

The latest document is on the GitHub.

If you have any questions or issues, please post an issue on GitHub.

<https://github.com/kshoji/Unity-MIDI-Plugin-supports>

---

This document explains how to install the plugin, and to use the plugin's features.

## Table of Contents

---

- [MIDI Plugin for Mobile and Desktop](#)
- [Available MIDI interfaces for platforms](#)
  - [About limitations](#)
- [How to install plugin](#)
- [About build PostProcessing](#)
  - [iOS](#)
  - [Android](#)
- [How to implement features](#)
  - [Initializing Plugin](#)
  - [Terminating Plugin](#)
  - [Work with RTP-MIDI \(experimental feature for non-iOS platforms\)](#)
  - [MIDI device attaching/detaching event handling](#)
  - [MIDI Event Receiving](#)
  - [MIDI Event Sending](#)
  - [Creating and start using a Sequencer](#)
  - [Read SMF as a Sequence, and play it](#)
  - [Record a sequence](#)
  - [Write the sequence to a SMF file](#)
  - [Android: Use CompanionDeviceManager to find BLE MIDI devices](#)
- [Tested devices](#)
- [Version History](#)
- [Contacts](#)
  - [About plugin author](#)
  - [Used Open Source Softwares created by me:](#)
  - [Used example MIDI data by others](#)

# MIDI Plugin for Mobile and Desktop

---

This plugin provides MIDI transceiving features to your mobile app(iOS, Android, Universal Windows Platform), desktop app(Windows, OSX, Linux), and WebGL app.

Currently implemented `MIDI 1.0 protocol` only.

## Available MIDI interfaces for platforms

---

The available MIDI interfaces for each platforms are listed below.

Platform	Bluetooth MIDI	USB MIDI	Network MIDI (RTP-MIDI)
iOS	○	○	○
Android	○	○	△(experimental)
Universal Windows Platform	-	○	△(experimental)
Standalone OSX, Unity Editor OSX	○	○	○
Standalone Linux, Unity Editor Linux	○	○	△(experimental)
Standalone Windows, Unity Editor Windows	-	○	△(experimental)
WebGL	○	○	-

## About limitations

### Android

- USB MIDI is supported API Level 12 (Android 3.1) or above.
- Bluetooth MIDI is supported API Level 18 (Android 4.3) or above.

### iOS / OSX

- Supported iOS version 11.0 or above.
- Bluetooth MIDI support is Central mode only.

### UWP

- Supported UWP platform version 10.0.10240.0 or above.
- Bluetooth MIDI is not supported.
- To use Network MIDI(RTP-MIDI) feature, enable `PrivateNetworkClientServer` at `Project Settings > Player > Capabilities` setting.

### Windows

- Bluetooth MIDI is not supported.

## WebGL

- Supporting MIDI devices are depend on the running OS/Browser environment.
- WebGL may not access to another server resources with UnityWebRequest, so put resource files(such as SMF) into `StreamingAssets`.
- You *should* modify `index.html` file of `WebGLTemplates` directory like below. This enables to access Unity's runtime from `unityInstance` variable.
  - Or, copy `MIDI/Samples/WebGLTemplates` files to `Assets/WebGLTemplates`, and select `Default-MIDI` or `Minimal-MIDI` template from `Project Settings > Player > Resolution and Presentation > WebGL Template` setting.
  - For more information, see [Unity official document of WebGL Templates](#).

Original:

```
script.onload = () => {  
  createUnityInstance(canvas, config, (progress) => {  
    progressBarFull.style.width = 100 * progress + "%";  
  }).then((unityInstance) => {
```

Modified: add global `unityInstance` variable.

```
var unityInstance = null; // <- HERE  
script.onload = () => {  
  createUnityInstance(canvas, config, (progress) => {  
    progressBarFull.style.width = 100 * progress + "%";  
  }).then((unityInst) => { // <- HERE  
    unityInstance = unityInst; // <- HERE
```

## Network MIDI

- The error correction(RTP MIDI Journaling) protocol is not supported on experimental support specified above.

# How to install plugin

---

1. Import the unitypackage from the Asset Store view.
2. Select the app's platform; iOS or Android. and build the sample app.
  - The sample scene is found at Assets/MIDI/Samples directory.

