

BASS_WASAPI_CheckFormat

Checks if a particular sample format is supported by a device.

```
DWORD BASS_WASAPI_CheckFormat(  
    int device,  
    DWORD freq,  
    DWORD chans,  
    DWORD flags  
);
```

Parameters

device The device to use... -1 = default output device, -2 = default input device, -3 = default loopback input device.

[BASS_WASAPI_GetDeviceInfo](#) can be used to enumerate the available devices.

freq The sample rate.

chans The number of channels... 1 = mono, 2 = stereo, etc.

flags Any combination of these flags.

BASS_WASAPI_EXCLUSIVE Check the device in exclusive mode, else shared mode. The HIWORD - use `MAKELONG(flags,format)` - can be used to limit the sample formats that are checked in exclusive mode. The default is to check 32-bit floating-point, 32-bit integer, 24-bit integer, 16-bit integer, 8-bit integer, in that order. A `BASS_WASAPI_FORMAT` value (see [BASS_WASAPI_INFO](#)) can be used to bypass the formats that precede it in that list.

Return value

If the sample format is supported, the maximum supported resolution (a BASS_WASAPI_FORMAT value) is returned, else -1 is returned. Use [BASS_ErrorGetCode](#) to get the error code.

Error codes

BASS_ERROR_WASAPI	WASAPI is not available.
BASS_ERROR_DEVICE	<i>device</i> is invalid.
BASS_ERROR_NOTAVAIL	Exclusive mode is unavailable on loopback devices.
BASS_ERROR_DRIVER	The driver could not be initialized.
BASS_ERROR_FORMAT	The specified format is not supported by the device.

Remarks

Shared and exclusive modes may have different sample formats available. Only the "mix format" (available from [BASS_WASAPI_GetDeviceInfo](#)) is generally supported in shared mode.

See also

[BASS_WASAPI_Init](#)

BASS_WASAPI_Free

Frees the device.

```
BOOL BASS_WASAPI_Free();
```

Return value

If successful, then TRUE is returned, else FALSE is returned. Use [BASS_ErrorGetCode](#) to get the error code.

Error codes

BASS_ERROR_INIT [BASS_WASAPI_Init](#) has not been successfully called.

Remarks

This function should be called for all initialized devices before the program closes. Freed devices do not need to have been stopped with [BASS_WASAPI_Stop](#) beforehand.

When using multiple devices, the current thread's device setting (as set with [BASS_WASAPI_SetDevice](#)) determines which device this function call applies to.

See also

[BASS_WASAPI_Init](#)

BASS_WASAPI_GetCPU

Retrieves the current CPU usage of BASSWASAPI.

```
float BASS_WASAPI_GetCPU();
```

Return value

The BASSWASAPI CPU usage as a percentage of total CPU time.

Remarks

This function includes the time taken by [WASAPIPROC](#) callback functions, but not [BASS_WASAPI_PutData](#) calls.

BASS_WASAPI_GetData

Retrieves the immediate sample data or an FFT representation of it.

```
DWORD BASS_WASAPI_GetData(  
    void *buffer,  
    DWORD length  
);
```

Parameters

buffer Pointer to a buffer to receive the data.

length Number of bytes wanted, and/or the [BASS_ChannelGetData](#) flags.

Return value

If an error occurs, -1 is returned, use [BASS_ErrorGetCode](#) to get the error code. When requesting FFT data, the number of bytes read from the device buffer (to perform the FFT) is returned. When requesting sample data, the number of bytes written to *buffer* will be returned. When using the BASS_DATA_AVAILABLE flag, the number of bytes in the device's buffer is returned.

Error codes

BASS_ERROR_INIT

[BASS_WASAPI_Init](#) has not been successfully called.

BASS_ERROR_NOTAVAIL

The BASS_WASAPI_BUFFER flag was not specified in the device's initialization.

Remarks

This function uses [BASS_ChannelGetData](#) internally, so it has the same options available.

The `BASS_WASAPI_BUFFER` flag needs to have been specified in the device's initialization to enable the use of this function, except for with the `BASS_DATA_AVAILABLE` flag.

With an output device, the `BASS_DATA_AVAILABLE` return value may be larger than the buffer size indicated by [BASS_WASAPI_GetInfo](#) due to additional latency in the device/driver. When a mixer is feeding an output device, the `BASS_DATA_AVAILABLE` return value can be used with the [BASS_ATTRIB_MIXER_LATENCY](#) attribute and/or the [BASS_Mixer_ChannelGetPositionEx](#) function, to have the mixer account for the latency in its source position reporting and data/level retrieval.

