Multimedia ToolBook Online Encyclopedia
Version 1.0

August 14, 1991

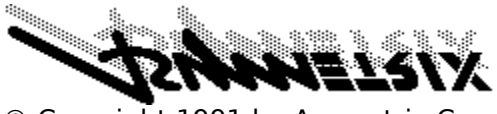
Multimedia ToolBook Encyclopedia

Topics:

MCI Device Errors

Syntax symbols

clearAllGlobalVars()
getGlobalVar()
globalVarCount()
MSFfromMillisec()
millisecFromHMS()
millisecFromMSF()
millisecFromSMPTE()
SMPTEfromMillisec()
setGlobalVar()
tbkBitmap()
tbkBitmapChk()
tbkBmpErrorString()
tbkMCI()
tbkMCIchk()
tbkMMDevices()
tbkMMErrorString()
tbkMMNotify
tbkMMTimer
tbkMMVersion()
tbkTimerCapability()
tbkTimerStart()
tbkTimerStop()



Syntax Symbols

Quick Reference

Symbols used in syntax statements

Symbol	Meaning
[|]	Optional words and parameters. Items enclosed in square brackets are not required in a statement, but they can be used to make a statement clearer or to add options. When two or more items can be used interchangeably, they are separated by a vertical bar.
< >	Parameters. The angle brackets in a statement and the words they enclose are replaced with a literal value or an expression that yields a value.
( )	Required parentheses. Items shown inside parentheses in a statement must be enclosed in parentheses when you type the statement into a script or the Command window.

clearAllGlobalVars()

TBKBMP.DLL

Syntax

```
clearAllGlobalVars()
```

Declaration

```
INT clearAllGlobalVars()
```

Description

Clears all global variables.

Returns

```
0
```

Example

```
get clearAllGlobalVars()
```

getGlobalVar() TBKBMP.DLL

Syntax

```
getGlobalVar(<varName>)
```

Declaration

```
STRING getGlobalVar(STRING)
```

Description

Gets the value of a global variable, a list of global variable names, or the name of a variable at a specific location in the variable table.

Parameter

`<varName>` The global variable name, or a reference to the variable's location in the variable table in the format `"#<n>"` where `n` is 1 through 64 or null.

Returns

If `<varName>` is the name of a variable, returns the value of the variable. If `<varName>` is a number indicating the location of a variable, returns the variable name. If `<varName>` is not specified, returns a list of all global variable names.

Examples

```
set gVarList to getGlobalVar(null)
while gVarList <>null
   pop gVarList into gVar
   request "Global variable:"&& gVar && "Value:" && \
      getGlobalVar(gVar)
end while
get getGlobalVar("#1")  -- Assuming status is first
                        -- item in the list of global
                        -- variables, puts "status"
                        -- into It
```

globalVarCount() TBKBMP.DLL

Syntax

```
globalVarCount()
```

Declaration

```
INT globalVarCount()
```

Description

Gets the number of global variables.

Returns

The number of global variables.

Example

```
if globalVarCount() > 0
   request "Clear all global variables?" with \
      "&Yes" or "&No"
   if it contains "Yes"
      get clearAllGlobalVars()
   end if
end if
```

HMSfromMillisec() TBKMM.DLL

Syntax

```
HMSfromMillisec(<milliseconds>)
```

Declaration

```
STRING HMSfromMillisec (DWORD)
```

Description

Converts a value in milliseconds into a time string in the format `hh:mm:ss`, where `hh` is hours, `mm` is minutes, and `ss` is seconds.

Parameter

`<milliseconds>`	A variable containing a value in milliseconds.

Returns

The corresponding formatted string.

Example

```
--Increments a position counter (in milliseconds) and
--displays the resulting value in a field using the
--hh:mm:ss format. (A ButtonDown handler sets the
--device timer format to milliseconds and specifies a
--starting value for pos. A ButtonUp handler plays
--from the new position and resets the device time
--format to hh:mm:ss.)
to handle buttonStillDown
   system pos

   --Increment the position by one second then
   --display the new position in hh:mm:ss format
   increment pos by 1000
   set text of field "position" to HMSfromMillisec(pos)
end
```

millisecFromHMS()

TBKMM.DLL

Syntax

```
millisecFromHMS(<HMS format>)
```

Declaration

```
LONG millisecFromHMS(STRING)
```

Description

Converts a time string in the format `hh:mm:ss` to milliseconds, where `hh` is hours, `mm` is minutes, and `ss` is seconds. The value for `ss` can be omitted if it is 0, and the value for `mm` can be omitted if both `mm` and `ss` are 0.

Parameter

`<HMS format>` A string defining the time format.

Returns

The number of milliseconds. If the string is invalid or null, returns `-1`.

Example

```
--Updates a position slider on the current page.
to handle idle
   system discLength    --Play length of the disc in
                        --milliseconds
   get tbkMCI("status videodisc position","")
   --Convert position to numeric value for arithmetic
   --operation
   set pos to millisecFromHMS(it)
   set sliderPosition of group "position slider" to \
      pos / discLength
end
```

millisecFromMSF()

TBKMM.DLL

Syntax

```
millisecFromMSF(<MSF format>)
```

Declaration

```
LONG millisecFromMSF (STRING)
```

Description

Converts a time string in the format `mm:ss:ff` to milliseconds, where `mm` is minutes, `ss` is seconds, and `ff` is CD Redbook-Audio frames (75 frames/second).

Parameter

`<MSF format>` A string defining the time format.

Returns

The number of milliseconds. If the string is invalid or null, returns `-1`.

Example

```
--Updates a position slider on the current page.
to handle idle
   system cdLength      --Play length of the disc in milliseconds
   get tbkMCI("status cdAudio position","")
   --Convert position to numeric value for arithmetic
   --operation
   set pos to millisecFromMSF(it)
   set sliderPosition of group "position slider" to \
      pos / cdLength
end
```

millisecFromSMPTE()

TBKMM.DLL

Syntax

```
millisecFromSMPTE(<SMPTE format>, <frames>)
```

Declaration

```
LONG millisecFromSMPTE(STRING, WORD)
```

Description

Converts a time string in the format `hh:mm:ss:ff` to milliseconds, where `hh` is hours, `mm` is minutes, `ss` is seconds, and `ff` is SMPTE audio frames.

Parameters

`<SMPTE format>` A string defining the time format.

`<frames>` Frames per second. If this is 0, the `ff` portion of the string is ignored. The number of frames depends on the file division type for the current MIDI file. Common values are 30 frames/second, 25 frames/second or 24 frames/second. The value for `ff` can be omitted if it is 0, and the value for `ss` can be omitted if `ss` and `ff` are 0.

Returns

The number of milliseconds. If the string is invalid or null, returns `-1`.

Example

```
--Updates a position slider on the current page.
to handle idle
   system midiLength       --Length of a MIDI file in milliseconds
   system fps        --Number of SMPTE frames per second for the
   file
   get tbkMCI("status midiFile position","")
   --Convert position to numeric value for arithmetic operation
   set pos to millisecFromSMPTE(it,fps)
   set sliderPosition of group "position slider" to pos /
   midiLength
end
```

MSFfromMillisec() TBKMM.DLL

Syntax

```
MSFfromMillisec(<milliseconds>)
```

Declaration

```
STRING MSFfromMillisec (DWORD)
```

Description

Converts milliseconds into a time string in the format `mm:ss:ff`, where `mm` is minutes, `ss` is seconds, and `ff` is CD Redbook-Audio frames (75 frames per second).

Parameter

`<milliseconds>`	A variable containing a value in milliseconds.

Returns

The corresponding formatted string.

Example

```
--Scans a CD audio disc and displays the position in mm:ss:ff
   format. (ButtonDown
--handler sets timer in ms format to facilitate scan, and
   specifies a starting
--position for pos. ButtonUp handler plays from new position then
   resets time format. )
to handle buttonStillDown
   system pos
   increment pos by 1000
   --Update a field in MSF format to keep track of scan position
   set text of field "position" to MSFfromMillisec(pos)
end
```

setGlobalVar() TBKBMP.DLL

Syntax

```
setGlobalVar(<varName>,<value>)
```

Declaration

```
STRING setGlobalVar(STRING,STRING)
```

Description

Creates and sets a new variable, or sets an existing variable to a new value.

Parameters

`<varName>` The name of the variable to set. The first character must be a letter, and the name cannot contain spaces or special characters except the underscore character.

`<value>` The value to set the variable to.

Returns

A string that contains `0` if an existing variable was set to null, `1` if a new variable was created and set to a value, and `2` if an existing variable was set to a new value. Returns null and sets the value of `sysErrorNumber` if the `<varName>` parameter is invalid or if the value cannot be set.

Example

```
get setGlobalVar ("status", "open") -- Stores "open" in a global
   variable
get setGlobalVar("status", null)    -- Clears global variable
   "status"
```

SMPTEfromMillisec() TBKMM.DLL

Syntax

```
SMPTEfromMillisec(<milliseconds>,<frames>)
```

Declaration

```
STRING SMPTEfromMillisec (DWORD, WORD)
```

Description

Converts milliseconds to a time string in the format `hh:mm:ss:ff`, where `hh` is hours, `mm` is minutes, `ss` is seconds, and `ff` is SMPTE audio frames.

Parameters

`<milliseconds>` A variable representing milliseconds.

`<frames>` A variable defining frames per second. The number of frames depends on the file division type for the current MIDI file. Common values are 30 frames per second, 25 frames per second, or 24 frames per second. The value for `ff` can be omitted if it is 0, and the value for `ss` can be omitted if `ss` and `ff` are 0.

Returns

The corresponding formatted string.

