Animation Player Index

The Autodesk Animation Player allows you to play animations created using Autodesk Animator, Autodesk 3D Studio, or Autodesk Animator Pro.

Commands

Procedures

File Menu Commands

New Script
Opens a new script, starts the Edit Script dialogue.
If a script is open with unsaved changes, you are given a chance to save it.

Load Script
Opens an existing animation script file.

Edit Script
Starts the Edit Script dialogue, which lets you add, delete, copy, and paste animation files in a script.

Save Script
Saves changes to the current animation script.
Save can only be used when a script has been opened.

Save Script As
Allows you to save an existing script with a new name, leaving the original unchanged. Save As can only be used when a script has been opened.

Related topics:
Creating a Script
Editing Scripts

Open Animation
Opens an existing animation file.
Several file formats can be opened:
- An Autodesk Animator animation.
- An Autodesk Animator Pro animation.
- An Autodesk 3D Studio animation.
- A Windows Device Independent Bitmap (DIB).

Get Sound
Adds sound to an animation. To be used after Open Animation. Cannot be used with scripts.
Use the **Edit Script** Command to add sound to scripts.
Related topics:
Using Sound
Scripting Animations

Anim Settings
Lets you adjust an animation's playback speed, duration, sound, and other characteristics.
Related topic:
Animation Settings

Convert Anim
Converts animation colors for optimal playback under Microsoft Windows.
Creates a new animation with the converted colors.
Cannot be used with scripts or DIBs.
Related Topic:
Animation Colors

Exit

Closes the current file and exits Autodesk Animation Player. If an unsaved script is open, Autodesk Animation Player lets you save it before exiting.

Options Menu Commands

The items checked in the Options Menu are applied to an animation when Autodesk Animation Player opens the file.

Load in Memory
Animations load into memory when opened, resulting in faster playback.

Loop Frame
The last image in a DIB is used as a transition frame between the end and beginning of an animation. Use this option for non-Autodesk animation formats such as .rle.

Full Screen
Animations play back in full-screen VGA mode. Animations smaller than 320 by 200 use the 320 by 200 low resolution mode of the VGA.

Hide Animation
Animations do not appear onscreen until they are played.

Color Cycling OK
Reserve Windows palette colors for color cycling as the colors change. This option is active by default. Turn it off if your animation is losing colors on playback.
Related Topic:
Animation Colors

Use All Colors
Reserve all available Windows palette colors for color cycling when the animation is opened. Use this option if your animation involves palette animation and if it uses fewer than 236 colors.
Related Topic:
Animation Colors

Frame Control Commands

The items **<**, **Stop**, **>**, and **>>** on the Menu Bar are commands that execute immediately when you click on them.

<
Moves the animation to the previous frame.
Moves from the first frame to the last.
Keyboard shortcut: the left arrow key.

Stop
Stops the animation.
Keyboard shortcut: the up arrow key.

>
Moves the animation to the next frame.
Moves from the last frame to the first.
Keyboard shortcut: the right arrow key.

>>
If the animation is not playing, the animation is started.
If the animation is playing, the animation is paused.
Keyboard shortcut: the down arrow key.

Creating a Script

An Autodesk Animation Player script is an ASCII text file containing a list of animations or DIBs to be played in the order they are listed.

You can create a script using either a standard text editor or the New Script command in Autodesk Animation Player.

You can edit a script with a text editor or by using the Edit Script command in Autodesk Animation Player when a script is open. (See Editing Scripts.)

The following are guidelines to follow if you create your own ASCII script outside the Autodesk Animation Player editor:

List a single file on each line, along with any nondefault playback settings.

The first thing on a script line is the name of the file to be played. If the file has an extension, it must be included as part of the name. Follow the file name with playback codes to indicate nondefault playback settings.

If a line becomes too long, continue it by placing a back slash character at the end of the line.

Animation settings codes change the playback settings from the Autodesk Animation Player defaults. Each option consists of a dash (-), followed by a single letter identifying the option, followed by any arguments the option uses. You may place a space between the option and the argument, or not. You must put a space between each of the option's arguments, and between the last argument and the next option.

For example:
D:\PLAYER\BIKER.FLC -L4:35 -F

Note: Playback option codes appear without dashes or arguments when the script file appears in the Autodesk Animation Player Edit Script dialogue.

Animation Settings Codes

The options used in scripts are as follows:

-N *[sound-string [device]]*
Change the sound for the script. *Sound-string* is the sound filename or media location, and *device* is the sound playback device. See Using Sound for more details on sound. If the -N option is missing, sound is turned off. If *sound-string* is missing, the sound already playing continues. Defaults to no sound.
-R *sound-repeat*
Set the number of times the sound repeats. The sound is not repeated if this option is missing. *Sound-repeat* can be 0 or **Forever**, either of which causes the sound to repeat until the animation ends. Defaults to 1.
-D *delay*
Delay the start of a sound from the beginning of an animation. You can use negative numbers to cause a sound to start before an animation. Defaults to 0.
-L *loops*
Set the number of times an animation is to repeat. You can state partial repeats in frames by following the loop count by a colon (:) and the number of frames in the last loop (e.g., **3:20** and **0:5**). A value of **Forever** causes the animation to loop until an external event intervenes. A value of **Sound** loops the animation until the current sound finishes. A value of **Sound:***nn* also loops the animation until the sound finishes, but then continues after the sound is finished until frame *nn* is displayed. For example, **Sound** loops until the sound finishes and ends the animation on the frame displayed when the sound finishes. **Sound:0** loops until the sound ends, but continues through the loop until the end of the animation is reached. **Sound:10** continues after the sound is finished until frame 10 is displayed and then ends the animation. Defaults to 1.
-S *speed*
Set the speed of the animation. Defaults to value designed in the animation.
-P *pause*
Set the length of time the animation pauses when all loops are completed. Defaults to 0.
-M
Load animation into memory. Defaults to playback from disk.
-F
Play animation on full screen. Defaults to playback in a window.
-E
No loop frame; last frame does not contain the deltas between the end and beginning of the animation. This code is irrelevant for Autodesk animations. Corresponds to Loop Frame Present item in the Animation Settings dialogue.
-C
Turn Color Cycling OK off; do not reserve Windows colors for palette animation. Mutually exclusive with -A.
-A
Reserve entire available Windows palette for palette animation. Mutually exclusive with -C.
-I *fade_string fade_duration*
Add a transition to the animation. Defaults to no transition (cut). The *fade_string* can be **cutfrom**, **fromblack**, **fromwhite**, **cutto**, **toblack**, or **towhite**. The *fade_duration* can range from 250 milliseconds to 10000 milliseconds (.25 seconds to 10 seconds).