See also

[BASS_WASAPI_GetDeviceLevel](#), [BASS_WASAPI_GetLevel](#),
[BASS_WASAPI_GetLevelEx](#)

[BASS_ChannelGetData](#)

BASS_WASAPI_GetDevice

Retrieves the device setting of the current thread.

```
DWORD BASS_WASAPI_GetDevice();
```

Return value

If successful, the device number is returned, else -1 is returned. Use [BASS_ErrorGetCode](#) to get the error code.

Error codes

BASS_ERROR_INIT [BASS_WASAPI_Init](#) has not been successfully called;
there are no initialized devices.

See also

[BASS_WASAPI_Init](#), [BASS_WASAPI_SetDevice](#)

BASS_WASAPI_GetDeviceInfo

Retrieves information on a device.

```
BOOL BASS_WASAPI_GetDeviceInfo(  
    DWORD device,  
    BASS_WASAPI_DEVICEINFO *info  
);
```

Parameters

device The device to get the information of... 0 = first.

info Pointer to a structure to receive the information.

Return value

If successful, then TRUE is returned, else FALSE is returned. Use [BASS_ErrorGetCode](#) to get the error code.

Error codes

BASS_ERROR_WASAPI WASAPI is not available.

BASS_ERROR_DEVICE *device* is invalid.

Remarks

This function can be used to enumerate the available devices for a setup dialog.

Example

Get the total number of output devices currently present.

```
int a, count=0;
BASS_WASAPI_DEVICEINFO info;
for (a=0; BASS_WASAPI_GetDeviceInfo(a, &info;); a++)
    if (!(info.flags&BASS;_DEVICE_INPUT) // device is an output dev.
        && (info.flags&BASS;_DEVICE_ENABLED)) // and it is enab.
        count++; // count it
```

See also

[BASS_WASAPI_GetInfo](#), [BASS_WASAPI_Init](#), [BASS_WASAPI_SetNotify](#),
[BASS_WASAPI_DEVICEINFO](#) structure

BASS_WASAPI_GetDeviceLevel

Retrieves the level (peak amplitude) of a device.

```
float BASS_WASAPI_GetDeviceLevel(  
    DWORD device,  
    int chan  
);
```


Parameters

device The device to get the level from.

chan The channel to get the level of... 0 = first channel, -1 = all channels.

Return value

If successful, the level is returned, else -1 is returned. Use [BASS_ErrorGetCode](#) to get the error code.

Error codes

BASS_ERROR_WASAPI	WASAPI is not available.
BASS_ERROR_DEVICE	<i>device</i> is not valid.
BASS_ERROR_DRIVER	The device driver does not support level retrieval.
BASS_ERROR_ILLPARAM	<i>chan</i> is not valid.
BASS_ERROR_UNKNOWN	Some other mystery problem!

Remarks

This function gets the level from the device/driver, or WASAPI if the device does not have its own level meter. In the latter case, the level will be unavailable when exclusive mode is active.

See also

[BASS_WASAPI_GetDeviceInfo](#), [BASS_WASAPI_GetLevel](#)

BASS_WASAPI_GetInfo

Retrieves information on the device being used.

```
BOOL BASS_WASAPI_GetInfo(  
    BASS_WASAPI_INFO *info  
);
```

Parameters

info Pointer to a structure to receive the information.

Return value

If successful, TRUE is returned, else FALSE is returned. Use [BASS_ErrorGetCode](#) to get the error code.

Error codes

BASS_ERROR_INIT [BASS_WASAPI_Init](#) has not been successfully called.

Remarks

When using multiple devices, the current thread's device setting (as set with [BASS_WASAPI_SetDevice](#)) determines which device this function call applies to.

See also

[BASS_WASAPI_GetDeviceInfo](#), [BASS_WASAPI_INFO](#) structure

BASS_WASAPI_GetLevel

Retrieves the level (peak amplitude).

```
DWORD BASS_WASAPI_GetLevel();
```

Return value

If an error occurs, -1 is returned, use [BASS_ErrorGetCode](#) to get the error code. If successful, the level of the left channel is returned in the low word (low 16 bits), and the level of the right channel is returned in the high word (high 16 bits). If the channel is mono, then the low word is duplicated in the high word. The level ranges linearly from 0 (silent) to 32768 (max). 0 will be returned when a channel is stalled.

Error codes

BASS_ERROR_INIT	BASS_WASAPI_Init has not been successfully called.
BASS_ERROR_NOTAVAIL	The BASS_WASAPI_BUFFER flag was not specified in the device's initialization.

Remarks

This function uses [BASS_ChannelGetLevel](#) internally, so it behaves identically to that.

The BASS_WASAPI_BUFFER flag needs to have been specified in the device's initialization to enable the use of this function.

More flexible level retrieval is available with [BASS_WASAPI_GetLevelEx](#).

