

# **Midi Player Tool Kit for Unity**

Thierry Bachmann  
Version 2  
Sun Mar 3 2019

## Table of Contents

Namespace Index .....	1
Hierarchical Index .....	1
Data Structure Index .....	1
MidiPlayerTK.....	2
Data Structure Documentation .....	7
MidiPlayerTK.MidiExternalPlayer .....	7
MidiPlayerTK.MidiFilePlayer .....	17
MidiPlayerTK.MidiFileWriter .....	27
MidiPlayerTK.MidiListPlayer .....	31
MidiPlayerTK.MidiLoad .....	34
MidiPlayerTK.MidiPlayer .....	39
MidiPlayerTK.MidiPlayerGlobal .....	42
MidiPlayerTK.MidiStreamPlayer .....	46
MidiPlayerTK.MidiSynth.....	52
MidiPlayerTK.MidiListPlayer.MPTK_MidiPlayItem.....	55
MidiPlayerTK.MPTKEvent .....	56
MidiPlayerTK.MPTKListItem .....	59
MidiPlayerTK.TrackMidiEvent .....	60
Index .....	60

---

# Namespace Index

## Packages

Here are the packages with brief descriptions (if available):

<a href="#">MidiPlayerTK</a> .....	2
------------------------------------	---

---

# Hierarchical Index

## Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

MidiPlayerTK.MidiFileWriter .....	27
MidiPlayerTK.MidiListPlayer .....	31
MidiPlayerTK.MidiLoad.....	34
MidiPlayerTK.MidiPlayerGlobal .....	42
MidiPlayerTK.MidiSynth.....	52
MidiPlayerTK.MidiPlayer.....	39
MidiPlayerTK.MidiFilePlayer .....	17
MidiPlayerTK.MidiExternalPlayer .....	7
MidiPlayerTK.MidiStreamPlayer .....	46
MidiPlayerTK.MidiListPlayer.MPTK_MidiPlayItem .....	55
MidiPlayerTK.MPTKEvent .....	56
MidiPlayerTK.MPTKListItem .....	59
MidiPlayerTK.TrackMidiEvent .....	60

---

# Data Structure Index

## Data Structures

Here are the data structures with brief descriptions:

<a href="#">MidiPlayerTK.MidiExternalPlayer</a> (PRO Version - Script for the prefab <a href="#">MidiExternalPlayer</a> . See full example TestMidiExternalPlayer.cs with a light sequencer. Play a midi file from a path on the local desktop or from a web site ) .....	7
<a href="#">MidiPlayerTK.MidiFilePlayer</a> (Script for the prefab <a href="#">MidiFilePlayer</a> . Play a selected midi file. List of Midi file must be defined with Midi Player Setup (see Unity menu MPTK). ) .....	17
<a href="#">MidiPlayerTK.MidiFileWriter</a> (PRO Version - Write a midi file from differents sources based on NAudio framework. See full example TestMidiWriter.cs with a light sequencer. ) .....	27
<a href="#">MidiPlayerTK.MidiListPlayer</a> (PRO Version - Script for the prefab <a href="#">MidiListPlayer</a> . Play a list of pre-selected midi file from the dedicated inspector. List of Midi files must exists in MidiDB. See Midi Player Setup (Unity menu MPTK). ) .....	31

<a href="#">MidiPlayerTK.MidiLoad</a> (Base class for loading a Midi file. No sequencer, no synthesizer. Usefull to load all tje Midi events from a Midi. )	34
<a href="#">MidiPlayerTK.MidiPlayer</a> (Send event to the midi synthesizer thru thread. Don't instanciate this class, use rather <a href="#">MidiFilePlayer</a> or <a href="#">MidiStreamPlayer</a> . )	39
<a href="#">MidiPlayerTK.MidiPlayerGlobal</a> (Singleton class to manage all global features of MPTK. )	42
<a href="#">MidiPlayerTK.MidiStreamPlayer</a> (Play generated notes. Any Midi file is necessary rather create music from your own algorithm with <a href="#">MPTK_PlayEvent()</a> . Duration can be set in the <a href="#">MPTKEvent</a> , but a note can also be stopped with <a href="#">MPTK_StopEvent()</a> . )	46
<a href="#">MidiPlayerTK.MidiSynth</a> ()	52
<a href="#">MidiPlayerTK.MidiListPlayer.MPTK_MidiPlayItem</a> (Define a midi to be added in the list )	55
<a href="#">MidiPlayerTK.MPTKEvent</a> (Midi Event class for MPTK. Usage to generate Midi Music with <a href="#">MidiStreamPlayer</a> or to read midi events from a Midi file with <a href="#">MidiLoad</a> or to receive midi events from <a href="#">MidiFilePlayer</a> OnEventNotesMidi. )	56
<a href="#">MidiPlayerTK.MPTKListItem</a> (A list of string with index: midi, preset, bank, drum, ... )	59
<a href="#">MidiPlayerTK.TrackMidiEvent</a> (Midi event list (NAudio format) )	60

---

## Namespace Documentation

### MidiPlayerTK Namespace Reference

#### Data Structures

- class [MidiExternalPlayer](#)  
*PRO Version - Script for the prefab [MidiExternalPlayer](#). See full example [TestMidiExternalPlayer.cs](#) with a light sequencer. Play a midi file from a path on the local deskop or from a web site*
- class [MidiFilePlayer](#)  
*Script for the prefab [MidiFilePlayer](#). Play a selected midi file. List of Midi file must be defined with Midi Player Setup (see Unity menu MPTK).*
- class [MidiFileWriter](#)  
*PRO Version - Write a midi file from differents sources based on NAudio frawemork. See full example [TestMidiWriter.cs](#) with a light sequencer.*
- class [MidiListPlayer](#)  
*PRO Version - Script for the prefab [MidiListPlayer](#). Play a list of pre-selected midi file from the dedicated inspector. List of Midi files must exists in MidiDB. See Midi Player Setup (Unity menu MPTK).*
- class [MidiLoad](#)  
*Base class for loading a Midi file. No sequencer, no synthesizer. Usefull to load all tje Midi events from a Midi.*
- class [MidiPlayer](#)  
*Send event to the midi synthesizer thru thread. Don't instanciate this class, use rather [MidiFilePlayer](#) or [MidiStreamPlayer](#).*
- class [MidiPlayerGlobal](#)  
*Singleton class to manage all global features of MPTK.*
- class [MidiStreamPlayer](#)  
*Play generated notes. Any Midi file is necessary rather create music from your own algorithm with [MPTK\\_PlayEvent\(\)](#). Duration can be set in the [MPTKEvent](#), but a note can also be stopped with [MPTK\\_StopEvent\(\)](#).*
- class [MidiSynth](#)
- class [MPTKEvent](#)

Midi Event class for MPTK. Usage to generate Midi Music with [MidiStreamPlayer](#) or to read midi events from a Midi file with [MidiLoad](#) or to receive midi events from [MidiFilePlayer](#) [OnEventNotesMidi](#).

- class [MPTKListItem](#)  
A list of string with index: midi, preset, bank, drum, ...
- class [TrackMidiEvent](#)  
Midi event list (NAudio format)

## Enumerations

- enum [MPTKCommand](#) : byte { [MPTKCommand.NoteOff](#) = 0x80, [MPTKCommand.NoteOn](#) = 0x90, [MPTKCommand.KeyAfterTouch](#) = 0xA0, [MPTKCommand.ControlChange](#) = 0xB0, [MPTKCommand.PatchChange](#) = 0xC0, [MPTKCommand.ChannelAfterTouch](#) = 0xD0, [MPTKCommand.PitchWheelChange](#) = 0xE0, [MPTKCommand.Sysex](#) = 0xF0, [MPTKCommand.Eox](#) = 0xF7, [MPTKCommand.TimingClock](#) = 0xF8, [MPTKCommand.StartSequence](#) = 0xFA, [MPTKCommand.ContinueSequence](#) = 0xFB, [MPTKCommand.StopSequence](#) = 0xFC, [MPTKCommand.AutoSensing](#) = 0xFE, [MPTKCommand.MetaEvent](#) = 0xFF }
- MIDI command codes enum [MPTKController](#) : byte { [MPTKController.BankSelect](#) = 0, [MPTKController.Modulation](#) = 1, [MPTKController.BreathController](#) = 2, [MPTKController.FootController](#) = 4, [MPTKController.MainVolume](#) = 7, [MPTKController.Pan](#) = 10, [MPTKController.Expression](#) = 11, [MPTKController.BankSelectLsb](#) = 32, [MPTKController.Sustain](#) = 64, [MPTKController.Portamento](#) = 65, [MPTKController.Sostenuto](#) = 66, [MPTKController.SoftPedal](#) = 67, [MPTKController.LegatoFootswitch](#) = 68, [MPTKController.ResetAllControllers](#) = 121, [MPTKController.AllNotesOff](#) = 123, [MPTKController.AllSoundOff](#) = 120 }
- MidiController enumeration <http://www.midi.org/techspecs/midimessages.php#3> enum [MPTKMeta](#) : byte { [MPTKMeta.TrackSequenceNumber](#) = 0x00, [MPTKMeta.TextEvent](#) = 0x01, [MPTKMeta.Copyright](#) = 0x02, [MPTKMeta.SequenceTrackName](#) = 0x03, [MPTKMeta.TrackInstrumentName](#) = 0x04, [MPTKMeta.Lyric](#) = 0x05, [MPTKMeta.Marker](#) = 0x06, [MPTKMeta.CuePoint](#) = 0x07, [MPTKMeta.ProgramName](#) = 0x08, [MPTKMeta.DeviceName](#) = 0x09, [MPTKMeta.MidiChannel](#) = 0x20, [MPTKMeta.MidiPort](#) = 0x21, [MPTKMeta.EndTrack](#) = 0x2F, [MPTKMeta.SetTempo](#) = 0x51, [MPTKMeta.SmpteOffset](#) = 0x54, [MPTKMeta.TimeSignature](#) = 0x58, [MPTKMeta.KeySignature](#) = 0x59, [MPTKMeta.SequencerSpecific](#) = 0x7F }

*MIDI MetaEvent Type*

---

## Enumeration Type Documentation

enum [MidiPlayerTK.MPTKCommand](#) : byte[strong]

MIDI command codes

### Enumerator:

NoteOff	Note Off
NoteOn	Note On
KeyAfterTouch	Key After-touch

ControlChange	Control change
PatchChange	Patch change
ChannelAfterTouch	Channel after-touch
PitchWheelChange	Pitch wheel change
Sysex	Sysex message
Eox	Eox (comes at end of a sysex message)
TimingClock	Timing clock (used when synchronization is required)
StartSequence	Start sequence
ContinueSequence	Continue sequence
StopSequence	Stop sequence
AutoSensing	Auto-Sensing
MetaEvent	Meta-event

enum [MidiPlayerTK.MPTKController](#) : byte [strong]

MidiController enumeration <http://www.midi.org/techspecs/midimessages.php#3>

**Enumerator:**

BankSelect	Bank Select (MSB)
------------	-------------------

Modulation	Modulation (MSB)
BreathController	Breath Controller
FootController	Foot controller (MSB)
MainVolume	Main volume
Pan	Pan
Expression	Expression
BankSelectLsb	Bank Select LSB ** not implemented **
Sustain	Sustain
Portamento	Portamento On/Off
Sostenuto	Sostenuto On/Off
SoftPedal	Soft Pedal On/Off
LegatoFootswitch	Legato Footswitch
ResetAllControllers	Reset all controllers
AllNotesOff	All notes off
AllSoundOff	All sound off

--	--

enum [MidiPlayerTK.MPTKMeta](#) : byte [strong]

MIDI MetaEvent Type

**Enumerator:**

TrackSequenceNumber	Track sequence number
TextEvent	Text event
Copyright	Copyright
SequenceTrackName	Sequence track name
TrackInstrumentName	Track instrument name
Lyric	Lyric
Marker	Marker
CuePoint	Cue point
ProgramName	Program (patch) name
DeviceName	Device (port) name
MidiChannel	MIDI Channel (not official?)
MidiPort	MIDI Port (not official?)



