MCI Command Strings Contents

About MCI

The Media Control Interface (MCI) is a high-level command interface to multimedia devices and resource files. MCI provides applications with device-independent capabilities for controlling audio and visual peripherals. Your application can use MCI to control any supported multimedia device, including audio playback and recording. For a full overview of MCI, see the *Multimedia Programmer's Guide*.

MCI provides standard commands for playing multimedia devices and recording multimedia resource files. Using MCI, an application can control multimedia devices using simple commands like open, play, and close. MCI commands are a generic interface to multimedia devices.

MCI includes the following interfaces:

* The *command-message interface* consists of C constants and structures. The *Multimedia Programmer's Guide* describes the command-message interface and presents numerous examples on using the command messages to control audio devices. In the *Multimedia Programmer's Reference*, the individual command messages and structures are described in Chapter 4, Message Overview, Chapter 5, Message Directory, and Chapter 6, Data Types and Structures.
* The *command-string interface* provides a textual version of the command messages. The strings use an easy-to-read format that makes it easy to control MCI devices from C programs and multimedia authoring tools. Command strings duplicate the functionality of the command messages; Windows converts the command strings to command messages before sending them to the MCI driver for processing.

Controlling MCI Devices

To control an MCI device, you open the device, send the necessary commands to it, and then close the device. For example, the following series of MCI commands play track 6 of an audio CD:

```
open cdaudio
set cdaudio time format tmsf
play cdaudio from 6 to 7
close cdaudio
```

The next example shows a similar series of MCI commands that play the first 10,000 samples of a waveform audio file:

```
open c:\mmdata\purplefi.wav type waveaudio alias finch
set finch time format samples
play finch from 1 to 10000
close finch
```

You will note the following from the previous examples:

* The same basic commands (open, play, and close) are used with both devices.
* The open command for the "waveaudio" device includes a filename specification. The "waveaudio" device is a compound device (one associated with a media element), while the "cdaudio" device is a simple device (one without an associated media element).
* The set commands both specify time formats, but the time-format options for the "cdaudio" device differ from those used with the "waveaudio" device.
* The parameters used with the **from** and **to** flags are appropriate to the respective device. For the "cdaudio" device, the parameters specify a range of tracks; for the "waveaudio" device, the parameters specify a range of samples.

MCI Device Types and Drivers

MCI recognizes a basic set of *device types*. A device type is a set of MCI drivers that share a common command set and are used to control similar multimedia devices or data files.

The following table lists the currently defined device types. The current implementation of MCI includes command sets for a subset of these devices.

Device Type	Description
animation	Animation device
cdaudio	Audio CD player
dat	Digital audio tape player
digitalvideo	Digital video in a window (not GDI based)
other	Undefined MCI device
overlay	Overlay device (analog video in a window)
scanner	Image scanner
sequencer	MIDI sequencer
vcr	Videotape recorder or player
videodisc	Videodisc player
waveaudio	Audio device that plays digitized waveform files

MCI Device Names

For any given device type, there might be several MCI drivers that share the command set but operate on different data formats. For example, the *animation* device type might include several MCI drivers that use the same command set but play different types of animation files. To uniquely identify a MCI driver, MCI uses *device names*.
Device names are identified in the [mci] section of the SYSTEM.INI file. The [mci] section of SYSTEM.INI identifies all MCI drivers to Windows. The following example shows a typical [mci] section:

```
[mci]
waveaudio=mciwave.drv
sequencer=mciseq.drv
MMMovie=mcimmp.drv
cdaudio=mcicda.drv
```

The name on the left side of the equal sign is the driver device name. The value on the right side of the equal sign identifies the filename of the MCI driver. Frequently, the device name is the same as the device type for the driver, as is the case for the *waveaudio*, *sequencer*, and *cdaudio* devices in the preceding example. The *MMMovie* device is an *animation* device, but it uses a unique device name.

If an MCI driver is installed using a device name that already exists in the [mci] section, Windows appends an integer to the device name of the new driver, creating a unique device name. In the preceding example, a driver installed using the *cdaudio* device name would be assigned device name "cdaudio1." A subsequent *cdaudio* device would be assigned device name "cdaudio2."

MCI Drivers Included with Windows

The retail Windows package includes the following MCI drivers:

Device Name	Driver Filename	Description
sequencer	MCISEQ.DRV	An MCI device driver for playing MIDI audio files.
waveaudio	MCIWAVE.DRV	An MCI device driver for playing and recording waveform audio files.

The Windows setup program creates an [mci] section in SYSTEM.INI that lists the *sequencer* and *waveaudio* devices. The device names are the same as the device type names.

The Windows SDK includes the following additional MCI drivers:

Device Name	Driver Filename	Description
cdaudio	MCICDA.DRV	An MCI device driver for controlling compact disc audio.
videodisc	MCIPIONR.DRV	An MCI device driver for controlling the Pioneer LD-V4200 videodisc player.

To install these devices, use the Drivers setting in Control Panel. You can distribute the MCICDA.DRV and MCIPIONR.DRV drivers with any applications that require them. Other MCI drivers can be distributed with the applications or hardware devices that use them.

Driver Support for MCI Commands

MCI drivers provide the functionality for MCI commands. The Windows system software performs some housekeeping tasks, but all the multimedia playback, presentation, and recording is handled by the individual MCI drivers.

Drivers vary in their support for MCI commands and command options. Because multimedia devices can have widely different capabilities, MCI is designed to let individual drivers extend or reduce the command sets to match the capabilities of the device. For example, the record command is part of the command set for MIDI sequencers, but the MCISEQ driver included with Windows does not support the record command. Also, the MCI Movie Player driver (MCIMMP.DRV) included with the Microsoft Multimedia Development Kit extends the open command to include an **expanddibs** option that is not part of the basic command set for the *animation* device type.

Commands Available on All Drivers

MCI defines four classifications of commands. The commands and options comprising the following two classifications are defined as the minimum command set for any MCI driver:

* *System commands*. These commands are handled directly by MCI rather than by the driver.
* *Required commands*. These commands are handled by the driver. All drivers should support the required commands and options.

Regardless of the specific driver you're using, your application should be able to assume that the commands and options in the two preceding groups are available.

Driver-Specific Commands

The commands comprising the following two classifications are not supported by all drivers:

* *Basic commands*, or optional commands, are used by some devices. If a device supports a basic command, it must support a defined set of options for the command.
* *Extended commands* are specific to a certain device types or drivers. Extended commands include new commands (like the put and where commands for the animation device type) and extensions to existing commands (like the **can stretch** option added to the *animation* status command).

If your application needs to use a basic or extended command or option, it should query the driver before trying to use the command or option (that is, unless you are certain that the MCI driver you've used during development is the same one that will be available on the delivery system).

Default Behavior of MCI Drivers

In many situations, the MCI command specifications define the default values and behaviors for drivers of a particular device type. Drivers should use the documented default values and behaviors whenever possible.

Since multimedia devices can have a wide range of features (and limitations), there can be undefined areas of behavior. Also, drivers may handle exceptions differently, based on the device capabilities and the design goals of the programmer who developed the driver.

For example, consider the following commands sent to the *waveaudio* driver (MCIWAVE.DRV):

```
open sound.wav alias sound
play sound notify
record sound from 0 notify
```

The **record** command returns a "Parameter out of range" value and stops the playback started by the previous **play** command. One might expect the driver to validate the **record** command before stopping playback, but the driver stops the playback first.

Using MCI Command Strings

Command Basics

Controlling MCI Devices

Syntax of Command Strings

MCI command strings use a consistent verb-object-modifier syntax. Each command string includes a command, a device identifier, and command arguments. Arguments are optional on some commands and required on other commands.

A command string has the following form:

command device_id arguments

These components contain the following information:

* The *command* specifes an MCI command. Examples of commands include open, close, and play.
* The *device_id* specifies an instance of an MCI driver. The *device_id* is created when the device is opened. See Using the Open Command for information on how the device ID is assigned.
* The *arguments* specify the flags and parameters used by the *command*. Flags are key words recognized with the MCI command. Parameters are numbers or strings that apply to the MCI command or flag.

  For example, the play command uses the arguments **from** *position* and **to** *position* to indicate the positions to start and end playing. In this chapter, flags are bold, and variable parameters are italic. You can list the flags used with a command in any order. When you use a flag that has a parameter associated with it, you must supply a value for the parameter.

Unspecified (and optional) command arguments assume a default value.

Data Types for Command Parameters

You can use the following data types for the parameters in a command string:

Data Type	Description
Strings	String data types are delimited by leading and trailing white space and quotation marks ("). MCI removes single quotation marks from a string. To put a quotation mark in a string, use a set of two quotation marks where you want to embed your quotation mark. To use an empty string, use two quotation marks to delimit the empty string.
Signed long integers	Signed long integer data types are delimited by leading and trailing white space. Unless otherwise specified, integers can be positive or negative. If you use negative integers, don't put any space between the minus sign and the first digit.
Rectangles	Rectangle data types are an ordered list of four signed short values. White space delimits this data type as well as separates each integer in the list.

Using the Wait Flag

Normally, MCI commands return to the user immediately, even if it takes several minutes to complete the action initiated by the command. For example, after a VCR device receives a rewind command, the command returns when the tape starts to rewind. It does not wait for the tape to finish rewinding. You can use the **wait** flag to direct the device to wait until the requested action is completed before returning control to the application.

For example, the following **play** command won't return control to the application until the playback completes:

```
play mydevice from 0 to 100 wait
```

The user can cancel a wait operation by pressing a break key. By default, this key is CTRL+BREAK. When a wait operation is cancelled, MCI attempts to return control to the application without interrupting the command associated with the **wait** flag.

For example, in the preceding example, breaking the **play** command cancels the wait operation without interrupting the play operation.

To redefine the break key, use the break command.

Using the Notify Flag

The **notify** flag directs the device to post an MM_MCINOTIFY message when the device completes an action. Your application must have a window procedure to process the MM_MCINOTIFY message for notification to have any effect. The *Multimedia Programmer's Guide* includes examples of window procedures that process the MM_MCINOTIFY message.

A notification message can indicate one of the following results:

* The notification is successful
* The notification is superseded
* The notification is aborted
* The notification fails

A successful notification occurs when the conditions required for initiating the callback are satisfied and the command completed without interruption.

A notification is superseded when the device has a notification pending and you send it another notify request. When a notification is superseded, MCI resets the callback conditions to correspond to the notify request of the new command.

A notification is aborted when you send a new command that prevents the callback conditions set by a previous command from being satisfied. For example, sending the stop command cancels a notification pending for the "play to 500 notify" command. If your command interrupts a command that has a notification pending, and your command also requests notification, MCI will abort the first notification immediately and respond to the second notification.

A notification fails if a device error occurs while a device is executing the MCI command. For example, a notification fails when a hardware error occurs during a play command.

MCI Tips and Shortcuts

Opening a Device

Playing a Device

The play command starts playing a device. Without any flags, the **play** command starts playing from the current position and plays until the command is halted or until the end of the media or file is reached. For example, the following command starts playing an audio disc from the position where it was stopped:

```
play cdaudio
```

Most devices that support the **play** command also support the **from** and **to** flags. These flags indicate the position at which the device should start and stop playing. For example, the following command plays an audio disc from the beginning of the first track:

```
play cdaudio from 0
```

Some device types extend the **play** command to use the capabilities of a particular device. For example, the play command for videodisc devices adds the **fast**, **slow**, and **scan** flags.

Specifying Time Formats

The units assigned to the position value depend on the time format used by the device. Each device has a default time format, but it's always good practice to specify the time format (using the set command) before issuing any commands that use position values.

Stopping and Pausing a Device

The stop command suspends the playing or recording of a device. Many devices also include the basic command pause. The difference between **stop** and **pause** depends on the device. Usually **pause** suspends operation but leaves the device ready to resume playing or recording immediately.

Using play or record to restart a device will reset the **to** and **from** positions specified before the device was paused or stopped. Without the **from** flag, these commands reset the start position to the current position. Without the **to** flag, they reset the end position to the end of the media.

To continue playing or recording while stopping at a previously specified position, use the **to** flag with the **play** or **record** commands to specify an ending position.

Some devices include the resume command to restart a paused device. This command does not change the **to** and **from** positions specified with the **play** or **record** command which preceded the **pause** command.

Getting Information from a Device

Every device responds to the capability, status, and info commands. These commands obtain information about the device. For example, the following command returns **true** if the *cdaudio* device can eject the disc:

```
capability cdaudio can eject
```

With the MCICDA.DRV driver included with the SDK, this example returns **true**.

The flags listed for the required and basic commands provide a minimum amount of information about a device. Many devices supplement the required and basic flags with extended flags to provide additional information about the device.

When you request information by using the **capability**, **status**, or **info** command, the argument list can contain only one flag requesting information. The command-string interface can only return one string or value in response to a command requesting information.

Closing a Device

The close command releases access to a device or device element. To help MCI manage the devices, your application must explicitly close each device or device element when it is finished with it.

Related Topics
Using All as the Device ID

Using All as the Device ID

You can specify **all** as a device ID for any command that does not return information. When you specify **all**, MCI sequentially sends the command to all devices opened by the current application.

For example, "close all" closes all open devices and "play all" starts playing all devices opened by the application. Because MCI sequentially sends the commands to the MCI devices, there is a delay between when the first device receives the command and when the last device receives the command.

Combining the Device Name and Element Name

You can eliminate the **type** flag in the open command if you combine the device name with the device element name. MCI recognizes this combination when you use the following syntax:

```
device_name!element_name
```

The exclamation point separates the device name from the element name. There should not be any spaces between the exclamation point and the *device_name* or *element_name*.

The following example opens the RIGHT.WAV element using the "waveaudio" device:

```
open waveaudio!right.wav
```

Opening Devices Automatically

If MCI cannot identify the *device_name* of a command as an open device, MCI tries to automatically open the specified device. The following items apply to devices automatically opened:

* Automatic open works only with the command-string interface.
* Automatic open does not let your application specify the **type** flag. Without the device name, MCI determines the device name from the [mci extensions] section of the WIN.INI file. To use a specific device, you can combine the device name name with the device element name using the exclamation point. (See Combining the Device Name and Element Name for information).
* Automatic open will fail for commands that are specific to custom device drivers. For example, the command used to unlock the front panel of the Pioneer videodisc player will fail an automatic open. The command used for unlocking the front panel is specific only to this videodisc player.
* Automatically opened devices don't respond to commands that use **all** as a device name.