See also

[BASS_WASAPI_GetData](#), [BASS_WASAPI_GetDeviceLevel](#),
[BASS_WASAPI_GetLevelEx](#)

[BASS_ChannelGetLevel](#)

BASS_WASAPI_GetLevelEx

Retrieves the level.

```
BOOL BASS_WASAPI_GetLevelEx(  
    float *levels,  
    float length,  
    DWORD flags  
);
```

Parameters

- levels** An array to receive the levels.
- length** The amount of data to inspect to calculate the level, in seconds. The maximum is 1 second. Less data than requested may be used if the full amount is not available, eg. if the device's buffer is shorter.
- flags** A combination of these flags.
- BASS_LEVEL_MONO** Get a mono level. If neither this or the **BASS_LEVEL_STEREO** flag is used, then a separate level is retrieved for each channel; the number of channels is available from [BASS_WASAPI_GetInfo](#).
- BASS_LEVEL_STEREO** Get a stereo level. The left level will be from the even channels, and the right level will be from the odd channels. If there are an odd number of channels then the left and right levels will both include all channels.
- BASS_LEVEL_RMS** Get the RMS level. Otherwise the peak level.

Return value

If successful, TRUE is returned, else FALSE is returned. Use [BASS_ErrorGetCode](#) to get the error code.

Error codes

BASS_ERROR_INIT	<u>BASS_WASAPI_Init</u> has not been successfully called.
BASS_ERROR_NOTAVAIL	The BASS_WASAPI_BUFFER flag was not specified in the device's initialization.
BASS_ERROR_ILLPARAM	<i>length</i> is not valid.

Remarks

This function uses [BASS_ChannelGetLevelEx](#) internally, so it behaves identically to that.

The BASS_WASAPI_BUFFER flag needs to have been specified in the device's initialization to enable the use of this function.

See also

[BASS_WASAPI_GetData](#), [BASS_WASAPI_GetDeviceLevel](#),
[BASS_WASAPI_GetLevel](#)

[BASS_ChannelGetLevelEx](#)

BASS_WASAPI_GetMute

Retrieves the muted status of the device.

```
BOOL BASS_WASAPI_GetMute(  
    DWORD mode  
);
```

Parameters

mode The type of volume to get.

BASS_WASAPI_VOL_SESSION Get the session volume, else the device volume.

Return value

If successful, then TRUE or FALSE is returned to indicate the muted status, else -1 is returned. Use [BASS_ErrorGetCode](#) to get the error code.

Error codes

BASS_ERROR_INIT	<u>BASS_WASAPI_Init</u> has not been successfully called.
BASS_ERROR_NOTAVAIL	Volume control is unavailable.
BASS_ERROR_UNKNOWN	Some other mystery problem!

Remarks

When using multiple devices, the current thread's device setting (as set with [BASS_WASAPI_SetDevice](#)) determines which device this function call applies to.

See also

[BASS_WASAPI_SetMute](#)

BASS_WASAPI_GetVersion

Retrieves the version of BASSWASAPI that is loaded.

```
DWORD BASS_WASAPI_GetVersion();
```

Return value

The BASSWASAPI version. For example, 0x02040103 (hex), would be version 2.4.1.3

BASS_WASAPI_GetVolume

Retrieves the current volume level.

```
float BASS_WASAPI_GetVolume(  
    DWORD mode  
);
```

Parameters

mode	The type of volume to get and the curve to use.	
BASS_WASAPI_VOL_SESSION		Get the session volume, else the device volume.
BASS_WASAPI_CURVE_DB		Use a logarithmic curve. This is the default if no curve is specified.
BASS_WASAPI_CURVE_LINEAR		Use a linear curve.
BASS_WASAPI_CURVE_WINDOWS		Use Windows' hybrid curve, as used by Windows' volume controls.

Return value

If successful, the volume level is returned, else -1 is returned. Use [BASS_ErrorGetCode](#) to get the error code.

Error codes

BASS_ERROR_INIT	<u>BASS_WASAPI_Init</u> has not been successfully called.
BASS_ERROR_NOTAVAIL	Volume control is unavailable.
BASS_ERROR_UNKNOWN	Some other mystery problem!

Remarks

Session volume always uses the BASS_WASAPI_CURVE_WINDOWS curve.

When using multiple devices, the current thread's device setting (as set with [BASS_WASAPI_SetDevice](#)) determines which device this function call applies to.

