching ID handle is found
ID: 8-byte ID handle

Description: This command allows a controller to search through a specified extended field for a specified search string. Refer to the table shown in the **Get Extended Field from ID (C9 C3)** command to determine the field number desired. Only the first matching ID handle will be returned by this command. Other matching ID handles can be retrieved through the **Find Next ID Whose Extended Field Matches Criteria (C0 CB)** command.

If no matching ID handle is found, the server returns **D0 CB**.

Unicode Note: To make use of Unicode text in this command, the controller must have set the Unicode bit in the **Set Machine Control Options (C8 B1)** command.

C0 CB Find Next ID Whose Extended Field Matches Criteria

Syntax: **C0 CB**

Response: **D8 CB ID** or
D0 CB if no matching ID handle is found

ID: 8-byte ID handle

Description: This command returns the next matching ID handle based on the extended field search begun with the **Find First ID Whose Extended Field Matches Criteria (CF CB)** command. Only one ID will be returned with this command. To continue to receive matching ID handles, repeat this command until the server returns **D0 CB**, the indication that there are no more matching ID handles.

Unicode Note: To make use of Unicode text in this command, the controller must have set the Unicode bit in the **Set Machine Control Options (C8 B1)** command.

C8 84 Get Special ID Attributes

Syntax: **C8 84 ID**

ID: 8-byte ID handle

Response: **D2 84 MASK** or
D0 84 if no matching ID handle is found

MASK: 2 byte, unsigned integer, MSB first

Description: This command returns the special attributes of the specified ID handle. See the following table for the interpretation of the return data. Attributes are cumulative when an ID has more than one matching property.

If the ID is not found, the server returns **D0 84**.

Table 2-67: ID Attributes and Descriptions

Value	Attribute	Description
0x0001	Delete protected	The ID is protected from deletion until this attribute is unset
0x0002	Sliding ID	Indicates a circular recording in which the length is preset and material is automatically deleted from the front as it is recorded on the back
0x0004	Macro ID	Indicates the media is a timeline made up of pointers to other pieces of media
0x0008	Alpha File	Indicates a special file representing a character generator graphic
0x0010	Project File	Indicates a special file used by editing applications to represent a container of timelines, always with the Invisible attribute set
0x0020	Invisible ID	Indicates the file is hidden from the normal media database
0x0040	Purged ID	Indicates a consolidated piece of media in which the only area of the original clip that is retained is that area protected by subclips

0x0080	Reference ID	Indicates a special hidden pointer file used by editing applications to protect the source clip from which the edit came.
0x0100	Looping ID	Indicates a special property in which the clip will play in a continuous loop if loaded in a channel
0x0200	Not Ready to Play	Indicates the first audio buffer of the media has not yet been written to disk. A controller should wait to load and play any clips with this attribute set.
0x0400	Not Ready to Transfer	Indicates the ID is currently in record or being written to disk
0x0800	Not Ready to Archive	Indicates the ID is currently in record or being written to disk
0x1000	Transfer in Progress	Indicates the ID is currently being transferred or imported into the system

CC 84 — Set Special ID Attributes

Syntax: **CC 84 ID MASK VALUE**
ID: 8-byte ID handle
MASK: 2 byte, unsigned integer, MSB first
VALUE: 2 byte, unsigned integer, MSB first

Response: **10 01** or
D0 00 if ID handle not found

Description: This command sets an ID's special attributes based on the specified **MASK**. The **MASK** indicates which attributes will be affected; the **VALUE** indicates what value to set the attributes to. Specify a value of 0x0000 to reset the attribute. See the table below for the attributes that are settable.

If the **ID** is not found, **D0 00** is returned.

Table 2-68: Settable ID Attributes

Mask	Value	Attribute	Description
0x0000	0x0000	Normal ID	An ID that has no special attributes.
0x0001	0x0001	Delete Protected	An ID that is protected from deletion.
0x0002	0x0002	Sliding ID	An ID that is a circular recording.
0x0100	0x0100	Looping ID	An ID that will continuously playback in a loop indefinitely.

For example to set the Delete Protected attribute, the controller would send the following after the ID:

```
00 01 00 01
```

To unset the attribute, send the following:

```
00 01 00 00
```

It is also possible to set and unset multiple attributes in one command based on the bit mask.

A clip with the sliding ID attribute requires a preset in and out point to define the duration of the circular recording.

When a clip with the looping attribute is set, the server automatically auto-loads the same clip in the preview channel after any cue up command that involves the clip. The looping play back can be cancelled by the following commands:

- **Set Auto Mode On (40 41)**
- **Set Auto Mode Off (40 40)**
- **Reset Preview In Point (A0 06)**
- **Preset Preview In Point (Ax 04)** with another clip

Also, looping will not be enabled if the clip is loaded by the **Cue Up to Record (Ax 02)** command.

C8 CA — Get Audio Track Information

Syntax: **C8 CA ID**
ID: 8-byte ID handle

Response: **D6 CA SIZE PRECISION TRACKS** or
D0 CA if no valid audio data is found
SIZE: 1-byte unsigned integer indicating audio sample size in bits
PRECISION: 1-byte unsigned integer indicating audio sample precision in bits
TRACKS: 32-bit mask, represented MSB first, indicating audio tracks enabled

Description: This command provides information about the specified clip's audio tracks. Current support requires the audio sample **SIZE** and **PRECISION** (packed format) to be the same value.

The number of audio tracks is represented in a bit mask where the bit for each enabled audio track is set high. For example, 4 audio tracks represented in bits would be "00001111", or 0x0F.

If the ID is not found or it is an existing ID with no audio data such as an empty ID, the server returns **D0 CA**.

C6 CA — Set Manual Procamps

Syntax: **C6 CA DEVICE VIDEOGAIN VIDEOSETUP CHROMAGAIN HUE**
DEVICE: 1-byte unsigned integer indicating video stream affected
VIDEOGAIN: 1-byte unsigned integer
VIDEOSETUP: 1-byte unsigned integer
CHROMAGAIN: 1-byte unsigned integer
HUE: 2-byte unsigned integer, MSB first

Response: **10 01**

Description: This command allows a controller to interactively adjust the procamp settings of a loaded ID before deciding whether to write those values to the ID's metadata. There must be a clip loaded in the selected channel and the changes made will have an immediate effect on the clip with a playback-specific delay.

The **DEVICE** parameter is the number of the first defined video stream in the channel, which for normal media would be "0". The ranges and default values of the

remaining procamp parameters are the same as in the **Set ID Metadata (CF 44)** command.

The controller can send this command repeatedly as adjustments are made and the video output of the selected channel will follow those changes. When the desired effect or changes are reached, the controller can then use the **Set ID Metadata (CF 44)** command to write the final values in the ID. The speed of this command depends on network activity. The controller should verify the changes were received by polling the server with the **Get ID Metadata (C8 4A)** command at greater than 50 millisecond intervals.

When the changes are confirmed, the controller should send the **Reset Manual Procamps (C1 CA)** command to reset the channel back to using the preset procamp values of the loaded ID.

C1 CA — Reset Manual Procamps

Syntax: **C1 CA DEVICE**
DEVICE: 1-byte unsigned integer indicating video stream affected

Response: **10 01**

Description: This command disables the interactive procamp setting mode enabled by the **Set Manual Procamp (C6 CA)** command. It resets the channel back to using the currently loaded ID's procamp values as stored in the ID's metadata.

The **DEVICE** parameter must match the previously set parameter by the **Set Manual Procamp** command, but again for most clips this would equal "0".

If procamp changes were made to the loaded clip and those changes were written to the ID's metadata using the **Set ID Metadata (CF 44)** command, the controller should ensure the changes took effect by polling the **Get ID Metadata (C8 4A)** command at greater than 50 millisecond intervals.

This confirmation of metadata will insure that no spikes occur in the video when the channel is reset.

CF 66 — Set Manual Video ARC

Syntax: **CF 66 DEVICE MODE HSIZE HPOS HCROP VSIZE VPOS VCROP**
DEVICE: 1-byte unsigned integer indicating video stream affected
MODE: 2-byte unsigned integer, MSB first, representing the ARC_MODE
HSIZE: 2-byte unsigned integer, MSB first
HPOS: 2-byte unsigned integer, MSB first
HCROP: 2 1-byte unsigned integers
VSIZE: 2-byte unsigned integer, MSB first
VPOS: 2-byte unsigned integer, MSB first
VCROP: 2 1-byte unsigned integers

Response: **10 01**

Description: This command allows a controller to interactively adjust the video aspect ratio conversion of a loaded ID before deciding whether to write those values to the ID's

metadata. There must be a clip loaded in the selected channel and that clip must have the opposite resolution of the channel. For example, this command will only have an effect if an SD clip is loaded in an HD channel or an HD clip is loaded in an SD channel.

As the controller adjusts the ARC settings, they will have an immediate effect on the clip with a playback-specific delay. The **DEVICE** parameter is the number of the first defined video stream in the channel, which for normal media would be "0".

For a list of the possible values for this command, refer to the **Get Extended Field (C9 C3)** command, Field 19. The main difference with using this command is that the 2-byte variables here are represented MSB first. In addition, the aspect ratio override and AFD override settings do not apply here.

The controller can send the **Set Manual Video Arc** command repeatedly as adjustments are made and the video output of the selected channel will follow those changes. When the desired effect or changes are reached, the controller can then use the **Set Extended Field (CF 44)** command for extended field 19 to store the final values for the ID. However, when writing those values for the field, the controller needs to be sure to write the values represented in little endian (LSB first).

The speed of the **Set Extended Field** command depends on network activity. The controller should verify the changes were received by using the **Get Transaction Status (C4 B3)** command at greater than 50 ms intervals until the transaction returns a success or failure.

When the changes are confirmed, the controller should send the **Reset Manual Video ARC (C1 66)** command to reset the channel back to using the preset video ARC values of the loaded ID.

C1 66 — Reset Manual Video ARC

Syntax: **C1 66 DEVICE**
DEVICE: 1-byte unsigned integer indicating video stream affected

Response: **10 01**

Description: This command disables the interactive video ARC setting mode enabled by the **Set Manual Video ARC (CF 66)** command. It resets the channel back to using the currently loaded ID's video ARC setting as stored in the ID's metadata.

The **DEVICE** parameter must match the previously set parameter in the **Set Manual Video ARC** command, but again for most clips this would equal "0".

If video ARC changes were made to the loaded clip and those changes were written to the ID's metadata using the **Set Extended Field (CF 44)** command, the controller should ensure the changes took effect by using the **Get Transaction Status (C4 B3)** command at greater than 50 ms intervals until the transaction returns a success or failure.

This confirmation of metadata will ensure that transients occur in the video.

C9 C9 — Get Slow Access Data Tag

Syntax: **C9 C9 MODE ID**
MODE: Bitmap indicating a particular storage space associated with the ID handle
ID: 8-byte ID handle

Response: **D4 C9 DATATAG** or
D0 C9 if ID handle is not found
DATATAG: 4-byte unsigned integer, MSB first

Description: This command returns a specially formatted "data tag" associated with the specified ID handle and one of its special storage areas within the LLM system. The purpose of this data tag is to allow controllers quick access to information about the LLM's slow access storage areas, sometimes called user data areas. These storage areas are accessed through the **Get Slow Access Metadata (CF A0)** and **Set Slow Access Metadata (CF A1)** commands.

It should be understood that the Slow Access commands can take up to 0.5 seconds to read from or write to the LLM. By using this command to retrieve data tags associated with each storage area, a controller can quickly determine if there is anything stored in the storage areas and/or if those storage areas have changed since the controller last checked. The data tag changes every time the user data area is written to. The special value “0” indicates no user data is present. It is the responsibility of the controller to store the previous data tags and compare them against any newly returned data tags.

This command uses the same MODE as seen in the **Slow Access** commands to determine which storage area is to be referenced. The format of MODE is based on six bits as seen in the table shown with the **Get Slow Access Metadata (CF A0)** command.

Nexio system applications such as NXOS and VelocityNX currently use several of the available compartments for storage of information. Most controllers should not need this special information.

CF A0 — Get Slow Access Metadata

Syntax: **CF A0 BC DTATYPE DTASPEC MODE ID ***
BC: 1-byte unsigned integer, total amount of data to follow. For the following described uses of this command, this value will always be **10**.
DTATYPE: 1-byte unsigned integer. For the following described uses of this command, this value will always be **01**.
DTASPEC: 6 bytes indicating the DTA specification. For the following described uses of this command, the DTA spec represents in hex the server’s IP address as the first 4 bytes, MSB first, and the server’s TCP port as the last 2 bytes, MSB first.
MODE: Bitmap indicating a particular storage space associated with the ID handle.
ID: 8-byte ID handle

Response: **D4 A0 TID**
TID: 4-byte unsigned integer, MSB first, identifying a Transaction ID associated with the command.

Description: One of the purposes of this command is to retrieve metadata or files stored in special user data segments on the Nexio SAN. These storage areas are useful for controllers wishing to store a lot of data associated with a particular ID.

It should be understood that this command can take up to 0.5 seconds to read from the LLM. Controllers should first use the **Get Slow Access Data Tag (C9 C9)** command to quickly confirm if there is any data to be read and/or if that data has changed.

Prior to issuing this command, the controller needs to create a server side SOCK_STREAM (socket(), bind(), listen()), pass its coordinates for the server IP address and TCP port as the DTA specification, and prepare to accept a connection (accept()). The LLM data agent will connect as a client to the specified socket. If the connection cannot be established the transaction will end in TRANSACTION_CONNECTION_FAILED. Other than that, the connection will be established even if the transaction ends in TRANSACTION_NO_FILE or other errors, and then it will be closed.

The MODE parameter for this command determines which of the many available storage areas is to be read from. There are 16 different user data storage areas that can be associated with an ID.

Table 2-69: Get Slow Access Mode Bitmap

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Do not close socket	0	Data area represented in 6 bits: only supported values in their decimal equivalent are 16 - 31					

Currently the Nexio system uses several storage areas for internal uses. Controllers should avoid using user data segments 16 – 24 and 30 - 31.

If the “Do not close socket” bit is set to “0”, the LLM will automatically close the connection after all the data has been sent, at the first unrecoverable read error, if the ID handle does not exist, or if the storage area has been modified by a concurrent write. The controller can close the connection at any time to end the transfer.

If the “Do not close socket” bit is set to “1”, the LLM will not close the socket after transfer completion. In such a case, it would be up to the controller to query the transaction status using the **Get Transaction Status (C4 B3)** command in order to detect completion of the transfer. A new read operation can then be initiated on the same data socket (the transaction number will however be different).

There is one limitation to be aware of: If the controller begins this process with the “Do not close socket” bit set to “1”, it must send one last command to the LLM in which the “Do not close socket” bit is set to “0” before it can close the socket. This can be a dummy command with ID=0x00... 0x00 and mode=0x42.

* This is a limited description of the **Read File (CF A0)** command, not incorporating several advanced methods for storing and retrieving media within the LLM system. This command format violates the convention of data count embedded in LSN of CMD-1.

CF A1 Set Slow Access Metadata

Syntax: **CF A1 BC DTATYPE DTASPEC MODE ID ATTRIBUTES ***

BC: 1-byte unsigned integer, total amount of data to follow. For the following described uses of this command, this value will always be **12**.

DTATYPE: 1-byte unsigned integer. For the following described uses of this command, this value will always be **01**.

DTASPEC: 6 bytes indicating the DTA specification. For the following described uses of this command, the DTA spec represents the server's IP address as the first 4 bytes, MSB first, and the server's TCP port as the last 2 bytes, MSB first.

MODE: Bitmap indicating a particular storage space associated with the ID handle.

ID: 8-byte ID handle

ATTRIBUTES: 2-byte unsigned integer, MSB first, based on the table shown in the **Get Special ID Attributes (C8 84)** command.

Response: **D4 A1 TID**

TID: 4-byte unsigned integer, MSB first, identifying a Transaction ID associated with the command.

Description: One of the purposes of this command is to store metadata or files in special user data segments in the LLM system. These storage areas are useful for controllers wishing to store a lot of data associated with a particular ID.

Prior to issuing this command, the controller needs to create a server side SOCK_STREAM (socket(), bind(), listen()), pass its coordinates for the server IP address and TCP port as the DTA specification, and prepare to accept a connection (accept()). The LLM data agent will connect as a client to the specified socket. If the connection cannot be established, the transaction will end in TRANSACTION_CONNECTION_FAILED. Other than that, the connection will be established even if the transaction ends in TRANSACTION_NO_FILE or other errors, and then it will be closed.

The MODE parameter for this command determines which of the many available storage areas is to be written to. There are 16 different user data storage areas that can be associated with an ID.

Table 2-70: Set Slow Access Mode Bitmap

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Do not truncate	Data area represented in 7 bits: only supported values in their decimal equivalent are 16 - 31						

Currently the Nexio system uses several storage areas for internal uses. Controllers should avoid using user data segments 16 – 20 and 31.

If the "Do not truncate" bit is set to "0", data of all types in all storage areas will be deleted prior to writing to the LLM. If this bit is set to "1", existing data in storage areas other than the one being written to will be preserved. Existing data in the storage area being written to will be overwritten.

The controller will close the connection after the whole data has been sent. The LLM will close the connection only if the existing ID was write-protected, the disk is full, or an unrecoverable error occurred.

It is up to the controller to query the transaction status using the **Get Transaction Status (C4 B3)** command in order to detect completion of the transfer.

* This is a limited description of the **Write File (CF A1)** command, not incorporating several advanced methods for storing and retrieving media within the LLM system. This command format violates the convention of data count embedded in LSN of CMD-1.

CF Bx — Create Empty Media

Syntax: **CF B5 ID INTC OUTTC** or
CF B9 BC ID INTC OUTTC ATTRIBUTES

BC: 1-byte unsigned integer, indicating total amount of data to follow. For use with the **CF B9** command, this value will be either **10** or **12**.

ID: 8 null bytes to allow the LLM to auto-create a new 8-byte ID handle

INTC: 4-bytes indicating the ID's In Point timecode in the format of FF.SS.MM.HH

OUTTC: 4-bytes indicating the ID's Out Point timecode in the format of FF.SS.MM.HH

ATTRIBUTES: Optional 2-byte unsigned integer, MSB first, based on the table shown in the **Get Special ID Attributes (C8 84)** command.

Response: **D4 B5 TID** or
D4 B9 TID

TID: 4-byte unsigned integer, MSB first, identifying a Transaction ID associated with the command.

Description: This command creates an "empty" ID handle with the specified In and Out Point timecode settings and optionally the attribute bits for the ID. The Out Point should be 1 frame larger than the In Point to properly indicate that the clip is an “empty” ID.

This empty ID is essentially a placeholder ID used in the eventual creation of real media. It is used extensively in the process of creating ID handles for extended IDs.

If the specified ID handle exists and it's not write-protected, all the existing data associated with that ID will be deleted. If the specified ID handle is all zeroes, the server will automatically generate a unique name for the ID handle.

In order to retrieve the new ID handle, the controller should poll the transaction ID status using the **Get Transaction Status (C4 B3)** command at >50 ms intervals until the status changes from TRANSACTION_PENDING. If the status is TRANSACTION_SUCCESS, use the **Get Transaction ID (C4 B4)** command to retrieve the new ID handle.

Drop Frame Timecode Note: In the timecode syntax of FF.SS.MM.HH, the frames unit has a base value of 40 to indicate drop frame timecode in NTSC. PAL timecode and non-drop frame NTSC are “0”-based.

CF C2 — Create Empty Media with Extended ID

Syntax: **CF C2 BC ATTRIBUTES PARENT XID ***

BC: 1-byte unsigned integer, indicating total amount of data to follow.

ATTRIBUTES: 2-byte unsigned integer, MSB first, based on the table in the **Get Special ID Attributes (C8 84)** command.

PARENT: 8-byte ID handle

XID: Extended ID, in Unicode format when enabled

Response: **D4 C2 TID**
TID: 4-byte unsigned integer, MSB first, identifying a Transaction ID associated with the command.

Description: This command creates an "empty" ID with the specified name and attributes. It can also be "attached" to an existing parent clip's ID handle so that the new ID gets assigned to the same disk volume as its parent. This option is required when the newly created ID is intended to be used in a subsequent **Create Subclip (CF 85)** command, as subclips must reside on the same disk as their parent clip. If no parent ID is required for the empty ID, the **PARENT** parameter is sent with all nulls.

The controller can retrieve the new ID handle created for the empty ID by polling the transaction ID status using the **Get Transaction Status (C4 B3)** command at >50 ms intervals until the status changes from TRANSACTION_PENDING. If the status is TRANSACTION_SUCCESS, use the **Get Transaction ID Handle (C4 B4)** command to retrieve the new ID handle. This ID handle will normally be used in subsequent protocol commands for any further access.

If an ID with the same Extended ID already exists, it will be deleted unless it is write protected or has subclips, in which case the transaction will fail with TRANSACTION_WRITE_PROTECTED.

Unicode Note: To make use of Unicode text in this command, the controller must have set the Unicode bit in the **Set Machine Control Options (C8 B1)** command.

* Command format violates the convention of data count embedded in LSN of CMD-1.

CF 85 — Create Subclip

Syntax: **CF 85 CHILD BASE INTC OUTTC ***
CHILD: 8-byte ID handle
BASE: 8-byte ID handle of parent clip from which the CHILD, or subclip, will be generated
INTC: 4-bytes indicating the ID's In Point timecode in the format of FF.SS.MM.HH
OUTTC: 4-bytes indicating the ID's Out Point timecode in the format of FF.SS.MM.HH

Response: **10 01**

Description: This command is used to create a subclip of the specified **BASE** or Parent ID. The new subclip's ID handle will have been determined by a previous **Create Empty Media (CF Bx** or **CF C2)** command.

The new subclip will incorporate the timecode of the parent ID from the specified INTC to OUTTC. The timecode type of the subclip will be derived from INTC, and may differ from that of the Parent ID.

It is up to the controller to specify a valid Parent ID when creating the subclip. Either way, the server responds with an **ACK (10 01)** even if the Parent ID does not exist,.