MCI automatically closes any device automatically opened using the command-string interface. MCI closes a device in the following situations:

* When the command is completed
* When you abort the command
* When you request notification in a subsequent command
* When MCI detects a failure

Using the Open Command

Before using a device, you must initialize it by using the **open** command. The **open** command loads the driver into memory (if it isn't already loaded) and establishes the device ID that you will use to identify the device in subsequent MCI commands.

The number of devices you can have open is limited only by the amount of available memory.

The **open** command has the following form:

open *device* **shareable type** *device_name* **alias** *alias*

The parameters for the **open** command are as follows:

Parameters	Description
device	Identifies the MCI driver or media element to open.
shareable	Allows applications to share a common device or device element.
type *device_name*	Identifies the MCI device name when *device* refers to a media element instead of a specific MCI device name.
alias *alias*	Specifies replacement name *alias* for the device.

When specifying a device name in the *device* or *device_name* parameters, you can specify a device name from the SYSTEM.INI file or the filename of the device driver. If you specify the filename of the device driver, you can optionally include the .DRV extension; don't include the path to the file.

NOTE

Opening a device using the device driver filename makes the application device dependent and can prevent the application from running if the system configuration changes.

For example, the following command opens the *cdaudio* device and assigns alias "mycd":

```
open cdaudio alias mycd
```

The next command opens waveform file C:\WINDOWS\CHIMES.WAV and assigns alias "chimes":

```
open c:\windows\chimes.wav type waveaudio alias chimes
```

How the Device ID Is Assigned

The device ID established by the open command identifies the open MCI device in all subsequent commands. If you specify a device *alias* using the **alias** key word, the device ID will be the *alias*. Otherwise, the device ID will be the *device* name.

Related Topics

Automatically Opening a Device

Opening Simple Devices

MCI classifies device drivers as *compound* and *simple*. Simple device drivers don't require a device element for playback. Simple devices include *cdaudio* and *videodisc* devices.

For example, you can open a videodisc device using the following command:

```
open videodisc
```

Simple devices require only the *device_name* for operation. You don't need to provide any additional information (such as a name of a data file) to open these devices. For these devices, substitute the name of a device name from the [mci] section of SYSTEM.INI for the *device_name*.

Opening Compound Devices

Compound device drivers use a *device element*-a media element associated with a device-during operation. For most compound device drivers, the device element is the source or destination data file. For these file elements, the element name references a file and its path. Compound devices include waveform audio devices and and MIDI sequencers.

Depending on your needs, there are three ways you can open a compound device:

* By specifying just the device name (this lets you open a compound device without associating an element filename). When opened this way, most compound devices will only process the **capability** and **close** commands.
* By specifying just the element name (the device name is determined from the [mci extensions] section of the WIN.INI file).
* By specifying both the element name and the device name (MCI ignores the entries in the [mci extensions] section of the WIN.INI file and opens the specified device name).

To associate a device element with a particular device, you can specify the element name and device name. For example, the following command opens the *waveaudio* device with element filename "myvoice.snd":

```
open myvoice.snd type waveaudio
```

You can also abbreviate the device name specification using the alternative exclamation-point format. For more information, see Combining the Device Name and Element Name.

Using the Filename Extension to Select the Device

If the **open** command only specifies the element name, MCI uses the filename extension of the element name to select the appropriate device from the list in the [mci extensions] section of the SYSTEM.INI file. The entries in the [mci extensions] section use the following form:

```
filename_extension=device_name
```

MCI implicitly uses *device_name* if the extension is found and if the device name has not been specified in the **open** command.

The following example shows a typical [mci extensions] section:

```
[mci extensions]
wav=waveaudio
mid=sequencer
rmi=sequencer
```

With these definitions, MCI opens the *waveaudio* device if the following command is issued:

```
open train.wav
```

Opening Shareable Devices

The **shareable** flag lets multiple applications access the same device (or element) and device instance concurrently. If your application opens a device or device element as shareable, other applications can also access it by opening it as shareable. The shared device or device element gives each application the ability to change the parameters governing the operating state of the device or device element. Each time a device or device element is opened as shareable, MCI returns a unique device ID even though the IDs refer to the same instance.

If your application opens a device or device element without the **shareable** flag, no other application can access it simultaneously. Also, if a device can service only one open instance, the **open** command will fail if you specify the **shareable** flag.

If you make a device or device element shareable, your application should not make any assumptions about the state of a device. When working with shared devices, your application might need to compensate for changes made by other applications using the same services.

While most compound device elements are not shareable, you can open multiple elements (where each element is unique), or you can open a single element multiple times. If you open a single file element multiple times, MCI creates an independent instance for each, with each instance having a unique operating status.

If you open multiple instances of a file element, you must assign a unique device ID to each. The **alias** flag described in the following section lets you assign a unique name for each element.

Assigning a Device ID Using the Alias Flag

The **alias** flag specifies a device ID for the device. The alias flag lets you assign a short device ID for compound devices with lengthy filenames, and it lets you open multiple instances of the same file or device.

For example, the following command assigns device ID "birdcall" to a lengthy waveform filename:

```
open c:\nabirds\sounds\mockmtng.wav type waveaudio alias birdcall
```

If the alias flag were omitted, the device ID would be "c:\nabirds\sounds\mockmtng.wav."

Opening New Device Elements

To create a new device element without specifying an element name, specify **new** as a *device_name*. MCI does not save a new file element until you save it using the **save** command. When creating a new file, you must include a device alias with the **open** command.

The following example opens a new "waveaudio" device element, starts and stops recording, saves the file element, and closes the device element:

```
open new type waveaudio alias capture
record capture
stop capture
save capture orca.wav
close capture
```

System Commands

MCI directly processes the following system command rather than passing them to MCI devices:

Command	Description
break	Sets a break key for an MCI device.
sysinfo	Returns information about MCI devices.

Required Commands

All devices support the following commands:

Command	Description
capability	Obtains the capabilities of a device.
close	Closes the device.
info	Obtains textual information from a device.
open	Initializes the device.
status	Obtains status information from the device.

Devices must also support a standard set of command options for the required commands. For a complete description of the required commands and options, select the command names in the preceding table.

Basic Commands

The following list summarizes the basic commands. The use of these messages by a device is optional.

Command	Description
load	Recalls data from a disk file.
pause	Pauses playing or recording.
play	Starts transmitting output data.
record	Starts recording input data.
resume	Resumes playing or recording on a paused device.
save	Saves data to a disk file.
seek	Seeks forward or backward.
set	Sets the operating state of the device.
status	Obtains status information about the device. The **status** command is also listed in the group of required commands. In the basic group, options are added for devices that use linear media with identifiable positions.
stop	Stops playing or recording.

If a driver supports a basic command, it must also support a standard set of options for the command. For a complete description of the basic commands and options, select the command names in the preceding table.

Extended Commands

Some MCI devices have additional commands or add options to existing commands. While some extended commands only apply to a specific device driver, most of them apply to all drivers of a particular device type. For example, the *sequencer* command set extends the set command to add time formats that are needed by MIDI sequencers.

Unless you are certain that the specific MCI driver you use during development will be available on the delivery system, you should not assume that the device supports the extended commands or options. You can use the capability command to determine whether a specific feature is supported, and your application should be ready to deal with "unsupported command" or "unsupported function" return values.

The following extended commands are available with certain device types:

Command	Device Types	Description
cue	waveaudio	Prepares for playing or recording.
delete	waveaudio	Deletes a data segment from the MCI element.
escape	videodisc	Sends custom information to a device.
freeze	overlay	Disables video acquisition to the frame buffer.
put	animation	Defines the source, destination, and frame windows.
	overlay	
realize	animation	Tells the device to select and realize its palette into a display context of the displayed window.
spin	videodisc	Starts the disc spinning or stops the disc from spinning.
step	animation	Step the play one or more frames forward or reverse.
	videodisc	
unfreeze	overlay	Enables the frame buffer to acquire video data.
update	animation	Repaints the current frame into the display context.
where	animation	Obtains the rectangle specifying the source, destination, or frame area.
	overlay	
window	animation	Controls the display window.
	overlay	

For a complete description of each command, select the device type names in the preceding table.

Animation Commands

The following list summarizes the MCI commands used with the "animation" device type:

Command	Description
capability	Obtains the capabilities of a device.
close	Closes the device.
info	Obtains textual information from a device.
open	Initializes the device.
pause	Pauses the device.
play	Starts playing an animation sequence.
put	Defines the source, destination, and frame windows.
realize	Tells the device to select and realize its palette into a display context of the displayed window.
resume	Resumes playing or recording on a paused device.
seek	Moves to the specified position on the animation and stops.
set	Establishes control settings for the driver.
status	Obtains status information from the device.
step	Step the play one or more frames forward or reverse.
stop	Stops playing.
update	Repaints the current frame into the display context.
where	Obtains the rectangle specifying the source or destination area.
window	Tells the device to use a given window for display instead of the default window.

Audio CD Commands

The following list summarizes the commands used with the "cdaudio" device type:

Command	Description
capability	Obtains the capabilities of a device.
close	Closes the device.
info	Obtains textual information from a device.
open	Initializes the device.
pause	Pauses playing.
play	Starts playing the device.
resume	Resumes playing or recording on a paused device.
seek	Seeks forward or backward.
set	Sets the operating state of the device.
status	Obtains status information from the device.
stop	Stops playing.

MIDI Sequencer Commands

The following list summarizes the commands used with the "sequencer" device type:

Command	Description
capability	Obtains the capabilities of a device.
close	Closes the device.
info	Obtains textual information from a device.
open	Initializes the device.
pause	Pauses playing or recording.
play	Starts transmitting output data.
record	Starts recording input data.
resume	Resumes playing or recording on a paused device.
save	Saves data to a disk file.
seek	Seeks forward or backward.
set	Sets the operating state of the device.
status	Obtains status information from the device.
stop	Stops playing or recording.

Videodisc Command Overview

The following list summarizes the commands used with the "videodisc" device type:

Command	Description
capability	Obtains the capabilities of a device.
close	Closes the device.
escape	Sends custom information to a device.
info	Obtains textual information from a device.
open	Initializes the device.
pause	Pauses playing.
play	Starts playing the device.
resume	Resumes playing or recording on a paused device.
seek	Seeks forward or backward.
set	Sets the operating state of the device.
spin	Starts the disc spinning or stops the disc from spinning.
status	Obtains status information from the device.
step	Step the play one or more frames forward or reverse.
stop	Stops playing.

Video Overlay Command Overview

The following list summarizes the commands used with the "overlay" device type:

Command	Description
capability	Obtains the capabilities of a device.
close	Closes the device.
freeze	Disables video acquisition to the frame buffer.
info	Obtains textual information from a device.
load	Recalls data from a disk file.
open	Initializes the device.
put	Defines the source, destination, and frame windows.
save	Saves data to a disk file.
set	Sets the operating state of the device.
status	Obtains status information from the device.
unfreeze	Enables the frame buffer to acquire video data.
where	Obtains the rectangle specifying the source, destination, or frame area.
window	Tells the device to use a given window for display instead of the default window.

Waveform Audio Command Overview

The following list summarizes the commands used with the "waveaudio" device type:

Command	Description
capability	Obtains the capabilities of a device.
close	Closes the device.
cue	Prepares for playing or recording.
delete	Deletes a data segment from the MCI element.
info	Obtains textual information from a device.
open	Initializes the device.
pause	Pauses playing or recording.
play	Starts transmitting output data.
record	Starts recording input data.
resume	Resumes playing or recording on a paused device.
save	Saves data to a disk file.
seek	Seeks forward or backward.
set	Sets the operating state of the device.
status	Obtains status information from the device.
stop	Stops playing or recording.

break (System)

Syntax

break *device_id parameter* [**notify**] [**wait**]

The **break** command specifies a key to abort a **wait** command.

Parameters

Specify one of the following items for *parameter*:

on *virtual_key*
: Specifies the *virtual_key* code that aborts the **wait**. When the key is pressed, the device returns control to the application. If possible, the command continues execution. Substitute a Windows virtual key code for *virtual_key*.

off
: Disables the current break key.

Example

The following command sets F2 as the break key for the "mysound" device:

```
break mysound on 113
```

break (System)

Command	Arguments
break device_name	**{off** | **on** virtual key code**}**
	[notify]
	[wait]

sysinfo (System)

Syntax

sysinfo *device_id parameter* [**notify**] [**wait**]

The **sysinfo** command obtains MCI system information.

Parameters

Specify one of the following items for *parameter*:

installname
Returns the name listed in the SYSTEM.INI file used to install the open device with device ID *device_id*.

quantity
Returns the number of MCI devices listed in the SYSTEM.INI file of the type specified in the *device_id* field. The *device_id* must be a standard MCI device type name (listed in MCI Device Types and Drivers). Any digits after the device type are ignored. The *device_id* **all** returns the total number of MCI devices in the system.

quantity open
Returns the number of open MCI devices of the type specified in the *device_id* field. The "device_id" must be a standard MCI device type name (listed in "MCI Device Types and Drivers," earlier in this chapter). The **device_id all** returns the total number of open MCI devices in the system.

name *index*
Returns the name of an MCI device. The "device_id" must be a standard MCI device type name (listed in "MCI Device Types and Drivers," earlier in this chapter). The *index* ranges from 1 to the number of devices of that type. If **all** is specified for *device_id*, *index* ranges from 1 to the total number of devices in the system.

name *index* **open**
Returns the name of an open MCI device. The "device_id" must be a standard MCI device type name (listed in "MCI Device Types and Drivers," earlier in this chapter). The *index* ranges from 1 to the number of open devices of that device type. If **all** is specified for the device type, *index* ranges from 1 to the total number of open devices in the system.

Example

The following command returns the number of open waveform audio devices:

```
sysinfo waveaudio quantity open
```

The following command returns the name (device alias) of the first open waveform audio device:

```
sysinfo waveaudio name 1 open
```

sysinfo (System)

Command	Arguments
sysinfo *device_id*	**{installname** **| name** *index* **| name** *index***open** **| quantity** **| quantity open}** **[notify]** **[wait]**

capability (Required)

Syntax

capability *device_id parameter* [**notify**] [**wait**]

The **capability** command requests information about a particular capability of a device.

Parameters

Specify one of the following items for *parameter* (specific device types and drivers may define other items):

can eject
Returns **true** if the device can eject the media.

can play
Returns **true** if the device can play.

can record
Returns **true** if the device supports recording.

can save
Returns **true** if the device can save data.

compound device
Returns **true** if the device supports an element name.

device type
Returns one of the following device type names:

animation
cdaudio
dat
digitalvideo
other
overlay
scanner
sequencer
vcr
videodisc
waveaudio

has audio
Returns **true** if the device supports audio playback.

has video
Returns **true** if the device supports video.

uses files
Returns **true** if the element of a compound device is a file pathname.