See also

[BASS_WASAPI_SetVolume](#)

BASS_WASAPI_Init

Initializes a device.

```
BOOL BASS_WASAPI_Init(  
    int device,  
    DWORD freq,  
    DWORD chans,  
    DWORD flags,  
    float buffer,  
    float period,  
    WASAPIPROC *proc,  
    void *user  
);
```

Parameters

- device** The device to use... -1 = default output device, -2 = default input device, -3 = default loopback input device.
[BASS_WASAPI_GetDeviceInfo](#) can be used to enumerate the available devices.
- freq** The sample rate... 0 = "mix format" sample rate
- chans** The number of channels... 0 = "mix format" channels, 1 = mono, 2 = stereo, etc.
- flags** Any combination of these flags.
- BASS_WASAPI_AUTOFORMAT** Automatically choose another sample format if the specified format is not supported. If possible, a higher sample rate than *freq* will be used, rather than a lower one.
- BASS_WASAPI_BUFFER** Enable double buffering, for use by [BASS_WASAPI_GetData](#) and [BASS_WASAPI_GetLevel](#) and [BASS_WASAPI_GetLevelEx](#). This requires the BASS "no sound" device to have been initialized, via [BASS_Init](#).
- BASS_WASAPI_DITHER** Apply dither (TPDF) when converting floating-point sample data to the device's format. This flag only has effect on exclusive mode output.
- BASS_WASAPI_EVENT** Enable event-driven buffering. BASSWASAPI will normally periodically write data to (or read data from) the device's buffer according to the *period* parameter, but with the event-driven system WASAPI will signal to BASSWASAPI when

more data should be written to (or read from) the buffer. So the *period* parameter is ignored, and *buffer* is too in shared mode as the system will choose an appropriate buffer length. In exclusive mode, there are 2 buffers of *buffer* length that are processed alternately. Event-driven buffering is unavailable on loopback devices.

BASS_WASAPI_EXCLUSIVE

Initialize the device in exclusive mode, else shared mode. The HIWORD - use MAKELONG(flags,format) - can be used to limit the sample format that is used in exclusive mode. The default is to try 32-bit floating-point, 32-bit integer, 24-bit integer, 16-bit integer, 8-bit integer, in that order. A BASS_WASAPI_FORMAT value (see [BASS_WASAPI_INFO](#)) can be used to bypass the formats that precede it in that list. Exclusive mode is unavailable on loopback devices.

BASS_WASAPI_SAMPLES

buffer and *period* are in samples rather than seconds.

buffer The length of the device's buffer in seconds or samples, depending on BASS_WASAPI_SAMPLES. This is a minimum and the driver may choose to use a larger buffer; [BASS_WASAPI_GetInfo](#) can be used to confirm what the buffer size is. For an output device, the buffer size determines the latency. With event-driven exclusive mode, there will be 2 buffers of this length, so the total buffer length is double.

- period The interval (in seconds or samples depending on BASS_WASAPI_SAMPLES) between callback function calls... 0 = use default. If the specified period is below the minimum update period, it will automatically be raised to that. This is ignored when the BASS_WASAPI_EVENT flag is specified, except in shared mode when *buffer = 0* on Windows 10 (see remarks).
- proc The callback function to provide/receive the sample data... NULL = use [BASS_WASAPI_PutData](#) to feed the output. This cannot be NULL when the BASS_WASAPI_EVENT flag is specified.
- user User instance data to pass to the callback function.

Return value

If the device was successfully initialized, TRUE is returned, else FALSE is returned. Use [BASS_ErrorGetCode](#) to get the error code.

Error codes

BASS_ERROR_WASAPI	WASAPI is not available.
BASS_ERROR_DEVICE	<i>device</i> is invalid.
BASS_ERROR_ALREADY	The device has already been initialized. BASS_WASAPI_Free must be called before it can be initialized again.
BASS_ERROR_ILLPARAM	A WASAPIPROC must be provided for an input device or when using event-driven buffering.
BASS_ERROR_NOTAVAIL	Event-driven buffering and exclusive mode are unavailable on loopback devices.
BASS_ERROR_DRIVER	The driver could not be initialized.
BASS_ERROR_FORMAT	The specified format is not supported by the device. If the BASS_WASAPI_AUTOFORMAT flag was specified, no other format could be found either.
BASS_ERROR_BUSY	The device is already in use, eg. another process may have initialized it in exclusive mode.
BASS_ERROR_INIT	The BASS "no sound" device has not been initialized.
BASS_ERROR_WASAPI_BUFFER	<i>buffer</i> is too large or small (exclusive mode only).
BASS_ERROR_UNKNOWN	Some other mystery problem!

Remarks

For convenience, devices are always initialized to use their highest sample resolution (unless restricted by *flags*) and that is then converted to 32-bit floating-point, so that [WASAPIPROC](#) callback functions and the [BASS_WASAPI_PutData](#) and [BASS_WASAPI_GetData](#) functions are always dealing with the same sample format. The device's sample format can be obtained from [BASS_WASAPI_GetInfo](#).