Using Sound

You can associate sound with animations in scripts, or with a single animation in memory. Use the **Get Sound...** option of the File menu to associate sound with an animation or Windows DIB while it is loaded.

Use the **Get Sound** button in the Edit Script dialogue box to add sounds to scripted animations.

When you select a sound, you must also select the sound device. Autodesk Animation Player can play the following types of sound:

- waveaudio - digitized sound or waveAudio
- cdaudio - audio from a CDROM
- videodisc - audio from a videodisc player
- sequencer - MIDI sound

Digitized Sound

Digitized sound, also called wave audio, is sound that has been recorded in digital format.

Select digitized sound by specifying **waveaudio** as the sound device and selecting a file containing the digitized sound to be played. Usually these file end in .wav.

CDROM Audio

CDROM Audio is sound played from a compact disc on a CDROM player.

Select CDROM audio by specifying **cdaudio** as the sound device and entering the track to be played in the Filename field. Enter the following parameters instead of a file name:

from *pos* **to** *pos*

Pos gives a position on the CDROM disk. Positions are given in tracks, minutes, seconds, and frames in the format *tt*:*mm*:*ss*:*ff*. You can leave off trailing zeros. For example:

from 1 to 2 plays all of track 1.
from 3 to 3:3 plays three minutes of track 3.
from 5 to 6:2 plays all of track 5 and 2 minutes of track 6.

Videodisc Audio

Videodisc players are usually used for images and sound, but Autodesk Animation Player allows you to select a portion of a videodisc to be played along with an animation.

Select videodisc audio by specifying **videodisc** as the sound device and entering the start and stop frames to be played instead of a file name in the Filename field:

from *pos* **to** *pos*

Pos gives the position on the videodisc in frames. Always give the **to** argument to insure that the end of the sound is known. If **from** is not given, the sound starts from the current position.

In this case, position is a time in *hh*:*mm*:*ss* format. Your videodisc player may display position in this format, or it may display position as a frame number. To convert from frame numbers to time, divide by 30 to get seconds, and then convert to *hh*:*mm*:*ss* format. The from position must be at least 1 second. For example, enter

from 0:0:1 to 0:4:0

to play 4 minutes on the disc.

MIDI Sound

MIDI sound is a recording of notes and instruments that is played on electrical instruments or synthesizers.

Select MIDI sound by specifying the **sequencer** sound device and choosing a file containing the recorded notes and instrument data. Usually these files end in .mid or .rmi

Animation Settings

Animations have characteristics that affect how they play back. Some settings, such as number of frames and speed, are designed into the animation when it is created. Others Autodesk Animation Player assigns by default. You can adjust playback settings in Autodesk Animation Player either from the **Anim Settings...** entry of the File menu or by pressing the **Settings** button in the Edit Script dialogue.

The Speed, Loops, and Duration settings control the speed and length of an animation sequence. These three settings are related since the duration of an animation is determined by the speed and the number of times the animation is repeated (loops).

The Lock buttons to the left of these settings are used to fix one of the three values as the other two change. If you change the setting that is locked, it unlocks, and either Speed locks if you are changing Loops or Duration, or Loops locks if you are changing Speed.

Load in Memory
Causes the entire animation to load into memory when you open it. This can take a significant amount of time, but the animation plays faster once it is loaded.

Full Screen
Causes the animation to use the entire screen when playing. If you are playing an animation smaller than 320 pixels by 200 pixels on a VGA, the VGA changes to this resolution before playing the animation.

Loop Frame Present
Only applies to sequences of Windows DIBs. It indicates that the last DIB in the sequence is a loop frame, or transition, to the first DIB in the sequence. If this is not the case, toggle this option off.

Color Cycling OK
Controls palette animation under Windows. If it is checked, colors are reserved for cycling as they are changed. See Animation Colors for more details.

Use All Colors
Controls palette animation under Windows. If it is checked, colors are reserved for cycling when the animation is loaded. See Animation Colors for more details.

Jiffies
Controls the units used to display and enter speed values. If it is highlighted, the speed unit is jiffies.

Frames per Second
Controls the units used to display and enter speed values. If it is highlighted, the speed unit is frames per second.

Speed
Sets the speed of the animation.

Loops
Loops sets the number of times an animation repeats. You can also specify a partial repetition of an animation by following the number of loops with the number of frames in the partial repeat. For example, **3:20** indicates 3 full loops of the animation followed by 20 frames, while **0:5** specifies only 5 frames of the animation. You can also set the number of full repeats to **Sound**, which causes the animation to repeat until the sound finishes. If you specify a partial repeat (*e.g.*, **Sound:1**), the animation stops at the specified frame. **Sound:0** causes the animation to loop until the sound finishes, then complete the final

loop, stopping on the last frame of the animation.

Duration
Sets the duration of the animation in *minutes:seconds:milliseconds*.

Repeat Sound
Sets the number of times the sound is repeated. **Forever** means the sound repeats until the animation finishes.

Delay Sound
The time in seconds to delay the sound from the start of the animation. If the time is negative, the sound starts before the animation starts.

Pause at End
Specifies the length of time in seconds that the animation pauses at the end of all its repetitions. The final frame of the animation is displayed during the pause.

Transitions
You can add transitions to the beginning or the end of an animation by pressing the **Transitions** button. A dialogue appears that lets you enter the starting or ending transition and duration.

You can add fade transitions to smooth the transition between animations. A fade is a color cycling effect, and the **Color Cycling OK** option must be checked for a fade transition to perform correctly. (See Animation Colors.)

Editing Scripts

A script consists of a list of animations. The Edit Script dialogue is provided for editing scripts. You can also edit animation settings from this dialogue, by clicking on the Settings button.

The list of animations is displayed on the left side of the Edit Script dialogue. On the right side of the dialogue is a set of boxes for selecting a new animation to be put in the list. The new animation is inserted after the currently selected animation.

The set of buttons under the list of animations lets you edit the script. **Cut** deletes the currently selected lines in the list. **Copy** copies the lines to the Windows Clipboard. **Paste** pastes lines from the Clipboard after the currently selected line. **Clear** deletes the selected lines, but does not place them in the Clipboard as **Cut** does. **Undo** undoes the last edit.

Use the following keyboard alternatives as shortcuts for the editing buttons:

SHIFT+DEL	Cut
CTRL+INSERT	Copy
DEL	Clear
SHIFT+INSERT	Paste

The buttons on the right side of the dialogue box are:

Exit
Ends the dialogue. This does not save the script.