Example

```
--Scans a MIDI file and displays the position in hh:mm:ss:ff
   format. (ButtonDown handler
--sets timer to ms format to facilitate scan and specifies a
   starting position for pos. ButtonUp
--handler plays from new position then resets time format. )
to handle buttonStillDown
   system pos
   system fps     ---Number of SMPTE frames per second for the
   MIDI file
   increment pos by 1000
   --Update a field in HMS format to keep track of scan position
   set text of field "position" to SMPTEfromMillisec(pos,fps)
end
```

tbkBitmap()

TBKBMP.DLL

Syntax

```
tbkBitmap("<command> <device> [<command arguments>]")
```

Declaration

```
STRING tbkBitmap(STRING)
```

Description

Controls the display of a bitmap file in a child, popup, or overlapped window. This function works similar to way the `tbkMCI()` function works when it is used to control animation devices. For general information about controlling the display of bitmaps, see "Controlling Bitmaps from OpenScript" in Chapter 2, "Controlling Devices from OpenScript" of Using Multimedia ToolBook.

Each command used with this function has a separate entry in chapter; see that entry for details. For example, there is an entry for `tbkBitmap(open...)`. The `<command>` parameter can be one of these commands:

capability	info	play	window
close	open	status	

tbkBitmap("capability ...") TBKBMP.DLL

Syntax

```
tbkBitmap("capability <device> [<command argument>]")
```

Declaration

```
STRING tbkBitmap(STRING)
```

Description

Requests information about the capability of the bitmap player.

Parameters

`<device>` Any non-null string. The bitmap player does not have to be opened to get capability information, so the value of this parameter works as a placeholder.

`<command argument>` One of the following:

- `compound device` Returns `true`.
- `device type` Returns `bitmap`.
- `has files` Returns `true`; a bitmap player is an element of a compound device.
- `has video` Returns `true`.
- `static` Returns `true`, which means that the device does not support frame control as in `play from | play to`, `stop`, `pause`, or `step`.
- `uses palettes` Returns `true`.
- `<all others>` Returns `false`.

Example

```
get tbkBitmap("capability pic1 has video")
```

tbkBitmap("close ...") TBKBMP.DLL

Syntax

```
tbkBitmap("close <device> | all ")
```

Declaration

```
STRING tbkBitmap(STRING)
```

Description

Closes a bitmap.

Parameter

`<device>` | `all` The file name or alias for the bitmap file and the window in which it is to be displayed, as specified by the `open` command. If `all` is specified, all bitmaps and associated windows are closed. The rules for specifying the `<device>` parameter are the same as those for specifying an MCI device. For details, see "Opening and Using MCI Devices" in Chapter 2, "Controlling Devices from OpenScript" of Using Multimedia ToolBook.

Example

```
get tbkBitmap("close c:\tbkmm\earth.dib")
```

tbkBitmap("info ...") TBKBMP.DLL

Syntax

```
tbkBitmap("info <device> [<command argument>]")
```

Declaration

```
STRING tbkBitmap(STRING)
```

Description

Provides general information about the bitmap file and the window in which it is displayed.

Parameters

`<device>` The file name or alias for the bitmap file and the window in which it is displayed, as specified by the `open` command. The rules for specifying the device are the same as those for specifying an MCI device. For details, see "Opening and Using MCI Devices" in Chapter 2, "Controlling Devices from OpenScript" of Using Multimedia ToolBook.

`<command argument>` One of the following:

- `product` Returns "Asymetrix Bitmap Displayer"
- `text` Returns the window caption.
- `file` Returns the filename and path of the bitmap.

Example

```
-- Opens a bitmap file
get tbkBitmap("open c:\tbkmm\earth.dib alias pic1")
get tbkBitmap("info pic1 text")
request it
```

tbkBitmap("open ...") TBKBMP.DLL

Syntax

```
tbkBitmap("open <device> [<command arguments>]")
```

Declaration

```
STRING tbkBitmap(STRING)
```

Description

Initializes the bitmap displayer and returns the number of currently open bitmaps. After opening a bitmap window, the author should close it with `tbkBitmap(close...)`.

Parameters

`<device>` The file name or alias for the bitmap file and the window in which it will be displayed. The rules for specifying the device are the same as those for specifying an MCI device. For details, see "Opening and Using MCI Devices" in Chapter 2, "Controlling Devices from OpenScript" of Using Multimedia ToolBook.

Once you specify an alias, you must use it as the <device> parameter in all subsequent tbkBitmap() statements that control the bitmap. Aliases are frequently used as shorthand for long path names and compound device references.

`<command arguments>` None, one, or both of the following:

- `style [overlapped | popup | child | <style number>]` Specifies the style of the window in which to display the bitmap. The default is `overlapped`.

  If you specify `<style number>`, the value is passed to the Windows `CreateWindow` function. To determine the style number, add together the Windows constant integer for the basic window style (overlapped, popup, or child) and the integers for other aspects of style (such as removing the Minimize or Maximize box). For a partial list of styles and their corresponding integers, see "Controlling Bitmaps from OpenScript" in Chapter 2, "Controlling Devices from OpenScript" of Using Multimedia ToolBook.

  Windows are opened at a size equal to the size of the bitmap. The default position for popup and overlapped windows is the upper left corner of the screen. The default position for child windows is the upper left corner of the parent window. You can adjust the size and position of a window before displaying it.

- `parent <window handle>` Specifies the handle of the parent for the bitmap window. The default is the ToolBook main window (`sysWindowHandle`) for popup and child windows, and `null` for overlapped windows.

Example

```
get tbkBitmap("open c:\tbkmm\earth.dib alias earth style popup")
```

tbkBitmap("play ...") TBKBMP.DLL

Syntax

```
tbkBitmap("play <device>")
```

Declaration

```
STRING tbkBitmap(STRING)
```

Description

Displays the specified bitmap.

Parameter

`<device>` The file name or alias for the bitmap file and the window in which it is displayed, as specified by the `open` command. The rules for specifying the device are the same as those for specifying an MCI device. For details, see "Opening and Using MCI Devices" in Chapter 2, "Controlling Devices from OpenScript" of Using Multimedia ToolBook.

Example

```
to handle buttonUp
   get tbkBitmap("play earth")
end
```

tbkBitmap("status ...") TBKBMP.DLL

Syntax

```
tbkBitmap("status <device> [<command arguments>]")
```

Declaration

```
STRING tbkBitmap(STRING)
```

Description

Obtains status information from the device.

Parameters

`<device>` The file name or alias for the bitmap file and the window in which it is to be displayed, as specified by the `open` command. The rules for specifying the `<device>` parameter are the same as those for specifying an MCI device. For details, see "Opening and Using MCI Devices" in Chapter 2, "Controlling Devices from OpenScript" of Using Multimedia ToolBook.

`<command arguments>` One of the following:

- `window` Returns the window handle of an open bitmap.
- `position` Returns a list of two items that evaluates to the upper left corner of the window. The first item is the horizontal dimension and the second item is the vertical dimension. For popup and overlapped windows, this value is relative to the upper left corner of the screen. For child windows, this value is relative to the upper left corner of the parent window.
- `extent` Returns a list of two items indicating the dimensions of the bitmap in pixels. The first item indicates the horizontal dimension and the second item indicates the vertical dimension.
- `size` Returns a list of two items that evaluates to the size of the window in pixels. The first item is the horizontal dimension and the second item is the vertical dimension.
- `visible` Returns the window's visible status as `true` or `false`.
- `style` Returns the basic window style as `overlapped`, `popup`, or `child`.
- `palette` Returns the palette handle.

Example

```
-- Displays a Request box showing the bitmap window handle
to handle buttonUp
   request tbkBitmap("status c:\tbkmm\earth.dib window")
end
```

tbkBitmap("window ...") TBKBMP.DLL

Syntax

```
tbkBitmap("window <device> [<command argument>]")
```

Declaration

```
STRING tbkBitmap(STRING)
```

Description

Sets various options for the window in which a bitmap is displayed.

Parameters

`<device>` The file name or alias for the bitmap file and the window in which it is to be displayed, as specified by the `open` command. The rules for specifying the device are the same as those for specifying an MCI device. For details, see "Opening and Using MCI Devices" in Chapter 2, "Controlling Devices from OpenScript" of Using Multimedia ToolBook.

`<command argument>` One of the following:

`text <caption text>` | `default` Sets the window caption, which is meaningful for overlapped windows. The keyword `default` sets the caption to the file name.

`state <window state constant>` Controls the visibility, maximized and iconized state of the window. The valid window state constants are:

- `hide` Hides the window and passes activation to another window.
- `minimize` Minimizes the window and activates the top-level window.
- `minimized` Activates the window and displays it as an icon.
- `show` Activates the window and displays it in its current size and position.
- `maximize` Activates the window and displays it maximized.
- `iconic` Displays the window as an icon; the active window remains active.
- `asis` Displays the window in its current state; the active window remains active.
- `aswas` Displays the window with most recent size and position; the active window remains active.
- `normal` Activates and displays a window; if the window is minimized or maximized, restores the window to its original size and position.

The preceding keywords are equivalent to the constants passed to the Windows function `ShowWindow`. The `tbkBitmap("window...")` function can be used to show the window in various ways, unlike `tbkBitmap("play...")`, which shows the window only in its `normal` window state. For example, to keep ToolBook as the active window:

```
get tbkBitmap("window earth state asis")
```

`position <x, y>` Sets the position of the window in screen coordinates. For popup and overlapped windows, `x` is the horizontal distance and `y` is the vertical distance from the upper left corner of the screen. For child windows, `x` is the horizontal distance and `y` is the vertical distance from the upper left corner of the parent window.

`size <x,y>` Sets the size of the window in screen coordinates, where `x` is the horizontal dimension, and `y` is the vertical dimension.

Example

```
to handle buttonUp
   get tbkBitmap("window earth size 200,200")
   get tbkBitmap("window earth position 150,50")
end
```

tbkBitmapChk()

TBKMM.SBK

Syntax

```
tbkBitmap("<command> <device> [<command arguments>]" ,<message>,
   <break>)
```

Declaration

```
STRING tbkBitmap(STRING)
```

Description

Like the `tbkBitmap()` function, controls the display of a bitmap in a child, popup, or overlapped window. In addition, `tbkBitmapChk()` provides options for error checking. You can specify whether or not to display an error message when an error occurs and whether or not to break to the system when an error occurs.