WASAPI does not support arbitrary sample formats, like DirectSound does. In particular, only the "mix format" (available from [BASS_WASAPI_GetDeviceInfo](#)) is generally supported in shared mode. [BASS_WASAPI_CheckFormat](#) can be used to check whether a particular sample format is supported. The [BASSmix](#) add-on can be used to play (or record) in otherwise unsupported sample formats, as well as playing multiple sources.

A loopback device can only be used when the corresponding output device is not being used in exclusive mode, and it will only deliver data when the output device does; if the output device produces no data, then the loopback device will capture no data.

Shared mode usually has a fixed period of 10ms, but Windows 10 supports shorter periods, which allows smaller buffers and lower latency. A shorter period (and buffer) can be requested by setting *buffer* to 0 and *period* to the length wanted. If the requested period is lower than the device (or Windows) supports, then it will be automatically raised to the minimum supported. It will also be rounded up if it does not match the device's granularity. The actual period in use can be determined from the (minimum) amount of data that gets requested from the [WASAPIPROC](#) callback function. The shared mode period is a system-wide setting that affects all users of the device, particular those using event-driven buffering; they will be asked to provide data at the new period. If another process is already using a non-default period, then it will not be possible to set a different period until they finish; the existing period will have to be used in the meantime.

The initialized device will not begin processing data until [BASS_WASAPI_Start](#) is called.

Simultaneously using multiple devices is supported in the BASS API via a context switching system; instead of there being an extra "device" parameter in the function calls, the device to be used is set prior to calling the functions. [BASS_WASAPI_SetDevice](#) is used to switch the current device. When successful, BASS_WASAPI_Init automatically sets the current thread's device to the one that was just initialized.

When using the default output or input device, [BASS_WASAPI_GetDevice](#) can be used to find out which device it was mapped to.

Example

Initialize BASSWASAPI to use the default output device in exclusive mode, with 44100 Hz stereo output, and a 100ms buffer with the default period.

```
BASS_WASAPI_Init(-1, 44100, 2, BASS_WASAPI_EXCLUSIVE, 0.1, 0, MyWas:
```

See also

[BASS_WASAPI_CheckFormat](#), [BASS_WASAPI_Free](#),
[BASS_WASAPI_GetDeviceInfo](#), [BASS_WASAPI_GetInfo](#),
[BASS_WASAPI_Start](#), [WASAPIPROC](#) callback

BASS_WASAPI_IsStarted

Checks if processing has been started.

```
BOOL BASS_WASAPI_IsStarted();
```

Return value

If the device has been started, then TRUE is returned, else FALSE is returned.

See also

[BASS WASAPI Start](#), [BASS WASAPI Stop](#)

BASS_WASAPI_Lock

Locks the device to the current thread.

```
BOOL BASS_WASAPI_Lock(  
    BOOL lock  
);
```

Parameters

lock If FALSE, unlock the device, else lock it.

Return value

If successful, TRUE is returned, else FALSE is returned. Use [BASS_ErrorGetCode](#) to get the error code.

Error codes

BASS_ERROR_INIT [BASS_WASAPI_Init](#) has not been successfully called.

Remarks

Locking a device prevents other threads from accessing the device buffer, including a [WASAPIPROC](#). Other threads wanting to access a locked device will block until it is unlocked, so a device should only be locked very briefly. A device must be unlocked in the same thread that it was locked.

BASS_WASAPI_PutData

Adds sample data to an output device buffer.

```
DWORD BASS_WASAPI_PutData(  
    void *buffer,  
    DWORD length  
);
```

Parameters

buffer Pointer to the sample data.

length The amount of data in bytes.

Return value

If successful, the amount of data used is returned, else -1 is returned. Use [BASS_ErrorGetCode](#) to get the error code.

Error codes

BASS_ERROR_INIT	BASS_WASAPI_Init has not been successfully called.
BASS_ERROR_NOTAVAIL	The device is being fed by a WASAPIPROC callback function, or it is an input device.
BASS_ERROR_ILLPARAM	<i>length</i> is not valid, it must equate to a whole number of samples.
BASS_ERROR_UNKNOWN	Some other mystery problem!

Remarks

As much data as possible will be placed in the device's buffer; this function will have to be called again for any remainder.

Data should be provided at a rate sufficient to sustain playback. If the buffer gets exhausted, output will stall until more data is provided.

[BASS_WASAPI_GetData](#) (BASS_DATA_AVAILABLE) can be used to check how much data is buffered.

See also

[BASS_WASAPI_Init](#), [WASAPIPROC callback](#)

BASS_WASAPI_SetDevice

Sets the device to use for subsequent calls in the current thread.

```
BOOL BASS_WASAPI_SetDevice(  
    DWORD device  
);
```

Parameters

device The device to use.

Return value

If successful, then TRUE is returned, else FALSE is returned. Use [BASS_ErrorGetCode](#) to get the error code.

Error codes

BASS_ERROR_DEVICE *device* is invalid.

