

# MBS Real Studio Audio Plugin Documentation

Christian Schmitz

May 15, 2012

## 0.1 Introduction

This is the PDF version of the documentation for the Real Studio Plug-in from Monkeybread Software Germany. Plugin part: MBS Real Studio Audio Plugin

## 0.2 Content

- 1 List of all topics 3
- 2 All items in this plugin 11
- 4 List of all classes 105

# Chapter 1

## List of Topics

• 2 Audio	11
– 2.2 class PortAudioStreamBaseMBS	22
* 2.2.1 Abort as integer	22
* 2.2.1 Close as integer	22
* 2.2.1 CPULoad as double	22
* 2.2.1 HostError as PortAudioHostErrorInfoMBS	23
* 2.2.1 Info as PortAudioStreamInfoMBS	23
* 2.2.1 IsStreamActive as integer	23
* 2.2.1 IsStreamStopped as integer	24
* 2.2.1 Start as integer	24
* 2.2.1 Stop as integer	24
* 2.2.1 Time as double	24
* 2.2.1 UseSafeThreading as boolean	25
– 2.1 class PortAudioStreamBufferedMBS	11
* 2.1.1 AddAudio(Data as memoryblock, offsetBytes as integer=0, countBytes as integer=0, BitSize as integer=16, ClearBuffers as boolean=false) as boolean	11
* 2.1.1 AddAudioStereo(Data1 as memoryblock, Data2 as memoryblock, offsetBytes as integer=0, countBytes as integer=0, BitSize as integer=16, ClearBuffers as boolean=false) as boolean	12
* 2.1.1 AddFloatAudio(FloatData as memoryblock, offsetBytes as integer=0, countBytes as integer=0, ClearBuffers as boolean=false) as boolean	13
* 2.1.1 AddFloatAudioStereo(FloatData1 as memoryblock, FloatData2 as memoryblock, offsetBytes as integer=0, countBytes as integer=0, ClearBuffers as boolean=false) as boolean	14
* 2.1.1 FreeSpace as integer	15
* 2.1.1 HasFreeSpace as boolean	15

* 2.1.1	OpenDefaultStream(numOutputChannels as integer, sampleRate as double) as integer	15
* 2.1.1	OpenStream(outputParameters as PortAudioStreamParametersMBS, sampleRate as double, framesPerBuffer as integer, streamFlags as integer) as integer	16
* 2.1.1	PlayAudio(Data as memoryblock, offsetBytes as integer=0, countBytes as integer=0, BitSize as integer=16, ClearBuffers as boolean=false) as boolean	17
* 2.1.1	PlayAudioStereo(Data1 as memoryblock, Data2 as memoryblock, offsetBytes as integer=0, countBytes as integer=0, BitSize as integer=16, ClearBuffers as boolean=false) as boolean	18
* 2.1.1	PlayFloatAudio(FloatData as memoryblock, offsetBytes as integer=0, countBytes as integer=0, ClearBuffers as boolean=false) as boolean	19
* 2.1.1	PlayFloatAudioStereo(FloatData1 as memoryblock, FloatData2 as memoryblock, offsetBytes as integer=0, countBytes as integer=0, ClearBuffers as boolean=false) as boolean	19
* 2.1.2	HadUnderflow as Boolean	20
* 2.1.2	NoUnderflow as Boolean	20
* 2.1.2	OutputPosition as Double	21
* 2.1.2	OutputPositionRelative as Double	21
* 2.1.3	Finished	21
– 2.7	class PortAudioMBS	37
* 2.7.1	CountDevices as integer	38
* 2.7.1	DefaultHostApiIndexd as integer	38
* 2.7.1	DefaultInputDeviceID as integer	39
* 2.7.1	DefaultOutputDeviceID as integer	40
* 2.7.1	DeviceInfo(DeviceIndex as integer) as PortAudioDeviceInfoMBS	40
* 2.7.1	DisableHostAPI(API as string) as boolean	41
* 2.7.1	ErrorText(ErrorNumber as integer) as string	42
* 2.7.1	GetSampleSize(Format as integer) as integer	42
* 2.7.1	HostApiCount as integer	42
* 2.7.1	HostApiDeviceIndexToDeviceIndex(hostApiIndex as integer, hostApiDeviceIndex as integer) as integer	43
* 2.7.1	HostApiInfo(hostApiIndex as integer) as PortAudioHostApiInfoMBS	43
* 2.7.1	HostApiTypeIdToHostApiIndex(type as integer) as integer	44
* 2.7.1	HostError as PortAudioHostErrorInfoMBS	44
* 2.7.1	IsFormatSupported(input as PortAudioStreamParametersMBS, output as PortAudioStreamParametersMBS, sampleRate as double) as integer	45
* 2.7.1	SampleSize(theFormat as integer) as integer	45
* 2.7.1	Sleep(msec as integer)	46
* 2.7.1	Version as integer	46
* 2.7.1	VersionText as String	46
– 2.3	class PortAudioStreamInfoMBS	25
* 2.3.1	Constructor	25
* 2.3.2	InputLatency as Double	26

* 2.3.2 OutputLatency as Double	26
* 2.3.2 SampleRate as Double	26
– 2.4 class PortAudioStreamMBS	27
* 2.4.1 OpenDefaultStream(numInputChannels as integer, numOutputChannels as integer, sampleFormat as integer, sampleRate as double, framesPerBuffer as integer, Flags as integer) as integer	27
* 2.4.1 OpenStream(inputParameters as PortAudioStreamParametersMBS, outputParameters as PortAudioStreamParametersMBS, sampleRate as double, framesPerBuffer as integer, streamFlags as integer) as integer	28
* 2.4.1 Read(buffer as memoryblock, frameCount as integer) as integer	29
* 2.4.1 ReadAvailable as integer	30
* 2.4.1 Write(buffer as memoryblock, frameCount as integer) as integer	30
* 2.4.1 WriteAvailable as integer	31
* 2.4.2 Callback(InputBuffer as memoryblock, outputBuffer as memoryblock, FrameCount as integer, inputBufferAdcTime as double, currentTime as double, outputBufferDacTime as double, statusFlags as integer) as integer	31
* 2.4.2 Finished	33
– 2.6 class PortAudioHostApiInfoMBS	34
* 2.6.1 Constructor	35
* 2.6.2 defaultInputDevice as Integer	35
* 2.6.2 defaultOutputDevice as Integer	35
* 2.6.2 deviceCount as Integer	35
* 2.6.2 Index as Integer	36
* 2.6.2 Name as String	36
* 2.6.2 Type as Integer	36
– 2.5 class PortAudioHostErrorInfoMBS	33
* 2.5.1 Constructor	33
* 2.5.2 ErrorCode as Integer	34
* 2.5.2 ErrorText as String	34
* 2.5.2 HostApiType as Integer	34
– 2.10 class PortAudioDeviceInfoMBS	54
* 2.10.1 Constructor	54
* 2.10.2 DefaultHighInputLatency as Double	54
* 2.10.2 DefaultHighOutputLatency as Double	55
* 2.10.2 DefaultLowInputLatency as Double	55
* 2.10.2 DefaultLowOutputLatency as Double	55
* 2.10.2 DefaultSampleRate as Double	55
* 2.10.2 HostApiIndex as Integer	55
* 2.10.2 Index as Integer	56
* 2.10.2 MaxInputChannels as integer	56
* 2.10.2 MaxOutputChannels as integer	56

* 2.10.2 Name as String	56
– 2.8 class PortAudioStreamRecorderMBS	47
* 2.8.1 Constructor(BufferSize as integer)	47
* 2.8.1 Flush	47
* 2.8.1 OpenDefaultStream(numInputChannels as integer, sampleRate as double) as integer	48
* 2.8.1 OpenStream(inputParameters as PortAudioStreamParametersMBS, sampleRate as double, framesPerBuffer as integer, streamFlags as integer) as integer	48
* 2.8.1 ReadFrames(mem as memoryblock, SizeInBytes as integer) as integer	49
* 2.8.1 ResizeBuffer(BufferSize as integer)	50
* 2.8.2 Buffer as Memoryblock	50
* 2.8.2 BufferReadIndex as Integer	51
* 2.8.2 BufferSize as Integer	51
* 2.8.2 BufferWriteIndex as Integer	51
* 2.8.2 FramesAvailable as Integer	51
* 2.8.2 NumInputChannels as Integer	52
– 2.9 class PortAudioStreamParametersMBS	52
* 2.9.1 ChannelCount as Integer	52
* 2.9.1 Device as Integer	52
* 2.9.1 SampleFormat as Integer	53
* 2.9.1 SuggestedLatency as Double	53
• 3 MIDI	59
– 3.1 class PortMidiStreamMBS	59
* 3.1.1 Abort as integer	59
* 3.1.1 Close	60
* 3.1.1 ErrorText(ErrorNumber as integer) as string	60
* 3.1.1 OpenInput(DeviceID as integer, Buffersize as integer) as integer	60
* 3.1.1 OpenOutput(DeviceID as integer, Buffersize as integer, Latency as integer) as integer	61
* 3.1.1 Poll as integer	61
* 3.1.1 Read(byref data as PortMidiEventMBS) as integer	61
* 3.1.1 SetChannelMask(mask as integer) as integer	62
* 3.1.1 SetFilter(filters as integer) as integer	63
* 3.1.1 Write(data as PortMidiEventMBS) as integer	63
* 3.1.1 WriteShort(When as integer, message as integer) as integer	64
* 3.1.1 WriteSysEx(When as integer, message as memoryblock, offset as integer) as integer	64
* 3.1.1 WriteSysEx(When as integer, message as string) as integer	64
* 3.1.2 ChannelMask as Integer	64
* 3.1.2 Filters as Integer	65
* 3.1.3 FilterActive = & h4000	65
* 3.1.3 FilterAftertouch = & h6000000	65

* 3.1.3 FilterChannelAftertouch = & h20000000	65
* 3.1.3 FilterClock & h1D00	66
* 3.1.3 FilterControl & h8000000	66
* 3.1.3 FilterFD = & h2000	66
* 3.1.3 FilterMTC = 2	66
* 3.1.3 FilterNote = & h3000000	66
* 3.1.3 FilterPitchBend = & h40000000	67
* 3.1.3 FilterPlay = & h400	67
* 3.1.3 FilterPolyAftertouch = & h4000000	67
* 3.1.3 FilterProgram = & h10000000	67
* 3.1.3 FilterRealTime = & hFF01	67
* 3.1.3 FilterReset = & h8000	68
* 3.1.3 FilterSongPosition = 4	68
* 3.1.3 FilterSongSelect = 8	68
* 3.1.3 FilterSysEx = 1	68
* 3.1.3 FilterSystemCommon = & h4E	68
* 3.1.3 FilterTick = & h200	69
* 3.1.3 FilterTune = & h40	69
* 3.1.3 FilterUndefined = & h2000	69
– 3.3 class PortMidiMBS	70
* 3.3.1 CountDevices as integer	71
* 3.3.1 DefaultInputDeviceID as integer	71
* 3.3.1 DefaultOutputDeviceID as integer	72
* 3.3.1 DeviceInfo(DeviceID as integer) as PortMidiDeviceInfoMBS	73
* 3.3.1 ErrorText(ErrorNumber as integer) as string	74
* 3.3.1 ReInitialize as integer	74
* 3.3.2 pmBadData = -9994	74
* 3.3.2 pmBadPtr = -9995	74
* 3.3.2 pmBufferMaxSize = -9992	75
* 3.3.2 pmBufferOverflow = -9996	75
* 3.3.2 pmBufferTooSmall = -9997	75
* 3.3.2 pmHostError = -10000	75
* 3.3.2 pmInsufficientMemory = -9998	75
* 3.3.2 pmInternalError = -9993	75
* 3.3.2 pmInvalidDeviceId = -9999	76
* 3.3.2 pmNoDevice = -1	76
* 3.3.2 pmNoError = 0	76
– 3.2 class PortMidiDeviceInfoMBS	69
* 3.2.1 HasInput as Boolean	69
* 3.2.1 HasOutput as Boolean	70
* 3.2.1 InterfaceName as String	70

* 3.2.1 Name as String	70
– 3.4 class PortMidiEventMBS	76
* 3.4.1 Set(status as integer, data1 as integer, data2 as integer)	76
* 3.4.1 SetRaw(data0 as integer, data1 as integer, data2 as integer, data3 as integer)	77
* 3.4.2 Data1 as Integer	77
* 3.4.2 Data2 as Integer	77
* 3.4.2 RawData0 as Integer	77
* 3.4.2 RawData1 as Integer	77
* 3.4.2 RawData2 as Integer	78
* 3.4.2 RawData3 as Integer	78
* 3.4.2 RawMessage as Integer	78
* 3.4.2 Status as Integer	78
* 3.4.2 When as Integer	79
– 3.5 class WindowsMidiOutputMBS	79
* 3.5.1 Close	79
* 3.5.1 Open(DeviceID as integer)	79
* 3.5.1 OpenDefault	80
* 3.5.1 OutputErrorText(errorcode as integer) as string	80
* 3.5.1 Reset	80
* 3.5.1 SendData(data as memoryblock, size as integer)	80
* 3.5.1 SendData(data as string)	81
* 3.5.1 SendMessage(message as integer)	81
* 3.5.1 SendMessage(status as integer, data1 as integer, data2 as integer)	82
* 3.5.1 Volume as integer	82
* 3.5.2 DeviceClose	83
* 3.5.2 DeviceDataSent	83
* 3.5.2 DeviceOpen	84
* 3.5.2 DevicePositionCallback	84
– 3.9 class WindowsMidiOutputInfoMBS	94
* 3.9.1 ChannelMask as Integer	95
* 3.9.1 DriverVersion as Integer	95
* 3.9.1 Flags as Integer	95
* 3.9.1 ManufacturerID as Integer	96
* 3.9.1 Name as String	96
* 3.9.1 Notes as Integer	96
* 3.9.1 ProductID as Integer	96
* 3.9.1 Technology as Integer	96
* 3.9.1 Voices as Integer	97
* 3.9.1 Volume as Boolean	97
* 3.9.1 VolumeStereo as Boolean	97
– 3.6 class WindowsMidiStreamMBS	84

* 3.6.1 Close	84
* 3.6.1 Open(DeviceID as integer)	84
* 3.6.1 Pause	85
* 3.6.1 PositionBytes as integer	85
* 3.6.1 PositionMS as integer	85
* 3.6.1 PositionSamples as integer	85
* 3.6.1 PositionTicks as integer	86
* 3.6.1 Restart	86
* 3.6.1 SendMessage(message as integer)	86
* 3.6.1 SendMessage(status as integer, data1 as integer, data2 as integer)	87
* 3.6.1 Stop	87
* 3.6.1 Tempo as integer	87
* 3.6.1 TimeDiv as integer	88
* 3.6.1 Volume as integer	88
* 3.6.2 Handle as Integer	89
* 3.6.2 Lasterror as Integer	89
– 3.7 class WindowsMidiInputMBS	89
* 3.7.1 Close	89
* 3.7.1 Idle	90
* 3.7.1 InputErrorText(errorcode as integer) as string	90
* 3.7.1 Open(DeviceID as integer, BufferSize as integer)	90
* 3.7.1 Reset	91
* 3.7.1 Start	91
* 3.7.1 Stop	91
* 3.7.2 DeviceClose	91
* 3.7.2 DeviceData(timestamp as integer, status as integer, data1 as integer, data2 as integer)	91
* 3.7.2 DeviceError(timestamp as integer, status as integer, data1 as integer, data2 as integer)	92
* 3.7.2 DeviceLongData(timestamp as integer, data as string)	92
* 3.7.2 DeviceLongError(timestamp as integer, data as string)	92
* 3.7.2 DeviceOpen	93
– 3.8 class WindowsMidiInputInfoMBS	93
* 3.8.1 DriverVersion as Integer	93
* 3.8.1 Flags as Integer	93
* 3.8.1 ManufacturerID as Integer	94
* 3.8.1 Name as String	94
* 3.8.1 ProductID as Integer	94
– 3.10 class WindowsMidiMBS	98
* 3.10.1 Connect(output as WindowsMidiOutputMBS)	99
* 3.10.1 DataLost as integer	99

* 3.10.1 Disconnect(output as WindowsMidiOutputMBS)	100
* 3.10.1 EventsLost as integer	100
* 3.10.1 Idle	100
* 3.10.1 InputDevice(index as integer) as WindowsMidiInputInfoMBS	101
* 3.10.1 NumberOfMidiInputDevices as integer	101
* 3.10.1 NumberOfMidiOutputDevices as integer	102
* 3.10.1 OutputDevice(index as integer) as WindowsMidiOutputInfoMBS	102
* 3.10.2 Handle as Integer	103
* 3.10.2 Lasterror as Integer	103

# Chapter 2

## Audio

### 2.1 class PortAudioStreamBufferedMBS

class PortAudioStreamBufferedMBS

Plugin Version: 7.4 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** A portaudio stream class to use an internal buffer to feed its callback.