Example

The following command returns the device type of the "mysound" device:

```
capability mysound device type
```

capability (Required)

Command	Arguments
capability *device_id*	**{can eject** **| can play** **| can record** **| can save** **| compound device** **| device type** **| has audio** **| has video** **| uses files}** **[notify]** **[wait]**

close (Required)

Syntax

close *device_id* [**notify**] [**wait**]

The **close** command closes the device or device element. MCI unloads a device when all instances of the device or all device elements are closed.

Comments

To close all devices opened by your application, specify the **all** device ID with the **close** command.

Example

The following command closes the "mysound" device:

```
close mysound
```

close (Required)

Command	Arguments
close *device_id*	**[notify]** **[wait]**

info (Required)

Syntax

info *device_id parameter* [**notify**] [**wait**]

The **info** command gets textual information from the device.

Parameters

Specify one of the following items for *parameter* (specific device types and drivers may define other items):

product

Returns a description of the hardware associated with a device. This usually includes the manufacturer and model information.

Example

The following command gets a description of the hardware associated with the "mysound" device:

```
info mysound product
```

info (Required)

Command	Arguments
info *device_id*	**[product]** **[notify]** **[wait]**

open (Required)

Syntax

open *device* [*parameters*] [**notify**] [**wait**]

The **open** command initializes the device.

Parameters

You can specify one or more of the following optional items for *parameters* (specific device types and drivers may define other items):

alias *device_alias*
: Specifies an alternate name for the given device. If specified, it must be used as the *device_id* in subsequent commands.

shareable
: Initializes the device or element as shareable. Subsequent attempts to open it fail unless you specify **shareable** in both the original and later **open** commands.

  MCI returns an invalid device error if the device is already open and not shareable.

type *device_type*
: Specifies the device type of a device element. As an alternative to *type*, MCI can use the [mci extension] entries in the SYSTEM.INI file to select the device based on the extension used by the device element. You can also use the *device_name*!*element_name* abbreviation, described in Combining the Device Name and Element Name.

Example

The following command opens the **cdaudio** device and assigns an alias:

```
open cdaudio alias cd
```

open (Required)

Command	Arguments
open *device*	**[alias** *device_alias***]** **[shareable]** **[type** *device_type***]** **[notify]** **[wait]**

status (Required)

Syntax

status *device_id parameter* [**notify**] [**wait**]

The **status** command gets status information for the device.

Parameters

Specify one of the following items for *parameter* (specific device types and drivers may define other items):

mode

Returns the current mode of the device. All devices can return the following values:

not ready
paused
playing
stopped

Some devices can return the following additional mode values:

open
parked
recording
seeking

Example

The following command returns the current mode of the "mysound" device:

```
status mysound mode
```

status (Required)

Command	Arguments
status *device_id*	**[mode | ready]** **[notify]** **[wait]**

Load (Basic)

Syntax

load *device_id* [*filename*] [**notify**] [**wait**]

The **load** command loads a device element from disk.

Parameters

You can specify the following optional parameter:

filename
: Specifies the source path and file.

Example

The following command loads a file into the "vidboard" device:

```
load vidboard c:\vid\fish.vid notify
```

The **notify** flag tells MCI to send a notification message when the loading completes.

Load (Basic)

Command	Arguments
load *device_id*	**{***file_name***}** **[notify]** **[wait]**

pause (Basic)

Syntax

pause *device_id* [**notify**] [**wait**]

The **pause** command pauses playing or recording. Most drivers retain the current position, allowing playback or recording to continue at the current position.

Example

The following command pauses the "mysound" device:

```
pause mysound
```

pause (Basic)

Command	Arguments
pause *device_id*	**[notify]** **[wait]**

play (Basic)

Syntax

play *device_id* [*parameters*] [**notify**] [**wait**]

The **play** command starts playing the device.

Parameters

You can specify one or more of the following optional items for *parameters*:

from *position*
: Specifies a starting position for the playback. If the **from** paramer is not specified, playback begins at the current position.

to *position*
: Specifies an ending position for the playback. If the **to** parameter is not specified, playback ends at the end of the media.

Comments

Before issuing any commands that use position values, you should set the desired time format using the **set** command.

Example

The following command plays the mysound device from position 1000 through position 2000, sending a notification message when the playback completes:

```
play mysound from 1000 to 2000 notify
```

play (Basic)

Command	Arguments
play *device_id*	**[from** *position***]** **[to** *position***]** **[notify]** **[wait]**

record (Basic)

Syntax

record *device_id* [*parameters*] [**notify**] [**wait**]

The **record** command starts recording data. All data recorded after a file is opened is discarded if the file is closed without saving it.

Parameters

You can specify one or more of the following optional items for *parameters*:

insert
: Specifies that new data is added to the device element at the current position.

from *position*
: Specifies a starting position for the recording. If the **from** parameter is not specified, the device starts recording at the current position.

to *position*
: Specifies an ending position for the recording. If the **to** parameter is not specified, the device records until it receives a **stop** or **pause** command.

overwrite
: Specifies that new data will replace data in the device element.

Comments

Before issuing any commands that use position values, you should set the desired time format using the **set** command.

Example

The following command starts recording data into the "newsound" device at the current position:

```
record newsound
```

The recording stops when a **stop** or **pause** command is issued.

record (Basic)

Command	Arguments
record *device_id*	**[from** *position***]** **[to** *position***]** **[insert** | **overwrite]** **[notify]** **[wait]**

resume (Basic)

Syntax

resume *device_id* [**notify**] [**wait**]

The **resume** command continues playing or recording on a paused device.

Example

The following command continues playing or recording the "newsound" device:

```
resume newsound
```

resume (Basic)

Command	Arguments
resume *device_id*	**[notify]** **[wait]**

save (Basic)

Syntax

save *device_id* [*filename*] [**notify**] [**wait**]

The **save** command saves the MCI element.

Parameters

You can specify the following optional item:

filename
: Specifies the destination path and file.

Comments

The *filename* parameter is required if the device was opened using the **new** device ID.

Example

The following command saves the data in the "newsound" device to C:\SOUNDS\NEWSND.WAV:

```
save newsound c:\sounds\newsnd.wav
```

save (Basic)

Command	Arguments
save *device_id*	**[***file_name***]** **[notify]** **[wait]**

seek (Basic)

Syntax

seek *device_id parameter* [**notify**] [**wait**]

The **seek** command moves to the specified position and stops.

Parameters

Specify one of the following items for *parameter*:

to *position*
: Specifies the position to stop the seek.

to start
: Seeks to the start of the media.

to end
: Seeks to the end of the media.

Comments

Before issuing any commands that use position values, you should set the desired time format using the **set** command.

Example

The following command seeks to the start of the media file associated with the "mysound" device:

```
seek mysound to start
```

seek (Basic)

Command	Arguments
seek *device_id*	**{to** *position* **| to start | to end}** **[notify]** **[wait]**

set (Basic)

Syntax

set *device_id parameters* [**notify**] [**wait**]

The **set** command establishes control settings for the driver.

Parameters

Specify one or more of the following items for *parameters*:

audio all off
Disables audio output.

audio all on
Enables audio output.

audio left off
Disables output to the left audio channel.

audio left on
Enables output to the left audio channel.

audio right off
Disables output to the right audio channel.

audio right on
Enables output to the right audio channel.

door closed
Loads the media and closes the door if possible.

door open
Opens the door and ejects the media if possible.

time format milliseconds
Sets the time format to milliseconds. All commands that use position values will assume milliseconds. You can abbreviate milliseconds as **ms**.

video off
Disables video output.

video on
Enables video output.

Example

The following command sets the "mysound" device to use milliseconds as the time format:

```
set mysound time format ms
```

set (Basic)

Command	Arguments
set *device_id*	**[audio all off** **| audio all on** **| audio left off** **| audio left on** **| audio right off** **| audio right on** **| video off** **| video on]** **[door closed | door open]** **[time format milliseconds | time format ms]** **[notify]** **[wait]**

status (Basic)

Syntax

status *device_id parameter* [**notify**] [**wait**]

The **status** command gets status information for the device. This command is also listed as a required command. As a basic command, **status** adds options for devices with linear media.

Parameters

Specify one of the following items for *parameter*:

current track
Returns the current track.

length
Returns the total length of the media.

length track *track_number*
Returns the length of the track specified by *track_number*.

number of tracks
Returns the number of tracks on the media.

position
Returns the current position.

position track *track_number*
Returns the position of the start of the track specified by *track_number*.

ready
Returns **true** if the device is ready to play.

start position
Returns the starting position of the media.

time format
Returns the current time format.

Comments

Before issuing any commands that use position values, you should set the desired time format using the **set** command.

Example

The following command returns the time format used by the "mysound" device:

```
status mysound time format
```

status (Basic)

Command	Arguments
status *device_id*	**{current track** **| length** **| length track** *track_number* **| mode** **| number of tracks** **| position** **| position track** *track_number* **| ready** **| start position** **| time format}** **[notify]** **[wait]**

stop (Basic)

Syntax

stop *device_id* [**notify**] [**wait**]

The **stop** command stops playback or recording.

Example

The following command stops playback or recording on the "mysound" device:

```
stop mysound
```

stop (Basic)

Command	Arguments
stop *device_id*	**[notify]** **[wait]**

capability (Animation)

Syntax

capability *device_id parameter* [**notify**] [**wait**]

The **capability** command requests information about the capabilities of the animation driver.

Parameters

Specify one of the following items for *parameter*:

can eject
Returns **true** if the device can eject the media.

can play
Returns **true** if the device can play.

can record
Returns **false**. Animation devices cannot record.

can reverse
Returns **true** if the animation device can play in reverse.

can save
Returns **false**. Animation devices cannot save data.

can stretch
Returns **true** if the device can stretch frames to fill a given display rectangle.

compound device
Returns **true** if the device supports an element name.

device type
Animation devices return **animation**.

fast play rate
Returns fast play rate in frames per second.

has audio
Returns **true** if the device supports audio playback.

has video
Returns **true**.

normal play rate
Returns normal play rate in frames per second.

slow play rate
Returns the slow play rate in frames per second.

uses files
Returns **true** if the element of a compound device is a file pathname.

uses palettes
Returns **true** if the device uses palettes.

windows

Returns the number of windows the device can support.

Example

The following command returns the fast play rate of the "movie" device:

```
capability movie fast play rate
```

capability (Animation)

Command	Arguments
capability *device_id*	**{can eject** **| can play** **| can record** **| can reverse** **| can save** **| can stretch** **| compound device** **| device type** **| fast play rate** **| has audio** **| has video** **| normal play rate** **| slow play rate** **| uses files** **| uses palettes** **| windows}** **[notify]** **[wait]**

close (Animation)

Syntax

close *device_id* [**notify**] [**wait**]

The **close** command closes a device element and any associated resources.

Example

The following command closes the "movie" device:

```
close movie
```

close (Animation)

Command	Arguments
close *device_id*	**[notify]** **[wait]**

info (Animation)

Syntax

info *device_id parameter* [**notify**] [**wait**]

The **info** command gets textual information from the animation driver.

Parameters

Specify one of the following items for *parameter*:

file
: Returns the name of the file used by the animation device.

product
: Returns the product name and model of the current device.

window text
: Returns the caption of the window used by the device.

Example

The following command returns the caption of the display window for the "movie" device:

```
info movie window text
```

info (Animation)

Command	Arguments
info *device_id*	**[file | product | window text]** **[notify]** **[wait]**

open (Animation)

Syntax

open *device* [*parameters*] [**notify**] [**wait**]

The **open** command initializes the animation device.

Parameters

You can specify one or more of the following optional items for *parameters*:

alias *device_alias*
Specifies an alternate name for the animation element. If specified, it must be used as the *device_id* in subsequent commands.

nostatic
Indicates that the device should reduce the number of static (system) colors in the palette. Reducing the number of static colors increases the number of colors controlled by the animation.

parent *hwnd*
Specifies the window handle of the parent window.

shareable
Initializes a device element as shareable. Subsequent attempts to open it fail unless you specify **shareable** in both the original and subsequent **open** commands. MCI returns an error if the device is already open and not shareable.

style *style_type*
Indicates a window style.

style child
Opens a window with a child window style.

style overlapped
Opens a window with an overlapped window style.

style popup
Opens a window with a popup window style.

type *device_type*
Specifies the device type of the device element. As an alternative to **type**, MCI can use the [mci extension] entries in the SYSTEM.INI file to select the controlling device based on the extension used by the device element. You can also use the *device_name*!*element_name* abbreviation described in Combining the Device Name and Element Name.

Example

The following command opens animation file \MMM\SNQLMFLL.MMM:

```
open \mmm\snqlmfll.mmm type mmmovie alias movie style 13369344
```

The command specifies device type "mmmovie," assigns an alias of "movie," and specifies an overlapped window with a system menu, caption, thick border, and maximize box (combination of the window styles WS_OVERLAPPED, WS_SYSMENU, WS_BORDER, WS_THICKFRAME, and WS_MAXIMIZEBOX).

open (Animation)

Command	Arguments
open *device*	**[alias** *device_alias***]** **[nostatic]** **[parent** *hwnd***]** **[shareable]** **[style child** **| style overlapped** **| style popup]** **[type** *device_type***]** **[notify]** **[wait]**

pause (Animation)

Syntax

pause *device_id* [**notify**] [**wait**]

The **pause** command pauses playback of the animation. If the animation is stopped, **pause** displays the animation window if it is not already visible and in the foreground.

Example

The following command pauses the "movie" device:

```
pause movie
```

pause (Animation)

Command	Arguments
pause *device_id*	**[notify]** **[wait]**

play (Animation)

Syntax

play *device_id* [*parameters*] [**notify**] [**wait**]

The **play** command starts playing the animation sequence.

Parameters

You can specify one or more of the following optional items for *parameters*:

fast
Plays the animation sequence at a fast rate.

from *position*
Specifies the starting position for the playback. If the **from** parameter is not specified, playback begins at the current frame.

to *position*
Specifies the ending position for the playback. If the **to** parameter is not specified, play stops at the end of the media. You cannot specify the **reverse** key word with an ending position.

reverse
Specifies that the play direction is backwards. You cannot specify an ending playback position with the **reverse** key word.

scan
Plays the animation sequence as fast as possible without disabling video.

slow
Plays the animation sequence at a slow rate.

speed *fps*
Plays the animation sequence at the specified speed *fps*. Speed is specified in frames per second.

Comments

Before issuing any commands that use position values, you should set the desired time format using the **set** command.