BASS_ERROR_INIT The device has not been initialized.

Remarks

Simultaneously using multiple devices is supported in the BASS API via a context switching system; instead of there being an extra "device" parameter in the function calls, the device to be used is set prior to calling the functions. The device setting is local to the current thread, so calling functions with different devices simultaneously in multiple threads is not a problem.

All of the BASSWASAPI functions that do not have their own "device" parameter make use of this device selection. When one of them is called, BASSWASAPI will check the current thread's device setting, and if no device is selected (or the selected device is not initialized), BASSWASAPI will automatically select the lowest device that is initialized. This means that when using a single device, there is no need to use this function; BASSWASAPI will automatically use the device that is initialized. Even if you free the device, and initialize another, BASSWASAPI will automatically switch to the one that is initialized.

See also

[BASS_WASAPI_GetDevice](#), [BASS_WASAPI_Init](#)

BASS_WASAPI_SetMute

Mutes or unmutes the device.

```
BOOL BASS_WASAPI_SetMute(  
    DWORD mode,  
    BOOL mute  
);
```

Parameters

mode The type of volume to set.

BASS_WASAPI_VOL_SESSION Set the session volume, else the device volume.

mute Mute the device?

Return value

If successful, then TRUE is returned, else FALSE is returned. Use [BASS_ErrorGetCode](#) to get the error code.

Error codes

BASS_ERROR_INIT	<u>BASS_WASAPI_Init</u> has not been successfully called.
BASS_ERROR_NOTAVAIL	Volume control is unavailable.
BASS_ERROR_UNKNOWN	Some other mystery problem!

Remarks

When using multiple devices, the current thread's device setting (as set with [BASS_WASAPI_SetDevice](#)) determines which device this function call applies to.

See also

[BASS_WASAPI_GetMute](#), [BASS_WASAPI_SetVolume](#)

BASS_WASAPI_SetNotify

Sets a device change notification callback.

```
BOOL BASS_WASAPI_SetNotify(  
    WASAPINOTIFYPROC *proc,  
    void *user  
);
```

Parameters

proc User defined notification function... NULL = disable notifications.

user User instance data to pass to the callback function.

Return value

If successful, TRUE is returned, else FALSE is returned. Use [BASS_ErrorGetCode](#) to get the error code.

Error codes

BASS_ERROR_WASAPI WASAPI is not available.

Remarks

A previously set notification callback can be changed or removed at any time by calling this function again.

If the BASSWASAPI DLL is loaded dynamically (eg. via LoadLibrary), this function should be called with NULL parameters prior to unloading the DLL.

See also

[WASAPINOTIFYPROC callback](#)

BASS_WASAPI_SetVolume

Sets the device volume.

```
BOOL BASS_WASAPI_SetVolume(  
    DWORD mode,  
    float volume  
);
```

Parameters

mode	The type of volume to set and the curve to use.	
	BASS_WASAPI_VOL_SESSION	Set the session volume, else the device volume.
	BASS_WASAPI_CURVE_DB	Use a logarithmic curve. This is the default if no curve is specified.
	BASS_WASAPI_CURVE_LINEAR	Use a linear curve.
	BASS_WASAPI_CURVE_WINDOWS	Use Windows' hybrid curve, as used by Windows' volume controls.
volume	The volume level... 0 (silent) to 1 (max) if using the linear or Windows curves, else a dB level. The device's valid dB level range can be obtained from BASS_WASAPI_GetInfo .	

Return value

If successful, then TRUE is returned, else FALSE is returned. Use [BASS_ErrorGetCode](#) to get the error code.

Error codes

BASS_ERROR_INIT	<u>BASS_WASAPI_Init</u> has not been successfully called.
BASS_ERROR_NOTAVAIL	Volume control is unavailable.
BASS_ERROR_ILLPARAM	<i>volume</i> is invalid.
BASS_ERROR_UNKNOWN	Some other mystery problem!

Remarks

Session volume only affects the current process, so other users of the device are unaffected. It has no effect on exclusive mode output, and maps to the device volume with input devices (so does affect other users). Session volume always uses the `BASS_WASAPI_CURVE_WINDOWS` curve.

When the `BASS_WASAPI_CURVE_LINEAR` curve is used, the resulting volume level may not be exactly as requested because it gets translated to a dB value within the device's valid dB level range, which is available from [`BASS_WASAPI_GetInfo`](#).

When using multiple devices, the current thread's device setting (as set with [`BASS_WASAPI_SetDevice`](#)) determines which device this function call applies to.

See also

[BASS_WASAPI_GetVolume](#), [BASS_WASAPI_SetMute](#)

BASS_WASAPI_Start

Starts the device.

```
BOOL BASS_WASAPI_Start();
```

Return value

If successful, then TRUE is returned, else FALSE is returned. Use [BASS_ErrorGetCode](#) to get the error code.