**Notes:**

Currently this class allows you to add 200 buffers to the internal playlist. The buffer size is not limited. FreeSpace returns you the number of buffers you have currently. Buffers are freed after they are played. Subclass of the PortAudioStreamBaseMBS class.

#### 2.1.1 Methods

**AddAudio(Data as memoryblock, offsetBytes as integer=0, countBytes as integer=0, BitSize as integer=16, ClearBuffers as boolean=false) as boolean**

Plugin Version: 7.4 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Adds audio to the queue.

**Notes:**

Data: a memoryblock with the sound data  
offsetBytes: the number of the bytes to start playing (0=first)

countBytes: the number of bytes to play from offset. If countBytes is zero, the memoryblock's size property is used.

You need to set numOutputChannels to 1 or 2 when opening the stream. For 2 channels, sound data must be interleaved.

Values for bitsize:

7	signed byte
8	unsigned byte
15	signed short
16	unsigned short
24	unsigned medium
31	signed integer
32	unsigned integer

This method copies the data to the internal queue. It returns directly.

If ClearBuffers is true, the buffer list will be cleared before this new data is added. This allows to have the next minute in the buffers and still do a change in the stream quick.

Returns true on success and false on failure (e.g. out of memory).

**AddAudioStereo(Data1 as memoryblock, Data2 as memoryblock, offsetBytes as integer=0, countBytes as integer=0, BitSize as integer=16, ClearBuffers as boolean=false) as boolean**

Plugin Version: 7.4 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Adds audio to the queue.

**Notes:**

This is a special version of AddAudio which takes samples from both memoryblocks and interleaves them.

Data1: a memoryblock with the sound data

Data2: a memoryblock with the sound data

offsetBytes: the number of the bytes to start playing (0=first)

countBytes: the number of bytes to play from offset. If countBytes is zero, the memoryblock's size property

is used.

The stream must use 2 channels for this method to work correctly.

Values for bitsize:

7	signed byte
8	unsigned byte
15	signed short
16	unsigned short
24	unsigned medium
31	signed integer
32	unsigned integer

This method copies the data to the internal queue. It returns directly.

If ClearBuffers is true, the buffer list will be cleared before this new data is added. This allows to have the next minute in the buffers and still do a change in the stream quick.

Returns true on success and false on failure (e.g. out of memory).

**AddFloatAudio(FloatData as memoryblock, offsetBytes as integer=0, countBytes as integer=0, ClearBuffers as boolean=false) as boolean**

Plugin Version: 7.4 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Adds audio to the queue.

**Notes:**

FloatData: a memoryblock with the sound data filled with single values.

offsetBytes: the number of the bytes to start playing (0=first)

countBytes: the number of bytes to play from offset. If countBytes is zero, the memoryblock's size property is used.

You need to set numOutputChannels to 1 or 2 when opening the stream. For 2 channels, sound data must be interleaved.

The samples are stored in 32bit float values (`memoryblock.SingleValue`)

This method copies the data to the internal queue. It returns directly.

If `ClearBuffers` is true, the buffer list will be cleared before this new data is added. This allows to have the next minute in the buffers and still do a change in the stream quick.

Returns true on success and false on failure (e.g. out of memory).

**AddFloatAudioStereo(`FloatData1` as memoryblock, `FloatData2` as memoryblock, `offsetBytes` as integer=0, `countBytes` as integer=0, `ClearBuffers` as boolean=false) as boolean**

Plugin Version: 7.4 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Adds audio to the queue.

**Notes:**

This is a special version of `AddFloatAudio` which takes samples from both memoryblocks and interleaves them.

`FloatData1`: a memoryblock with the sound data filled with single values.

`FloatData2`: a memoryblock with the sound data filled with single values.

`offsetBytes`: the number of the bytes to start playing (0=first)

`countBytes`: the number of bytes to play from offset. If `countBytes` is zero, the memoryblock's size property is used.

The samples are stored in 32bit float values (`memoryblock.SingleValue`)

This method copies the data to the internal queue. It returns directly.

The stream must use 2 channels for this method to work correctly.

If `ClearBuffers` is true, the buffer list will be cleared before this new data is added. This allows to have the next minute in the buffers and still do a change in the stream quick.

Returns true on success and false on failure (e.g. out of memory).

### **FreeSpace as integer**

Plugin Version: 7.4 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Returns the number of buffer entries available in the queue.

**Notes:** The size of the buffers is not limited except your available memory.

### **HasFreeSpace as boolean**

Plugin Version: 7.4 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Returns true if the internal sound buffer has free space.

**Notes:**

Returns true if freespace returns a value greater than zero.

This function was named IsQueueEmpty in plugin version 7.4.

### **OpenDefaultStream(numOutputChannels as integer, sampleRate as double) as integer**

Plugin Version: 7.4 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** A simplified version of `OpenStream()` that opens the default input and/or output devices.

**Notes:**

Sampleformat is always 32bit float in this class. All data you add the queue is converted to float internally.

`numOutputChannels`: The number of channels of sound to be delivered to the stream callback or passed to `Write`. It can range from 1 to the value of `maxOutputChannels` in the `PortAudioDeviceInfoMBS` object for the default output device. If 0 the stream is opened as an output-only stream.

`sampleRate`: Same as `OpenStream` parameter of the same name.

Returns an error code.

**OpenStream(outputParameters as PortAudioStreamParametersMBS, sampleRate as double, framesPerBuffer as integer, streamFlags as integer) as integer**

Plugin Version: 7.4 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Opens a stream for either input, output or both.

**Notes:**

Sampleformat is always 32bit float in this class. All data you add the queue is converted to float internally.

outputParameters: A object that describes the output parameters used by the opened stream. See PortAudioStreamParametersMBS for a description of these parameters.

sampleRate: The desired sampleRate. For full-duplex streams it is the sample rate for both input and output

framesPerBuffer: The number of frames passed to the stream callback function, or the preferred block granularity for a blocking read/write stream. The special value paFramesPerBufferUnspecified (0) may be used to request that the stream callback will receive an optimal (and possibly varying) number of frames based on host requirements and the requested latency settings.

Note: With some host APIs, the use of non-zero framesPerBuffer for a callback stream may introduce an additional layer of buffering which could introduce additional latency. PortAudio guarantees that the additional latency will be kept to the theoretical minimum however, it is strongly recommended that a non-zero framesPerBuffer value only be used when your algorithm requires a fixed number of frames per stream callback.

const paFramesPerBufferUnspecified=0

streamFlags: Flags which modify the behaviour of the streaming process. This parameter may contain a combination of flags ORed together. Some flags may only be relevant to certain buffer formats.

Upon success OpenStream() returns paNoError and places a pointer to a valid PaStream in the stream argument. The stream is inactive (stopped).

If a call to OpenStream() fails, a non-zero error code is returned (see PaError for possible error codes) and the value of stream is invalid.

const paNoFlag	= 0	no flags
const paClipOff	= 1	Disable default clipping of out of range samples.
const paDitherOff	= 2	Flag requests that where possible a full duplex stream will not discard overflowed input samples without calling the stream callback. This flag is only valid for full duplex callback streams and only when used in combination with the paFramesPerBufferUnspecified (0) framesPerBuffer parameter. Using this flag incorrectly results in a paInvalidFlag error being returned from OpenStream and OpenDefaultStream.
const paNeverDropInput	= 4	Flag requests that where possible a full duplex stream will not discard overflowed input samples without calling the stream callback. This flag is only valid for full duplex callback streams and only when used in combination with the paFramesPerBufferUnspecified (0) framesPerBuffer parameter. Using this flag incorrectly results in a paInvalidFlag error being returned from OpenStream and OpenDefaultStream.
const paPrimeOutputBuffersUsingStreamCallback	= 8	Call the stream callback to fill initial output buffers, rather than the default behavior of priming the buffers with zeros (silence). This flag has no effect for input-only and blocking read/write streams.

**PlayAudio(Data as memoryblock, offsetBytes as integer=0, countBytes as integer=0, BitSize as integer=16, ClearBuffers as boolean=false) as boolean**

Plugin Version: 7.4 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Adds audio to the queue and starts playback.

**Notes:**

Data: a memoryblock with the sound data

offsetBytes: the number of the bytes to start playing (0=first)

countBytes: the number of bytes to play from offset. If countBytes is zero, the memoryblock's size property is used.

You need to set numOutputChannels to 1 or 2 when opening the stream. For 2 channels, sound data must be interleaved.

Values for bitsize:

- 7 signed byte
- 8 unsigned byte
- 15 signed short
- 16 unsigned short
- 24 unsigned medium
- 31 signed integer
- 32 unsigned integer

This method copies the data to the internal queue. It returns directly.

If `ClearBuffers` is true, the buffer list will be cleared before this new data is added. This allows to have the next minute in the buffers and still do a change in the stream quick.

**PlayAudioStereo(Data1 as memoryblock, Data2 as memoryblock, offsetBytes as integer=0, countBytes as integer=0, BitSize as integer=16, ClearBuffers as boolean=false) as boolean**

Plugin Version: 7.4 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Adds audio to the queue and starts playback.

**Notes:**

This is a special version of `AddAudio` which takes samples from both memoryblocks and interleaves them.

Data1: a memoryblock with the sound data

Data2: a memoryblock with the sound data

offsetBytes: the number of the bytes to start playing (0=first)

countBytes: the number of bytes to play from offset. If countBytes is zero, the memoryblock's size property is used.

The stream must use 2 channels for this method to work correctly.

Values for bitsize:

- 7 signed byte
- 8 unsigned byte
- 15 signed short
- 16 unsigned short
- 24 unsigned medium
- 31 signed integer
- 32 unsigned integer

This method copies the data to the internal queue. It returns directly.

If `ClearBuffers` is true, the buffer list will be cleared before this new data is added. This allows to have the next minute in the buffers and still do a change in the stream quick.#

**PlayFloatAudio(FloatData as memoryblock, offsetBytes as integer=0, countBytes as integer=0, ClearBuffers as boolean=false) as boolean**

Plugin Version: 7.4 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Adds audio to the queue and starts playback.

**Notes:**

FloatData: a memoryblock with the sound data filled with single values.

offsetBytes: the number of the bytes to start playing (0=first)

countBytes: the number of bytes to play from offset. If countBytes is zero, the memoryblock's size property is used.

You need to set numOutputChannels to 1 or 2 when opening the stream. For 2 channels, sound data must be interleaved.

The samples are stored in 32bit float values (memoryblock.SingleValue)

This method copies the data to the internal queue. It returns directly.

If ClearBuffers is true, the buffer list will be cleared before this new data is added. This allows to have the next minute in the buffers and still do a change in the stream quick. #

**PlayFloatAudioStereo(FloatData1 as memoryblock, FloatData2 as memoryblock, offsetBytes as integer=0, countBytes as integer=0, ClearBuffers as boolean=false) as boolean**

Plugin Version: 7.4 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Adds audio to the queue and starts playback.

**Notes:**

This is a special version of AddFloatAudio which takes samples from both memoryblocks and interleaves them.

FloatData1: a memoryblock with the sound data filled with single values.

FloatData2: a memoryblock with the sound data filled with single values.

offsetBytes: the number of the bytes to start playing (0=first)

countBytes: the number of bytes to play from offset. If countBytes is zero, the memoryblock's size property is used.

The samples are stored in 32bit float values (`memoryblock.SingleValue`)

This method copies the data to the internal queue. It returns directly.

The stream must use 2 channels for this method to work correctly.

If `ClearBuffers` is true, the buffer list will be cleared before this new data is added. This allows to have the next minute in the buffers and still do a change in the stream quick.

## 2.1.2 Properties

### HadUnderflow as Boolean

Plugin Version: 7.5 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** A flag set if a data underflow was found while `NoUnderflow` is true.

#### Notes:

You may want to set this to false after your application recovered from a data underflow.  
(Read and Write property)

### NoUnderflow as Boolean

Plugin Version: 7.5 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Whether underflows should be prevented.

#### Notes:

If no audio data is there and `NoUnderflow=true`, the flag `HadUnderflow` is set to true and 0 values (Silence) is played.

Switching from sound to no sound and back may add some noise.  
(Read and Write property)

### OutputPosition as Double

Plugin Version: 7.4 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** The current position in the playing stream.

**Notes:**

May point between samples.  
(Read only property)

### OutputPositionRelative as Double

Plugin Version: 7.4 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** The current position in the playing buffer.

**Notes:**

May point between samples.  
Will reset to 0 when a new buffer is used.  
(Read only property)

## 2.1.3 Events

### Finished

Plugin Version: 7.0 Console & Web: No Mac: Yes, Win: Yes, Linux: Yes, . **Function:** This event is implemented by PortAudio clients.

**Notes:**

Once registered they are called when the stream becomes inactive (ie once a call to Stop() will not block). A stream will become inactive after the stream callback returns non-zero, or when Stop or Abort is called. For a stream providing audio output, if the stream callback returns paComplete, or Stop is called, the stream finished callback will not be called until all generated sample data has been played.

## 2.2 class PortAudioStreamBaseMBS

### class PortAudioStreamBaseMBS

Plugin Version: 7.4 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** The base class for the portaudio streams.

### 2.2.1 Methods

#### Abort as integer

Plugin Version: 7.4 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Terminates audio processing immediately without waiting for pending buffers to complete.

#### Close as integer

Plugin Version: 7.4 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Closes an audio stream.

**Notes:** If the audio stream is active it discards any pending buffers as if Abort() had been called.

#### CPULoad as double

Plugin Version: 7.4 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Retrieve CPU usage information for the specified stream.

