

最新的文档在 GitHub 上。

如果有任何问题或问题，请在 GitHub 上发布问题。

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

本文档解释了如何安装 插件，以及如何使用插件的功能。

这是机器翻译的文件。更多 详细信息，请参 阅原始英文文档。

目录

- [移动和桌面的 MIDI 插件](#)
- [适用于平台的 MIDI 接口](#)
 - [关于限制](#)
- [安装插件](#)
- [关于构建 PostProcessing](#)
 - [iOS](#)
 - [Android](#)
- [实现功能](#)
 - [初始化插件](#)
 - [终止插件](#)
 - [使用 RTP-MIDI \(非 iOS 平台的实验功能\)](#)
 - [MIDI 使用 RTP-MIDI 特性](#)
 - [MIDI 信号接收](#)
 - [MIDI 信号发送](#)
 - [创建并开始使用 Sequence](#)
 - [将 SMF 读取为 Sequence，并播放它](#)
 - [记录一个 Sequence](#)
 - [将序列写入 SMF 文件](#)
 - [Android: 使用 CompanionDeviceManager 查找 BLE MIDI 设备](#)
- [测试设备](#)
- [版本历史](#)
- [联系](#)
 - [在 GitHub 上报告问题](#)
 - [关于插件作者](#)
 - [使用的开源软件由我创建](#)
 - [其他人使用的示例 MIDI 数据](#)

移动和桌面的 MIDI 插件

此插件为您的移动应用程序（iOS、Android、通用 Windows 平台）、桌面应用程序（Windows、OSX、Linux）和 WebGL 应用程序提供 MIDI 收发功能。
当前仅实现了“MIDI 1.0 协议”。

适用于平台的 MIDI 接口

下面列出了每个平台可用的 MIDI 接口。

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

关于限制

Android

- API 级别 12 (Android 3.1) 或更高版本支持 USB MIDI。
- API 级别 18 (Android 4.3) 或更高版本支持蓝牙 MIDI。

iOS / OSX

- 支持 iOS 11.0 或更高版本。
- Bluetooth MIDI 支持仅限于 Central 模式。

UWP

- 支持的 UWP 平台版本 10.0.10240.0 或以上。
- 不支持 Bluetooth MIDI。
- 要使用网络 MIDI(RTP-MIDI) 功能，请在 `Project Settings > Player > Capabilities` 设置中启用 `PrivateNetworkClientServer`。

Windows

- 不支持 Bluetooth MIDI。

WebGL

- 支持的 MIDI 设备取决于运行的操作系统/浏览器环境。
- WebGL 可能无法通过 UnityWebRequest 访问其他服务器资源，因此请将资源文件（例如 SMF）放入 `StreamingAssets`。
- 您 应该修改 `WebGLTemplates` 目录的 `index.html` 文件，如下所示。这可以从 `unityInstance` 变量访问 Unity 的运行时。
 - 或者，将 `MIDI/Samples/WebGLTemplates` 文件复制到 `Assets/WebGLTemplates`，然后从 `Project Settings > Player > Resolution and Presentation > WebGL Template` 设置中选择 `Default-MIDI` 或 `Minimal-MIDI` 模板。
 - 有关更多信息，请参阅 [WebGL 模板的 Unity 官方文档](#)。

原文:

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

修改的: 添加 `unityInstance` 全局变量。

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

Network MIDI

- 上面指定的实验支持不支持纠错（RTP MIDI Journaling）协议。

安装插件

1. 从 Asset Store 视图导入 unitypackage。
2. 选择应用的平台；iOS 或安卓。并构建示例应用程序。
 - 示例场景位于 Assets/MIDI/Samples 目录中。

关于构建 PostProcessing

PostProcessing: iOS

- 在构建后期处理时会自动添加额外的框架。
 - 附加框架：`CoreMIDI.framework`、`CoreAudioKit`
- `Info.plist` 将在构建后期处理时自动调整。
 - 附加属性：`NSBluetoothAlwaysUsageDescription`

PostProcessing: Android

- `AndroidManifest.xml` 将在构建后处理时自动调整。
 - 附加权限：`android.permission.BLUETOOTH`, `android.permission.BLUETOOTH_ADMIN`, `android.permission.ACCESS_FINE_LOCATION`, `android.permission.BLUETOOTH_SCAN`, `android.permission.BLUETOOTH_CONNECT`, `android.permission.BLUETOOTH_ADVERTISE`.
 - 附加功能：`android.hardware.bluetooth_le`, `android.hardware.usb.host`
- 如果您想在 Oculus(Meta) Quest 2 上使用 USB MIDI 功能，请在下方取消注释以检测 USB MIDI 设备连接。