## About build PostProcessing

---

### PostProcessing: iOS

- Additional framework will be automatically added while building postprocess.
  - Additional frameworks: `CoreMIDI.framework`, `CoreAudioKit.framework`
- `Info.plist` will be automatically adjusted while building postprocess.
  - Additional property: `NSBluetoothAlwaysUsageDescription`

### PostProcessing: Android

- `AndroidManifest.xml` will be automatically adjusted while building postprocess.
  - Additional permissions: `android.permission.BLUETOOTH`,  
`android.permission.BLUETOOTH_ADMIN`, `android.permission.ACCESS_FINE_LOCATION`,  
`android.permission.BLUETOOTH_SCAN`, `android.permission.BLUETOOTH_CONNECT`,  
`android.permission.BLUETOOTH_ADVERTISE`.
  - Additional feature: `android.hardware.bluetooth_le`, `android.hardware.usb.host`
- If you want to use the USB MIDI feature on Oculus(Meta) Quest 2, please UNCOMMENT below to detect USB MIDI device connections.

The part of `PostProcessBuild.cs`

```
public class ModifyAndroidManifest : IPostGenerateGradleAndroidProject
{
    public void OnPostGenerateGradleAndroidProject(string basePath)
    {
        :

        // androidManifest.AddUsbIntentFilterForOculusDevices(); // UNCOMMENT
        THIS LINE FOR OCULUS QUEST 2
    }
}
```

# How to implement features

---

## Initializing Plugin

1. Call `MidiManager.Instance.InitializeMidi` method on `Awake` method in `MonoBehaviour`.
  - A GameObject named `MidiManager` will be created in `DontDestroyOnLoad` at the hierarchy view.

NOTE: If the EventSystem component already exists in another place, remove `gameObject.AddComponent()` method calling at `MidiManager.Instance.InitializeMidi` method.

2. (BLE MIDI only)
  - Call `MidiManager.Instance.StartScanBluetoothMidiDevices` method to scan BLE MIDI devices.
    - This method should be called in the `InitializeMidi` method's callback action.
3. (RTP-MIDI only)
  - Call `MidiManager.Instance.StartRtpMidi` method with session name and udp port number to start RTP-MIDI session accepting.

```
private void Awake()  
{  
    MidiManager.Instance.RegisterEventHandleObject(gameObject);  
    MidiManager.Instance.InitializeMidi(() =>  
    {  
        MidiManager.Instance.StartScanBluetoothMidiDevices(0);  
    }));  
}
```

**Fig.1 Awake method will be like above.**

## Terminating Plugin

1. Call `MidiManager.Instance.TerminateMidi` method on `OnDestroy` method in `MonoBehaviour`.
  - This method should be called on finishing Scene or finishing to use MIDI functions.
2. (RTP-MIDI only)
  - Call `MidiManager.Instance.StopRtpMidi` method to stop RTP-MIDI session communications.

```
private void OnDestroy()  
{  
    MidiManager.Instance.TerminateMidi();  
}
```

**Fig.2 MidiManager termination.**

## Work with RTP-MIDI (experimental feature for non-iOS platforms)

- Start RTP-MIDI session:
  - Call `MidiManager.Instance.StartRtpMidi` method with session name and udp port number to start RTP-MIDI session accepting.
  - This will start listening to the udp port with the specified port number. The another computer can connect with the app.
- Stop RTP-MIDI session:
  - Call `MidiManager.Instance.StopRtpMidi` method to stop RTP-MIDI session communications.
- Connect another RTP-MIDI running computer:
  - Call `MidiManager.Instance.ConnectToRtpMidiClient` method to start connection with another computer.

```
// Starts to listen UDP 5004 port with session name "RtpMidiSession".
MidiManager.Instance.StartRtpMidi("RtpMidiSession", 5004);

...
// Stop the session
MidiManager.Instance.StopRtpMidi(5004);

// Connect to another machine's RTP-MIDI session
MidiManager.Instance.ConnectToRtpMidiClient("RtpMidiSession", 5004, new
IPEndPoint(IPAddress.Parse("192.168.0.111"), 5004));
```

**Fig.3 using RTP-MIDI features**

## MIDI device attaching/detaching event handling

1. Call `MidiManager.Instance.RegisterEventHandleObject` method to register GameObject to receive event;
2. Implement interface `IMidiDeviceEventHandler` for event receiving.
  - `OnMidiInputDeviceAttached`, `OnMidiOutputDeviceAttached` will be called on a new MIDI device connected.
  - `OnMidiInputDeviceDetached`, `OnMidiOutputDeviceDetached` will be called on a MIDI device disconnected.

```
public void OnMidiInputDeviceAttached(string deviceId)
{
}

public void OnMidiOutputDeviceAttached(string deviceId)
{
    receivedMidiMessages.Add($"MIDI device attached. deviceId: {deviceId}, name: {MidiManager.Instance.GetDeviceName(deviceId)}");
}

public void OnMidiInputDeviceDetached(string deviceId)
{
}

public void OnMidiOutputDeviceDetached(string deviceId)
{
    receivedMidiMessages.Add($"MIDI device detached. deviceId: {deviceId}, name: {MidiManager.Instance.GetDeviceName(deviceId)}");
}
```

**Fig.4 device attach/detach events handler**

All codes are found at `Assets/MIDI/Samples/Scripts/MidiSampleScene.cs` file.