EndTrack	End track
SetTempo	Set tempo
SmppteOffset	SMPTE offset
TimeSignmature	Time signature
KeySignature	Key signature
SequencerSpecific	Sequencer specific

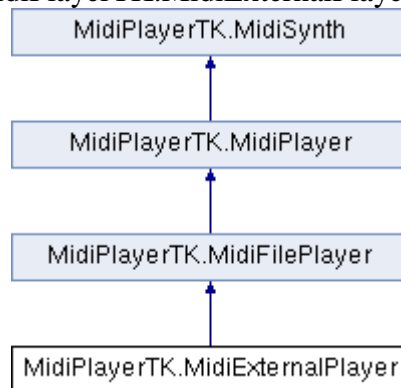
---

## Data Structure Documentation

### MidiPlayerTK.MidiExternalPlayer

PRO Version - Script for the prefab [MidiExternalPlayer](#). See full example TestMidiExternalPlayer.cs with a light sequencer. Play a midi file from a path on the local deskop or from a web site

Inheritance diagram for MidiPlayerTK.MidiExternalPlayer:



### Public Member Functions

- override void [MPTK\\_Play](#) ()  
*Play the midi file defined in MPTK\_MidiName*
- override void [MPTK\\_Next](#) ()  
*Play next Midi - NO EFFECT for external*
- override void [MPTK\\_Previous](#) ()  
*Play previous Midi - NO EFFECT for external*
- virtual void [MPTK\\_Stop](#) ()  
*Stop playing*

- virtual void [MPTK\\_RePlay](#) ()  
*Restart playing of the current midi file*
- virtual void [MPTK\\_Pause](#) (float timeToPauseMS=-1f)  
*Pause the current playing*
- void [MPTK\\_ReSyncTime](#) ()  
*In case of delay in the application, resync is usefull to avoid multi tock play at the same time*
- [MPTKEvent.EnumLength](#) [MPTK\\_NoteLength](#) ([MPTKEvent](#) note)  
*Return note length as [https://en.wikipedia.org/wiki/Note\\_value](https://en.wikipedia.org/wiki/Note_value)*
- void [MPTK\\_InitSynth](#) (int channelCount=16)  
*Init the synthesizer. Prefabs automatically initialize the synthesizer (see events). It's not usefull to call this method.*
- void [MPTK\\_ClearAllSound](#) (bool destroyAudioSource=false)  
*Clear all sound*

## Data Fields

- EventNotesMidiClass [OnEventNotesMidi](#)  
*Define unity event to trigger when notes available from the Midi file.*
- EventStartMidiClass [OnEventStartPlayMidi](#)  
*Define unity event to trigger at start of playing the Midi.*
- EventEndMidiClass [OnEventEndPlayMidi](#)  
*Define unity event to trigger at end of playing the midi.*
- EventSynthClass [OnEventSynthAwake](#)  
*Unity event fired at awake of the synthesizer. Name of the gameobject component is passed as a parameter.*
- EventSynthClass [OnEventSynthStarted](#)  
*Unity event fired at start of the synthesizer. Name of the gameobject component is passed as a parameter.*
- bool [MPTK\\_PauseOnDistance](#)  
*Should the Midi playing must be paused if distance between AudioListener and [MidiFilePlayer](#) is greater than MaxDistance*
- bool [MPTK\\_EnablePanChange](#)  
*Should change pan from Midi Events or from SoundFont ?*
- bool [MPTK\\_WeakDevice](#)  
*Should play on a weak device (cheaper smartphone) ? Playing Midi files with WeakDevice activated could cause some bad interpretation of Midi Event, consequently bad sound.*
- float [MPTK\\_ReleaseTimeMin](#) = 50f  
*Define a minimum release time at noteoff in milliseconds. Default 50 ms is a good tradeoff. Below some unpleasant sound could be heard.*

## Properties

- override string [MPTK\\_MidiName](#) [get, set]  
*Full path to Midi file or URL to play. must start with [file://](#) or [http://](#) or [https://](#).*
- override int [MPTK\\_MidiIndex](#) [get, set]  
*Index Midi to play or playing - NO EFFECT for external*
- virtual bool [MPTK\\_PlayOnStart](#) [get, set]  
*Should the Midi start playing when application start ?*
- virtual bool [MPTK\\_Loop](#) [get, set]  
*Should automatically restart when Midi reach the end ?*
- virtual double [MPTK\\_Tempo](#) [get]

Get default tempo defined in Midi file or modified with Speed. Return *QuarterPerMinuteValue* similar to BPM (Beat Per Measure)

- virtual float [MPTK\\_Speed](#) [get, set]  
*Speed of playing. Between 0.1 (10%) to 5.0 (500%). Set to 1 for normal speed.*
- virtual double [MPTK\\_Position](#) [get, set]  
*Set or Get midi position time from 0 to lenght time of midi playing (in millisecond)*
- virtual bool [MPTK\\_IsPaused](#) [get]  
*Is Midi file playing is paused ?*
- virtual bool [MPTK\\_IsPlaying](#) [get]  
*Is Midi file is playing ?*
- virtual TimeSpan [MPTK\\_Duration](#) [get]  
*Value updated only when playing in Unity (for inspector refresh)*
- virtual long [MPTK\\_TickLast](#) [get]  
*Last tick position in Midi: Value of the tick for the last midi event in sequence expressed in number of "ticks".  $MPTK\_TickLast / MPTK\_DeltaTicksPerQuarterNote$  equal the duration time of a quarter-note regardless the defined tempo.*
- virtual long [MPTK\\_TickCurrent](#) [get, set]  
*Current tick position in Midi: Time of the current midi event expressed in number of "ticks".  $MPTK\_TickCurrent / MPTK\_DeltaTicksPerQuarterNote$  equal the duration time of a quarter-note regardless the defined tempo.*
- virtual double [MPTK\\_PulseLenght](#) [get]  
*Lenght in millisecond of a quarter*
- virtual TimeSpan [MPTK\\_PlayTime](#) [get]  
*Updated only when playing in Unity (for inspector refresh)*
- virtual bool [MPTK\\_LogEvents](#) [get, set]  
*Log midi events*
- virtual bool [MPTK\\_EnableChangeTempo](#) [get, set]  
*Should accept change tempo from Midi Events ?*
- virtual bool [MPTK\\_KeepNoteOff](#) [get, set]  
*Should keep note off event Events ?*
- virtual bool [MPTK\\_DirectSendToPlayer](#) [get, set]  
*If true (default) then Midi events are sent automatically to the midi player. Set to false if you want to process events without playing sound. *OnEventNotesMidi* Unity Event can be used to process each notes.*
- virtual int [MPTK\\_Quantization](#) [get, set]  
*Level of quantization :*
- virtual List< [TrackMidiEvent](#) > [MPTK\\_MidiEvents](#) [get]  
*Get all the raw midi events available in the midi file*
- virtual int [MPTK\\_DeltaTicksPerQuarterNote](#) [get]  
*Delta Ticks Per Quarter Note. Indicate the duration time in "ticks" which make up a quarter-note. For instance, if 96, then a duration of an eighth-note in the file would be 48.*
- virtual bool [MPTK\\_EnablePresetDrum](#) [get, set]  
*Should accept change Preset for Drum canal 10 ? Disabled by default. Could sometimes create bad sound with midi files not really compliant with the Midi norm.*
- virtual float [MPTK\\_MaxDistance](#) [get, set]  
*MaxDistance to use for *PauseOnDistance**
- virtual float [MPTK\\_Volume](#) [get, set]  
*Volume of midi playing. Must be  $\geq 0$  and  $\leq 1$*
- virtual int [MPTK\\_Transpose](#) [get, set]  
*Transpose note from -24 to 24*

---

## Detailed Description

PRO Version - Script for the prefab [MidiExternalPlayer](#). See full example TestMidiExternalPlayer.cs with a light sequencer. Play a midi file from a path on the local desktop or from a web site

---

## Member Function Documentation

**void MidiPlayerTK.MidiPlayer.MPTK\_ClearAllSound (bool *destroyAudioSource* = false)[inherited]**

Clear all sound

### Parameters:

<i>destroyAudioSource</i>	Destroy also audioSource (default:false)
---------------------------	--

```
if (GUILayout.Button("Clear"))
    midiStreamPlayer.MPTK_ClearAllSound(true);
```

**void MidiPlayerTK.MidiPlayer.MPTK\_InitSynth (int *channelCount* = 16)[inherited]**

Init the synthesizer. Prefabs automatically initialize the synthesizer (see events). It's not usefull to call this method.

### Parameters:

<i>channelCount</i>	Number of channel to create
---------------------	-----------------------------

**override void MidiPlayerTK.MidiExternalPlayer.MPTK\_Next ()[virtual]**

Play next Midi - NO EFFECT for external

Reimplemented from [MidiPlayerTK.MidiFilePlayer](#).

**[MPTKEvent.EnumLength](#) MidiPlayerTK.MidiFilePlayer.MPTK\_NoteLength ([MPTKEvent note](#))[inherited]**

Return note length as [https://en.wikipedia.org/wiki/Note\\_value](https://en.wikipedia.org/wiki/Note_value)

### Parameters:

<i>note</i>	
-------------	--

### Returns:

[MPTKEvent.EnumLength](#)

**virtual void MidiPlayerTK.MidiFilePlayer.MPTK\_Pause (float *timeToPauseMS* = -1f)[virtual], [inherited]**

Pause the current playing

**Parameters:**

<i>timeToPauseMS</i>	time to pause in milliseconds. default: indefinitely
----------------------	--

**override void MidiPlayerTK.MidiExternalPlayer.MPTK\_Play () [virtual]**

Play the midi file defined in MPTK\_MidiName

```
MidiExternalPlayer midiExternalPlayer = FindObjectOfType<MidiExternalPlayer>();
MidiExternalPlayer.MPTK MidiName = @"C:\Users\xxx\Midi\Bach The Art of Fugue -
Nol.mid";
//or
MidiExternalPlayer.MPTK_MidiName =
"http://www.midiworld.com/midis/other/bach/bwv1060b.mid";
MidiExternalPlayer.MPTK_Play();
!
```

Reimplemented from [MidiPlayerTK.MidiFilePlayer](#).

**override void MidiPlayerTK.MidiExternalPlayer.MPTK\_Previous () [virtual]**

Play previous Midi - NO EFFECT for external

Reimplemented from [MidiPlayerTK.MidiFilePlayer](#).

**virtual void MidiPlayerTK.MidiFilePlayer.MPTK\_RePlay () [virtual], [inherited]**

Restart playing of the current midi file

**void MidiPlayerTK.MidiFilePlayer.MPTK\_ReSyncTime () [inherited]**

In case of delay in the application, resync is usefull to avoid multi tock play at the same time

**virtual void MidiPlayerTK.MidiFilePlayer.MPTK\_Stop () [virtual], [inherited]**

Stop playing

---

## Field Documentation

**bool MidiPlayerTK.MidiSynth.MPTK\_EnablePanChange [inherited]**

Should change pan from Midi Events or from SoundFont ?

### **bool MidiPlayerTK.MidiSynth.MPTK\_PauseOnDistance [inherited]**

Should the Midi playing must be paused if distance between AudioListener and [MidiFilePlayer](#) is greater than MaxDistance

### **float MidiPlayerTK.MidiSynth.MPTK\_ReleaseTimeMin = 50f [inherited]**

Define a minimum release time at noteoff in milliseconds. Default 50 ms is a good tradeoff. Below some unpleasant sound could be heard.

### **bool MidiPlayerTK.MidiSynth.MPTK\_WeakDevice [inherited]**

Should play on a weak device (cheaper smartphone) ? Playing Midi files with WeakDevice activated could cause some bad interpretation of Midi Event, consequently bad sound.

### **EventEndMidiClass MidiPlayerTK.MidiFilePlayer.OnEventEndPlayMidi [inherited]**

Define unity event to trigger at end of playing the midi.

```
MidiFilePlayer midiFilePlayer = FindObjectOfType<MidiFilePlayer>();
...
if (!midiFilePlayer.OnEventEndPlayMidi.HasEvent())
{
    // No listener defined, set now by script. EndPlay will be called.
    midiFilePlayer.OnEventEndPlayMidi.AddListener(EndPlay);
}
...
public void EndPlay(string midiname, EventEndMidiEnum reason)
{
    Debug.LogFormat("End playing midi {0} reason:{1}", midiname, reason);
}
```

### **EventNotesMidiClass MidiPlayerTK.MidiFilePlayer.OnEventNotesMidi [inherited]**

Define unity event to trigger when notes available from the Midi file.

```
MidiFilePlayer midiFilePlayer = FindObjectOfType<MidiFilePlayer>();
...
if (!midiFilePlayer.OnEventNotesMidi.HasEvent())
{
    // No listener defined, set now by script. NotesToPlay will be called for each new
    notes read from Midi file
    midiFilePlayer.OnEventNotesMidi.AddListener(NotesToPlay);
}
...
public void NotesToPlay(List<MPTKEvent> notes)
{
    Debug.Log(notes.Count);
    foreach (MPTKEvent midievent in notes)
```

```

    {
        ...
    }
}
!

```

### EventStartMidiClass MidiPlayerTK.MidiFilePlayer.OnEventStartPlayMidi [inherited]

Define unity event to trigger at start of playing the Midi.

```

MidiFilePlayer midiFilePlayer = FindObjectOfType<MidiFilePlayer>();
...
if (!midiFilePlayer.OnEventStartPlayMidi.HasEvent())
{
    // No listener defined, set now by script. StartPlay will be called.
    midiFilePlayer.OnEventStartPlayMidi.AddListener(StartPlay);
}
...
public void StartPlay(string midiname)
{
    Debug.LogFormat("Start playing midi {0}", midiname);
}

```

### EventSynthClass MidiPlayerTK.MidiSynth.OnEventSynthAwake [inherited]

Unity event fired at awake of the synthesizer. Name of the gameobject component is passed as a parameter.

```

...
if (!midiStreamPlayer.OnEventSynthAwake.HasEvent())
    midiStreamPlayer.OnEventSynthAwake.AddListener(StartLoadingSynth);
...
public void StartLoadingSynth(string name)
{
    Debug.LogFormat("Synth {0} loading", name);
}

```

### EventSynthClass MidiPlayerTK.MidiSynth.OnEventSynthStarted [inherited]

Unity event fired at start of the synthesizer. Name of the gameobject component is passed as a parameter.

```

...
if (!midiStreamPlayer.OnEventStartSynth.HasEvent())
    midiStreamPlayer.OnEventStartSynth.AddListener(EndLoadingSynth);
...
public void EndLoadingSynth(string name)
{
    Debug.LogFormat("Synth {0} loaded", name);
    midiStreamPlayer.MPTK_PlayEvent(
        new MPTKEvent() { Command = MPTKCommand.PatchChange, Value =
CurrentPatchInstrument, Channel = StreamChannel});
}

```

## Property Documentation

**virtual int MidiPlayerTK.MidiFilePlayer.MPTK\_DeltaTicksPerQuarterNote [get], [inherited]**

Delta Ticks Per Quarter Note. Indicate the duration time in "ticks" which make up a quarter-note. For instance, if 96, then a duration of an eighth-note in the file would be 48.

**virtual bool MidiPlayerTK.MidiFilePlayer.MPTK\_DirectSendToPlayer [get], [set], [inherited]**

If true (default) then Midi events are sent automatically to the midi player. Set to false if you want to process events without playing sound. OnEventNotesMidi Unity Event can be used to process each notes.

**virtual TimeSpan MidiPlayerTK.MidiFilePlayer.MPTK\_Duration [get], [inherited]**

Value updated only when playing in Unity (for inspector refresh)

Get duration of current Midi with current tempo

**virtual bool MidiPlayerTK.MidiFilePlayer.MPTK\_EnableChangeTempo [get], [set], [inherited]**

Should accept change tempo from Midi Events ?