Example

The following command plays the "movie" device, starting at the current position and continuing until the end of the media:

```
play movie
```

A notification will be sent when the playback is finished.

play (Animation)

Command	Arguments
play *device_id*	**[fast** | **slow** | **speed** *fps***]** **[from** *position***]** **[to** *position*| **reverse]** **[scan]** **[notify]** **[wait]**

put (Animation)

Syntax

put *device_id parameter* [**notify**] [**wait**]

The **put** command defines the area of the source image and destination window used for display.

Parameters

Specify one of the following items for *parameter*:

destination
: Sets the whole window as the destination window.

destination at *rectangle*
: Specifies a rectangle for the area of the window used to display the image. The *rectangle* coordinates are relative to the window origin and are specified as *X1 Y1 X2 Y2*. The coordinates *X1 Y1* specify the top, left corner, and the coordinates *X2 Y2* specify the width and height of the rectangle.

  When an area of the display window is specified, and the device supports stretching, the source image is stretched to the destination offset and extent.

source
: Selects the whole image for display in the destination window.

source at *rectangle*
: Specifies a rectangle for the image area used for display. The *rectangle* coordinates are relative to the image origin and are specified as *X1 Y1 X2 Y2*. The coordinates *X1 Y1* specify the top, left corner, and the coordinates *X2 Y2* specify the width and height of the rectangle.

  When an area of the source image is specified, and the device supports stretching, the source image is stretched to the destination offset and extent.

Example

The following command sets the display area for the movie device. It specifies an offset of 10 pixels from the top left corner of the playback window. The playback area is clipped to 250 pixels wide and 340 pixels high.

```
put movie destination at 10 10 250 340
```

put (Animation)

Command	Arguments
put *device_id*	**{destination** **| destination at** *destination_rectangle* **| source** **| source at** *source_rectangle***}** **[notify]** **[wait]**

realize (Animation)

Syntax

realize *device_id parameter* [**notify**] [**wait**]

The **realize** command tells the device to select and realize its palette into the display context of the displayed window.

Parameters

Specify one of the following items for *parameter*:

background
Realizes the palette as a background palette.

normal
Realizes the palette normally.

Example

The following command tells the "myvideo" device to realize its palette:

```
realize myvideo normal
```

realize (Animation)

Command	Arguments
realize *device_id*	**{background | normal}** **[notify]** **[wait]**

resume (Animation)

Syntax

resume *device_id* [**notify**] [**wait**]

The **resume** command continues playback after a pause.

Example

The following command continues playback of the "movie" device:

```
resume movie
```

resume (Animation)

Command	Arguments
resume *device_id*	**[notify]** **[wait]**

seek (Animation)

seek *device_id parameter* [**notify**] [**wait**]

The **seek** command moves to the specified position and stops.

Parameters

Specify one of the following items for *parameter*:

to *position*
: Specifies the position to stop the seek.

to start
: Seeks to the start of the device element.

to end
: Seeks to the end of the device element.

Comments

Before issuing any commands that use position values, you should set the desired time format using the **set** command.

Example

The following command seeks to the beginning of the animation file associated with the "movie" device:

```
seek movie to start
```

seek (Animation)

Command	Arguments
seek *device_id*	**[to** *position* **| to start | to end]** **[notify]** **[wait]**

set (Animation)

Syntax

set *device_id parameters* [**notify**] [**wait**]

The **set** command establishes control settings for the driver.

Parameters

Specify one or more of the following items for *parameters*:

audio all off
Disables audio output.

audio all on
Enables audio output.

audio left off
Disables output to the left audio channel.

audio left on
Enables output to the left audio channel.

audio right off
Disables output to the right audio channel.

audio right on
Enables output to the right audio channel.

time format frames
Sets the time format to frames. All commands that use position values will assume frames. When the device is opened, frames is the default mode.

time format milliseconds
Sets the time format to milliseconds. All commands that use position values will assume milliseconds. You can abbreviate milliseconds as **ms**.

video off
Disables video output.

video on
Enables video output.

Example

The following command cancels audio playback on the "movie" device and sets the time format to frames:

```
set movie time format frames audio all off
```

set (Animation)

Command	Arguments
set *device_id*	**[audio all off** **| audio all on** **| audio left off** **| audio left on** **| audio right off** **| audio right on** **| video off** **| video on]** **[time format milliseconds** **| time format ms** **| time format frames]** **[notify]** **[wait]**

status (Animation)

Syntax

status *device_id parameter* [**notify**] [**wait**]

The **status** command gets status information for the device.

Parameters

Specify one of the following items for *parameter*:

current track
Returns the current track.

forward
Returns **true** if the play direction is forward or if the device is not playing.

length
Returns the total number of frames.

length track *track_number*
Returns the total number of frames in the track specified by *track_number*.

media present
Returns **true** if the media is inserted in the device; otherwise it returns **false**.

mode
Returns **not ready**, **paused**, **playing**, **seeking**, or **stopped** for the current mode.

number of tracks
Returns the number of tracks on the media.

palette handle
Returns the handle of the palette used for the animation.

position
Returns the current position.

position track *number*
Returns the position of the start of the track specified by *number*.

ready
Returns **true** if the device is ready.

speed
Returns the current speed of the device in frames per second.

start position
Returns the starting position of the media.

stretch
Returns **true** if stretching is enabled.

time format
Returns the current time format.

window handle

Returns the handle of the window used for the animation.

Comments

Before issuing any commands that use position values, you should set the desired time format using the **set** command.

Example

The following command a handle to the current palette used by the "movie" device:

```
status movie palette handle
```

status (Animation)

Command	Arguments
status *device_id*	**{current track** **| forward** **| length** **| length track** *track_number* **| media present** **| mode** **| number of tracks** **| palette handle** **| position** **| position track** *track_number* **| ready** **| speed** **| start position** **| stretch** **| time format** **| window handle}** **[notify]** **[wait]**

step (Animation)

Syntax

step *device_id* [*parameter*] [**notify**] [**wait**]

The **step** command steps the play one or more frames forward or reverse. The default action is to step forward one frame.

Parameters

You can specify one or both of the following optional items for *parameter*:

by *frames*
: Indicates the number of frames to step. If you specify a negative *frames* value, you cannot specify the **reverse** flag.

reverse
: Step the frames in reverse.

Example

The following command plays five frames of the animation file associated with the movie device, starting at the current frame:

```
step movie by 5
```

step (Animation)

Command	Arguments
step *device_id*	**[by** *frames***]** **[reverse]** **[notify]** **[wait]**

stop (Animation)

Syntax

stop *device_id* [**notify**] [**wait**]

The **stop** command stops playback.

Example

The following command stops playback of the "movie" device:

```
stop movie
```

stop (Animation)

Command	Arguments
stop *device_id*	**[notify]** **[wait]**

update (Animation)

Syntax

update *device_id* **hdc** *hDC* [**at** *rectangle*] [**notify**] [**wait**]

The **update** command repaints the current frame into the specified display context.

Parameters

You must specify the handle to the device context to paint; you can also specify the clipping rectangle:

hdc *hDC*
: Specifies the handle to the device context to update.

at *rectangle*
: Specifies the clipping rectangle.

Example

The following command updates the entire display window used by the "movie" device. The number following **hdc** is a handle to a device context obtained from the **BeginPaint** function:

```
update movie hdc 203
```

update (Animation)

Command	Arguments
update *device_id*	**[at** *update_rect***]** **[hdc** *hdc***]** **[notify]** **[wait]**

where (Animation)

Syntax

where *device_id parameter* [**notify**] [**wait**]

The **where** command gets the rectangle specifying the source or destination area.

Parameters

Specify one of the following items for *parameter*:

destination
: Requests the destination offset and extent.

source
: Requests the source offset and extent.

Example

The following command returns the display rectangle of the "movie" device:

```
where movie destination
```

where (Animation)

Command	Arguments
where *device_id*	**{destination | source}** **[notify]** **[wait]**

window (Animation)

Syntax

window *device_id parameters* [**notify**] [**wait**]

The **window** command controls the animation display window. You can change the display characteristics of the window or provide a display window for the driver to use in place of the default display window.

Generally, animation devices create a window when opened but don't display the window until they receive a **play** command. If your application provides a window to the driver, your application is responsible for managing the messages sent to the window.

The window command provides several flags that let you manipulate the window. Since you can use the **status** command to get the handle to the driver display window, you can also use the standard window manager functions (like **ShowWindow**) to manipulate the window.

Parameters

Specify one or more of the following items for *parameters*:

fixed
Disables stretching of the image.

handle *window_handle*
Specifies the handle of the destination window to use instead of the default window.

handle default
Specifies that the animation device should set the display window back to the driver's default window.

state hide
Hides the display window.

state iconic
Displays the window as iconic.

state maximized
Maximizes the display window.

state minimize
Minimizes the display window and activates the top-level window in the window-manager's list.

state minimized
Minimizes the display window.

state no action
Displays the window in its current state. The window that is currently active remains active.

state no activate
Displays the window in its most recent size and state. The window that is currently active remains active.

state normal
Activates and displays the display window in its original size and position.

state show
Shows the display window.

stretch
Enables stretching of the image.

text *caption*
Specifies the *caption* for the display window.

Example

The following command displays and sets the caption for the "movie" playback window:

```
window movie text "Welcome to the Movies" state show
```

window (Animation)

Command	Arguments
window *device_id*	**[fixed | stretch]** **[handle** *window_handle* **| handle default]** **[state hide** **| state iconic** **| state maximized** **| state minimize** **| state minimized** **| state no action** **| state no activate** **| state normal** **| state show]** **[text** *caption_text***]** **[notify]** **[wait]**

capability (CD Audio)

Syntax

capability *device_id parameter* [**notify**] [**wait**]

The **capability** command requests information about the capabilities of the CD audio device.

Parameters

Specify one of the following items for *parameter*:

can eject
Returns **true** if the CD audio device can eject the media.

can play
Returns **true** if the CD audio device can play the media.

can record
Returns **false**. CD audio devices cannot record.

can save
Returns **false**. CD audio devices cannot save data.

compound device
Returns **false**. CD audio devices are simple devices.

device type
Returns **cdaudio**.

has audio
Returns **true**.

has video
Returns **false**. CD audio devices don't support video.

uses files
Returns **false**. Simple devices don't use files.

Example

The following command returns **true** if the device can eject the compact disc:

```
capability cdaudio can eject
```

capability (CD Audio)

Command	Arguments
capability *device_id*	**{can eject** **| can play** **| can record** **| can save** **| compound device** **| device type** **| has audio** **| has video** **| uses files}** **[notify]** **[wait]**

close (CD Audio)

Syntax

close *device_id* [**notify**] [**wait**]

The **close** command closes the device.

Example

The following command closes the "cdaudio" device:

```
close cdaudio
```

close (CD Audio)

Command	Arguments
close *device_id*	**[notify]** **[wait]**

info (CD Audio)

Syntax

info *device_id parameter* [**notify**] [**wait**]

The **info** command obtains textual information from a device.

Parameters

Specify the following item for *parameter*:

product
: Returns the product name and model of the CD audio device. The MCICDA device driver returns **CD Audio**.

Example

The following command returns the product name of the CD audio device:

```
info cdaudio product
```

info (CD Audio)

Command	Arguments
info *device_id*	**[product]** **[notify]** **[wait]**

open (CD Audio)

Syntax

open *device* [*parameters*] [**notify**] [**wait**]

The **open** command initializes the device. MCI reserves "cdaudio" for the compact disc audio device type.

Parameters

You can specify one or more of the following items for *parameters*:

alias *device_alias*
: Specifies an alternate name for the given device. If specified, it must be used as the *device_id* in subsequent commands.

shareable
: Initializes the device as shareable. Subsequent attempts to open it fail unless you specify **shareable** in both the original and subsequent **open** commands. MCI returns an error if the device is already open and not shareable.

Example

The following command opens the "cdaudio" device:

```
open cdaudio
```

open (CD Audio)

Command	Arguments
open *device*	**[alias** *device_alias***]** **[shareable]** **[notify]** **[wait]**

pause (CD Audio)

Syntax

pause *device_id* [**notify**] [**wait**]

The **pause** command pauses playing. With the MCICDA driver, the **pause** command works the same as the **stop** command.

Example

The following command pauses the "cdaudio" device:

```
pause cdaudio
```

pause (CD Audio)

Command	Arguments
pause *device_id*	**[notify]** **[wait]**

play (CD Audio)

Syntax

play *device_id* [*parameters*] [**notify**] [**wait**]

The **play** command starts playing the audio disc.

Parameters

You can specify one or more of the following optional items for *parameters*:

from *position*
: Specifies the position to start playback. If the **from** parameter is not specified, playback begins at the current position. If the **from** position is greater than the end position of the disc, or if the **from** position is greater than the **to** position, the driver returns an error.

to *position*
: Specifies the position to stop playback. By default, playback continues to the end of the disc. If the **to** position is greater than the length of the disc, MCI returns an error.

Comments

Before issuing any commands that use position values, you should set the desired time format using the **set** command.

Example

The following command plays tracks two through five of the audio disc (assuming the time format is set to TMSF):

```
play cdaudio from 2 to 6
```

play (CD Audio)

Command	Arguments
play *device_id*	**[from** *position***]** **[to** *position***]** **[notify]** **[wait]**

resume (CD Audio)

Syntax

resume *device_id* [**notify**] [**wait**]

The **resume** command continues playback of a paused device. The MCICDA device driver does not support this command.

Example

The following command continues playback of the "myredbook" device:

```
resume myredbook
```

resume (CD Audio)

Command	Arguments
resume *device_id*	**[notify]** **[wait]**

seek (CD Audio)

Syntax

seek *device_id parameter* [**notify**] [**wait**]

The **seek** command moves to the specified position and stops. If the device was playing when the command was issued, playback is stopped.

Parameters

Specify one of the following items for *parameter*:

to *position*
: Specifies the destination position for the seek. If it is greater than the length of disc, MCI returns an out-of-range error.

to start
: Specifies seek to the start of the audio data on the disc.

to end
: Specifies seek to the end of the audio data on the disc.

Comments

Before issuing any commands that use position values, you should set the desired time format using the **set** command.

Example

The following command seeks to track 2 of the disc (assuming the time format is set to TMSF):

```
seek cdaudio to 2
```

seek (CD Audio)

Command	Arguments
seek *device_id*	**[to** *position* **| to start | to end]** **[notify]** **[wait]**

set (CD Audio)

Syntax

set *device_id parameters* [**notify**] [**wait**]

The **set** command establishes control settings for the driver.