Test
Performs a test of the script. The test begins at the selected entry in the list.

Save
Saves the script. If the script has never been saved, this option invokes the Save Script dialogue.

Settings
Displays the Animation Settings dialogue and allows you to adjust the playback options of the selected animation.

Get Sound
Changes the dialogue so you can locate a sound to associate with the selected animation. See Using Sound for details on selecting sounds.

You can choose to continue playing the sound associated with the previous animation by pressing the **Continue** button. The **Off** button turns off sound for the selected animation.

When you click on the **Get Sound** button, it changes to a **Cancel** button. Pressing Cancel reverts the Edit Script dialogue back to animation selection mode, canceling changes you've made to sound.

Animation Colors

Windows requires 20 colors for system use, which reduces the available colors for animations from 256 to 236. Normally this is not a problem, because Windows matches colors in order to display an animation properly.

However, there are circumstances in which color matching is not possible. If your animation involves palette animation (that is, if color registers change color), Windows maps unavailable colors to black. Palette animation is typically used for wipes, dissolves, fades, and color cycling.

Windows also remaps color changes between animations in scripts, unless the **Use All Colors** option is checked in the Animation Settings dialogue. Checking this option can improve the transition between animations in scripts. **Use All Colors** can cause colors to be lost if the animation uses more than 236 colors.

Autodesk Animation Player can convert an animation's colors to fit the Windows color model. This process removes any colors from the palette that are not actually used in the animation, and reduces the number of colors used to 236 if more than that are used in the animation. This change prevents Windows from dropping or changing colors. To convert an animation's colors, load the animation and select **Convert Anim...** from the **File** menu.

Authoring Hints

Here are a few things to keep in mind as you create animations for playback under Autodesk Animation Player for Windows:

Colors in Animations
Windows restricts an animation playing in a window to 236 colors. Autodesk Animation Player tries to overcome this restriction (see Animation Colors), but for best results, design your animation with fewer than 236 colors.

Colors in Scripts
The transition between animations in a script may have a short period where the colors are incorrect. This is usually caused when Windows remaps the colors for two different animations. You can alleviate this problem by checking the **Use All Colors** option in the Animation Settings dialogue. (Also See Animation Colors.)

Playback Speed
Playing animations under Windows requires translation from Autodesk Animator format to Windows format, which decreases playback speed. In designing animations, try to minimize changes from one frame to another. For example, full-frame dissolves do not play well.

When you play full-screen animations smaller than 320 by 200 on a VGA, this translation is not required, and you can achieve better performance.

Transitions
Fades can help smooth the transition between animations. A fade is a color cycling effect, and the **Color Cycling OK** option must be active for a fade transition to perform correctly. **Use All Colors** should also be active when you use fades. (See Animation Colors and Animation Settings.)

DIB - Windows Device Independent Bitmap. It may be compressed or uncompressed.

MIDI - Musical Instrument Digital Interface. A standard way to connect electrical musical instruments.

PALETTE ANIMATION - In Autodesk Animator and Autodesk Animator Pro, palette animation occurs when the colors in the palette change over time.

Autodesk Animation DLL

AAPLAY.DLL is a dynamic link library which will play Autodesk Animator animation files and Windows device independent bitmap files.

Using AAPLAY.DLL, applications can load, play and control animations under windows. AAPLAY can use Autodesk Animator Files, or Windows Device Independent Bitmap Files. AAPLAY.DLL can play an animation inside of a window, or it will take over the entire screen for animation playing.

The C language header file aaplay.h makes all of the declarations needed for using the AA Play Library.

See Using the Windows Animation Player

Function Summary

Using the Windows Animation Player

There are three basic phases to using the player. During the first phase, an application must connect to the player. Once the connection is established, animations can be loaded and played. Finally the application must disconnect from the player before exiting.

In order to connect to the player, the application must call the function aaOpen. This function has no arguments and returns a zero if the connection fails, or a non-zero value if the connection succeeds. If the connection fails, no other call to the library should be made. aaOpen should not be called again until after a call to aaClose is made. The function aaGetCaps can be used to determine what player capabilities are available. Depending on the system and version, sound or scripting may not be available, and only reduced frame timing may be available.

To disconnect from the player, the function aaClose is called. This function has no arguments or return value. The library should not be used after aaClose is called.

In order to play animations they must first be loaded using aaLoad. aaLoad returns an unsigned integer which is used to reference a loaded animation in other library functions. These functions can be used to begin play, stop or pause play, and reposition the animation. When an application is finished with an animation, it can be unloaded using aaUnload.

After the library has been opened, an application can use aaGetFile to prompt for an animation or sound file name. Of course, an application can also use its own file name dialog, or some other mechanism to obtain the names of animations.

Two primitives are supplied for frame synchronization. These primitives allow an application to request call backs from the animation player when certain frames are to be drawn. Using these call backs, the application can synchronize any other event with an animation frame. The primitives are aaNotify which schedules a call back, and aaCancel which cancels a call back. When a script is playing, call backs can only be scheduled for the currently playing animation. In order to facilitate synchronization in scripts, a call back is made at the start of each animation in the script.

Call backs are provided for frame synchronization and state and error notification. The call back is made to the Window Procedure of the window set as the call back window. When an animation is loaded, the call back window is set to the hwnd parameter, but it may be changed by aaSetParm or aaSetParmIndirect. Two messages are used and the message number of these messages is determined using RegisterWindowMessage. The messages are identified by the following strings:

"AAPLAY Notify"
Message string for frame synchronization messages.

"AAPLAY Stop"
Message string for status and error messages.

The call back function should be declared as follows:

```
LONG FAR PASCAL CallBack(hwnd, msg, wparam, lparam);
HWND hwnd;
WORD msg;
WORD wparam;
DWORD lparam;
```

The hwnd parameter always identifies the owner of the animation window. The msg parameter identifies the message, and is found by calling RegisterWindowMessage with the appropriate

message string. The wparam parameter always identifies the animation that generated the message. It gives the value returned by aaLoad when the animation was loaded.

For frame synchronization messages, lparam is the value requested by the application when the call back was requested, or zero for frame synchronization call backs generated at the beginning of an animation in a script.

For status and error messages, lparam is an error code. If it is zero, the animation stopped at the end without an error. The following error codes are defined:

AA_BADPLAY - 1
An error occurred reading the animation.

AA_BADNOTIFY - 2
A memory error occurred attempting a frame synchronization message.