**virtual bool MidiPlayerTK.MidiPlayer.MPTK\_EnablePresetDrum [get], [set], [inherited]**

Should accept change Preset for Drum canal 10 ? Disabled by default. Could sometimes create bad sound with midi files not really compliant with the Midi norm.

**virtual bool MidiPlayerTK.MidiFilePlayer.MPTK\_IsPaused [get], [inherited]**

Is Midi file playing is paused ?

**virtual bool MidiPlayerTK.MidiFilePlayer.MPTK\_IsPlaying [get], [inherited]**

Is Midi file is playing ?

**virtual bool MidiPlayerTK.MidiFilePlayer.MPTK\_KeepNoteOff [get], [set], [inherited]**



Should keep note off event Events ?

**virtual bool MidiPlayerTK.MidiFilePlayer.MPTK\_LogEvents [get], [set], [inherited]**

Log midi events

**virtual bool MidiPlayerTK.MidiFilePlayer.MPTK\_Loop [get], [set], [inherited]**

Should automatically restart when Midi reach the end ?

**virtual float MidiPlayerTK.MidiSynth.MPTK\_MaxDistance [get], [set], [inherited]**

MaxDistance to use for PauseOnDistance

**virtual List<[TrackMidiEvent](#)> MidiPlayerTK.MidiFilePlayer.MPTK\_MidiEvents [get], [inherited]**

Get all the raw midi events available in the midi file

**override int MidiPlayerTK.MidiExternalPlayer.MPTK\_MidiIndex [get], [set]**

Index Midi to play or playing - NO EFFECT for external

**override string MidiPlayerTK.MidiExternalPlayer.MPTK\_MidiName [get], [set]**

Full path to Midi file or URL to play. must start with [file://](#) or [http://](#) or [https://](#).

```
MidiExternalPlayer midiExternalPlayer = FindObjectOfType<MidiExternalPlayer>();
MidiExternalPlayer.MPTK_MidiName = @"C:\Users\xxx\Midi\Bach The Art of Fugue -
Nol.mid";
//or
MidiExternalPlayer.MPTK_MidiName =
"http://www.midiworld.com/midis/other/bach/bwv1060b.mid";
MidiExternalPlayer.MPTK_Play();
!
```

**virtual bool MidiPlayerTK.MidiFilePlayer.MPTK\_PlayOnStart [get], [set], [inherited]**

Should the Midi start playing when application start ?

**virtual TimeSpan MidiPlayerTK.MidiFilePlayer.MPTK\_PlayTime [get], [inherited]**

Updated only when playing in Unity (for inspector refresh)  
Time from the start of playing the current midi

**virtual double MidiPlayerTK.MidiFilePlayer.MPTK\_Position [get], [set], [inherited]**

Set or Get midi position time from 0 to lenght time of midi playing (in millisecond)

**virtual double MidiPlayerTK.MidiFilePlayer.MPTK\_PulseLenght [get], [inherited]**

Lenght in millisecond of a quarter

**virtual int MidiPlayerTK.MidiFilePlayer.MPTK\_Quantization [get], [set], [inherited]**

Level of quantization :

- 0 = None
- 1 = Quarter Note
- 2 = Eighth Note
- 3 = 16th Note
- 4 = 32th Note
- 5 = 64th Note

**virtual float MidiPlayerTK.MidiFilePlayer.MPTK\_Speed [get], [set], [inherited]**

Speed of playing. Between 0.1 (10%) to 5.0 (500%). Set to 1 for normal speed.

**virtual double MidiPlayerTK.MidiFilePlayer.MPTK\_Tempo [get], [inherited]**

Get default tempo defined in Midi file or modified with Speed. Return QuarterPerMinuteValue similar to BPM (Beat Per Measure)

**virtual long MidiPlayerTK.MidiFilePlayer.MPTK\_TickCurrent [get], [set], [inherited]**

Current tick position in Midi: Time of the current midi event expressed in number of "ticks".  
 $MPTK\_TickCurrent / MPTK\_DeltaTicksPerQuarterNote$  equal the duration time of a quarter-note regardless the defined tempo.

**virtual long MidiPlayerTK.MidiFilePlayer.MPTK\_TickLast [get], [inherited]**

Last tick position in Midi: Value of the tick for the last midi event in sequence expressed in number of "ticks".  $MPTK\_TickLast / MPTK\_DeltaTicksPerQuarterNote$  equal the duration time of a quarter-note regardless the defined tempo.

**virtual int MidiPlayerTK.MidiSynth.MPTK\_Transpose [get], [set], [inherited]**

Transpose note from -24 to 24

**virtual float MidiPlayerTK.MidiSynth.MPTK\_Volume [get], [set], [inherited]**

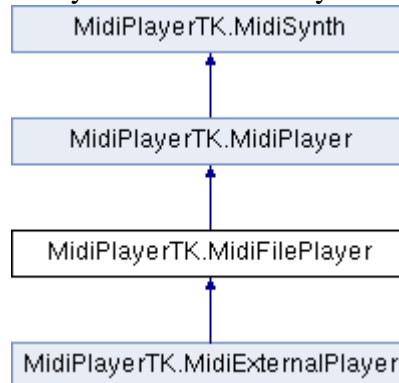
Volume of midi playing. Must be  $\geq 0$  and  $\leq 1$

---

## MidiPlayerTK.MidiFilePlayer

Script for the prefab [MidiFilePlayer](#). Play a selected midi file. List of Midi file must be defined with Midi Player Setup (see Unity menu MPTK).

Inheritance diagram for MidiPlayerTK.MidiFilePlayer:



### Public Member Functions

- virtual void [MPTK\\_Play](#) ()  
*Play the midi file defined with MPTK\_MidiName or MPTK\_MidiIndex*
- virtual void [MPTK\\_Stop](#) ()  
*Stop playing*
- virtual void [MPTK\\_RePlay](#) ()  
*Restart playing of the current midi file*
- virtual void [MPTK\\_Pause](#) (float timeToPauseMS=-1f)  
*Pause the current playing*
- virtual void [MPTK\\_Next](#) ()  
*Play next Midi from the list of midi defined in MPTK (see Unity menu Midi)*
- virtual void [MPTK\\_Previous](#) ()

Play previous Midi from the list of midi defined in MPTK (see Unity menu Midi)

- void [MPTK\\_ReSyncTime](#) ()  
*In case of delay in the application, resync is usefull to avoid multi tock play at the same time*
- [MPTKEvent.EnumLength](#) [MPTK\\_NoteLength](#) ([MPTKEvent](#) note)  
*Return note length as [https://en.wikipedia.org/wiki/Note\\_value](https://en.wikipedia.org/wiki/Note_value)*
- void [MPTK\\_InitSynth](#) (int channelCount=16)  
*Init the synthetizer. Prefabs automatically initialize the synthetizer (see events). It's not usefull to call this method.*
- void [MPTK\\_ClearAllSound](#) (bool destroyAudioSource=false)  
*Clear all sound*

## Data Fields

- EventNotesMidiClass [OnEventNotesMidi](#)  
*Define unity event to trigger when notes available from the Midi file.*
- EventStartMidiClass [OnEventStartPlayMidi](#)  
*Define unity event to trigger at start of playing the Midi.*
- EventEndMidiClass [OnEventEndPlayMidi](#)  
*Define unity event to trigger at end of playing the midi.*
- EventSynthClass [OnEventSynthAwake](#)  
*Unity event fired at awake of the synthesizer. Name of the gameobject component is passed as a parameter.*
- EventSynthClass [OnEventSynthStarted](#)  
*Unity event fired at start of the synthesizer. Name of the gameobject component is passed as a parameter.*
- bool [MPTK\\_PauseOnDistance](#)  
*Should the Midi playing must be paused if distance between AudioListener and [MidiFilePlayer](#) is greater than MaxDistance*
- bool [MPTK\\_EnablePanChange](#)  
*Should change pan from Midi Events or from SoundFont ?*
- bool [MPTK\\_WeakDevice](#)  
*Should play on a weak device (cheaper smartphone) ? Playing Midi files with WeakDevice activated could cause some bad interpretation of Midi Event, consequently bad sound.*
- float [MPTK\\_ReleaseTimeMin](#) = 50f  
*Define a minimum release time at noteoff in milliseconds. Default 50 ms is a good tradeoff. Below some unpleasant sound coule be heard.*

## Properties

- virtual string [MPTK\\_MidiName](#) [get, set]  
*Midi name to play. Use the exact name defined in Unity resources folder MidiDB without any path or extension. Tips: Add Midi files to your project with the Unity menu MPTK or add it directly in the ressource folder and open Midi File Setup to automatically integrate Midi in MPTK.*
- virtual int [MPTK\\_MidiIndex](#) [get, set]  
*Index Midi. Find the Index of Midi file from the popup in [MidiFilePlayer](#) inspector. Tips: Add Midi files to your project with the Unity menu MPTK or add it directly in the ressource folder and open Midi File Setup to automatically integrate Midi in MPTK. return -1 if not found*
- virtual bool [MPTK\\_PlayOnStart](#) [get, set]  
*Should the Midi start playing when application start ?*
- virtual bool [MPTK\\_Loop](#) [get, set]  
*Should automatically restart when Midi reach the end ?*
- virtual double [MPTK\\_Tempo](#) [get]

Get default tempo defined in Midi file or modified with Speed. Return *QuarterPerMinuteValue* similar to BPM (Beat Per Measure)

- virtual float [MPTK\\_Speed](#) [get, set]  
*Speed of playing. Between 0.1 (10%) to 5.0 (500%). Set to 1 for normal speed.*
- virtual double [MPTK\\_Position](#) [get, set]  
*Set or Get midi position time from 0 to lenght time of midi playing (in millisecond)*
- virtual bool [MPTK\\_IsPaused](#) [get]  
*Is Midi file playing is paused ?*
- virtual bool [MPTK\\_IsPlaying](#) [get]  
*Is Midi file is playing ?*
- virtual TimeSpan [MPTK\\_Duration](#) [get]  
*Value updated only when playing in Unity (for inspector refresh)*
- virtual long [MPTK\\_TickLast](#) [get]  
*Last tick position in Midi: Value of the tick for the last midi event in sequence expressed in number of "ticks".  $MPTK\_TickLast / MPTK\_DeltaTicksPerQuarterNote$  equal the duration time of a quarter-note regardless the defined tempo.*
- virtual long [MPTK\\_TickCurrent](#) [get, set]  
*Current tick position in Midi: Time of the current midi event expressed in number of "ticks".  $MPTK\_TickCurrent / MPTK\_DeltaTicksPerQuarterNote$  equal the duration time of a quarter-note regardless the defined tempo.*
- virtual double [MPTK\\_PulseLenght](#) [get]  
*Lenght in millisecond of a quarter*
- virtual TimeSpan [MPTK\\_PlayTime](#) [get]  
*Updated only when playing in Unity (for inspector refresh)*
- virtual bool [MPTK\\_LogEvents](#) [get, set]  
*Log midi events*
- virtual bool [MPTK\\_EnableChangeTempo](#) [get, set]  
*Should accept change tempo from Midi Events ?*
- virtual bool [MPTK\\_KeepNoteOff](#) [get, set]  
*Should keep note off event Events ?*
- virtual bool [MPTK\\_DirectSendToPlayer](#) [get, set]  
*If true (default) then Midi events are sent automatically to the midi player. Set to false if you want to process events without playing sound. *OnEventNotesMidi* Unity Event can be used to process each notes.*
- virtual int [MPTK\\_Quantization](#) [get, set]  
*Level of quantization :*
- virtual List< [TrackMidiEvent](#) > [MPTK\\_MidiEvents](#) [get]  
*Get all the raw midi events available in the midi file*
- virtual int [MPTK\\_DeltaTicksPerQuarterNote](#) [get]  
*Delta Ticks Per Quarter Note. Indicate the duration time in "ticks" which make up a quarter-note. For instance, if 96, then a duration of an eighth-note in the file would be 48.*
- virtual bool [MPTK\\_EnablePresetDrum](#) [get, set]  
*Should accept change Preset for Drum canal 10 ? Disabled by default. Could sometimes create bad sound with midi files not really compliant with the Midi norm.*
- virtual float [MPTK\\_MaxDistance](#) [get, set]  
*MaxDistance to use for *PauseOnDistance**
- virtual float [MPTK\\_Volume](#) [get, set]  
*Volume of midi playing. Must be  $\geq 0$  and  $\leq 1$*
- virtual int [MPTK\\_Transpose](#) [get, set]  
*Transpose note from -24 to 24*

---

## Detailed Description

Script for the prefab [MidiFilePlayer](#). Play a selected midi file. List of Midi file must be defined with Midi Player Setup (see Unity menu MPTK).

---

## Member Function Documentation

**void MidiPlayerTK.MidiPlayer.MPTK\_ClearAllSound (bool *destroyAudioSource* = false)[inherited]**

Clear all sound

### Parameters:

<i>destroyAudioSource</i>	Destroy also audioSource (default:false)
---------------------------	--

```
if (GUILayout.Button("Clear"))
    midiStreamPlayer.MPTK_ClearAllSound(true);
```

**void MidiPlayerTK.MidiPlayer.MPTK\_InitSynth (int *channelCount* = 16)[inherited]**

Init the synthesizer. Prefabs automatically initialize the synthesizer (see events). It's not useful to call this method.

### Parameters:

<i>channelCount</i>	Number of channel to create
---------------------	-----------------------------

**virtual void MidiPlayerTK.MidiFilePlayer.MPTK\_Next () [virtual]**

Play next Midi from the list of midi defined in MPTK (see Unity menu Midi)

Reimplemented in [MidiPlayerTK.MidiExternalPlayer](#).