Parameters

Specify one or more of the following items for *parameters*:

audio all off
Disables audio output.

audio all on
Enables audio output.

audio left off
Disables output to the left audio channel.

audio left on
Enables output to the left audio channel.

audio right off
Disables output to the right audio channel.

audio right on
Enables output to the right audio channel.

door closed
Retracts the tray and closes the door if possible.

door open
Opens the door and ejects the tray if possible.

time format milliseconds
Sets the time format to milliseconds. All commands that use position values will assume milliseconds. You can abbreviate milliseconds as **ms**.

time format msf
Sets the time format to minutes, seconds, and frames. All commands that use position values will assume MSF. MSF is the default format for CD audio.

Specify an MSF value as *mm*:*ss*:*ff*, where *mm* is minutes, *ss* is seconds, and *ff* is frames. You can omit a field if it and all following fields are zero. For example, 3, 3:0, and 3:0:0 are valid ways to express 3 minutes.

The MSF fields have the following maximum values:

Minutes	99
Seconds	59
Tracks	74

time format tmsf
Sets the time format to tracks, minutes, seconds, and frames. All commands that use position values will assume TMSF.

Specify a TMSF value as *tt*:*mm*:*ss*:*ff*, where *tt* is tracks, *mm* is minutes, *ss* is seconds, and *ff* is

frames. You can omit a field if it and all following fields are zero. For example, 3, 3:0, 3:0:0, 3:0:0:0 all specify track 3.

The TMSF fields have the following maximum values:

Tracks	99
Minutes	99
Seconds	59
Frames	74

Example

The following command sets the time format of the cdaudio device to tracks, minutes, seconds, and frames:

```
set cdaudio time format tmsf
```

set (CD Audio)

Command	Arguments
set *device_id*	**[audio all off** **| audio all on** **| audio left off** **| audio left on** **| audio right off** **| audio right on** **| video off** **| video on]** **[door closed | door open]** **[time format milliseconds** **| time format ms** **| time format msf** **| time format tmsf]** **[notify]** **[wait]**

status (CD Audio)

Syntax

status *device_id parameter* [**notify**] [**wait**]

The **status** command gets status information for the device.

Parameters

Specify one of the following items for *parameter*:

current track
Returns the current track.

length
Returns the total length of the disc.

length track *track_number*
Returns the length of the track specified by *track_number*.

media present
Returns **true** if the disc is inserted in the drive; otherwise, it returns **false**.

mode
Returns one of the following values indicating the current mode of the device:

not ready
open
paused
playing
seeking
stopped

number of tracks
Returns the number of tracks on the disc.

position
Returns the current position.

position track *track_number*
Returns the starting position of the track specified by *track_number*.

ready
Returns **true** if the device is ready.

start position
Returns the starting position of the disc or device element.

time format
Returns the current time format.

Comments

Before issuing any commands that use position values, you should set the desired time format using the **set** command.

Example

The following command returns the number of tracks on the current disc:

```
status cdaudio number of tracks
```

status (CD Audio)

Command	Arguments
status *device_id*	**{current track** **| length** **| length track** *track_number* **| media present** **| mode** **| number of tracks** **| position** **| position track** *track_number* **| ready** **| start position** **| time format}** **[notify]** **[wait]**

stop (CD Audio)

Syntax

stop *device_id* [**notify**] [**wait**]

The **stop** command stops playback.

Example

The following command stops the "cdaudio" device:

```
stop cdaudio
```

stop (CD Audio)

Command	Arguments
stop *device_id*	**[notify]** **[wait]**

capability (MIDI Sequencer)

Syntax

capability *device_id parameter* [**notify**] [**wait**]

The **capability** command requests information about a particular capability of the MIDI sequencer.

Parameters

Specify one of the following items for *parameter*:

can eject
Returns **false**. Sequencers cannot eject.

can play
Returns **true**.

can record
Returns **true** if the sequencer can record MIDI data. The MCISEQ sequencer cannot record and returns **false**.

can save
Returns **true** if the sequencer can save MIDI data. The MCISEQ sequencer cannot save data and returns **false**.

compound device
Returns **true**; sequencers are compound devices.

device type
Returns **sequencer**.

has audio
Returns **true**. Sequencers support playback.

has video
Returns **false**. Sequencers don't support video.

uses files
Returns **true**. Sequencers use files for operation.

Example

The following command returns true if the "music" device can record:

```
capability music can record
```

capability (MIDI Sequencer)

Command	Arguments
capability *device_id*	**{can eject** **| can play** **| can record** **| can save** **| compound device** **| device type** **| has audio** **| has video** **| uses files}** **[notify]** **[wait]**

close (MIDI Sequencer)

Syntax

close *device_id* [**notify**] [**wait**]

The **close** command closes the sequencer device, as well as the associated port and file.

Example

The following command closes the "music" device:

```
close music
```

close (MIDI Sequencer)

Command	Arguments
close *device_id*	**[notify]** **[wait]**

info (MIDI Sequencer)

Syntax

info *device_id parameter* [**notify**] [**wait**]

The **info** command gets textual information from the device.

Parameters

Specify one of the following items for *parameter*:

file
Returns the filename of the current MIDI file.

product
Returns the product name of the sequencer. The MCISEQ sequencer returns **MIDI Sequencer**.

Example

The following command returns the filename of the MIDI file associated with the "music" device:

```
info music file
```

info (MIDI Sequencer)

Command	Arguments
info *device_id*	**[file | product]** **[notify]** **[wait]**

open (MIDI Sequencer)

Syntax

open *device* [*parameters*] [**notify**] [**wait**]

The **open** command initializes the device.

MIDI files played using the MCISEQ sequencer should be authored to conform to the Microsoft MIDI file authoring guidelines, which are discussed in the *Multimedia Programmer's Guide*. If the file does not conform, the MCISEQ sequencer displays a warning dialog box when you issue the **play** command for the file.

Parameters

You can specify one or more of the following optional items for *parameters*:

alias *device_alias*
Specifies an alternate name for the sequencer element. If specified, it must also be used as the *device_id* in subsequent commands.

shareable
Initializes the sequencer element as shareable. Subsequent attempts to open it fail unless you specify shareable in both the original and subsequent **open** commands. MCI returns an invalid device error if the device is already open and not shareable. The MCISEQ sequencer does not support shared files.

type *device_type*
Specifies the device type of the device element. MCI reserves "sequencer" for the MIDI sequencer device type. As an alternative to **type**, MCI can use the [mci extension] entries in the SYSTEM.INI file to select the sequencer based on the extension used by the device element. You can also use the *device_name*!*element_name* abbreviation described in Combining the Device Name and Element Name.

Example

The following command opens MIDI filename C:\MIDI\CLAVIER.MID and assigns device alias "clavier":

```
open sequencer!c:\midi\clavier.mid alias clavier
```

open (MIDI Sequencer)

Command	Arguments
open *device*	**[alias** *device_alias***]** **[shareable]** **[type** *device_type***]** **[notify]** **[wait]**

pause (MIDI Sequencer)

Syntax

pause *device_id* [**notify**] [**wait**]

The **pause** command pauses playing.

Example

The following command pauses playing of the "clavier" device:

```
pause clavier
```

pause (MIDI Sequencer)

Command	Arguments
pause *device_id*	**[notify]**
	[wait]

play (MIDI Sequencer)

Syntax

play *device_id* [*parameters*] [**notify**] [**wait**]

The **play** command starts playing the sequencer.

MIDI files played using the MCISEQ sequencer should be authored to conform to the Microsoft MIDI file authoring guidelines, which are discussed in the *Multimedia Programmer's Guide*. If the file does not conform, the MCISEQ sequencer displays a warning dialog box when you issue the **play** command for the file.

Parameters

You can specify one or more of the following optional items for *parameters*:

from *position*
: Specifies a starting position for the playback. If the **from** parameter is not specified, playback begins at the current position.

to *position*
: Specifies an ending position for the playback. If the **to** parameter is not specified, playback ends at the end of the media.

Comments

Before issuing any commands that use position values, you should set the desired time format using the **set** command.

Example

The following command starts playing the "clavier" device from the start of the MIDI file:

```
play clavier from 0
```

play (MIDI Sequencer)

Command	Arguments
play *device_id*	**[from** *position***]** **[to** *position***]** **[notify]** **[wait]**

record (MIDI Sequencer)

Syntax

record *device_id* [*parameters*] [**notify**] [**wait**]

The **record** command starts recording MIDI data. All data recorded after a file is opened is discarded if the file is closed without saving it. The MCISEQ sequencer does not support recording.

Parameters

You can specify one or more of the following optional items for *parameters*:

from *position*
Specifies a starting position for the recording. If the **from** parameter is not specified, the sequencer starts recording at the current position.

to *position*
Specifies an ending position for the recording. If the **to** parameter is not specified, the sequencer records until it receives a **stop** or **pause** command.

insert
Specifies that new data is added to the device element.

overwrite
Specifies that new data will replace data in the device element.

Comments

Before issuing any commands that use position values, you should set the desired time format using the **set** command.

Example

The following command starts the "clavier" device recording at the current position:

```
record clavier
```

record (MIDI Sequencer)

Command	Arguments
record *device_id*	**[from** *position***]** **[to** *position***]** **[insert** | **overwrite]** **[notify]** **[wait]**

resume (MIDI Sequencer)

Syntax

resume *device_id* [**notify**] [**wait**]

The **resume** command continues playing or recording on a paused device. The MCISEQ sequencer does not support this command.

Example

The following command continues playing or recording of the "clavier" device:

```
resume clavier
```

resume (Sequencer)

Command	Arguments
resume *device_id*	**[notify]** **[wait]**

save (MIDI Sequencer)

Syntax

save *device_id* [*filename*] [**notify**] [**wait**]

The **save** command saves the MCI element. The MCISEQ sequencer does not support this command.

Parameters

You can specify the following optional item:

filename
: The *filename* specifies the destination path and file.

Comments

The *filename* parameter is required if the device was opened using the **new** device ID.

Example

The following command saves the MIDI data recorded into the "clavier" device as filename C:\MIDI\MYMIDI.MID:

```
save clavier c:\midi\mymidi.mid
```

save (MIDI Sequencer)

Command	Arguments
save *device_id*	**[***file_name***]** **[notify]** **[wait]**

seek (MIDI Sequencer)

Syntax

seek *device_id parameter* [**notify**] [**wait**]

The **seek** command moves to the specified position in the file.

Parameters

Specify one of the following items for *parameter*:

to *position*
: Specifies to seek to *position*.

to start
: Seeks to the start of the sequence.

to end
: Seeks to the end of the sequence.

Comments

Before issuing any commands that use position values, you should set the desired time format using the **set** command.

Example

The following command moves to the beginning of the MIDI file associated with the "clavier" device:

```
seek clavier to start
```

seek (MIDI Sequencer)

Command	Arguments
seek *device_id*	**[to** *position* **| to start | to end]** **[notify]** **[wait]**

set (MIDI Sequencer)

Syntax

set *device_id parameters* [**notify**] [**wait**]

Parameters

The set command specifies various settings for the driver.

Specify one or more of the following items for *parameters*:

audio all off
Disables audio output. The MCISEQ sequencer does not support this option.

audio all on
Enables audio output. The MCISEQ sequencer does not support this option.

audio left off
Disables output to the left audio channel. The MCISEQ sequencer does not support this option.

audio left on
Enables output to the left audio channel. The MCISEQ sequencer does not support this option.

audio right off
Disables output to the right audio channel. The MCISEQ sequencer does not support this option.

audio right on
Enables output to the right audio channel. The MCISEQ sequencer does not support this option.

master MIDI
Sets the MIDI sequencer as the synchronization source. Synchronization data is sent in MIDI format. The MCISEQ sequencer does not support this option.

master none
Inhibits the sequencer from sending synchronization data. The MCISEQ sequencer does not support this option.

master SMPTE
Sets the MIDI sequencer as the synchronization source. Synchronization data is sent in SMPTE format. The MCISEQ sequencer does not support this option.

offset *time*
Sets the SMPTE offset *time*. The offset is the beginning time of a SMPTE based sequence. The *time* is expressed as *hh*:*mm*:*ss*:*ff*, where *hh* is hours, *mm* is minutes, *ss* is seconds, and *ff* is frames.

port *port_number*
Sets the MIDI port receiving the MIDI messages. This command will fail if the port you are trying to open is being used by another application.

port mapper
Sets the MIDI mapper as the port receiving the MIDI messages. This command will fail if the MIDI mapper or a port it needs is being used by another application.

port none
Disables the sending of MIDI messages. This command also closes a MIDI port.

slave file
Sets the MIDI sequencer to use file data as the synchronization source. This is the default.

slave MIDI
Sets the MIDI sequencer to use incoming data MIDI for the synchronization source. The sequencer recognizes synchronization data with the MIDI format. The MCISEQ sequencer does not support this option.

slave none
Sets the MIDI sequencer to ignore synchronization data.

slave SMPTE
Sets the MIDI sequencer to use incoming MIDI data for the synchronization source. The sequencer recognizes synchronization data with the SMPTE format. The MCISEQ sequencer does not support this option.

tempo *tempo_value*
Sets the tempo of the sequence according to the current time format. For a ppqn-based file, the *tempo_value* is interpreted as beats per minute. For a SMPTE-based file, the *tempo_value* is interpreted as frames per second.

time format milliseconds
Sets the time format to milliseconds. All commands that use position values will assume milliseconds. You can abbreviate milliseconds as **ms**.

The sequence file sets the default format to ppqn or SMPTE.

time format song pointer
Sets the time format to song pointer (sixteenth notes). All commands that use position values will assume song pointer units. This option is valid only for a sequence of division type ppqn.

time format SMPTE 24
Sets the time format to SMPTE 24 frame rate. All commands that use position values will assume SMPTE format. The sequence file sets the default format to ppqn or SMPTE.

time format SMPTE 25
Sets the time format to SMPTE 25 frame rate. All commands that use position values will assume SMPTE format. The sequence file sets the default format to ppqn or SMPTE.

time format SMPTE 30
Sets the time format to SMPTE 30 frame rate. All commands that use position values will assume SMPTE format. The sequence file sets the default format to ppqn or SMPTE.

time format SMPTE 30 drop
Sets the time format to SMPTE 30 drop frame rate. All commands that use position values will assume SMPTE format. The sequence file sets the default format to ppqn or SMPTE.