For general information about controlling the display of bitmaps, see "Controlling Bitmaps from OpenScript" in Chapter 2, "Controlling Devices from OpenScript" of Using Multimedia ToolBook.

The `tbkBitmapChk()` function can use the same commands as a parameter as the `tbkBitmap()` function. For details see:

`tbkBitmap (capability...)`
`tbkBitmap (close...)`
`tbkBitmap (info...)`
`tbkBitmap (open...)`
`tbkBitmap (play...)`
`tbkBitmap (status...)`
`tbkBitmap (window...)`

Parameters

Same as for `tbkBitmap()`, with the addition of the following parameters:

`<message>` If non-null, displays an error message when an error occurs; if null, does not display a message.

`<break>` If non-null, breaks to the system when an error occurs; if null, continues execution.

Example

```
to handle buttonUp
   --Break to system and request error message if open
   --fails
   get tbkBitmapChk("open c:\dibs\picture.dib alias pict",1,1)
   --Request error message if any of these fail, but
   --don't break execution
   get tbkBitmapChk("window pict position 100,200",1)
   get tbkBitmapChk("window pict state show",1)
end
```

tbkBmpErrorString()

TBKBMP.DLL

Syntax

```
tbkBmpErrorString( <sysErrorNumber> )
```

Declaration

```
STRING tbkBmpErrorString( WORD )
```

Description

Returns the error message associated the value of `sysErrorNumber`. If TBKMM.SBK and TBKMM.DLL are in your path, you can use either this function or the `tbkMMErrorString()` function to determine errors. If TBKMM.SBK and TBKMM.DLL are not in your path, use this function to determine which error occurred.

Parameter

`<sysErrorNumber>` The error number returned by `tbkBitmap()`.

Returns

12001	The device independent bitmap file could not be found.
12002	There was an error reading that bitmap. It may be a corrupt file.
12003	The bitmap cannot be initialized in memory. It may contain a corrupted color table.
12004	The bitmap display window could not be created.
12005	The bitmap could not be drawn.
12006	That bitmap could not be found. The alias may not be valid.
12007	An internal error has occurred attempting to register a window class.
12008	That variable name is invalid.
12009	The value for the global variable could not be set.
12010	No variable with that name.

tbkMCI() TBKMM.DLL

Syntax

```
tbkMCI("<MCI command> <device> <command arguments> [wait]",
   <object identifier>)
```

Declaration

```
STRING tbkMCI(STRING,STRING)
```

Description

Provides OpenScript access to devices that are compatible with the MCI component of Windows with Multimedia. For general information about using MCI devices see "Opening and Using MCI Devices" in Chapter 2, "Controlling Devices from OpenScript" in Using Multimedia ToolBook.

Devices and commands currently supported are:

Animation (capability, close, info, open, pause, play, put, realize, resume, seek, set, status, step, stop, update, where, and window).

CD Audio (capability, close, info, open, pause, play, seek, resume, status, and stop)

MIDI Sequencer (capability, close, info, open, pause, play, record, resume, save, seek, set, status, and stop).

Video overlay (capability, close, freeze, info, load, open, put, save, resume, set, status, unfreeze, where, and window).

Videodisc player (capability, close, escape, info, open, pause, play, resume, seek, set, spin, status, step, and stop).

Waveform audio (capability, close, cue, info, open, pause, play, record, resume, save, seek, set, status, and stop).

Parameters

<MCI command> One of the following MCI commands, or, for some devices, additional commands are available:

- break Disables or enables control to be returned from the driver to the application when a specific break key is pressed.
- capability Requests information about the capability of the device.
- close Closes the device.
- info Requests information from a device driver.
- open Initializes the device. Include a shareable flag if you want other tasks and applications to concurrently use the device. Include alias<alias> if you want to use an alias name for the device with subsequent commands.
- pause Pauses playing.

- `play` Starts playing. If `from` and `to` positions are specified, plays from and to those positions. Otherwise, plays from the current position until the device receives a `pause` or `stop` command or until the end of the media or file.
- `seek` Seeks the specified location in the file. If the device is currently playing or recording, this function continues playing or recording at the specified location.
- `set` Sets various options.
- `sound` Plays the specified sound.
- `status` Requests status information from the device.
- `stop` Stops playing.
- `sysinfo` Requests MCI system information.

`<device>` The device to be controlled by the command, identified by the device name, device element name, device alias, or other identifier. If you specify an alias for the device when you open the device, you must use the alias in subsequent `tbkMCI()` statements that control the device.

`<command arguments>` Parameters and flags related to <`MCI command`>. Options vary depending on the device and command. You can use the following data types for the parameters:

- Strings, delimited by leading and trailing spaces. Quotation marks are automatically removed from the string. If you want to embed quotation marks in a string, use double quotation marks (""""). If you want to use an empty string, use two quotation marks ("") for the string.
- Signed long integers, delimited by a leading and trailing space. Unless otherwise specified, integers can be positive or negative. If an integer is negative, do not leave a space between the negative sign and the first digit.
- Rectangle, as an ordered list of four signed integers. Spaces delimits this data type and separates each integer in the list.

`wait` If `wait` is specified, the `tbkMCI()` function does not return a value until the specified operation is complete. Otherwise, `tbkMCI()` returns a value immediately.

`<object identifier>` An expression that evaluates to the unique name of the object to be notified when the requested operation is completed. This parameter sends the `tbkMMNotify` message to the object. The value of this parameter can be null or two quotation marks ("") to indicate an empty string; in this case, no notification is sent. For details, see "Handling Object Notification, Return Strings, and Errors" in Chapter 2, "Controlling Devices from OpenScript." See also the entry for `tbkMMNotify` in this chapter.

Note: Don't include the `notify` flag in a `tbkMCI()` statement. The `<object identifier>` parameter takes care of notification.

Example

```
--Plays a specified track on a CD
to handle playTrack tracknum
   get tbkMCI("open cdAudio wait","")
   set endPos to tbkMCI("status cdAudio length track" &&
   trackNum,"")
   get tbkMCI("set cdAudio time format tmsf","")
   --Play track and notify this page when the play is finished so
   --the device can be closed
   get tbkMCI("play cdAudio from" && trackNum & ":00:00:00 to " &&
   \
     trackNum & ":" & endpos, uniquename of this page)
   get tbkMCI("set cdAudio time format msf","")
end
```

tbkMCIchk()

TBKMM.SBK

Syntax

```
tbkMCIchk("<MCI command><device><arguments>[wait]",<object
   identifier>,<message>,<break>)
```

Description

Like the `tbkMCI()` function, `tbkMCIchk()` provides OpenScript access to devices that are compatible with the MCI component of Windows with Multimedia. In addition, `tbkMCIchk()` provides options for error checking. You can specify whether or not to display an error message when an error occurs, and specify whether or not to break to the system when an error occurs.

Devices and commands currently supported are:

Animation (capability, close, info, open, pause, play, put, realize, resume, seek, set, status, step, stop, update, where, and window).

CD Audio (capability, close, info, open, pause, play, seek, resume, status, and stop)

MIDI Sequencer (capability, close, info, open, pause, play, record, resume, save, seek, set, status, and stop).

Video overlay (capability, close, freeze, info, load, open, put, save, resume, set, status, unfreeze, where, and window).

Videodisc player (capability, close, escape, info, open, pause, play, resume, seek, set, spin, status, step, and stop).

Waveform audio (capability, close, cue, info, open, pause, play, record, resume, save, seek, set, status, and stop).

Parameters

`<MCI command>` One of the following MCI commands, or, for some devices, additional commands are available:

- `break` Disables or enables control to be returned from the driver to the application when a specific break key is pressed.
- `capability` Requests information about the capability of the device.
- `close` Closes the device.
- `info` Requests information from a device driver.
- `open` Initializes the device. Include a `shareable` flag if you want other tasks and applications to concurrently use the device. Include `alias<alias>` if you want to use an alias name for the device with subsequent commands.
- `pause` Pauses playing.
- `play` Starts playing. If `from` and `to` positions are specified, plays from and to those positions. Otherwise, plays from the current position until the device receives a `pause` or `stop` command or until the end of the media or

file.

- `seek` Seeks the specified location in the file. If the device is currently playing or recording, this function continues playing or recording at the specified location.
- `set` Sets various options.
- `sound` Plays the specified sound.
- `status` Requests status information from the device.
- `stop` Stops playing.
- `sysinfo` Requests MCI system information.

`<device>` The device to be controlled by the command, identified by the device name, device element name, device alias, or other identifier. If you specify an alias for the device when you open the device, you must use the alias in subsequent `tbkMCI()` statements that control the device.

`<command arguments>` Parameters and flags related to <`MCI command`>. Options vary depending on the device and command. You can use the following data types for the parameters:

- Strings, delimited by leading and trailing spaces. Quotation marks are automatically removed from the string. If you want to embed quotation marks in a string, use double quotation marks (""""). If you want to use an empty string, use two quotation marks ("") for the string.
- Signed long integers, delimited by a leading and trailing space. Unless otherwise specified, integers can be positive or negative. If an integer is negative, do not leave a space between the negative sign and the first digit.
- Rectangle, as an ordered list of four signed integers. Spaces delimits this data type and separates each integer in the list.

`wait` If `wait` is specified, the `tbkMCI()` function does not return a value until the specified operation is complete. Otherwise, `tbkMCI()` returns a value immediately.

`<object identifier>` An expression that evaluates to the unique name of the object to be notified when the requested operation is completed. This parameter sends the `tbkMMNotify` message to the object. The value of this parameter can be null or two quotation marks ("") to indicate an empty string; in this case, no notification is sent. For details, see "Handling Object Notification, Return Strings, and Errors" in Chapter 2, "Controlling Devices from OpenScript." See also the entry for `tbkMMNotify` in this chapter.