**Notes:**

The "CPU Load" is a fraction of total CPU time consumed by a callback stream's audio processing routines including, but not limited to the client supplied stream callback. This function does not work with blocking read/write streams.

This function may be called from the stream callback function or the application.

Returns a floating point value, typically between 0.0 and 1.0, where 1.0 indicates that the stream event is consuming the maximum number of CPU cycles possible to maintain real-time operation. A value of 0.5 would imply that PortAudio and the stream event was consuming roughly 50% of the available CPU time.

The return value may exceed 1.0. A value of 0.0 will always be returned for a blocking read/write stream, or if an error occurs.

### HostError as PortAudioHostErrorInfoMBS

Plugin Version: 7.4 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Return information about the last host error encountered.

#### Notes:

This function is provided as a last resort, primarily to enhance debugging by providing clients with access to all available error information.

Returns an object constaining information about the host error. The values in this structure will only be valid if a PortAudio function has previously returned the paUnanticipatedHostError.

### Info as PortAudioStreamInfoMBS

Plugin Version: 7.4 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Retrieve a PortAudioStreamInfoMBS object containing information about the specified stream.

**Notes:** If the stream parameter invalid, or an error is encountered, the function returns nil.

### IsStreamActive as integer

Plugin Version: 7.4 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Determine whether the stream is active.

#### Notes:

A stream is active after a successful call to Start(), until it becomes inactive either as a result of a call to Stop() or Abort(), or as a result of a return value other than paContinue from the stream callback. In the latter case, the stream is considered inactive after the last buffer has finished playing.

Returns one (1) when the stream is active (ie playing or recording audio), zero (0) when not playing or, a PaErrorCode (which are always negative) if PortAudio is not initialized or an error is encountered.

**IsStreamStopped as integer**

Plugin Version: 7.4 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Determine whether the stream is stopped.

**Notes:**

A stream is considered to be stopped prior to a successful call to Start and after a successful call to Stop or Abort.

If a stream value returns a value other than paContinue (0) the stream is NOT considered to be stopped.

Returns one (1) when the stream is stopped, zero (0) when the stream is running or, a PaErrorCode (which are always negative) if PortAudio is not initialized or an error is encountered.

**Start as integer**

Plugin Version: 7.4 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Commences audio processing.

**Notes:**

Returns an error code.

(0 for success, -1 for no stream)

**Stop as integer**

Plugin Version: 7.4 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Terminates audio processing.

**Notes:** It waits until all pending audio buffers have been played before it returns.

**Time as double**

Plugin Version: 7.4 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Determine the current time for the stream according to the same clock used to generate buffer timestamps.

**Notes:**

This time may be used for synchronising other events to the audio stream, for example synchronizing audio to MIDI.

Returns the stream's current time in seconds, or 0 if an error occurred.

### UseSafeThreading as boolean

Plugin Version: 7.4 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Enables or disables thread safe event handling.

#### Notes:

Realbasic works normally only on one preemptive thread as the framework is not reentrant and not in general preemptive thread safe.

Still newer Realbasic versions get better so you can switch it off and get a better performance.

For most usages you need to turn it off. See the examples. A lot of pragma lines are needed to disable everything which can slow down processing. Also you are limited in a preemptive thread to do only math and no object creating/deleting.

(Read and Write computed property)

## 2.3 class PortAudioStreamInfoMBS

### class PortAudioStreamInfoMBS

Plugin Version: 7.0 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** A class to hold time information of the current stream.

### 2.3.1 Methods

#### Constructor

Plugin Version: 12.1 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** The private constructor.

## 2.3.2 Properties

### InputLatency as Double

Plugin Version: 7.0 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** The input latency of the stream in seconds.

**Notes:**

This value provides the most accurate estimate of input latency available to the implementation. It may differ significantly from the suggestedLatency value passed to `OpenStream()`.

The value of this field will be zero (0.) for output-only streams.

(Read only property)

### OutputLatency as Double

Plugin Version: 7.0 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** The output latency of the stream in seconds.

**Notes:**

This value provides the most accurate estimate of output latency available to the implementation. It may differ significantly from the suggestedLatency value passed to `OpenStream()`.

The value of this field will be zero (0.) for input-only streams.

(Read only property)

### SampleRate as Double

Plugin Version: 7.0 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** The sample rate of the stream in Hertz (samples per second).

**Notes:**

In cases where the hardware sample rate is inaccurate and `PortAudio` is aware of it, the value of this field may be different from the `sampleRate` parameter passed to `OpenStream()`. If information about the actual hardware sample rate is not available, this field will have the same value as the `sampleRate` parameter passed to `OpenStream()`.

(Read only property)

## 2.4 class PortAudioStreamMBS

### class PortAudioStreamMBS

Plugin Version: 6.0 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** A single PortAudioStreamMBS can provide multiple channels of real-time streaming audio input and output to a client application.

**Notes:**

A stream provides access to audio hardware represented by one or more devices. Depending on the underlying Host API, it may be possible to open multiple streams using the same device, however this behavior is implementation defined. Portable applications should assume that a device may be simultaneously used by at most one PortAudioStreamMBS.

Subclass of the PortAudioStreamBaseMBS class.

### 2.4.1 Methods

**OpenDefaultStream(numInputChannels as integer, numOutputChannels as integer, sampleFormat as integer, sampleRate as double, framesPerBuffer as integer, Flags as integer) as integer**

Plugin Version: 7.0 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** A simplified version of OpenStream() that opens the default input and/or output devices.

**Notes:**

numInputChannels: The number of channels of sound that will be supplied to the stream callback or returned by ReadStream. It can range from 1 to the value of maxInputChannels in the PaDeviceInfo record for the default input device. If 0 the stream is opened as an output-only stream.

numOutputChannels: The number of channels of sound to be delivered to the stream callback or passed to Write. It can range from 1 to the value of maxOutputChannels in the PaDeviceInfo record for the default output device. If 0 the stream is opened as an output-only stream.

sampleFormat: The sample format of both the input and output buffers provided to the callback or passed to and from Read and Write.

sampleFormat may be any of the formats described by the PaSampleFormat enumeration.

sampleRate: Same as OpenStream parameter of the same name.

framesPerBuffer: Same as OpenStream parameter of the same name.

constants for the flags value:

<code>const paNoFlag</code>	= 0	no flags
<code>const paClipOff</code>	= 1	Disable default clipping of out of range samples.
<code>const paDitherOff</code>	= 2	Flag requests that where possible a full duplex stream will not discard overflowed input samples without calling the stream callback. This flag is only valid for full duplex callback streams and only when used in combination with the <code>paFramesPerBufferUnspecified</code> (0) <code>framesPerBuffer</code> parameter. Using this flag incorrectly results in a <code>paInvalidFlag</code> error being returned from <code>OpenStream</code> and <code>OpenDefaultStream</code> .
<code>const paNeverDropInput</code>	= 4	Flag requests that where possible a full duplex stream will not discard overflowed input samples without calling the stream callback. This flag is only valid for full duplex callback streams and only when used in combination with the <code>paFramesPerBufferUnspecified</code> (0) <code>framesPerBuffer</code> parameter. Using this flag incorrectly results in a <code>paInvalidFlag</code> error being returned from <code>OpenStream</code> and <code>OpenDefaultStream</code> .
<code>const paPrimeOutputBuffersUsingStreamCallback</code>	= 8	Call the stream callback to fill initial output buffers, rather than the default behavior of priming the buffers with zeros (silence). This flag has no effect for input-only and blocking read/write streams.

Returns an error code.

**OpenStream**(`inputParameters` as `PortAudioStreamParametersMBS`, `outputParameters` as `PortAudioStreamParametersMBS`, `sampleRate` as `double`, `framesPerBuffer` as `integer`, `streamFlags` as `integer`) as `integer`

Plugin Version: 7.0 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Opens a stream for either input, output or both.

**Notes:**

`inputParameters`: A object that describes the input parameters used by the opened stream. See `PortAudioStreamParametersMBS` for a description of these parameters. `inputParameters` must be `nil` for output-only streams.

`outputParameters`: A object that describes the output parameters used by the opened stream. See `PortAudioStreamParametersMBS` for a description of these parameters. `outputParameters` must be `nil` for input-only streams.

`sampleRate`: The desired `sampleRate`. For full-duplex streams it is the sample rate for both input and output

`framesPerBuffer`: The number of frames passed to the stream callback function, or the preferred block granularity for a blocking read/write stream. The special value `paFramesPerBufferUnspecified` (0) may be used to request that the stream callback will receive an optimal (and possibly varying) number of frames based on host requirements and the requested latency settings.

Note: With some host APIs, the use of non-zero framesPerBuffer for a callback stream may introduce an additional layer of buffering which could introduce additional latency. PortAudio guarantees that the additional latency will be kept to the theoretical minimum however, it is strongly recommended that a non-zero framesPerBuffer value only be used when your algorithm requires a fixed number of frames per stream callback.

```
const paFramesPerBufferUnspecified=0
```

streamFlags: Flags which modify the behaviour of the streaming process. This parameter may contain a combination of flags ORed together. Some flags may only be relevant to certain buffer formats.

const paNoFlag	= 0	no flags
const paClipOff	= 1	Disable default clipping of out of range samples.
const paDitherOff	= 2	Flag requests that where possible a full duplex stream will not discard overflowed input samples without calling the stream callback. This flag is only valid for full duplex callback streams and only when used in combination with the paFramesPerBufferUnspecified (0) framesPerBuffer parameter. Using this flag incorrectly results in a paInvalidFlag error being returned from OpenStream and OpenDefaultStream.
const paNeverDropInput	= 4	Flag requests that where possible a full duplex stream will not discard overflowed input samples without calling the stream callback. This flag is only valid for full duplex callback streams and only when used in combination with the paFramesPerBufferUnspecified (0) framesPerBuffer parameter. Using this flag incorrectly results in a paInvalidFlag error being returned from OpenStream and OpenDefaultStream.
const paPrimeOutputBuffersUsingStreamCallback	= 8	Call the stream callback to fill initial output buffers, rather than the default behavior of priming the buffers with zeros (silence). This flag has no effect for input-only and blocking read/write streams.

If this the callback event is left empty the stream will be opened in 'blocking read/write' mode. In blocking mode, the client can receive sample data using Read and write sample data using Write, the number of samples that may be read or written without blocking is returned by ReadAvailable and WriteAvailable respectively.

Upon success OpenStream() returns paNoError and places a pointer to a valid PaStream in the stream argument. The stream is inactive (stopped).

If a call to OpenStream() fails, a non-zero error code is returned (see PaError for possible error codes) and the value of stream is invalid.

**Read(buffer as memoryblock, frameCount as integer) as integer**

Plugin Version: 7.0 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Read samples from an input stream.

**Notes:**

The function doesn't return until the entire buffer has been filled - this may involve waiting for the operating system to supply the data.

**buffer:** A buffer of sample frames. The buffer contains samples in the format specified by the `input.sampleFormat` field used to open the stream, and the number of channels specified by `input.numChannels`. If non-interleaved samples were requested, `buffer` is a pointer to the first element of an array of non-interleaved buffer pointers, one for each channel.

**frameCount:** The number of frames to be read into `buffer`. This parameter is not constrained to a specific range, however high performance applications will want to match this parameter to the `framesPerBuffer` parameter used when opening the stream.

Returns on success `PaNoError` will be returned, or `PaInputOverflowed` (-9981) if input data was discarded by `PortAudio` after the previous call and before this call.

### **ReadAvailable as integer**

Plugin Version: 7.0 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Retrieve the number of frames that can be read from the stream without waiting.

**Notes:** Returns a non-negative value representing the maximum number of frames that can be read from the stream without blocking or busy waiting or, a `PaErrorCode` (which are always negative) if `PortAudio` is not initialized or an error is encountered.

### **Write(buffer as memoryblock, frameCount as integer) as integer**

Plugin Version: 7.0 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Write samples to an output stream.

**Notes:**

This function doesn't return until the entire buffer has been consumed - this may involve waiting for the operating system to consume the data.

**buffer:** A buffer of sample frames. The buffer contains samples in the format specified by the `outputParameters.sampleFormat` field used to open the stream, and the number of channels specified by `outputParameters.numChannels`. If non-interleaved samples were requested, `buffer` is a pointer to the first element of an array of non-interleaved buffer pointers, one for each channel.

frameCount: The number of frames to be written from buffer. This parameter is not constrained to a specific range, however high performance applications will want to match this parameter to the framesPerBuffer parameter used when opening the stream.

On success PaNoError (0) will be returned, or paOutputUnderflowed (-9980) if additional output data was inserted after the previous call and before this call.

### WriteAvailable as integer

Plugin Version: 7.0 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Retrieve the number of frames that can be written to the stream without waiting.

**Notes:** Returns a non-negative value representing the maximum number of frames that can be written to the stream without blocking or busy waiting or, a PaErrorCode (which are always negative) if PortAudio is not initialized or an error is encountered.

## 2.4.2 Events

**Callback(InputBuffer as memoryblock, outputBuffer as memoryblock, FrameCount as integer, inputBufferAdcTime as double, currentTime as double, outputBufferDacTime as double, statusFlags as integer) as integer**

Plugin Version: 7.0 Console & Web: No Mac: Yes, Win: Yes, Linux: Yes, . **Function:** The event called when new data is needed or received.

**Notes:**

This event is implemented by PortAudio clients. They consume, process or generate audio in response to requests from an active PortAudio stream.

InputBuffer and outputBuffer are arrays of interleaved samples, the format, packing and number of channels used by the buffers are determined by parameters to OpenStream().

frameCount: The number of sample frames to be processed by the stream callback.

inputBufferAdcTime, currentTimeTiming and outputBufferDacTime: The time in seconds when the first sample of the input buffer was received at the audio input, the time in seconds when the first sample of the output buffer will begin being played at the audio output, and the time in seconds when the stream callback was called.

`statusFlags`: Flags indicating whether input and/or output buffers have been inserted or will be dropped to overcome underflow or overflow conditions.

Returns the stream callback should return one of the values in the `PaStreamCallbackResult` enumeration. To ensure that the callback continues to be called, it should return `paContinue` (0). Either `paComplete` or `paAbort` can be returned to finish stream processing, after either of these values is returned the callback will not be called again. If `paAbort` is returned the stream will finish as soon as possible. If `paComplete` is returned, the stream will continue until all buffers generated by the callback have been played. This may be useful in applications such as soundfile players where a specific duration of output is required. However, it is not necessary to utilise this mechanism as `Stop()`, `Abort()` or `Close()` can also be used to stop the stream. The callback must always fill the entire output buffer irrespective of its return value.

With the exception of `CpuLoad()` it is not permissible to call PortAudio API functions from within the stream callback.

Flag bit constants for the `statusFlags` to `Callback`:

`paInputUnderflow = 1`

In a stream opened with `paFramesPerBufferUnspecified`, indicates that input data is all silence (zeros) because no real data is available. In a stream opened without `paFramesPerBufferUnspecified`, it indicates that one or more zero samples have been inserted into the input buffer to compensate for an input underflow.

`paInputOverflow = 2`

In a stream opened with `paFramesPerBufferUnspecified`, indicates that data prior to the first sample of the input buffer was discarded due to an overflow, possibly because the stream callback is using too much CPU time. Otherwise indicates that data prior to one or more samples in the input buffer was discarded.