Error codes

BASS_ERROR_INIT [BASS_WASAPI_Init](#) has not been successfully called.

BASS_ERROR_UNKNOWN Some other mystery problem!

Remarks

When using multiple devices, the current thread's device setting (as set with [BASS_WASAPI_SetDevice](#)) determines which device this function call applies to.

See also

[BASS_WASAPI_IsStarted](#), [BASS_WASAPI_Stop](#)

BASS_WASAPI_Stop

Stops the device.

```
BOOL BASS_WASAPI_Stop(  
    BOOL reset  
);
```

Parameters

reset Flush the device buffer?

Return value

If successful, then TRUE is returned, else FALSE is returned. Use [BASS_ErrorGetCode](#) to get the error code.

Error codes

BASS_ERROR_INIT [BASS_WASAPI_Init](#) has not been successfully called.

BASS_ERROR_UNKNOWN Some other mystery problem!

Remarks

If the device buffer is left unflushed (*reset=FALSE*), a subsequent [BASS_WASAPI_Start](#) call will resume things with the buffered data, otherwise it will resume with fresh data.

Exclusive mode output should generally be flushed when stopped to avoid glitches upon resumption.

When using multiple devices, the current thread's device setting (as set with [BASS_WASAPI_SetDevice](#)) determines which device this function call applies to.

See also

[BASS_WASAPI_IsStarted](#), [BASS_WASAPI_Start](#)

WASAPINOTIFYPROC callback

User defined notification callback function.

```
void CALLBACK WasapiNotifyProc(  
    DWORD notify,  
    DWORD device,  
    void *user  
);
```


Parameters

notify	The notification, one of the following.	
	BASS_WASAPI_NOTIFY_ENABLED	The device has been enabled.
	BASS_WASAPI_NOTIFY_DISABLED	The device has been disabled/disconnected.
	BASS_WASAPI_NOTIFY_DEFINPUT	The device is now the default input device.
	BASS_WASAPI_NOTIFY_DEFOUTPUT	The device is now the default output device.
	BASS_WASAPI_NOTIFY_FAIL	The device has failed and been stopped. If the device is still enabled and shared mode was being used, then it may be that the device's sample format has changed. It can be freed and reinitialized, with BASS_WASAPI_Free and BASS_WASAPI_Init , to resume in that case.
device	The device that the notification applies to.	
user	The user instance data given when BASS_WASAPI_SetNotify was called.	

Remarks

BASS_WASAPI_Free should not be called from within a WASAPINOTIFYPROC callback.

See also

[BASS_WASAPI_GetDeviceInfo](#), [BASS_WASAPI_SetNotify](#)

WASAPIPROC callback

User defined output/input processing callback function.

```
DWORD CALLBACK WasapiProc(  
    void *buffer,  
    DWORD length,  
    void *user  
);
```

Parameters

- buffer Pointer to the buffer to put the sample data for an output device, or to get the data from an input device. The sample data is always 32-bit floating-point.
- length The number of bytes to process.
- user The user instance data given when [BASS_WASAPI_Init](#) was called.

Return value

In the case of an output device, the number of bytes written to the buffer. If the value is negative (high bit set), it will be treated as 0. In the case of an input device, 0 = stop the device, else continue.

Remarks

An output/input processing function should obviously be as quick as possible, to avoid buffer underruns (output) or overruns (input). Using a larger buffer makes that less crucial. [BASS_WASAPI_GetData](#) (BASS_DATA_AVAILABLE) can be used to check how much data is buffered.

If an output device has been initialized to use exclusive mode and less data than requested is returned, the remainder of the buffer will be filled with silence.

Do not call [BASS_WASAPI_Free](#) from within a callback function.

[BASS_WASAPI_GetDevice](#) can be used by the callback function to check which device it is dealing with.

Example

Feed a BASS decoding channel to an output device, and stop the device at the end.

```
DWORD CALLBACK OutputWasapiProc(void *buffer, DWORD length, void *u:
{
    int c=BASS_ChannelGetData(decoder, buffer, length);
    if (c<0) { // at the end
        if (!BASS_WASAPI_GetData(NULL, BASS_DATA_AVAILABLE)) // che
            BASS_WASAPI_Stop(FALSE); // stop the output
        return 0;
    }
    return c;
}
```


See also

[BASS_WASAPI_Init](#), [BASS_WASAPI_PutData](#)

BASS_WASAPI_DEVICEINFO structure

Used with [BASS_WASAPI_GetDeviceInfo](#) to retrieve information on a device.

```
typedef struct {
    char *name;
    char *id;
    DWORD type;
    DWORD flags;
    float minperiod;
    float defperiod;
    DWORD mixfreq;
    DWORD mixchans;
} BASS_WASAPI_DEVICEINFO;
```