Note: Don't include the `notify` flag in a `tbkMCI()` statement. The `<object identifier>` parameter takes care of notification.

`<message>` If non-null, displays an error message when an error occurs; if null, does not display the message.

`<break>` If non-null, breaks to the system when an error occurs; if null, continues execution.

Example

```
--Opens a wave audio file and then plays and closes it
--if opened successfully
to handle buttonUp
   --Request error and break to system if open fails
   get tbkMCIchk("open c:\waves\noise.wav alias waveFile","",1,1)
   --Only request error if play or close fails
   get tbkMCIchk("play wavefile from 0 wait","",1)
   get tbkMCIchk("close waveFile","",1)
end
```

tbkMMDevices() TBKMM.SBK

Syntax

```
tbkMMDevices()
```

Description

Specifies which devices were successfully loaded from the `[devices]` section of TBKMM.INI. (However, because a device is listed in TBKMM.INI does not guarantee that the physical or logical device is available or working.)

Returns

A list of devices.

Example

```
if "bitmap" is not in tbkmmdevices()
   request "Bitmap display support has not been installed."
end
```

tbkMMErrorString() TBKMM.DLL

Syntax

```
tbkMMErrorString (<sysErrorNumber>)
```

Declaration

```
STRING tbkMMErrorString (WORD)
```

Description

Returns the error message associated with the error number in `sysErrorNumber` after the `tbkMCI()` or the `tbkBitmap()` function is executed. For errors that occur when the `tbkBitmap()` function is executed, you can also use the `tbkBmpErrorString()` function to determine errors.

Parameter

`<sysErrorNumber>` The number returned by `tbkBitmap()` or `tbkMCI()`, which is the value of the `sysErrorNumber` property after the function is executed.

Returns

A error message associated with an error number. For a list of error numbers and related messages, see "MCI System Commands" in Appendix A, "MCI Commands."

Example

```
to get tbkMCIchk cmd,notif,brk
   local retVal
   set sysErrorNumber to 0
   set retVal to tbkmci(cmd,notif)
   if sysErrorNumber<>0
      request tbkMMErrorString(sysErrorNumber)
      if brk <> null and brk is true
         set sysCursor to 1
         break to system
      end
   end
   return retVal
end
to get tbkBitmapChk cmd, brk
   local retVal
   set sysErrorNumber to 0
   set retVal to tbkBitmap(cmd)
   if sysErrorNumber<>0
      request tbkMMErrorString(sysErrorNumber)
      if brk<> null and brk is true
         set sysCursor to 1
         break to system
      end
   end
   return retVal
end
```

tbkMMNotify

Notification message

Syntax

```
tbkMMNotify <status>, <operation>, <device>
```

Description

Sent to the object specified by the `<object identifier>` parameter of the `tbkMCI()` or `tbkMCIchk()` function after ToolBook finishes executing the function. For details, see "Handling Object Notification, Return Strings, and Errors" in Chapter 2, "Controlling Devices from OpenScript."

Parameters

`<status>` Evaluates to one of the following:

- `successful` Indicates that the MCI command was executed without interruption and that the conditions for initiating a callback have been satisfied.
- `superseded` Indicates that the notification message sent by one statement has been superseded by the notification message sent by another statement. The callback conditions are reset to correspond to the notification message sent as a result of executing the most recently executed statement.
- `aborted` Cancels a notification that is pending and prevents the callback conditions set by a previously executed statement. For example, the `stop` command cancels a notification pending for the `play to 500` command.
- `failure` Indicates that a device error occurred while a device was executing the MCI command. For example, this message is sent when a hardware error occurs while a device attempts to execute a `play` command.

`<operation>` The name of the MCI command passed with the `tbkmci()` function. For example, if this notification message is sent as a result of the statement `get tbkMCI("open..."),<operation>` evaluates to `open`.

`<device>` Evaluates to the device type.

Example

```
--Closes a CD audio device if a play command was successful
to handle tbkMMNotify status, operation, device
   if status && operation && device = "successful play cdAudio"
      get tbkMCI("close cdaudio","")
   end
end
```

tbkMMTimer

Notification message

Syntax

```
tbkMMTimer <timer ID>
```

Description

Sent to the object specified by the `<object identifier>` parameter of the `tbkTimerStart()` function when ToolBook finishes executing this function. For details, see "Handling Object Notification, Return Strings, and Errors" in Chapter 2, "Controlling Devices from OpenScript."

Parameter

`<timer ID>` A positive number that identifies the timer.

Example

```
--Updates status information when a periodic timer
--event message is received
to handle tbkMMTimer timerID
   system statusTimer   --The ID number of the periodic
                        --status timer
   if timerID = statusTimer
      set text of field "CD position" to \
         tbkMCI("status cdaudio position","")
      ...(Other status information)
   else
      --Invalid timer sent message, so close it
      get tbkTimerStop(timerID)
   end
end
```

tbkMMVersion() TBKMM.DLL

Syntax

```
tbkMMVersion()
```

Declaration

```
STRING tbkMMVersion()
```

Description

Specifies copyright information, version number, and version date for TBKMM.DLL.

Returns

A string containing version information.

Example

```
-- Display the current version in a Request box
request tbkMMVersion()
```

tbkTimerCapability() TBKMM.DLL

Syntax

```
tbkTimerCapability()
```

Declaration

```
STRING tbkTimerCapability( )
```

Description

Requests information about the capability of the timer device.

Returns

A list of two items in milliseconds. The first item is the minimum resolution of the timer device; the second item is its maximum period.

Example

```
--Start the longest timer possible
to handle buttonUp
   get tbkTimerCapability()
   set delay to item 2 of it
   get tbkTimerStart("single",delay,1000,self)
end
```

tbkTimerStart() TBKMM.DLL

Syntax

```
tbkTimerStart("single" | "periodic", <delay>, <resolution>,
   <object identifier>)
```

Declaration

```
STRING tbkTimerStart(STRING,WORD,WORD,STRING)
```

Description

Provides OpenScript access to the timer services supported by Windows with Multimedia.

Parameters

`"single" | "periodic"` Sets the timer type to `single` for a one-time notification when the timer expires or to `periodic` for periodic notification each time the timer expires. If a timer type is `periodic`, you must explicitly stop the timer with `tbkTimerStop()`.

`<delay>` Sets the timer period to the specified number of milliseconds, after which the timer expires.

`<resolution>` Sets the timer resolution to the specified number of milliseconds, which determines the accuracy with which the timer attempts to execute the timer callback function.

`<object identifier>` The unique name of the object to be notified when the timer expires. This parameter sends the `tbkMMTimer` message to the object. Specifying a `null` value causes an error. For details, see "Handling Object Notification, Return Strings, and Errors" in Chapter 2, "Controlling Devices from OpenScript"; see also the entry for `tbkMMTimer` in this chapter.

Returns

The timer ID, if no error occurs. If an error occurs, returns null and sets `sysErrorNumber`.

Example

```
to handle buttonUp
   if myTimerID of self <> null
      set myTimerID of self to
   tbkTimerStart("periodic",5000,1000,self)
   ... (Statements here to specify other actions related to
   buttonUp)
end
```

tbkTimerStop() TBKMM.DLL

Syntax

```
tbkTimerStop(<timerID>)
```

Declaration

```
STRING tbkTimerStop(WORD)
```

Description

Stops a timer started with `tbkTimerStart()`.

Parameter

`<timer ID>` Must be the value returned by the `tbkTimerStart()` function to stop the timer. The value can also be `0` to indicate that all timers should be stopped.

Returns

`0` to indicate success. If an error occurs, returns null and sets `sysErrorNumber`.

Example

```
to handle rightButtonUp
   get tbkTimerStop (myTimerID of self)
   clear myTimerID of self
end
```

The device to be controlled by the command, identified by the device name, device element name, device alias, or other identifier. If you specify an alias for the device when you open the device, you must use the alias in subsequent `tbkMCI()` statements that control the device.

The MCIMMP.DRV device driver supports playing of animation movies created in MacroMind Director.

The MCICDA.DRV device driver supports control of CD Audio devices via MSCDEX version 2.2.

The MCIPIONR.DRV device driver supports control of Pioneer LD-V4200 laserdisc players.

The MCISEQ.DRV device driver supports control of MIDI sequencer files stored in the generic .MID format.

The MCIWAVE.DRV device driver supports control of digital audio files stored in linear PCM format.

MCI device errors

The following table list errors returned in the special variable `sysErrorNumber`:

Value	Error
10011	This error code is not within the supported range.
10012	The initialization file TBKMM.INI could not be found.
10013	The [mmerrors] section in TBKMM.INI does not contain a string for that error code.
10014	No string for that error code was found.
10015	That function has been disabled in this release of the ToolBook Multimedia Extensions.
10016	An internal tbkmm data structure could not be found.
10017	An internal tbkmm window could not be created.
10018	A required word in the command string could not be found.
10019	The notification information for the current operation could not be found.
10020	The notification information for the current timer could not be found.
10021	The value requested with this command is not within the legal range.
10022	Attempting to notify the ToolBook object has failed.
10023	That command is not available for this device.
10024	This device does not support that capability.
10025	This device does not support that attribute. No value can be set for it.
10026	That information is not available for this device.
10027	The device is not open. Open the device and try the command again.
10028	That command argument is not valid in this context.
10030	The timer services are currently disabled.
10031	More than one error occurred while loading devices. Possible errors are: support drivers were not found, or had invalid names.
10032	The name supplied is not valid.
10033	That device is not currently available or installed.
10034	There is no device of that type with that alias.
10035	That alias is already in use.
10036	The device could not be opened.
10037	The value of that attribute could not be retrieved.
10038	No ToolBook object was specified for the notification message.
10101	A memory block could not be locked. Windows maybe having a serious problem.
10102	A memory block could not be locked. Windows maybe having a serious problem.
10103	An invalid argument was given to a memory management routine.
10104	The system has run out of local memory.
10105	A local memory block could not be freed.
10106	An invalid local memory handle was passed to the memory management system.
10107	The system has run out of global memory. Close down other applications and try again.