`paOutputUnderflow = 4`

Indicates that output data (or a gap) was inserted, possibly because the stream callback is using too much CPU time.

`paOutputOverflow = 8`

Indicates that output data will be discarded because no room is available.

`paPrimingOutput = 16`

Some of all of the output data will be used to prime the stream, input data may be zero.

Allowable return values for the callback: (PaStreamCallbackResult)

```
const paContinue    = 0
const paComplete   = 1
const paAbort      = 2
```

### Finished

Plugin Version: 7.0 Console & Web: No Mac: Yes, Win: Yes, Linux: Yes, . **Function:** This event is implemented by PortAudio clients.

#### Notes:

Once registered they are called when the stream becomes inactive (ie once a call to Stop() will not block). A stream will become inactive after the stream callback returns non-zero, or when Stop or Abort is called. For a stream providing audio output, if the stream callback returns paComplete, or Stop is called, the stream finished callback will not be called until all generated sample data has been played.

## 2.5 class PortAudioHostErrorInfoMBS

### class PortAudioHostErrorInfoMBS

Plugin Version: 7.0 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** A class used to return information about a host error condition.

### 2.5.1 Methods

#### Constructor

Plugin Version: 12.1 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** The private constructor.

## 2.5.2 Properties

### **ErrorCode as Integer**

Plugin Version: 7.0 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** the error code returned.

**Notes:** (Read only property)

### **ErrorText as String**

Plugin Version: 7.0 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** A textual description of the error if available, otherwise a zero-length string.

**Notes:** (Read only property)

### **HostApiType as Integer**

Plugin Version: 7.0 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** The host API which returned the error code.

**Notes:** (Read only property)

## 2.6 class PortAudioHostApiInfoMBS

### **class PortAudioHostApiInfoMBS**

Plugin Version: 7.0 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** A class containing information about a particular host API.

## 2.6.1 Methods

### Constructor

Plugin Version: 12.1 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** The private constructor.

## 2.6.2 Properties

### defaultInputDevice as Integer

Plugin Version: 7.0 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** The default input device for this host API.

#### Notes:

The value will be a device index ranging from 0 to (PortAudioHostApiInfoMBS.deviceCount-1), or paNoDevice (-1) if no default input device is available.  
(Read only property)

### defaultOutputDevice as Integer

Plugin Version: 7.0 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** The default output device for this host API.

#### Notes:

The value will be a device index ranging from 0 to (PortAudioHostApiInfoMBS.deviceCount-1), or paNoDevice (-1) if no default output device is available.  
(Read only property)

### deviceCount as Integer

Plugin Version: 7.0 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** The number of devices belonging to this host API.

#### Notes:

This field may be used in conjunction with HostApiDeviceIndexToDeviceIndex() to enumerate all devices

for this host API.  
(Read only property)

### Index as Integer

Plugin Version: 7.0 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** The index of this host API.

**Notes:** (Read only property)

### Name as String

Plugin Version: 7.0 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** A textual description of the host API for display on user interfaces.

**Notes:** (Read only property)

### Type as Integer

Plugin Version: 7.0 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** The well known unique identifier of this host API.

**Notes:**

Useful constants for the different types:

paInDevelopment	=0
paDirectSound	=1
paMME	=2
paASIO	=3
paSoundManager	=4
paCoreAudio	=5
paOSS	=7
paALSA	=8
paAL	=9
paBeOS	=10
paWDMKS	=11
paJACK	=12
paWASAPI	=13
paAudioScienceHPI	=14

(Read only property)

## 2.7 class PortAudioMBS

### class PortAudioMBS

Plugin Version: 6.0 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** A class to run the opensource PortAudio library in Realbasic.

#### Notes:

Constants for error codes:

paNoError	= 0
Plugin parameter error	= -1
paNotInitialized	= -10000,
paUnanticipatedHostError	= -9999
paInvalidChannelCount	= -9998
paInvalidSampleRate	= -9997
paInvalidDevice	= -9996
paInvalidFlag	= -9995
paSampleFormatNotSupported	= -9994
paBadIODeviceCombination	= -9993
paInsufficientMemory	= -9992
paBufferTooBig	= -9991
paBufferTooSmall	= -9990
paNullCallback	= -9989
paBadStreamPtr	= -9988
paTimedOut	= -9987
paInternalError	= -9986
paDeviceUnavailable	= -9985
paIncompatibleHostApiSpecificStreamInfo	= -9984
paStreamIsStopped	= -9983
paStreamIsNotStopped	= -9982
paInputOverflowed	= -9981
paOutputUnderflowed	= -9980
paHostApiNotFound	= -9979
paInvalidHostApi	= -9978
paCanNotReadFromACallbackStream	= -9977
paCanNotWriteToACallbackStream	= -9976
paCanNotReadFromAnOutputOnlyStream	= -9975
paCanNotWriteToAnInputOnlyStream	= -9974
paIncompatibleStreamHostApi	= -9973
paBadBufferPtr	= -9972

Initialization and Termination of the PortAudio library are done in background automatically.

Requires libasound.so.2 on Linux to be installed.

### 2.7.1 Methods

#### CountDevices as integer

Plugin Version: 6.0 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Retrieve the number of available devices.

**Example:**

```
dim pa as new PortAudioMBS

Dim c as integer = pa.CountDevices
msgbox str(c)+" devices found"

for i as integer = 0 to c-1
MsgBox pa.DeviceInfo(i).Name
next
```

**Notes:**

The number of available devices may be zero.

Returns a non-negative value indicating the number of available devices or, a PaErrorCode (which are always negative) if PortAudio is not initialized or an error is encountered.

#### DefaultHostApiIndexd as integer

Plugin Version: 7.0 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Retrieve the index of the default host API.

**Example:**

```
dim pa as new PortAudioMBS

dim DefaultHostApiIndexd as integer = pa.DefaultHostApiIndexd

if DefaultHostApiIndexd >= 0 then
```

```

dim d as PortAudioHostApiInfoMBS = pa.HostApiInfo(DefaultHostApiIndexd)
MsgBox "Default host API is: "+d.Name
else
MsgBox "No default host API."
end if

```

**Notes:**

The default host API will be the lowest common denominator host API on the current platform and is unlikely to provide the best performance.

Returns a non-negative value ranging from 0 to (HostApiCount-1) indicating the default host API index or, a PaErrorCode (which are always negative) if PortAudio is not initialized or an error is encountered.

**DefaultInputDeviceID as integer**

Plugin Version: 6.0 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Retrieve the index of the default input device.

**Example:**

```

dim pa as new PortAudioMBS

dim DefaultInputDeviceID as integer = pa.DefaultInputDeviceID

if DefaultInputDeviceID >= 0 then
dim d as PortAudioDeviceInfoMBS = pa.DeviceInfo(DefaultInputDeviceID)
MsgBox "Default input device is: "+d.Name
else
MsgBox "No default input device."
end if

```

**Notes:**

The result can be used in the inputDevice parameter to `OpenStream()`.

Returns the default input device index for the default host API, or paNoDevice (-1) if no default input device is available or an error was encountered.

**DefaultOutputDeviceID as integer**

Plugin Version: 6.0 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Retrieve the index of the default output device.

**Example:**

```
dim pa as new PortAudioMBS

dim DefaultOutputDeviceID as integer = pa.DefaultOutputDeviceID

if DefaultOutputDeviceID >= 0 then
dim d as PortAudioDeviceInfoMBS = pa.DeviceInfo(DefaultOutputDeviceID)
MsgBox "Default output device is: " + d.Name
else
MsgBox "No default output device."
end if
```

**Notes:**

The result can be used in the `outputDevice` parameter to `OpenStream()`.

Returns the default output device index for the default host API, or `paNoDevice` (-1) if no default output device is available or an error was encountered.

On the PC, the user can specify a default device by setting an environment variable. For example, to use device # 1.

```
set PA_RECOMMENDED_OUTPUT_DEVICE=1
```

The user should first determine the available device ids by using the supplied application "pa\_devs".

**DeviceInfo(DeviceIndex as integer) as PortAudioDeviceInfoMBS**

Plugin Version: 6.0 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Retrieve an object containing information about the specified device.

**Example:**

```

dim pa as new PortAudioMBS

Dim u as integer = pa.CountDevices-1
for i as integer = 0 to u
dim d as PortAudioDeviceInfoMBS = pa.DeviceInfo(i)
MsgBox d.Name+" with default "+str(D.DefaultSampleRate)+" Hz"
next

```

**Notes:**

Returns an object of class PortAudioDeviceInfoMBS with the requested information. If the device parameter is out of range the function returns nil.

DeviceIndex: A valid device index in the range 0 to (PortAudioMBS.CountDevices-1)

**DisableHostAPI(API as string) as boolean**

Plugin Version: 12.1 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Disables a PortAudio Host API.

**Example:**

```

if PortAudioMBS.DisableHostAPI("Core Audio") then // for Mac
MsgBox "OK"
else
MsgBox "Failed"
end if

```

**Notes:**

This must be called before using any PortAudio function.

It removes the API from the list of APIs to be used when PortAudio initializes. This way you can avoid loading interfaces you don't need.

API name can be "MME", "Windows DirectSound", "Windows WASAPI", "ASIO", "Core Audio", "ALSA", "OSS".

Returns true on success.

**ErrorText(ErrorNumber as integer) as string**

Plugin Version: 6.0 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Translate the error number into a human readable message.

**GetSampleSize(Format as integer) as integer**

Plugin Version: 7.0 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Returns the sample size in bytes of the given format.

**Example:**

```
dim pa as new PortAudioMBS  
  
const paFloat32 = 1  
  
MsgBox str(pa.GetSampleSize(paFloat32))+” bytes per sample”
```

**HostApiCount as integer**

Plugin Version: 7.0 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Retrieve the number of available host APIs.

**Example:**

```
dim pa as new PortAudioMBS  
  
Dim c as integer = pa.HostApiCount  
msgbox str(c)+” host APIs found”
```

**Notes:**

Even if a host API is available it may have no devices available.

Returns a non-negative value indicating the number of available host APIs or, a PaErrorCode (which are always negative) if PortAudio is not initialized or an error is encountered.

**HostApiDeviceIndexToDeviceIndex(hostApiIndex as integer, hostApiDeviceIndex as integer) as integer**

Plugin Version: 7.0 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Convert a host-API-specific device index to standard PortAudio device index.

**Notes:**

This function may be used in conjunction with the deviceCount field of PaHostApiInfo to enumerate all devices for the specified host API.

hostApiIndex: A valid host API index ranging from 0 to (HostApiCount-1)

hostApiDeviceIndex: A valid per-host device index in the range 0 to (HostApiInfo(hostApi).deviceCount-1)

Returns a non-negative device index ranging from 0 to (DeviceCount-1) or, an error code (which are always negative) if PortAudio is not initialized or an error is encountered.

A paInvalidHostApi (-9978) error code indicates that the host API index specified by the hostApi parameter is out of range.

A paInvalidDevice (-9996) error code indicates that the hostApiDeviceIndex parameter is out of range.

**HostApiInfo(hostApiIndex as integer) as PortAudioHostApiInfoMBS**

Plugin Version: 7.0 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Retrieve an PortAudioHostApiInfoMBS object containing information about a specific host Api.

**Example:**

```
dim pa as new PortAudioMBS
Dim u as integer = pa.HostApiCount - 1

for i as integer = 0 to u
dim d as PortAudioHostApiInfoMBS = pa.HostApiInfo(i)
MsgBox d.name
next
```

**Notes:**

hostApiIndex: A valid host API index ranging from 0 to (HostApiCount-1)

Returns the information object. If the hostApi parameter is out of range or an error is encountered, the function returns nil.

### HostApiTypeIdToHostApiIndex(type as integer) as integer

Plugin Version: 7.0 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Convert a static host API unique identifier, into a runtime host API index.

#### Example:

```
dim IndexDirectSound as integer
dim p as new PortAudioMBS
const paDirectSound=1
IndexDirectSound=p.HostApiTypeIdToHostApiIndex(paDirectSound)
```

#### Notes:

type: A unique host API identifier. See PortAudioHostApiInfoMBS.Type for the list of constants.

Returns a valid PaHostApiIndex ranging from 0 to (HostApiCount-1) or, a PaErrorCode (which are always negative) if PortAudio is not initialized or an error is encountered.

The paHostApiNotFound (-9979) error code indicates that the host API specified by the type parameter is not available.

### HostError as PortAudioHostErrorInfoMBS

Plugin Version: 7.0 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Return information about the last host error encountered.

#### Notes:

This function is provided as a last resort, primarily to enhance debugging by providing clients with access to all available error information.

Returns an object constaining information about the host error. The values in this structure will only be valid if a PortAudio function has previously returned the paUnanticipatedHostError.

**IsFormatSupported**(input as PortAudioStreamParametersMBS, output as PortAudioStreamParametersMBS, sampleRate as double) as integer

Plugin Version: 7.0 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Determine whether it would be possible to open a stream with the specified parameters.

**Notes:**

input: An object that describes the input parameters used to open a stream. The suggestedLatency field is ignored. inputParameters must be nil for output-only streams.

output: An object that describes the output parameters used to open a stream. The suggestedLatency field is ignored. outputParameters must be nil for input-only streams.

sampleRate: The required sampleRate. For full-duplex streams it is the sample rate for both input and output

Returns 0 if the format is supported, and an error code indicating why the format is not supported otherwise. The constant paFormatIsSupported (0) is provided to compare with the return value for success.

**SampleSize**(theFormat as integer) as integer

Plugin Version: 6.0 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Return size in bytes of a single sample in a given sample format or paSampleFormatNotSupported (-9994).

**Notes:**

Returns 0 on any error.

Constants for sample format:

The standard formats paFloat32, paInt16, paInt32, paInt24, paInt8 and aUInt8 are usually implemented by all implementations.

```

const paFloat32      = 1
const paInt32        = 2
const paInt24        = 4
const paInt16        = 8
const paInt8         = 16
const paUInt8        = 32
const paCustomFormat = 65536
const paNonInterleaved = negative sign

```

The floating point representation (`paFloat32`) uses +1.0 and -1.0 as the maximum and minimum respectively.

`paUInt8` is an unsigned 8 bit format where 128 is considered "ground"

The `paNonInterleaved` flag indicates that a multichannel buffer is passed as a set of non-interleaved pointers.

### **Sleep(msec as integer)**

Plugin Version: 7.0 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Put the caller to sleep for at least 'msec' milliseconds.

#### **Notes:**

This function is provided only as a convenience for authors of portable code (such as the tests and examples in the PortAudio distribution.)

The function may sleep longer than requested so don't rely on this for accurate musical timing.

### **Version as integer**

Plugin Version: 7.0 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Retrieve the release number of the currently running PortAudio build.

**Notes:** e.g. 1900

### **VersionText as String**

Plugin Version: 7.0 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Retrieve a textual description of the current PortAudio build.

**Notes:** e.g. "PortAudio V19-devel 13 October 2002"

## 2.8 class PortAudioStreamRecorderMBS

### class PortAudioStreamRecorderMBS

Plugin Version: 9.1 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** A portaudio stream class to use an internal buffer to record audio.

**Notes:**

This class has a ring buffer to store the audio samples which they are being recorded. Your application can in a timer or thread process this samples.