Members

name	Description of the device.
id	The device's ID.
type	The type of device, which may be one of the following.
BASS_WASAPI_TYPE_NETWORKDEVICE	An audio endpoint device that the user accesses remotely through a network.
BASS_WASAPI_TYPE_SPEAKERS	A set of speakers.
BASS_WASAPI_TYPE_LINELEVEL	An audio endpoint device that sends a line-level analog signal to a line-input jack on an audio adapter or that receives a line-level analog signal from a line-output jack on the adapter.
BASS_WASAPI_TYPE_HEADPHONES	A set of headphones.
BASS_WASAPI_TYPE_MICROPHONE	A microphone.
BASS_WASAPI_TYPE_HEADSET	An earphone or a pair of earphones with an attached mouthpiece for two-way communication.
BASS_WASAPI_TYPE_HANDSET	The part of a telephone that is held in the hand

BASS_WASAPI_TYPE_DIGITAL

and that contains a speaker and a microphone for two-way communication.

An audio endpoint device that connects to an audio adapter through a connector for a digital interface of unknown type.

BASS_WASAPI_TYPE_SPDIF

An audio endpoint device that connects to an audio adapter through a Sony/Philips Digital Interface (S/PDIF) connector.

BASS_WASAPI_TYPE_HDMI

An audio endpoint device that connects to an audio adapter through a High-Definition Multimedia Interface (HDMI) connector or a DisplayPort.

BASS_WASAPI_TYPE_UNKNOWN

An audio endpoint device with unknown physical

attributes.

flags	The device's current and input/output status... a combination of these flags.
BASS_DEVICE_ENABLED	The device is enabled and ready for use. It will not be possible to initialize the device if this flag is not present.
BASS_DEVICE_DEFAULT	The device is the system default.
BASS_DEVICE_INIT	The device is initialized, ie. BASS_WASAPI_Init has been called.
BASS_DEVICE_INPUT	The device is a recording device, otherwise it is an output device.
BASS_DEVICE_LOOPBACK	The device is a loopback input device; it captures the sound from an output device.
BASS_DEVICE_UNPLUGGED	The device does not have an audio jack connected. This is only applicable to devices that have jack-presence detection.
BASS_DEVICE_DISABLED	The device has been disabled in the Sound control panel.
minperiod	The minimum update period in seconds.
defperiod	The default update period in seconds.
mixfreq	The sample rate in shared mode.
mixchans	The number of channels in shared mode.

Remarks

If none of the `BASS_DEVICE_ENABLED`, `BASS_DEVICE_DISABLED`, or `BASS_DEVICE_UNPLUGGED` flags are present, then the device is not present. That could be because it is an unplugged USB device or it has been disabled in Device Manager, for example.

Depending on the `BASS_CONFIG_UNICODE` config setting, *name* and *id* can be in ANSI or UTF-8 form.

The corresponding BASS (DirectSound) device can be found by its `BASS_DEVICEINFO` "driver" member matching *id*.

See also

[BASS_WASAPI_GetDeviceInfo](#)

BASS_WASAPI_INFO structure

Used with [BASS_WASAPI_GetInfo](#) to retrieve information on the current device.

```
typedef struct {  
    DWORD initflags;  
    DWORD freq;  
    DWORD chans;  
    DWORD format;  
    DWORD buflen;  
    DWORD volmax;  
    DWORD volmin;  
    DWORD volstep;  
} BASS_WASAPI_INFO;
```


Members

initflags	The <i>flags</i> parameter of the BASS_WASAPI_Init call.
freq	The sample rate.
chans	The number of channels... 1 = mono, 2 = stereo, etc.
format	The device's sample format. One of the following. BASS_WASAPI_FORMAT_8BIT 8-bit integer. BASS_WASAPI_FORMAT_16BIT 16-bit integer. BASS_WASAPI_FORMAT_24BIT 24-bit integer. BASS_WASAPI_FORMAT_32BIT 32-bit integer. BASS_WASAPI_FORMAT_FLOAT 32-bit floating-point.
buflen	The buffer size in bytes.
volmax	The maximum volume setting in dB.
volmin	The minimum volume setting in dB.
volstep	The volume step size in dB.

Remarks

format indicates the device's sample format, which is not necessarily the same as what a [WASAPIPROC](#) callback function or the [BASS_WASAPI_PutData](#) and [BASS_WASAPI_GetData](#) functions deal in; that is always 32-bit floating-point.

volmin and *volmax* indicate the valid device volume range for [BASS_WASAPI_SetVolume](#) and [BASS_WASAPI_GetVolume](#) when using the logarithmic curve.

See also

[BASS_WASAPI_GetInfo](#)