10108	A global memory block could not be freed.
10109	An invalid global memory handle was passed to the memory management system.
10201	The variable length table system has run out of memory.
10202	The variable length table manager could not free a memory block.
10203	An attempt was made to access a nonexistent element in a variable length table.
10204	An invalid handle was passed to the variable length table manager.
10205	The variable length table manager has detected a corrupted table.
10206	Internal error: full iteration.
10301	The current instance could not be found.
10302	The is no instance table for the active instance.
10303	This Dynamic Link Library can only be used with ToolBook.
11000	The function was successful.
11001	Unspecified error.
11002	Device id is out of range.
11003	The driver failed to enable.
11004	The device has already been allocated.
11005	The given device handle is invalid.
11006	No driver present.
11007	Error in allocating memory.
11008	This function is not supported.
11009	The given error number is out of range.
11032	This format is not supported by the hardware.
11033	There is still data playing.
11034	The WAVEHDR was not prepared before writing.
11064	The MIDIHDR was not prepared before writing.
11065	There is still data playing.
11066	There is no current MIDI map.
11067	The port is busy outputting data.
11068	Current setup contains nonexistent device(s).
11257	Invalid device ID.
11259	Unknown command parameter.
11261	Unknown command.
11262	Hardware error on media device.
11263	The device is not open or is not known.
11264	Not enough memory for requested operation.
11265	The device name is in use by this task. Use a unique alias.
11266	Error loading media device driver.
11267	No command was specified.
11268	The output string was not long enough.
11269	A string value was missing from the command.
11270	Invalid or missing integer in command.
11271	Internal parser error.
11272	Internal driver error.
11273	Required parameter is missing.
11274	Action not available for this device.
11275	Requested file not found.
11276	Device not ready.
11277	Internal error.
11278	Unspecified device error.
11279	The device name ‘all’ is not allowed for this command.

11280	Errors occurred in more than one device.
11281	Cannot deduce a device type from the given extension.
11282	Parameter value out of range.
11283	The device was opened but cannot create alias.
11284	Incompatible flags were specified.
11285	The ‘nounload’ flag is only valid for a device element.
11286	The file was not saved.
11287	The device name must be a valid device type.
11288	The device is locked until it is closed automatically.
11289	The specified alias is an open device in this task.
11290	Unknown value for parameter.
11291	The device is already open, use the ‘shareable’ flag with each ‘open’.
11292	No device name was specified.
11293	Illegal value for time format.
11294	A closing quotation mark is missing.
11295	A flag or value was specified twice.
11296	Invalid file format.
11297	Parameter block pointer was NULL.
11298	Attempt to save unnamed file.
11299	An alias must be used with the ‘new’ device name.
11300	The "notify’ flag is illegal with auto-open.
11301	An element name cannot be used with this device.
11320	No compatible waveform playback device is free.
11321	Set waveform playback device is in use.
11322	No compatible waveform recording device is free.
11323	Set waveform recording device is in use.
11324	Any compatible waveform playback device may be used.
11325	Any compatible waveform recording device may be used.
11326	No compatible waveform playback devices.
11327	Set waveform playback device is incompatible with set format.
11328	No compatible waveform recording devices.
11329	Set waveform recording device is incompatible with set format.
11336	Set Song Pointer incompatible with SMPTE files.
11337	Specified port is in use.
11338	Specified port does not exist.
11339	Current map uses nonexistent device(s).
11340	Miscellaneous error with specified port.
11341	Timer error.
11342	No current MIDI port.
11346	There is no display window.
11347	Could not create or use window.
11348	A read from the file failed.
11349	A write to the file failed.
12001	The device independent bitmap file could not be found.
12002	There was an error reading that bitmap. It may be a corrupt file.
12003	The bitmap cannot be initialized in memory. It may contain a corrupted color table.
12004	The bitmap display window could not be created.
12005	The bitmap could not be drawn.
12006	That bitmap could not be found. The alias may not be valid.
12007	An internal error has occurred attempting to register a

	window class.
12008	That variable name is invalid.
12009	The value for the global variable could not be set.
12010	No variable with that name.

break <device> <argument> MCI system command

Description

Specifies the key that aborts the `wait` notification.

Argument

One of the following:

`on <virtual key>` Specifies the `<virtual key>` that aborts the `wait`. When `<virtual 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 break key set.

sound <device>

MCI system command

Description

Plays a specified sound. The `<device>` parameter specifies a sound from the `[sound]` section of WIN.INI. If it is not found, MCI uses the SystemDefault sound.

sysinfo <device> <argument> MCI system command

Description

Obtains MCI system information.

Argument

One of the following:

`installname` Returns the name from SYSTEM.INI that is used to install the device.

`quantity` Returns the number of MCI devices listed in SYSTEM.INI file of the type specified by the device name. The device name must be a standard MCI device type. Any digits after the name are ignored. The special device name `all` returns the total number of MCI devices in the system.

`quantity open` Returns the number of MCI devices listed in SYSTEM.INI file that are open of the type specified by the device name. The device name must be a standard MCI device type. Any digits after the name are ignored. The special device name `all` returns the total number of MCI devices in the system that are open.

`name <index>` Returns the name of an MCI device. The `<index>` ranges from 1 to the number of devices of that type. If `all` is specified for the device name, `<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 `<index>` ranges from 1 to the number of devices of that type. It `all` is specified for the device name, `<index>` ranges from 1 to the total number of devices in the system.

See the table of MCI device errors for values of sysErrorNumber returned that are common to all commands.

Animation commands

The animation commands provide a common method for displaying animation sequences in the Windows environment. The MCI commands for displaying animation sequences are: capability, close, info, open, pause, play, put, realize, resume, seek, set, status, step, stop, update, where, and window.

capability <device> <argument> Animation command

Description

Requests information about the capabilities of the graphics driver.

Argument

`can eject` Returns `true` if the device can eject the media. The MCIMMP device returns `false`.

`can play` Returns `true` if the device can play. The MCIMMP device returns `true`.

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

`can reverse` Returns `true` if the animation 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 display rectangle. The MCIMMP device returns `false`.

`compound device` Returns `true` if the device requires an element name.

`device type` Returns `animation`.

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

`has audio` Returns `true` if the device supports audio playback. The MCIMMP device returns `true`.

`has video` Returns `true`. Animation devices are video devices.

`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 path name.

`uses palettes` Returns `true` if the animation device uses palettes. The MCIMMP device returns `true`.

`windows` Returns the number of windows the device can support. The MCIMMP device returns `8`.

close <device>

Animation command

Description

Closes an animation player element and any resources associated with it. When the last element is closed, MCI also closes the device.

info <device> <argument>

Animation command

Description

Provides general information about a device.

Argument

One of the following:

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

`product` Returns the product name and model of the current animation device. The MCIMMP driver returns `Microsoft Multimedia Movie Player`.

`window text` Returns the caption of the window used by the animation device.

load <device> <filename>

Animation command

Description

Loads a device element from disk.

Arguments

`<filename>` specifies the path and file name of the device element to be loaded.

open <device> [<arguments>] Animation command

Description

Initializes the animation player.

Arguments

None to all of the following:

`alias <device alias>` Specifies an alternate name for the animation player element. If specified, the alias must also be used for subsequent references.

`nostatic` Indicates that the device should reduce the number of static (system) colors in the palette, which increases the number of colors controlled by the animation. The MCIMMP device driver reduces the static colors to black and white while in the foreground.

`parent <window handle>` Specifies the window handle of the parent window. The default is the value of `sysWindowHandle`.

`shareable` Initializes an animation player 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 MCIMMP device driver does not support shared files.

`style <child` | `overlapped` | `popup>` Indicates a window style.

`type <device type>` Specifies the compound device used to control a device element. As an alternative to `type`, MCI can use the `[`MCI `extensions]` entries in the SYSTEM.INI file to select the controlling device based on the extension used by the device element.

pause <device>

Animation command

Description

Pauses playing.

play <device> [<arguments>] Animation command

Description

Starts playing the animation player.

Arguments

None to all of the following:

`fast` Plays the animation sequence at a fast rate.

`from <pos>` Specifies the frame at which to start playing. If `from <pos>` is omitted, playing starts at the current frame.

`to <pos>` Specifies the frame at which to stop playing. If `to <pos>` is omitted, playing stops at the end of the frame.

`reverse` Indicates that the play direction is backwards.

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

`slow` Plays the animation sequence at a slow rate.

`speed <fps>` Plays animation at the specified speed in frames per second.

put <device> <argument> Animation command

Description

Defines the source image or destination window.

Argument

One of the following:

`destination` Sets the whole window as the destination window.

`destination at <rect>` Specifies a rectangle array defining a clipping rectangle relative to the window origin. The rectangle array `<rect>` is specified as `x1 y1 z2 y2`, where `x1 y1` specify the top left corner, and `x2 y2` specify the width and height of the rectangle. This argument does not specify the size of the MCI window used to display the image but rather the size of the rectangle within the window to use in displaying the image.

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

`source at <rect>` Specifies a rectangle array defining a clipping rectangle relative to the image origin. The `<rect>` is specified as `x1 y1 z2 y2` where `x1 y1` specify the top left corner, and `x2 y2` specify the width and height of the rectangle.

realize <device> <argument> Animation command

Description

Tells the animation device to select and realize its palette into a display context of the displayed window. The MCIMMP device driver does not support this command.

Argument

One of the following:

`background` Realizes the palette as a background palette.

`normal` Realizes the palette normally.

resume <device>

Animation command

Description

Resumes a paused animation. The MCIMMP device driver does not support this command.

seek <device> <argument>

Animation command

Description

Resumes a paused animation.

Argument

One of the following:

`to end` Moves to the end of the animation.

`to <pos>` Specifies the animation frame to seek.