AA_BADSCRIPT - 3
An error occurred loading an animation or sound for a script.

Several function are provided for retrieving and changing parameters of an animation. The application can provide a user interface or use the one provided in aaPrompt. For normal animations, the application should retrieve the values of an animations parameters and save them for later use. Scripted animations will save the animation parameters in the script.

aaOpen

BOOL aaOpen(void)

aaOpen initializes the player for the application. TRUE is returned if the initialization succeeded. Otherwise FALSE is returned.

An application must call aaOpen before making any other calls to the AA Play Library. aaOpen should not be called again before aaClose is called.

aaClose

void aaClose(void)

aaClose releases the player from the application. Any loaded animations used by the application are unloaded.

Other calls to the AA Play Library should not be performed after this call is made. aaOpen should be called again to reopen the library, if necessary.

aaLoad

HAA aaLoad(lpzFileName, hWnd, wMode, x, y, cx, cy, orgx, orgy)

aaLoad loads an animation in preparation for playing. It returns a number between 1 and 65535, which is used to reference the loaded animation in other library calls. Once an animation is loaded, it can be played, and various parameters controlling its playing can be altered.

If aaLoad is unable to load the animation, NULL (zero) is returned.

Parameters

LPSTR lpzFileName

The name of the animation file to be opened or the contents of an animation script depending on the value of wMode. OpenFile is used to open animation files and so the normal Windows file searching algorithm is used. This search algorithm will first search the current directory, then each directory in the PATH environment variable.

If the wMode value indicates that lpzFileName is the contents of a script, the animation is opened as an untitled script.

HWND hWnd

The handle of a window that is to own this animation. It must be supplied but may be null. Coordinates are given relative to this window and it is used to receive status and frame synchronization messages, when these are sent. The message numbers used for the status and notification messages are retrieved using RegisterWindowMessage. The following two messages are used by AAPLAY:

AA_STOP - "AAPLAY Stop"

The animation is being stopped. If lParam is non zero, the animation is being stopped because of an error. This message is not sent when the application stops the animation using aaStop.

AA_NOTIFY - "AAPLAY Notify"

Sent because the user requested notification when a frame is drawn. See aaNotify and aaCancel.

WORD wMode

A word which indicates how the file is to be loaded. This value is found by adding the values desired below:

AA_MEMORYLOAD - 1

Loads the entire animation into memory. This requires more time, and may fail because of insufficient memory.

AA_HIDEWINDOW - 2

Normally the frame at which the animation is stopped is displayed on the screen. If this value is added into the mode, the animation is only displayed when it is playing.

AA_NOPALETTE - 4

Inhibits palette animation. When palette animation is enabled, some colors may be lost if more that 236 colors are used.

AA_RESERVEPALETTE - 8

If palette animation is not inhibited, this flag will reserve all of the palette entries for animation. Some colors may be lost if more that 236 colors are used. If the palette is changed without this flag being set, Windows will send a message to all windows on the screen informing them of the palette change. This can cause the animation to stop momentarily.

In order to prevent the palette change messages when changing animations in scripts, or using aaReLoad, this flag must be set in the current animation and the following animation.

AA_LOOPFRAME - 16

Indicates that the last frame of a Windows device independent bitmap animation is actually the deltas between the last frame of the animation and the first. This value is not used for Autodesk Animator Animations.

AA_FULLSCREEN - 32

Indicates that the animation is to be played on the full screen, not within the window hWnd.

AA_STOPNOTIFY - 64

Prevents notification messages from being sent to the window identified by hWnd.

AA_STOPSTATUS - 128

Prevents status messages from begin sent to the window identified by hWnd.

AA_NOFAIL - 256

If a memory load fails due to memory limitations, the animation is loaded to play from disk.

AA_BUILDSCRIPT - 1024

Indicates that an untitled script is to be built using the contents of the string addressed by lpzFileName. If this mode is used for the developer version of the player, FALSE is returned.

WORD x, y, cx, cy

The coordinate of the upper left corner of the window used to display the animation, and the width (cx) and height (cy) of that window. X and y are relative to the upper left corner of the client are of the window hWnd. These values are modified to force the window to be contained in the window identified by hWnd.

WORD orgx, orgy

The coordinate of the animation displayed at the upper left corner of the window used to display the animation.

aaUnload

BOOL aaUnload(hAa)

Unloads an animation loaded by aaLoad. If the animation is playing, it is stopped. aaUnload will ask the user to save a modified script. This can be inhibited by clearing the modified flag using aaSetParm.

Parameters

HAA hAa

A handle returned by aaLoad.

aaReLoad

BOOL aaReLoad(hAa, lpzFileName, wMode, wMask)

Loads another animation over an existing one. If the two animations use different colors, then setting AA_RESERVEPALETTE in both the existing animation and in wMode can improve the performance of this function. The existing animation must be stopped, paused, or have finished playing in order to make this call.

Parameters

HAA hAa

The handle of the animation returned by aaLoad.

LPSTR lpzFileName

The name of the animation file to be opened or the contents of an animation script depending on the value of wMode. OpenFile is used to open animation files and so the normal Windows file searching algorithm is used. This search algorithm will first search the current directory, then each directory in the PATH environment variable.

If the wMode value indicates that lpzFileName is the contents of a script, the animation is opened as an untitled script.

WORD wMode

This value is used exactly the same as in aaLoad. It uses the same values with the following additional value:

AA_DONTPAINT - 512

Inhibits painting of the initial frame until the animation begins playing.

WORD wMask

A mask used for setting the mode. Setting of specific mode values can be inhibited by adding the value not to be set into this argument.

aaPlay

BOOL aaPlay(hAa)

Plays the animation loaded by aaLoad. Play begins from the current position. aaPlay returns after the animation has begun playing. Returns TRUE if the animation is playing.

Parameters

HAA hAa

A handle returned by aaLoad.

aaNotify

BOOL aaNotify(hAa, lPosition, lParam)

Requests a frame synchronization message for the application using the DLL. The message is sent immediately before the frame at position lPosition is drawn. lParam is a parameter set by the user which is passed to the message handler. The word parameter to the message routine is always the handle hAa. If the position at the time of the call is needed, it can be retrieved using aaGetParm. This call is used to synchronize other events to an animation frame.

The player will automatically generate frame synchronization messages whose lParam is zero, when an animation begins playing. If the application requests frame synchronization with lParam zero, it is the responsibility of the application to determine which synchronization message is begin sent.

Parameters

HAA hAa

A handle returned by aaLoad.