Drop Frame Timecode Note: In the timecode syntax of FF.SS.MM.HH, the frames unit has a base value of 40 to indicate drop frame timecode in NTSC. PAL timecode and non-drop frame NTSC are “0”-based.

* Command format violates the convention of data count embedded in LSN of CMD-1.

CC B6 — Redefine ID In Point

Syntax: **CC B6 INTC ID**
INTC: 4-bytes indicating the ID's In Point timecode in the format of FF.SS.MM.HH
ID: 8-byte ID handle

Response: **10 01** or
D0 B6 if ID handle is not found

Description: This command redefines the In Point of the specified ID handle. No material is moved or released in the process, however the timecode type will change to that specified in **INTC**. This command is useful in altering the start time of a clip without affecting the ID's length. The controller should think of this command as striping a new timecode track onto existing material.

If the ID handle is not found, the server returns **D0 B6**.

Drop Frame Timecode Note: In the timecode syntax of FF.SS.MM.HH, the frames unit has a base value of 40 to indicate drop frame timecode in NTSC. PAL timecode and non-drop frame NTSC are “0”-based.

CF A3 — Redefine ID Timecode Bounds

Syntax: **CF A3 ID INTC OUTTC ***
ID: 8-byte ID handle
INTC: 4-bytes indicating the ID's In Point timecode in the format of FF.SS.MM.HH
OUTTC: 4-bytes indicating the ID's Out Point timecode in the format of FF.SS.MM.HH

Response: **10 01** or
D0 A3 if ID handle is not found

Description: This command redefines the In Point and Out Point of the specified ID handle. No material is moved or released in the process. This command is most useful for adjusting the timecode bounds of a subclip. However, use this command with caution. If you set the **INTC** and **OUTTC** values outside the actual timecode values of the parent clip, those unrecorded areas will play black to air.

If the ID is not found, the server returns **D0 A3**.

Drop Frame Timecode Note: In the timecode syntax of FF.SS.MM.HH, the frames unit has a base value of 40 to indicate drop frame timecode in NTSC. PAL timecode and non-drop frame NTSC are “0”-based.

* Command format violates the convention of data count embedded in LSN of CMD-1.

A8 10 — Erase ID

Syntax: **A8 10 ID**
ID: 8-byte ID handle

Response: **10 01**

Description: This command is used to erase an ID in the Nexio storage system. The actual video and audio data is not erased, but the areas containing the deleted segments are internally marked as available for recording.

Regardless of whether or not the specified ID exists, the server responds with **ACK (10 01)**. For a command that provides better feedback on the results of a delete, use the **Erase ID with Transaction (C8 4F)** command.

C8 4E — Erase ID

Syntax: **C8 4E ID**
ID: 8-byte ID handle

Response: **10 01** or
11 12 if ID handle is not found

Description: This command is used to erase a specified ID handle from the Nexio storage system. The actual video and audio data is not erased, but the areas containing the deleted segments are internally marked as available for recording. The server returns a **NACK (11 12)** if the ID is not found.

For a command that provides better feedback on the results of a delete, use the **Erase ID with Transaction (C8 4F)** command.

C8 4F — Erase ID with Transaction

Syntax: **C8 4F ID**
ID: 8-byte ID handle

Response: **D4 4F TID**
TID: 4-byte unsigned integer, MSB first, identifying a Transaction ID associated with the command.

Description: This command is used to erase a specified ID handle from the Nexio storage system with transaction tracking turned on. In this way, not only can a controller be sure if the ID is actually deleted, but if not, why it failed.

There are three reasons why an existing ID cannot be deleted:

1. The ID's Delete Protect special attribute is set
2. Subclips have been created from the ID which must be deleted first before the parent ID
3. It is currently loaded in a server channel that has the ID reference protection enabled

In order to retrieve the status of the delete operation, the controller should use the **Get Transaction Status (C4 B3)** command at greater than 50 millisecond intervals until it changes from TRANSACTION_PENDING to one of the following:

- TRANSACTION_SUCCESS
- TRANSACTION_NO_FILE
- TRANSACTION_WRITE_PROTECTED

If the status is TRANSACTION_WRITE_PROTECTED, the delete failed because the spot is protected. The reason why can be determined with the **Get Transferred Field Count (C4 BC)** command. If the field count returned is 0xFFFF,FFFF then the spot is protected by either the Delete Protect attribute or by having subclips.

Any other result indicates the ID is protected by reference, which typically means it is currently loaded in a channel somewhere on the Nexio system. The first two bytes of the field count returned indicates the LLM node protecting the ID and the last two bytes indicate the controller connection number that set the reference.

Use the **Get Transaction ID Handle (C4 B4)** command to retrieve the user assigned controller identification, if set, of the connection protecting the ID.

C8 4D — Query ID as Erasable

Syntax: **C8 4D ID**
ID: 8-byte ID handle

Response: **D4 4D TID**
TID: 4-byte unsigned integer, MSB first, identifying a Transaction ID associated with the command.

Description: This command can be used to determine whether a particular clip is protected from deletion and, if so, how it's protected. In the response to the command, the server returns a transaction ID which can be used to find the erasable status of the clip.

In order to retrieve the status, use the **Get Transaction Status (C4 B3)** command at greater than 50 millisecond intervals until it changes from TRANSACTION_PENDING to one of the following:

- TRANSACTION_SUCCESS
- TRANSACTION_NO_FILE
- TRANSACTION_WRITE_PROTECTED

If the status is TRANSACTION_WRITE_PROTECTED, the clip is protected from deletion. The reason why can be determined with the **Get Transferred Field Count (C4 BC)** command. If the field count returned is 0xFFFF,FFFF then the spot is protected by either the Delete Protect attribute or by having subclips.

Any other result indicates the ID is protected by reference, which typically means it is currently loaded in a channel somewhere on the Nexio system. The first two bytes of the field count returned indicates the LLM node protecting the ID and the last two bytes indicate the controller connection number that set the reference.

Use the **Get Transaction ID Handle (C4 B4)** command to retrieve the user assigned controller identification, if set, of the connection protecting the ID.

During the course of this command interchange, the status of the clip may change. An ID which was found erasable by this operation is not guaranteed to remain erasable and therefore a subsequent erase ID may still fail. To be sure, the controller should attempt the delete with the **Erase ID as Transaction (C8 4F)** command.

A8 11 — Erase Segment (Trim)

Syntax: **A8 11 TC1 TC2**
TC1: 4-bytes indicating a timecode point in the format of FF.SS.MM.HH
TC2: 4-bytes indicating a second timecode point in the format of FF.SS.MM.HH

Response: **10 01**

Description: This command is used to erase or trim a portion of material within the currently loaded ID. **TC1** specifies the starting timecode position and **TC2** specifies the ending timecode position of the portion to erase. All timecode positions within will be erased and de-allocated. The new In Point, Out Point, and duration of the loaded ID will automatically be calculated and reflected in the ID database.

It is typical to use this command incorporating either the starting timecode or the ending timecode of the loaded ID so that it is trimmed from one side or the other. Erasing segments in the middle of the loaded clip will cause black to play back during those timecode points.

Drop Frame Timecode Note: In the timecode syntax of FF.SS.MM.HH, the frames unit has a base value of 40 to indicate drop frame timecode in NTSC. PAL timecode and non-drop frame NTSC are “0”-based.

CF 11 — Erase Segment (Trim) by ID

Syntax: **CF 11 ID TC1 TC2**
ID: 8-byte ID handle
TC1: 4-bytes indicating a timecode point in the format of FF.SS.MM.HH
TC2: 4-bytes indicating a second timecode point in the format of FF.SS.MM.HH

Response: **10 01** or
D0 11 if ID does not exist

Description: This command provides the same function as **Erase Segment (A8 11)** except it does not require the clip being trimmed to first be cued in a Nexio server channel.

This command is used to erase or trim a portion of material within a specified clip. **TC1** specifies the starting timecode position and **TC2** specifies the ending timecode position of the portion to erase. All timecode positions within will be erased and deallocated. The new in point, out point, and duration of the loaded ID automatically will be calculated and reflected in the ID database.

It is typical to use this command incorporating either the starting timecode or the ending timecode of the loaded ID so that it is trimmed from one side or the other. Erasing segments in the middle of the loaded clip will cause black to play back during those timecode points.

DropFrame Timecode Note: In the timecode syntax of FF.SS.MM.HH, the frames unit has a base value of 40 to indicate drop frame timecode in NTSC. PAL timecode and non-drop frame NTSC are 0-based.

C8 C5 — Consolidate ID

Syntax: **C8 C5 ID**
ID: 8-byte ID handle

Response: **D4 C5 TID**
TID: 4-byte unsigned integer, MSB first, identifying a Transaction ID associated with the command.

Description: This command purges the specified ID handle such that all of its video and audio material not included in subclips will be released from the system. In effect, it consolidates the ID down to only that media within it that are in use by its subclips. During this process, the server sets the Purged special attribute (Mask 0x0040) for the ID.

The controller must use the **Get Transaction Status (C4 B3)** command at greater than 50 millisecond intervals until it changes from TRANSACTION_PENDING until it returns to TRANSACTION_SUCCESS or an error.

If the specified ID handle is that of a subclip or of an ID handle that does not exist, the command will fail with a transaction status of TRANSACTION_NO_FILE.

C4 B3 — Get Transaction Status

Syntax: **C4 B3 TID**
TID: 4-byte unsigned integer, MSB first, identifying the Transaction ID to be queried.

Response: **D4 B3 STATUS** or
D0 B3 if specified transaction ID does not exist
STATUS: 4-byte unsigned integer, MSB first

Description: This command is used to follow up previous commands in which a transaction ID was set so that the controller can check the progress of the previous command's completion. The controller is responsible for retrieving the transaction ID as part of the server response in the previous command and using that transaction ID as part of this command.

In response to this command, the server responds with the current **STATUS**, one of the following states.

Table 2-71: Transaction Status Responses

Response	State
0x0000	TRANSACTION_PENDING
0x0001	TRANSACTION_SUCCESS
0x0002	TRANSACTION_NO_FILE
0x0003	TRANSACTION_DIRECTORY_FULL
0x0004	TRANSACTION_DISK_FULL
0x0005	TRANSACTION_DISK_ERROR
0x0006	TRANSACTION_WRITE_PROTECTED
0x0007	TRANSACTION_INVALID_FORMAT
0x0008	TRANSACTION_ABORTED
0x0009	TRANSACTION_FILE_CHANGED

0x000A	TRANSACTION_CONNECT_FAILED
0x000B	TRANSACTION_NAME_CONFLICT

The status of a completed transaction will be remembered for a reasonable amount of time after its completion.

It is recommended that the controller should poll the transaction status at greater than 50 millisecond intervals until the status changes from TRANSACTION_PENDING. The new status will be the final one.

C4 B4 — Get Transaction ID Handle

Syntax: **C4 B4 TID**
TID: 4-byte unsigned integer, MSB first, identifying the Transaction ID to be queried.

Response: **D8 B4 ID** or
D0 B4 if specified transaction ID does not exist
ID: 8-byte ID handle or other identification

Description: This command typically is used to retrieve a newly created clip's ID handle. This command should not be used until a previous **Get Transaction Status (C4 B3)** command has returned TRANSACTION_SUCCESS.

This command is also used as part of the **Erase ID with Transaction (C8 4F)** command in which it returns a controller ID, not an ID handle.

C4 BC — Get Transferred Field Count

Syntax: **C4 BC TID**
TID: 4-byte unsigned integer, MSB first, identifying the Transaction ID to be queried.

Response: **D4 BC FIELDS** or
D0 BC if specified transaction ID does not exist
FIELDS: 4-byte unsigned integer, MSB first

Description: This command is typically used following a read file or write file command (**CF A0** and **CF A1**, respectively) during which the transaction ID was returned. It provides the number of fields transferred as of the request. This is just a progress indication. It should not be relied upon for detecting completion of the transaction.

This command is also used as part of the **Erase ID with Transaction (C8 4F)** command in which it returns information about a clip protected from deletion.

C4 BD — Get Transaction Result

Syntax: **C4 BD TID**
TID: 4-byte unsigned integer, MSB first, identifying the Transaction ID to be queried.

Response: **DF BD BC RESULT** or
D0 BD if specified transaction ID does not exist
BC: 1-byte unsigned integer, indicating total amount of data to follow
RESULT: Variable length transaction result

Description: This command is used in conjunction with the **Get Transaction Status (C4 B3)** command to retrieve specific data associated with another command.

One of the primary roles of this command is associated with the **Get Disk Information (C1 C7)** command for retrieving storage volume write errors. See the **C1 C7** command for more information.

C4 A5 Change Notification

Syntax: **C4 A5 MASK**
MASK: 4-byte unsigned integer, MSB first

Response: **10 01** immediately and
CF A6 BC NN DATA when notifications are ready
BC: 1-byte unsigned integer, indicating total amount of data to follow.
NN: 1-byte unsigned integer, indicating the notification number
DATA: See table below

Description: A controller can connect to the server for the purpose of monitoring record processes and the added/deleted ID lists without having to frequently poll for status. This helps to reduce network traffic and the possibility of data packet collisions.

The MASK determines which list or process is to be monitored, as specified in the following table.

Table 2-72: Notification Mask and Response

Mask	NN	Description	Server Notification
00 00 00 00	---	Disables notifications	---
00 00 00 01	0x00	IDs in Added List	BC=0x01, NN=0x00, DATA=none
00 00 00 02	0x01	IDs in Deleted List	BC=0x01, NN=0x01, DATA=none
00 00 00 04	0x02	Recording started	BC=0x11, NN=0x02, DATA=TSTC LOCTC ID
00 00 00 08	0x03	Recording stopped	BC=0x11, NN=0x03, DATA=TSTC LOCTC ID

More than one notification can be enabled at the same time by including all the bits to be covered in the **MASK**. When a condition is such that it meets one of the notifications, the server sends the controller the **CF A6** command. When a controller receives the notification command from the server, the controller must respond with an **Ack** (**10 01)** or else the notifications will stop being sent. In addition, if an Added or Deleted notification is sent, the controller must respond with an **Ack (10 01)** and clear the appropriate ID list using the **List First ID List (C1 4C)** and **List Next ID List (C0 4D)** commands.

Notifications are serviced per controller session. The session is created when the serial or Ethernet connection is established with the server. The session that initiates the command is the same session that must clear the added or deleted lists in order to receive future notifications.

The format for the DATA parameter in Notification Numbers 0x02 and 0x03 are as follows:

TSTC: Time-of-day time stamp indicating when the recording started or stopped, in the format of FF.SS.MM.HH.

LOCTC: Timecode location of the recording when it started or stopped, in the format of FF.SS.MM.HH.

ID: 8-byte ID handle

Example: Use of the Change Notification command in which all notifications are enabled. The controller sends the following:

```
C4 A5 00 00 00 0F       // Set notification for all
```

The server returns the following as each instance is encountered:

```
CF A6 01 00             // Added IDs list changed
CF A6 01 01             // Deleted IDs list changed
CF A6 11 02 01 02 03 04 00 00 00 00 25 30 31 32 33 34 35 36
                        // Indicates recording started
CF A6 11 03 10 20 30 04 09 18 27 00 25 30 31 32 33 34 35 36
                        // Indicates recording stopped
```


Cue Commands

2x 31 — Cue Up with Data

Syntax: **20 31** or
24 31 TC or
28 31 ID or
2C 31 TC ID

TC: 4-bytes indicating a timecode point in the format of FF.SS.MM.HH

ID: 8-byte ID handle

Response: **10 01**

Status bits set: **PRRL, (DIR)** during search, then
SHTL, STILL, CUED, PST IN, PST OUT when complete

Codec Mode: **Decompression**

Description: This command is used to cue an ID, based on the given information. The server will search to and display the requested frame. If successful, the PRESET IN register will be set to the requested time, as if a **Preset In Point (40 14)** command had been issued, and the CUED bit (status byte 2, bit 0) will be set high, as retrieved using the **Get Channel Status (61 20)** command.

It is up to the controller to determine if a clip is allowed to be loaded and played from any particular channel. Use the **Get Special Machine Properties (C4 B1)** command for mask 17 (0x00020000) to determine if the channel will accept the resolution and format of the clip about to be loaded. Loading an unsupported clip in a channel can result in black on air or cause an overload of the CPUs.

There are four different uses of this command, each one requiring slightly different parameters:

20 31 Cues the channel to the first frame of the currently loaded ID. If no ID is currently loaded in the channel, the command will fail.

24 31 TC Cues the channel to the indicated timecode value. If an ID is currently loaded in the channel, this command will load the ID to the specified timecode value. If the ID does not contain the specified timecode value, the clip will be loaded to the closest possible frame. This can be the first or last frame of the clip. If no ID is currently loaded, this command will cue to the desired timecode within Timecode space and display its essence (typically black).

28 31 ID Cues the channel to the first frame of the specified ID handle. This will load the ID in the channel if it was not already there. If the ID handle does not exist, the command will fail.

2C 31 TC ID Cues the channel to the indicated timecode within the specified ID handle. This will load the ID in the channel if it was not already there. If the ID handle does not exist, the command will fail. If the indicated timecode does not exist for that ID, the server will cue to the ID's In Point or Out Point, whichever is closer.

If the ID does not include the timecode value specified in the command, the clip will be loaded to the timecode position nearest the specified value.

Drop Frame Timecode Note: In the timecode syntax of FF.SS.MM.HH, the frames unit has a base value of 40 to indicate drop frame timecode in NTSC. PAL timecode and non-drop frame NTSC are "0"-based.

Cx 86 — Fast Cue Up with Data

Syntax:
C0 86 or
C4 86 TC or
C8 86 ID or
CC 86 TC ID

TC: 4-bytes indicating a timecode point in the format of FF.SS.MM.HH
ID: 8-byte ID handle

Response: **10 01**
Status bits set: **PRRL, (DIR)** during search, then
SHTL, STILL, CUED, PST IN, PST OUT when search is complete
Codec Mode: Decompression

Description: This command has the same behavior as the **Cue Up with Data (2x 31)** command except for the following:

- This command completes much faster.
- The pre-roll flag will drop faster.
- The latency of a subsequent **Play (20 01)** command is not guaranteed.
- The time code only format, **C4 86**, will cue inside the currently loaded ID, thus avoiding directory search. If the command is issued very frequently, that may save some processor time.

It is up to the controller to determine if a clip is allowed to be loaded and played from any particular channel. Use the **Get Special Machine Properties (C4 B1)** command for mask 17 (0x00020000) to determine if the channel will accept the resolution and format of the clip about to be loaded. Loading an unsupported clip in a channel can result in black on air or cause an overload of the CPUs.

As with the other Cue command, there are four different uses of this command, each one requiring slightly different parameters:

C0 86 Cues the channel to the first frame of the currently loaded ID. If no ID is currently loaded in the channel, the command will fail.

C4 86 TC Cues the currently loaded ID to the specified timecode.

C8 86 ID Cues the channel to the first frame of the specified ID handle. This will load the ID in the channel if it was not already there. If the ID handle does not exist, the command will fail.

CC 86 TC ID Cues the channel to the indicated timecode within the specified ID handle. This will load the ID in the channel if it was not already there. If the ID handle does not exist, the command will fail. If the indicated timecode does not exist for that ID, the server will cue to the ID's In Point or Out Point, whichever is closer.

Drop Frame Timecode Note: In the timecode syntax of FF.SS.MM.HH, the frames unit has a base value of 40 to indicate drop frame timecode in NTSC. PAL timecode and non-drop frame NTSC are "0"-based.

4x 14 Preset In Point

Syntax: **40 14** or
44 14 TC or
48 14 ID or
4C 14 TC ID
TC: **4-bytes indicating a timecode point in the format of FF.SS.MM.HH**
ID: **8-byte ID handle**

Response: **10 01**
Status bits set: **PRRL, (DIR)** during search, then
SHTL, STILL, CUED, PST IN, PST OUT when complete
Codec Mode: Decompression

Description: This command has exactly the same behavior as the **Cue Up with Data (2x 31)** command. The server will search to and display the requested frame. If successful, the PRESET IN register will be set to the requested time, and the CUED bit (status byte 2, bit 0) will be set high, as retrieved using the **Get Channel Status (61 20)** command.

It is up to the controller to determine if a clip is allowed to be loaded and played from any particular channel. Use the **Get Special Machine Properties (C4 B1)** command for mask 17 (0x00020000) to determine if the channel will accept the resolution and format of the clip about to be loaded. Loading an unsupported clip in a channel can result in black on air or cause an overload of the CPUs.

As with the other Cue commands, there are four different uses of this command, each one requiring slightly different parameters:

40 14 Cues the channel to the first frame of the currently loaded ID. If no ID is currently loaded in the channel, the command will fail.

44 14 TC Cues the channel to the indicated timecode value. If an ID is currently loaded in the channel, this command will load the ID to the specified timecode value. If the ID does not contain the specified timecode value, the clip will be loaded to the closest possible frame. This can be the first or last frame of the clip. If no ID is currently loaded, this command will cue to the desired timecode within Timecode space and display its essence (typically black).

48 14 ID Cues the channel to the first frame of the specified ID handle. This will load the ID in the channel if it was not already there. If the ID handle does not exist, the command will fail.

4C 14 TC ID Cues the channel to the indicated timecode within the specified ID handle. This will load the ID in the channel if it was not already there. If the ID handle does not exist, the command will fail. If the indicated timecode does not exist for that ID, the server will cue to the ID's In Point or Out Point, whichever is closer.

If the ID does not include the timecode value specified in the command, the clip will be loaded to the timecode position nearest the specified value.

Drop Frame Timecode Note: In the timecode syntax of FF.SS.MM.HH, the frames unit has a base value of 40 to indicate drop frame timecode in NTSC. PAL timecode and non-drop frame NTSC are "0"-based.

4x 15 — Preset Out Point

Syntax: **40 15** or
44 15 TC
TC: 4-bytes indicating a timecode point in the format of FF.SS.MM.HH

Response: **10 01**
Status bits set: **PST OUT**

Description: This command is used to set the PRESET OUT register to the information given. If media is playing and the current timecode reaches the timecode in the register, then playback will stop at that point. This has the effect of setting a new Out Point for the currently loaded ID.