Subclass of the PortAudioStreamBaseMBS class.

### 2.8.1 Methods

#### Constructor(BufferSize as integer)

Plugin Version: 9.1 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Creates a new object using a buffer of the given size.

**Notes:**

The buffer must be a power of 2. For example one Megabyte ( $2^{20}$ ).

At 44100 Hz, and 4 bytes per value and 2 channels, you will need 352800 bytes per second on storage.

#### Flush

Plugin Version: 10.1 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Clears the buffer and discards all current samples.

**OpenDefaultStream(numInputChannels as integer, sampleRate as double) as integer**

Plugin Version: 9.1 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** A simplified version of `OpenStream()` that opens the default input devices.

**Notes:**

Sampleformat is always 32bit float in this class. (use `memoryblock.singlevalue`)

numInputChannels: The number of channels of sound to be delivered. It can range from 1 to the value of `maxInputChannels` in the `PortAudioDeviceInfoMBS` object for the default output device. If 0 the stream is opened as an output-only stream.

sampleRate: Same as `OpenStream` parameter of the same name.

Returns an error code.

Error -2 is from the plugin and reports that the buffer was not created before.

**OpenStream(inputParameters as PortAudioStreamParametersMBS, sampleRate as double, framesPerBuffer as integer, streamFlags as integer) as integer**

Plugin Version: 9.1 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Opens a stream for input.

**Notes:**

Sampleformat is always 32bit float in this class. (use `memoryblock.singlevalue`)

outputParameters: A object that describes the input parameters used by the opened stream. See `PortAudioStreamParametersMBS` for a description of these parameters.

sampleRate: The desired sampleRate.

framesPerBuffer: The number of frames passed to the stream callback function, or the preferred block granularity for a blocking read/write stream. The special value `paFramesPerBufferUnspecified` (0) may be used to request that the stream callback will receive an optimal (and possibly varying) number of frames based on host requirements and the requested latency settings.

Note: With some host APIs, the use of non-zero `framesPerBuffer` for a callback stream may introduce an additional layer of buffering which could introduce additional latency. `PortAudio` guarantees that the additional latency will be kept to the theoretical minimum however, it is strongly recommended that a non-zero `framesPerBuffer` value only be used when your algorithm requires a fixed number of frames per stream call-

back.

```
const paFramesPerBufferUnspecified=0
```

streamFlags: Flags which modify the behaviour of the streaming process. This parameter may contain a combination of flags ORed together. Some flags may only be relevant to certain buffer formats.

const paNoFlag	= 0	no flags
const paClipOff	= 1	Disable default clipping of out of range samples.
const paDitherOff	= 2	Flag requests that where possible a full duplex stream will not discard overflowed input samples without calling the stream callback. This flag is only valid for full duplex callback streams and only when used in combination with the paFramesPerBufferUnspecified (0) framesPerBuffer parameter. Using this flag incorrectly results in a paInvalidFlag error being returned from OpenStream and OpenDefaultStream.
const paNeverDropInput	= 4	Call the stream callback to fill initial output buffers, rather than the default behavior of priming the buffers with zeros (silence). This flag has no effect for input-only and blocking read/write streams.
const paPrimeOutputBuffersUsingStreamCallback	= 8	Call the stream callback to fill initial output buffers, rather than the default behavior of priming the buffers with zeros (silence). This flag has no effect for input-only and blocking read/write streams.

Upon success OpenStream() returns paNoError and places a pointer to a valid PaStream in the stream argument. The stream is inactive (stopped).

If a call to OpenStream() fails, a non-zero error code is returned (see PaError for possible error codes) and the value of stream is invalid.

Error -2 is from the plugin and reports that the buffer was not created before.

### ReadFrames(mem as memoryblock, SizeInBytes as integer) as integer

Plugin Version: 9.1 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Reads frames into the memoryblock.

#### Example:

```
dim s as PortAudioStreamRecorderMBS
```

```
dim m as memoryblock
```

```
m=newmemoryblock(1024*1024)
```

```
// initialize
```

```
dim frames as integer
```

```
frames=s.ReadFrames(m,m.size)
```

```
msgbox "we got "+str(frames)+" frames."
```

### Notes:

You pass a memoryblock and the size of this memoryblock in bytes. Values are stored in floats (memoryblock.singlevalue) so you get at maximum SizeInBytes/4 values. And if you use more than one channel, you will receive them interlaced.

ReadFrames uses a mutex to access share data, so this call is expensive. Use a big buffer.

### ResizeBuffer(BufferSize as integer)

Plugin Version: 9.1 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Resizes the buffer.

#### Notes:

Do not resize while you are recording (this can crash).

The buffer must be a power of 2. For example one Megabyte (2<sup>20</sup>).

At 44100 Hz, and 4 bytes per value and 2 channels, you will need 352800 bytes per second on storage.

## 2.8.2 Properties

### Buffer as Memoryblock

Plugin Version: 10.1 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Returns a memoryblock which points to the ring buffer used in the recorder object.

#### Notes:

This memoryblock has no size.

It is only for debugging and only valid as long as the PortAudioStreamRecorderMBS object is living. (Read only property)

**BufferReadIndex as Integer**

Plugin Version: 10.1 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** The read index inside the ring buffer.

**Notes:**

Only for debugging.  
(Read only property)

**BufferSize as Integer**

Plugin Version: 10.1 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** The size of the ring buffer in bytes.

**Notes:** (Read only property)

**BufferWriteIndex as Integer**

Plugin Version: 10.1 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** The write index inside the ring buffer.

**Notes:**

Only for debugging.  
(Read only property)

**FramesAvailable as Integer**

Plugin Version: 9.1 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** The number of frames available in the buffer.

**Notes:**

FramesAvailable uses a mutex to access share data, so this call is expensive. Do not call it to decide whether to call ReadFrames. ReadFrames calls FramesAvailable itself.  
(Read only property)

**NumInputChannels as Integer**

Plugin Version: 9.1 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** The number of input channels used with the OpenStream function.

**Notes:** (Read only property)

**2.9 class PortAudioStreamParametersMBS****class PortAudioStreamParametersMBS**

Plugin Version: 7.0 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Parameters for one direction (input or output) of a stream.

**2.9.1 Properties****ChannelCount as Integer**

Plugin Version: 7.0 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** The requested channel count.

**Notes:**

The number of channels of sound to be delivered to the stream callback or accessed by Read() or Write(). It can range from 1 to the value of maxInputChannels in the DeviceInfo object for the device specified by the device parameter.

(Read and Write property)

**Device as Integer**

Plugin Version: 7.0 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** The device ID to be used.

**Notes:**

A valid device index in the range 0 to (PortAudioMBS.CountDevices-1) specifying the device to be used or the special constant paUseHostApiSpecificDeviceSpecification which indicates that the actual device(s) to use are specified in hostApiSpecificStreamInfo.

This field must not be set to paNoDevice (-1).  
(Read and Write property)

### SampleFormat as Integer

Plugin Version: 7.0 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** The sample format of the buffer provided to the stream callback, Read() or Write().

#### Notes:

Constants for sample format:

```
const paFloat32      = 1
const paInt32        = 2
const paInt24        = 4
const paInt16        = 8
const paInt8         = 16
const paUInt8        = 32
const paCustomFormat = 65536
const paNonInterleaved = negative sign
```

The standard formats paFloat32, paInt16, paInt32, paInt24, paInt8 and aUInt8 are usually implemented by all implementations.

The floating point representation (paFloat32) uses +1.0 and -1.0 as the maximum and minimum respectively.

paUInt8 is an unsigned 8 bit format where 128 is considered "ground"

The paNonInterleaved flag indicates that a multichannel buffer is passed as a set of non-interleaved pointers.  
(Read and Write property)

### SuggestedLatency as Double

Plugin Version: 7.0 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** The desired latency in seconds.

#### Notes:

Where practical, implementations should configure their latency based on these parameters, otherwise they

may choose the closest viable latency instead. Unless the suggested latency is greater than the absolute upper limit for the device implementations should round the suggestedLatency up to the next practical value - ie to provide an equal or higher latency than suggestedLatency wherever possible. Actual latency values for an open stream may be retrieved using the inputLatency and outputLatency fields of the PortAudioStreamInfoMBS object returned by PortAudioStreamMBS.Info(). (Read and Write property)

## 2.10 class PortAudioDeviceInfoMBS

### class PortAudioDeviceInfoMBS

Plugin Version: 6.0 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** A class providing information and capabilities of PortAudio devices.

**Notes:** Devices may support input, output or both input and output.

### 2.10.1 Methods

#### Constructor

Plugin Version: 12.1 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** The private constructor.

### 2.10.2 Properties

#### DefaultHighInputLatency as Double

Plugin Version: 7.0 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Default latency values for robust non-interactive applications (eg. playing sound files).

**Notes:** (Read only property)

**DefaultHighOutputLatency as Double**

Plugin Version: 7.0 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Default latency values for robust non-interactive applications (eg. playing sound files).

**Notes:** (Read only property)

**DefaultLowInputLatency as Double**

Plugin Version: 7.0 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Default latency values for interactive performance.

**Notes:** (Read only property)

**DefaultLowOutputLatency as Double**

Plugin Version: 7.0 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Default latency values for interactive performance.

**Notes:** (Read only property)

**DefaultSampleRate as Double**

Plugin Version: 7.0 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** The default sample rate.

**Notes:** (Read only property)

**HostApiIndex as Integer**

Plugin Version: 7.0 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** The host API index for this device.

**Notes:** (Read only property)

**Index as Integer**

Plugin Version: 7.0 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** The index of this device.

**Notes:** (Read only property)

**MaxInputChannels as integer**

Plugin Version: 6.0 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** The number of input channels.

**Notes:**

Returns 0 on any error.

(Read only property)

**MaxOutputChannels as integer**

Plugin Version: 6.0 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** The number of output channels for this device.

**Notes:**

0 on any error.

Seems like an iMac returns two here, but the internal microphone has only one channel. So be aware that the memoryblocks in the events do have the actual size of the data that is coming in. So stereo has there a larger buffer.

(Read only property)

**Name as String**

Plugin Version: 6.0 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** The name of this device.

**Notes:**

Returns "" on any error.

(Read only property)





# Chapter 3

## MIDI

### 3.1 class PortMidiStreamMBS

**class PortMidiStreamMBS**

Plugin Version: 5.2 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** The Realbasic class to represent a PortMidi stream.

**Notes:** A single PortMidiStream is a descriptor for an open MIDI device.

#### 3.1.1 Methods

**Abort as integer**

Plugin Version: 5.2 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Terminates outgoing messages immediately.

**Notes:**

The caller should immediately close the output port; this call may result in transmission of a partial midi message. There is no abort for Midi input because the user can simply ignore messages in the buffer and close an input device at any time.

Returns an error code.

**Close**

Plugin Version: 5.2 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** The destructor.

**Notes:**

There is no need to call this method except you want to free all resources of this object now without waiting for Realbasic to do it for you.

(e.g. some Realbasic versions crash on Windows if there are plugin objects not closed.)

**ErrorText(ErrorNumber as integer) as string**

Plugin Version: 5.2 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** The error message for this error number.

**OpenInput(DeviceID as integer, BufferSize as integer) as integer**

Plugin Version: 5.2 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Opens a device for reading.

**Notes:**

DeviceID is the id of the device used for input.

For input, the buffersize specifies the number of input events to be buffered waiting to be read using Pm\_Read().

(In some cases – see below – PortMidi does not buffer output at all and merely passes data to a lower-level API, in which case buffersize is ignored.)

Latency is the delay in milliseconds applied to timestamps to determine when the output should actually occur. (If latency is <0, 0 is assumed.)

If latency is zero, timestamps are ignored and all output is delivered immediately. If latency is greater than zero, output is delayed until the message timestamp plus the latency. (NOTE: time is measured relative to the time source indicated by time\_proc. Timestamps are absolute, not relative delays or offsets.) In some cases, PortMidi can obtain better timing than your application by passing timestamps along to the device driver or hardware. Latency may also help you to synchronize midi data to audio data by matching midi latency to the audio buffer latency.

return value:

Upon success `OpenInput` returns `PmNoError`.

If a call to `OpenInput` fails a nonzero error code is returned (see `PMError` above) and the value of port is invalid.

### **OpenOutput(DeviceID as integer, Bufferize as integer, Latency as integer) as integer**

Plugin Version: 5.2 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Opens a device for writing.

#### **Notes:**

`DeviceID` is the id of the device used for input.

For output, `bufferize` specifies the number of output events to be buffered waiting for output. (In some cases – see below – `PortMidi` does not buffer output at all and merely passes data to a lower-level API, in which case `bufferize` is ignored.)

return value:

Upon success `OpenInput` returns `PmNoError`.

If a call to `OpenInput` fails a nonzero error code is returned (see `PMError` above) and the value of port is invalid.

### **Poll as integer**

Plugin Version: 5.2 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Tests whether input is available,

**Notes:** Returns 1 on success and 0 on failure.

### **Read(byref data as PortMidiEventMBS) as integer**

Plugin Version: 5.2 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Reads midi data.

#### **Notes:**

Returns the number of read items.

(0 for error and 1 for success)

Read retrieves midi data into a buffer, and returns the number of events read. Result is a non-negative number unless an error occurs, in which case a PmError value will be returned.

### Buffer Overflow

The problem: if an input overflow occurs, data will be lost, ultimately because there is no flow control all the way back to the data source. When data is lost, the receiver should be notified and some sort of graceful recovery should take place, e.g. you shouldn't resume receiving in the middle of a long sysex message.

With a lock-free fifo, which is pretty much what we're stuck with to enable portability to the Mac, it's tricky for the producer and consumer to synchronously reset the buffer and resume normal operation.

Solution: the buffer managed by PortMidi will be flushed when an overflow occurs. The consumer (Read()) gets an error message (pmBufferOverflow) and ordinary processing resumes as soon as a new message arrives. The remainder of a partial sysex message is not considered to be a "new message" and will be flushed as well.

### SetChannelMask(mask as integer) as integer

Plugin Version: 5.2 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Filters incoming messages based on channel.

#### Example:

```
dim s as PortMidiStreamMBS // your midi stream
```

```
call s.SetChannelMask(1+4) // Channel 1 and 3.
```

#### Notes:

The mask is a 16-bit bitfield corresponding to appropriate channels

All channels are allowed by default.

Returns an error code.

**SetFilter(filters as integer) as integer**

Plugin Version: 5.2 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** SetFilter() sets filters on an open input stream to drop selected input types.

**Notes:**

By default, only active sensing messages are filtered.

To prohibit, say, active sensing and sysex messages, call SetFilter(FilterActive + FilterSysEx);

Filtering is useful when midi routing or midi thru functionality is being provided by the user application. For example, you may want to exclude timing messages (clock, MTC, start/stop/continue), while allowing note-related messages to pass.

Or you may be using a sequencer or drum-machine for MIDI clock information but want to exclude any notes it may play.

Returns an error code.

**Write(data as PortMidiEventMBS) as integer**

Plugin Version: 5.2 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Writes midi data from a buffer.

**Notes:**

This may contain:

- short messages or
- sysex messages that are converted into a sequence of PortMidiStreamMBS objects, e.g. sending data from a file or forwarding them from midi input.