DWORD lPosition

The position at which the notification is to take place. The high order word is the loop and the low order word is the frame.

DWORD lParam

A double word passed to the notification function.

aaCancel

WORD aaCancel(hAa, lLowPos, lHighPos)

Cancels frame synchronization messages requested using aaNotify. The number of messages canceled is returned.

Parameters

HAA hAa

A handle returned by aaLoad.

DWORD lLowPos

The lower loop and frame count where notification messages are canceled.

DWORD lHighPos

The upper loop and frame count where notification messages are canceled.

aaStop

BOOL aaStop(hAa)

Stops the playing animation began by aaPlay. Returns TRUE if the animation is stopped. Stopped animations begin with the first frame of the animation, when they are replayed. You may also use aaSetParm to start a stopped animation at a frame other than the first frame.

Parameters

HAA hAa

A handle returned by aaLoad.

aaPause

BOOL aaPause(hAa)

Pauses a playing animation. Returns TRUE if the animation is paused. A paused animation is still considered playing, so no other full screen animations will play. Paused animations will continue playing from the current frame, when they are replayed by calling aaPlay.

Parameters

HAA hAa

A handle returned by aaLoad.

aaPrompt

BOOL aaPrompt(hAa, lpName)

Produces a dialog box that prompts the user for parameters for playing the animation whose handle is hAa.

This call acts differently for scripts and normal animations. For normal animations, the values entered by the user can be retrieved using aaGetParm or aaGetParmIndirect. Scripts can be saved in a file using aaSave, or the contents of the script can be retrieved using aaGetParm. Scripts are not available in the developer version of the player.

If a script is modified, aaUnload will ask the user to save the script when it is unloaded. This can be prevented by setting the modified flag to zero, using aaSetParm.

Parameters

HAA hAa

A handle returned by aaLoad.

LPSTR lpName

The name of the animation. This name is used only to provide a caption for the dialog for normal animations. If it is NULL, no name is provided in the dialog caption in this case.

aaGetParm

LONG aaGetParm(hAa, wType)

Gets current parameters for playing the animation. Parameters that are not valid are returned as zero. If the animation is a script, the parameters returned are for the currently visible animation.

Parameters

HAA hAa

A handle returned by aaLoad.

WORD wType

The type of parameter to be retrieved:

AA_STATUS - 1

Returns a word containing current status of the animation. The status of an animation can be:

AA_STOPPED - 1

The animation is loaded and is not playing.

AA_QUEUED - 2

The animation is loaded and aaPlay has been used to start the animation, but the animation cannot be started because either it is a full screen animation, or a full screen animation is playing, or both. The animation will be begun as soon as possible.

AA_PLAYING - 3

The animation is playing.

AA_PAUSED - 4

The animation has been paused using aaPause.

AA_DONE - 5

A transitory state after an animation has finished playing, but before it has been stopped. The animation may be restarted from this state using aaPlay or reloaded using aaReLoad.

AA_FILETYPE - 2

Returns a word containing the format of the animation on disk. The values returned are:

AA_FLI - 1

Animator FLI format.

AA_DIB - 2

Windows DIB format.

AA_SCRIPT - 3

The animation is a script. This value will never be returned by the developer version of the player.

AA_MODE - 3

Returns a word containing the current mode of the animation. The values are the same as the mode described in aaLoad, except AA_NOFAIL and AA_BUILDSCRIPT are never set.

AA_WINDOW - 4

Returns a word containing the handle of the window that owns the animation.

AA_SPEED - 5

Returns a word containing the speed of the animation, given in milliseconds per frame. This is the current speed of the animation.

AA_DESIGNSPEED - 6

Returns a word containing the designed speed of the animation, given in milliseconds per frame.

AA_FRAMES - 7

Returns a word containing the number of frames in the animation. If the animation type is a DIB file, this number will be unknown, until the animation has been completely played. If the number of frames is unknown, 0 is returned.

AA_POSITION - 8

Returns a long containing the current position in the animation. The high order word is the current loop, the low order word is the current frame. The first loop is loop 0, and the first frame is frame 0.

AA_LOOPS - 9

Returns a long returning the position of the frame at which the animation ends. A 0 indicates that the animation will not stop until an aaStop call is made. This number has the same format as AA_POSITION.

If the value of the high order word of AA_LOOPS is AA_LOOPSOUND (which is 65535), then the animation loops until the sound finishes playing. If the low order word is also this value, the animation will end with the sound. Otherwise, the animation will stop on the frame number given by the low order word, after the sound has finished.

AA_X - 10

Returns the x coordinate of the upper left corner of the animation window.

AA_Y - 11

Returns the y coordinate of the upper left corner of the animation window.

AA_CX - 12

Returns the width of the animation window.

AA_CY - 13

Returns the height of the animation window.

AA_ORGX - 14

Returns the x coordinate in the animation, displayed at the upper left corner of the animation window.

AA_ORGY - 15

Returns the y coordinate in the animation, display at the upper left corner of the animation window.

AA_WIDTH - 16

Returns the width of the animation.

AA_HEIGHT - 17

Returns the height of the animation.

AA_RPTSOUND - 18

Returns the number of times the sound will repeat before stopping. Zero indicates that the sound will repeat until the animation ends. Zero is always returned in the developer version of the player.

AA_PAUSE - 19

Returns the length of time to pause this animation at the end in milliseconds.

AA_DELAYSND - 20

Returns length of time to delay the sound from the start of the animation in milliseconds. The delay is negative if the sound is to start before the animation. Zero is always returned in the developer version of the player.

AA_TRANSIN - 21
Returns the transition at the beginning of the animation. This transition may be:

AA_CUT - 0

No transition is performed.

AA_FADEBLACK - 1

Fade from black transition is performed.

AA_FADEWHITE - 2

Fade from white transition is performed.

AA_TRANSOUT - 22

Returns the transition at the end of the animation. The transition may be:

AA_CUT - 0

No transition is performed.

AA_FADEBLACK - 1

Fade to black transition is performed.

AA_FADEWHITE - 2

Fade to white transition is performed.

AA_TIMEIN - 23

Returns the duration of the transition at the beginning of the animation in milliseconds. This duration may be from .250 seconds to 10.000 seconds.

AA_TIMEOUT - 24

Returns the duration of the transition at the end of the animation in milliseconds. This duration may be from .250 seconds to 10.00 seconds.

AA_CALLBACK - 25

Returns the handle of the window used to receive the status and notification messages.

AA_ANIMWND - 26

Returns the handle of the window that actually contains the animation.