**[MPTKEvent.EnumLength](#) MidiPlayerTK.MidiFilePlayer.MPTK\_NoteLength ([MPTKEvent note](#))**

Return note length as [https://en.wikipedia.org/wiki/Note\\_value](https://en.wikipedia.org/wiki/Note_value)

### Parameters:

<i>note</i>	
-------------	--

### Returns:

[MPTKEvent.EnumLength](#)

**virtual void MidiPlayerTK.MidiFilePlayer.MPTK\_Pause (float *timeToPauseMS* = -1f)[virtual]**

Pause the current playing

**Parameters:**

<i>timeToPauseMS</i>	time to pause in milliseconds. default: indefinitely
----------------------	--

**virtual void MidiPlayerTK.MidiFilePlayer.MPTK\_Play () [virtual]**

Play the midi file defined with MPTK\_MidiName or MPTK\_MidiIndex

Reimplemented in [MidiPlayerTK.MidiExternalPlayer](#).

**virtual void MidiPlayerTK.MidiFilePlayer.MPTK\_Previous () [virtual]**

Play previous Midi from the list of midi defined in MPTK (see Unity menu Midi)

Reimplemented in [MidiPlayerTK.MidiExternalPlayer](#).

**virtual void MidiPlayerTK.MidiFilePlayer.MPTK\_RePlay () [virtual]**

Restart playing of the current midi file

**void MidiPlayerTK.MidiFilePlayer.MPTK\_ReSyncTime ()**

In case of delay in the application, resync is usefull to avoid multi tock play at the same time

**virtual void MidiPlayerTK.MidiFilePlayer.MPTK\_Stop () [virtual]**

Stop playing

---

## Field Documentation

**bool MidiPlayerTK.MidiSynth.MPTK\_EnablePanChange [inherited]**

Should change pan from Midi Events or from SoundFont ?

**bool MidiPlayerTK.MidiSynth.MPTK\_PauseOnDistance [inherited]**

Should the Midi playing must be paused if distance between AudioListener and [MidiFilePlayer](#) is greater than MaxDistance

### **float MidiPlayerTK.MidiSynth.MPTK\_ReleaseTimeMin = 50f [inherited]**

Define a minimum release time at noteoff in milliseconds. Default 50 ms is a good tradeoff. Below some unpleasant sound could be heard.

### **bool MidiPlayerTK.MidiSynth.MPTK\_WeakDevice [inherited]**

Should play on a weak device (cheaper smartphone) ? Playing Midi files with WeakDevice activated could cause some bad interpretation of Midi Event, consequently bad sound.

### **EventEndMidiClass MidiPlayerTK.MidiFilePlayer.OnEventEndPlayMidi**

Define unity event to trigger at end of playing the midi.

```
MidiFilePlayer midiFilePlayer = FindObjectOfType<MidiFilePlayer>();
...
if (!midiFilePlayer.OnEventEndPlayMidi.HasEvent())
{
    // No listener defined, set now by script. EndPlay will be called.
    midiFilePlayer.OnEventEndPlayMidi.AddListener(EndPlay);
}
...
public void EndPlay(string midiname, EventEndMidiEnum reason)
{
    Debug.LogFormat("End playing midi {0} reason:{1}", midiname, reason);
}
```

### **EventNotesMidiClass MidiPlayerTK.MidiFilePlayer.OnEventNotesMidi**

Define unity event to trigger when notes available from the Midi file.

```
MidiFilePlayer midiFilePlayer = FindObjectOfType<MidiFilePlayer>();
...
if (!midiFilePlayer.OnEventNotesMidi.HasEvent())
{
    // No listener defined, set now by script. NotesToPlay will be called for each new
    notes read from Midi file
    midiFilePlayer.OnEventNotesMidi.AddListener(NotesToPlay);
}
...
public void NotesToPlay(List<MPTKEvent> notes)
{
    Debug.Log(notes.Count);
    foreach (MPTKEvent midievent in notes)
    {
        ...
    }
}
!
```



## EventStartMidiClass MidiPlayerTK.MidiFilePlayer.OnEventStartPlayMidi

Define unity event to trigger at start of playing the Midi.

```
MidiFilePlayer midiFilePlayer = FindObjectOfType<MidiFilePlayer>();
...
if (!midiFilePlayer.OnEventStartPlayMidi.HasEvent())
{
    // No listener defined, set now by script. StartPlay will be called.
    midiFilePlayer.OnEventStartPlayMidi.AddListener(StartPlay);
}
...
public void StartPlay(string midiname)
{
    Debug.LogFormat("Start playing midi {0}", midiname);
}
```

## EventSynthClass MidiPlayerTK.MidiSynth.OnEventSynthAwake [inherited]

Unity event fired at awake of the synthesizer. Name of the gameobject component is passed as a parameter.

```
...
if (!midiStreamPlayer.OnEventSynthAwake.HasEvent())
    midiStreamPlayer.OnEventSynthAwake.AddListener(StartLoadingSynth);
...
public void StartLoadingSynth(string name)
{
    Debug.LogFormat("Synth {0} loading", name);
}
```

## EventSynthClass MidiPlayerTK.MidiSynth.OnEventSynthStarted [inherited]

Unity event fired at start of the synthesizer. Name of the gameobject component is passed as a parameter.

```
...
if (!midiStreamPlayer.OnEventStartSynth.HasEvent())
    midiStreamPlayer.OnEventStartSynth.AddListener(EndLoadingSynth);
...
public void EndLoadingSynth(string name)
{
    Debug.LogFormat("Synth {0} loaded", name);
    midiStreamPlayer.MPTK PlayEvent(
        new MPTKEvent() { Command = MPTKCommand.PatchChange, Value =
        CurrentPatchInstrument, Channel = StreamChannel});
}
```

---

## Property Documentation

### virtual int MidiPlayerTK.MidiFilePlayer.MPTK\_DeltaTicksPerQuarterNote [get]

Delta Ticks Per Quarter Note. Indicate the duration time in "ticks" which make up a quarter-note. For instance, if 96, then a duration of an eighth-note in the file would be 48.

**virtual bool MidiPlayerTK.MidiFilePlayer.MPTK\_DirectSendToPlayer [get], [set]**

If true (default) then Midi events are sent automatically to the midi player. Set to false if you want to process events without playing sound. OnEventNotesMidi Unity Event can be used to process each notes.

**virtual TimeSpan MidiPlayerTK.MidiFilePlayer.MPTK\_Duration [get]**

Value updated only when playing in Unity (for inspector refresh)

Get duration of current Midi with current tempo

**virtual bool MidiPlayerTK.MidiFilePlayer.MPTK\_EnableChangeTempo [get], [set]**

Should accept change tempo from Midi Events ?

**virtual bool MidiPlayerTK.MidiPlayer.MPTK\_EnablePresetDrum [get], [set],  
[inherited]**

Should accept change Preset for Drum canal 10 ? Disabled by default. Could sometimes create bad sound with midi files not really compliant with the Midi norm.

**virtual bool MidiPlayerTK.MidiFilePlayer.MPTK\_IsPaused [get]**

Is Midi file playing is paused ?

**virtual bool MidiPlayerTK.MidiFilePlayer.MPTK\_IsPlaying [get]**

Is Midi file is playing ?

**virtual bool MidiPlayerTK.MidiFilePlayer.MPTK\_KeepNoteOff [get], [set]**

Should keep note off event Events ?

**virtual bool MidiPlayerTK.MidiFilePlayer.MPTK\_LogEvents [get], [set]**

Log midi events

**virtual bool MidiPlayerTK.MidiFilePlayer.MPTK\_Loop [get], [set]**

Should automatically restart when Midi reach the end ?

**virtual float MidiPlayerTK.MidiSynth.MPTK\_MaxDistance [get], [set], [inherited]**

MaxDistance to use for PauseOnDistance

**virtual List<[TrackMidiEvent](#)> MidiPlayerTK.MidiFilePlayer.MPTK\_MidiEvents [get]**

Get all the raw midi events available in the midi file

**virtual int MidiPlayerTK.MidiFilePlayer.MPTK\_MidiIndex [get], [set]**

Index Midi. Find the Index of Midi file from the popup in [MidiFilePlayer](#) inspector. Tips: Add Midi files to your project with the Unity menu MPTK or add it directly in the ressource folder and open Midi File Setup to automatically integrate Midi in MPTK. return -1 if not found

**Parameters:**

<i>index</i>	
--------------	--

**virtual string MidiPlayerTK.MidiFilePlayer.MPTK\_MidiName [get], [set]**

Midi name to play. Use the exact name defined in Unity resources folder MidiDB without any path or extension. Tips: Add Midi files to your project with the Unity menu MPTK or add it directly in the ressource folder and open Midi File Setup to automatically integrate Midi in MPTK.

```
midiFilePlayer.MPTK_MidiName = "Albinoni - Adagio";
```

**virtual bool MidiPlayerTK.MidiFilePlayer.MPTK\_PlayOnStart [get], [set]**

Should the Midi start playing when application start ?

**virtual TimeSpan MidiPlayerTK.MidiFilePlayer.MPTK\_PlayTime [get]**

Updated only when playing in Unity (for inspector refresh)

Time from the start of playing the current midi

**virtual double MidiPlayerTK.MidiFilePlayer.MPTK\_Position [get], [set]**

Set or Get midi position time from 0 to lenght time of midi playing (in millisecond)

**virtual double MidiPlayerTK.MidiFilePlayer.MPTK\_PulseLenght [get]**

Lenght in millisecond of a quarter

**virtual int MidiPlayerTK.MidiFilePlayer.MPTK\_Quantization [get], [set]**

Level of quantization :

- 0 = None
- 1 = Quarter Note
- 2 = Eighth Note
- 3 = 16th Note
- 4 = 32th Note
- 5 = 64th Note

**virtual float MidiPlayerTK.MidiFilePlayer.MPTK\_Speed [get], [set]**

Speed of playing. Between 0.1 (10%) to 5.0 (500%). Set to 1 for normal speed.

**virtual double MidiPlayerTK.MidiFilePlayer.MPTK\_Tempo [get]**

Get default tempo defined in Midi file or modified with Speed. Return QuarterPerMinuteValue similar to BPM (Beat Per Measure)

**virtual long MidiPlayerTK.MidiFilePlayer.MPTK\_TickCurrent [get], [set]**

Current tick position in Midi: Time of the current midi event expressed in number of "ticks".  $MPTK\_TickCurrent / MPTK\_DeltaTicksPerQuarterNote$  equal the duration time of a quarter-note regardless the defined tempo.

**virtual long MidiPlayerTK.MidiFilePlayer.MPTK\_TickLast [get]**

Last tick position in Midi: Value of the tick for the last midi event in sequence expressed in number of "ticks".  $MPTK\_TickLast / MPTK\_DeltaTicksPerQuarterNote$  equal the duration time of a quarter-note regardless the defined tempo.

**virtual int MidiPlayerTK.MidiSynth.MPTK\_Transpose [get], [set], [inherited]**

Transpose note from -24 to 24

virtual float MidiPlayerTK.MidiSynth.MPTK\_Volume [get], [set], [inherited]

Volume of midi playing. Must be  $\geq 0$  and  $\leq 1$

---

## MidiPlayerTK.MidiFileWriter

PRO Version - Write a midi file from differents sources based on NAudio framework. See full example TestMidiWriter.cs with a light sequencer.

### Public Member Functions

- [MidiFileWriter \(\)](#)  
*Create an empty [MidiFileWriter](#)*
- [MidiFileWriter \(int deltaTicksPerQuarterNote, int midiFileType\)](#)  
*Create a [MidiFileWriter](#) with an empty Midi Event list*
- bool [MPTK\\_LoadFromMPTK \(List< \[TrackMidiEvent\]\(#\) > MidiSorted\)](#)  
*Create a [MidiFileWriter](#) from a MPTK list of midi events. A midi file must be loaded before from a [MidiFilePlayer](#) gameobject (as in example) or from a call to [MidiFileWriter.MPTK\\_LoadFromFile\(filename\)](#).*
- bool [MPTK\\_LoadFromMidiDB \(int indexMidiDb\)](#)  
*Create a [MidiFileWriter](#) from a Midi found in MPTK MidiDB*
- void [MPTK\\_CreateTrack \(int count\)](#)  
*Create tracks*
- void [MPTK\\_EndTrack \(int trackNumber\)](#)  
*Close the track (mandatory for a well formed midi file)*
- void [MPTK\\_AddEvent \(int track, MidiEvent midievent\)](#)  
*Add a generic Midi event*
- void [MPTK\\_AddNote \(int track, long absoluteTime, int channel, int note, int velocity, int duration\)](#)  
*Add a note event. the corresponding Noteoff is automatically created.*
- bool [MPTK\\_LoadFromFile \(string filename\)](#)  
*Load a Midi file from OS system file (could be dependant of the OS)*
- bool [MPTK\\_WriteToFile \(string filename\)](#)  
*Write Midi file to an OS folder*
- bool [MPTK\\_WriteToMidiDB \(string filename\)](#)  
*Write Midi file to MidiDB. To be used only in edit mode not in a standalone application.*

### Static Public Member Functions

- static int [MPTK\\_GetMicrosecondsPerQuarterNote \(int bpm\)](#)  
*Convert BPM to duration or a quarter in microsecond*

### Properties

- int [MPTK\\_DeltaTicksPerQuarterNote](#) [get]  
*Get the DeltaTicksPerQuarterNote of the loaded midi*
- int [MPTK\\_TrackCount](#) [get]  
*Get the track count of the loaded midi*

- int [MPTK\\_MidiFileType](#) [get]  
Get the midi file type of the loaded midi (0,1,2)

## Detailed Description

PRO Version - Write a midi file from differents sources based on NAudio framework. See full example TestMidiWriter.cs with a light sequencer.