## MIDI Event Receiving

1. Implement event receiving interface written in `IMidiEventHandler.cs` source code, named like `IMidiXXXXXEventHandler`.
  - If you want to receive a Note On event, implement the `IMidiNoteOnEventHandler` interface.
2. Call `MidiManager.Instance.RegisterEventHandleObject` method to register GameObject to receive event;
3. On receiving a MIDI event, the implemented method will be called.

```
public class MidiSampleScene : MonoBehaviour, IMidiAllEventsHandler,
IMidiDeviceEventHandler
{
    private void Awake()
    {
        MidiManager.Instance.RegisterEventHandleObject(gameObject);
        ...
    }
}
```

**Fig.5 example of implementation IMidiAllEventHandler and calling the `RegisterEventHandleObject` method in `Awake`.**

```
public void OnMidiNoteOn(string deviceId, int group, int channel, int note, int
velocity)
{
    receivedMidiMessages.Add($"OnMidiNoteOn channel: {channel}, note: {note},
velocity: {velocity}");
}

public void OnMidiNoteOff(string deviceId, int group, int channel, int note, int
velocity)
{
    receivedMidiMessages.Add($"OnMidiNoteOff channel: {channel}, note: {note},
velocity: {velocity}");
}
```

**Fig.6 MIDI Note On/ Note Off event receiving handler**

**All codes are found at `Assets/MIDI/Samples/Scripts/MidiSampleScene.cs` file.**



## MIDI Event Sending

1. Call the `MidiManager.Instance.SendMidiXXXXXX` method somewhere. Like this:

```
MidiManager.Instance.SendMidiNoteOn("deviceId", 0/*groupId*/, 0/*channel*/,  
60/*note*/, 127/*velocity*/);
```

2. deviceId can be obtained from `MidiManager.Instance.DeviceIdSet` property (type: `HashSet<string>`).

```
if (GUILayout.Button("NoteOn"))  
{  
    MidiManager.Instance.SendMidiNoteOn(deviceIds[deviceIdIndex], 0, (int)channel,  
(int)noteNumber, (int)velocity);  
}
```

**Fig.7 Sending MIDI Note On message**

All codes are found at `Assets/MIDI/Samples/Scripts/MidiSampleScene.cs` file.

## Creating and start using a Sequencer

```
var isSequencerOpened = false;  
var sequencer = new SequencerImpl(() => { isSequencerOpened = true; });  
sequencer.Open();
```

**Fig.8 Creating SequencerImpl instance and Open it.**

All codes are found at `Assets/MIDI/Samples/Scripts/MidiSampleScene.cs` file.

## Read SMF as a Sequence, and play it

```
sequencer.UpdateDeviceConnections();  
  
using var stream = new FileStream(smfPath, FileMode.Open, FileAccess.Read);  
sequencer.SetSequence(stream);  
sequencer.Start();  
  
...  
  
sequencer.Stop();
```

**Fig.9 Read a SMF and play it.**

## Record a sequence

```
sequencer.UpdateDeviceConnections();

sequencer.SetSequence(new Sequence(Sequence.Ppq, 480));
sequencer.StartRecording();

...

sequencer.Stop();
```

**Fig.10 Set a new Sequence to record, and start recording MIDI data**

## Write the sequence to a SMF file

```
var sequence = sequencer.GetSequence();
if (sequence.GetTickLength() > 0)
{
    using var stream = new FileStream(recordedSmfPath, FileMode.Create,
    FileAccess.Write);
    MidiSystem.WriteSequence(sequence, stream);
}
```

**Fig.11 Write a SMF from recorded Sequence**

## Android: Use CompanionDeviceManager to find BLE MIDI devices

You can use [CompanionDeviceManager](#) for BLE MIDI device connection on Android.