Example

The following command sets the tempo on the "clavier" device to 170 beats per minute (assuming a ppqn-based file):

```
set clavier tempo 170
```

set (MIDI Sequencer)

Command	Arguments
set *device_id*	**[audio all off** **| audio all on** **| audio left off** **| audio left on** **| audio right off** **| audio right on** **| video off** **| video on]** **[master MIDI** **| master none** **| master SMPTE]** **[offset** *hmsf_value***]** **[port** *port_number* **| port mapper** **| port none]** **[slave file** **| slave MIDI** **| slave none** **| slave SMPTE]** **[tempo** *tempo_value***]** **[time format milliseconds** **| time format ms** **| time format smpte 24** **| time format smpte 25** **| time format smpte 30** **| time format smpte 30 drop** **| time format song pointer]** **[notify]** **[wait]**

status (MIDI Sequencer)

Syntax

status *device_id parameter* [**notify**] [**wait**]

The **status** command gets status information for the device.

Parameters

Specify one of the following items for *parameter*:

current track
Returns the current track number. The MCISEQ sequencer returns 1.

division type
Returns one of the following file division types:
PPQN
SMPTE 24 frame
SMPTE 25 frame
SMPTE 30 drop frame
SMPTE 30 frame
Use this information to determine the format of the MIDI file and the meaning of tempo and position information.

length
Returns the length of a sequence in the current time format. For ppqn files, this will be song pointer units. For SMPTE files, this will be expressed as *hh*:*mm*:*ss*:*ff*, where *hh* is hours, *mm* is minutes, *ss* is seconds, and *ff* is frames.

length track *track_number*
Returns the length of the sequence in the current time format. For ppqn files, this will be song pointer units. For SMPTE files, this will be expressed as *hh*:*mm*:*ss*:*ff*, where *hh* is hours, *mm* is minutes, *ss* is seconds, and *ff* is frames.

master
Returns **midi**, **none**, or **smpte** depending on the type of synchronization set.

media present
The sequencer returns **true**.

mode
Returns **not ready**, **paused**, **playing**, **seeking**, or **stopped**.

number of tracks
Returns the number of tracks. MCISEQ returns 1.

offset
Returns the offset of a SMPTE-based file. The offset is the start time of a SMPTE based sequence. The time is returned as *hh*:*mm*:*ss*:*ff*, where *hh* is hours, *mm* is minutes, *ss* is seconds, and *ff* is frames.

port
Returns the MIDI port number assigned to the sequence.

position
Returns the current position of a sequence in the current time format. For ppqn files, this will be song pointer units. For SMPTE files, this will be in form *hh*:*mm*:*ss*:*ff*, where *hh* is hours, *mm* is minutes, *ss* is seconds, and *ff* is frames.

position track *track_number*
Returns the current position of the track specified by *track_number* in the current time format. For ppqn files, this will be song pointer units. For SMPTE files, this will be in form *hh*:*mm*:*ss*:*ff*, where *hh* is hours, *mm* is minutes, *ss* is seconds, and *ff* is frames. The MCISEQ sequencer returns 0.

ready
Returns **true** if the device is ready.

slave
Returns **file**, **midi**, **none**, or **smpte** depending on the type of synchronization set.

start position
Returns the starting position of the media.

tempo
Returns the current tempo of a sequence in the current time format. For files with ppqn format, the tempo is in beats per minute. For files with SMPTE format, the tempo is in frames per second.

time format
Returns the current time format.

Comments

Before issuing any commands that use position values, you should set the desired time format using the **set** command.

Example

The following command returns the tempo for the clavier device:

```
status clavier tempo
```

status (MIDI Sequencer)

Command	Arguments
status *device_id*	**{current track** **| division type** **| length** **| length track** *track_number* **| master** **| media present** **| mode** **| number of tracks** **| offset** **| port** **| position** **| position track** *track_number* **| ready** **| slave** **| start position** **| tempo** **| time format}** **[notify]** **[wait]**

stop (MIDI Sequencer)

Syntax

stop *device_id* [**notify**] [**wait**]

The **stop** command stops playing.

Example

The following command stops playing on the "clavier" device:

```
stop clavier
```

stop (MIDI Sequencer)

Command	Arguments
stop *device_id*	**[notify]** **[wait]**

capability (Videodisc)

Syntax

capability *device_id parameter* [**notify**] [**wait**]

The **capability** command requests information about a particular capability of a device. The return information is for the type of disc inserted unless the **CAV** or **CLV** options are used to override the format. If no disc is present, information is returned for **CAV** discs.

Parameters

Specify one of the following items for *parameter*:

can eject
Returns **true** if the device can eject the disc. The MCIPIONR device returns **true**.

can play
Returns **true** if the device supports playing. The MCIPIONR device returns **true**.

can record
Returns **true** if the video device can record. The MCIPIONR device returns **false**.

can reverse
Returns **true** if the device can play in reverse, **false** otherwise. CLV discs return **false**.

can save
Returns **false**. MCI videodisc players cannot save data.

CAV
When combined with other items, **CAV** specifies that the return information applies to CAV format discs. This is the default if no disc is inserted.

CLV
When combined with other items, **CLV** specifies that the return information applies to CLV format discs.

compound device
Returns **false**. MCI videodisc players are simple devices.

device type
Returns **videodisc**.

fast play rate
Returns the standard fast play rate of the player in frames per second. Returns 0 if the device cannot play fast.

has audio
Returns **true** if the videodisc player has audio.

has video
Returns **true**.

normal play rate
Returns the normal play rate in frames per second.

slow play rate

Returns the standard slow play rate in frames per second. Returns 0 if the device cannot play slow.

uses files
Returns **false**. Simple devices don't use files.

Example

The following command returns the slow play rate of the "videodisc" device:

```
capability videodisc slow play rate
```

capability (Videodisc)

Command	Arguments
capability *device_id*	**{can eject** **| can play** **| can record** **| can reverse** **| can save** **| compound device** **| device type** **| fast play rate** **| has audio** **| has video** **| normal play rate** **| slow play rate** **| uses files}** **[CAV | CLV]** **[notify]** **[wait]**

close (Videodisc)

Syntax

close *device_id* [**notify**] [**wait**]

The **close** command closes the device.

Example

The following command closes the "videodisc" device:

```
close videodisc
```

close (Videodisc)

Command	Arguments
close *device_id*	**[notify]** **[wait]**

escape (Videodisc)

Syntax

escape *device_id string* [**notify**] [**wait**]

The **escape** command sends device-specific information to a device.

Parameters

Specify the following parameter:

string
: Specifies the custom information to send to the device.

Example

The following command sends the escape string "SA" to the "videodisc" device:

```
escape videodisc SA
```

escape (Videodisc)

Command	Arguments
escape *device_id*	**{***command_string***}**
	[notify]
	[wait]

info (Videodisc)

Syntax

info *device_id parameter* [**notify**] [**wait**]

The **info** command obtains textual information from a device.

Parameters

Specify the following item for *parameter*:

product
: Returns the product name of the device that the driver is controlling. The MCIPIONR device returns **Pioneer LD-V4200.**

Example

The following command returns the product name of the device controlled by the "videodisc" device:

```
info videodisc product
```

info (Videodisc)

Command	Arguments
info *device_id*	**[product]** **[notify]** **[wait]**

open (Videodisc)

Syntax

open *device* [*parameters*] [**notify**] [**wait**]

The **open** command initializes the device. MCI reserves "videodisc" for the compact disc audio device type.

Parameters

You can specify one or more of the following items for *parameters*:

alias *device_alias*
Specifies an alternate name for the given device. If specified, it must be used as the *device_id* in subsequent commands.

shareable
Initializes the device as shareable. Subsequent attempts to open it fail unless you specify **shareable** in both the original and subsequent **open** commands. MCI returns an invalid device error if the device is already open and not shareable.

Example

The following command opens the "videodisc" device:

```
open videodisc
```

open (Videodisc)

Command	Arguments
open *device*	**[alias** *device_alias***]** **[shareable]** **[notify]** **[wait]**

pause (Videodisc)

Syntax

pause *device_id* [**notify**] [**wait**]

The **pause** command pauses playing. For CAV discs, the video frame will freeze. For CLV discs, the player is stopped.

Example

The following command pauses the playback:

```
pause videodisc
```

pause (Videodisc)

Command	Arguments
pause *device_id*	**[notify]** **[wait]**

play (Videodisc)

Syntax

play *device_id* [*parameters*] [**notify**] [**wait**]

The **play** command starts playing the device.

Parameters

You can specify one or more of the following optional items for *parameters*:

fast
: Indicates that the device should play faster than normal. To determine the exact speed on a particular player, use the **status speed** command. To specify the speed more precisely, use the **speed** flag.

slow
: Indicates that the device should play slower than normal. To determine the exact speed on a particular player, use the **status speed** command. To specify the speed more precisely, use the **speed** flag. This flag applies only to CAV discs.

from *position*
: Specifies a starting position for the playback. If the **from** parameter is not specified,playback begins at the current position. The default positions are in frames for CAV discs and in hours, minutes, and seconds for CLV discs.

to *position*
: Specifies an ending position for the playback. If the **to** parameter is not specified, playback continues to the end of the media. The default positions are in frames for CAV discs and in hours, minutes, and seconds for CLV discs.

reverse
: Sets the play direction to backwards. This applies only to CAV discs.

scan
: Indicates the device should play as fast as possible, possibly with audio disabled. This flag applies only to CAV discs.

speed *integer*
: Specifies the rate of play in frames per second (for example, speed 15 means 15 frames per second). This applies only to CAV discs.

Comments

Before issuing any commands that use position values, you should set the desired time format using the **set** command.

Example

The following command starts fast playback:

```
play videodisc scan
```

play (Videodisc)

Command	Arguments
play *device_id*	**[fast** | **slow** | **speed** *fps***]** **[from** *position***]** **[reverse]** **[scan]** **[to** *position*| **reverse]** **[notify]** **[wait]**

resume (Videodisc)

Syntax

resume *device_id* [**notify**] [**wait**]

The **resume** command continues playing or recording on a paused device. The MCIPIONR driver does not support this command.

Example

The following command continues playback on the "videodisc" device:

```
resume videodisc
```

resume (Videodisc)

Command	Arguments
resume *device_id*	**[notify]**
	[wait]

seek (Videodisc)

Syntax

seek *device_id* [*parameter*] [**notify**] [**wait**]

The **seek** command searches using fast forward or fast reverse with video and audio off.

Parameters

You can specify one of the following optional items for *parameter*:

reverse
Indicates the seek direction on CAV discs is backwards. This modifier is invalid if **to** is specified.

to *position*
Specifies the position to stop the seek. If **to** is not specified, the seek continues to the end of the disc.

to start
Seeks to the start of the disc.

to end
Seeks to the end of the disc.

Comments

Before issuing any commands that use position values, you should set the desired time format using the **set** command.

Example

The following command seeks to the end of the videodisc:

```
seek videodisc to end
```

seek (Videodisc)

Command	Arguments
seek *device_id*	**[reverse** | **to** *position* | **to start** | **to end]** **[notify]** **[wait]**

set (Videodisc)

Syntax

set *device_id parameters* [**notify**] [**wait**]

The set command specifies various settings for the driver.

Parameters

Specify one or more of the following items for *parameters*:

audio all off
Disables audio output.

audio all on
Enables audio output.

audio left off
Disables output to the left audio channel.

audio left on
Enables output to the left audio channel.

audio right off
Disables output to the right audio channel.

audio right on
Enables output to the right audio channel.

door open
Opens the door and ejects the tray, if possible.

door closed
Retracts the tray and closes the door, if possible.

time format frames
Sets the position format to frames on CAV discs. All commands that use position values will assume frames. This is the default for CAV discs.

time format hms
Sets the time format to hours, minutes, and seconds. All commands that use position values will assume HMS. HMS is the default format for CLV discs.
Specify an HMS value as *hh*:*mm*:*ss*, where *hh* is hours, *mm* is minutes, and *ss* is seconds. You can omit a field if it and all following fields are zero. For example, 3, 3:0, and 3:0:0 are valid ways to express 3 hours.

time format milliseconds
Sets the time format to milliseconds. All commands that use position values will assume milliseconds. You can abbreviate milliseconds as **ms**.

time format track
Sets the position format to tracks. All commands that use position values will assume tracks.

video off
Disables video output.

video on

Enables video output.

Example

The following command closes the door of the device and sets the time format to milliseconds:

```
set videodisc time format ms door closed
```

set (Videodisc)

Command	Arguments
set *device_id*	**[audio all off** **| audio all on** **| audio left off** **| audio left on** **| audio right off** **| audio right on** **| video off** **| video on]** **[door closed | door open]** **[time format milliseconds** **| time format ms** **| time format frames** **| time format hms** **| time format track]** **[notify]** **[wait]**

spin (Videodisc)

Syntax

spin *device_id parameter* [**notify**] [**wait**]

The **spin** command starts the disc spinning or stops the disc from spinning.

Parameters

Specify one of the following items for *parameter*:

down
: Stops the disc from spinning.

up
: Starts the disc spinning.

Example

The following command spins up the "videodisc" device:

```
spin videodisc up
```

spin (Videodisc)

Command	Arguments
spin *device_id*	**[down** | **up]** **[notify]** **[wait]**

status (Videodisc)

Syntax

status *device_id parameter* [**notify**] [**wait**]

The **status** command gets status information for the device.

Parameters

Specify one of the following items for *parameter*:

current track
Returns the current track (chapter) number.

disc size
Returns either 8 or 12 to indicate the size of the loaded disc in inches.

forward
Returns **true** if the play direction is forward or if the device is not playing; **false** if the play direction is backward.

length
Returns the total length of the disc.

length track *track_number*
Returns the length of the track (chapter) specified by *track_number*.

media present
Returns **true** if a disc is inserted in the device, **false** otherwise.

media type
Returns either **CAV**, **CLV**, or **other** depending on the type of videodisc.

mode
Returns **not ready**, **open**, **paused**, **parked**, **playing**, **seeking**, or **stopped**.

number of tracks
Returns the number of tracks (chapters) on the disc. The MCIPIONR device does not support this option.

position
Returns the current position.

position track *track_number*
Returns the position of the start of the track (chapter) specified by *track_number*. The MCPIONR device does not support this option.

ready
Returns **true** if the device is ready.

side
Returns 1 or 2 to indicate which side of the disc is loaded.

speed
Returns the current speed in frames per second. The MCIPIONR videodisc player does not support this option.

start position
Returns the starting position of the disc.

time format
Returns the current time format.

Comments

Before issuing any commands that use position values, you should set the desired time format using the **set** command.