## Constructor & Destructor Documentation

### MidiPlayerTK.MidiFileWriter.MidiFileWriter ()

Create an empty [MidiFileWriter](#)

### MidiPlayerTK.MidiFileWriter.MidiFileWriter (int *deltaTicksPerQuarterNote*, int *midiFileType*)

Create a [MidiFileWriter](#) with an empty Midi Event list

#### Parameters:

<i>deltaTicksPerQuarterNote</i>	
<i>midiFileType</i>	

## Member Function Documentation

### void MidiPlayerTK.MidiFileWriter.MPTK\_AddEvent (int *track*, MidiEvent *midievent*)

Add a generic Midi event

#### Parameters:

<i>track</i>	
<i>midievent</i>	

### void MidiPlayerTK.MidiFileWriter.MPTK\_AddNote (int *track*, long *absoluteTime*, int *channel*, int *note*, int *velocity*, int *duration*)

Add a note event. the corresponding Noteoff is automatically created.

#### Parameters:

<i>track</i>	
<i>absoluteTime</i>	
<i>channel</i>	
<i>note</i>	

<i>velocity</i>	
<i>duration</i>	

**void MidiPlayerTK.MidiFileWriter.MPTK\_CreateTrack (int *count*)**

Create tracks

**Parameters:**

<i>count</i>	number of tracks to create
--------------	----------------------------

**void MidiPlayerTK.MidiFileWriter.MPTK\_EndTrack (int *trackNumber*)**

Close the track (mandatory for a well formed midi file)

**Parameters:**

<i>trackNumber</i>	Track number to close
--------------------	-----------------------

**static int MidiPlayerTK.MidiFileWriter.MPTK\_GetMicrosecondsPerQuarterNote (int *bpm*) [static]**

Convert BPM to duration or a quarter in microsecond

**Parameters:**

<i>bpm</i>	beat per measure
------------	------------------

**Returns:**

**bool MidiPlayerTK.MidiFileWriter.MPTK\_LoadFromFile (string *filename*)**

Load a Midi file from OS system file (could be dependant of the OS)

**Parameters:**

<i>filename</i>	
-----------------	--

**Returns:**

**bool MidiPlayerTK.MidiFileWriter.MPTK\_LoadFromMidiDB (int *indexMidiDb*)**

Create a [MidiFileWriter](#) from a Midi found in MPTK MidiDB

**Parameters:**

<i>indexMidiDb</i>	
--------------------	--

**bool MidiPlayerTK.MidiFileWriter.MPTK\_LoadFromMPTK (List< [TrackMidiEvent](#) > *MidiSorted*)**

Create a [MidiFileWriter](#) from a MPTK list of midi events. A midi file must be loaded before from a [MidiFilePlayer](#) gameobject (as in example) or from a call to `MidiFileWriter.MPTK_LoadFromFile(filename)`.

**Parameters:**

<i>MidiSorted</i>	
-------------------	--

**bool MidiPlayerTK.MidiFileWriter.MPTK\_WriteToFile (string *filename*)**

Write Midi file to an OS folder

**Parameters:**

<i>filename</i>	filename of the midi file
-----------------	---------------------------

**Returns:**

**bool MidiPlayerTK.MidiFileWriter.MPTK\_WriteToMidiDB (string *filename*)**

Write Midi file to MidiDB. To be used only in edit mode not in a standalone application.

**Parameters:**

<i>filename</i>	filename of the midi file without any folder and any extension
-----------------	--

**Returns:**

---

## Property Documentation

**int MidiPlayerTK.MidiFileWriter.MPTK\_DeltaTicksPerQuarterNote [get]**

Get the DeltaTicksPerQuarterNote of the loaded midi

**int MidiPlayerTK.MidiFileWriter.MPTK\_MidiFileType [get]**

Get the midi file type of the loaded midi (0,1,2)

**int MidiPlayerTK.MidiFileWriter.MPTK\_TrackCount [get]**

Get the track count of the loaded midi



## MidiPlayerTK.MidiListPlayer

PRO Version - Script for the prefab [MidiListPlayer](#). Play a list of pre-selected midi file from the dedicated inspector. List of Midi files must exists in MidiDB. See Midi Player Setup (Unity menu MPTK).

Inherits MonoBehaviour.

### Data Structures

- class [MPTK\\_MidiPlayItem](#)  
*Define a midi to be added in the list*

### Public Member Functions

- virtual void [MPTK\\_AddMidi](#) (string name)  
*Add a Midi name to the list. Use the exact name defined in Unity resources folder MidiDB without any path or extension. Tips: Add Midi files to your project with the Unity menu MPTK or add it directly in the ressource folder and open Midi File Setup to automatically integrate Midi in MPTK.*
- virtual void [MPTK\\_RemoveMidi](#) (string name)  
*Remove a Midi name from the list. Use the exact name defined in Unity resources folder MidiDB without any path or extension.*
- virtual void [MPTK\\_ReIndexMidi](#) ()  
*Recalculate the index of the midi from the list.*
- virtual void [MPTK\\_Play](#) ()  
*Play the midi file defined in MPTK\_MidiName*
- virtual void [MPTK\\_Stop](#) ()  
*Stop playing*
- virtual void [MPTK\\_RePlay](#) ()  
*Restart playing the current midi file*
- virtual void [MPTK\\_Pause](#) (float timeToPauseMS=-1f)  
*Pause the current playing*
- virtual void [MPTK\\_Next](#) ()  
*Play next Midi in list*
- virtual void [MPTK\\_Previous](#) ()  
*Play previous Midi in list*

### Data Fields

- List< [MPTK\\_MidiPlayItem](#) > [MPTK\\_PlayList](#)  
*Play list*
- UnityEvent [OnEventStartPlayMidi](#)  
*Define unity event to trigger at start*
- UnityEvent [OnEventEndPlayMidi](#)  
*Define unity event to trigger at end*
- [MidiFilePlayer](#) [MPTK\\_MidiFilePlayer\\_1](#)  
*[MidiFilePlayer](#) to play the Midi*

### Properties

- int [MPTK\\_PlayIndex](#) [get, set]  
*Play a specific Midi in the list.*
- virtual bool [MPTK\\_PlayOnStart](#) [get, set]  
*Should the Midi start playing when application start ?*

- virtual bool [MPTK\\_Loop](#) [get, set]  
*Should automatically restart when Midi reach the end ?*
- virtual bool [MPTK\\_IsPaused](#) [get]  
*Is Midi file playing is paused ?*
- virtual bool [MPTK\\_IsPlaying](#) [get]  
*Is Midi file is playing ?*

## Detailed Description

PRO Version - Script for the prefab [MidiListPlayer](#). Play a list of pre-selected midi file from the dedicated inspector. List of Midi files must exists in MidiDB. See Midi Player Setup (Unity menu MPTK).

## Member Function Documentation

**virtual void MidiPlayerTK.MidiListPlayer.MPTK\_AddMidi (string *name*) [virtual]**

Add a Midi name to the list. Use the exact name defined in Unity resources folder MidiDB without any path or extension. Tips: Add Midi files to your project with the Unity menu MPTK or add it directly in the ressource folder and open Midi File Setup to automatically integrate Midi in MPTK.

```
midiListPlayer.MPTK_AddMidi("Albinoni - Adagio");
```

**virtual void MidiPlayerTK.MidiListPlayer.MPTK\_Next () [virtual]**

Play next Midi in list

**virtual void MidiPlayerTK.MidiListPlayer.MPTK\_Pause (float *timeToPauseMS* = -1f) [virtual]**

Pause the current playing

### Parameters:

<i>timeToPauseMS</i>	time to pause in milliseconds. default: indefinitely
----------------------	--

**virtual void MidiPlayerTK.MidiListPlayer.MPTK\_Play () [virtual]**

Play the midi file defined in MPTK\_MidiName

**virtual void MidiPlayerTK.MidiListPlayer.MPTK\_Previous () [virtual]**

Play previous Midi in list

**virtual void MidiPlayerTK.MidiListPlayer.MPTK\_ReIndexMidi () [virtual]**

Recalculate the index of the midi from the list.

**virtual void MidiPlayerTK.MidiListPlayer.MPTK\_RemoveMidi (string *name*) [virtual]**

Remove a Midi name from the list. Use the exact name defined in Unity resources folder MidiDB without any path or extension.

```
midiListPlayer.MPTK_RemoveMidi("Albinoni - Adagio");
```

**virtual void MidiPlayerTK.MidiListPlayer.MPTK\_RePlay () [virtual]**

Restart playing the current midi file

**virtual void MidiPlayerTK.MidiListPlayer.MPTK\_Stop () [virtual]**

Stop playing

---

## Field Documentation

[MidiFilePlayer](#) MidiPlayerTK.MidiListPlayer.MPTK\_MidiFilePlayer\_1

[MidiFilePlayer](#) to play the Midi

List<[MPTK MidiPlayItem](#)> MidiPlayerTK.MidiListPlayer.MPTK\_PlayList

Play list

**UnityEvent MidiPlayerTK.MidiListPlayer.OnEventEndPlayMidi**

Define unity event to trigger at end

**UnityEvent MidiPlayerTK.MidiListPlayer.OnEventStartPlayMidi**

Define unity event to trigger at start

---

## Property Documentation

**virtual bool MidiPlayerTK.MidiListPlayer.MPTK\_IsPaused [get]**

Is Midi file playing is paused ?

**virtual bool MidiPlayerTK.MidiListPlayer.MPTK\_IsPlaying [get]**

Is Midi file is playing ?

**virtual bool MidiPlayerTK.MidiListPlayer.MPTK\_Loop [get], [set]**

Should automatically restart when Midi reach the end ?

**int MidiPlayerTK.MidiListPlayer.MPTK\_PlayIndex [get], [set]**

Play a specific Midi in the list.

**virtual bool MidiPlayerTK.MidiListPlayer.MPTK\_PlayOnStart [get], [set]**

Should the Midi start playing when application start ?

---

## MidiPlayerTK.MidiLoad

Base class for loading a Midi file. No sequencer, no synthesizer. Useful to load all the Midi events from a Midi.

### Public Member Functions

- bool [MPTK\\_Load](#) (int index)  
*Load Midi from midi MPTK referential (Unity resource).*
- bool [MPTK\\_Load](#) (byte[] datamidi)  
*Load Midi from an array of bytes*
- bool [MPTK\\_Load](#) (string midiname)  
*Load Midi from a Midi file from Unity resources. The Midi file must be present in Unity MidiDB resource folder.*
- bool [MPTK\\_Load](#) (string pathfilename, bool strict)

Load Midi from a folder from a folder anywhere on the desktop.

- List< [MPTKEvent](#) > [MPTK\\_ReadMidiEvents](#) (long fromTicks=0, long toTicks=long.MaxValue)  
Read the list of midi events available in the Midi from a ticks position to an end position.
- double [MPTK\\_ConvertTickToTime](#) (long tick)  
Convert the tick duration to a real time duration in millisecond regarding the current tempo.
- long [MPTK\\_ConvertTimeToTick](#) (double time)  
Convert a real time duration in millisecond to a number of tick regarding the current tempo.

## Data Fields

- TimeSpan [MPTK\\_Duration](#)  
Duration of the midi. Updated when ChangeSpeed is called.
- long [MPTK\\_TickLast](#)  
Last tick position in Midi: Time of the last midi event in sequence expressed in number of "ticks".  
 $MPTK\_TickLast / MPTK\_DeltaTicksPerQuarterNote$  equal the duration time of a quarter-note regardless the defined tempo.
- long [MPTK\\_TickCurrent](#)  
Current tick position in Midi: Time of the current midi event expressed in number of "ticks".  
 $MPTK\_TickCurrent / MPTK\_DeltaTicksPerQuarterNote$  equal the duration time of a quarter-note regardless the defined tempo.
- int [MPTK\\_NumberBeatsMeasure](#)  
From TimeSignature event: The numerator counts the number of beats in a measure. For example a numerator of 4 means that each bar contains four beats. This is important to know because usually the first beat of each bar has extra emphasis. <http://www.deluge.co/?q=midi-tempo-bpm>
- int [MPTK\\_NumberQuarterBeat](#)  
From TimeSignature event: number of quarter notes in a beat. Equal 2 Power TimeSigDenominator. <http://www.deluge.co/?q=midi-tempo-bpm>
- int [MPTK\\_TimeSigNumerator](#)  
From TimeSignature event: The numerator counts the number of beats in a measure. For example a numerator of 4 means that each bar contains four beats. This is important to know because usually the first beat of each bar has extra emphasis. In MIDI the denominator value is stored in a special format. i.e. the real denominator =  $2^{[dd]}$  <http://www.deluge.co/?q=midi-tempo-bpm>
- int [MPTK\\_TimeSigDenominator](#)  
From TimeSignature event: The denominator specifies the number of quarter notes in a beat. 2 represents a quarter-note, 3 represents an eighth-note, etc. .  
<http://www.deluge.co/?q=midi-tempo-bpm>
- int [MPTK\\_TicksInMetronomeClick](#)  
From TimeSignature event: The standard MIDI clock ticks every 24 times every quarter note (crotchet) so a [cc] value of 24 would mean that the metronome clicks once every quarter note. A [cc] value of 6 would mean that the metronome clicks once every 1/8th of a note (quaver).  
<http://www.deluge.co/?q=midi-tempo-bpm>
- int [MPTK\\_No32ndNotesInQuarterNote](#)  
From TimeSignature event: This value specifies the number of 1/32nds of a note happen every MIDI quarter note. It is usually 8 which means that a quarter note happens every quarter note.  
<http://www.deluge.co/?q=midi-tempo-bpm>
- int [MPTK\\_MicrosecondsPerQuarterNote](#)  
From the SetTempo event: The tempo is given in micro seconds per quarter beat. To convert this to BPM we needs to use the following equation: $BPM = 60,000,000/[tt \ tt \ tt]$   
<http://www.deluge.co/?q=midi-tempo-bpm>
- int [MPTK\\_DeltaTicksPerQuarterNote](#)  
Midi Header: Delta Ticks Per Quarter Note. Represent the duration time in "ticks" which make up a quarter-note. For instance, if 96, then a duration of an eighth-note in the file would be 48.

## Detailed Description

Base class for loading a Midi file. No sequencer, no synthesizer. Useful to load all the Midi events from a Midi.