`to start` Moves to the start of the animation.

set <device> <argument> Animation command

Description

Sets various control device attributes.

Argument

One of the following:

`audio all off` | `audio all on` Enables or disables audio output.

`audio left off` | `audio left on` Enables or disables output to the left audio channel.

`audio right off` | `audio right on` Enables or disables output to the right audio channel.

`time format frames` Sets time format to `frames`. All position information is specified in frames following this command. When the device is opened, `frames` is the default mode.

`time format milliseconds` Sets time format to `milliseconds`. All position information is this format after this command. You can abbreviate milliseconds as `ms`. The MCIMMP device driver does not support this argument.

`video off` | `video on` Enables or disables video output. The MCIMMP device driver does not support this argument.

status <device> [<argument>] Animation command

Description

Obtains status information for the device.

Argument

One of the following:

`current track` Returns the current track. The MCIMMP device driver returns `1`.

`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 #>` Returns the total number of frames in the track.

`media present` Returns `true` if the media is inserted in the device; otherwise, 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 in the low word of the return value.

`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 animation device is ready.

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

`start position` Returns the starting position of the device media or element.

`time format` Returns the time format.

`window handle` Returns the handle of the window used for the animation in the low word of the return value.

step <device> [<arguments>] Animation command

Description

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

Arguments

None, one, or both of the following:

`by <frames>` Indicates the number of frames to step.

`reverse` Step the frames in reverse.

stop <device>

Animation command

Description

Stops playing.

update <device> <argument> Animation command

Description

Repaints the current frame into the specified display context.

Argument

One of the following:

`at <rect>` Specifies the clipping rectangle.

`hdc <handle>` Specifies the handle of the display context to paint.

where <device> <argument>

Animation command

Description

Obtains the rectangle specifying the source or destination area.

Argument

One of the following:

`destination` Requests the destination offset and extent.

`source` Requests the source offset and extent.

window <device> <argument> Animation command

Description

Specifies a window to display the animation instead of the default window created by the driver. By default, animation players should create a window when opened but should not display it until the animation player receives the `play` command. Applications providing window handles should manage the display issues that result when the window is sized or when the window handle is switched during play.

Several flags manipulate the window. Since the `status` command can obtain the handle to the current display window, you can use the standard window functions instead.

Argument

One of the following:

`handle <window handle>` Specifies the window handle of the destination window used as an alternate to the default window.

`handle default` Specifies that the animation player should create and manage its own window. This flag can be used to set the display back to the driver's default window.

`state hide` Hides the current display window.

`state iconic` Displays the window as an icon.

`state maximize` Maximizes the current display window.

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

`state minimized` Minimizes the current window.

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

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

`state normal` Displays the current display window as it was created.

`state show` Shows the current display window.

`text <caption>` Specifies the caption for the display window.

CD audio (Redbook) commands

MCI command summary

The CD audio commands provide a common method for playing CD audio in the Windows environment. The CD audio device type includes the MCICDA.DRV device driver for CD-ROM. The CD audio commands are: capability, close, info, open, pause, play, seek, resume, status, and stop.

capability <device> <argument> CD audio command

Description

Requests information about the capabilities of a CD audio device.

Argument

One of the following:

`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`. MCI CD audio devices cannot record.

`can save` Returns `false`. MCI CD audio devices cannot save.

`compound device` Returns `false`. MCI CD audio devices are simple devices.

`device type` Returns `CDaudio`.

`has audio` Returns `true`.

`has video` Returns `false`. MCI CD audio devices do not support video.

`uses files` Returns `false`. MCI CD audio devices do not use files.

close <device>

CD audio command

Description

Closes the device. MCI unloads a device when it is no longer being used.

info <device> [product]

CD audio command

Description

Provides general information about a device.

Argument

`product` Returns the product name and model of the current audio device. The MCICDA.DRV device driver returns `CD Audio Player`.

open <device> [<arguments>] CD audio command

Description

Initializes the device.

Arguments

None, one, or both of the following:

`alias <device alias>` Specifies an alternate name for the given device. If specified, the alias must also be used for subsequent references.

`shareable` Initializes the device as shareable. Subsequent attempts to open the device fail unless you specify `shareable` in both the original and subsequent `open` commands. An error is returned if the device is already open and not shareable.

pause <device>

CD audio command

Description

Pauses playing. Same as `stop` for the MCICDA.DRV device driver.

play <device> [from <pos> to <pos>] CD audio command

Description

Starts playing audio.

Arguments

None, one, or both of the following:

`from <pos>` Specifies the position to start. If `from <pos>` is omitted, playing starts at the current position. If `<pos>` is greater than the end position of the disc or is greater than the `to` position, MCI returns an error.

`to <pos>` Specifies the position to stop playing. If `to <pos>` is omitted, playing stops at the end of the disk. If `<pos>` is greater than the length of disc, MCI plays to the end of the disc.

resume <device>

CD audio command

Description

Restarts a paused device.

seek <device> to <pos>

CD audio command

Description

Moves to the specified location on the disc. If already playing, the device continues at the new location.

Arguments

`to end` Moves to the end of the device.

`to <pos>` Specifies the position to seek. If the destination is greater than the disc length, MCI positions the device at the end of the disc. This argument is required.

`to start` Moves to the start of the device.

set <device> <argument>

CD audio command

Description

Sets various control device attributes.

Argument

One of the following:

`audio all off` | `audio all on` Enables or disables audio output.

`audio left off` | `audio left on` Enables or disables output to the left audio channel.

`audio right off` | `audio right on` Enables or disables 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 time format to milliseconds. All position information is in this format after this command. You can abbreviate milliseconds as `ms`.

`time format msf` Sets time format to `mm:ss:ff` where `mm` is minutes, `ss` is seconds, and `ff` is frames. When time and position values are used, they are expressed in this format. This is the default for CD audio. On input, `ff` can be omitted if it is 0, `ss` can be omitted if both it and `ff` are 0. The maximum values for these fields are `mm`=99, `ss`=59, and `ff`=74.

`time format tmsf` Sets time format to `tt:mm:ss:ff` where `tt` is tracks, `mm` is minutes, `ss` is seconds, and `ff` is frames. When time or position values are used, they are expressed in this format after this command. On input, `ff` can be omitted if it is 0, `ss` can be omitted if both it and `ff` are 0, and `mm` can be omitted if it, `ss`, and ff are 0. The maximum values for these fields are `tt`=99, `mm`=99, `ss`=59, and `ff`=74.

status <device> <argument>

CD audio command

Description

Obtains status information for the device.

Argument

One of the following:

`current track` Returns the current track.

`length` Returns the total length of the disc.

`length track <track #>` Returns the length of the specified track.

`media present` Returns `true` if a disc is in the device; otherwise, returns `false`.

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

`number of tracks` Returns the number of tracks.

`position` Returns the current position.

`position track <track #>` Returns the position of the start of the specified track.

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

`time format` Returns the time format.

stop <device>

CD audio command

Description

Stops playing.

MIDI sequencer commands

Thhe MCI commands that support MIDI sequencers are: capability, close, info, open, pause, play, record, resume, save, seek, set, status, and stop.

capability <device> <argument> MIDI sequencer command

Description

Requests information about the capabilities of the MIDI sequencer.

Argument

One of the following:

`can eject` Returns `false`. Sequencers cannot eject the media.

`can play` Returns `true` if the sequencer can play.

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

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

`compound device` Returns `true`. Sequencers are compound devices.

`device type` Returns `sequencer`.

`has audio` Returns `true`. Sequencers supports playback.

`has video` Returns `false`. Sequencers do not support video.

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

close <device>

MIDI sequencer command

Description

Closes the sequencer element and the port and file associated with it.

info <device> product

MIDI sequencer command

Description

Provides general information about a device.

Argument

`product` Returns the product name of the current MIDI sequencer. The MCISEQ.DRV sequencer returns `MIDI Sequencer`.

open <device> [<arguments>] MIDI sequencer command

Description

Initializes the sequencer.

Arguments

None to all of the following:

`alias <device alias>` Specifies an alternate name for the sequencer element. If specified, the alias must also be used for subsequent references.

`shareable` Initializes the sequencer element as shareable. Subsequent attempts to open the device 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. Files cannot be shared when using the MCISEQ.DRV sequencer.

`type <device type>` Specifies the sequencer device used to control a device element. As an alternative to `type`, MCI can use the `[`MCI `extensions]` entries in the SYSTEM.INI file to select the sequencer based on the extension used by the device element.

pause <device>

MIDI sequencer command

Description

Pauses playing.

play <device> [from <pos> to <pos>] MIDI sequencer command

Description

Starts playing the sequencer.

Arguments

None, one, or both of the following:

`from <pos>` Specifies the position at which to start playing. If `from <pos>` is omitted, playing starts at the current position.

`to <pos>` Specifies the position at which to stop playing. If `to <pos>` is omitted, playing stops at the end of the file.

record <device> [<arguments>] MIDI sequencer command

Description

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

Arguments

None to all of the following:

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

`from <pos>` Specifies the position to start recording. If `from <pos>` is omitted, the device starts recording at the current position.

`to <pos>` Specifies the position to stop recording. If `to <pos>` is omitted, the device records until a `stop` or `pause` command is received.

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

resume <device>

MIDI sequencer command

Description

Restarts a paused sequence.

save <device> <filename> MIDI sequencer command

Description

Saves an MCI element. The MCISEQ.DRV sequencer does not support this option.

Argument

`<filename>` Specifies the destination path and file.

seek <device> to <pos>

MIDI sequencer command

Description

Moves to the specified position in the file.

Argument

`to end` Moves to the end of the sequence.

`to <pos>` Specifies the position to seek.

`to start` Moves to the start of the sequence.

set <device> <argument> MIDI sequencer command

Description

Sets various device control attributes.

Argument

One of the following:

`audio all off` | `audio all on` Enables or disables audio output. The MCISEQ.DRV sequencer does not support this option.