There are two different uses of this command:

40 15 Sets the channel's PRESET OUT register to the last frame of the loaded ID.

44 15 TC Sets the channel's PRESET OUT register to the indicated timecode value.

Drop Frame Timecode Note: In the timecode syntax of FF.SS.MM.HH, the frames unit has a base value of 40 to indicate drop frame timecode in NTSC. PAL timecode and non-drop frame NTSC are "0"-based.

Cx AB — Reverse Preset Out Point

Syntax: **C0 AB** or
C4 AB TC
TC: 4-bytes indicating a timecode point in the format of FF.SS.MM.HH

Response: **10 01**

Description: This command is used to set a channel's Preset Out Point register for when the media is playing in reverse. If the media is playing in reverse and the current timecode reaches the timecode in the register, then playback will stop at that point.

There are two different uses of this command:

C0 AB Sets the channel's Reverse Preset Out Point register to the first frame of the loaded ID.

C4 AB TC Sets the channel's Reverse Preset Out Point register to the indicated timecode value.

Drop Frame Timecode Note: In the timecode syntax of FF.SS.MM.HH, the frames unit has a base value of 40 to indicate drop frame timecode in NTSC. PAL timecode and non-drop frame NTSC are "0"-based.

40 20 — Reset In Point

Syntax: **40 20**

Response: **10 01**

Description: This command clears the Preset In register and the PST IN bit (Status byte 3, bit 0), as retrieved using the **Get Channel Status (61 20)** command.

This command is not intended for use when navigating to a new cued position within a clip. The controller should instead use the **Cue with Data (24 31)** command.

40 21 — Reset Out Point

Syntax: **40 21**

Response: **10 01**

Description: This command clears the Preset Out register and the PST OUT bit (Status byte 3, bit 1), as retrieved using the **Get Channel Status (61 20)** command.

This command is not intended for use when navigating to a new cued position within a clip. The controller should instead use the **Cue with Data (24 31)** command.

Ax 04 — Preset Preview In Point

Syntax: **A0 04** or
A4 04 TC or
A8 04 ID or
AC 04 TC ID

TC: 4-bytes indicating a timecode point in the format of FF.SS.MM.HH
ID: 8-byte ID handle

Response: **10 01**
Status bits set: **PVW IN**

Description: This command is used to specify the In Point of the next desired clip to be played when the end of the current clip is reached. After completion of this command, the PREVIEW IN status bit (byte 9 bit 1) will be set high, as retrieved using the **Get Channel Status (61 20)** command.

It is common practice to use this command in conjunction with the **Preset Preview Out Point (A0 05)** command.

It is up to the controller to determine if a clip is allowed to be loaded and played from any particular channel. Use the **Get Special Machine Properties (C4 B1)** command for mask 17 (0x00020000) to determine if the channel will accept the resolution and format of the clip about to be loaded. Loading an unsupported clip in a channel can result in black on air or cause an overload of the CPUs.

There are four different possible uses of this command:

A0 04 The Preview In register will be set to the first frame of the currently loaded ID.

A4 04 TC The Preview In register will be set to the indicated timecode.

A8 04 ID The Preview In register will be set to the first frame of the specified ID handle.

AC 04 TC ID The Preview In register will be set to the indicated timecode of the specified ID handle.

Drop Frame Timecode Note: In the timecode syntax of FF.SS.MM.HH, the frames unit has a base value of 40 to indicate drop frame timecode in NTSC. PAL timecode and non-drop frame NTSC are "0"-based.

Ax 05 — Preset Preview Out Point

Syntax: **A0 05** or
A4 05 TC

TC: 4-bytes indicating a timecode point in the format of FF.SS.MM.HH

Response: **10 01**
Status bits set: **PVW OUT**

Description: This command is used to specify the Out Point of the next desired clip to be played when the end of the current segment is reached. After completion of this command, the PREVIEW OUT status bit (byte 9, bit 0) will be set high, as retrieved using the **Get Channel Status (61 20)** command.

There are two different possible uses of this command:

A0 05 The Preview Out register will be set to the last frame of the currently loaded ID.

A4 05 TC The Preview Out register will be set to the indicated timecode.

Drop Frame Timecode Note: In the timecode syntax of FF.SS.MM.HH, the frames unit has a base value of 40 to indicate drop frame timecode in NTSC. PAL timecode and non-drop frame NTSC are "0"-based.

A0 06 — Reset Preview In Point

Syntax: **A0 06**

Response: **10 01**

Description: This command clears the Preview In register and the PVW IN bit (Status byte 9, bit 0), and invalidates the previewed clip, if one has been previously specified.

A0 07 — Reset Preview Out Point

Syntax: **A0 07**

Response: **10 01**

Description: This command clears the Preview Out register and the PVW OUT bit (Status byte 9, bit 1).

C1 16 — Select Preset Audio Routing Profile

Syntax: **C1 16 PROFILE**
PROFILE: 1-byte unsigned integer, valid values between 00 and 31

Response: **10 01**

Description: This command is used to apply predefined audio routing to the currently cued or playing clip based on the output masks of the selected audio **PROFILE**.

It works similarly to the **Cue Up with Data** (**2x 31** and **4x 14**) commands in that once called, you must wait for the CUED status bit as an indication that the operation has completed. The recommended usage is to first cue up the ID, wait for the CUED status bit, select the preset audio profile, and wait for the CUED status bit again. Once cued, the Play command can be sent, and the audio routing will take effect immediately.

If the ID is recued, this command must be resent to reapply the desired audio routing. Otherwise, the channel will apply the default audio routing. Once the next ID is cued, this command must be sent again if a specific **PROFILE** is desired. In addition, when an ID is stacked for sequenced play back, you can send the **Select Preview Audio Routing Profile (C1 17)** command.

If audio profiles do not exist on the machine, then the channel uses the default audio routing, or those set by the **Set Audio Routing Parameters (CF 55)** command.

If this command is sent mid-playback, there will be a latency of approximately 1 second before the new routing kicks in.

Note: This command was introduced as part of the Nexio 7.0 Software Release and does not function in prior released software.

C1 17 — Select Preview Audio Routing Profile

Syntax: **C1 17 PROFILE**
PROFILE: 1-byte unsigned integer, valid values between 00 and 31

Response: **10 01**

Description: This command is used to apply predefined audio routing to the next clip stacked and ready for playback in a channel based on the output masks of the selected audio **PROFILE**.

It works similarly to the **Preset Preview In Point (Ax 04)** command in that once called, you must wait for the PREVIEW READY status bit as an indication that the operation has completed. The recommended usage is to first preview the ID, wait for the PREVIEW READY status bit, select the preview audio profile, and wait for the PREVIEW READY status bit again. The audio routing takes effect immediately upon transition to the stacked clip.

If the ID in preview is overridden with another clip, this command must be resent to reapply the desired audio routing to the new clip. Otherwise, the channel will apply the default audio routing. If audio profiles do not exist on the machine, then the channel uses the default audio routing, or those set by the **Set Audio Routing Parameters (CF 55)** command.

Note: This command was introduced as part of the Nexio 7.0 Software Release and does not function in prior released software.

C0 02	Next Cue

Syntax: **C0 02**

Response: **10 01**

Description: In Sequenced or Stacked Playback mode, this command causes a seamless switch from the currently playing clip to the next preloaded clip.

If there is no preloaded clip, this command does nothing.

C0 04	Play Next Cue

Syntax: **C0 04**

Response: **10 01**

Description: This command enables the VDCP-style stacked playback mode, where an explicit call of this command is required to play the preloaded clip. If this command has not been received before the currently playing clip ends, the Codec will either stop on the last frame of the clip or the first frame of the next clip depending on the registry key VdcpStopOnNext.

In order to enable the use of this new stacked playback mode, the **Play Next Cue** command must be sent instead of the **Play (20 01)** command to play the first clip in the stack. If there is then no preloaded clip in the stack, this command will do nothing.

Use of either the **Cue Up with Data (2x 31)** or **Preset In Point (4x 14)** command will cancel this mode.

This command is designed to work by itself and not in conjunction with the **Play (20 01)** and **Next Cue (C0 02)** commands.

Ax 02 — Cue Up to Record

Syntax: **A0 02** or
A4 02 TC or
A8 02 ID or
AC 02 TC ID
TC: 4-bytes indicating a timecode point in the format of FF.SS.MM.HH
ID: 8-byte ID handle

Response: **10 01**
Status bits set: **PRRL, (DIR)** during search, then
CUED, STILL, LAMP STILL when search is complete
Codec mode: Compression

Description: This command is used to cue the channel to a specified point and prepare the CODEC for recording, given certain information. The server will start recording at this point 4 frames after it receives the **Record (20 02)** command.

If nulls (00h) are specified for the ID handle, the server will create a unique ID in the form of "%*nnnnnnn*", where "*n*" is any combination of integers 0-9 or letters A-F. Use the **Get Loaded ID (C0 03)** command to retrieve this name, after the cue is complete.

There are four different possible uses of this command:

A0 02 Arms the channel to start recording at the first frame of the loaded ID.

A4 02 TC Arms the channel to start recording at the specified timecode location within the currently loaded ID.

A8 02 ID Arms the channel to start recording at the first frame of the specified ID handle. The recording will end when the media assigned to the ID is exhausted. If the ID handle does not exist, it will be created and the channel will be prepared to start recording at timecode 00:00:00:00. Unless followed by a **Preset Out Point (40 15)** command, the recording will be open-ended.

AC 02 TC ID If the specified ID handle exists, the channel will be armed to start recording at the specified timecode. If the ID handle does not exist, it will be created and the first frame of the ID will be set to the specified timecode. Unless followed by a **Preset Out Point (40 15)** command, the recording will be open-ended.

Drop Frame Timecode Note: In the timecode syntax of FF.SS.MM.HH, the frames unit has a base value of 40 to indicate drop frame timecode in NTSC. PAL timecode and non-drop frame NTSC are "0"-based.

C4 87 — Set New Out Point

Syntax: **C4 87 TC**
TC: 4-bytes indicating a timecode point in the format of FF.SS.MM.HH

Response: **10 01**
Status bits set: PST OUT

Description: This command sets both the Out Point and the end of material for the currently loaded ID to the given value. It can be used to set an arbitrary duration after a **Cue Up to Record (Ax 02)** command. This command works for open-ended or fixed-duration recordings. If successful, the channel's PRESET OUT register is set to the indicated timecode value and the PST OUT status bit (Byte 3, Bit 1) is set, as retrieved using the **Get Channel Status (61 20)** command.

Duration information about the clip is updated upon completion.

If the current timecode is greater than the new specified Out Point, the recording will terminate at the point where the command was issued. The updated Out Point may not be equal to the specified new Out Point.

Drop Frame Timecode Note: In the timecode syntax of FF.SS.MM.HH, the frames unit has a base value of 40 to indicate drop frame timecode in NTSC. PAL timecode and non-drop frame NTSC are "0"-based.

40 41	Set Auto Mode On

Syntax: **40 41**

Response: **10 01**
Status bits set: None (resets AUTO bit)

Description: This command sets the channel into AUTO mode and clears the PST bits and Preset registers, as retrieved using the **Get Channel Status (61 20)** command.

40 40	Set Auto Mode Off

Syntax: **40 40**

Response: **10 01**
Status bits set: None (resets AUTO bit)

Description: This command removes the channel from AUTO mode and clears the PST bits and Preset registers, as retrieved using the **Get Channel Status (61 20)** command.

Transport Commands

20 00 Stop

Syntax: **20 00**

Response: **10 01**
Status bits set: **READY, STOP, LAMP STILL**
Codec Mode: Compression

Description: This command terminates the playback or recording of an ID in the current channel.

20 01 Play

Syntax: **20 01**

Response: **10 01**
Status bits set: **READY, PLAY, SRV, LAMP FWD**
Codec Mode: Decompression

Description: This command starts the playback of the currently loaded segment. If no valid segment is loaded, black is played out.

20 02 Record

Syntax: **20 02**

Response: **10 01**
Status bits set: **READY, REC, PLAY, SRV, EE ON, LAMP FWD**
Codec Mode: Compression

Description: This command starts the recording of the currently loaded ID. If no valid ID is loaded, it may be possible to record the video content into timecode space. However, the timecode space feature is disabled by default and should not be used without good reason.

20 0F Eject

Syntax: **20 0F**

Response: **10 01**
Status bits set: **READY, STOP, E-E ON, LAMP STILL**
Codec Mode: Compression

Description: This command behaves like the **Stop (20 00)** command except it places the channel into E-E.

20 10 Fast Forward

Syntax: **20 10**

Response: **10 01**
Status bits set: **READY, FF, LAMP FWD**
Codec Mode: Decompression

Description: This command places the channel into fast forward mode (4000% of Normal Speed).

20 20 — Rewind

Syntax: **20 20**

Response: **10 01**
Status bits set: **READY, REW, DIR, LAMP REV**
Codec Mode: Decompression

Description: This command places the channel into rewind mode (-4000% of Normal Speed).

60 2E — Get Variable Speed Sense

Syntax: **60 2E**

Response: **72 2E SPEED**
SPEED: 2-bytes signed integer, MSB first

Description: This command is used to request the current speed of the channel. The server response is a two-byte integer indicating the current speed, where 100 (0x0064) equals real time playback and -100 (0xFF9C) equals reverse at 100%.

2x 1x — Set Variable Speed Forward

Syntax:
21 11 SPEEDDATAx or
21 12 SPEEDDATAx or
21 13 SPEEDDATAx or
22 11 SPEEDDATAx SPEEDDATAy or
22 12 SPEEDDATAx SPEEDDATAy or
22 13 SPEEDDATAx SPEEDDATAy

SPEEDDATAx: 1-byte unsigned integer
SPEEDDATAy: 1-byte unsigned integer, presented in conjunction with **SPEEDDATAx** as MSB first

Response: **10 01**
Status bits set: **(JOG, SHTL, VAR), SRV, LAMP FWD**
Codec Mode: Decompression

Description: These commands place the channel into variable speed forward mode. Although they are technically different commands - **Jog (2x 11)**, **Variable Play (2x 12)**, and **Shuttle (2x 13)** - these commands may be used interchangeably.

The commands that make use of just one **SPEEDDATA** parameter use the following formula to provide values:

$$\mathbf{SPEED=10^{((SPEEDDATAx/32)-2)}}$$

The **SPEEDDATA** value as used in the protocol string is in hex. The value as used in the formula above is in decimal.

The formula is invalid for SPEEDDATA=0, which is STILL or 0% playback. A SPEED result of 1 is 100% speed, or normal playback. A SPEED result of 0.5 would be 50%, or half normal, and a result of 2 would be 200%, or double time.

The following table was calculated based on the above formula**.**

Table 2-73: 1-Byte Speed Data Calculations

Speed Data	Speed (%)	Speed Data	Speed (%)	Speed Data	Speed (%)	Speed Data	Speed (%)
0x00	**STILL**	0x20	10	0x40	**100**	0x60	1000
0x01	1.0	0x21	11	0x41	108	0x61	1075
0x02	1.1	0x22	11	0x42	116	0x62	1155
0x03	1.2	0x23	12	0x43	124	0x63	1241
0x04	1.3	0x24	13	0x44	133	0x64	1334
0x05	1.4	0x25	14	0x45	143	0x65	1433
0x06	1.5	0x26	15	0x46	154	0x66	1540
0x07	1.7	0x27	17	0x47	166	0x67	1655
0x08	1.8	0x28	18	0x48	178	0x68	1778
0x09	1.9	0x29	19	0x49	191	0x69	1911
0x0A	2.0	0x2A	20	0x4A	205	0x6A	2054
0x0B	2.2	0x2B	22	0x4B	221	0x6B	2207
0x0C	2.4	0x2C	24	0x4C	237	0x6C	2371
0x0D	2.6	0x2D	26	0x4D	255	0x6D	2548
0x0E	2.7	0x2E	27	0x4E	274	0x6E	2738
0x0F	2.9	0x2F	29	0x4F	294	0x6F	2943
0x10	3.2	0x30	32	0x50	316	0x70	3162
0x11	3.4	0x31	34	0x51	340	0x71	3398
0x12	3.7	0x32	37	0x52	365	0x72	3652
0x13	3.9	0x33	39	0x53	392	0x73	3924
0x14	4.2	0x34	42	0x54	422	0x74	4217
0x15	4.5	0x35	45	0x55	453	0x75	4532
0x16	4.9	0x36	49	0x56	487	0x76	4870
0x17	5.2	0x37	52	0x57	523	0x77	5233
0x18	5.6	0x38	56	0x58	562	0x78	5623
0x19	6.0	0x39	60	0x59	604	0x79	6043
0x1A	6.5	0x3A	65	0x5A	649	0x7A	6494
0x1B	7.0	0x3B	70	0x5B	698	0x7B	6978
0x1C	7.5	0x3C	75	0x5C	750	0x7C	7499
0x1D	8.1	0x3D	80	0x5D	806	0x7D	8058
0x1E	8.6	0x3E	87	0x5E	866	0x7E	8660
0x1F	9.3	0x3F	93	0x5F	931	0x7F	9306

Speeds above 110% are dependent on drive performance and may not be exact. **SPEEDDATA** values above 0x7F will translate to 9306% speed.

The commands that allow two bytes for SPEEDDATA provide for a much more precise value of speed. For example, with the one-byte commands, exactly 50% playback is not possible. By manipulating the two SPEEDDATA values in the two-byte commands, it is possible to set literally any speed value allowed.

The two-byte commands use the following formula:

$$SPEED=10^{((SPEEDDATAx/32)-2)} + SPEEDDATAy/256\ (10^{(((SPEEDDATAx+1)/32)-2)} - 10^{((SPEEDDATAx/32)-2)})$$

The SPEEDDATA values as used in the protocol string are in hex. Their values as used in the formulas above are in decimal.

The following table provides a few examples of the precision difference between the one byte formula and the two-byte formula**.**

Table 2-74: 2-Byte Speed Data Calculations

SPEEDDATAx	SPEEDDATAy	1-Byte	2-Bytes
00	00	STILL	STILL
16	5F	4.87%	5.00%
20	00	10.00%	10.00%
25	A0	14.33%	15.00%
29	A0	19.11%	20.00%
2C	BA	23.71%	25.00%
2F	43	29.43%	30.00%
33	42	39.24%	40.00%
36	5C	48.70%	50.00%
3C	01	74.99%	75.01%
40	00	100.00%	100.00%
49	A0	191.10%	200.01%
53	42	392.42%	399.97%
5C	E5	749.89%	799.94%
66	86	1539.93%	1600.06%

The formulas are invalid for SPEEDDATAx=0, which is STILL or 0% playback. A SPEED result of 1 is 100% speed, or normal playback. A SPEED result of 0.5 would be 50%, or half normal, and a result of 2 would be 200%, or double time.

Speeds above 110% are dependent on drive performance and may not be exact. **SPEEDDATA** values above 0x7F will translate to 9306% speed. Where available on a server that supports off speed audio, audio is usually not played at speeds greater than 156%.

Nexio AMP, Volt, and XS Note: The Nexio AMP, Volt, and XS servers are capable of scrubbing audio during off speed play of up to 156%.

2x 2x — Set Variable Speed Reverse

Syntax:
21 21 SPEEDDATAx or
21 22 SPEEDDATAx or
21 23 SPEEDDATAx or
22 21 SPEEDDATAx SPEEDDATAy or
22 22 SPEEDDATAx SPEEDDATAy or
22 23 SPEEDDATAx SPEEDDATAy

SPEEDDATAx: 1-byte unsigned integer

SPEEDDATAy: 1-byte unsigned integer, presented in conjunction with **SPEEDDATAx** as MSB first

Response: **10 01**

Status bits set: **(JOG, SHTL, VAR), SRV, LAMP REV, DIR**

Codec Mode: Decompression

Description: These commands place the channel into variable speed reverse mode. Although they are technically different commands - **Jog Reverse (2x 21)**, **Variable Play Reverse (2x 22)**, and **Shuttle Reverse (2x 23)** - these commands may be used interchangeably.

These commands make use of the same formulas and tables as referenced in the **Set Variable Speed Forward (2x 1x)** commands except that the resulting SPEED must be set to negative such that the play direction is in reverse.

Nexio AMP and XS Note: The Nexio AMP, Volt, and XS servers are capable of scrubbing audio during off speed play of up to -156%.

Cx 00 — Incremental Jog

Syntax: **C1 00 FIELDS** or
C4 00 FIELDS

FIELDS: 1-byte or 4-byte signed integer, depending on the command prefix, providing a signed displacement in fields, MSB first

Response: **10 01**

Description: Based on a controller repeatedly sending this command, the Nexio server is able to provide a smooth jog response in either direction. The **FIELDS** parameter indicates the number of fields to jump forward or backward with each use of the command. And then based on the series of commands sent, the Nexio server will average out the actual jog response so as to most smoothly match the desired variable playback speed.

When the resulting play back speed is within an acceptable range, the off-speed play back will include audio scrub.

Cx 01 — Jump Forward

Syntax: **C1 01 FIELDS** **or**
C4 01 FIELDS

FIELDS: 1-byte or 4-byte integer, depending on the command prefix, MSB first

Response: **10 01**

Description: This command will cause the channel to move forward the requested number of fields. This command is ignored if the channel is not in STILL mode.

Cx 02 — Jump Backward

Syntax: **C1 02 FIELDS** or
C4 02 FIELDS

FIELDS: 1-byte or 4-byte integer, depending on the command prefix, MSB first

Response: **10 01**

Description: This command will cause the channel to move backward the requested number of fields. This command is ignored if the channel is not in STILL mode.

20 61 — Full E-E On

Syntax: **20 61**

Response: **10 01**
Status bits set: **E-E ON**

Description: This command causes the channel to enter the FULL E-E ON mode, one of the channel states retrieved using the **Get Channel Status (61 20)** command. The channel will remain in this mode until it is reversed with a **Full E-E Off (20 60)** command.