Example

The following command returns **true** if a videodisc is inserted in the device controlled by the "videodisc" driver:

```
status videodisc media present
```

status (Videodisc)

Command	Arguments
status *device_id*	**{current track** **| disc size** **| forward** **| length** **| length track** *track_number* **| media present** **| media type** **| mode** **| number of tracks** **| position** **| position track** *track_number* **| ready** **| side** **| speed** **| start position** **| time format}** **[notify]** **[wait]**

step (Videodisc)

Syntax

step *device_id* [*parameter*] [**notify**] [**wait**]

The **step** command steps the play one or more frames forward or reverse. The default action is to step forward one frame. The **step** command applies only to CAV discs.

Parameters

You can specify one or both of the following optional items for *parameter*:

by *frames*
: Specifies the number of *frames* to step. If you specify a negative *frames* value, you cannot specify the **reverse** flag.

reverse
: Step backward.

Example

The following command steps the videodisc one frame forward:

```
step videodisc
```

step (Videodisc)

Command	Arguments
step *device_id*	**[by]** *position* **[reverse]** **[notify]** **[wait]**

stop (Videodisc)

Syntax

stop *device_id* [**notify**] [**wait**]

The **stop** command stops playing.

Example

The following command stops the videodisc playback:

```
stop videodisc
```

stop (Videodisc)

Command	Arguments
stop *device_id*	**[notify]** **[wait]**

capability (Video Overlay)

Syntax

capability *device_id parameter* [**notify**] [**wait**]

The **capability** command requests information about the capabilities of the video overlay device.

Parameters

Specify one of the following items for *parameter*:

can eject
Returns **false**. Video overlay devices have no media to eject.

can freeze
Returns true if the device can freeze data in the frame buffer.

can play
Returns **false**.

can record
Returns **false**.

can save
Returns **true** if the device can save the current contents of the frame buffer to disk.

can stretch
Returns **true** if the device can stretch frames to fill a given display rectangle.

compound device
Returns **true** if the device supports an element name. Video overlay devices which support multiple windows may be compound devices.

device type
Returns **overlay**.

has audio
Returns **true** if the device supports audio playback.

has video
Returns **true**. Video overlay devices are video devices.

uses files
Returns **true** if elements of the device are filenames.

windows
Returns the number of simultaneous display windows the device can support.

Example

The following command returns **true** if the overlay device supports stretching:

```
capability vboard can stretch
```

capability (Video Overlay)

Command	Arguments
capability *device_id*	**{can eject** **| can freeze** **| can play** **| can record** **| can save** **| can stretch** **| compound device** **| device type** **| has audio** **| has video** **| uses files** **| windows}** **[notify]** **[wait]**

close (Video Overlay)

Syntax

close *device_id* [**notify**] [**wait**]

The **close** command closes a video overlay element and any associated resources.

Example

The following command closes the "vboard" device:

```
close vboard
```

close (Video Overlay)

Command	Arguments
close *device_id*	**[notify]** **[wait]**

freeze (Video Overlay)

Syntax

freeze *device_id* [*parameter*] [**notify**] [**wait**]

The **freeze** command disables video acquisition to the frame buffer. This is supported only if **capability can freeze** returns **true**.

Parameters

You can specify the following optional item for *parameter*:

at *rectangle*

Specifies the rectangular region that will have video acquisition disabled. To specify irregular acquisition regions, use a series of **freeze** and **unfreeze** commands. Some video overlay devices limit the complexity of the acquisition region.

The *rectangle* region is relative to the video buffer origin and is specified as *X1 Y1 X2 Y2*. The coordinates *X1 Y1* specify the top, left corner of the rectangle, and the coordinates *X2 Y2* specify the width and height.

Example

The following command disables video acquisition in a 100-pixel square at the top left corner of the video buffer:

```
freeze vboard at 0 0 100 100
```

freeze (Video Overlay)

Command	Arguments
freeze *device_id*	**[at** *rectangle***]** **[notify]** **[wait]**

info (Video Overlay)

Syntax

info *device_id parameter* [**notify**] [**wait**]

The **info** command gets textual information from the device.

Parameters

Specify one of the following items for *parameter*:

file
Returns the name of the file used by the video overlay device.

product
Returns the product name and model of the current video overlay device.

window text
Returns the caption of the window used by the video overlay device.

Example

The following command returns the caption of the destination window for the "vboard" device:

```
info vboard window text
```

info (Video Overlay)

Command	Arguments
info *device_id*	**[product | file | window text]** **[notify]** **[wait]**

load (Video Overlay)

Syntax

load *device_id* [*parameters*] [**notify**] [**wait**]

The **load** command loads the contents of the video buffer in a device specific format.

Parameters

You can specify the following optional parameters:

filename
: Specifies the file and pathname used to load the data.

at *rectangle*
: Specifies a rectangle relative to the video buffer origin. The *rectangle* is specified as *X1 Y1 X2 Y2*. The coordinates *X1 Y1* specify the top, left corner of the rectangle, and the coordinates *X2 Y2* specify the width and height.

Example

The following command loads the video capture file C:\VCAP\VCAPFILE.TGA into the video buffer:

```
load vboard c:\vcap\vcapfile.tga notify
```

The "vboard" device sends a notification message when the loading is completed.

load (Video Overlay)

Command	Arguments
load *device_id*	**[***file_name***]** **[at** *buffer_rectangle***]** **[notify]** **[wait]**

open (Video Overlay)

Syntax

open *device* [*parameters*] [**notify**] [**wait**]

The **open** command initializes the video overlay device.

Parameters

You can specify one or more of the following optional items for *parameters*:

alias *device_alias*
Specifies an alternate name for the device element. If specified, it must be used as the *device_id* in subsequent commands.

parent *hwnd*
Specifies the window handle of the parent window.

shareable
Initializes the device element as shareable. Subsequent attempts to open it fail unless you specify **shareable** in both the original and subsequent **open** commands. MCI returns an error if the device is already open and not shareable.

style *style_type*
Indicates a window style.

style child
Opens a window with a child window style.

style overlapped
Opens a window with an overlapped window style.

style popup
Opens a window with a popup window style.

type *device_type*
Specifies the device type of the device element. MCI reserves overlay for the video overlay device type. As an alternative to **type**, MCI can use the [mci extension] entries in the SYSTEM.INI file to select the controlling device based on the extension used by the device element. You can also use the *device_name*!*element_name* abbreviation described in Combining the Device Name and Element Name.

Example

The following command opens a capture device with device name "digitalvideo" and assigns alias "vboard":

```
open digitalvideo alias vboard
```

open (Video Overlay)

Command	Arguments
open *device*	**[alias** *device_alias***]** **[parent** *hwnd***]** **[shareable]** **[style** *style_type* **| style child** **| style overlapped** **| style popup]** **[type** *device_type***]** **[notify]** **[wait]**

put (Video Overlay)

Syntax

put *device_id parameters* [**notify**] [**wait**]

The **put** command defines one or more of the following rectangles:

* The video rectangle defines the region of the incoming video image to capture.
* The frame rectangle defines the region of the frame buffer that receives the incoming video image.
* The source rectangle defines which region of the frame buffer is copied to the destination rectangle.
* The destination rectangle defines the region of the display window client area that receives the video image.

The video rectangle is related to the frame rectangle in the same way the source rectangle is related to the destination rectangle. Stretching can occur from the video rectangle to the frame rectangle, and from the source rectangle to the destination rectangle. Not all devices support stretching, and stretching must be enabled (using the **set** command).

Parameters

Specify one or more of the following items for *parameter*:

video
Selects the entire incoming video image to capture in the frame buffer.

video at *rectangle*
Selects a portion of the incoming video image to capture in the frame buffer. The *rectangle* coordinates are relative to the video origin and are specified as *X1 Y1 X2 Y2*. The coordinates *X1 Y1* specify the top, left corner of the rectangle, and the coordinates *X2 Y2* specify the width and height.

frame
Selects the entire frame buffer to receive the incoming video images.

frame at *rectangle*
Selects a portion of the frame buffer to receive the incoming video images. The *rectangle* coordinates are relative to the video buffer origin and are specified as *X1 Y1 X2 Y2*. The coordinates *X1 Y1* specify the top, left corner of the rectangle, and the coordinates *X2 Y2* specify the width and height.

source
Selects the entire video buffer to display in the destination window.

source at *rectangle*
Selects a portion of the video buffer to display in the destination window. The *rectangle* coordinates are relative to the video buffer origin and are specified as *X1 Y1 X2 Y2*. The coordinates *X1 Y1* specify the top, left corner of the rectangle, and the coordinates *X2 Y2* specify the width and height.

destination
Selects the entire client area of the destination window to display the video data from the frame buffer.

destination at *rectangle*
Selects a portion of the client area of the destination window to display the video data from the

frame buffer. The *rectangle* coordinates are relative to the window origin and are specified as *X1 Y1 X2 Y2*. The coordinates *X1 Y1* specify the top, left corner of the rectangle, and the coordinates *X2 Y2* specify the width and height.

Example

The following command defines three regions for the video, frame, and source:

```
put vboard video 120 120 200 200 frame 0 0 200 200 source 0 0 200 200
```

The regions are defined as follows:

* A 200- by 200-pixel region of the incoming video data, starting at an origin 120 pixels from the top left corner, will be captured to the frame buffer.
* The video data will be placed in a 200- by 200-pixel region at the top left corner of the frame buffer.
* Transfers are made from the 200- by 200-pixel region at the top left corner of the frame buffer to the destination window.

put (Video Overlay)

Command	Arguments
put *device_id*	**{destination** | **at** *destination_rectangle* | **frame** | **frame at** *frame_rectangle* | **source** | **source at** *source_rectangle* | **video** | **video at** *video_rectangle***}** **[notify]** **[wait]**

save (Video Overlay)

Syntax

save *device_id filename* [**at** *rectangle*] [**notify**] [**wait**]

The **save** command saves the contents of the video buffer to path name *filename*.

Parameters

You can specify the following optional item:

at *rectangle*

Specifies a rectangle relative to the video buffer origin. The *rectangle* is specified as *X1 Y1 X2 Y2*. The coordinates *X1*, *Y1* specify the top, left corner of the rectangle, and the coordinates *X2*, *Y2* specify the width and height.

Example

The following command saves the entire video buffer to filename C:\VCAP\VCAPFILE.TGA:

```
save vboard c:\vcap\vcapfile.tga
```

save (Video Overlay)

Command	Arguments
save *device_id*	*file_name* **[at** *buffer_rectangle***]** **[notify]** **[wait]**

set (Video Overlay)

Syntax

set *device_id parameters*

The set command establishes control settings for the driver.

Parameters

Specify one or more of the following items for *parameters*:

audio all off
Disables audio output. Video overlay devices don't support this option.

audio all on
Enables audio output. Video overlay devices don't support this option.

audio left off
Disables output to the left audio channel. Video overlay devices don't support this option.

audio left on
Enables output to the left audio channel. Video overlay devices don't support this option.

audio right off
Disables output to the right audio channel. Video overlay devices don't support this option.

audio right on
Enables output to the right audio channel. Video overlay devices don't support this option.

time format milliseconds
Video overlay devices don't support this option.

video off
Disables video output.

video on
Enables video output.

Example

The following command enables video output on the "vboard" device:

```
set vboard video on
```

set (Video Overlay)

Command	Arguments
set *device_id*	**[audio all off** **| audio all on** **| audio left off** **| audio left on** **| audio right off** **| audio right on** **| video off** **| video on]** **[time format milliseconds | time format ms]** **[notify]** **[wait]**

status (Video Overlay)

Syntax

status *device_id parameter* [**notify**] [**wait**]

The **status** command gets status information for the device.

Parameters

Specify one of the following items for *parameter*:

media present
Returns **true**.

mode
Returns **not ready**, **recording**, or **stopped** for the current mode.

ready
Returns **true** if the video overlay device is ready.

stretch
Returns **true** if stretching is enabled.

window handle
Returns the handle of the window used for the video overlay display in the low word of the return value.

Example

The following command returns true if stretching is enabled on the "vboard" device:

```
status vboard stretch
```

status (Video Overlay)

Command	Arguments
status *device_id*	**{ media present** **| mode** **| ready** **| stretch** **| window handle}** **[notify]** **[wait]**

unfreeze (Video Overlay)

Syntax

unfreeze *device_id* [*parameter*] [**notify**] [**wait**]

The **unfreeze** command enables the frame buffer to acquire video data. The command is supported by devices that support the **freeze** command.

Parameters

You can specify the following optional item for *parameter*:

at *rectangle*

Specifies a rectangle relative to the video buffer origin. The *rectangle* is specified as *X1 Y1 X2 Y2*. The coordinates *X1 Y1* specify the top, left corner of the rectangle, and the coordinates *X2 Y2* specify the width and height.

Example

The following command unfreezes a region of the video buffer:

```
unfreeze vboard at 10 20 90 165
```

unfreeze (Video Overlay)

Command	Arguments
unfreeze *device_id*	**[at** *freeze_rectangle***]**
	[notify]
	[wait]

where (Video Overlay)

Syntax

where *device_id parameter* [**notify**] [**wait**]

The **where** command gets the rectangle specifying the video, frame, source, or destination area.

Parameters

Specify one of the following items for *parameter*:

video
: Requests the offset and extent of the video rectangle. The video rectangle defines the region of the incoming video data that is transferred to the frame buffer.

frame
: Requests the offset and extent of the frame buffer rectangle. The frame buffer rectangle defines the area of the frame buffer that receives incoming video data.

source
: Requests the offset and extent of the source rectangle. The source rectangle defines the region of the frame buffer that is displayed in the destination window.

destination
: Requests the offset and extent of the destination rectangle. The destination rectangle defines the area of the display window client area that displays the image data from the frame buffer.

Example

The following command returns the region coordinates for the source rectangle:

```
where vboard source
```

where (Video Overlay)

Command	Arguments
where *device_id*	**{destination | frame | source | video}** **[notify]** **[wait]**

window (Video Overlay)

Syntax

window *device_id parameters* [**notify**] [**wait**]

The **window** command controls the destination window. The destination window is the window in which the image is displayed. You can change the display characteristics of the window or provide a destination window for the driver to use in place of the default destination window.

Generally, video overlay devices create a window when opened but do not display the window until they receive a **play** command. If your application provides a window to the driver, your application is responsible for managing the messages sent to the window.

The window command provides several flags that let you manipulate the window. Since you can use the **status** command to get the handle to the destination window, you can also use the standard window manager functions (like **ShowWindow**) to manipulate the window.