---

## Member Function Documentation

### double MidiPlayerTK.MidiLoad.MPTK\_ConvertTickToTime (long *tick*)

Convert the tick duration to a real time duration in millisecond regarding the current tempo.

**Parameters:**

<i>tick</i>	duration in ticks
-------------	-------------------

**Returns:**

duration in milliseconds

### long MidiPlayerTK.MidiLoad.MPTK\_ConvertTimeToTick (double *time*)

Convert a real time duration in millisecond to a number of tick regarding the current tempo.

**Parameters:**

<i>time</i>	duration in milliseconds
-------------	--------------------------

**Returns:**

duration in ticks

### bool MidiPlayerTK.MidiLoad.MPTK\_Load (int *index*)

Load Midi from midi MPTK referential (Unity resource).

**Parameters:**

<i>index</i>	
--------------	--

### bool MidiPlayerTK.MidiLoad.MPTK\_Load (byte [] *datamidi*)

Load Midi from an array of bytes

**Parameters:**

<i>datamidi</i>	byte array midi
-----------------	-----------------

### bool MidiPlayerTK.MidiLoad.MPTK\_Load (string *midiname*)

Load Midi from a Midi file from Unity resources. The Midi file must be present in Unity MidiDB resource folder.

**Parameters:**

<i>midiname</i>	midi file name without path and extension
-----------------	---

**bool MidiPlayerTK.MidiLoad.MPTK\_Load (string *pathfilename*, bool *strict*)**

Load Midi from a folder from a folder anywhere on the desktop.

**Parameters:**

<i>pathfilename</i>	complete path + filename to the Midi file
<i>strict</i>	if true, check strict compliance with the Midi norm

**Returns:****List<[MPTKEvent](#)> MidiPlayerTK.MidiLoad.MPTK\_ReadMidiEvents (long *fromTicks* = 0, long *toTicks* = long.MaxValue)**

Read the list of midi events available in the Midi from a ticks position to an end position.

**Parameters:**

<i>fromTicks</i>	ticks start
<i>toTicks</i>	ticks end

**Returns:****Field Documentation****int MidiPlayerTK.MidiLoad.MPTK\_DeltaTicksPerQuarterNote**

Midi Header: Delta Ticks Per Quarter Note. Represent the duration time in "ticks" which make up a quarter-note. For instance, if 96, then a duration of an eighth-note in the file would be 48.

**TimeSpan MidiPlayerTK.MidiLoad.MPTK\_Duration**

Duration of the midi. Updated when ChangeSpeed is called.

**int MidiPlayerTK.MidiLoad.MPTK\_MicrosecondsPerQuarterNote**

From the SetTempo event: The tempo is given in micro seconds per quarter beat. To convert this to BPM we needs to use the following equation:  $BPM = 60,000,000/[tt \ tt \ tt]$   
<http://www.deluge.co/?q=midi-tempo-bpm>

**int MidiPlayerTK.MidiLoad.MPTK\_No32ndNotesInQuarterNote**

From TimeSignature event: This value specifies the number of 1/32nds of a note happen every MIDI quarter note. It is usually 8 which means that a quarter note happens every quarter note. <http://www.deluge.co/?q=midi-tempo-bpm>

#### **int MidiPlayerTK.MidiLoad.MPTK\_NumberBeatsMeasure**

From TimeSignature event: The numerator counts the number of beats in a measure. For example a numerator of 4 means that each bar contains four beats. This is important to know because usually the first beat of each bar has extra emphasis. <http://www.deluge.co/?q=midi-tempo-bpm>

#### **int MidiPlayerTK.MidiLoad.MPTK\_NumberQuarterBeat**

From TimeSignature event: number of quarter notes in a beat. Equal 2 Power TimeSigDenominator. <http://www.deluge.co/?q=midi-tempo-bpm>

#### **long MidiPlayerTK.MidiLoad.MPTK\_TickCurrent**

Current tick position in Midi: Time of the current midi event expressed in number of "ticks".  $MPTK\_TickCurrent / MPTK\_DeltaTicksPerQuarterNote$  equal the duration time of a quarter-note regardless the defined tempo.

#### **long MidiPlayerTK.MidiLoad.MPTK\_TickLast**

Last tick position in Midi: Time of the last midi event in sequence expressed in number of "ticks".  $MPTK\_TickLast / MPTK\_DeltaTicksPerQuarterNote$  equal the duration time of a quarter-note regardless the defined tempo.

#### **int MidiPlayerTK.MidiLoad.MPTK\_TicksInMetronomeClick**

From TimeSignature event: The standard MIDI clock ticks every 24 times every quarter note (crotchet) so a [cc] value of 24 would mean that the metronome clicks once every quarter note. A [cc] value of 6 would mean that the metronome clicks once every 1/8th of a note (quaver). <http://www.deluge.co/?q=midi-tempo-bpm>

#### **int MidiPlayerTK.MidiLoad.MPTK\_TimeSigDenominator**

From TimeSignature event: The denominator specifies the number of quarter notes in a beat. 2 represents a quarter-note, 3 represents an eighth-note, etc. . <http://www.deluge.co/?q=midi-tempo-bpm>

#### **int MidiPlayerTK.MidiLoad.MPTK\_TimeSigNumerator**

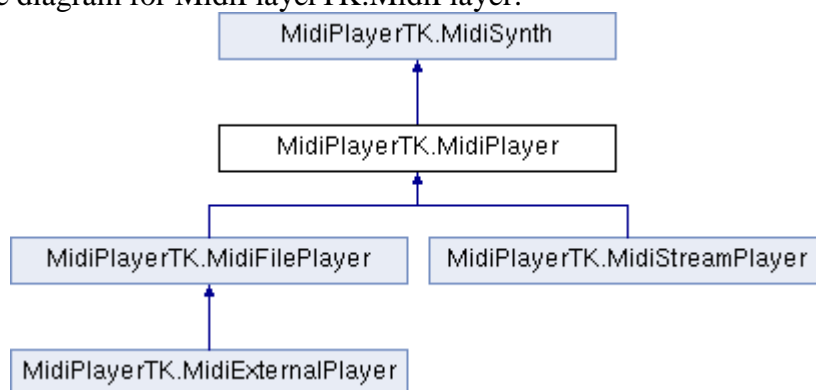


From TimeSignature event: The numerator counts the number of beats in a measure. For example a numerator of 4 means that each bar contains four beats. This is important to know because usually the first beat of each bar has extra emphasis. In MIDI the denominator value is stored in a special format. i.e. the real denominator =  $2^{[dd]}$  <http://www.deluge.co/?q=midi-tempo-bpm>

## MidiPlayerTK.MidiPlayer

Send event to the midi synthetizer thru thread. Don't instanciate this class, use rather [MidiFilePlayer](#) or [MidiStreamPlayer](#).

Inheritance diagram for MidiPlayerTK.MidiPlayer:



### Public Member Functions

- void [MPTK\\_InitSynth](#) (int channelCount=16)  
*Init the synthesizer. Prefabs automatically initialize the synthesizer (see events). It's not usefull to call this method.*
- void [MPTK\\_ClearAllSound](#) (bool destroyAudioSource=false)  
*Clear all sound*

### Data Fields

- EventSynthClass [OnEventSynthAwake](#)  
*Unity event fired at awake of the synthesizer. Name of the gameobject component is passed as a parameter.*
- EventSynthClass [OnEventSynthStarted](#)  
*Unity event fired at start of the synthesizer. Name of the gameobject component is passed as a parameter.*
- bool [MPTK\\_PauseOnDistance](#)  
*Should the Midi playing must be paused if distance between AudioListener and [MidiFilePlayer](#) is greater than MaxDistance*
- bool [MPTK\\_EnablePanChange](#)  
*Should change pan from Midi Events or from SoundFont ?*
- bool [MPTK\\_WeakDevice](#)  
*Should play on a weak device (cheaper smartphone) ? Playing Midi files with WeakDevice activated could cause some bad interpretation of Midi Event, consequently bad sound.*
- float [MPTK\\_ReleaseTimeMin](#) = 50f  
*Define a minimum release time at noteoff in milliseconds. Default 50 ms is a good tradeoff. Below some unpleasant sound coule be heard.*

## Properties

- virtual bool [MPTK\\_EnablePresetDrum](#) [get, set]  
*Should accept change Preset for Drum canal 10 ? Disabled by default. Could sometimes create bad sound with midi files not really compliant with the Midi norm.*
  - virtual float [MPTK\\_MaxDistance](#) [get, set]  
*MaxDistance to use for PauseOnDistance*
  - virtual float [MPTK\\_Volume](#) [get, set]  
*Volume of midi playing. Must be >=0 and <= 1*
  - virtual int [MPTK\\_Transpose](#) [get, set]  
*Transpose note from -24 to 24*
- 

## Detailed Description

Send event to the midi synthetizer thru thread. Don't instanciate this class, use rather [MidiFilePlayer](#) or [MidiStreamPlayer](#).

---

## Member Function Documentation

**void MidiPlayerTK.MidiPlayer.MPTK\_ClearAllSound (bool *destroyAudioSource* = false)**

Clear all sound

### Parameters:

<i>destroyAudioSource</i>	Destroy also audioSource (default:false)
---------------------------	--

```
if (GUILayout.Button("Clear"))
    midiStreamPlayer.MPTK_ClearAllSound(true);
```

**void MidiPlayerTK.MidiPlayer.MPTK\_InitSynth (int *channelCount* = 16)**

Init the synthetizer. Prefabs automatically initialize the synthetizer (see events). It's not usefull to call this method.

### Parameters:

<i>channelCount</i>	Number of channel to create
---------------------	-----------------------------

---

## Field Documentation

**bool MidiPlayerTK.MidiSynth.MPTK\_EnablePanChange [inherited]**

Should change pan from Midi Events or from SoundFont ?

### **bool MidiPlayerTK.MidiSynth.MPTK\_PauseOnDistance [inherited]**

Should the Midi playing must be paused if distance between AudioListener and [MidiFilePlayer](#) is greater than MaxDistance

### **float MidiPlayerTK.MidiSynth.MPTK\_ReleaseTimeMin = 50f [inherited]**

Define a minimum release time at noteoff in milliseconds. Default 50 ms is a good tradeoff. Below some unpleasant sound could be heard.

### **bool MidiPlayerTK.MidiSynth.MPTK\_WeakDevice [inherited]**

Should play on a weak device (cheaper smartphone) ? Playing Midi files with WeakDevice activated could cause some bad interpretation of Midi Event, consequently bad sound.

### **EventSynthClass MidiPlayerTK.MidiSynth.OnEventSynthAwake [inherited]**

Unity event fired at awake of the synthesizer. Name of the gameobject component is passed as a parameter.

```
...
if (!midiStreamPlayer.OnEventSynthAwake.HasEvent())
    midiStreamPlayer.OnEventSynthAwake.AddListener(StartLoadingSynth);
...
public void StartLoadingSynth(string name)
{
    Debug.LogFormat("Synth {0} loading", name);
}
```

### **EventSynthClass MidiPlayerTK.MidiSynth.OnEventSynthStarted [inherited]**

Unity event fired at start of the synthesizer. Name of the gameobject component is passed as a parameter.

```
...
if (!midiStreamPlayer.OnEventStartSynth.HasEvent())
    midiStreamPlayer.OnEventStartSynth.AddListener(EndLoadingSynth);
...
public void EndLoadingSynth(string name)
{
    Debug.LogFormat("Synth {0} loaded", name);
    midiStreamPlayer.MPTK_PlayEvent(
        new MPTKEvent() { Command = MPTKCommand.PatchChange, Value =
CurrentPatchInstrument, Channel = StreamChannel});
}
```

## Property Documentation

virtual bool MidiPlayerTK.MidiPlayer.MPTK\_EnablePresetDrum [get], [set]

Should accept change Preset for Drum canal 10 ? Disabled by default. Could sometimes create bad sound with midi files not really compliant with the Midi norm.

virtual float MidiPlayerTK.MidiSynth.MPTK\_MaxDistance [get], [set], [inherited]

MaxDistance to use for PauseOnDistance

virtual int MidiPlayerTK.MidiSynth.MPTK\_Transpose [get], [set], [inherited]

Transpose note from -24 to 24

virtual float MidiPlayerTK.MidiSynth.MPTK\_Volume [get], [set], [inherited]

Volume of midi playing. Must be  $\geq 0$  and  $\leq 1$

---

## MidiPlayerTK.MidiPlayerGlobal

Singleton class to manage all global features of MPTK.  
Inherits MonoBehaviour.