Nexio AMP and XS Note: Some channels of the Nexio AMP and XS servers are configured to not display pass-through video. This command will not function on those channels.

20 60 — Full E-E Of

Syntax: **20 60**

Response: **10 01**
Status bits set: None (resets **E-E ON** bit)

Description: This command causes the channel to exit FULL E-to-E ON mode. This command does nothing if the REC status bit is high, as retrieved using the **Get Channel Status (61 20)** command.

C4 CD — Timestamped Play and Record

Syntax: **C4 CD TC**
TC: 4-bytes indicating a timecode point in the format of FF.SS.MM.HH

Response: **10 01**

Description: This command allows a controller to assign a time in the future when the next transport command issued to this channel should take effect. The **TC** parameter is expressed in terms of the "Timer 0" as returned by the **Get Current Timer Value (C1 A4 00)** command. For NTSC systems, **TC** should always be in drop frame.

If the specified **TC** is past due or if it doesn't allow enough time for the control latency, then the transport command that follows will be started asynchronously.

Therefore, for synchronous operation, **TC** should be at least:

```
current_Timer_0 + device_latency + max_network_delay
```

Typically 2 seconds is enough. The following transport commands are affected by this command:

20 02	Record	22 11	Jog Fwd
20 00	Stop	22 12	Var Fwd
20 01	Play	22 13	Shuttle Fwd

21 11	Jog Fwd	22 21	Jog Rev
21 12	Var Fwd	22 22	Var Rev
21 13	Shuttle Fwd	22 23	Shuttle Rev
21 21	Jog Rev	20 10	Fast Fwd
21 22	Var Rev	20 20	Rewind
21 23	Shuttle Rev	C0 02	Next Cue

Field accurate operation is guaranteed EXCEPT as follows:

- **2x 1x** when going at more than 150 % speed
- **2x 2x** when going at more than -150 % speed
- All decompression commands above when issued in E-E state. They will however synchronously terminate an ongoing recording.

This command cannot be used to schedule more than one transport command in advance and therefore is not recommended for the synchronous termination of record or playback.

Drop Frame Timecode Note: In the timecode syntax of FF.SS.MM.HH, the frames unit has a base value of 40 to indicate drop frame timecode in NTSC. PAL timecode and non-drop frame NTSC are “0”-based.

Next Cue Support: Support for the Next Cue command was added to the Nexio AMP line of products as of the 4.5 Release (June 2008).

C5 B7 — Timecode Triggered Record

Syntax: **C5 B7 SOURCE TC**
SOURCE: 1-byte indicating timecode source
TC: 4-bytes indicating a timecode point in the format of FF.SS.MM.HH

Response: **10 01**

Description: This command allows a controller to prepare a clip to start recording at a specified external timecode value. The clip should already have been cued up to record in the channel before this command is sent and a record command is required after it is sent.

The **SOURCE** can be VITC, LTC, or both, as defined in the following bit field structure:

Table 2-75: Record Timecode Source

BIT7 - BIT2	BIT1	BIT0
0 0 0 0 0 0	VITC	LTC

If multiple timecode sources are specified, the recording will start when either source reaches the **TC** value. For NTSC systems, the **TC** value should always be in drop frame.

The **TC** value is in the format of FF.MM.SS.HH where only the digits are matched against the incoming timecode. All flag bits are ignored.

If the timecode value is never reached, the controller must send an **Eject (20 0F)** command to the channel to clear the record state.

If no trigger **SOURCE** is selected then the recording will start at the channel's command time as specified by the record command.

Example: To start the recording when either LTC or VITC reaches a timecode of 01:02:03:04 send:

```
C5 B7 03 04 03 02 01
```

To start recording when only VITC reaches an NTSC timecode value of 01:02:03:04 send:

```
C5 B7 02 44 03 02 01
```

Drop Frame Timecode Note: In the timecode syntax of FF.SS.MM.HH, the frames unit has a base value of 40 to indicate drop frame timecode in NTSC. PAL timecode and non-drop frame NTSC are 0-based.

CF 88 Local Timeline Contro

Syntax: **CF 88 BC SUB BITMAP DATA**
BC: 1-byte unsigned integer, indicating total amount of data to follow
SUB: 1-byte unsigned integer, indicating sub-code for the command
BITMAP: 1-byte bitmap, indicating data arguments to apply
DATA: Variable amount of data based on data arguments selected

Response: **DF 88 BC ERROR BITMAP DATA**
BC: 1-byte unsigned integer, indicating total amount of data to follow
ERROR: 1-byte unsigned integer, indicating error code status for the command
BITMAP: 1-byte bitmap, indicating data arguments to apply
DATA: Variable amount of data based on data arguments selected

Description: This command allows a controller to replace the standard cue up and play set of commands with a more tightly controlled timeline playback system which allows building a list of clips or pieces of clips that are as short as 10 frames in length. Normal stacked play back requires clips no shorter than 3 seconds in length.

The **Local Timeline Control** command is more than just a single command. It's actually several different commands built into one. The idea is that the controller prepares a playlist of clips, starts them playing back, and then dynamically manages the list as it plays to air. And the controller does it all with the CF 88 command.

There are three different sets of data managed within this command: the data bitmap variables, the sub-code functions, and the error code status. The data bitmap identifies the parameters used within the functions.

Table 2-76: Timeline Data Bitmap

Bit	Mask	Size	Description
0	0x01	2	Entry ID, MSB first
1	0x02	2	New Entry ID, MSB first
2	0x04	4	Timecode in point
3	0x08	4	Duration timecode
4	0X10	8	8-byte ID Handle
6	0x40	Variable	Transition effects descriptor (not currently available)

7	0x80	Variable	Extended ID name

The first two entry ID parameters refer to the placement of an ID within the timeline. The first entry should be position 00 with each each incrementing by one up to a maximum entry ID of 255. No more than 256 entries can be managed within the timeline at any one time.

The two timecode parameters are entered in the syntax of FF SS MM HH (frames, seconds, minutes, hours). The controller can identify clips by their 8-byte ID handle or the extended ID name. And the transition effects descriptor is for future implementation.

The available functions that call these parameters are as follows:

Table 2-77: Timeline Sub-code Functions

Sub-Code	Function	Available Parameters	Response Parameters
0	Append new entry	1,([2],[3],4|7)|(3,6)	
0	Insert before existing entry	0,1,([2],[3],4|7)|(3,6)	
1	Modify existing entry	0,([2],[3],[4|7])|(3,6)	
2	Delete entire timeline		
2	Delete entry	0	
3	Query timeline position		[0,3]
4	Get first entry ID		[0]
4	Get next entry ID	0	[0]
5	Query entry with 8-byte ID	0	[(2,3,4)|(3,6)]
6	Query entry with extended ID	0	[(2,3,7)|(3,6)]

Items in square brackets, [ ], are optional. Items separated by the vertical bar, |, represent either one set of parameters or the other listed. The duration response parameter for sub-code 3 reports the timeline position based on the timecode progress from the in-point of the clip, not the actual timecode of the clip itself.

All server responses include an error code.

Table 2-78: Timeline Error Code

Code	Description
0	Success
1	Entry not found
2	New entry ID already exists
3	No room for the new entry
4	ID name not found
5	Not in local timeline playback mode
6	Invalid sub-code
7	Invalid sub-code/parameter combination
8	Invalid parameter list size

For more information on the workflow and proper implementation of this command, please review the section *Short Clips Stacked Play Back* at the beginning of this chapter.

It is important to note that the **Local Timeline Control** does not work the same as the normal stacked play back of clips. Instead of using one of the cue commands,

timeline control makes use of the **Preroll (20 30)** command to cue the timeline. The **Set Automode On (40 41)** and **Off (40 40)** commands are used to delete the currently active timeline. And timeline playback mode is terminated by any normal cue up command or the Preroll command when the timeline is currently empty.

Local Timeline Control is also limited in its use of transport commands. In the default set up, only the following transport commands function in this mode: Play, Shuttle Forward in a linear speed range, Stop, and Eject. Reverse does not work. And once the timeline is fully played out, the timeline is auto-deleted. And subsequent "Get First Entry ID" sub-code will return the currently playing or loaded entry, but nothing else.

There is also an advanced mode available to controllers. This mode allows reverse playback. It allows cueing to an absolute timecode position within the timeline. And it does not auto-delete the timeline at the end of play back.

To set this advanced mode, the controller should use the **Set Channel Properties (CF B1)** command for mask 0x00040000 and set the channel to a value of 1.

RAIDSoft Information Commands

C0 81 — List First Disk

Syntax: **C0 81**

Response: **DC 81 LABEL** or
D0 81 if no disks found

LABEL: 12-byte string, null padded

Description: This command returns the label of the first disk volume in the data array. If there are no valid disks, **D0 81** is returned. This command is followed up with the **C0 82 (List Next Disk)** command to retrieve the rest of the disk volumes, if any.

C1 81 — List First Disk with Status

Syntax: **C1 81 MODE**

MODE: 1-byte unsigned integer indicating the command mode desired

Response: **DC 81 LABEL** or
DD 81 LABEL STATUS or
D0 81 if no disks found

LABEL: 12-byte string, null padded

STATUS: 1-byte unsigned integer

Description: This command returns the label of the first disk volume in the data array with the option of also requesting the status of the disk array. The **MODE** options are as follows:

Table 2-79: Disk Array Status Modes

MODE	Description	Response
0	List only valid disks, no status. Same as **C0 81** command	DC 81 **LABEL**
1	List all disks with status of 1=Valid, 0 – Invalid	DD 81 **LABEL** 01
2	List all disks with extended status	DD 81 **LABEL STATUS**

The STATUS for the second MODE can be one of four possible conditions when the disk array is valid:

Table 2-80: Disk Array Status

STATUS	Description
1	The disk array is valid
2	The disk array is in a broken state
3	The disk array is in a degraded state
4	The disk array is currently rebuilding

If there are no valid disks, **D0 81** is returned. This command is followed up with the **C0 82 (List Next Disk)** command to retrieve the rest of the disk volumes, if any.

C0 82 — List Next Disk

Syntax: **C0 82**

Response: **DC 81 LABEL** or
DD 81 LABEL STATUS or
D0 81 if no disks found
LABEL: 12-byte string, null padded
STATUS: 1-byte unsigned integer

Description: This command is used after the **C0 81 (List First Disk)** or the **C1 81 (List First Disk with Status)** command has been used to retrieve the first disk volume in the data array. This command returns the label of the next disk volume in the data array and optionally the STATUS of the disk array. See the **C1 81 (List First Disk with Status)** command for a description of the possible **STATUS** values for when this command follows the **C1 81** command,

This command should be used repeatedly until there are no more disk volumes. If there are no more valid disk volumes, **D0 81** is returned.

CC 80 — Select Disk

Syntax: **CC 80 LABEL**
LABEL: 12-byte string, null padded

Response: **10 01** or
11 12

Description: This command selects the disk specified in **LABEL**. Any subsequent ID commands are confined to the specified volume. **Cue Up to Record (Ax 02)** commands will create the ID on the selected array.

A **NACK (11 12)** is returned if either the disk was not found or the array is already selected.

C0 80 — Select All Disks

Syntax: **C0 80**

Response: **10 01**

Description: This command selects all disk volumes available. This requires the ID commands to search across all the disks before executing. **Cue Up to Record (Ax 02)** commands will create the ID on the disk that has the largest available space. This is the default mode.

C1 C7 — Get Disk Informatio

Syntax: **C1 C7 MODE**
MODE: 1-byte unsigned integer

Response: **Dx C7 DATA**
DATA: This parameter depends on the **MODE** selected and is defined in the table below.

Description: This command retrieves the disk information specified by **MODE** for the selected disk. If no disk is selected, the function will return the sum over all the disks.

Table 2-81: Disk Information Options

MODE	Server Response	Description	
0	D8 C7 FREE_SEG TOTAL_SEG	FREE_SEG: TOTAL_SEG:	4-bytes, MSB first, listing segments still available 4-bytes, MSB first, listing total segments on system
1	D8 C7 FREE_TIME TOTAL_TIME	FREE_TIME: TOTAL_TIME:	4-bytes, in format of FF.SS.MM.HH, displaying time remaining on system 4-bytes, in format of FF.SS.MM.HH, displaying total time on system
2	D8 C7 FREE_MBIT TOTAL_MBIT	FREE_MBIT: TOTAL_MBIT:	4-bytes, MSB first, listing Megabits remaining on system 4-bytes, MSB first, listing total Megabits available on system
3	D4 C7 BPS	BPS:	4-bytes, MSB first, displaying bits per second. See note below *
5	DA C7 FREE_TIME TOTAL_TIME	FREE_TIME: TOTAL_TIME:	5-bytes, in format of FF.SS.MM.HHHH, displaying time remaining 5-bytes, in format of FF.SS.MM.HHHH, displaying total time on system
6	D4 C7 EFS	EFS:	4-byte unsigned integer, MSB first, displaying the current extended fields size in bytes
7	D4 C7 TID	TID:	4-byte unsigned integer, MSB first, providing access to information about volume write errors
8	D1 C7 STATE	STATE:	1-byte unsigned integer, where 0 is normal and 1 indicates the server has entered a play-only mode where writes cannot be made to the storage
9	DC C7 FREE_TIME TOTAL_TIME	FREE_TIME: TOTAL_TIME:	6-bytes, in format of FF.SS.MM.HHHHHH, displaying time remaining 6-bytes, in format of FF.SS.MM.HHHHHH, displaying total time on system
Other	D0 C7	No other information is returned	

Because of the way the Nexio server writes data to the storage system, there may be a negligible change in available storage when configuring channels for less than 14 M/bits per second.

MODE 3 Note: This provides the aggregate record data rate needed for the currently selected record parameters on the currently selected disk. If no disk is selected and disks with different allocation unit sizes are present, a maximum over all the disks will be taken.

MODE 7 Note: This is a transaction based method to retrieve storage volume write error information. In order to retrieve the error information, the controller should poll using the **Get Transaction Status (C4 B3)** command at greater than 50 millisecond intervals until it changes from TRANSACTION_PENDING. If the status changes to

TRANSACTION_SUCCESS, the controller then uses the **Get Transaction Result (C4 BD)** command to retrieve the transaction result.

The format of the transaction result is as follows:

<Transaction Result (n)>=
<member-recoverable errors 0 (4)><volume-recoverable errors 0 (4)><unrecoverable errors 0 (4)>
<member-recoverable errors 1 (4)><volume-recoverable errors 1 (4)><unrecoverable errors 1 (4)>
...
<member-recoverable errors m-1 (4)><volume-recoverable errors m-1 (4)><unrecoverable errors m-1 (4)>

where “m” is the volume's member number (currently either 1 or 2) and n=12*m. All errors are expressed in allocation units, MSB first. Presently m=1 for single volume systems and m=2 for mirrored volumes.

The <member-recoverable errors> are the allocation units that are correctable by the member's ECC parity with no help from mirroring.

The <volume-recoverable errors> are the allocation units unrecoverable within the member but recoverable via mirroring (null when m=1).

The <unrecoverable errors> are the allocation units that are unrecoverable on either member and therefore play as black.

A0 1C — Get Maximum Storage Lengt

Syntax: **A0 1C**

Response: **84 1C TC**
TC: 4-bytes indicating timecode in the format of FF.SS.MM.HH

Description: This command returns the length of the largest available contiguous storage area for an ID.

C0 B2 — Get Allocation Unit Size

Syntax: **C0 B2**

Response: **D4 B2 SIZE**
SIZE: 4-byte unsigned integer, MSB first

Description: This command retrieves the allocation unit size in bytes for the selected disk. If no disk is selected the function will return the minimum over all the disks.

Cx C9 — Get ID File Size

Syntax: **C8 C9 ID** or
CA C9 ID TYPE
ID: 8-byte ID handle
TYPE: 2-byte unsigned integer, MSB first, based on table below

Response: **D8 C9 SIZE** or
D0 C9 if ID handle is not found
SIZE: 8-byte unsigned integer, MSB first

Description: This command retrieves the file size of the specified ID. Depending on the **TYPE** selected, the file size will be its exact disk size or its approximate transfer size. If no **TYPE** is included with the **C8 C9** command, the **SIZE** returned is the approximate transfer size.

Because of the formatting process involved in transferring a file out of the Nexio system, the file size on disk does not exactly match its size in transfer.

Table 2-82: Get File Size Types

Type	Value Returned
0x0005	Exact disk size
0x0006	Approximate transfer size

The **SIZE** returned by the server is a 64 bit value, represented MSB first. If the ID handle is not found, the server returns **D0 C9**.

Chapter 3: VDCP Protocol

General Information

***Note*:** This portion of the document was copyrighted by LOUTH AUTOMATION, INC. It has been edited with permission to reflect the precise Nexio implementation of this protocol. This document coincides with LOUTH's 2003 Protocol Manual. Any Nexio-specific additions to the text will be preceded with the declaration "**Nexio Addendum**".

The video disk communications protocol (VDCP) uses a tightly coupled master-slave methodology. The controlling device will take the initiative in communications between the controlling device (automation) and the controlled device (server). The topology will be point-to-point. The video disk protocol conforms to the OSI (open system interconnection) reference model. Layer 1 is the physical layer which consists of the electrical and mechanical specifications. Layer 2, the data link level, covers the synchronization and error control for the information transmitted over the physical link. Layers 3 and 4 provide network functionality and are not applicable. Layer 5, the session layer, provides the control structure for communications between applications: establishes, manages, and terminates connections (sessions) between cooperating applications. Level 6, the presentation layer contains the control language (dialect). The command table and command description provide this functionality.

Command And Response

Data communications between the CONTROLLING DEVICE (automation) and the CONTROLLED DEVICE (server) is performed in accordance with the following format:

Table 3-83: VDCP Command Structure

Field	Format	Description
STX	1 byte	Start of text code: hard-coded to 02h
BC	1 byte	Byte count: the number of bytes between the byte count and the checksum
CMD-1	1 byte consisting of 2 nibbles	Command 1: the Unit Address Nibble (Bit 0-3) should be set to 0 for Nexio servers; the Command Type Nibble (Bit 4-7) defines the type of command being sent.
CMD-2	1 byte	Command 2: the command code which identifies the syntax of the data, if any, that follows.
DATA-1 – DATA-N	N bytes, size depends on command/response	Command data
CHECKSUM	1 byte	The 2's complement of the least significant byte of the sum of all command and data bytes from the first command byte to immediately before the checksum.

Bits 4-7 of the CMD-1 byte define the type of command sent by the controller. They are broken up into the following command groups:

Table 3-84: VDCP Command Groups

Type	Description
0	System command
1	Immediate command
2	Preset/Select command
3	Sense request
4	(Not Supported)
5	(Not Supported)
8	Variable length system command
A	Variable length preset/select command
B	Variable length sense request
D	(Not Supported)
F	Archive command

The response to command types 0, 1, and 2 will be an ACK (04h) or a NAK (05h). The response to command type 3 will set the most significant bit of the CMD-2 response to a 1. For example, the response to command 30 29 is 30 A9. The command codes form a unique device dialect.

Examples of command types 1, 2, and 3 are shown below:

Immediate Command

Sample Command: Play
Command Code: 10 01
Data Configuration:

02	02	10	01	EF
ST0	BC	CMD-1	CMD-2	CS

Return: ACK (04h) or
NAK (05h)

Preset/Select Command

Sample Command: Select Input
Command Code: 20 39 + Mode
Mode Byte:
01h Off (Black w/ sync)
02h Composite
04h S-Video
08h YUV
10h D1
Data Configuration:

02	02	20	39	*	
ST0	BC	CMD-1	CMD-2	MODE	CS

Return: ACK (04h) or
NAK (05h)

Sense Request Command

Sample Command: Port Status Request
Command Code: 30 05 + Bit map

Bit Map:

PS8	PS7	PS6	PS5	PS4	PS3	PS2	PS1

Data Configuration:

02	03	30	05	*	
ST0	BC	CMD-1	CMD-2	BIT	CS

Return Code: 30 85

Data Configuration:

02		30	85	*1	*2	
ST0	BC	CMD-1	CMD-2	BIT	DATA	CS

When a bitmap is added to a command block, the data corresponding to the designated bit is accessed in the bit map. The data corresponding to bit map data 1 is added after the map. Data is added sequentially from the low order bit of the map data.

The entire command message is comprised of a sequence between 2 and 256 bytes.

Unless otherwise specified in the bit maps, a bit set (1) is true. The LSB is the "right" bit and the MSB the "left" bit. All numbers are in hexadecimal. All time values are in frames, seconds, minutes, and hours in BCD with the frames sent first and hours sent last.

Media Identifiers

The VDCP protocol is ID based. All audio/video material is referenced or accessed by an ID. The ID must be unique to a given piece of material in the system at all times. The video disk system must be a "black box" to the controller. It records and plays back without the controller knowing how or where the audio/video material is encoded, decoded, and stored.

Fixed 8-Character IDs

In the major command types 00, 20, and 30, IDs are 8 characters and are to be padded with ASCII spaces (20h) when the ID is less than 8 characters. Eight characters are always to be sent for an ID and any character not used must be set to a space.

Variable Length IDs

In the major command types 80, A0, and B0, IDs are not padded with any additional characters beyond the visible ID characters. All IDs are sent as a length byte followed by that number of characters (only visible ASCII characters allowed). All 80, A0, B0 command types are identical to the corresponding 00, 20, 30 command types except that the 8-character padded ID is replaced with this ID encoding.

There are no 10 or 40 command types with IDs, so 90 and C0 command types will not be implemented. The implementation of variable length ID major commands is optional, but all disk systems that implement them should accept all commands equivalent to the 00, 20, 30 commands they implement. Controllers may use all one type of ID encoding major command or mix them as long as the commands do not conflict.

Commands and replies should not exceed 100 bytes to limit the serial data transmission time and maintain frame accurate commands. On commands involving multiple IDs (ID List, Next, IDs Added…) no more than 80 bytes of ID data should be sent. For example, if each ID was 25 characters, only three IDs should be sent in reply to a List Type Command. This also implies 40 one-character IDs (One byte for the ID, one byte for the length of the ID) are the maximum number of IDs that may be sent in a list type reply.