Use WriteSysEx() to write a sysex message stored as a contiguous array of bytes.

Sysex data may contain embedded real-time messages.

Returns an error code.

**WriteShort(When as integer, message as integer) as integer**

Plugin Version: 5.2 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Writes a timestamped non-system-exclusive midi message.

**Notes:** Messages are delivered in order as received, and timestamps must be non-decreasing. (But timestamps are ignored if the stream was opened with latency = 0.)

**WriteSysEx(When as integer, message as memoryblock, offset as integer) as integer**

Plugin Version: 9.1 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Writes a timestamped system-exclusive midi message.

**Notes:**

The data is taken from the memoryblock at the given offset.

The message must be 0 terminated.

This message must be valid and contain the special start value and a EOX value on the end.

See also:

- 3.1.1 WriteSysEx(When as integer, message as string) as integer 64

**WriteSysEx(When as integer, message as string) as integer**

Plugin Version: 5.2 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Writes a timestamped system-exclusive midi message.

**Notes:** This message must be valid and contain the special start value and a EOX value on the end.

See also:

- 3.1.1 WriteSysEx(When as integer, message as memoryblock, offset as integer) as integer 64

**3.1.2 Properties****ChannelMask as Integer**

Plugin Version: 5.2 Console & Web: No Mac: Yes, Win: Yes, Linux: Yes, . **Function:** The channel mask used.

**Notes:**

Use SetChannelMask to change it.  
(Read only property)

### Filters as Integer

Plugin Version: 5.2 Console & Web: No Mac: Yes, Win: Yes, Linux: Yes, . **Function:** The filters setting.  
**Notes:**

Use SetFilters to change it.  
(Read only property)

### 3.1.3 Constants

#### FilterActive = & h4000

Plugin Version: 9.4 Console & Web: No Mac: Yes, Win: Yes, Linux: Yes, . **Function:** One of the PortMidi filter constants.

**Notes:** filter active sensing messages (& hFE)

#### FilterAftertouch = & h6000000

Plugin Version: 9.4 Console & Web: No Mac: Yes, Win: Yes, Linux: Yes, . **Function:** One of the PortMidi filter constants.

**Notes:** filter both channel and poly aftertouch

#### FilterChannelAftertouch = & h20000000

Plugin Version: 9.4 Console & Web: No Mac: Yes, Win: Yes, Linux: Yes, . **Function:** One of the PortMidi filter constants.

**Notes:** filter channel aftertouch (most midi controllers use this) (& hD0-& hDF)

**FilterClock & h1D00**

Plugin Version: 9.4 Console & Web: No Mac: Yes, Win: Yes, Linux: Yes, . **Function:** One of the PortMidi filter constants.

**Notes:** filter clock messages (CLOCK & hF8, START & hFA, STOP & hFC, and CONTINUE & hFB)

**FilterControl & h8000000**

Plugin Version: 9.4 Console & Web: No Mac: Yes, Win: Yes, Linux: Yes, . **Function:** One of the PortMidi filter constants.

**Notes:** Control Changes (CC's) (& hB0-& hBF)

**FilterFD = & h2000**

Plugin Version: 9.4 Console & Web: No Mac: Yes, Win: Yes, Linux: Yes, . **Function:** One of the PortMidi filter constants.

**Notes:** filter undefined FD messages

**FilterMTC = 2**

Plugin Version: 9.4 Console & Web: No Mac: Yes, Win: Yes, Linux: Yes, . **Function:** One of the PortMidi filter constants.

**Notes:** MIDI Time Code (& hF1)

**FilterNote = & h3000000**

Plugin Version: 9.4 Console & Web: No Mac: Yes, Win: Yes, Linux: Yes, . **Function:** One of the PortMidi filter constants.

**Notes:** filter note-on and note-off (& h90-& h9F and & h80-& h8F)

**FilterPitchBend = & h4000000**

Plugin Version: 9.4 Console & Web: No Mac: Yes, Win: Yes, Linux: Yes, . **Function:** One of the PortMidi filter constants.

**Notes:** Pitch Bender (& hE0-& hEF)

**FilterPlay = & h400**

Plugin Version: 9.4 Console & Web: No Mac: Yes, Win: Yes, Linux: Yes, . **Function:** One of the PortMidi filter constants.

**Notes:** filter play messages (start & hFA, stop & hFC, continue & hFB)

**FilterPolyAftertouch = & h4000000**

Plugin Version: 9.4 Console & Web: No Mac: Yes, Win: Yes, Linux: Yes, . **Function:** One of the PortMidi filter constants.

**Notes:** per-note aftertouch (& hA0-& hAF)

**FilterProgram = & h10000000**

Plugin Version: 9.4 Console & Web: No Mac: Yes, Win: Yes, Linux: Yes, . **Function:** One of the PortMidi filter constants.

**Notes:** Program changes (& hC0-& hCF)

**FilterRealTime = & hFF01**

Plugin Version: 9.4 Console & Web: No Mac: Yes, Win: Yes, Linux: Yes, . **Function:** One of the PortMidi filter constants.

**Notes:** filter all real-time messages

**FilterReset = & h8000**

Plugin Version: 9.4 Console & Web: No Mac: Yes, Win: Yes, Linux: Yes, . **Function:** One of the PortMidi filter constants.

**Notes:** filter reset messages (& hFF)

**FilterSongPosition = 4**

Plugin Version: 9.4 Console & Web: No Mac: Yes, Win: Yes, Linux: Yes, . **Function:** One of the PortMidi filter constants.

**Notes:** Song Position (& hF2)

**FilterSongSelect = 8**

Plugin Version: 9.4 Console & Web: No Mac: Yes, Win: Yes, Linux: Yes, . **Function:** One of the PortMidi filter constants.

**Notes:** Song Select (& hF3)

**FilterSysEx = 1**

Plugin Version: 9.4 Console & Web: No Mac: Yes, Win: Yes, Linux: Yes, . **Function:** One of the PortMidi filter constants.

**Notes:** filter system exclusive messages (& hF0)

**FilterSystemCommon = & h4E**

Plugin Version: 9.4 Console & Web: No Mac: Yes, Win: Yes, Linux: Yes, . **Function:** One of the PortMidi filter constants.

**Notes:** All System Common messages (mtc, song position, song select, tune request)

**FilterTick = & h200**

Plugin Version: 9.4 Console & Web: No Mac: Yes, Win: Yes, Linux: Yes, . **Function:** One of the PortMidi filter constants.

**Notes:** filter tick messages (& hF9)

**FilterTune = & h40**

Plugin Version: 9.4 Console & Web: No Mac: Yes, Win: Yes, Linux: Yes, . **Function:** One of the PortMidi filter constants.

**Notes:** Tuning request (& hF6)

**FilterUndefined = & h2000**

Plugin Version: 9.4 Console & Web: No Mac: Yes, Win: Yes, Linux: Yes, . **Function:** One of the PortMidi filter constants.

**Notes:** filter undefined real-time messages

## 3.2 class PortMidiDeviceInfoMBS

**class PortMidiDeviceInfoMBS**

Plugin Version: 5.2 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** A class for device information.

### 3.2.1 Properties

**HasInput as Boolean**

Plugin Version: 5.2 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** True if input is available.

**Notes:** (Read only property)

### HasOutput as Boolean

Plugin Version: 5.2 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** True if output is available.

**Notes:** (Read only property)

### InterfaceName as String

Plugin Version: 5.2 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Underlying MIDI API.

**Notes:**

e.g. MMSystem, DirectX or CoreMidi.  
(Read only property)

### Name as String

Plugin Version: 5.2 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** The device name.

**Notes:**

e.g. USB MidiSport 1x1  
(Read only property)

## 3.3 class PortMidiMBS

### class PortMidiMBS

Plugin Version: 5.2 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** The PortMidi library wrapped for use in Realbasic.

**Notes:**

Error codes:

```

const pmNoError          = 0
const pmHostError       = -10000
const pmInvalidDeviceId = -9999
const pmInsufficientMemory = -9998
const pmBufferTooSmall  = -9997
const pmBufferOverflow  = -9996
const pmBadPtr          = -9995
const pmBadData         = -9994
const pmInternalError   = -9993
const pmBufferMaxSize   = -9992

```

Requires libasound.so.2 on Linux to be installed.

### 3.3.1 Methods

#### CountDevices as integer

Plugin Version: 5.2 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Counts the devices.

**Example:**

```

dim pa as new PortMidiMBS

Dim u as integer = pa.CountDevices-1
for i as integer = 0 to u
dim d as PortMidiDeviceInfoMBS = pa.DeviceInfo(i)
MsgBox d.Name+", "+D.InterfaceName
next

```

**Notes:** Returns 0 on any error.

#### DefaultInputDeviceID as integer

Plugin Version: 5.2 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Returns the default device ID or pmNoDevice (-1) if there are no devices.

**Notes:**

On the PC, the user can specify a default device by setting an environment variable. For example, to use device # 1.

```
set PM_RECOMMENDED_OUTPUT_DEVICE=1
```

The user should first determine the available device ID by using the supplied application "testin" or "testout".

In general, the registry is a better place for this kind of info, and with USB devices that can come and go, using integers is not very reliable for device identification. Under Windows, if `PM_RECOMMENDED_OUTPUT_DEVICE` (or `PM_RECOMMENDED_INPUT_DEVICE`) is *\*NOT\** found in the environment, then the default device is obtained by looking for a string in the registry under: `HKEY_LOCAL_MACHINE/SOFTWARE/PortMidi/Recommended_Input_Device` and `HKEY_LOCAL_MACHINE/SOFTWARE/PortMidi/Recommended_Output_Device` for a string. The number of the first device with a substring that matches the string exactly is returned. For example, if the string in the registry is "USB", and device 1 is named "In USB MidiSport 1x1", then that will be the default input because it contains the string "USB".

In addition to the name, `PmDeviceInfo` has the member "interf", which is the interface name. (The "interface" is the underlying software system or API used by PortMidi to access devices. Examples are `MMSystem`, `DirectX` (not implemented), `ALSA`, `OSS` (not implemented), etc.) At present, the only Win32 interface is "MMSystem", the only Linux interface is "ALSA", and the only Mac OS X interface is "CoreMIDI".

To specify both the interface and the device name in the registry, separate the two with a comma and a space, e.g.:

```
MMSystem, In USB MidiSport 1x1
```

In this case, the string before the comma must be a substring of the "interf" string, and the string after the space must be a substring of the "name" name string in order to match the device.

Note: in the current release, the default is simply the first device (the input or output device with the lowest `PmDeviceID`).

### DefaultOutputDeviceID as integer

Plugin Version: 5.2 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Returns the default device ID or `pmNoDevice` (-1) if there are no devices.

#### Notes:

On the PC, the user can specify a default device by setting an environment variable. For example, to use device # 1.

```
set PM_RECOMMENDED_OUTPUT_DEVICE=1
```

The user should first determine the available device ID by using the supplied application "testin" or "testout".

In general, the registry is a better place for this kind of info, and with USB devices that can come and go, using integers is not very reliable for device identification. Under Windows, if `PM_RECOMMENDED_OUTPUT_DEVICE` (or `PM_RECOMMENDED_INPUT_DEVICE`) is *\*NOT\** found in the environment, then the default device is obtained by looking for a string in the registry under: `HKEY_LOCAL_MACHINE/SOFTWARE/PortMidi/Recommended_Input_Device` and `HKEY_LOCAL_MACHINE/SOFTWARE/PortMidi/Recommended_Output_Device` for a string. The number of the first device with a substring that matches the string exactly is returned. For example, if the string in the registry is "USB", and device 1 is named "In USB MidiSport 1x1", then that will be the default input because it contains the string "USB".

In addition to the name, `PmDeviceInfo` has the member "interf", which is the interface name. (The "interface" is the underlying software system or API used by PortMidi to access devices. Examples are `MMSystem`, `DirectX` (not implemented), `ALSA`, `OSS` (not implemented), etc.) At present, the only Win32 interface is "MMSystem", the only Linux interface is "ALSA", and the only Mac OS X interface is "CoreMIDI".

To specify both the interface and the device name in the registry, separate the two with a comma and a space, e.g.:

```
MMSystem, In USB MidiSport 1x1
```

In this case, the string before the comma must be a substring of the "interf" string, and the string after the space must be a substring of the "name" name string in order to match the device.

Note: in the current release, the default is simply the first device (the input or output device with the lowest `PmDeviceID`).

### DeviceInfo(DeviceID as integer) as PortMidiDeviceInfoMBS

Plugin Version: 5.2 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Returns information about a certain device.

#### Example:

```
dim pa as new PortMidiMBS
```

```
Dim u as integer = pa.CountDevices-1
```

```
for i as integer = 0 to u
```

```
dim d as PortMidiDeviceInfoMBS = pa.DeviceInfo(i)
```

```
MsgBox d.Name+", "+D.InterfaceName
```

```
next
```

**Notes:** Returns nil on any error.

### **ErrorText(ErrorNumber as integer) as string**

Plugin Version: 5.2 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** The error text for the given error code.

**Notes:** Returns "" on any error.

### **ReInitialize as integer**

Plugin Version: 9.4 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Shuts down PortMidi and initializes it again.

**Notes:**

As PortMidi does not recognize the attachment of new MIDI devices, you can only reinitialize.

Returns a PortMidi error code.

## **3.3.2 Constants**

### **pmBadData = -9994**

Plugin Version: 9.4 Console & Web: No Mac: Yes, Win: Yes, Linux: Yes, . **Function:** One of the PortMidi errors.

**Notes:** illegal midi data, e.g. missing EOX

### **pmBadPtr = -9995**

Plugin Version: 9.4 Console & Web: No Mac: Yes, Win: Yes, Linux: Yes, . **Function:** One of the PortMidi errors.

**pmBufferMaxSize = -9992**

Plugin Version: 9.4 Console & Web: No Mac: Yes, Win: Yes, Linux: Yes, . **Function:** One of the PortMidi errors.

**Notes:** buffer is already as large as it can be.

**pmBufferOverflow = -9996**

Plugin Version: 9.4 Console & Web: No Mac: Yes, Win: Yes, Linux: Yes, . **Function:** One of the PortMidi errors.

**pmBufferTooSmall = -9997**

Plugin Version: 9.4 Console & Web: No Mac: Yes, Win: Yes, Linux: Yes, . **Function:** One of the PortMidi errors.

**pmHostError = -10000**

Plugin Version: 9.4 Console & Web: No Mac: Yes, Win: Yes, Linux: Yes, . **Function:** One of the PortMidi errors.

**pmInsufficientMemory = -9998**

Plugin Version: 9.4 Console & Web: No Mac: Yes, Win: Yes, Linux: Yes, . **Function:** One of the PortMidi errors.

**pmInternalError = -9993**

Plugin Version: 9.4 Console & Web: No Mac: Yes, Win: Yes, Linux: Yes, . **Function:** One of the PortMidi errors.

**pmInvalidDeviceId = -9999**

Plugin Version: 9.4 Console & Web: No Mac: Yes, Win: Yes, Linux: Yes, . **Function:** One of the PortMidi errors.

**Notes:** out of range or output device when input is requested or input device when output is requested or device is already opened.

**pmNoDevice = -1**

Plugin Version: 9.4 Console & Web: No Mac: Yes, Win: Yes, Linux: Yes, . **Function:** One of the PortMidi errors.