`audio left off` | `audio left on` Enables or disables output to the left audio channel. The MCISEQ sequencer does not support this option.

`audio right off` | `audio right on` Enables or disables 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>` in the format `hh:mm:ss:ff`. The offset is the beginning time of a SMPTE-based sequence.

`port <integer>` Sets the MIDI port receiving the MIDI messages. This command fails if the port you are trying to open is being used by another application.

`port mapper` Sets the MIDI mapper as the port receiving 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 as the synchronization source. This is the default.

`slave midi` Sets the MIDI sequencer to use incoming MIDI data 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 <integer>` Sets the tempo of the sequence according to the current time format. For a ppqn-based file, the `<integer>` is interpreted as beats per

minute. For a SMPTE based file, the `<integer>` is interpreted as frames per second.

`time format milliseconds` Sets time format to `milliseconds`. All position information is specified as milliseconds following this command. The sequence file sets the default format to ppqn or SMPTE. You can abbreviate milliseconds as `ms`.

`time format song pointer` Sets time format to song pointer (sixteenth notes). This can only be performed for a sequence of division type ppqn.

`time format smpte 24` Sets time format to SMPTE 24-frame rate. All position information is specified in SMPTE format following this command. The sequence file sets the default format to ppqn or SMPTE.

`time format smpte 25` Sets time format to SMPTE 25-frame rate. All position information is specified in SMPTE format following this command. The sequence file sets the default format to ppqn or SMPTE.

`time format smpte 30` Sets time format to SMPTE 30-frame rate. All position information is specified in SMPTE format following this command. The sequence file sets the default format to ppqn or SMPTE.

`time format smpte 30 drop` Sets time format to SMPTE 30-drop-frame rate. All position information is specified in SMPTE format following this command. The sequence file sets the default format to ppqn or SMPTE.

status <device> <argument> MIDI sequencer command

Description

Obtains status information for the MIDI sequencer.

Argument

One of the following:

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

`division type` Returns one of the following file division types: `ppqn`, `smpte 24 frame`, `smpte 25 frame`, `smpte 30 frame`, or `smpte 30 drop frame`. Use this information to determine the format of the MIDI file and the meaning of `tempo` and `position`.

`length` Returns the length of a sequence in the current time format. For ppqn files, the length is in song pointer units. For SMPTE files, the format is `hh:mm:ss:ff`.

`length track <track #>` Returns the length of a track in the current time format. The ppqn files, the length track is in song pointer units; for SMPTE files, the format is `hh:mm:ss:ff`.

`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` for the device mode.

`number of tracks` Returns the number of tracks. The MCISEQ sequencer returns `1`.

`offset` Returns the offset of a SMPTE-based file. The time is in the format `hh:mm:ss:ff`. The offset is the beginning time of the SMPTE-based sequence.

`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, the position is in song pointer units. For SMPTE files, the format is `hh:mm:ss:ff`.

`position track <track #>` Returns the current position of the specified track in the current time format. For SMPTE files, the format is `hh:mm:ss:ff`. 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.

`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 time format.

stop <device>

MIDI sequencer command

Description

Stops playing.

Video overlay commands

The video overlay commands provide a common method for displaying animation, bitmaps, digital video, and video overlay in the Windows environment. These commands are: capability, close, freeze, info, open, put, resume, save, set, status, unfreeze, where, and window.

capability <device> <argument> Video overlay command

Description

Requests information about the capabilities of the video overlay device.

Argument

One of the following:

`can eject` Returns `false` if the video overlay device cannot eject the media.

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

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

`can record` Returns `false`. Video overlay devices cannot record.

`can save` Returns `true` if the device can save frames in frame buffer-specific format.

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

`compound device` Returns `true` if the device requires an element name.

`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 the element of the compound is a file path name.

`windows` Returns the number of windows the device can support.

close <device>

Video overlay command

Description

Closes a video overlay element and any resources associated with it. When the element is closed, MCI also closes the device.

freeze <device> at <rect>

Video overlay command

Description

Disables video acquisition of the frame buffer. This command is supported only if `capability can freeze` returns `true`.

Argument

`at <rect>` Specifies a rectangle array defining a rectangle relative to the video buffer origin. The rectangle array `<rect>` is specified as `x1 y1 x2 y2`, where `x1 y1` specify the top left corner, and `x2 y2` specify the width and height of the rectangle.

info <device> <argument>

Video overlay command

Description

Provides general information about a device.

Argument

One of the following:

`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.

load <device> [<arguments>] Video overlay command

Description

Loads the video buffer.

Argument

None to all of the following:

`file` Specifies the name of the file from which to load data.

`at <rectangle>` Specifies a rectangle relative to the video buffer origin. The rectangle is specified as X1 Y1 X2 Y2. X1 and Y1 are coordinates of the top-left corner of the rectangle; X2 and Y2 specify the width and height of the rectangle.

open <device> [<arguments>] Video overlay command

Description

Initializes the video overlay device.

Arguments

None to all of the following:

`alias <device alias>` Specifies an alternate name for the device element. If specified, the alias must also be used for subsequent references.

`parent <window handle>` Specifies the handle of the parent window. The default is the value of `sysWindowHandle`.

`shareable` Initializes a device element as shareable. Subsequent attempts to open the device 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 <child` | `overlapped` | `popup>` Indicates the style of window to open.

`type <device type>` Specifies the compound device used to control a device element. MCI reserves `overlay` as the overlay player device type. As an alternative to `type`, MCI can use the `[MCI extensions]` entries in the SYSTEM.INI file to select the controlling device based on the extension used by the device element.

put <device> <argument> Video overlay command

Description

Defines the source and destination windows.

Argument

One of the following:

`destination` Sets the whole window as the destination window.

`destination at <rect>` Specifies a rectangle array defining a clipping rectangle relative to the window origin. The rectangle array `<rect>` is specified as `x1 y1 x2 y2`, where `x1 y1` specify the top left corner, and `x2 y2` specify the width and height of the rectangle.

`frame` Specifies that the whole video buffer is used to capture the video image.

`frame at <rect>` Specifies a rectangle array defining a clipping rectangle relative to the video buffer origin. The rectangle array `<rect>` is specified as `x1 y1 x2 y2`, where `x1 y1` specify the top left corner, and `x2 y2` specify the width and height of the rectangle.

`source` selects the whole video buffer for display in the destination window.

`source at <rect>` Specifies a rectangle array defining a clipping rectangle relative to the image origin. The rectangle array `<rect>` is specified as `x1 y1 x2 y2`, where `x1 y1` specify the top left corner, and `x2 y2` specify the width and height of the rectangle.

`video` Selects the whole input video source for display in the destination window.

resume <device>

Video overlay command

Description

Restarts a paused sequence.

save <device> <filename>

Video overlay command

Description

Saves an MCI element.

Arguments

`<filename>`	Specifies the file name and path name used to save the data.

set <device> <argument>

Video overlay command

Description

Sets various control device attributes.

Argument

`video off` | `video on`	Enables or disables video output.

status <device> <argument> Video overlay command

Description

Obtains status information for the device.

Argument

One of the following:

`media present` Returns `true` if the media is in the device; otherwise, returns `false`.

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

`ready` Returns `true` if the animation device is ready.

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

unfreeze <device> at <rect>

Video overlay command

Description

Enables the frame buffer to acquire video data. This command is supported only if `capability can freeze` returns `true`.

Argument

`at <rect>` Specifies a rectangle array defining a rectangle relative to the video buffer origin. The rectangle array `<rect>` is specified as `x1 y1 x2 y2`, where `x1 y1` specify the top left corner, and `x2 y2` specify the width and height of the rectangle.

where <device> <argument> Video overlay command

Description

Obtains the rectangle array specifying the source or destination area.

Argument

One of the following:

`destination` Requests the offset and extent of the destination rectangle.

`frame` Requests the offset and extent of the frame buffer rectangle.

`source` Requests the offset and extent of the source rectangle.

`video` Requests the offset and extent of the video rectangle.

window <device> <argument> Video overlay command

Description

Specifies a given window to display instead of the default window created by the driver. By default, video overlay devices should create a window when opened but should not display it until the device receives the `play` command. Applications providing window handles should manage the display issues that result when the window is sized or when the window handle is switched during play.

Several flags manipulate the window. Since the `status` command can obtain the handle to the current display window, you can use the standard window functions instead.

Argument

One of the following:

`handle <window handle>` Specifies the handle of the destination window used as an alternate to the default window.

`handle default` Specifies that the animation player should create and manage its own window. This flag can be used to set the display back to the driver's default window.

`state hide` Hides the current display window.

`state iconic` Displays the window as an icon.

`state maximized` Maximizes the current display window.

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

`state minimized` Minimizes the current window.

`state no action` Displays a 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` Displays the current display window as it was created.

`state show` Shows the current display window.

`text <caption>` Specifies the caption for the display window.

Videodisc player commands

The MCI commands that support videodisc players are: capability, close, escape, info, open, pause, play, resume, seek, set, spin, status, step, and stop.

capability <device> <argument> Videodisc player command

Description

Reports the capability of the device.

Argument

One of the following:

`can eject` Returns `true` if the device can eject the media. The MCIPIONR.DRV device driver 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; otherwise, returns `false`. CLV discs always return `false`.

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

`CAV` When combined with other items, specifies that the return information applies to CAV-format discs. This is the default.

`CLV` When combined with other items, 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 in frames per second. Returns `0` if the device cannot play fast.

`has audio` Returns `true` if the video device has audio.

`has video` Returns `true`.

`normal play rate` Returns the rate in frames per second. Returns `30` for CLV discs.

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

`uses files` Returns `false`. MCI videodisc players do not use files.

close <device>

Videodisc player command

Description

Closes the device. When the device is no longer used, MCI unloads the device.

escape <device> <string>

Videodisc player command

Description

Sends custom information to a device.