Nexio Addendum: Even though it is not defined that way in the VDCP protocol, some controllers using variable length IDs set the high-bit in the first byte of the command even in commands that do not include the ID in one of the command parameters.

For example, none of the Command types includes the ID in their parameters. And even though the VDCP specification says there are no 90 command types, the Nexio server does accept them as valid. Just as the LLM accepts all the other non-ID commands with the high-bit set, they behave exactly like their counterparts.

These command extensions are not included in the command table listed below.

Unicode and Variable Length IDs

Nexio Addendum: It should also be noted that VDCP does not support the Unicode storage of text and as such will not ever enable the Unicode bit for communicating with the LLM as detailed in the Native Protocol chapter. However, the LLM will convert the incoming ANSI text to Unicode for storage in the MediaBase. It will also convert it back when retrieving data for the VDCP controller. In the case of dealing with variable length IDs, which are stored by the LLM in Unicode, the VDCP controllers should continue to treat the IDs as non-Unicode in all the commands that require them.

In addition, most Nexio Servers are set up to support variable length IDs (extended IDs) and the VDCP commands that support them. IDs may be up to 32 characters in length. If a VDCP controller tries to create an ID longer than that, the ID will be truncated.

It is technically possible for a VDCP controller that does not support the variable length ID set of commands to continue to work with the fixed 8-character ID set of commands. The 8-character IDs will be reflected in the extended IDs, however, any such controller should operate only in a closed Nexio system to prevent another controller or interface from generating its own extended IDs.

Nexio-Supported VDCP Command List

Table 3-85: Nexio-Supported VDCP Command List

Command from Controller — Return from Controlled Disk System

B.C.	CMD-1	CMD-2	CODE	NAME	B.C.	CMD-1	CMD-2	NAME
0A	00 / 80	15	+Opt	Delete Protect ID			04	ACK
0A	00 / 80	16	+Opt	UnDelete Protect ID			04	ACK

B.C.	CMD-1	CMD-2	CODE	NAME	B.C.	CMD-1	CMD-2	NAME
02	10 / 90	00	-Req	Stop			04	ACK
02	10 / 90	01	-Req	Play			04	ACK
02	10 / 90	02	-Req	Record			04	ACK
02	10 / 90	04	-Opt	Still			04	ACK
02	10 / 90	05	-Opt	Step			04	ACK
02	10 / 90	06	-Opt	Continue			04	ACK
03	10 / 90	07	-Opt	Jog			04	ACK
05	10 / 90	08	-Opt	Variable. Play			04	ACK
03	10 / 90	0A	-Opt	EE Mode			04	ACK

B.C.	CMD-1	CMD-2	CODE	NAME	B.C.	CMD-1	CMD-2	NAME
12	A0	1D	+Opt	Rename ID			04	ACK
1A	20 /A0	1F	+Opt	New Copy			04	ACK
03	20 /A0	20	+Opt	Sort Mode			04	ACK
03	20 /A0	21	-Req	Close Port			04	ACK
03	20 /A0	22	-Req	Select Port			04	ACK
0E	20 /A0	23	-Req	Record Init			04	ACK
0A	20 /A0	24	-Req	Play Cue			04	ACK
12	20 /A0	25	-Opt	Cue with Data			04	ACK
0A	20 /A0	26	+Req	Delete ID			04	ACK
03	20 /A0	2B	+Opt	% to Signal Full			04	ACK
12	20 /A0	2C	-Opt	Record Init with Data			04	ACK
0A	20 /A0	2E	-Opt	System Delete ID			04	ACK
04	20 /A0	34	-Opt	Audio In Level			04	ACK
04	20 /A0	35	-Opt	Audio Out Level			04	ACK
03	20 /A0	39	-Opt	Select Input			04	ACK
04	20 /A0	41	-Opt	Sub Carrier Adjust			04	ACK
04	20 /A0	42	-Opt	Horizontal Position Adjust			04	ACK
04	20/A0	43	-Opt	Disk Preroll			04	ACK
0C+	20 /A0	50	-Opt	Copy File To			04	ACK
0B+	20 /A0	51	-Opt	Delete File From			04	ACK
0B+	20 /A0	52	-Opt	Abort Copy File To			04	ACK

B.C.	CMD-1	CMD-2	CODE	NAME	B.C.	CMD-1	CMD-2	NAME
04	30 / B0	01	Req	Open Port	03	30 / B0	81	Grant/Denied
02	30 / B0	02	+Req	Next	00	30 / B0	82	List of IDs
03	30 / B0	05	+Opt	Port Status Request	00	30 / B0	85	State Status
03	30 / B0	06	-Opt	Position Request	07	30 / B0	86	Position
02	30 / B0	07	-Opt	Active ID Request	11	30 / B0	87	Active ID
02	30 / B0	08	+Opt	Device Type Request	00	30 / B0	88	Device Type
03	30 / B0	10	+Opt	Syst. Status Request	00	30 / B0	90	System Status
02	30 / B0	11	+Req	ID List	00	30 / B0	91	List of IDs
0A	30 / B0	14	+Opt	ID Size Request	06	30 / B0	94	ID Size
0A	30 / B0	16	+Req	ID Request	03	30 / B0	96	ID Presence
03	30 / B0	17	-Opt	Compr. Settings Request	00	30 / B0	97	Compr Settings
02	30 / B0	18	+Req	IDs Added List	00	30 / B0	98	List IDs Added
02	30 / B0	19	+Req	IDs Deleted List	00	30 / B0	99	List IDs Deleted

Legend for code column:

B.C. Byte Count of command or reply, or command or reply length.
Req Required command for a controller application to function.
Opt Optional command, and is not needed for a controller application to function, but may augment the controllers functions if implemented. Note that these commands should be replied with an

ACK if not implemented in a disk system so no error message is generated and error handling performed by the controller, but the Not Supported bit in Error Status 3A of the PORT STATUS REQUEST should be set.

+ If implemented, command is based on a communications port, not a signal port. Must function even if no signal port is open or selected.

- If implemented, command is based on a signal port, not a communications port. Can only function if a signal port is open and selected.

Unless otherwise specified in the bit maps, a bit set (1) is true. The LSB is the "right" bit and the MSB the "left" bit. All numbers are in hexadecimal. All time values are in frames, seconds, minutes, and hours in BCD with the frames sent first and hours sent last.

Server Response

04 ACK

Syntax: **04**

When the controlled device receives a command from the CONTROLLING DEVICE, the CONTROLLED DEVICE will send back this command as acknowledgment. All commands defined in this protocol should be Acked, (except when a message or status reply is sent back to the controller instead) even if not implemented by the disk. When a command not implemented is received and Acked, the Not Supported bit in Status 3A of the PORT STATUS request should be set.

05 NAK

Syntax: **05 DATA**
DATA: 1-byte bitmap, indicating type of error

When the CONTROLLED DEVICE has detected any of the errors below or does not recognize the command as one from this protocol, it will send back this reply as negative acknowledgment to the CONTROLLING DEVICE. Depending on the nature of the error, one of the DATA-1 bits will be set. **NAK** will always return 2 bytes.

Table 3-86: NAK Return Data-1

BIT7	BIT6	BIT5	BIT4	BIT3	BIT2	BIT1	BIT0
Timeout	Framing error	Overrun error	Parity error	---	Checksum error	---	Undefined error

Undefined Error: The command received could not be interpreted as one from this protocol. *Command Ignored*.

Checksum Error: The checksum of the data received as calculated by the disk system per this protocol, does not match the checksum sent in the command (last byte of the command). *Command Ignored.*

Parity Error, Overrun Error, Framing Error: Per RS-422 definition or communications definition, generally detected by communications hardware. *Disk system flushes input buffer and wait for new message.*

Time Out: When more than 10 milliseconds has past since last byte received and the number of bytes required by the byte count has not been received, or only one byte has been received. *Disk system flushed the input buffer and waits for a new message.*

System Commands

00/80 15	Delete Protect ID

Syntax: **00 15 ID or**
80 15 BC XID
ID: 8 bytes, indicating clip ID
BC: 1-byte unsigned integer, indicating total amount of data to follow
XID: Extended ID

This command prevents an ID from being deleted by an ID **DELETE** command. An **UNPROTECT** command must be used to enable **DELETE** ID once the ID is protected. The **PROTECT** ID command has no effect on the ID being played or copied.

SEND DATA 1 through SEND DATA 8 contain the ID to be protected.

Note: *Use of the **80 15** command bears the format described in the section on Variable Length IDs.*

00/80 16	Undelete Protect ID

Syntax: **00 16 ID** or
80 16 BC XID
ID: 8 bytes, indicating clip ID
BC: 1-byte unsigned integer, indicating total amount of data to follow
XID: Extended ID

This command is the opposite of **PROTECT** ID and allows the ID to be deleted but does not delete the ID by itself. This command has no effect on the ID being played or copied.

SEND DATA 1 through SEND DATA 8 contains the ID to be unprotected.

Note: *Use of the **80 16** command bears the format described in the section on Variable Length IDs.*

Immediate Commands

10 00	Stop

Syntax: **10 00**

The **STOP** command will return the selected port to the IDLE state.

The **STOP** command can be issued to a port in any state. When in IDLE, the port will output black and any material that is playing out, CUED, or cueing is aborted. No action will result if the **STOP** command is received when no material is PLAYing. If the port was in the RECORD state, then the system will stop recording at the next REF interval. A partial recording will have occurred and the internal database for the length of the material will be updated to reflect this reduced length. The part of the material received is kept stored on the disk and is available for play.

10 01 Play

Syntax: **10 01**

The **PLAY** command causes the specified ID to play out.

If the port status contains CUE/INIT DONE, then the PLAY command causes the ID specified by the preceding CUE command to start being played out of the selected port. If the system is currently playing a spot and the next spot is already cued when a PLAY command is received, then the current spot will be aborted and replaced by the new spot.

It is recommended that the port should remain in PLAY and continuously output the last frame of video (audio mute) of the ID if the media in play reaches the end of the ID and a STOP or another PLAY command has not been received. This prevents drops to black if a few frame gaps are in between playing of IDs and permits easy verification of the last frame of video after an ID is recorded onto the disk. In all other cases, the disk should output black.

An example of a normal command sequence to play 4 spots is:

PLAY CUE or **CUE WITH DATA** (Spot 1)
    PLAY (Spot 1 start playing)
PLAY CUE or **CUE WITH DATA** (Spot 2) (This is sent right after Spot 1 starts playing)
    PLAY (Spot 1 ends, Spot 2 start playing on next frame)
PLAY CUE or **CUE WITH DATA** (Spot 3) (This is sent right after Spot 2 starts playing)
    PLAY (Spot 2 ends, Spot 3 start playing on next frame)
PLAY CUE or **CUE WITH DATA** (Spot 4) (This is sent right after Spot 3 starts playing)
    PLAY (Spot 3 ends, Spot 4 start playing on next frame)
STOP (Spot 4 ends, Disk is Idle)

If this disk was to play out the next commercial pod and the program material was played out by another device, the PLAY CUE or CUE WITH DATA command would be sent for the first spot in the next commercial pod right after the PLAY command for Spot 4. The STOP command would not be sent. This sequence would keep the disk cued for the next spot at all times.

Nexio Addendum: The VDCP specification for this command includes with this command an optional parameter for a “Prepared File Handle.” This feature is not supported in Nexio servers.

Also, even though the VDCP specification says PLAY should only function if the port status contains CUE/INIT DONE, as of LLM version numbers 6.07.96.25 and 5.07.160, it is possible to put a channel into PLAY when the channel is currently in a variable play state.

10 02 Record

Syntax: **10 02**

Issuing the **RECORD** command will cause the system to begin recording on the next REF interval. It will also clear the CUE/INIT and cueing bits in the port status.

The **RECORD** command may only be issued when the port has been prepared for recording with a **RECORD INIT**. If a **RECORD INIT** command had not been issued prior to the **RECORD** command, an error will result (not in cue/init state) and the port will remain in its former state. If CUE/ INIT DONE, the system will then record for the LENGTH specified in the **RECORD CUE** command. At the end of the record interval, the port will leave the record state and return to the IDLE state.

An example of a command sequence that is used to record new material into the disk system is:

RECORD INIT or **RECORD INIT WITH DATA** (Spot 1)
RECORD (Starts recording Spot 1)
STOP (Ends recording of Spot 1)

If the port status contains CUE/INIT DONE, then the RECORD command causes the ID specified by the preceding **RECORD INIT** command to start recording. An Input Port is the only Port that can add to the contents of the disk system. All recording is done through Input ports.

Recording material into the system requires the sequence **RECORD INIT**, **RECORD**. At any point in time the port can be in one of three states: IDLE, INIT or RECORD. If back to back recording is permitted, then the command sequence may be extended as in the **PLAY** example above.

It is recommended to output the input signal on the corresponding output port (i.e., E.E. to output port 1 if recording on input port 1) if it is not currently selected by a different port.

Nexio Addendum: The VDCP specification for this command includes the ability to record clips back-to-back. This feature is not supported in Nexio servers.

```
10 04                                                    Still
```

Syntax: **10 04**

The **STILL** command causes the currently playing ID to pause.

The last frame played prior to receiving the **STILL** command will continue being displayed. The output port must be in PLAY, or in the CUED state, or an error will be logged. Play of the current ID is resumed by a **CONTINUE** command.

```
10 05                                                    Step
```

Syntax: **10 05**

The **STEP** command causes the currently playing and paused ID in STILL state or in a play state to advance to the next frame and STILL. The output port must be in a PLAY, CUED, or STILL state or an error will be logged. This is equivalent to **JOG** with +1 data.

```
10 06                                                  Continue
```

Syntax: **10 06**

The **CONTINUE** command causes the ID currently in the STILL state or a PLAY state to exit that state and continue playing. The output port must be in a PLAY, CUED, or STILL state or an error will be logged.

```
10 07                                                     Jog
```

Syntax: **10 07 FRAMES**

FRAMES: 1 byte or 4 bytes, indicating number of frames to jump forward or backward

The **JOG** command causes the controlled device to move the specified number of frames forward or backward with respect to its current position. The output port must be in a PLAY, CUED, or STILL state or an error will be logged.

SEND DATA-1 can be either one or four bytes.

A one byte field: contains an 8 bit signed number (2's compliment) indicating the range from +128 to -128 frames.

A four byte field: contains a 32 bit signed number (2's compliment) indicating the range from +2592000 to -2592000 frames. The four byte field is large enough to 'jump' to any relative position in a video file up to 24 hours in length.

Nexio Addendum: An option is available in the Nexio AMP and XS servers that can automatically calculate smooth off-speed play back based on a series of Jog commands sent by the controller. Instead of jumping the specified number of frames forward or backward with every issued command, the server instead calculates the resulting variable play speed based on the number of frames the controller requests over a series of Jog commands. When the resulting play back speed is within an acceptable range, the off-speed play back will include audio scrub.

There is a registry setting that enables the smooth jog feature. It's under the Control branch of the LLM registry. When VdcpJogSmooth = 0, the server reponds to the Jog command as it always has, by treating the command the same of the native protocol command **Jump Forward (C1 01)** and **Jump Backward (C1 02)**. When VdcpJogSmooth = 1, the server implements the smooth jog control, the same as the native protocol command **Incremental Jog (Cx 00)**.

10 08 — Variable Play

Syntax: **10 08 SPEED**
SPEED: 24-bit signed binary 2, indicating a speed value based on scale below

When the **VARIABLE PLAY** command is received, the controlled device will start running in accordance with the speed and direction data defined in SEND DATA-1, SEND DATA-2, and SEND DATA-3. The output port must be in a PLAY, CUED, or STILL state or an error will be logged. The port state will then be in PLAY.

SEND DATA-1, SEND DATA-2, and SEND DATA-3 contain a 24 bit signed binary 2's complement number that represents the direction and speed at which the output signal port should play.

Scale:
0x000000 = still
0x010000 = std play speed forward
0x7F0000 = approximately 127 times std. play speed forward
0xFF0000 = standard play speed reverse
0x800000 = 128 times std. play speed reverse

10 0A — EE Mode

Syntax: **10 0A MODE**
MODE: 1-byte, indicating mode of EE implemented

SEND DATA 1 specifies the EE MODE to put the output in.

When SEND DATA 1 is zero: The output should be put in the EE OFF mode. EE OFF specifies that the input signal is not routed through to the output.

When SEND DATA 1 is a value of one: The output should be put in the EE ON mode. EE ON specifies the input signal is always routed through to the output.

When SEND DATA 1 is a value of two: **T**he output should be put in the EE AUTO mode. EE AUTO specifies the input signal is routed through to the output when the video port is not in play mode, and the playing material is routed through to the output when the video port is in play mode. This is an immediate command that takes effect on the current or selected video port only.

Preset/Select Commands

A0 1D	Rename ID

Syntax: **A0 1D BC XID BC2 NEWXID**
BC: 1-byte unsigned integer, indicating amount of data of the original ID
XID: Extended ID
BC2: 1-byte unsigned integer, indicating amount of data to of the new ID
NEWXID: New Extended ID

This command renames an ID in the video disk from the Original ID to the New ID. The Original ID will no longer exist once the command is executed. Video data does not need to be changed. The disk must put the Original ID in the IDs Delete List, and the New ID in the IDs Added List. It must also set both the IDs Added and IDs Deleted in the port status data.

The data format is: SEND DATA 1 to SEND DATA 8 the Original ID, SEND DATA 9 to SEND DATA 16 New ID.

Note: *Use of the **A0 1D** command bears the format described in the section on Variable Length IDs.*

The disk may set the busy bit, (bit 6, Byte 1 in PORT STATUS 1) begin the operation, and clear the busy bit when the operation is complete. All busy bit rules apply (see STATUS DATA).

Nexio Addendum: Nexio server systems do not support the short ID version of this command, **20 1D**.

20/A0 1F	New Cop

Syntax: **20 1F ID NEWID SOM DUR** or
A0 1F BC XID BC2 NEWID SOM DUR
ID: 8-byte ID
NEWID: 8-byte ID of subclip
SOM: 4-bytes, indicating start timecode of subclip
DUR: 4-bytes, indicating duration of subclip
BC: 1-byte unsigned integer, indicating amount of data of the original ID
XID: Extended ID
BC2: 1-byte unsigned integer, indicating amount of data of the new ID
NEWXID: New Extended ID

This command creates a new ID in the video disk from an existing ID. How this command is implemented in the video disk is device dependent. Ideally no copy of media data will result. A new ID would be entered into the disk's ID table with pointers to the existing media data. Alternately a direct data copy could be done so media quality would not be lost in the encode and decode processes, but this takes more time and storage space.

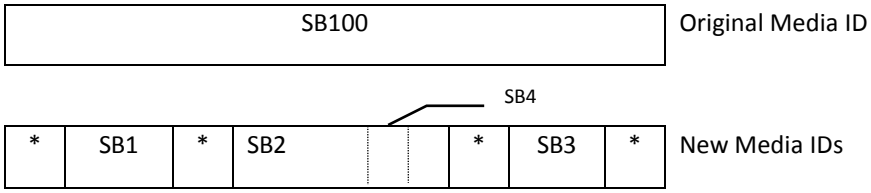
If the first two methods cannot be performed, then a full copy of the media could be executed. In any implementation, the **NEW COPY** command must have no effect on the original ID. It may be played or deleted as if no **NEW COPY** was implemented. The new copy is independent of the original file. It may be played or deleted as if created with the record command. Multiple new copies (including overlapping media data) of an ID may be made. When the original file is deleted, any media data not associated with a valid ID should be released and the disk space available for use.

The original purpose of this command is to allow a recorded program to be segmented into smaller spots that would then be played out with breaks in between, or as individual completed spots.

The data format is: SEND DATA 1 to SEND DATA 8 the original ID, SEND DATA 9 to SEND DATA 16 new ID, SEND DATA 17 to SEND DATA 20 offset to start the new copy within the original ID, and SEND DATA 21 to SEND DATA 24 duration of the new ID. If the offset and duration exceeds the original file length, the new file will start at the file offset and go to the end of the original file.

The disk may set the busy bit (bit 6, Byte 1 in PORT STATUS 1) and begin the operation. The disk may also clear the busy bit when the operation is complete.

All busy bit rules apply (see STATUS DATA).



Note: Use of the **A0 1F** command bears the format described in the section on Variable Length IDs.

Nexio Addendum: The Nexio implementation creates an ID that has pointers referencing the desired media data. No physical copy of the original ID is made. In addition, the original ID cannot be deleted until all the copies derived from it are deleted.

In addition, Nexio does not support creating multiple new copies at the same time. Each copy will have to be created with one command at a time.

20 20 Sort Mode

Syntax: **20 20 MODE**
MODE: 1-byte indicating ID sort order

The **SORT MODE** command is used to set a particular sort order.

DATA-1 contains the sort order. A “0” requests the IDs be returned in alphabetical order (see addendum below) and a “1” requests that the IDs be returned in FIFO order (oldest items listed first, newest items listed last). The default mode must be FIFO upon disk system power up, and must be the mode used if this command is not implemented. This is because a controller can always sort an ID list into alphabetical order, but there is no other way for a controller to determine the oldest spots on the disk for automatically deleting the oldest items that are not needed.

The disk may set the busy bit (bit 6, Byte 1 in PORT STATUS 1) to begin the sort and clear the busy bit when the sort is complete.

Nexio Addendum: Alphabetical sort mode is unavailable for the Nexio server. Its default sort mode is set to FIFO.

20 21 Close Port

Syntax: **20 21 PORT**
PORT: 1-byte indicating port number

The **CLOSE** command is used to break communication to a Signal (audio/video) Port connection established by a preceding **OPEN** command. **PORT** is a number representing the available video ports as in **OPEN**.

SEND DATA 1 contains an 8 bit signed number representing **PORT**.

20 22 Select Port

Syntax: **20 22 PORT**
PORT: 1-byte indicating port number

The **SELECT PORT** command selects a Signal Port from the signal ports that are currently opened by this communications port.