To enable this feature, add `FEATURE_ANDROID_COMPANION_DEVICE` to the `Scripting Define Symbols` setting.

```
Project Settings > Other Settings > Script Compilation > Scripting Define Symbols
```

# Tested devices

---

- Android: Pixel 4a, Oculus Quest2
- iOS: iPod touch 7th gen
- UWP/Standalone Windows/Unity Editor Windows: Surface Go 2
- Standalone OSX/Unity Editor OSX: Mac mini 3,1
- Standalone Linux/Unity Editor Linux: Ubuntu 20.04 on VirtualBox
- MIDI devices:
  - Quicco mi.1 (BLE MIDI)
  - Miselu C.24 (BLE MIDI)
  - TAHORNG Elefue (BLE MIDI)
  - Roland UM-ONE (USB MIDI)
    - NOTE: This device didn't work with iOS.
  - Gakken NSX-39 (USB-MIDI)
  - MacOS Audio MIDI Setup (RTP-MIDI)

# Version History

---

- v1.0 Initial release
- v1.1 Update release
  - Add MIDI sequencer(playing / recording MIDI sequence) feature
  - Add SMF reading / writing feature
  - Add BLE MIDI Peripheral feature on Android
  - Fix USB MIDI receiving issues on Android
  - Fix BLE MIDI sending issues on Android / iOS
  - Fix BLE MIDI receiving issue(NoteOn with velocity = 0) on Android
- v1.2.0 Update release
  - Add experimental RTP-MIDI support for Android, or other platforms.
  - Add USB MIDI support for Universal Windows Platform(UWP).
  - Add Android 12's new Bluetooth permissions support.
  - Fix MIDI tranceiving performance improvement on iOS, Android.
  - Fix EventSystem duplication error when the sample scene appended multiple times.
  - Fix Android BLE MIDI's issue around fixed timestamp.
- v1.2.1 Bugfix release
  - Fix sequencer thread remains after closing
  - Fix Android ProgramChange message failure
  - Fix System exclusive logging issue
  - Fix ThreadInterruptedException issue on UWP
  - Fix SMF reading/writing issues around System exclusive
  - Some performance improvements
- v1.3.0 Update release
  - Add platform support for Standalone OSX, Windows, Linux
  - Add platform support for WebGL
  - Add support for Unity Editor OSX, Windows, Linux
  - Changed Sequencer implementation from Thread to Coroutine
  - Fix iOS/OSX device attaching/detaching issue
- v1.3.1 Bugfix release
  - [Issue connecting to Quest 2 via cable](#)
  - [Sample scene stops working.](#)
  - [Byte is obsolete on android](#)
  - [Any way of negotiating MTU?](#)
  - [Can't get it to work on iOS](#)
  - [Have errors with sample scene](#)
  - Android permissions requesting issue
  - Add: Android CompanionDeviceManager support
- v1.3.2 Bugfix release
  - Fixed Android compile error
  - Fixed MIDI event order while playing SMF sequence

# Contacts

---

## Report issues on GitHub

- GitHub supports repository: <https://github.com/kshoji/Unity-MIDI-Plugin-supports>
  - Search and report issues: <https://github.com/kshoji/Unity-MIDI-Plugin-supports/issues>

## About plugin author

- Kaoru Shoji : [0x0badc0de@gmail.com](mailto:0x0badc0de@gmail.com)
- github: <https://github.com/kshoji>

## Used Open Source Softwares created by me:

- Android Bluetooth MIDI library: <https://github.com/kshoji/BLE-MIDI-for-Android>
- Android USB MIDI library: <https://github.com/kshoji/USB-MIDI-Driver>
- iOS MIDI library: <https://github.com/kshoji/Unity-MIDI-Plugin-iOS>
- MidiSystem for .NET(sequencer, SMF importer/exporter): <https://github.com/kshoji/MidiSystem-for-.NET>
- RTP-MIDI for .NET: <https://github.com/kshoji/RTP-MIDI-for-.NET>
- Unity MIDI Plugin UWP: <https://github.com/kshoji/Unity-MIDI-Plugin-UWP>
- Unity MIDI Plugin Linux: <https://github.com/kshoji/Unity-MIDI-Plugin-Linux>
- Unity MIDI Plugin OSX: <https://github.com/kshoji/Unity-MIDI-Plugin-OSX>

## Used example MIDI data by others

specified as UnityWebRequest's URL source. The SMF binary file is not included.

- Prelude and Fugue in C minor BWV 847 Music by J.S. Bach
  - The MIDI, audio(MP3, OGG) and video files of Bernd Krueger are licensed under the cc-by-sa Germany License.
  - This means, that you can use and adapt the files, as long as you attribute to the copyright holder
  - Name: Bernd Krueger
  - Source: <http://www.piano-midi.de>
  - The distribution or public playback of the files is only allowed under identical license conditions.