Argument

`<string>` Specifies the custom information sent to the device.

info <device> product

Videodisc player command

Description

Provides general information about a device.

Argument

`product` Returns the product name of the device that the peripheral is controlling. The MCIPIONR.DRV device returns `Pioneer LD-V4200.`

open <device> [<arguments>] Videodisc player command

Description

Initializes the device.

Arguments

None, one, or both of the following:

`alias <device alias>` Specifies an alternate name for the given device. If specified, the alias must also be used for subsequent references.

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

pause <device>

Videodisc player command

Description

Stops playing. For CAV discs, also freezes the video frame.

play <device> [<arguments>] Videodisc player command

Description

Starts playing.

Arguments

None to all of the following:

`fast` | `slow` Indicates that the device should play faster or 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 `fps` flag. `Slow` applies only to CAV discs.

`from <pos>` Specifies the position to start playing in frames for CAV discs and in seconds for CLV discs, unless `track` is also used (in which case, the position is given in tracks). If `from <pos>` is omitted, playing starts at the current position.

`to <pos>` Specifies the position to stop playing in frames for CAV discs and in seconds for CLV discs, unless `chapter` is also used (in which case, the position is given in chapters). If `to <pos>` is omitted, playing stops at the end of the disc.

`reverse` Sets the play direction to backwards. Applies only to CAV discs.

`scan` Indicates the play speed is as fast as possible, possibly with audio disabled. 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.) Applies only to CAV discs.

resume <device>

Videodisc player command

Description

Restarts a paused device.

seek <device> <argument> Videodisc player command

Description

Searches using fast forward or fast reverse with video and audio off.

Argument

One of the following:

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

`to end` Moves to the end of the media.

`to [track] <pos>` Specifies the position to seek. if `track` is used the position is given in tracks.

`to start` Moves to the start of the media.

set <device> <argument> Videodisc player command

Description

Sets various control device attributes.

Argument

One of the following:

`audio all off` | `audio all on` Enables or disables audio output.

`audio left off` | `audio left on` Enables or disables output to the left audio channel.

`audio right off` | `audio right on` Enables or disables 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 `position` information is specified in this format following this command. This time format is the default for CAV discs.

`time format hms` Sets position format to `h:mm:ss` where `h` is hours, `mm` is minutes, and `ss` is seconds. When time and position values are used, they are expressed in this format. On input, `h` can be omitted if it is 0, and `mm` can be omitted if both it and `h` are 0. This time format is the default for CLV discs.

`time format milliseconds` Sets the `position` format to `milliseconds`. All position information is in this format following this command. You can abbreviate milliseconds as `ms`.

`time format track` Sets the `position` format to `tracks`. All position information is specified in this format following this command.

`video on` | `video off` Turns the video on or off.

spin <device> <argument>

Videodisc player command

Description

Starts the disc spinning or stops the disc from spinning.

Argument

One of the following:

`down` Stops the disc from spinning.

`up` Starts the disc spinning.

status <device> <argument> Videodisc player command

Description

Obtains status information for the device.

Argument

One of the following:

`current track` Returns the current track number.

`disk 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; returns `false` if the play direction is backward.

`length` Returns the total length of a disc.

`length track <number>` Returns the length of a track specified by `<number>`.

`media present` Returns `true` if the disc is inserted in the device; otherwise, returns `false`.

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

`mode` Returns `not ready`, `open`, `paused`, `parked`, `playing`, `seeking`, or `stopped` for the device mode.

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

`position` Returns the current position.

`position track <number>` Returns the position of the start of the track specified by `<number>`. The MCIPIONR 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 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 time format.

step <device> [<arguments>] Videodisc player command

Description

Steps the play one or more frames forward or backward. The default action is to step one frame forward. The `step` command applies only to CAV discs.

Arguments

None, one, or both of the following:

`by <frames>` Specifies the number of `<frames>` to step. If a negative value is used, the `reverse` flag is ignored.

`reverse` Steps backward.

stop <device>

Videodisc player command

Description

Stops playing.

Waveform audio commands

The MCI commands that support waveform audio drivers are: capability, close, cue, info, open, pause, play, record, resume, save, seek, set, status, and stop.

capability <device> <argument> Waveform audio command

Description

Requests information about the capabilities of the waveform audio driver.

Argument

One of the following:

`can eject` Returns `false`. Waveform audio drivers cannot eject the media.

`can play` Returns `true` if the device can play and an output device is available.

`can record` Returns `true` if the current audio device supports input.

`can save` Returns `true` if the waveform audio device can save data.

`compound device` Returns `true`. Waveform audio devices are compound devices.

`device type` Returns `waveaudio`.

`has audio` Returns `true` if the current audio device supports playback.

`has video` Returns `false`. Waveform audio devices do not 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.

close <device>

Waveform audio command

Description

Closes the device element and any resources associated with it. MCI unloads the waveform audio device when the last element is closed.

cue <device> [<arguments>] Waveform audio command

Description

Prepares for playing or recording. The `cue` command does not have to be issued prior to playing or recording. However, depending on the device, this command might reduce the delay associated with the `play` or `record` command. This command fails if playing or recording is in progress.

Arguments

None, one, or both of the following:

`input` Prepares the input for recording.

`output` Prepares the output for playing. This is the default.

delete <device> [from <pos> to <pos>] Waveform audio command

Description

Deletes a data segment from the MCI element.

Arguments

None, one, or both of the following:

`from <pos>` Specifies the position to start cutting data. If `from <pos>` is omitted, the cut starts at the current position.

`to <pos>` Specifies the position to stop cutting data. If `to<pos>` is omitted, the cut stops at the end of the file or waveform.

info <device> <argument> Waveform audio command

Description

Provides general information about a device.

Argument

One of the following:

`file` Returns the current file name.

`input` Returns the description of the current waveform audio input device. Returns `none` if an input device is not set. The MCIWAVE.DRV driver returns `Wave Audio Input and Output Device`.

`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`.

open <device> [<arguments>] Waveform audio command

Description

Initializes the device.

Arguments

None to all of the following:

`alias <device alias>` Specifies an alternate name for the given device. If specified, the alias must also be used for subsequent references.

`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 setup. Typically the buffer size is set to four seconds.

`shareable` Initializes the device element as shareable. Subsequent attempts to open the device 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. The MCIWAVE.DRV device does not support shared files.

`type <device type>` Specifies the compound device used to control a device element. As an alternative to `type`, MCI can use the `[`MCI `extensions]` entries in the SYSTEM.INI file to select the controlling device based on the extension used by the device element.

pause <device>

Waveform audio command

Description

Pauses playing or recording.

play <device> [from <pos> to <pos>] Waveform audio command

Description

Starts playing audio.

Argument

None, one, or both of the following:

`from <pos>` Specifies the position to start playing. If `from <pos>` is omitted, playing starts at the current position.

`to <pos>` Specifies the position to stop playing. If `to <pos>` is omitted, playing stops at the end of the file or waveform.

record <device> [<arguments>] Waveform audio command

Description

Starts recording audio. Recording does not overwrite existing data; new data is inserted at the current position. All data recorded after a file is opened is discarded if the file is closed without saving it.

Arguments

None to all of the following:

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

`from <pos>` Specifies the position to start recording. If `from <pos>` is omitted, the device starts recording at the current position.

`to <pos>` Specifies the position to stop recording. If `to <pos>` is omitted, the device records until a `stop` or `pause` command is received.

`overwrite` Specifies that new data will replace data in the device element. The MCIWAVE.DRV device does not support this option.

resume <device>

Waveform audio command

Description

Restarts a paused device.

save <device> <filename> Waveform audio command

Description

Saves the MCI element in its current format.

Argument

`<filename>` Specifies the file name and path name used to save data.

seek <device> to <pos>

Waveform audio command

Description

Moves to the specified location in the file. If the device is already playing or recording, it continues at the new location.

Arguments

`to end` Moves to the end of the device.

`to <pos>` Specifies the position to seek.

`to start` Moves to the start of the device.

set <device> <argument> Waveform audio command

Description

Sets various control device attributes.

Argument

One of the following:

`alignment <integer>` Sets the alignment of data in bytes.

`any input` Uses any input that supports the current format when recording. This is the default.

`any output` Uses any output that supports the current format when playing. This is the default.

`audio all off` | `audio all on` Enables or disables audio output. The MCIWAVE.DRV device does not support this option.

`audio left off` | `audio left on` Enables or disables output to the left audio channel. The MCIWAVE device does not support this option.

`audio right off` | `audio right on` Enables or disables output to the right audio channel. The MCIWAVE device does not support this option.

`bitspersample <integer>` Sets the number of bits per sample played or recorded. The file is saved in this format.

`channels <integer>` Sets the channel 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 as the input. Channel 0 is the first channel.

`output <integer>` Sets the audio channel as the output. Channel 0 is the first channel.

`samplepersec <integer>` Sets the sample rate for playing and recording. The file is saved in this format.

`time format bytes` Sets time format to `bytes`. All position information is specified as bytes following this command.

`time format milliseconds` Sets the position format to `milliseconds`. All position information is in this format following this command. 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.

status <device> <argument> Waveform audio command

Description

Obtains status information for the device.

Argument

One of the following:

`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 `1` for current track.

`format tag` Returns the format type for playing and recording. The file is saved in this format.

`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 #>` Returns the length of the specified track.

`level` Returns the current audio sample value.

`media present` Returns `true`.

`mode` Returns `not ready`, `paused`, `playing`, `recording`, `seeking`, or `stopped` for the device mode. The MCIWAVE device does not return `seeking`.

`number of tracks` Returns the number of tracks. The MCIWAVE device returns `1`.

`output` Returns the output set. If one is not set, the error returned indicates that any device can be used.

`position` Returns the current position.

`position track <track #>` Returns the position of the specified track. The MCIWAVE device returns `0`.

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

`samplepersec` Returns the number of samples per second played or recorded.

`time format` Returns the time format.

stop <device>

Waveform audio command

Description

Stops playing or recording.