All subsequent commands arriving at the associated RS-422 port will be routed to the assigned Signal Port until another **SELECT PORT** command is received. Only one signal port may be selected by a single communications port at any time. A **CLOSE**, or **SELECT PORT** command following, breaks or closes this selection. **PORT** is a number representing the available I/O signal ports.

SEND DATA 1 contains an 8 bit signed number representing **PORT**.

20/A0 23 Record Init

Syntax: **20 23 ID LENGTH** or
A0 23 BC XID LENGTH
ID: 8-byte ID
LENGTH: 4-bytes, indicating duration of clip
BC: 1-byte unsigned integer, indicating amount of data of the original ID
XID: Extended ID

Issuing the **RECORD INIT** command with a Video Input Port selected causes the system to prepare for recording.

The **RECORD INIT** command consists of the command itself followed by an **ID**, followed by a **LENGTH**. The **ID** is an 8 character identifier. The **LENGTH** is the duration of record in FF.SS.MM.HH (BCD) format. The **RECORD INIT** command may be issued when the signal port is recording, if the disk system supports back-to-back records. In this case, every frame of video is recorded.

Upon the next **RECORD** command, the disk will direct all video data starting on the next frame to the new ID or file name (this feature is required for a video disk to perform a delay box function using a single input port). If the disk system does not support back-to-back recording and the port is not in the IDLE state when a **RECORD INIT** command is received, an error is logged and the port remains in its prior state. The disk system should check if the ID is unique.

If the ID is not unique: An error is logged in the ID Already Exists bit in PORT STATUS byte 3C, and the port is left in the IDLE state.

If the ID is unique: The port is put into the CUE/INIT state while initializing. When initialized and the port is ready to receive a **RECORD** command, then the CUE/INIT DONE bit is set in the port status. The CUE/INIT bit may or may not be cleared, but both must be cleared when a record or stop command is received.

The requirement for the disk to verify that there is enough disk space to record a clip is not needed. The controller can obtain the available space through the SYSTEM STATUS REQUEST and make that decision. As the recording occurs, the controller may delete items to keep disk space available for the current record.

If an output port is selected, an error is generated. An error condition will result in the appropriate bit being set in the port status error byte. When the media file for the ID is available for play out from the disk (this may be as soon as recording has started, or when recording is complete), the ID must be put into the ID ADDED LIST for all communications ports. This includes the port that issued the record command. This action will also cause the IDs ADDED bit to be set in the Status 2 byte of the PORT STATUS for all communications ports.

SEND DATA 1 through SEND DATA 8 contain the ID; SEND DATA 9 through 12 contain the desired length in BCD.

Note: *Use of the **A0 23** command bears the format described in the section on Variable Length IDs.*

Nexio Addendum: The Nexio server does not support back-to-back records.

20/A0 24 — Play Cue

Syntax: **20 24 ID or**
A0 24 BC XID
ID: 8-byte ID
BC: 1-byte unsigned integer, indicating amount of data of the original ID
XID: Extended ID

The **PLAY CUE** command causes the selected port to prepare to play the specified ID.

The video disk should accept the **PLAY CUE** command while playing another ID, cueing the new ID so that it may get a play command any time during the play of the current ID or after the current ID is played out. If the disk is cueing or currently in the cued state, then the previous ID that is cueing or cued should be aborted, and the new ID cued for play out. If the disk is idle when a cue command is received, it should cue the ID for play out and wait for a **PLAY** command. If the ID is not found, then an error occurs and the status returns to its previous state. When the command is received and validated that the disk can cue, the CUE/INIT bit in the port status is set. When the cueing process is complete and the ID is available to play the CUE/INIT DONE bit in the port status should be set.

The **PLAY** command may only be sent when the CUE/INIT DONE bit is set or an error will be logged for the **PLAY** command. When the **PLAY** command is received, the material will begin playing on the next video reference interval. If another **CUE** command is received prior to receiving a **PLAY** command, the previous **CUE** will be aborted and CUE process will begin for the new ID. The cueing bit may be cleared or left active once cued, but both CUE/INIT and CUE/INIT DONE bits must be cleared when a **PLAY** or **STOP** command is received. The port must be a signal output port or an error is logged and the port will remain in its previous state.

It is recommended that once the CUE/INIT DONE bit is set (the disk is cued), and the disk is not playing a previous ID, that the disk blacks until a **PLAY**, **STOP**, or **CUE** command is received. This prevents any incorrect video picture from being played to air if there is a video disk/switcher-timing mismatch. If still video is output, and the switcher switches the video disk to air a few frames early or late, then undesired frames will be seen. Outputting black video will hide this defect.

For previewing purposes, it is desirable to see the first frame of video of a clip that is cued. This could be accomplished either through a user configurable setting in the video disk on a video-port-by-video-port basis (just set on preview ports). This could also be accomplished by allowing the customer to send the **STILL** command when in the CUE DONE state, which would cause the video disk to display the first frame of video.

SEND DATA 1 through SEND DATA 8 contain the ID to be cued for Play. SEND DATA 9 is the optional Prepared File Handle data. SEND DATA 9 must be included if *and only if* the Prepare ID To Play command was issued for this ID. If SEND DATA 9 is included, it indicates which Prepared File Handle to cue. Note that a prepared file may or may not have the optional Duration and Start Of Message data included.

Note: *Use of the **A0 24** command bears the format described in the section on Variable Length IDs.*

20/A0 25 Cue with Data

Syntax: **20 25 ID SOM DUR** or
A0 25 BC XID SOM DUR
ID: 8-byte ID
SOM: 4-bytes, indicating start timecode of clip
DUR: 4-bytes, indicating duration of clip
BC: 1-byte unsigned integer, indicating amount of data of the original ID
XID: Extended ID

This command is similar to the **CUE** command but allows play out of just a part of the ID.

SEND DATA 1 through SEND DATA 8 contains the **ID**, SEND DATA 9 through 12 contains the start position within the ID (play out start) and SEND DATA 13 through SEND DATA 16 contains the duration (play out duration). This command is not permitted on files that have been prepared with the **PREPARE TO PLAY ID** command.

ID BYTE 0		DATA 2 (46H)
ID BYTE 1		DATA 3 (52H)
ID BYTE 2		DATA 4 (45H)
ID BYTE 3		DATA 5 (44H)
ID BYTE 4		DATA 6 (31H)
ID BYTE 5		DATA 7 (20H)
ID BYTE 6		DATA 8 (20H)
ID BYTE 7		DATA 9 (20H)
FRAME X10	FRAME X1	DATA 10 (22H)
SEC X10	SEC X1	DATA 11 (35H)
MIN X10	MIN X1	DATA 12 (07H)
HR X10	HR X1	DATA 13 (00H)
FRAME X10	FRAME X1	DATA 14 (00H)
SEC X10	SEC X1	DATA 15 (30H)
MIN X10	MIN X1	DATA 16 (06H)
HR X10	HR X1	DATA 17 (00H)

Note: Use of the **A0 25** command bears the format described in the section on Variable Length IDs.

20/A0 26 Delete ID

Syntax: **20 26 ID or**
A0 26 BC XID

ID: 8-byte ID
BC: 1-byte unsigned integer, indicating amount of data of the original ID
XID: Extended ID

The **ID DELETE** command is used to remove material from the disk system.

If the ID is in the Get From Archive List still to be transferred to the disk, it should be removed from the Get From Archive List. When the media file for the ID is deleted from the disk, the ID must be put in the IDs DELETED LIST for all communications (including the port that issued the delete command). This will cause the IDs DELETED bit to be set in the Status 2 byte of the PORT STATUS for all communications ports.

SEND DATA 1-8 contain the ID name to be deleted.

Upon receipt of an **ID DELETE** command, the system will check if the ID is present in the system. If the ID is not found, then an error will be logged (the "ID Not Found" bit being set in the PORT STATUS error byte 3B).

If the ID is present, the system will check to see if the ID is currently active on any signal port.

If the ID is active on a signal port, an error will be logged in the Port Playing/Active bit of PORT STATUS byte 3C. If the ID is not active on a signal port, the system will check to see if the ID has been delete protected.

If the ID is delete protected, an error will be logged in the ID Delete Protected error bit in PORT STATUS byte 3B, and the ID is not deleted, otherwise the ID will be deleted from the system.

Note: *Use of the **A0 26** command bears the format described in the section on Variable Length IDs.*

20 2B — % to Signal Full

Syntax: **20 2B DATA**
DATA: 1-byte indicating percentage

The **% TO SIGNAL FULL** command alerts operator to % of disk space available.

SEND DATA 1 is a number between 0 and 100 that represents the % of disk space still available when the full flag bit should set in the system status 3 DISK STATUS request. This should be set between 1 and 10% for normal operation. The automation software will try to delete an unused media file as long as the disk full bit is set. By keeping the size of the disk space empty, (slightly greater than the longest normal spot put into the disk), then the disk full error will rarely occur when a record init command is issued. This maintains system efficiency and disk space utilization.

20/A0 2C — Record Init with Data

Syntax: **20 2C ID SOM DUR** or
A0 2C BC XID SOM DUR
ID: 8-byte ID
SOM: 4-bytes, indicating start timecode of clip
DUR: 4-bytes, indicating duration of clip
BC: 1-byte unsigned integer, indicating amount of data of the original ID
XID: Extended ID

This **RECORD INIT WITH DATA** command has all the features and requirements of the **RECORD INIT** command with the following changes:

The **ID** may already exist on the disk (see addendum below).

If the **ID** exists on the disk, the dub over starts at the start position given in SEND DATA 9 through 12, and has the duration given in SEND DATA 13 through 16 (see addendum below).

If the **ID** does not exist in the disk, a new media file is recorded for the **ID** and if the disk keeps timecode internally, the start position given in SEND DATA 9 through 12 should be the timecode of the first frame of video.

SEND DATA 1 through SEND DATA 8 contain the **ID**, SEND DATA 9 through 12 contain the desired start position within ID, SEND DATA 13 through 16 contain the desired length in BCD

Note: *Use of the **A0 2C** command bears the format described in the section on Variable Length IDs.*

Nexio Addendum: Cases 1 and 2 are not supported by the Nexio server. As of the Nexio 5.7.0 Software Release, there is a new option with Case 3 which will automatically unload the ID after it completes the recording. This option is enabled in the LLM registry in the Control Branch. The DWord registry setting is "VdcpUnloadRecordComplete". Set this registry to "1" to enable the option to unload clips at the end of a recording whose duration was preset. By default, this option is turned off, set to "0", so as not to interfere with existing VDCP controller logic.

20/A0 2E — System Delete ID

Syntax: **20 2E ID or**
A0 2E BC XID
ID: 8-byte ID
BC: 1-byte unsigned integer, indicating amount of data of the original ID
XID: Extended ID

The **SYSTEM DELETE ID** command is used to remove material from the disk system and all other disk systems on its network, or any other disk systems that it communicates with. This command is to be used in situations where several disk systems may communicate with one another and transfer files between them.

The disk system that receives this command must broadcast it on its network or communication channel to other disk systems it communicates to. All other disk systems that receive the command must act on it in the same way. When a piece of material is no longer needed, it must be removed from *all* disk systems in the network. Removing the material from all of the systems will ensure that only the newest version of an **ID** will be used on any disk in the system. In a typical configuration, there may be one or more disks that material is dubbed into originally that are considered the library disk(s).

There would also be one or more playout disks on the network that would transfer the needed files from the library disks in advance of when they need to play them. The controller of the playout disks would use the **GET FROM ARCHIVE** command to inform the playout disk to transfer the IDs it needs from other disks on the network. The controller of the playout disk would also use **DELETE ID (20 26)** to remove IDs only from the playout disk when it needs to make room for additional material. The **SYSTEM DELETE ID** command would only be used by the controller who dubs material into the library disks and manages the deletion of the library material. This command only acts on disk systems, and should not delete an ID from an Archive storage device. Deleting material from an Archive storage device can only be done through **Delete From Archive (00 14)**.

Upon receipt of a **SYSTEM DELETE ID** command, the system will check if the **ID** is present in the system. If it is not, then an error will be logged (the "ID Not Found" bit being set in the PORT STATUS error byte 3B). If the **ID** is present, the system will check to see if the **ID** is currently active on any signal port. If the **ID** is currently active, an error will be logged in the Port Playing/Active bit of PORT STATUS byte 3C.

If the **ID** is not currently active, the system will check to see if the **ID** has been delete protected. If the **ID** is delete protected, an error will be logged in the ID Delete Protected error bit in PORT STATUS byte 3B, and the **ID** is not deleted; otherwise, the **ID** will be deleted from the system. If the **ID** is in the Get From Archive List still to be transferred to the disk, it should be removed from the Get From Archive List. When the media file for the **ID** is deleted from the disk, the ID must be put in the IDs DELETED LIST for all communication ports (including the port that issued the delete command). This action will cause the IDs DELETED bit to be set in the Status 2 byte of the PORT STATUS for all communication ports.

SEND DATA 1-8 contain the ID name to be deleted.

Note: *Use of the **A0 2E** command bears the format described in the section on Variable Length IDs.*

Nexio Addendum: This command only affects IDs on the server system that the controller is connected to.

20 34 Audio In Level

Syntax: **20 34 DATA**
DATA: 2-bytes indicating audio level

Issuing the **AUDIO IN LEVEL** command selects the audio LEVEL for the input audio record signal.

SEND DATA 1 and SEND 2 contain the LEVEL value. (Device Dependent)

20 35 Audio Out Level

Syntax: **20 35 DATA**
DATA: 2-bytes indicating audio level

Issuing the AUDIO OUT LEVEL command selects the audio LEVEL for the output signal.

SEND DATA 1 and SEND DATA 2 contains the LEVEL value. (Device Dependent)

20 39 Select Input

Syntax: **20 39 MODE**
MODE: 1-byte indicating input format

Issuing the **SELECT INPUT** command selects which input format and connector will be used for encoding from the input port selected. **MODE** is defined as 0 (OFF), 1 (Composite), 2 (S-VIDEO), 3 (YUV), 4 (D1). OFF provides BLACK with sync. An error condition will result in the appropriate bit being set in the port status error byte(s).

SEND DATA 1 contains a bit map specifying the input MODE.

Table 3-87: Select Input Bitmap

BIT7	BIT6	BIT5	BIT4	BIT3	BIT2	BIT1	BIT0
---	---	---	D1	YUV	S-Video	Composite	Off

Nexio Addendum: The input format for a particular Nexio server is limited to the capabilities of the individual server. For example, the TXS server accepts SDI input only, no matter what value is set here.

20 43 Disk Preroll

Syntax: **20 43 FRAMES SECONDS**
FRAMES: 1-byte indicating the number of preroll frames desired
SECONDS: 1-byte indicating the number of preroll seconds desired

The DISK PREROLL command allows the controlling devices to specify to the controlled disk system that it should always begin the play or record function a set time after receiving the command.

The Disk Preroll is designated as the time that a disk system should delay the operation after a PLAY or RECORD command. The disk should use the value in the command until changed by another Disk Preroll command. The default should be zero, unless otherwise specified in the disk manufacturer's specifications. A value of zero is interpreted as: Start The Function on the frame after receiving a Play or Record command. Most devices have a fixed latency time to process commands consistently. This command allows the controlling device to specify a value that the device can handle and maintain frame accuracy under all conditions, or a greater value.

SEND DATA 1 is the preroll frames, SEND DATA 2 is the preroll seconds in decimal.

Nexio Addendum: By default, a Nexio Server does not support the Disk Preroll command. For those controllers that depend on support for this command, there is an option to enable this command via an LLM registry in the Control branch. The DWord registry setting is VdcpDiskPrerollMode. Set to "1" to enable. The actual preroll delay of the server will be based on this Disk Preroll value in the registry plus any existing "VdcpPlayLatency" registry setting already existing on the affected server.

20 50 Copy File To

Syntax: **20 50 DATA** or
A0 50 DATA

This command will cause the disk systems to transfer the video file identified by the ID to be copied from the source disk system to the destination disk system(s).

This command should only be implemented on systems where files are to be transferred between multiple video disks and/or archive systems that may be linked together on a network. The Source and Destination are a byte for each of the disk systems for the action. Multiple destination disk systems may be specified in SEND DATA 3 by placing a one byte address for each disk identifier and the command length byte will be set accordingly.

The disk systems are responsible for the complete file transfer when they have available time and bandwidth to perform the transfer. Since requests may come in faster than they can be performed, the disk system must accept requests at any time and queue up file transfer requests as they come in.

When the file transfer is complete, the destination system(s) should indicate that the ID has been added by the IDs ADDED bit in the Port Status, and add the ID to its IDs ADDED LIST. If the command cannot be performed, then the ID NOT TRANSFERRED error bit in the Port Status should be set. If the ID does not exist, then the ID NOT FOUND bit should be set. If the ID exists in all of the destination disks, then the ID ALREADY EXISTS bit should be set. The video disk system that receives this command may or may not be one of the source or destination video disks.

SEND DATA 1 is the ID, SEND DATA 2 is the source, SEND DATA 3 is the Destination.

Note: *Use of the **A0 50** command bears the format described in the section on Variable Length IDs.*

Nexio Addendum: Nexio systems do not support multiple destination transfers. A File Transfer Agent is required on every system that makes use of the FTP Copy commands.

20 51 Delete File From

Syntax: **20 51 DATA** or
A0 51 DATA

This command will cause the disk system to delete the video file identified by the ID from the destination disk system(s).

This command should only be implemented on systems where multiple video disks and/or archive systems may be linked together on a network where files are to be transferred between them. The Destination is a byte for each of the disk system(s) for the action. Multiple destination disk systems may be specified in SEND DATA 3 by placing a one byte address for each disk identifier and the command length byte will be set accordingly.

The disk systems are responsible for the complete file removal when they have available time. The disk system must accept requests at any time and queue up delete requests, as requests may come in faster than they can be performed. When the file removal is complete, the destination system(s) should indicate that the ID has been deleted by the IDs DELETED bit in the Port Status, and add the ID to its IDs DELETED LIST. If the command can not be performed then the CUE OR OPERATION FAILED error bit in the Port Status should be set. If the ID does not exist, then the ID NOT FOUND bit should be set. The video disk system that receives this command may or may not be one of the source or destination video disks.

SEND DATA 1 is the ID; SEND DATA 2 is the Destination.

Note: *Use of the **A0 51** command bears the format described in the section on Variable Length IDs.*

Nexio Addendum: Nexio systems do not support multiple destination transfers. A File Transfer Agent is required on every system that makes use of the FTP Copy commands.

20 52 Abort Copy File To

Syntax: **20 52 DATA** or
A0 52 DATA

This command will cause the disk systems to ABORT the transfer of a video file that was identified by the ID to be copied from the source disk system to the destination disk system(s) that was previously sent.

This command should only be implemented on systems where multiple video disks and/or archive systems may be linked together on a network where files are to be transferred between them. The Source and Destination are a byte for each of the disk systems for the action. The ID, source, and destination must match the original command to abort a previous copy command. Multiple destination disk systems may be specified in SEND DATA 3 by placing a one byte address for each disk identifier. The command length byte will be set accordingly.

The disk system must accept requests at any time. If the command can not be performed then the CUE OR OPERATION FAILED error bit (or another more appropriate error bit) in the Port Status should be set. The video disk system that receives this command may or may not be one of the source or destination video disks.

SEND DATA 1 is the ID, SEND DATA 2 is the source, SEND DATA 3 is the Destination.

Note: *Use of the **A0 51** command bears the format described in the section on Variable Length IDs.*

Nexio Addendum: Nexio systems do not support multiple destination transfers. A File Transfer Agent is required on every system that makes use of the FTP Copy commands.

Sense Commands

30 01	Open Port

Syntax: **30 01 PORT LOCK**
PORT: 1-byte, indicating port number
LOCK: 1-byte, always set to 0

The Signal Ports consist of audio and video channels as configured by the device. Any signal port can be controlled from any RS-422 control port with the following Port assignment commands; **OPEN**, **CLOSE** and **SELECT**. Only one communications port can have a given signal port open at a given time. The system commands are organized with reference to the Signal Port that they effect. The ports consist of SIP (Signal Input Ports, range -1 to -127) and SOP (Signal Output Ports range 1 to 127).

SEND DATA 1 contains an 8 bit signed magnitude number representing **PORT**. PORT 0 is not used. SEND DATA-2 contains the port security mode. For Nexio servers, this must always be set to 0 for the unlocked mode.

PORT is a signed number representing the available signal ports, for example: -1(SIP1), 1(SOP1), 2(SOP2). This **PORT** number is also used with the **PORT SELECT** and **CLOSE** commands

RETURN DATA 1:

- An OPEN request (in either mode) for an unopened (available) signal port: will result in a 1 (port granted) being returned in RETURN DATA-1.
- An OPEN request (in either mode) for an opened and LOCKED signal port: will result in a 0 (port denied) being returned in RETURN DATA-1.
- An OPEN request (in either mode) for a port that has been opened in the UNLOCKED mode: will result in a 1 (port granted) being returned to the requesting communications port and the PORT BUSY bit being set in the port status until the port is reset to its idle state. The original communications port will have the INVALID PORT bit set in the port error status.

An example of a normal command sequence, start up of a controller, re-initializing port communications, or establishing a session with a signal port:

- OPEN XX (Opens port XX, may be an input or an output port)
- SELECT XX (Selects the port just opened)
- (performs play back or recording for just a few seconds, or many months)
- CLOSE XX (Closes the port opened above)

This cycle may be repeated many times every few minutes (such as when recording items into the disk system for preview), or may last many months (such as on-air play out while other ports are used to record new items into the disk and do disk media management).

30/B0 02 — ID Next

Syntax: **30 02** or
B0 02

The **NEXT** command is used to transfer any remaining IDs in groups of up to ten. It has the same format as **LIST** commands and **NEXT** is called repeatedly until all IDs have been transferred. See the **LIST** command **(30 11)** for more details.

Note: *Use of the **B0 02** command bears the format described in the section on Variable Length IDs.*

30 05 — Port Status

Syntax: **30 05 DATA**
DATA: 1-byte indicating which status information to return

DATA 1 contains a bitmap specifying which status should be returned. This command returns to the controller the status bytes specified for the selected video port preceded by the bit map.