`PostProcessBuild.cs` 的部分

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

        // androidManifest.AddUsbIntentFilterForOculusDevices(); // 为 Oculus
        Quest 2 取消注释此行
```

实现功能

初始化插件

1. 调用 `MidiManager.Instance.InitializeMidi` 方法 `Awake` 在 `MonoBehaviour` 中
 - 一个名为 `MidiManager` 的游戏对象将 `DontDestroyOnLoad` 在层次视图

注意：如果 `EventSystem` 组件已经存在于另一个代码中，请删除在 `MidiManager.Instance.InitializeMidi` 方法中调用的 `gameObject.AddComponent()` 方法。
2. (仅限BLE MIDI)
 - 调用 `MidiManager.Instance.StartScanBluetoothMidiDevices` 方法来扫描 BLE MIDI 设备。
 - 此方法应在 `InitializeMidi` 方法的回调操作中调用。
 - 调用 `MidiManager.Instance.StartRtpMidi` 方法 会话名称 和 udp 端口号 以启动 RTP-MIDI 会话接受。

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

图 1 唤醒方法将如上所示。

终止插件

1. 调用 `MidiManager.Instance.TerminateMidi` 方法 `OnDestroy` 在 `MonoBehaviour` 中
 - 在完成场景或完成使用 MIDI 功能时应调用此方法。
2. (仅限RTP-MIDI)
 - 调用 `MidiManager.Instance.StopRtpMidi` 方法停止 RTP-MIDI 会话通信。

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

图 2 MidiManager termination.

使用 RTP-MIDI（非 iOS 平台的实验功能）

- 启动 RTP-MIDI 会话:
 - 调用 `MidiManager.Instance.StartRtpMidi` 方法 会话名称 和 udp 端口号 以启动 RTP-MIDI 会话接受。
 - 这将开始侦听具有指定端口号的 udp 端口。另一台计算机可以与该应用程序连接。
- 停止 RTP-MIDI 会话:
 - 调用 `MidiManager.Instance.StopRtpMidi` 方法停止 RTP-MIDI 会话通信。
- Connect another RTP-MIDI running computer:
 - 调用 `MidiManager.Instance.ConnectToRtpMidiClient` 方法开始连接另一台电脑。

```
// 开始侦听会话名称为 "RtpMidiSession" 的 UDP 5004 端口。
MidiManager.Instance.StartRtpMidi("RtpMidiSession", 5004);
...
// 停止会话
MidiManager.Instance.StopRtpMidi(5004);

// 连接到另一台机器的 RTP-MIDI 会话
MidiManager.Instance.ConnectToRtpMidiClient("RtpMidiSession", 5004, new
IPEndPoint(IPAddress.Parse("192.168.0.111"), 5004));
```

图 3 使用 RTP-MIDI 特性

MIDI 使用 RTP-MIDI 特性

1. 调用 `MidiManager.Instance.RegisterEventHandleObject` 方法注册 GameObject 以接收信号;
2. 实现接口 `IMidiDeviceEventHandler` 用于信号接收。
 - `OnMidiInputDeviceAttached`, `OnMidiOutputDeviceAttached` 将在连接的新 MIDI 设备上调用。
 - `OnMidiInputDeviceDetached`, `OnMidiOutputDeviceDetached` 将在 MIDI 设备断开连接时调用。

```
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)}");
}
```

图 4 设备附加/分离信号处理程序

所有代码都在 `Assets/MIDI/Samples/Scripts/MidiSampleScene.cs` 中。

MIDI 信号接收

1. 实现信号接收接口, 用 `IMidiEventHandler.cs` 源码编写, 命名为 `IMidiXXXXXEventHandler`。
 - 如果要接收 Note On 信号, 请实现 `IMidiNoteOnEventHandler` 接口。
2. 调用 `MidiManager.Instance.RegisterEventHandleObject` 方法注册 GameObject 接收信号;
3. 收到 MIDI 信号后, 将调用已实现的方法。

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

图 5 实现 `IMidiAllEventHandler` 并在 `Awake` 中调用 `RegisterEventHandleObject` 方法的示例。