AA_MODFLAG - 100

Returns the modified flag for the script. Zero means the script is not modified and a non-zero value means the script is modified.

AA_SCRIPTNAME - 101

Returns a global memory handle containing the name of the script. If the script is untitled, or the animation is not a script, zero is returned.

AA_ANIMATION - 102

Returns the number of the currently visible animation in a script. Zero is the first animation of a script, or returned if the animation is not a script.

AA_ANIMATIONCOUNT - 103

Returns the number of animations in a script. Zero is returned if the animation is not a script.

AA_SCRIPTCONTENTS - 104

Returns a global memory handle containing the contents of the script. This handle contains the script as it would appear in a text file, and is terminated by a null character. If the script is empty, or the animation is not a script, zero is returned.

AA_LASTERROR - 1001

Return the code for the last error detected by the player. This type should be used immediately after detecting an error to determine the error. If the animation handle is zero, the player will search for the last error for the calling application. If the animation handle is a valid handle, the player searches for the last error for the application owning the animation. The error codes are:

AA_ERR_NOERROR - 0

No error was detected.

AA_ERR_NOMEMORY - 256

An out of memory condition was detected.

The handle used in the last call to the player was invalid.

AA_ERR_NOTIMERS - 258

A system timer could not be allocated for the animation.

AA_ERR_BADSOUND - 259

The sound could not be opened.

AA_ERR_NOSCRIPT - 260

The operation in error requires a script.

AA_ERR_WRITEERR - 261

An error occured while saving the script.

AA_ERR_BADANIMATION-262

The animation could no be opened.

AA_ERR_BADWINDOWHANDLE-512

An invalid window handle was given to the player.

AA_ERR_WINDOWCREATE-513

An error occured when creating the animation window.

AA_ERR_DLGERROR - 514

An unexpected error occured while processing a dialog box.

AA_ERR_INVALIDSTATUS 768

The function failed because the state of the animation did not allow it.

AA_ERR_BADDIBFORMAT-769

The Windows DIB file is corrupted.

AA_ERR_BADFLIFORMAT- 770

The Autodesk Animator or Animator Pro file is corrupted.

AA_ERR_UNRECOGNIZEDFORMAT771

The file is in an unrecognized format.

AA_ERR_NOSOUND - 772

Sound is not supported by the player.

AA_ERR_NOTVALIDFORSCRIPTS773

The request is not allowed for scripts, only for animations.

AA_ERR_INVALIDFILE- 774

The file handle for the animation is invalid.

AA_ERR_NOSCRIPTS - 775

Scripts are not supported by the player.

AA_ERR_SPEED - 1024

The speed is invalid. The speed must be from 19 to 1000 milliseconds.

AA_ERR_LOOPS - 1025

The loops are invalid. The number of loops may range from 0 to 999.

AA_ERR_RPTSOUND - 1026

The number of sound repetitions are invalid. This value may range from 0 to 1000.

AA_ERR_PAUSE - 1027

The time to pause at the end of the animation is invalid. This value may range from 0 to 50000 milliseconds.

AA_ERR_TRANSIN - 1028

The starting transition is invalid. Transitions must be one of the codes described above.

AA_ERR_TIMEIN - 1029

The duration of the starting transition is invalid. It ranges from 250 to 10000 milliseconds.

AA_ERR_TRANSOUT- 1030

The ending transition is invalid. It must be one of the codes described above.

AA_ERR_TIMEOUT - 1031

The time of the ending transition is invalid. It may rang from 250 to 10000 milliseconds.

AA_ERR_DELAYSND - 1032

The delay of the sound from the start of the animation is invalid. It may range from -3000000 to 3000000 milliseconds.

AA_ERR_INVALIDTYPE- 1033

The type parameter to aaGetParm, aaSetParm or aaSetParmIndirect is invalid.

AA_ERR_DUPLICATENOTIFY-1280

An attempt was made to add a duplicate notification.

AA_ERR_NOSWITCH - 1536

A script line is missing an option switch, or the switch is invalid.

AA_ERR_PARSELOOPS- 1537

The number of loops on a script line is invalid.

AA_ERR_PARSESPEED- 1538

The speed on a script line is invalid.

AA_ERR_BADRPTSOUND- 1540

The number of sound repetitions on a script line is invalid.

AA_ERR_PARSEPAUSE- 1541

The pause at the end of an animation on a script line is invalid.

AA_ERR_PARSETRANS- 1542

A transition value is invalid.

AA_ERR_PARSEDELAYSND-1543

The delay of a sound from the beginning of the animation is invalid.

AA_LASTERRORMESSAGE - 1002

A handle to a string containing text for the error code retrieved by AA_LASTERROR is returned. This string may be displayed in a message box. If the error is in a script, and the animation name of the line which caused the error is known, the name of the animation will be in the message. The player will also copy the message into a buffer, if aaSetParm is used.

aaGetParmIndirect

BOOL aaGetParmIndirect(hAa, lpAparm, wSize)

Gets parameters for playing the animation into a structure. This allows an application to easily obtain all of an animations parameters.

Parameters

HAA hAa

A handle returned by aaLoad.

LPAAPARMS lpAparm

A long pointer to the structure used to hold the parameters. The format of this structure is:

```
struct {
    BYTE    aa_status;              /* offset  0 */
    BYTE    aa_filetype;            /* offset  1 */
    BYTE    aa_mode;                /* offset  2 */
    BYTE    aa_bitpix;              /* offset  3 */

    HWND    aa_hwnd;                /* offset  4 */
    int     aa_x,                   /* offset  6 */
            aa_y,                   /* offset  8 */
            aa_cx,                  /* offset 10 */
            aa_cy;                  /* offset 12 */
    int     aa_orgx,                /* offset 14 */
            aa_orgy;                /* offset 16 */

    AASPEED aa_speed;               /* offset 18 */
    AASPEED aa_designspeed;         /* offset 20 */

    WORD    aa_frames;              /* offset 22 */

    DWORD   aa_position;            /* offset 24 */
    DWORD   aa_loops;               /* offset 28 */
    WORD    aa_rptsound;            /* offset 30 */
    WORD    aa_pause;               /* offset 32 */
    LONG    aa_delaysnd;            /* offset 34 */
    BYTE    aa_transin;             /* offset 36 */
    BYTE    aa_transout;            /* offset 37 */
    WORD    aa_timein;              /* offset 38 */
    WORD    aa_timeout;             /* offset 40 */
    WORD    aa_callback;    /* offset 42 */
    WORD    aa_animwnd;             /* offset 44 */
};
```

The contents of each field is described in aaGetParm.