Nexio Addendum: Shaded boxes denote unsupported bits.

DATA 1 - Bit map for PORT STATUS

	EXTENDED PS2	EXTENDED PS3	PORT STATUS 5	PORT STATUS 4	PORT STATUS 3	PORT STATUS 2	PORT STATUS 1

The port status items available and their contents are shown in the Figures below:

Status 1 - State and Flag Status

CUE/INIT-DONE	PORT BUSY	VARIABLE PLAY	JOG	STILL	PLAY OR RECORD	CUE/INIT	IDLE
PORT ID							

Status 2 – Short Option- Port HW\Media Status

AUDIO OVERLOAD	NO AUDIO INPUT	NO VIDEO INPUT	NO REF INPUT	IDs ADDED TO ARCH.	IDs DELETED	IDs ADDED	PORT DOWN

Status 2 – Extended Option- Port HW\Media Status

AUDIO OVERLOAD	NO AUDIO INPUT	NO VIDEO INPUT	NO REF INPUT	IDs ADDED TO ARCH.	IDs DELETED	IDs ADDED	PORT DOWN
							NO TIMECODE INPUT

Status 3 – Short Option- Port Error Status

NOT SUPPORTED	CMD WHILE BUSY	DISK FULL	COMMAND QUEUE FULL	WRONG PORT TYPE	INVALID PORT	ILLEGAL VALUE	SYSTEM ERROR
ID DELETE PROTECTED	ID TRANS-FERRED	ID NOT TRANS-FERRED	ID STILL PLAYING	ID STILL RECORDING	ID ALREADY EXISTS	ID NOT FOUND	INVALID ID
SYSTEM REBOOTED	NETWORK ERROR	CUE OR OPERATION FAILED	PORT NOT ACTIVE	PORT PLAYING /ACTIVE	PORT NOT IDLE	CUE NOT DONE	NOT IN CUE/ INIT STATE

Status 3 – Extended Option- Port Error Status

NOT SUPPORTED	CMD WHILE BUSY	DISK FULL	COMMAND QUEUE FULL	WRONG PORT TYPE	INVALID PORT	ILLEGAL VALUE	SYSTEM ERROR
ID DELETE PROTECTED	ID TRANSFERRED	ID NOT TRANSFERRED	ID STILL PLAYING	ID STILL RECORDING	ID ALREADY EXISTS	ID NOT FOUND	INVALID ID
SYSTEM REBOOTED	NETWORK ERROR	CUE OR OPERATION FAILED	PORT NOT ACTIVE	PORT PLAYING/ ACTIVE	PORT NOT IDLE	CUE NOT DONE	NOT IN CUE/ INIT STATE
						EXCESSIVE DRIFT	NO AUDIO OR VIDEO
							PVW READY

Status 4 - Port Settings

			D1	YUV	S-VIDEO	COMPOSITE	OFF

Status 5 – Video Compression Types Supported

NUMBER OF VIDEO TYPES THIS PORT SUPPORTS THAT ARE LISTED FOLLOWING THIS BYTE
TYPE X
TYPE Y
TYPE Z

PORT STATUS DATA DESCRIPTION:

Status 1 - State and Flag Status

Byte 1, Bit 0: IDLE - The system is in the IDLE state. The output is black and there is no signal port activity.
Byte 1, Bit 1: CUE/INIT - The system is in the cue or record init state (cueing or initializing).
Byte 1, Bit 2: PLAY/RECORD - The system is in the play or record state.
Byte 1, Bit 3: STILL - The system is in the Still state.
Byte 1, Bit 4: JOG - The system is in the jog state.
Byte 1, Bit 5: VARIABLE PLAY – Can be set in addition to the PLAY/RECORD bit, but can not be set alone. (See addendum below)
Byte 1 Bit 6: PORT BUSY - The system is in the busy state and can only accept immediate commands and status requests: STOP, PLAY, RECORD, STILL, STEP, CONTINUE, PORT STATUS, SYSTEM STATUS.
Byte 1 Bit 7: CUE/INIT-DONE - The play cue or record init has been completed. The signal port can now accept a PLAY or RECORD command.
Byte 2 Bits 0-7: PORT ID - The port ID currently selected. If the port returned is not the port that the controller last selected, then the controller needs to (1) close the incorrect port being returned (if not zero) and (2) open and select the needed port and re-initialize it. (See addendum below)

Nexio Addendum: Because setting the Variable Play bit is optional, if a controller does not use this function, there is a registry to disable the Nexio server's setting of this bit. The DWORD setting ***LouthVariablePlayBit*** resides in the LLM's ***Control*** registry branch**.** Its default value is "1," which enables the Variable Play bit. If set to "0," the server will not show the Variable Play bit.

As of the Nexio 6.5.2 Software Release, the Port ID byte will return a value of "0" if the channel is an input port and the LLM has entered a play-only state (and thus the input channel cannot record).

Status 2 - Port Hardware\Media Status

Byte 1, Bit 0: PORT DOWN - The selected port is inoperative. (See addendum below)
Byte 1, Bit 1: IDS ADDED - New IDs have been added to the disk system by recording or transferring from an archive system, and the controller of the port has not yet asked for that list of the IDs.
Byte 1, Bit 2: IDS DELETED - IDs have been deleted from the disk and the controller of the port has not yet asked for that list of IDs.
Byte 1, Bit 3: IDS ADDED TO ARCH. - New IDs have been added to an archive system connected to the disk system and the controller of the port has not yet asked for that list of the IDs.
Byte 1, Bit 4: NO REF INPUT - The system has no input reference.
Byte 1, Bit 5: NO VIDEO INPUT - The port has no video input (input port only).
Byte 1, Bit 6: NO AUDIO INPUT - The port has no audio input (input port only).
Byte 1, Bit 7: AUDIO OVERLOAD - The audio level in is beyond limit (input port only).

Extended option:
Byte 2, Bit 0: NO TIMECODE INPUT - The input does not have timecode (input port only).

Nexio Addendum: As of the Nexio 6.5.2 Software Release, the Port Down bit will be set high if the LLM has entered a play-only state. The bit will remain high until the LLM is restarted and the issue is corrected.

Status 3 - Port Error Status

All of the error bits are to be set for the appropriate port when they occur. All bits should be cleared as soon as the Port Status is read by the controller. Thus the controller will see a bit set only once for each occurrence of the condition.

Byte 1, Bit 0: SYSTEM ERROR - The system has detected a functional error. (See addendum below)
Byte 1, Bit 1: ILLEGAL VALUE - The system has received a command with an illegal value, (e.g., NEW COPY, SORT MODE, CLOSE PORT, SELECT PORT, RECORD INIT, % TO SIGNAL FULL, VIDEO COMPRESSION RATE, AUDIO SAMPLE RATE, AUDIO COMPRESSION RATE, AUDIO IN LEVEL, AUDIO OUT LEVEL, SELECT OUTPUT, SELECT INPUT, RECORD MODE, SC ADJUST, HORIZONTAL POSITION ADJUST, OPEN PORT.) - Command Ignored.
Byte 1, Bit 2: INVALID PORT - The communications port has selected a signal port that it may not open because the port is already open and locked or it is an invalid port number. - Signal port commands will not be executed.
Byte 1, Bit 3: WRONG PORT TYPE - The controlling device has issued a command not applicable to the open port , (e.g.,
RECORD, RECORD INIT, FREEZE, UNFREEZE to a signal output port or PLAY CUE, CUE WITH DATA, PLAY, STILL STEP CONTINUE JOG, VARIABLE PLAY to a signal input port.) - Command Ignored.
Byte 1, Bit 4: COMMAND QUEUE FULL - Not Used.
Byte 1, Bit 5: DISK FULL - The disk is unable to store any more audio/video data, (e.g., RECORD INIT when not enough storage space to record for duration specified in RECORD INIT command.) - Command Ignored.
Byte 1, Bit 6: CMD WHILE BUSY - A command, other than an Immediate Command was issued while the busy bit was set. – Command Ignored.
Byte 1, Bit 7: NOT SUPPORTED - A command was issued that is not supported by the device. Any optional command. – Command Ignored.
Byte 2, Bit 0: INVALID ID - An invalid ID was specified in a command with an ID parameter, (e.g., PLAY CUE, CUE WITH DATA, NEW COPY, DELETE, GET FROM ARCHIVE, SEND TO ARCHIVE, DELETE FROM ARCHIVE, DELETE PROTECT, UNDELETE PROTECT, ID SIZE REQUEST.) - Command Ignored.
Byte 2, Bit 1: ID NOT FOUND - The ID was not found, (e.g., PLAY CUE, CUE WITH DATA, NEW COPY, DELETE, GET FROM ARCHIVE, SEND TO ARCHIVE, DELETE FROM ARCHIVE, DELETE PROTECT, UNDELETE PROTECT, ID SIZE REQUEST.) - Command Ignored.
Byte 2, Bit 2: ID ALREADY EXISTS - An ID specified in RECORD INIT was already in the system. - Command Ignored.
Byte 2, Bit 3: ID STILL RECORDING - A command was issued for an ID while that ID was still recording that cannot be performed until that ID is done recording, (e.g., PLAY CUE, CUE WITH DATA, DELETE PROTECT ID,

NEW COPY, DELETE, SEND TO ARCHIVE, GET FROM ARCHIVE, ID SIZE REQUEST.) – Command Ignored.

Byte 2, Bit 4: ID STILL PLAYING - A DELETE command was issued while the ID was playing. - Command Ignored.

Byte 2, Bit 5: ID NOT TRANSFERRED - A PLAY CUE or CUE WITH DATA command was issued before ID has been transferred from Archive. - Command Ignored.

Byte 2, Bit 6: ID TRANSFERRED - A GET FROM ARCHIVE command was issued for an ID that is already in the disk. – Command Ignored.

Byte 2, Bit 7: ID DELETE PROTECTED - A DELETE command was issued for an ID that is delete protected. - Command Ignored.

Byte 3, Bit 0: NOT IN CUE / INIT STATE - A command was issued that required the system to be in the cue state (cueing state - no commands require the disk to be in this state currently).

Byte 3, Bit 1: CUE NOT DONE - A command was issued that required the system to be in the cue/init done state, (e.g., PLAY, RECORD, JOG, VARIABLE PLAY, STEP, CONTINUE, FREEZE, UNFREEZE.) - Command Ignored.

Byte 3, Bit 2: PORT NOT IDLE - A command was issued that required the system to be in the idle state, (e.g., RECORD INIT in some disks.) - Command Ignored.

Byte 3, Bit 3: PORT PLAYING / ACTIVE - A command was issued that required the signal port to not be in the play state. (No command required, not to be in the play state at this time.)

Byte 3, Bit 4: PORT NOT ACTIVE - A command was issued that required the signal port to be playing, recording, or active (not idle), (e.g., STILL, STEP, CONTINUE, JOG, VARIABLE PLAY, POSITION REQUEST, ACTIVE ID REQUEST, PLAY, RECORD, FREEZE, UNFREEZE, STOP.) - Command Ignored.

Byte 3, Bit 5: CUE OR OPERATION FAILED - A CUE command or other command that has been ACKed and started failing for some unknown reason. - Command will not be executed properly.

Byte 3, Bit 6: - NETWORK ERROR - Not Used.

Byte 3, Bit 7: SYSTEM REBOOTED - Indicates the disk system was rebooted. The controller needs to do a PORT OPEN and SELECT command and any other start up command sequence.

Note: The error bits should be cleared after the port status is read by the controlling device.

Nexio Addendum: As of the Nexio 6.5.2 Software Release, the SYSTEM ERROR bit will be set high if the LLM has entered a play-only state. The bit will remain high until the LLM is restarted and the issue is corrected.

Extended option:

Byte 4, Bit 0: No audio or video of any nature (unrecorded, disk errors, decode errors)

Byte 4, Bit 1: Excessive drift

Byte 5, Bit 0: Clip stacked in Preview is ready to play

Status 4 - Port Settings

One or more bits (if multiple outputs are active) should be set to a “1” to indicate the format of the active ports.

Status 5 – Video Compression Type

As of the Nexio 5.7.0 Software Release, Nexio platforms support an optional configuration that provides the video compressions and video resolutions for each server channel. This option must be enabled in the LLM registry: ..\LLM\Control\VideoStatusEnable. When set to the default value of “0,” the LLM reports Port Status as it always has. When set to a value of “1,” the LLM will provide informatio about what types of clips may be loaded in a particular channel. That information is sent as a series of bytes.

Byte 1: Byte Count: The total bytes of data to follow. If 0, feature not supported.

Byte 2: LSB Nibble: First video format supported by the port.

Byte 2: MSB Nibble: Video resolutions supported by the port.

Byte 3: LSB Nibble: Second video format supported by the port.

Byte 3: MSB Nibble: Same video resolutions supported by the port.

Byte *N:* LSB Nibble: Last video format supported by the port.

Byte *N*: MSB Nibble: Same video resolutions supported by the port.

The first byte returned is a byte count identifying the number of bytes to follow. A return value of 0 indicates the feature is not currently enabled or implemented. The remaining bytes are a series of one byte values indicating which formats and resolutions are supported. The format and resolution indicators are split into the two nibbles of each byte. Information about the structure of these bytes is detailed as part of the **ID Request (30 16)** command.

30 06 — Position Request

Syntax: **30 06 DATA**
DATA: 1-byte indicating which mode is desired

The **POSITION REQUEST** query returns the current position "timecode" or time remaining within the ID which is currently playing on the selected port.

The selected port must be in PLAY, RECORD, CUED, OR STILL state or an error will be logged. An error condition will result in the appropriate bit being set in the port status error bytes. The POSITION/TIME returned is in RETURN DATA 2-5 in FRAMES, SEC, MIN, HOURS BCD format is preceded by RETURN DATA 1 the time type.

SEND DATA 1/RETURN DATA 1 contains the time type to be returned.

- 0 requests the time remaining
- 1 requests the (SOM-based) time code of the current frame position
- 2 requests the (zero-based) time code of the current frame expressed as an offset from the start of the ID.

Nexio Addendum: The VDCP protocol use of this command was revised in the August 2000 document, effectively reversing the implementation in place by Leitch. To provide backward compatibility for controllers still making use of the 1999 protocol definition (interpretation), a new setting was added to differentiate between the two command interpretations.

Use the NxManager utility and navigate to the Control setup tab or change the following registry values for each server in the system:

- HKCU/Software/Asc Audio Video/LLM/Control/Protocol0
- HKCU /Software/Asc Audio Video/LLM/Control/Protocol1
- HKCU /Software/Asc Audio Video/LLM/Control/Protocol2
- HKCU /Software/Asc Audio Video/LLM/Control/Protocol3

Set this value to "Harris" for the 2000 interpretation of the command, as defined above. Or change it to “Louth” for the 1999 interpretation:

SEND DATA 1/RETURN DATA 1 contains the time type to be returned.

- 0 requests the time remaining
- 1 requests position (zero-based as an offset from the start of the ID)
- 2 requests the time code of the current frame (SOM-based)

30/B0 07 — Active ID Request

Syntax: **30 07** or
B0 07

This command returns information to the controller about whether a queried port is active (an active port is one that is either recording, playing, cued or cueing), and what the active ID is. This query does not effect the output of the system.

The format returns the active status in RETURN DATA 1 and the ID in RETURN DATA 2-9. In RETURN DATA 1, a "1" indicates that the port is active and "0" indicates that the port is not active. The ID is an eight character identifier. If the port is not currently active, then only a "0" is returned.

Nexio Addendum: As of the Nexio 7.0 Software Release, a new option was created whereby the **Active ID Request** command will return positive data when a clip is in a **STILL** state in the channel. There is a DWord registry setting in LLM\Control called VdcpActiveIdMode. "1" means channels with clips in still are active. "0" is the legacy behavior which reports the channel is not active.

Note: *Use of the **B0 07** command bears the format described in the section on Variable Length IDs.*

30 08 Device Type Request

Syntax: **30 08**

The **DEVICE TYPE REQUEST** command is used to request the specifications of the Controlled Device.

The response to this command is a 16-byte (maximum) data message advising of the specifications of the CONTROLLED DEVICE. The first N bytes will be the manufacturer ID followed by a colon "**:**"

Nexio Addendum: For Nexio servers, the command returns data in the format of "ASC:*xxxx*", where *"xxxx"* is 2 bytes as defined in the following table:

Table 3-88: Server Types

Code	Description
0xAA11	2-channel server, FILM
0xAA12	2-channel server, NTSC
0xAA13	2-channel server, PAL
0xAA15	2-channel server, FILM, VBI
0xAA16	2-channel server, NTSC, VBI
0xAA17	2-channel server, PAL, VBI
0xAA31	4-channel server, FILM
0xAA32	4-channel server, NTSC
0xAA33	4-channel server, PAL
0xAA35	4-channel server, FILM, VBI
0xAA36	4-channel server, NTSC, VBI
0xAA37	4-channel server, PAL, VBI

30 10 System Status Request

Syntax: **30 10 DATA**
DATA: 1-byte bitmap indicating the status information requested

DATA 1 contains a bitmap specifying which bytes should be returned. This command returns to the controller information about the MAIN storage system. DATA 1 is the bit map indicating which system information is returned.

DATA 1 Bit map for SYSTEM STATUS

	Extended SS1 & SS2	SYSTEM STATUS 6	SYSTEM STATUS 5	SYSTEM STATUS 4	SYSTEM STATUS 3	SYSTEM STATUS 2	SYSTEM STATUS 1

The EXTENDED SS1 & SS2 bit is set true/false to cause the SYSTEM STATUS 1 and SYSTEM STATUS 2 bits to select the extended/short options respectively of the System Status 1 and 2 fields.

Nexio Addendum: To provide backward compatibility for controllers who do not support the extended options for Storage Time Remaining, there are two registry settings in HKCU/Software/asc audio video/llm/control: one is a DWORD value called ***Enable3X10Fix;*** and the other is a string value called ***3X10HardReturn***.

- If ***Enable3X10Fix*** =0, the server returns the old 4-byte format. This provides no support for the extended option for Storage Time Remaining.
- If ***Enable3X10Fix*** =1, the server returns the extended format for storage time remaining and largest contiguous block.
- If ***Enable3X10Fix*** =2, the server returns the 4 byte format as specified in the ***3X10HardReturn*** setting.
- If ***Enable3X10Fix*** =3, the server returns the extended format as specified in the ***3X10HardReturn*** setting.
- If the registry setting does not exist, the result is the same as if the setting were ***Enable3X10Fix*** =1.

3X10HardReturn is a hard-coded value in the format of HHHH:MM:SS:FF. It is only used when ***Enable3X10Fix*** is greater than "1". If the ***3X10HardReturn*** setting does not exist the value is defaulted to 9999:59:59:2x, where x=9 for NTSC and x=4 for PAL.

Shaded boxes below denote unsupported bits within the Leitch server system.

System Status 1 - STORAGE TIME REMAINING
TOTAL TIME REMAINING

FRAMES X 10	FRAMES X 1
SECONDS X 10	SECONDS X 1
MINUTES X 10	MINUTES X 1
HOURS X 10	HOURS X 1

LARGEST CONTIGUOUS BLOCK

FRAMES X 10	FRAMES X 1
SECONDS X 10	SECONDS X 1
MINUTES X 10	MINUTES X 1
HOURS X 10	HOURS X 1

System Status 1 Extended Option

STORAGE TIME REMAINING
TOTAL TIME REMAINING

FRAMES X 10	FRAMES X 1
SECONDS X 10	SECONDS X 1
MINUTES X 10	MINUTES X 1
HOURS X 10	HOURS X 1
HOURS X 1000	HOURS X 100
HOURS X 100000	HOURS X 10000

LARGEST CONTIGUOUS BLOCK

FRAMES X 10	FRAMES X 1
SECONDS X 10	SECONDS X 1
MINUTES X 10	MINUTES X 1
HOURS X 10	HOURS X 1
HOURS X 1000	HOURS X 100
HOURS X 100000	HOURS X 10000

System Status 2

NUMBER OF IDs STORED
NUMBER OF IDs STORED - MS BYTE
NUMBER OF IDs STORED - LS BYTE

System Status 2 Extended Option

NUMBER OF IDs STORED
NUMBER OF IDs STORED - MS BYTE
NUMBER OF IDs STORED - BYTE 2
NUMBER OF IDs STORED - BYTE 3
NUMBER OF IDs STORED - LS BYTE

System Status 3 - DISK STATUS

				REMOTE CONTROL DISABLED	DISK DOWN	SYSTEM DOWN	DISK FULL

System Status 4

SUBSYSTEM STATUS (User defined)							
						ARCHIVE FULL	ARCHIVE AVAILABLE
				SYSTEM OFFLINE STORAGE FULL	LOCAL OFFLINE STORAGE FULL	SYSTEM OFFLINE STORAGE AVAILABLE	LOCAL OFFLINE STORAGE AVAILABLE

System Status 5

STANDARD TIME	
FRAMES X10	FRAMES X1
SECONDS X10	SECONDS X1
MINUTES X10	MINUTES X1
HOURS X10	HOURS X1

System Status 6

SIGNAL FULL LEVEL
% TO SIGNAL FULL LEVEL

The Total Time Remaining

The Total Time Remaining is an estimate of the amount of video data that could be stored on the available disk space without consideration of fragmentation or partitions.

The Largest Contiguous Block is the amount of time the largest single recording would currently succeed for. These times are best estimates based on default or current compression settings and typical video streams. They are not precise, but should be a reasonable enough estimate to allow a controller to detect when the amount of space on the video disk has fallen below a threshold, and something should be deleted to allow enough room to continue recording.

Total Time Remaining and Largest Contiguous Block will be the same value if the disk system has no partition or fragmentation issues that prevent a recording from using all available disk space. The use of disk partitions is not recommended, as the controller can not efficiently manage disk space allocation unless the free space of each partition was given across the protocol (this is not done due to the open nature of the number of partitions).