### Static Public Member Functions

- static bool [MPTK\\_IsReady](#) (float delay=0.5f)  
*Check if SoudFont is loaded. Add a default wait time because Unity AudioSource need a delay to be really ready to play. Hummm, like a diesel motor ?*
- static void [MPTK\\_SelectSoundFont](#) (string name)  
*Changing the current Soundfont on fly. If some Midis are playing they are restarted.*
- static void [MPTK\\_SelectBankInstrument](#) (int nbank)  
*Change default current bank on fly*
- static void [MPTK\\_SelectBankDrum](#) (int nbank)  
*Change current bank on fly*
- static int [MPTK\\_FindMidi](#) (string name)  
*Find index of a Midi by name. Use the exact name defined in Unity resources folder MidiDB without any path or extension. Tips: Add Midi files to your project with the Unity menu MPTK or add it directly in the ressource folder and open Midi File Setup to automatically integrate Midi in MPTK.*
- static float [MPTK\\_DistanceToListener](#) (Transform trf)  
*Calculate distance with the AudioListener.*

## Static Public Attributes

- static string [MPTK\\_PathToResources](#) = "MidiPlayer/Resources/"  
*This path could change depending your project. Change the path before any actions in MPTK.*
- static int [MPTK\\_CountWaveLoaded](#)  
*Count of wave loaded*
- static bool [MPTK\\_SoundFontLoaded](#) = false  
*True if soundfont is loaded*
- static List< [MPTKListItem](#) > [MPTK\\_ListMidi](#)  
*List of midi(s) available*
- static List< [MPTKListItem](#) > [MPTK\\_ListPreset](#)  
*Get the list of presets available for instruments for the selected bank*
- static List< [MPTKListItem](#) > [MPTK\\_ListBank](#)  
*Get the list of banks available*
- static List< [MPTKListItem](#) > [MPTK\\_ListPresetDrum](#)  
*Get the list of presets available for instrument*
- static List< [MPTKListItem](#) > [MPTK\\_ListDrum](#)  
*Get the list of presets available*

## Properties

- static TimeSpan [MPTK\\_TimeToLoadSoundFont](#) [get]  
*Load time for the current SoundFont*
- static TimeSpan [MPTK\\_TimeToLoadWave](#) [get]  
*Load time for the wave*
- static int [MPTK\\_CountPresetLoaded](#) [get]  
*Count of preset loaded*
- static UnityEvent [OnEventPresetLoaded](#) [get, set]  
*Event triggered at end of loading a soundfont. Warning: when defined by script, this event is not triggered at first load of MPTK because [MidiPlayerGlobal](#) is loaded before any other gamecomponent. Set this event in the Inspector of [MidiPlayerGlobal](#) to get at first load this information.*
- static List< string > [MPTK\\_ListSoundFont](#) [get]  
*List of Soundfont(s) available*

---

## Detailed Description

Singleton class to manage all global features of MPTK.

---

## Member Function Documentation

**static float MidiPlayerTK.MidiPlayerGlobal.MPTK\_DistanceToListener (Transform *trf*) [static]**

Calculate distance with the AudioListener.

### Parameters:

<i>trf</i>	Transform of the object to calculate the distance.
------------	--

**Returns:**

**static int MidiPlayerTK.MidiPlayerGlobal.MPTK\_FindMidi (string *name*)[static]**

Find index of a Midi by name. Use the exact name defined in Unity resources folder MidiDB without any path or extension. Tips: Add Midi files to your project with the Unity menu MPTK or add it directly in the ressource folder and open Midi File Setup to automatically integrate Midi in MPTK.

**Parameters:**

<i>name</i>	name of the midi without path nor extension
-------------	---

**Returns:**

-1 if not found else return the index of the midi.

**static bool MidiPlayerTK.MidiPlayerGlobal.MPTK\_IsReady (float *delay* = 0.5f)[static]**

Check if SoudFont is loaded. Add a default wait time because Unity AudioSource need a delay to be really ready to play. Hummm, like a diesel motor ?

**Parameters:**

<i>delay</i>	
--------------	--

**Returns:**

**static void MidiPlayerTK.MidiPlayerGlobal.MPTK\_SelectBankDrum (int *nbank*)[static]**

Change current bank on fly

**Parameters:**

<i>nbank</i>	Number of the SoundFont Bank to load for drum.
--------------	--

**static void MidiPlayerTK.MidiPlayerGlobal.MPTK\_SelectBankInstrument (int *nbank*)[static]**

Change default current bank on fly

**Parameters:**

<i>nbank</i>	Number of the SoundFont Bank to load for instrument.
--------------	--

**static void MidiPlayerTK.MidiPlayerGlobal.MPTK\_SelectSoundFont (string *name*)[static]**

Changing the current Soundfont on fly. If some Midis are playing they are restarted.

**Parameters:**

<i>name</i>	SoundFont name
-------------	----------------

---

**Field Documentation**

**int MidiPlayerTK.MidiPlayerGlobal.MPTK\_CountWaveLoaded [static]**

Count of wave loaded

**List<[MPTKListItem](#)> MidiPlayerTK.MidiPlayerGlobal.MPTK\_ListBank [static]**

Get the list of banks available

**List<[MPTKListItem](#)> MidiPlayerTK.MidiPlayerGlobal.MPTK\_ListDrum [static]**

Get the list of presets available

**List<[MPTKListItem](#)> MidiPlayerTK.MidiPlayerGlobal.MPTK\_ListMidi [static]**

List of midi(s) available

**List<[MPTKListItem](#)> MidiPlayerTK.MidiPlayerGlobal.MPTK\_ListPreset [static]**

Get the list of presets available for instruments for the selected bank

**List<[MPTKListItem](#)> MidiPlayerTK.MidiPlayerGlobal.MPTK\_ListPresetDrum [static]**

Get the list of presets available for instrument

**string MidiPlayerTK.MidiPlayerGlobal.MPTK\_PathToResources = "MidiPlayer/Resources/" [static]**

This path could change depending your project. Change the path before any actions in MPTK.

**bool MidiPlayerTK.MidiPlayerGlobal.MPTK\_SoundFontLoaded = false [static]**

True if soundfont is loaded

---

## Property Documentation

**int MidiPlayerTK.MidiPlayerGlobal.MPTK\_CountPresetLoaded** [static], [get]

Count of preset loaded

**List<string> MidiPlayerTK.MidiPlayerGlobal.MPTK\_ListSoundFont** [static], [get]

List of Soundfont(s) available

**TimeSpan MidiPlayerTK.MidiPlayerGlobal.MPTK\_TimeToLoadSoundFont** [static], [get]

Load time for the current SoundFont

**TimeSpan MidiPlayerTK.MidiPlayerGlobal.MPTK\_TimeToLoadWave** [static], [get]

Load time for the wave

**UnityEvent MidiPlayerTK.MidiPlayerGlobal.OnEventPresetLoaded** [static], [get], [set]

Event triggered at end of loading a soundfont. Warning: when defined by script, this event is not triggered at first load of MPTK because [MidiPlayerGlobal](#) is loaded before any other gamecomponent. Set this event in the Inspector of [MidiPlayerGlobal](#) to get at first load this information.

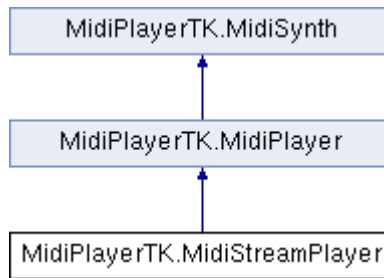
---

## MidiPlayerTK.MidiStreamPlayer

Play generated notes. Any Midi file is necessary rather create music from your own algorithm with [MPTK\\_PlayEvent\(\)](#). Duration can be set in the [MPTKEvent](#), but a note can also be stopped with [MPTK\\_StopEvent\(\)](#).

Inheritance diagram for MidiPlayerTK.MidiStreamPlayer:





## Public Member Functions

- virtual void [MPTK\\_PlayEvent](#) ([MPTKEvent](#) note)  
*Play one midi event with a thread so the call return immediately.*

```

midiStreamPlayer.MPTK_PlayEvent
(
    new MPTKEvent()
    {
        Channel = 9,
        Duration = 0.2f,
        Value = 60,
        Velocity = 100
    }
);
  
```

- virtual void [MPTK\\_PlayEvent](#) (List< [MPTKEvent](#) > notes)  
*Play a list of midi events with a thread so the call return immediately.*

```

void Update()
{
    // Checj that SoundFont is loaded and add a little wait (0.5 s by default)
    because Unity AudioSource need some time to be started
    if (!MidiPlayerGlobal.MPTK_IsReady())
        return;

    if (midiStreamPlayer != null && Isplaying)
    {
        float time = Time.realtimeSinceStartup - LastTimeChange;
        if (time > DelayTimeChange)
        {
            // It's time to generate a note
            LastTimeChange = Time.realtimeSinceStartup;

            if (RandomPlay)
            {
                //
                // First method to play notes: send a list of notes directly
                to the MidiStreamPlayer // Useful for a long list of notes when the duration of the
                note is lnown.
                //
                List<MPTKEvent> notes = new List<MPTKEvent>();
                // Very light random notes generator
                if (!DrumKit)
                {
                    // Play 3 notes with no delay
                    int rnd = UnityEngine.Random.Range(-8, 8);
                    notes.Add(CreateNote(60 + rnd, 0));
                    notes.Add(CreateNote(64 + rnd, 0));
                    notes.Add(CreateNote(67 + rnd, 0));
                }
                else
                {
                    // Play 3 hit with a short delay
                    notes.Add(CreateDrum(UnityEngine.Random.Range(0, 127),
0));
                    notes.Add(CreateDrum(UnityEngine.Random.Range(0, 127),
150));
                    notes.Add(CreateDrum(UnityEngine.Random.Range(0, 127),
300));
                }
            }
        }
    }
}
  
```



---

## Detailed Description

Play generated notes. Any Midi file is necessary rather create music from your own algorithm with [MPTK\\_PlayEvent\(\)](#). Duration can be set in the [MPTKEvent](#), but a note can also be stopped with [MPTK\\_StopEvent\(\)](#).

---

## Member Function Documentation

**void MidiPlayerTK.MidiPlayer.MPTK\_ClearAllSound (bool *destroyAudioSource* = false)[inherited]**

Clear all sound

### Parameters:

<i>destroyAudioSource</i>	Destroy also audioSource (default:false)
---------------------------	--

```
if (GUILayout.Button("Clear"))
    midiStreamPlayer.MPTK_ClearAllSound(true);
```

**void MidiPlayerTK.MidiPlayer.MPTK\_InitSynth (int *channelCount* = 16)[inherited]**

Init the synthesizer. Prefabs automatically initialize the synthesizer (see events). It's not useful to call this method.

### Parameters:

<i>channelCount</i>	Number of channel to create
---------------------	-----------------------------

**virtual void MidiPlayerTK.MidiStreamPlayer.MPTK\_PlayEvent ([MPTKEvent](#) *note*)[virtual]**

Play one midi event with a thread so the call return immediately.

```
midiStreamPlayer.MPTK_PlayEvent
(
    new MPTKEvent ()
    {
        Channel = 9,
        Duration = 0.2f,
        Value = 60,
        Velocity = 100
    }
);
```

**virtual void MidiPlayerTK.MidiStreamPlayer.MPTK\_PlayEvent (List< [MPTKEvent](#) > *notes*)[virtual]**

Play a list of midi events with a thread so the call return immediately.

```

void Update ()
{
    // Checj that SoundFont is loaded and add a little wait (0.5 s by default)
    because Unity AudioSource need some time to be started
    if (!MidiPlayerGlobal.MPTK IsReady())
        return;

    if (midiStreamPlayer != null && Isplaying)
    {
        float time = Time.realtimeSinceStartup - LastTimeChange;
        if (time > DelayTimeChange)
        {
            // It's time to generate a note
            LastTimeChange = Time.realtimeSinceStartup;

            if (RandomPlay)
            {
                //
                // First method to play notes: send a list of notes directly
                to the MidiStreamPlayer // Useful for a long list of notes when the duration of the
                note is lnown.
                //
                List<MPTKEvent> notes = new List<MPTKEvent>();
                // Very light random notes generator
                if (!DrumKit)
                {
                    // Play 3 notes with no delay
                    int rnd = UnityEngine.Random.Range(-8, 8);
                    notes.Add(CreateNote(60 + rnd, 0));
                    notes.Add(CreateNote(64 + rnd, 0));
                    notes.Add(CreateNote(67 + rnd, 0));
                }
                else
                {
                    // Play 3 hit with a short delay
                    notes.Add(CreateDrum(UnityEngine.Random.Range(0, 127),
0));
                    notes.Add(CreateDrum(UnityEngine.Random.Range(0, 127),
150));
                    notes.Add(CreateDrum(UnityEngine.Random.Range(0, 127),
300));
                }
                // Send the note to the player. Notes are plays in a thread,
                so call returns immediatly
                midiStreamPlayer.MPTK PlayEvent (notes);
            }
            else
            {
                //
                // Second method to play and stop a notes: the duration is not
                known
                // Here, a new note stop the previous
                //
                if (++CurrentNote > EndNote) CurrentNote = StartNote;
                if (CurrentNote < StartNote) CurrentNote = StartNote;
                PlayOneNote();
            }
        }
    }
}

```

**virtual void MidiPlayerTK.MidiStreamPlayer.MPTK\_StopEvent ([MPTKEvent](#) pnote)[virtual]**

Stop playing the note. All waves associated to the note are stop by sending a noteoff.

**Parameters:**

<i>pnote</i>	
--------------	--

---

**Field Documentation****bool MidiPlayerTK.MidiSynth.MPTK\_EnablePanChange [inherited]**

Should change pan from Midi Events or from SoundFont ?

**bool MidiPlayerTK.MidiSynth.MPTK\_PauseOnDistance [inherited]**

Should the Midi playing must be paused if distance between AudioListener and [MidiFilePlayer](#) is greater than MaxDistance

**float MidiPlayerTK.MidiSynth.MPTK\_ReleaseTimeMin = 50f [inherited]**

Define a minimum release time at noteoff in milliseconds. Default 50 ms is a good tradeoff. Below some unpleasant sound could be heard.

**bool MidiPlayerTK.MidiSynth.MPTK\_WeakDevice [inherited]**

Should play on a weak device (cheaper smartphone) ? Playing Midi files with WeakDevice activated could cause some bad interpretation of Midi Event, consequently bad sound.

**EventSynthClass MidiPlayerTK.MidiSynth.OnEventSynthAwake [inherited]**

Unity event fired at awake of the synthesizer. Name of the gameobject component is passed as a parameter.

```

...
if (!midiStreamPlayer.OnEventSynthAwake.HasEvent())
    midiStreamPlayer.OnEventSynthAwake.AddListener(StartLoadingSynth);
...
public void StartLoadingSynth(string name)
{
    Debug.LogFormat("Synth {0} loading", name);
}

```

**EventSynthClass MidiPlayerTK.MidiSynth.OnEventSynthStarted [inherited]**

Unity event fired at start of the synthesizer. Name of the gameobject component is passed as a parameter.

```

...
if (!midiStreamPlayer.OnEventStartSynth.HasEvent())

```

```

midiStreamPlayer.OnEventStartSynth.AddListener(EndLoadingSynth);
...
public void EndLoadingSynth(string name)
{
    Debug.LogFormat("Synth {0} loaded", name);
    midiStreamPlayer.MPTK_PlayEvent(
        new MPTKEvent() { Command = MPTKCommand.PatchChange, Value =
CurrentPatchInstrument, Channel = StreamChannel});
}

```

---

## Property Documentation

**virtual bool MidiPlayerTK.MidiPlayer.MPTK\_EnablePresetDrum** [get], [set], [inherited]

Should accept change Preset for Drum canal 10 ? Disabled by default. Could sometimes create bad sound with midi files not really compliant with the Midi norm.