WORD wSize

The size of the parameter structure.

aaSetParm

HAA aaSetParm(hAa, wType, wValue1, lValue2)

Sets parameters for playing the animation. The use of wValue1 and lValue2 depends on wType. The values of each field is described in aaGetParm. The handle of the animation with the new parameters is returned. If the value could not be set, NULL is returned. When NULL is returned, the original animation handle is still valid.

Parameters

HAA hAa

A handle returned by aaLoad.

WORD wType

The type of parameter to be set. The values of wValue1 and lValue2 vary depending on the parameter type.

AA_MODE - 3

wValue1 is the current mode of the animation. The low order word of lValue2 is a mask for setting the mode. Mode values which are not to be set can inhibited by adding the value into the mask.

AA_WINDOW - 4

wValue1 is a the handle of a window in which the animation is to play. The call back window is changed if it has not been changed to another window handle.

AA_SPEED - 5

wValue1 is the speed, in milliseconds per frame.

AA_POSITION - 8

lValue2 is the number of frames and loops the frame position is to be moved. The high order word is the number of loops, the low order word is the number of frames. If the frames value wraps past the end of the animation, it is carried into the loops value.

wValue1 specifies the starting position. It is 0 if the starting position is the beginning of the animation, 1 if the starting position is the current frame, and 2 if the starting position is the end of the animation.

If the position is set while an animation is stopped, the animation will start from the given position, instead of from the beginning. In this case any sound associated with the animation is also started from its current position.

AA_LOOPS - 9

lValue2 is the position of the frame at which the animation ends.

If the value of the high order word of AA_LOOPS is AA_LOOPSOUND (which is 65535), then

the animation loops until the sound finishes playing. If the low order word is also this value, the animation will end with the sound. Otherwise, the animation will stop on the frame number given by the low order word, after the sound has finished. In the developer version of the player the value AA_LOOPSOUND is accepted, but it is interpreted as though the animation plays forever.

AA_X - 10

wValue1 is the x coordinate of the upper left corner of the animation window.

AA_Y - 11

wValue1 is the y coordinate of the upper left corner of the animation window.

AA_CX - 12

wValue1 is the width of the animation window.

AA_CY - 13

wValue1 is the height of the animation window.

AA_ORGX - 14

wValue1 is the x coordinate in the animation, displayed at the upper left corner of the animation window.

AA_ORGY - 15

wValue1 is the y coordinate in the animation, display at the upper left corner of the animation window.

AA_RPTSOUND - 18

wValue1 is the number of times the sound is to repeat before stopping. The sound is always stopped when the animation is stopped or paused. The developer version of the player ignore this parameter.

AA_PAUSE - 19

wValue1 is the length of time to pause the animation at its end, in milliseconds.

AA_DELAYSND - 20

lValue2 is the length of time to delay the sound from the start of the animation in milliseconds. The delay is negative if the sound is to start before the animation. The developer version of the player ignore this parameter.

AA_TRANSIN - 21

wValue1 is the transition at the beginning of the animation. This transition may be:

AA_CUT - 0

No transition is performed.

AA_FADEBLACK - 1

Fade from black transition is performed.

AA_FADEWHITE - 2

Fade from white transition is performed.

AA_TRANSOUT - 22

wValue1 is the transition at the end of the animation. The transition may be:

AA_CUT - 0

No transition is performed.

AA_FADEBLACK - 1

Fade to black transition is performed.

AA_FADEWHITE - 2

Fade to white transition is performed.

AA_TIMEIN - 23

wValue1 is the duration of the transition at the beginning of the animation in milliseconds. This duration may be from .250 seconds to 10.000 seconds.

AA_TIMEOUT - 24

wValue1 is the duration of the transition at the end of the animation in milliseconds. This duration may be from .250 seconds to 10.00 seconds.

AA_CALLBACK - 25

wValue1 is the handle of the window used to receive the status and notification messages.

AA_MODFLAG - 100

wValue1 contains the modified flag for the script. Zero means the script is not modified and a non-zero value means the script is modified. The developer version of the player ignore this parameter.

AA_SCRIPTNAME - 101

lValue2 contains a pointer to the new name of the script. If lValue2 is zero, the script is untitled. The developer version of the player ignore this parameter.

AA_ANIMATION - 102

lValue2 contains the number of the animation in the script which is to become visible. The developer version of the player ignore this parameter.

wValue1 specifies the starting position. It is 0 if the starting position is the beginning of the

animation, 1 if the starting position is the current frame, and 2 if the starting position is the end of the animation.

AA_LASTERRORMESSAGE - 1002

The string containing text for the error code retrieved by AA_LASTERROR is copied into a buffer. The buffer size is given in wValue1, and the address of the buffer is given in lValue2. This string may be displayed in a message box. If the error is in a script, and the animation name of the line which caused the error is known, the name of the animation will be in the message. The player will also return a handle to the message text, if aaGetParm is used.

aaSetParmIndirect

HAA aaSetParmIndirect(hAa, dwType, lpAparm, wMask)

Sets the animation parameters from a structure. The handle of the animation with the new parameters is returned. If the value could not be set, NULL is returned. When NULL is returned, the original animation handle is still valid.

Parameters

HAA hAa

A handle returned by aaLoad.

DWORD dwType

A mask containing a bit for each parameter. Any parameter is set only when the corresponding mask bit is on. The following values define the mask bits:

AA_SETMODE	-	1
AA_SETWINDOW	-	2
AA_SETSPEED	-	4
AA_SETPOSITION	-	8
AA_SETLOOPS	-	16
AA_SETX	-	32
AA_SETY	-	64
AA_SETCX	-	128
AA_SETCY	-	256
AA_SETORGX	-	512
AA_SETORGY	-	1024
AA_SETRPTSOUND	-	2048
AA_SETPAUSE	-	4096
AA_SETDELAYSND	-	8192
AA_SETTRANSIN	-	16384
AA_SETTRANSOUT	-	32768
AA_SETTIMEIN	-	65536
AA_SETTIMEOUT	-	131072
AA_SETCALLBACK	-	262144

LPAAPARMS lpAparm

A long pointer to the structure containing the parameters being set. See aaGetParmIndirect for a description of this structure.

WORD wMask

A mask used for setting the mode if AA_SETMODE is set in wType. Setting of specific mode values can be inhibited by adding the value not to be set into this argument. This argument is used only when AA_SETMODE is set in wType.

aaShow

BOOL aaShow(hAa, bShow)

Shows or hides the animation window. This setting is independent of the mode value AA_HIDEWINDOW. This routine will hide playing animations and show hidden stopped animations.