Parameters

Specify one or more of the following items for *parameters*:

fixed
Disables stretching of the image.

handle *window_handle*
Specifies the handle of a window to use instead of the default destination window.

handle default
Specifies that the video overlay device should create and manage its own destination window. This flag can be used to set the display back to the driver's default window.

state hide
Hides the destination window.

state iconic
Displays the destination window as an icon.

state maximized
Maximizes the destination window.

state minimize
Minimizes the destination window and activates the top-level window in the window-manager's list.

state minimized
Minimizes the destination window.

state no action
Displays the destination window in its current state. The window currently active remains active.

state no activate
Displays the destination window in its most recent size and state. The currently active window remains active.

state normal
Displays the destination window as it was created.

state show
Shows the destination window.

stretch
Enables stretching of the image.

text *caption*
Specifies the *caption* for the destination window.

Example

The following command enables stretching in the destination window of the "vboard" dvice:

```
window vboard stretch
```

window (Video Overlay)

Command	Arguments
window *device_id*	**[fixed | stretch]** **[handle** *window_handle* **| handle default]** **[state hide** **| state iconic** **| state maximized** **| state minimize** **| state minimized** **| state no action** **| state no activate** **| state normal** **| state show]** **[text** *caption_text***]** **[notify]** **[wait]**

capability (Waveform Audio)

Syntax

capability *device_id parameter* [**notify**] [**wait**]

The **capability** command requests information about the capabilities of the waveform audio driver.

Parameters

Specify one of the following items for *parameter*:

can eject
Returns **false**. Waveform audio devices have no media to eject.

can play
Returns **true** if the device can play. The device returns **true** if an output device is available.

can record
Returns **true** if the device can record.

can save
Returns **true** if the device can save data.

compound device
Returns **true**; waveform audio devices are compound devices.

device type
Returns **waveaudio**.

has audio
Returns **true**.

has video
Returns **false**. Waveform audio devices don't support video.

inputs
Returns the total number of input devices.

outputs
Returns the total number of output devices.

uses files
Returns **true**. Waveform audio devices use files for operation.

Example

The following command returns the number of waveform output devices:

```
capability mysound outputs
```

capability (Waveform Audio)

Command	Arguments
capability *device_id*	**{can eject** **| can play** **| can record** **| can save** **| compound device** **| device type** **| has audio** **| has video** **| inputs** **| outputs** **| uses files}** **[notify]** **[wait]**

close (Waveform Audio)

Syntax

close *device_id* [**notify**] [**wait**]

The **close** command closes the device element and any associated resources.

Example

The following command closes the "mysound" device:

```
close mysound
```

close (Waveform Audio)

Command	Arguments
close *device_id*	**[notify]** **[wait]**

cue (Waveform Audio)

Syntax

cue *device_id* [*parameter*] [**notify**] [**wait**]

The **cue** command prepares for playing or recording. Although you don't have to issue the **cue** command before playing or recording, on some devices it might reduce the delay before the device starts playing or recording.

This command fails if playing or recording is in progress.

Parameters

You can specify one of the following optional items for *parameter*:

input
: Prepares for recording.

output
: Prepares for playing. This is the default.

Example

The following command prepares the "mysound" device for recording:

```
cue mysound input
```

cue (Waveform Audio)

Command	Arguments
cue *device_id*	**[input | output]** **[notify]** **[wait]**

delete (Waveform Audio)

Syntax

delete *device_id* [*parameters*] [**notify**] [**wait**]

The **delete** command deletes a data segment from the MCI element.

Parameters

You can specify one or both of the following optional items for *parameters*:

from *position*
Specifies a starting position for the deletion. If the **from** parameter is not specified, the deletion begins at the current position.

to *position*
Specifies an ending position for the deletion. If the **to** parameter is not specified, the device deletes all data from the starting position through the end of the media.

Comments

Before issuing any commands that use position values, you should set the desired time format using the **set** command.

Example

The following command deletes the waveform data from one millisecond through 900 milliseconds (assuming the time format is set to milliseconds):

```
delete mysound from 1 to 900
```

delete (Waveform Audio)

Command	Arguments
delete *device_id*	**[from** *position***]** **[to** *position***]** **[notify]** **[wait]**

info (Waveform Audio)

Syntax

info *device_id parameter* [**notify**] [**wait**]

The **info** command gets textual information from the device.

Parameters

Specify one of the following items for *parameter*:

input
Returns the description of the current waveform audio input device. Returns **none** if an input device is not set. The MCIWAVE driver returns **Wave Audio Input and Output Device.**

file
Returns the current filename.

output
Returns the description of the current waveform audio output device. Returns **none** if an output device is not set. The MCIWAVE driver returns **Wave Audio Input and Output Device.**

product
Returns the description of the current waveform audio output device. The MCIWAVE driver returns **Wave Audio Input and Output Device.**

Example

The following command returns the filename of the waveform file associated with the "mysound" device:

```
info mysound file
```

info (Waveform Audio)

Command	Arguments
info *device_id*	**[file** **| input** **| output** **| product]** **[notify]** **[wait]**

open (Waveform Audio)

Syntax

open *device* [*parameters*] [**notify**] [**wait**]

The **open** command initializes the device.

Parameters

You can specify one or more of the following optional items for *parameters*:

alias *device_alias*
Specifies an alternate name for the given device. If specified, it must be used as the *device_id* in subsequent commands.

buffer *buffer_size*
Sets the size in seconds of the buffer used by the waveform audio device. The default size of the buffer is set when the waveform audio device is installed or configured. Typically the buffer size is set to 4 seconds. With the MCIWAVE device, the minimum size is 2 seconds and the maximum size is 9 seconds.

shareable
Initializes the device element as shareable. Subsequent attempts to open it fail unless you specify **shareable** in both the original and subsequent **open** commands. MCI returns an error if it the device is already open and not shareable. The MCIWAVE device does not support shared files.

type *device_type*
Specifies the device type of a device element. MCI reserves waveaudio for the waveform audio device type. As an alternative to **type**, MCI can use the [mci extension] entries in the SYSTEM.INI file to select the controlling device based on the extension used by the device element. You can also use the *device_name*!*element_name* abbreviation described in Combining the Device Name and Element Name.

Comments

The MCIWAVE device included with Windows requires an asynchronous waveform driver. It does not work with synchronous drivers like the PC Speaker driver.

Example

The following command opens the "mysound" device:

```
open new type waveaudio alias mysound buffer 6
```

With device name "new", the waveform driver prepares a new waveform resource. The command assigns device alias "mysound" and specifies a 6-second buffer.

open (Waveform Audio)

Command	Arguments
open *device*	**[alias** *device_alias***]** **[buffer** *buffer_size***]** **[shareable]** **[type** *device_type***]** **[notify]** **[wait]**

pause (Waveform Audio)

Syntax

pause *device_id*

The **pause** command pauses playing or recording.

Example

The following command pauses playback or recording in the "mysound" device:

```
pause mysound
```

pause (Waveform Audio)

Command	Arguments
pause *device_id*	**[notify]** **[wait]**

play (Waveform Audio)

Syntax

play *device_id* [*parameters*] [**notify**] [**wait**]

The **play** command starts playing audio.

Parameters

You can specify one or more of the following optional items for *parameters*:

from *position*
: Specifies the starting position for the playback. If the **from** parameter is not specified, playback begins at the current position.

to *position*
: Specifies the ending position for the playback. If the **to** parameter is not specified, play stops at the end of the media.

Comments

Before issuing any commands that use position values, you should set the desired time format using the **set** command.

Example

The following command plays the "mysound" device from the beginning:

```
play mysound from 1
```

play (Waveform Audio)

Command	Arguments
play *device_id*	**[from** *position***]** **[to** *position***]** **[notify]** **[wait]**

record (Waveform Audio)

Syntax

record *device_id* [*parameters*] [**notify**] [**wait**]

The **record** command starts recording audio. All data recorded after a file is opened is discarded if the file is closed without saving it.

Parameters

You can specify one or more of the following optional items for *parameters*:

insert
Specifies that new data is added to the device element.

from *position*
Specifies a starting position for the recording. If the **from** parameter is not specified, the device starts recording at the current position.

to *position*
Specifies an ending position for the recording. If the **to** parameter is not specified, the device records until it receives a **stop** or **pause** command.

overwrite
Specifies that new data will replace data in the device element. The MCIWAVE driver does not support this option.

Comments

Before issuing any commands that use position values, you should set the desired time format using the **set** command.

Example

The following command starts recording at the current position:

```
record mysound
```

record (Waveform Audio)

Command	Arguments
record *device_id*	**[from** *position***]** **[to** *position***]** **[insert** | **overwrite]** **[notify]** **[wait]**

resume (Waveform Audio)

Syntax

resume *device_id* [**notify**] [**wait**]

The **resume** command continues playing or recording of a paused device.

Example

The following command continues playback or recording on the "mysound" device:

```
resume mysound
```

resume (Waveform Audio)

Command	Arguments
resume *device_id*	**[notify]** **[wait]**

save (Waveform Audio)

Syntax

save *device_id* [*filename*] [**notify**] [**wait**]

The **save** command saves the MCI element in its current format.

Parameters

You can specify the following optional item:

filename
: Specifies the file and pathname used to save data.

Comments

The *filename* parameter is required if the device was opened using the **new** device ID.

Example

The following command saves the waveform data recorded into the "mysound" device:

```
save mysound c:\sounds\mysound.wav
```

save (Waveform Audio)

Command	Arguments
save *device_id*	**[***file_name***]** **[notify]** **[wait]**

seek (Waveform Audio)

Syntax

seek *device_id parameter* [**notify**] [**wait**]

The **seek** command moves to the specified position and stops.

Parameters

Specify one of the following items for *parameter*:

to *position*
: Specifies the stop position.

to start
: Seeks to the start of the first sample.

to end
: Seeks to the end of the last sample.

Comments

Before issuing any commands that use position values, you should set the desired time format using the **set** command.

Example

The following command seeks to the end of the waveform data in "mysound":

```
seek mysound to end
```

seek (Waveform Audio)

Command	Arguments
seek *device_id*	**[to** *position* **| to start | to end]** **[notify]** **[wait]**

set (Waveform Audio)

Syntax

set *device_id parameters* [**notify**] [**wait**]

The set command specifies various settings for the driver.

Parameters

Specify one or more of the following items for *parameters*:

alignment *integer*
Sets the alignment of data blocks relative to the start of waveform data passed to the waveform audio device. The file is saved in this format.

any input
Use any input that supports the current format when recording. This is the default.

any output
Use any output that supports the current format when playing. This is the default.

audio all off
Disables audio output. The MCIWAVE sequencer does not support this option.

audio all on
Enables audio output. The MCIWAVE sequencer does not support this option.

audio left off
Disables output to the left audio channel. The MCIWAVE sequencer does not support this option.

audio left on
Enables output to the left audio channel. The MCIWAVE sequencer does not support this option.

audio right off
Disables output to the right audio channel. The MCIWAVE sequencer does not support this option.

audio right on
Enables output to the right audio channel. The MCIWAVE sequencer does not support this option.

bitspersample *bit_count*
Sets the number of bits per sample played or recorded. The file is saved in this format.

bytespersec *byte_rate*
Sets the average number of bytes per second played or recorded. The file is saved in this format.

channels *channel_count*
Sets the channels for playing and recording. The file is saved in this format.

format tag *tag*
Sets the format type for playing and recording. The file is saved in this format.

format tag pcm
Sets the format type to PCM for playing and recording. The file is saved in this format.

input *integer*
Sets the audio channel used as the input.

output *integer*
Sets the audio channel used as the output.

samplespersec *integer*
Sets the sample rate for playing and recording. The file is saved in this format.

time format bytes
Sets the time format to bytes. All position information is specified as bytes following this command.

time format milliseconds
Sets the time format to milliseconds. All commands that use position values will assume milliseconds. You can abbreviate milliseconds as **ms**.

time format samples
Sets the time format to samples. All position information is specified as samples following this command.

Example

The following command sets the time format to milliseconds and sets the waveform audio format to 8 bit, mono, 11 kHz:

```
set mysound time format ms bitspersample 8 channels 1 samplespersec 11025
```

set (Waveform Audio)

Command	Arguments
set *device_id*	**[alignment** *block_alignment***]** **[any input]** **[any output]** **[audio all off** **| audio all on** **| audio left off** **| audio left on** **| audio right off** **| audio right on** **| video off** **| video on]** **[bitspersample** *bit_count***]** **[bytespersec** *byte_rate***]** **[channels** *channel_count***]** **[format tag** *tag* **| format tag pcm]** **[input** *device_number***]** **[output** *device_number***]** **[samplespersec** *sample_rate***]** **[time format milliseconds** **| time format ms** **| time format bytes** **| time format samples]** **[notify]** **[wait]**

status (Waveform Audio)

Syntax

status *device_id parameter* [**notify**] [**wait**]

The **status** command gets status information for the device.

Parameters

Specify one of the following items for *parameter*:

alignment
Returns the block alignment of data in bytes.

bitspersample
Returns the bits per sample.

bytespersec
Returns the average number of bytes per second played or recorded.

channels
Returns the number of channels set (1 for mono, 2 for stereo).

current track
Returns the index for the current track. The MCIWAVE device returns 1.

format tag
Returns the format tag.

input
Returns the input set. If one is not set, the error returned indicates that any device can be used.

length
Returns the total length of the waveform.

length track *track_number*
Returns the length of the specified track.

level
Returns the current audio sample value.

media present
Returns **true**.

mode
Returns **not ready, paused**, **playing**, **stopped**, **recording**, or **seeking** for the device mode.

number of tracks
Returns the number of tracks. The MCIWAVE device returns 1.

output
Returns the currently set output. If no output is set, the error returned indicates that any device can be used.

position
Returns the current position.

position track *track_number*
Returns the position of the track specified by *track_number*. The MCIWAVE device returns 0.

ready
Returns **true** if the device is ready.

samplespersec
Returns the number of samples per second played or recorded.

start position
Returns the starting position of the media.

time format
Returns the current time format.

Comments

Before issuing any commands that use position values, you should set the desired time format using the **set** command.

Example

The following command returns the length of the waveform data in milliseconds (assuming the time format is set to milliseconds):

```
status mysound length
```

status (Waveform Audio)

Command	Arguments
status *device_id*	**{alignment** **| bitspersample** **| bytespersec** **| channels** **| current track** **| format tag** **| input** **| length** **| length track** *track_number* **| level** **| media present** **| mode** **| number of tracks** **| output** **| position** **| position track** *track_number* **| ready** **| samplespersec** **| start position** **| time format}** **[notify]** **[wait]**

stop (Waveform Audio)

Syntax

stop *device_id* [**notify**] [**wait**]

The **stop** command stops playing or recording.

Example

The following command stops playback or recording in the "mysound" device:

```
stop mysound
```

stop (Waveform Audio)

Command	Arguments
stop *device_id*	**[notify]** **[wait]**