**pmNoError = 0**

Plugin Version: 9.4 Console & Web: No Mac: Yes, Win: Yes, Linux: Yes, . **Function:** One of the PortMidi errors.

## 3.4 class PortMidiEventMBS

**class PortMidiEventMBS**

Plugin Version: 5.2 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** The class for a piece of midi data.

### 3.4.1 Methods

**Set(status as integer, data1 as integer, data2 as integer)**

Plugin Version: 5.2 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Sets the value to the given data.

**SetRaw(data0 as integer, data1 as integer, data2 as integer, data3 as integer)**

Plugin Version: 11.0 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Sets the raw data by setting all 4 bytes together.

### 3.4.2 Properties

#### Data1 as Integer

Plugin Version: 5.2 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** The first data value in the midi event.

**Notes:** (Read and Write property)

#### Data2 as Integer

Plugin Version: 5.2 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** The second data value in the midi event.

**Notes:** (Read and Write property)

#### RawData0 as Integer

Plugin Version: 11.0 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** The first byte of the raw data in this event.

**Notes:**

Same as Status property.  
(Read and Write property)

#### RawData1 as Integer

Plugin Version: 11.0 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** The second byte of the raw data in this event.

**Notes:**

Same as Data1 property.  
(Read and Write property)

### **RawData2 as Integer**

Plugin Version: 11.0 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** The third byte of the raw data in this event.

**Notes:**

Same as Data2 property.  
(Read and Write property)

### **RawData3 as Integer**

Plugin Version: 11.0 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** The forth byte of the raw data in this event.

**Notes:** (Read and Write property)

### **RawMessage as Integer**

Plugin Version: 5.4 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** The raw content of the event as a integer.

**Notes:**

Take care about platform differences like the byte order.  
(Read and Write property)

### **Status as Integer**

Plugin Version: 5.2 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** The status value.

**Notes:** (Read and Write property)

### When as Integer

Plugin Version: 5.2 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** The time value of this midi data.

**Notes:**

Should be milliseconds.  
(Read and Write property)

## 3.5 class WindowsMidiOutputMBS

### class WindowsMidiOutputMBS

Plugin Version: 6.1 Console & Web: Yes Mac: No, Win: Yes, Linux: No, . **Function:** A class to represent a Midi Output device.

**Notes:** Subclass of the WindowsMidiMBS class.

### 3.5.1 Methods

#### Close

Plugin Version: 6.1 Console & Web: Yes Mac: No, Win: Yes, Linux: No, . **Function:** Closes the output device.

**Notes:**

Closes the device with waiting till device is done.  
Handle is set to 0 and lasterror is set.

#### Open(DeviceID as integer)

Plugin Version: 6.1 Console & Web: Yes Mac: No, Win: Yes, Linux: No, . **Function:** Opens the midi device with the given index.

**Notes:**

DeviceID is from 0 to NumberOfMidiOutputDevices-1.

Lasterror is set.

### OpenDefault

Plugin Version: 6.1 Console & Web: Yes Mac: No, Win: Yes, Linux: No, . **Function:** Opens the Midi mapper which opens the device the user selected as the default midi device.

**Notes:**

If only one midi output device is available this one is opened.

On success the handle property is not zero.

Lasterror is set.

### OutputErrorText(errorcode as integer) as string

Plugin Version: 6.1 Console & Web: Yes Mac: No, Win: Yes, Linux: No, . **Function:** Translates an error number into a human readable text.

**Notes:**

Returns "" on unknown errors.

String returned has Windows ANSI text encoding.

### Reset

Plugin Version: 6.1 Console & Web: Yes Mac: No, Win: Yes, Linux: No, . **Function:** Resets the output device.

**Notes:** Lasterror is set.

### SendData(data as memoryblock, size as integer)

Plugin Version: 6.1 Console & Web: Yes Mac: No, Win: Yes, Linux: No, . **Function:** Sends sysex data.

**Example:**

```
dim m as MemoryBlock
dim o as WindowsMidiOutputMBS // your midi output
```

```
m=NewMemoryBlock(8)
```

```
m.Byte(0)=& hF0
m.Byte(1)=& h7F
m.Byte(2)=& h7F
m.Byte(3)=& h04
m.Byte(4)=& h01
m.Byte(5)=& h7F
m.Byte(6)=& h7F
m.Byte(7)=& hF7
```

```
o.SendData m
```

### Notes:

Lasterror is set.

size is the size of the memoryblock to use. A wrong value will crash the application.

See also:

- 3.5.1 SendData(data as string)

81

### SendData(data as string)

Plugin Version: 6.1 Console & Web: Yes Mac: No, Win: Yes, Linux: No, . **Function:** Sends sysex data.

**Notes:** Lasterror is set.

See also:

- 3.5.1 SendData(data as memoryblock, size as integer)

80

### SendMessage(message as integer)

Plugin Version: 6.1 Console & Web: Yes Mac: No, Win: Yes, Linux: No, . **Function:** Sends a short midi message immediately.

**Example:**

```
dim o as WindowsMidiOutputMBS // your windows midi output
// & h90 = Note down
// & h43 = the note number
// & h40 = the velocity
o.SendMessage & h404390
```

**Notes:**

The message is stored in one 32bit integer.  
lowest 8 bit is status, second 8 bit is data1, third 8 bit is data2 and highest 8 bit is left 0.

Between sending note on and off messages, you need to leave time for actual playback.  
LastError is set.  
See also:

- 3.5.1 SendMessage(status as integer, data1 as integer, data2 as integer)

82

**SendMessage(status as integer, data1 as integer, data2 as integer)**

Plugin Version: 6.1 Console & Web: Yes Mac: No, Win: Yes, Linux: No, . **Function:** Sends a short midi message immediately.

**Example:**

```
dim o as WindowsMidiOutputMBS // your windows midi output
// & h90 = Note down
// & h3C = the note number
// & h40 = the velocity
o.SendMessage & h90, & h3C, & h40
```

**Notes:**

LastError is set.  
Between sending note on and off messages, you need to leave time for actual playback.  
See also:

- 3.5.1 SendMessage(message as integer)

81

**Volume as integer**

Plugin Version: 6.1 Console & Web: Yes Mac: No, Win: Yes, Linux: No, . **Function:** The volume to be used for this device.

**Example:**

```
dim m as WindowsMidiOutputMBS // your midi output
m.Volume=0 // all silent
m.Volume=& hFFFF0000 // right only
m.Volume=& h0000FFFF // left only or max volume for mono device
m.Volume=& h7FFF7FFF // half volume for both channels
```

**Notes:**

Not all devices can set the volume.  
LastError is set.

The low-order word contains the left-channel volume setting, and the high-order word contains the right-channel setting. A value of & hFFFF represents full volume, and a value of & h0000 is silence.

If a device does not support both left and right volume control, the low-order word of dwVolume specifies the mono volume level, and the high-order word is ignored.  
(Read and Write computed property)

### 3.5.2 Events

**DeviceClose**

Plugin Version: 6.1 Console & Web: No Mac: No, Win: Yes, Linux: No, . **Function:** An event called when the device is closed.

**Notes:** Should be called when you call close or the object dies.

**DeviceDataSent**

Plugin Version: 6.1 Console & Web: No Mac: No, Win: Yes, Linux: No, . **Function:** An event called whenever SysEx data was sent.

### DeviceOpen

Plugin Version: 6.1 Console & Web: No Mac: No, Win: Yes, Linux: No, . **Function:** An event called whenever the device was opened successfully.

### DevicePositionCallback

Plugin Version: 6.1 Console & Web: No Mac: No, Win: Yes, Linux: No, . **Function:** Called when a MEVT\_F\_CALLBACK Midi event is about to be executed.

**Notes:** A way to track progress in playback.

## 3.6 class WindowsMidiStreamMBS

### class WindowsMidiStreamMBS

Plugin Version: 6.1 Console & Web: Yes Mac: No, Win: Yes, Linux: No, . **Function:** A class for a Windows Midi Stream.

#### 3.6.1 Methods

##### Close

Plugin Version: 6.1 Console & Web: Yes Mac: No, Win: Yes, Linux: No, . **Function:** Closes the Midi stream.

##### Open(DeviceID as integer)

Plugin Version: 6.1 Console & Web: Yes Mac: No, Win: Yes, Linux: No, . **Function:** Opens a MIDI stream for output.

**Notes:**

By default, the device is opened in paused mode.

Lasterror is set.

DeviceID: The device is opened on behalf of the stream and closed again when the stream is closed.

### Pause

Plugin Version: 6.1 Console & Web: Yes Mac: No, Win: Yes, Linux: No, . **Function:** Pauses playback of a specified MIDI stream.

**Notes:** Lasterror is set.

### PositionBytes as integer

Plugin Version: 6.1 Console & Web: Yes Mac: No, Win: Yes, Linux: No, . **Function:** Position of the stream in bytes.

**Notes:** Lasterror is set.

### PositionMS as integer

Plugin Version: 6.1 Console & Web: Yes Mac: No, Win: Yes, Linux: No, . **Function:**

Position of the stream in milliseconds.

**Notes:** Lasterror is set.

### PositionSamples as integer

Plugin Version: 6.1 Console & Web: Yes Mac: No, Win: Yes, Linux: No, . **Function:** Position of the stream in samples.

**Notes:** Lasterror is set.

**PositionTicks as integer**

Plugin Version: 6.1 Console & Web: Yes Mac: No, Win: Yes, Linux: No, . **Function:** Position of the stream in ticks.

**Notes:** Lasterror is set.

**Restart**

Plugin Version: 6.1 Console & Web: Yes Mac: No, Win: Yes, Linux: No, . **Function:** The Restart function restarts a paused MIDI stream.

**Notes:** Lasterror is set.

**SendMessage(message as integer)**

Plugin Version: 6.1 Console & Web: Yes Mac: No, Win: Yes, Linux: No, . **Function:** Sends a short midi message immediately.

**Example:**

```
dim o as WindowsMidiStreamMBS // your midi stream
```

```
// & h90 = Note down
// & h43 = the note number
// & h40 = the velocity
o.SendMessage & h404390
```

**Notes:**

The message is stored in one 32bit integer.

lowest 8 bit is status, second 8 bit is data1, third 8 bit is data2 and highest 8 bit is left 0.

Between sending note on and off messages, you need to leave time for actual playback.

Lasterror is set.

See also:

- 3.6.1 SendMessage(status as integer, data1 as integer, data2 as integer)

**SendMessage(status as integer, data1 as integer, data2 as integer)**

Plugin Version: 6.1 Console & Web: Yes Mac: No, Win: Yes, Linux: No, . **Function:** Sends a short midi message immediately.

**Example:**

```
dim o as WindowsMidiStreamMBS // your midi stream

// & h90 = Note down
// & h3C = the note number
// & h40 = the velocity
o.SendMessage & h90, & h3C, & h40
```

**Notes:**

Lasterror is set.

Between sending note on and off messages, you need to leave time for actual playback.

See also:

- 3.6.1 SendMessage(message as integer)

86

**Stop**

Plugin Version: 6.1 Console & Web: Yes Mac: No, Win: Yes, Linux: No, . **Function:** The Stop function turns off all notes on all MIDI channels for the specified MIDI output device.

**Notes:** Lasterror is set.

**Tempo as integer**

Plugin Version: 6.1 Console & Web: Yes Mac: No, Win: Yes, Linux: No, . **Function:** Tempo of the stream, in microseconds per quarter note.

**Notes:**

The tempo is honored only if the time division for the stream is specified in quarter note format.

Lasterror is set.

(Read and Write computed property)

**TimeDiv as integer**

Plugin Version: 6.1 Console & Web: Yes Mac: No, Win: Yes, Linux: No, . **Function:** Time division for this stream, in the format specified in the Standard MIDI Files 1.0 specification.

**Notes:**

The low 16 bits of this integer value contain the time division.

Lasterror is set.

(Read and Write computed property)

**Volume as integer**

Plugin Version: 6.1 Console & Web: Yes Mac: No, Win: Yes, Linux: No, . **Function:** The volume to be used for this stream.

**Example:**

```
dim m as WindowsMidiStreamMBS // your midi stream
```

```
m.Volume=0 // all silent
```

```
m.Volume=& hFFFF0000 // right only
```

```
m.Volume=& h0000FFFF // left only or max volume for mono device
```

```
m.Volume=& h7FFF7FFF // half volume for both channels
```

**Notes:**

Not all devices can set the volume.

Lasterror is set.

The low-order word contains the left-channel volume setting, and the high-order word contains the right-channel setting. A value of & hFFFF represents full volume, and a value of & h0000 is silence.

If a device does not support both left and right volume control, the low-order word of dwVolume specifies the mono volume level, and the high-order word is ignored.

Lasterror is set.

(Read and Write computed property)

### 3.6.2 Properties

#### Handle as Integer

Plugin Version: 6.1 Console & Web: Yes Mac: No, Win: Yes, Linux: No, . **Function:** The handle for the midi stream.

**Notes:**

Type is HMIDISTRM.  
(Read only property)

#### Lasterror as Integer

Plugin Version: 6.1 Console & Web: Yes Mac: No, Win: Yes, Linux: No, . **Function:** The last error code reported by one of the class functions.

**Notes:** (Read only property)

## 3.7 class WindowsMidiInputMBS

### class WindowsMidiInputMBS

Plugin Version: 6.1 Console & Web: Yes Mac: No, Win: Yes, Linux: No, . **Function:** A class to represent an open Midi Input device in Realbasic.

**Notes:** Subclass of the WindowsMidiMBS class.

### 3.7.1 Methods

#### Close

Plugin Version: 6.1 Console & Web: Yes Mac: No, Win: Yes, Linux: No, . **Function:** Closes the device.

**Notes:**

First stops recording, second resets midi output device and third Closes the device with waiting till device is done.

Handle is set to 0 and lasterror is set.

### Idle

Plugin Version: 9.7 Console & Web: Yes Mac: No, Win: Yes, Linux: No, . **Function:** Processes events.

**Notes:**

Midi events are buffered in data structures. This method dispatches them to the Realbasic event handlers. Call this method as often as you need events to fire. For example every 100ms in a timer.

same as WindowsMidiMBS.Idle

### InputErrorText(errorcode as integer) as string

Plugin Version: 6.1 Console & Web: Yes Mac: No, Win: Yes, Linux: No, . **Function:** Translates an error number into a human readable text.

**Notes:**

Returns "" on unknown errors.  
String returned has Windows ANSI text encoding.

### Open(DeviceID as integer, BufferSize as integer)

Plugin Version: 6.1 Console & Web: Yes Mac: No, Win: Yes, Linux: No, . **Function:** Opens a Midi device.

**Notes:**

DeviceID is from 0 to NumberOfMidiInputDevices-1.  
Buffersize is the maximum size to allocate for each SysEx receive buffer.  
Minimum is 256 bytes. Windows does not handle SysEx messages bigger than 64K.  
Lasterror is set.  
On success the handle property is non zero.

### Reset

Plugin Version: 6.1 Console & Web: Yes Mac: No, Win: Yes, Linux: No, . **Function:** Resets the device.  
**Notes:** Lasterror is set.