**virtual float MidiPlayerTK.MidiSynth.MPTK\_MaxDistance** [get], [set], [inherited]

MaxDistance to use for PauseOnDistance

**virtual int MidiPlayerTK.MidiSynth.MPTK\_Transpose** [get], [set], [inherited]

Transpose note from -24 to 24

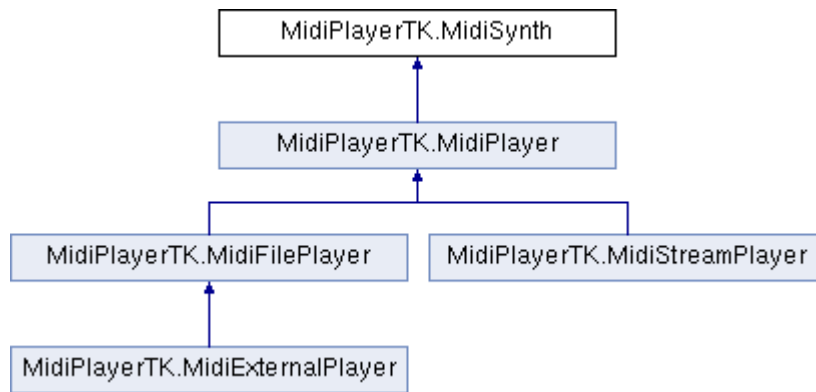
**virtual float MidiPlayerTK.MidiSynth.MPTK\_Volume** [get], [set], [inherited]

Volume of midi playing. Must be  $\geq 0$  and  $\leq 1$

---

## MidiPlayerTK.MidiSynth

Inheritance diagram for MidiPlayerTK.MidiSynth:



## Data Fields

- EventSynthClass [OnEventSynthAwake](#)  
*Unity event fired at awake of the synthesizer. Name of the gameobject component is passed as a parameter.*
- EventSynthClass [OnEventSynthStarted](#)  
*Unity event fired at start of the synthesizer. Name of the gameobject component is passed as a parameter.*
- bool [MPTK\\_PauseOnDistance](#)  
*Should the Midi playing must be paused if distance between AudioListener and [MidiFilePlayer](#) is greater than MaxDistance*
- bool [MPTK\\_EnablePanChange](#)  
*Should change pan from Midi Events or from SoundFont ?*
- bool [MPTK\\_WeakDevice](#)  
*Should play on a weak device (cheaper smartphone) ? Playing Midi files with WeakDevice activated could cause some bad interpretation of Midi Event, consequently bad sound.*
- float [MPTK\\_ReleaseTimeMin](#) = 50f  
*Define a minimum release time at noteoff in milliseconds. Default 50 ms is a good tradeoff. Below some unpleasant sound could be heard.*

## Properties

- virtual float [MPTK\\_MaxDistance](#) [get, set]  
*MaxDistance to use for PauseOnDistance*
- virtual float [MPTK\\_Volume](#) [get, set]  
*Volume of midi playing. Must be >=0 and <= 1*
- virtual int [MPTK\\_Transpose](#) [get, set]  
*Transpose note from -24 to 24*

## Detailed Description

Base class for Midi Synthesizer. Migrated from fluidsynth. It's not recommended to instantiate this class. Instead use [MidiFilePlayer](#) or [MidiStreamPlayer](#).

## Field Documentation

**bool MidiPlayerTK.MidiSynth.MPTK\_EnablePanChange**

Should change pan from Midi Events or from SoundFont ?

### **bool MidiPlayerTK.MidiSynth.MPTK\_PauseOnDistance**

Should the Midi playing must be paused if distance between AudioListener and [MidiFilePlayer](#) is greater than MaxDistance

### **float MidiPlayerTK.MidiSynth.MPTK\_ReleaseTimeMin = 50f**

Define a minimum release time at noteoff in milliseconds. Default 50 ms is a good tradeoff. Below some unpleasant sound could be heard.

### **bool MidiPlayerTK.MidiSynth.MPTK\_WeakDevice**

Should play on a weak device (cheaper smartphone) ? Playing Midi files with WeakDevice activated could cause some bad interpretation of Midi Event, consequently bad sound.

### **EventSynthClass MidiPlayerTK.MidiSynth.OnEventSynthAwake**

Unity event fired at awake of the synthesizer. Name of the gameobject component is passed as a parameter.

```
...
if (!midiStreamPlayer.OnEventSynthAwake.HasEvent())
    midiStreamPlayer.OnEventSynthAwake.AddListener(StartLoadingSynth);
...
public void StartLoadingSynth(string name)
{
    Debug.LogFormat("Synth {0} loading", name);
}
```

### **EventSynthClass MidiPlayerTK.MidiSynth.OnEventSynthStarted**

Unity event fired at start of the synthesizer. Name of the gameobject component is passed as a parameter.

```
...
if (!midiStreamPlayer.OnEventStartSynth.HasEvent())
    midiStreamPlayer.OnEventStartSynth.AddListener(EndLoadingSynth);
...
public void EndLoadingSynth(string name)
{
    Debug.LogFormat("Synth {0} loaded", name);
    midiStreamPlayer.MPTK_PlayEvent(
        new MPTKEvent() { Command = MPTKCommand.PatchChange, Value =
CurrentPatchInstrument, Channel = StreamChannel});
}
```



## Property Documentation

virtual float **MidiPlayerTK.MidiSynth.MPTK\_MaxDistance** [get], [set]

MaxDistance to use for PauseOnDistance

virtual int **MidiPlayerTK.MidiSynth.MPTK\_Transpose** [get], [set]

Transpose note from -24 to 24

virtual float **MidiPlayerTK.MidiSynth.MPTK\_Volume** [get], [set]

Volume of midi playing. Must be  $\geq 0$  and  $\leq 1$

---

## MidiPlayerTK.MidiListPlayer.MPTK\_MidiPlayItem

Define a midi to be added in the list

### Data Fields

- string [MidiName](#)  
*Midi Name. Use the exact name defined in Unity resources folder MidiDB without any path or extension. Tips: Add Midi files to your project with the Unity menu MPTK or add it directly in the ressource folder and open Midi File Setup to automatically integrate Midi in MPTK.*
- bool [Selected](#)  
*Select or unselect this Midi in the Inspector to apply actions (reorder, delete, ...)*
- int [Index](#)  
*Position of the Midi in the list. Use method [MPTK\\_ReIndexMidi\(\)](#) recalculate the index.*

---

### Detailed Description

Define a midi to be added in the list

---

### Field Documentation

int **MidiPlayerTK.MidiListPlayer.MPTK\_MidiPlayItem.Index**

Position of the Midi in the list. Use method [MPTK\\_ReIndexMidi\(\)](#) recalculate the index.

## string MidiPlayerTK.MidiListPlayer.MPTK\_MidiPlayItem.MidiName

Midi Name. Use the exact name defined in Unity resources folder MidiDB without any path or extension. Tips: Add Midi files to your project with the Unity menu MPTK or add it directly in the resource folder and open Midi File Setup to automatically integrate Midi in MPTK.

## bool MidiPlayerTK.MidiListPlayer.MPTK\_MidiPlayItem.Selected

Select or unselect this Midi in the Inspector to apply actions (reorder, delete, ...)

---

## MidiPlayerTK.MPTKEvent

Midi Event class for MPTK. Usage to generate Midi Music with [MidiStreamPlayer](#) or to read midi events from a Midi file with [MidiLoad](#) or to receive midi events from [MidiFilePlayer](#) OnEventNotesMidi.

### Public Types

- enum [EnumLength](#)

### Note length as [https://en.wikipedia.org/wiki/Note\\_value](https://en.wikipedia.org/wiki/Note_value) Public Member Functions

- void [Play](#) ([MidiStreamPlayer](#) streamPlayer)  
*Play a note which is stoppable. DEPRECATED in V2. Replaced by MPTK\_PlayEvent in [MidiStreamPlayer](#).*
- void [Stop](#) ()  
*Stop the note. DEPRECATED in V2. Replaced by MPTK\_StopEvent in [MidiStreamPlayer](#).*

### Data Fields

- long [Tick](#)  
*Time in Midi Tick (part of a Beat) of the Event since the start of playing the midi file. This time is independant of the Tempo or Speed. Not used for [MidiStreamPlayer](#).*
- [MPTKCommand](#) [Command](#)  
*Midi Command code. Defined the type of message (Note On, Control Change, Patch Change...)*
- [MPTKController](#) [Controller](#)  
*Controller code. When the Command is ControlChange, contains the code fo the controller to change (Modulation, Pan, Bank Select ...). Value will contains the value of the controller.*
- [MPTKMeta](#) [Meta](#)  
*MetaEvent Code. When the Command is MetaEvent, contains the code of the meta event (Lyric, TimeSignature, ...). . Info will contains the value of the meta.*
- string [Info](#)  
*Information hold by textual meta event when Command=MetaEvent*
- int [Value](#)  
*Contains a value between 0 and 127 in relation with the Command. For:*
- int [Channel](#)  
*Midi channel fom 0 to 15 (9 for drum)*

- int [Velocity](#)  
*Velocity between 0 and 127*
- double [Duration](#)  
*Duration of the note in millisecond*
- int [Length](#)  
*Duration of the note in Midi Tick. [MidiFilePlayer.MPTK\\_NoteLength](#) can be used to convert this duration. Not used for [MidiStreamPlayer](#). [https://en.wikipedia.org/wiki/Note\\_value](https://en.wikipedia.org/wiki/Note_value)*
- List< fluid\_voice > [Voices](#)  
*List of voices associated to this Event for playing a NoteOn event.*

## Detailed Description

Midi Event class for MPTK. Usage to generate Midi Music with [MidiStreamPlayer](#) or to read midi events from a Midi file with [MidiLoad](#) or to receive midi events from [MidiFilePlayer](#) OnEventNotesMidi.

## Member Enumeration Documentation

enum [MidiPlayerTK.MPTKEvent.EnumLength](#) [**strong**]

Note length as [https://en.wikipedia.org/wiki/Note\\_value](https://en.wikipedia.org/wiki/Note_value)

## Member Function Documentation

void [MidiPlayerTK.MPTKEvent.Play](#) ([MidiStreamPlayer](#) *streamPlayer*)

Play a note which is stoppable. DEPRECATED in V2. Replaced by MPTK\_PlayEvent in [MidiStreamPlayer](#).

### Parameters:

<i>streamPlayer</i>	A <a href="#">MidiStreamPlayer</a> component
---------------------	--

void [MidiPlayerTK.MPTKEvent.Stop](#) ()

Stop the note. DEPRECATED in V2. Replaced by MPTK\_StopEvent in [MidiStreamPlayer](#).

## Field Documentation

int [MidiPlayerTK.MPTKEvent.Channel](#)

Midi channel fom 0 to 15 (9 for drum)

### **MPTKCommand MidiPlayerTK.MPTKEvent.Command**

Midi Command code. Defined the type of message (Note On, Control Change, Patch Change...)

### **MPTKController MidiPlayerTK.MPTKEvent.Controller**

Controller code. When the Command is ControlChange, contains the code fo the controller to change (Modulation, Pan, Bank Select ...). Value will contains the value of the controller.

### **double MidiPlayerTK.MPTKEvent.Duration**

Duration of the note in millisecond

### **string MidiPlayerTK.MPTKEvent.Info**

Information hold by textual meta event when Command=MetaEvent

### **int MidiPlayerTK.MPTKEvent.Length**

Duration of the note in Midi Tick. [MidiFilePlayer.MPTK\\_NoteLength](#) can be used to convert this duration. Not used for [MidiStreamPlayer](#). [https://en.wikipedia.org/wiki/Note\\_value](https://en.wikipedia.org/wiki/Note_value)

### **MPTKMeta MidiPlayerTK.MPTKEvent.Meta**

MetaEvent Code. When the Command is MetaEvent, contains the code of the meta event (Lyric, TimeSignature, ...). . Info will contains the value of the meta.

### **long MidiPlayerTK.MPTKEvent.Tick**

Time in Midi Tick (part of a Beat) of the Event since the start of playing the midi file. This time is independant of the Tempo or Speed. Not used for [MidiStreamPlayer](#).

### **int MidiPlayerTK.MPTKEvent.Value**

Contains a value between 0 and 127 in relation with the Command. For:

- Command = NoteOn then Value contains midi note
- Command = ControlChange then Value contains controller value
- Command = PatchChange then Value contains patch value

## **int MidiPlayerTK.MPTKEvent.Velocity**

Velocity between 0 and 127

## **List<fluid\_voice> MidiPlayerTK.MPTKEvent.Voices**

List of voices associated to this Event for playing a NoteOn event.

---

## **MidiPlayerTK.MPTKListItem**

A list of string with index: midi, preset, bank, drum, ...

### **Data Fields**

- int [Index](#)  
*Index in the list:*
- string [Label](#)  
*Label*

---

### **Detailed Description**

A list of string with index: midi, preset, bank, drum, ...

---

### **Field Documentation**

#### **int MidiPlayerTK.MPTKListItem.Index**

Index in the list:

- `patch` num if patch list,
- `bank` number if bank list,
- `index` in list for midi.

#### **string MidiPlayerTK.MPTKListItem.Label**

Label

## **MidiPlayerTK.TrackMidiEvent**

Midi event list (NAudio format)

---

### **Detailed Description**

Midi event list (NAudio format)

---

## **Index**

INDEX