Parameters

HAA hAa

The handle of the animation returned by aaLoad.

BOOL bShow

The animation window is shown if bShow is TRUE, and hidden if bShow is FALSE.

aaSound

BOOL aaSound(hAa, lpszDevice, lpszSound, wMode)

Associates a sound with an animation. The sound is begun with the animation and will end when the animation ends, unless it is shorter than the animation. If it is shorter than the animation, it can be looped using aaSetParm with AA_RPTSOUND. FALSE is always returned by the developer version of the player.

If the appropriate flag is set in wMode, when the sound is associated, an alias for the sound is created.

Parameters

HAA hAa

The handle of the animation returned by aaLoad.

LPSTR lpszDevice

If AA_SNDDEVICEID is not set in wMode, this argument contains the name of the device used to play the sound. If you have setup MCI extensions, this argument can be NULL. If AA_SNDDEVICEID is set in wMode, this argument contains the MultiMedia Control Interface Type ID.

LPSTR lpszSound

Either the name of the file containing the sound to be played, or a string used to play the sound. The format of the file must be supported by the device used to play it. If the device specified by lpszDevice is not given, or requires a file to be played, this argument is the name of the file to play. If the device does not use files for playing (cdaudio, videodisc, videotape, etc), this argument contains the play arguments for an MCI Command string to the device.

WORD wMode

This argument gives the mode of the sound The following modes are valid:

AA_SNDFREEZE - 1

Normally the animation continues to play when the sound is being prepared for playing. Setting this value prevents this from happening.

AA_SNDDEVICEID - 256

Indicates that the device value is not the name of a sound device, but is the MultiMedia Control Interface Type ID.

AA_SNDBUILDALIAS - 512

Builds an alias for the requested sound. This alias is formed by concatenating the word "AAPLAY" and the decimal value of the handle, hAa. This alias can be used to alter the playing of the sound.

aaGetCaps

WORD aaGetCaps(type)

Returns information on the current capabilities of the animation player.

Parameters

WORD type

The type of information to be returned. This parameter can take on the following values:

AA_CAP_TIMER - 1

Returns the accuracy of the timer in milliseconds. The time between successive frames is always an integral multiple of the value returned. The value returned is 55 milliseconds if the Multimedia extensions are not installed, or 1 millisecond if the extensions are installed.

AA_CAP_SOUND - 2

Returns zero if sound support is not available in the player. Otherwise a non-zero value is returned. Sound is not available in the developer version of the player, when the Multimedia extensions are not installed, or when there are no drivers in the [MCI] section of system.ini. Some versions of the player may not support sound even when the Multimedia extensions are installed and there are drivers in the [MCI] section of system.ini.

AA_CAP_SCRIPT - 3

Returns zero if script support is not available in the player. Otherwise a non-zero value is returned. Scripts are not available in the developer version of the player.

aaSave

int aaSave(hAa, wMode)

Saves the script of the animation whose handle is hAa. -1 is returned if the hAa is not valid or an I/O error occurs during the save. 0 is returned, if the user cancels the save. Otherwise a positive integer is returned. 0 is always returned by the developer version of the player.

Parameters

HAA hAa

The handle of the animation returned by aaLoad.

WORD wMode

The mode used to save the animation. It is formed by adding the following values:

AA_SAVE_IFMODIFIED - 1

The script is only saved if it has been modified.

AA_SAVE_AS - 2

The user is prompted for a new file name to save the script. If the script is untitled, the used is always prompted for a file name, whether or not this flag is set.

AA_SAVE_CANCEL - 4

When saving a modified file, a Cancel button is added to the Save File Alert.

aaGetFile

int aaGetFile(wFlags, lpzFileName, wSizeFile, lpzDeviceName, wSizeDevice)

Prompts the user for the name of a file, and copies the name entered to the location addressed by lpzFileName. If the file is the name of a sound file, the Sound device name is copied to the location addressed by lpzDeviceName. The return value is 0, if the user presses the cancel button, -1 if the selected file does not exist, or a positive number if the selected file does exist.

Parameters

WORD wFlags

Flags used to control the dialog box. It is formed by adding the following values:

AA_GETFILE_MUSTEXIST- 1

If set the file entered must exist before aaGetFile will return.

AA_GETFILE_NOSHOWSPEC- 2

If set the search specification is not displayed in the file name edit control.

AA_GETFILE_SAVE - 4

If set the OK button is changed to read "Save", and the caption reads "Save". Setting this value automatically sets the AA_GETFILE_USEFILE flag. This flag is ignored in the developer version of the player.

AA_GETFILE_OPEN - 8

If set the OK button is changed to read "Open", and the caption reads "Open"

AA_GETFILE_USEDIR - 16

If set the directory of the current contents of the string addressed by lpzFileName is used as the initial directory in the dialog.

AA_GETFILE_USEFILE - 32

If set the file name of the current contents of the string addressed by lpzFileName is put in the file name edit control.

AA_GETFILE_SOUND - 64

If set the file specification for sound files is used in the dialog, and the caption reads "Sound". If none of AA_GETFILE_SCRIPT, AA_GETFILE_ANIMATION and AA_GETFILE_SOUND are set in wFlags, the file specification for animation and script files is used in the dialog, and the caption reads "Animation". Setting this flag will cause the developer version of the player to return 0, indicating the cancel button was pressed.

AA_GETFILE_SCRIPT - 128

If set the file specification for script files is used in the dialog, and the caption reads "Script". If

none of AA_GETFILE_SCRIPT, AA_GETFILE_ANIMATION and AA_GETFILE_SOUND are set in wFlags, the file specification for animation and script files is used in the dialog, and the caption reads "Animation". Setting this flag will cause the developer version of the player to return 0, indicating the cancel button was pressed.

AA_GETFILE_ANIMATION - 256

If set the file specification for animation files is used in the diloag and the caption reads "Animation". If none of AA_GETFILE_SCRIPT, AA_GETFILE_ANIMATION and AA_GETFILE_SOUND are set in wFlags, the file specification for animation and script files is used in the dialog, and the caption reads "Animation".

LPSTR lpzFileName

A pointer to a string used to hold the file name entered.

WORD wSizeFile

The maximum size of the file name which can be returned.

LPSTR lpzDeviceName

A pointer to a string used to hold the Sound Device name entered. This parameter is only used if AA_GETFILE_SOUND is set in wFlags.

WORD wSizeDevice

The maximum size of the Sound Device name which can be returned.