### Start

Plugin Version: 6.1 Console & Web: Yes Mac: No, Win: Yes, Linux: No, . **Function:** Starts listening for events.  
**Notes:** Lasterror is set.

### Stop

Plugin Version: 6.1 Console & Web: Yes Mac: No, Win: Yes, Linux: No, . **Function:** Stops recording.  
**Notes:**

Lasterror is set.  
You should not need this and just call close or let the object die.

## 3.7.2 Events

### DeviceClose

Plugin Version: 6.1 Console & Web: No Mac: No, Win: Yes, Linux: No, . **Function:** An event called whenever the output device is closed.  
**Notes:** Called when you call close.

### DeviceData(timestamp as integer, status as integer, data1 as integer, data2 as integer)

Plugin Version: 6.1 Console & Web: No Mac: No, Win: Yes, Linux: No, . **Function:** A simple Midi message has been received.  
**Notes:**

Process this event fast to avoid losing events.

Status, Data1 and Data2 are all 8 bit values.

The time stamp is specified in milliseconds, beginning at zero when the Start function was called.

#### **DeviceError(timestamp as integer, status as integer, data1 as integer, data2 as integer)**

Plugin Version: 6.1 Console & Web: No Mac: No, Win: Yes, Linux: No, . **Function:** A bad Midi message has been received.

##### **Notes:**

Process this event fast to avoid losing events.

Status, Data1 and Data2 are all 8 bit values.

The time stamp is specified in milliseconds, beginning at zero when the Start function was called.

#### **DeviceLongData(timestamp as integer, data as string)**

Plugin Version: 6.1 Console & Web: No Mac: No, Win: Yes, Linux: No, . **Function:** SysEx data was received.

##### **Notes:**

If data is "" no free buffer was available to store the data.

The time stamp is specified in milliseconds, beginning at zero when the Start function was called.

#### **DeviceLongError(timestamp as integer, data as string)**

Plugin Version: 6.1 Console & Web: No Mac: No, Win: Yes, Linux: No, . **Function:** Bad SysEx data was received.

##### **Notes:**

If data is "" no free buffer was available to store the data.

The time stamp is specified in milliseconds, beginning at zero when the Start function was called.

### DeviceOpen

Plugin Version: 6.1 Console & Web: No Mac: No, Win: Yes, Linux: No, . **Function:** An event called whenever the output device is opened.

**Notes:** Called when you call open.

## 3.8 class WindowsMidiInputInfoMBS

### class WindowsMidiInputInfoMBS

Plugin Version: 6.1 Console & Web: Yes Mac: No, Win: Yes, Linux: No, . **Function:** A class for information about a certain Midi Device.

### 3.8.1 Properties

#### DriverVersion as Integer

Plugin Version: 6.1 Console & Web: Yes Mac: No, Win: Yes, Linux: No, . **Function:** Version number of the device driver for the MIDI input device.

**Notes:**

The high-order byte is the major version number, and the low-order byte is the minor version number.  
(Read only property)

#### Flags as Integer

Plugin Version: 6.1 Console & Web: Yes Mac: No, Win: Yes, Linux: No, . **Function:** Some flags.

**Notes:**

Currently unused in Windows XP.  
(Read only property)

**ManufacturerID as Integer**

Plugin Version: 6.1 Console & Web: Yes Mac: No, Win: Yes, Linux: No, . **Function:** Manufacturer identifier of the device driver for the MIDI input device.

**Notes:** (Read only property)

**Name as String**

Plugin Version: 6.1 Console & Web: Yes Mac: No, Win: Yes, Linux: No, . **Function:** Product name.

**Notes:**

Currently an ANSI string.  
(Read only property)

**ProductID as Integer**

Plugin Version: 6.1 Console & Web: Yes Mac: No, Win: Yes, Linux: No, . **Function:** Product identifier of the MIDI input device.

**Notes:** (Read only property)

### 3.9 class WindowsMidiOutputInfoMBS

**class WindowsMidiOutputInfoMBS**

Plugin Version: 6.1 Console & Web: Yes Mac: No, Win: Yes, Linux: No, . **Function:** A class for information about a certain Midi Device.

### 3.9.1 Properties

#### ChannelMask as Integer

Plugin Version: 6.1 Console & Web: Yes Mac: No, Win: Yes, Linux: No, . **Function:** Channels that an internal synthesizer device responds to, where the least significant bit refers to channel 0 and the most significant bit to channel 15.

**Notes:**

Port devices that transmit on all channels set this member to & hFFFF.  
(Read only property)

#### DriverVersion as Integer

Plugin Version: 6.1 Console & Web: Yes Mac: No, Win: Yes, Linux: No, . **Function:** Version number of the device driver for the MIDI input device.

**Notes:**

The high-order byte is the major version number, and the low-order byte is the minor version number.  
(Read only property)

#### Flags as Integer

Plugin Version: 6.1 Console & Web: Yes Mac: No, Win: Yes, Linux: No, . **Function:** Optional functionality supported by the device.

**Notes:**

It can be one or more of the following:

MIDICAPS_ CACHE	= 4	Supports patch caching.
MIDICAPS_ LRVOLUME	= 2	Supports separate left and right volume control.
MIDICAPS_ STREAM	= 8	Provides direct support for the midiStreamOut function.
MIDICAPS_ VOLUME	= 1	Supports volume control.

If a device supports volume changes, the MIDICAPS\_ VOLUME flag will be set for the dwSupport member. If a device supports separate volume changes on the left and right channels, both the MIDICAPS\_ VOLUME and the MIDICAPS\_ LRVOLUME flags will be set for this member.

(Read only property)

**ManufacturerID as Integer**

Plugin Version: 6.1 Console & Web: Yes Mac: No, Win: Yes, Linux: No, . **Function:** Manufacturer identifier of the device driver for the MIDI input device.

**Notes:** (Read only property)

**Name as String**

Plugin Version: 6.1 Console & Web: Yes Mac: No, Win: Yes, Linux: No, . **Function:** Product name.

**Notes:**

Currently an ANSI string.

(Read only property)

**Notes as Integer**

Plugin Version: 6.1 Console & Web: Yes Mac: No, Win: Yes, Linux: No, . **Function:** Maximum number of simultaneous notes that can be played by an internal synthesizer device.

**Notes:**

If the device is a port, this member is not meaningful and is set to 0.

(Read only property)

**ProductID as Integer**

Plugin Version: 6.1 Console & Web: Yes Mac: No, Win: Yes, Linux: No, . **Function:** Product identifier of the MIDI input device.

**Notes:** (Read only property)

**Technology as Integer**

Plugin Version: 6.1 Console & Web: Yes Mac: No, Win: Yes, Linux: No, . **Function:** Type of the MIDI output device.

**Notes:**

This value can be one of the following:

```

MOD_ MIDIPORT      = 1  // output port
MOD_ SYNTH         = 2  // generic internal synth
MOD_ SQSYNTH      = 3  // square wave internal synth
MOD_ FMSYNTH      = 4  // FM internal synth
MOD_ MAPPER       = 5  // MIDI mapper
MOD_ WAVETABLE    = 6  // hardware wavetable synth
MOD_ SWSYNTH     = 7  // software synth

```

(Read only property)

### Voices as Integer

Plugin Version: 6.1 Console & Web: Yes Mac: No, Win: Yes, Linux: No, . **Function:** Number of voices supported by an internal synthesizer device.

**Notes:**

If the device is a port, this member is not meaningful and is set to 0.

(Read only property)

### Volume as Boolean

Plugin Version: 6.1 Console & Web: Yes Mac: No, Win: Yes, Linux: No, . **Function:** Whether volume control is available.

**Notes:**

True if yes and False if no.

(Read only property)

### VolumeStereo as Boolean

Plugin Version: 6.1 Console & Web: Yes Mac: No, Win: Yes, Linux: No, . **Function:** Whether the device can control volume on two independent channels.

**Notes:**

True if stereo, False if mono.

(Read only property)

### 3.10 class WindowsMidiMBS

#### class WindowsMidiMBS

Plugin Version: 6.1 Console & Web: Yes Mac: No, Win: Yes, Linux: No, . **Function:** Windows Midi base class.

#### Example:

```

dim midi as WindowsMidiMBS // your midi object

Sub Open()
dim i as WindowsMidiInputInfoMBS
dim o as WindowsMidiOutputInfoMBS
dim c,n as integer

midi=new WindowsMidiMBS

c=midi.NumberOfMidiInputDevices-1

for n=0 to c
i=midi.InputDevice(n)

listbox1.AddRow str(n+1)
listbox1.Cell(listbox1.LastIndex,1)=i.Name
listbox1.Cell(listbox1.LastIndex,2)=hex(i.DriverVersion)

next

c=midi.NumberOfMidiOutputDevices-1

for n=0 to c
o=midi.OutputDevice(n)

listbox2.AddRow str(n+1)
listbox2.Cell(listbox2.LastIndex,1)=o.Name
listbox2.Cell(listbox2.LastIndex,2)=hex(o.DriverVersion)

next

End Sub

```

### 3.10.1 Methods

#### Connect(output as WindowsMidiOutputMBS)

Plugin Version: 6.1 Console & Web: Yes Mac: No, Win: Yes, Linux: No, . **Function:** The Connect function connects a MIDI input device to a MIDI thru or output device, or connects a MIDI thru device to a MIDI output device.

**Notes:**

self must be a MIDI input device or a MIDI thru device.  
output must be the MIDI output or thru device.

After calling this function, the MIDI input device receives event data in an DeviceData event whenever a message with the same event data is sent to the output device driver.

A thru driver is a special form of MIDI output driver. The system will allow only one MIDI output device to be connected to a MIDI input device, but multiple MIDI output devices can be connected to a MIDI thru device. Whenever the given MIDI input device receives event data in an DeviceData event, a message with the same event data is sent to the given output device driver (or through the thru driver to the output drivers).

Lasterror is set.

#### DataLost as integer

Plugin Version: 6.1 Console & Web: Yes Mac: No, Win: Yes, Linux: No, . **Function:** Number of data blocks lost.

**Notes:**

The buffers in the plugins have a certain size.  
In case midi events are coming fast in and the Idle method is not called often enough events are lost.  
In that case increase the frequency of calling Idle or request the buffer size to be increased in the next plugin version.

**Disconnect(output as WindowsMidiOutputMBS)**

Plugin Version: 6.1 Console & Web: Yes Mac: No, Win: Yes, Linux: No, . **Function:** The Disconnect function disconnects a MIDI input device from a MIDI thru or output device, or disconnects a MIDI thru device from a MIDI output device.

**Notes:**

self must be a MIDI input device or a MIDI thru device.

output must be the MIDI output device to be disconnected.

MIDI input, output, and thru devices can be connected by using the Connect function. Thereafter, whenever the MIDI input device receives event data in an DeviceData event, a message with the same event data is sent to the output device driver (or through the thru driver to the output drivers).

Lasterror is set.

**EventsLost as integer**

Plugin Version: 6.1 Console & Web: Yes Mac: No, Win: Yes, Linux: No, . **Function:** Number of events lost.

**Notes:**

The buffers in the plugins have a certain size.

In case midi events are coming fast in and the Idle method is not called often enough events are lost.

In that case increase the frequency of calling Idle or request the buffer size to be increased in the next plugin version.

**Idle**

Plugin Version: 6.1 Console & Web: Yes Mac: No, Win: Yes, Linux: No, . **Function:** Processes events.

**Notes:**

Midi events are buffered in data structures. This method dispatches them to the Realbasic event handlers. Call this method as often as you need events to fire. For example every 100ms in a timer.

**InputDevice(index as integer) as WindowsMidiInputInfoMBS**

Plugin Version: 6.1 Console & Web: Yes Mac: No, Win: Yes, Linux: No, . **Function:** The InputDevice function queries a specified MIDI input device to determine its capabilities.

**Example:**

```
dim midi as WindowsMidiMBS // your midi object
dim n,c as integer
dim i as WindowsMidiInputInfoMBS

c=midi.NumberOfMidiInputDevices-1

for n=0 to c
i=midi.InputDevice(n)

listbox1.AddRow str(n+1)
listbox1.Cell(listbox1.LastIndex,1)=i.Name
listbox1.Cell(listbox1.LastIndex,2)=hex(i.DriverVersion)

next
```

**Notes:** Index is from 0 to NumberOfMidiInputDevices-1.

**NumberOfMidiInputDevices as integer**

Plugin Version: 6.1 Console & Web: Yes Mac: No, Win: Yes, Linux: No, . **Function:** Returns the number of MIDI input devices present in the system.

**Example:**

```
dim midi as new WindowsMidiMBS
MsgBox str(midi.NumberOfMidiInputDevices)
```

**Notes:** A return value of zero means that there are no devices.

**NumberOfMidiOutputDevices as integer**

Plugin Version: 6.1 Console & Web: Yes Mac: No, Win: Yes, Linux: No, . **Function:** Returns the number of MIDI output devices present in the system.

**Example:**

```
dim midi as new WindowsMidiMBS
MsgBox str(midi.NumberOfMidiOutputDevices)
```

**Notes:** A return value of zero means that there are no devices.

**OutputDevice(index as integer) as WindowsMidiOutputInfoMBS**

Plugin Version: 6.1 Console & Web: Yes Mac: No, Win: Yes, Linux: No, . **Function:** The OutputDevice function queries a specified MIDI output device to determine its capabilities.

**Example:**

```
dim midi as WindowsMidiMBS // your midi object
dim c,n as integer
dim o as WindowsMidiOutputInfoMBS

c=midi.NumberOfMidiOutputDevices-1

for n=0 to c
o=midi.OutputDevice(n)

listbox2.AddRow str(n+1)
listbox2.Cell(listbox2.LastIndex,1)=o.Name
listbox2.Cell(listbox2.LastIndex,2)=hex(o.DriverVersion)

next
```

**Notes:** Index is from 0 to NumberOfMidiOutputDevices-1.

### 3.10.2 Properties

#### Handle as Integer

Plugin Version: 6.1 Console & Web: Yes Mac: No, Win: Yes, Linux: No, . **Function:** The handle for this Midi input or output port.

**Notes:**

Depending on which Realbasic class this is, value is a HMIDI, HMIDIIN or HMIDIOUT handle.  
(Read only property)

#### Lasterror as Integer

Plugin Version: 6.1 Console & Web: Yes Mac: No, Win: Yes, Linux: No, . **Function:** The last error code reported.

**Notes:**

0 is no error and -1 is parameter error from the plugin.  
(Read only property)



# Chapter 4

## List of all classes

• PortAudioDeviceInfoMBS	54
• PortAudioHostApiInfoMBS	34
• PortAudioHostErrorInfoMBS	33
• PortAudioMBS	37
• PortAudioStreamBaseMBS	22
• PortAudioStreamBufferedMBS	11
• PortAudioStreamInfoMBS	25
• PortAudioStreamMBS	27
• PortAudioStreamParametersMBS	52
• PortAudioStreamRecorderMBS	47
• PortMidiDeviceInfoMBS	69
• PortMidiEventMBS	76
• PortMidiMBS	70
• PortMidiStreamMBS	59
• WindowsMidiInputInfoMBS	93
• WindowsMidiInputMBS	89
• WindowsMidiMBS	98
• WindowsMidiOutputInfoMBS	94
• WindowsMidiOutputMBS	79
• WindowsMidiStreamMBS	84