If a disk system has multiple partitions and video files may not span partitions, the Largest Contiguous Block must be reported as the value of the partition that currently has the smallest size/Largest Contiguous Block. This is to guarantee that the controller will delete enough material to make space for the recording to continue until that partition has space for recording to continue. Since the controller

does not know what partition a video file is on, many files may be deleted from other partitions to make space to allow recording to continue on a different partition. This makes partitions inefficient for disk space use, unless the controller knows which file is on which partition, and what the Largest Contiguous Block is for each partition. This protocol only supports a disk system with a single storage partition efficiently.

30/B0 11 — ID List

Syntax: **30 11** or
B0 11

This command returns a list of all IDs currently stored on the system to the controller. The format will return the number of IDs remaining to be transmitted in subsequent transmissions in RETURN DATA 1 and RETURN DATA 2 (RETURN DATA 1 MSB, RETURN DATA 2 LSB), followed by ten 8 byte IDs in RETURN DATA 3 through RETURN DATA 82.

The **NEXT** command is used to transfer any remaining IDs in groups of up to ten. **NEXT** is called repeatedly until all IDs have been transferred.

Note: *Use of the **B0 11** command bears the format described in the section on Variable Length IDs.*

30/B0 14 — ID Size Request

Syntax: **30 14 ID** or
B0 14 BC XID
ID: 8 bytes, indicating clip ID
BC: 1-byte unsigned integer, indicating total amount of data to follow
XID: Extended ID

This command returns the duration of the specified **ID** to the controller. The format returns the frames in RETURN DATA 1, seconds in RETURN DATA 2, minutes in RETURN DATA 3, and hours in RETURN DATA 4, in BCD.

SEND DATA 1-8 contains the **ID** name.

Note: *Use of the **B0 14** command bears the format described in the section on Variable Length IDs.*

30/B0 16 — ID Request

Syntax: **30 16 ID** or
B0 16 BC XID

This command tells the controller whether the **ID** is currently in the “get from archive list” for the selected port and whether or not the **ID** is currently in the disk and the ARCHIVE. This command allows the automation controller to ask if an **ID** it needs in the future for play out is in the DISK or ARCHIVE.

RETURN DATA 1 returned contains a bit map of the ID status, a “1” indicates true and zero (0) indicates false. RETURN DATA 1 returned must contain at least one byte. The second byte of RETURN DATA 1 is optional, and only needs to be present on systems that support those statuses.

SEND DATA 1-8 contains the **ID** name.

Note: *Use of the **B0 16** command bears the format described in the section on Variable Length IDs.*

Nexio Addendum: Shaded boxes below denote unsupported bits.

RETURN DATA 1 Bit map for ID status

QUERY PENDING	TRANSFER IN PROGRESS	IN OFFLINE STORAGE TRANSFER LIST	IN OFFLINE STORAGE	DELETE PROTECT	IN ARCHIVE	IN ARCHIVE TRANSFER LIST	IN DISK
					NOT READY TO ARCHIVE	NOT READY TO TRANSFER	NOT READY TO PLAY
VIDEO TYPE (See VIDEO TYPE DEFINITION below)							
DISK HANDLE (optional byte)							

The In Disk Bit indicates the ID is on the video disk. If the In Disk Bit is only set after the ID is available to play or any other supported function, the Not Ready Bits are not required. If the In Disk bit is set and the ID is put in the IDs Added List when the ID is created on the video disk, but before it can be cued for play out, then the appropriate Not Ready bits must be set.

The Delete Protected Bit is set if the ID cannot be deleted. An Undelete Protect ID command clears this bit. A Delete Protect ID command sets this Bit.

The Transfer In Progress bit should be set only during the time the ID is being transferred (not a base band video recording) to or from an Archive, off line Storage device, or fibre channel copy to the disk. This bit should be reflected in all ports on the disk if they are receiving or sending the ID

NOT READY bits in Byte 2 of RETURN DATA 1 indicate that, although the ID file may reside on the disk, it is not complete enough for the applicable action. This would be true during recording, or during a copy of the video file from a tape archive, a fiber channel network, or any other source. If a file does not complete successful creation, but does exist, these bits may be left on. These bits indicate to the controller that it is not safe to perform these functions on this ID despite the fact that the ID does exist in the disk. If the IN DISK bit is not set, none of these bits need to be set as the controller will not try any of these functions on an ID that is not in the disk.

The Video Type only needs to be supported if a video disk can have more than one format of video files on it and cannot play any file type on any playout port. This function must be supported on both the Port Status Request and the ID Request so the controller can determine if a particular video port can play a particular ID.

Nexio Addendum: The Video Type functionality is optional and will report data of “0” unless the option is turned on and the Nexio server has been upgraded to at least the Nexio 5.7.0 Software Release. This option must be enabled in the LLM registry:

..LLM\Control\VideoStatusEnable. When set to a value of “1,” Byte 3 will indicate the video resolution and video format of the ID requested according to the following tables.

The LSB nibble is used to identify the video format of the ID.

Data	Video Format Nibble
0xn0	Default: Assume all files may be played on any port
0xn1	JPEG
0xn2	MPEG2 4:2:0
0xn3	MPEG3 4:2:2
0xn4	DV, all types
0xn5	JPEG2000
0xn6	DNxHD
0xn7	H.263, MPEG4 Part 2 (Long GOP, ASF)
0xn8	H.264, MPEG4 Part 10 (Long GOP, AVC)

0xn9	AVC-Intra, MPEG4 Part 10 (I-Frame, AVC)
0xnA	Apple Pro-Res
0xnB	XDCAM HD (1440x1080)
0xnC	REDCODE
0xnD	Reserved
0xnE	Reserved
0xnF	Reserved

The MSB nibble is used to identify the video resolution of the ID.

Data	Video Resolution Nibble
0x0n	Default: Assume all files may be played on any port
0x1n	SD
0x2n	720p
0x4n	1080i
0x8n	1080p

For example, an MPEG 4:2:2 clip recording in 1080i will have a value of "43."

30 17 Compression Settings Request

Syntax: **30 17 DATA**
DATA: 1 byte bitmap indicating settings requested

DATA 1 contains a bitmap specifying which information should be returned.

This command returns to the controller information about the selected port encoding parameters. The controller specifies in DATA 1 which parameter information should be returned. For example, Compression Setting 1 returns video Compression rates.

DATA 1 Bit map for SYSTEM STATUS

Compression Settings 8	Compression Settings 7	Compression Settings 6	Compression Settings 5	Compression Settings 4	Compression Settings 3	Compression Settings 2	Compression Settings 1

COMPRESSION SETTINGS 1: VIDEO COMPRESSION

Bytes 1 to 4 are the Video Compression Rate. It is the echo of the data sent in the **VIDEO COMPRESSION RATE** command (**20 31**).

Bytes 5 to 8 are the Video Compression Parameters. It is the echo of the data sent in the **VIDEO COMPRESSION PARAMETERS** command (**20 37**).

VIDEO COMPRESSION

VIDEO COMPRESSION RATE 1
VIDEO COMPRESSION RATE 2
VIDEO COMPRESSION RATE 3
VIDEO COMPRESSION RATE 4
VIDEO COMPRESSION PARAMETER 1
VIDEO COMPRESSION PARAMETER 2
VIDEO COMPRESSION PARAMETER 3
VIDEO COMPRESSION PARAMETER 4

COMPRESSION SETTINGS 2: AUDIO SETTINGS

Byte 1 is the Audio Sample Rate. It is the echo of the data sent in the **AUDIO SAMPLE RATE** command (**20 32**).

Byte 2 is the Audio Compression Rate. It is the echo of the data sent in the **AUDIO COMPRESSION RATE** command (**20 33**).

Byte 3 and 4 are the Audio Input Level. It is the echo of the data sent in the **AUDIO IN LEVEL** command (**20 34**).

Byte 5 and 6 are the Audio Output Level. It is the echo of the data sent in the **AUDIO OUT LEVEL** command (**20 35**).

AUDIO SETTINGS

AUDIO SAMPLE RATE 1
AUDIO COMPRESSION RATE
AUDIO INPUT LEVEL 1
AUDIO INPUT LEVEL 2
AUDIO OUTPUT LEVEL 1
AUDIO OUTPUT LEVEL 2

COMPRESSION SETTINGS 3: PORT TYPE SELECTS

Byte 1 is the Select Output Port Video Type. It is the echo of the data sent in the **SELECT OUTPUT** command (20 38). Byte 2 is the Select Input Port Video Type. It is the echo of the data sent in **SELECT INPUT** command (**20 39**).

PORT TYPE SELECTS

SELECT OUTPUT
SELECT INPUT

COMPRESSION SETTINGS 4: RECORD MODE

Byte 1 and 2 is the Record Mode data. It is the echo of the data sent in the **RECORD MODE** command (**20 3A**).

RECORD MODE

RECORD MODE 1
RECORD MODE 2

COMPRESSION SETTINGS 5: SUBCARRIER ADJUST

Byte 1 and 2 are the Subcarrier Adjust data. It is the echo of the data sent in the **SUBCARRIER ADJUST** command (20 41).

SUBCARRIER ADJUST

SUBCARRIER ADJUST 1
SUBCARRIER ADJUST 2

COMPRESSION SETTINGS 6: HORIZONTAL SYNC ADJUST

Byte 1 and 2 are the Horizontal Sync Timing data. It is the echo of the data sent in the **HORIZONTAL SYNC ADJUST** command (**20 42**).

HORIZONTAL SYNC ADJUST

HORIZONTAL SYNC ADJUST 1
HORIZONTAL SYNC ADJUST 2

30/B0 18 IDs Added List

Syntax: **30 18** or
B0 18

This request allows a controller to inquire about items that were added to the disk system by another signal port. The command returns to the controller a list of the IDs that have been added to the disk system since the last **IDs ADDED** request, or unreported IDs from before the last **IDs ADDED** request if not all were read. The list is kept for each active communications port.

Note: *Even the communications port that added the item should get the item in its list.*

The data format will return the number of IDs remaining to be transmitted in subsequent transmissions in RETURN DATA 1 and RETURN DATA 2 (DATA 1 MSB, DATA 2 LSB), followed by ten 8 byte IDs in RETURN DATA 3 through RETURN DATA 82.

A list of IDs ADDED must be kept for each port that has access to the disk. When an ID is completely recorded into the disk or transferred into the disk from an archive system and can be played at any time, that ID must be entered into the list for each communications port. When any IDs are in the added list for that communications port, the ID ADDED bit will be set in the port status sent to that communications port. If **an ID LIST** is performed, then all IDs may be removed from the added list for that communications port. As a communication port sends each ID (in-groups of ten) to its controller, it must remove that ID from the added list for that port.

On disks that may play an ID once recording or transfer from an archive has started but not completed, then this action should take place as soon as the ID is available for **PLAY CUE** or **CUE WITH DATA**.

The **NEXT** command is used to transfer any remaining IDs in groups of up to ten. **NEXT** is called repeatedly until all IDs have been transferred.

Note: *Use of the **B0 18** command bears the format described in the section on Variable Length IDs.*

30/B0 19 IDs Deleted List

Syntax: **30 19** or
B0 19

This command returns to the controller a list of the IDs that have been deleted from the disk system since the last **IDs DELETED** request, or unreported IDs from before the last **IDs DELETED** request if not all were read. This list is kept for each active communications port. This request allows a controller to find out about items added to the disk system that it may need that were deleted by another signal port.

Note**:** *Even the communications port that deleted the item should get the item in its list.*

The data format will return the number of IDs remaining to be transmitted in subsequent transmissions in RETURN DATA 1 and RETURN DATA 2 (RETURN DATA 1 MSB, RETURN DATA 2 LSB), followed by ten 8 byte IDs in RETURN DATA 3 through RETURN DATA 82. A list of IDs DELETED must be kept for each communications port that has access to the disk. When an ID is deleted from the disk and can not be played any more, that ID must be entered into the deleted list for each communications port. As a port sends each ID (in-groups of ten) to its controller, it must remove that ID from the deleted list for that port.

When an ID is in the deleted list, the ID DELETED bit will be set in the port status that is sent to that communications port. If an ID LIST is performed by a communications port, then all IDs may be removed from the deleted list for that communications port.

The NEXT command is used to transfer any remaining IDs in groups of ten. NEXT is called repeatedly until all IDs have been transferred.

Note: *Use of the **B0 19** command bears the format described in the section on Variable Length IDs.*

VDCP Examples

Cueing and Playing

There are two different commands to cue a clip in the server:

- Play Cue (20 24)
- Cue with Data (20 25)

The **Play Cue** command loads the specified ID to an aliased In Point and Out Point while the **Cue with Data** command allows the In and Out Points to be specified within the context of the command.

The server will roll the cued clip at sync speed 3 frames after a **Play** command (**10 01**) is received. Controllers should not attempt to compensate for capstan run-up ballistics when controlling a Nexio.

Sequenced, or stacked, playback in VDCP works exactly the same as Sequenced Playback in LEITCH Mode. The **Play Cue** and **Cue with Data** commands have the dual purpose to be able to cue an ID whether the machine is in PLAY or in IDLE mode. When in IDLE, they will search and display the first frame of the ID. A subsequent **Play** command will play the currently loaded ID.

If the **Play Cue** or **Cue with Data** commands are received while in PLAY, the machine will continue to play the current ID and cue the new ID specified to have it on standby until another **Play** command is received.

Example 1 - Recording Media

The following is an example of recording a 30 second segment called SEGMENT1 on channel 1.

30 01 RECORDPORT
OPEN PORT
If not already done, this opens a signal port upon which to record. RECORDPORT = 81.

20 22 RECORDPORT
SELECT PORT
Selects the opened signal port before recording.
RECORDPORT = 81.

A0 23 BC SEGMENT1 00 30 00 00
RECORD INIT
Creates a new ID and assigns 00:00:00:00 as the first aliased timecode frame of that ID. Record time is set for thirty seconds. The CUED status bit will be set high.

10 02
RECORD
The server selects a contiguous free area of the disk storage large enough to hold the 30 seconds of material. The recording will begin and the PLAY/REC status bit will be set

high. The server will stop recording when the time assigned in the Record Init command has elapsed, when it receives a Stop command, or when the disk space assigned to the selected ID is exhausted.

Note: Actual amount of disk space used will be assigned to that ID when the recording ends.

30 05 0F
PORT STATUS REQUEST
The controller should monitor the progress of the recording on a regular basis.

30 06 01
POSITION REQUEST
The controller should monitor the timecode progression of the recording on a regular basis.

Example 2 - Open-ended Recording

The following is an example of creating and recording an open-ended ID called SEGMENT1. These commands do not perform an open-ended recording in the strictest sense, but do a very close imitation of such a task. The trick here is to fool the server into allotting a large space for recording, and then terminating the recording when the desired length is reached.

30 01 RECORDPORT
OPEN PORT
If not already done, this opens a signal port upon which to record. RECORDPORT = 81.

20 22 RECORDPORT
SELECT PORT
Selects the opened signal port before recording.
RECORDPORT = 81.

A0 23 BC SEGMENT1 00 00 01 00
RECORD INIT
Creates the new ID and assigns 00:00:00:00 as the first aliased timecode frame of that ID. Its record time is set for one hour and the CUED status bit will be set high. The duration is purposely set for a very long time so long as it is within the limits of the disk storage. The server will stop recording when such time has elapsed, when it receives a Stop command, or when the disk space assigned to the selected ID is exhausted.

10 02
RECORD
The server selects a contiguous free area of the disk storage large enough to hold the material. The PLAY/REC status bit will be set high.

30 05 0F
PORT STATUS REQUEST
The controller should monitor the progress of the recording on a regular basis.

30 06 01
POSITION REQUEST
The controller should monitor the timecode progression of the recording on a regular basis.

10 00
STOP
When the desired length has been reached, this command will stop the server from recording. The ID's new duration will be updated in the internal database. The IDLE status bit will be set high.

Example 3 - Media Playback

The following is an example of playing to air a single ID named SEGMENT1.

30 01 PLAYPORT
OPEN PORT
If not already done, this opens a signal port upon which to play.
PLAYPORT = 01.

20 22 PLAYPORT
SELECT PORT
Selects the opened signal port before playing.
PLAYPORT = 01.

A0 24 BC SEGMENT1
PLAY CUE
If SEGMENT1 exists, the server displays field 1 of the first frame assigned to that ID. The CUED status bit will be high upon successful completion.

10 01
PLAY
The server enters into play and will play from the first frame to the last frame of the loaded ID. The server will remain on the last frame of the ID until stopped. The PLAY/REC status bit will be set high until the server has been manually stopped.

30 05 0F
PORT STATUS REQUEST
The controller should monitor the progress of ID playback on a regular basis.

B0 07
ACTIVE ID REQUEST
The controller should monitor the current ID on a regular basis in order to be aware of which ID is playing.

30 06 01
POSITION REQUEST
The controller should monitor the timecode progression of ID playback on a regular basis.

10 00
STOP
This will terminate playback. The IDLE status bit will be set high.

Example 4: Playback of Multiple IDs

The following is an example of playing to air a series of IDs.

30 01 PLAYPORT
OPEN PORT

If not already done, this opens a signal port upon which to play.
PLAYPORT = 01.

20 22 PLAYPORT
SELECT PORT
Selects the opened signal port before playing.
PLAYPORT = 01.

A0 24 BC SEGMENT1
PLAY CUE
If SEGMENT1 exists, the server displays field 1 of the first frame assigned to that ID. The CUED status bit will be high upon successful completion.

10 01
PLAY
The server enters into PLAY and will play from the first frame to the last frame of the loaded ID. The server will remain on the last frame of the ID until stopped. The PLAY/REC status bit will be set high until the server has been manually stopped.

A0 24 SEGMENT2
PLAY CUE
Issued 4 frames or more after the play command, this preloads SEGMENT2 in the channel, if SEGMENT2 exists. The server continues to play SEGMENT1 and cues SEGMENT2, placing it in standby. The CUED and PLAY/REC status bits will be high upon successful completion.

10 01
PLAY
This is issued a pre-determined latency before the end of SEGMENT1. The server will play to the end of SEGMENT1 and then play SEGMENT2. When SEGMENT2 begins playing, the CUED status bit drops and the PLAY/REC status bit remains high. If the Play command arrives late or is never issued, the video will still on the first frame of the next ID (SEGMENT2).

A0 24 BC SEGMENT3
PLAY CUE
Issued 4 frames or more after the play command, this preloads SEGMENT3 in the channel, if it exists. The server continues to play SEGMENT2 and cues SEGMENT3, placing it in standby. The CUED and PLAY/REC status bits will be high upon successful completion.

10 01
PLAY
This is issued a pre-determined latency before the end of SEGMENT2. The server will play to the end of SEGMENT2 and then play SEGMENT3. When SEGMENT3 begins playing, the CUED status bit drops and the PLAY/REC status bit remains high. If the Play command arrives late or is never issued, the video will still on the first frame of the next ID (SEGMENT3).

30 05 0F
PORT STATUS REQUEST
The controller should monitor the progress of ID playback on a regular basis.

B0 07
ACTIVE ID REQUEST
The controller should monitor the current ID on a regular basis in order to be aware of which ID is playing.

30 06 01
POSITION REQUEST
The controller should monitor the timecode progression of ID playback on a regular basis.

10 00
STOP
This will terminate playback. The IDLE status bit will be set high.

Unsupported VDCP Command List

The following VDCP commands are not supported by Nexio servers. Please review the VDCP manual for further reference to and definition of these commands.

Table 3-89: Unsupported VDCP Commands

B.C.	CMD-1	CMD-2	CODE	NAME	B.C.	CMD-1	CMD-2	NAME
02	0X / 8X	0C	-Opt	Local Disable			04	ACK
02	0X / 8X	0D	-Opt	Local Enable			04	ACK
0A	0X / 8X	14	+Opt	Delete From Archive			04	ACK

B.C.	CMD-1	CMD-2	CODE	NAME	B.C.	CMD-1	CMD-2	NAME
02	1X / 9X	03	-Opt	Freeze			04	ACK
02	1X / 9X	09	-Opt	Unfreeze			04	ACK

B.C.	CMD-1	CMD-2	CODE	NAME	B.C.	CMD-1	CMD-2	NAME
07	2X /AX	1E	+Opt	Preset Standard Time			04	ACK
0A	2X /AX	27	+Opt	Get From Archive			04	ACK
02	2X /AX	29	+Opt	Clear			04	ACK
0A	2X /AX	2A	+Opt	Send To Archive			04	ACK
03	2X /AX	2D	-Opt	Select Logical Drive			04	ACK
02	2X /AX	30	-Opt	Preset			04	ACK
06	2X /AX	31	-Opt	Video Compression Rate			04	ACK
03	2X /AX	32	-Opt	Audio Sample Rate			04	ACK
03	2X /AX	33	-Opt	Audio Compression Rate			04	ACK
06	2X /AX	37	-Opt	Video Compression Parameters			04	ACK
03	2X /AX	38	-Opt	Select Ouput			04	ACK
04	2X /AX	3A	-Opt	Record Mode			04	ACK

B.C.	CMD-1	CMD-2	CODE	NAME	B.C.	CMD-1	CMD-2	NAME
02	3X / BX	03	+Opt	Last	XX	3X / BX	83	Last Response
02	3X / BX	15	+Opt	IDs Added to Archive	XX	3X / BX	95	List IDs Trnsfrd.
04	3X / BX	25	-Opt	Multi Port Status Request	XX	3X / BX	A5	State Status

B.C.	CMD-1	CMD-2	CODE	NAME	B.C.	CMD-1	CMD-2	NAME
03	5X / DX	60	+Opt	Abort Macro #			04	ACK
02	5X / DX	61	+Opt	Active Macro List	XX	5X / DX	E1	Active Macros
03	5X / DX	62	+Opt	Macro Status	XX	5X / DX	E2	Macro Return
0C+	5X / DX	63	+Opt	Copy File To			04	ACK
0A	5X / DX	64	+Opt	Get From Archive			04	ACK
0A	5X / DX	65	+Opt	Send To Archive			04	ACK
0A+	5X / DX	66	-Opt	Prepare ID To Play			04	ACK
0A	5X / DX	67	-Opt	Close ID From Play			04	ACK