```
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}");
}
```

图 6 MIDI Note On/ Note Off 信号接收处理程序

所有代码都在 `Assets/MIDI/Samples/Scripts/MidiSampleScene.cs` 中。

MIDI 信号发送

1. 调用 `MidiManager.Instance.SendMidiXXXXXX` 方法。
像这样:

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

2. deviceId 可以从 `MidiManager.Instance.DeviceIdSet` 属性（类型：HashSet<string>）中获取。

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

图 7 发送 MIDI Note On

所有代码都在 `Assets/MIDI/Samples/Scripts/MidiSampleScene.cs` 中。

创建并开始使用 Sequence

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

图 8 创建 SequencerImpl 实例并打开它。

所有代码都在 `Assets/MIDI/Samples/Scripts/MidiSampleScene.cs` 中。

将 SMF 读取为 Sequence，并播放它

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

图 9 读取 SMF 并播放。

记录一个 Sequence

```
sequencer.UpdateDeviceConnections();

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

...

sequencer.Stop();
```

图 10 设置新的 Sequence 进行录制，开始录制 MIDI 数据

将序列写入 SMF 文件

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

图 11 写入 SMF

Android: 使用 CompanionDeviceManager 查找 BLE MIDI 设备

您可以使用 [CompanionDeviceManager](#) 在 Android 上连接 BLE MIDI 设备。

要启用此功能，请将 `FEATURE_ANDROID_COMPANION_DEVICE` 添加到 `Scripting Define Symbols` 设置。

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

测试设备

- 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 设备:
 - Quicco mi.1 (BLE MIDI)
 - Miselu C.24 (BLE MIDI)
 - TAHORNG Elefue (BLE MIDI)
 - Roland UM-ONE (USB MIDI)
 - 注意：此设备不适用于 iOS。
 - Gakken NSX-39 (USB-MIDI)
 - MacOS Audio MIDI Setup (RTP-MIDI)

版本历史

- v1.0 初始版本
- v1.1 更新版本
 - 添加 MIDI 音序器（播放/录制 MIDI 序列）功能
 - 添加 SMF 读/写功能
 - 添加 BLE MIDI 外设功能在 Android 上
 - 修复 USB MIDI 接收问题 在 Android 上
 - 修复 BLE MIDI 在 Android / iOS 上发送问题 在
 - 上修复 BLE MIDI 接收问题（NoteOn with velocity = 0）
- v1.2.0 更新版本
 - 添加对 Android 或其他平台的实验性 RTP-MIDI 支持。
 - 在通用 Windows 平台 (UWP) 上添加 USB MIDI 支持。
 - 添加 Android 12 的新蓝牙权限支持。
 - 修复 iOS、Android 上的 MIDI 收发性能改进。
 - 修复示例场景多次附加时的 EventSystem 重复错误。
 - 修复 Android BLE MIDI 关于固定时间戳的问题。
- v1.2.1 修正版本
 - 修复排序器线程在关闭后仍然存在
 - 修复 Android ProgramChange 消息失败
 - 修复 System Exclusive 日志记录问题
 - 修复 UWP 上的 ThreadInterruptedException 问题
 - 修复围绕 System Exclusive 的 SMF 读/写问题
 - 一些性能改进
- v1.3.0 更新版本
 - 添加对 Standalone OSX、Windows、Linux 的平台支持
 - 添加对 WebGL 的平台支持
 - 添加对 Unity Editor OSX、Windows、Linux 的支持
 - 将 Sequencer 实现从 Thread 更改为 Coroutine
 - 修复 iOS/OSX 设备附加/分离问题
- v1.3.1 修正版本
 - [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 权限请求问题
 - 添加对 Android CompanionDeviceManager 支持
- v1.3.2 修正版本
 - 修复 Android 上的编译错误
 - 修复 播放 SMF 时 MIDI 事件的顺序
- v1.3.3 修正版本
 - 修复 WebGL 上 MIDI 发送失败
- v1.3.4 修正版本

- 修复：与之前连接的设备ID相同但设备名称不同的设备时，获取到错误的设备名称。
- iOS: 将“完成”按钮添加到 BLE MIDI 搜索弹出框
- Sample scene: BLE MIDI 扫描功能仅限 Android/iOS
- MidiManager 单例模式细化

联系

在 GitHub 上报告问题

- GitHub 支持仓库: <https://github.com/kshoji/Unity-MIDI-Plugin-supports>
 - 搜索和报告问题: <https://github.com/kshoji/Unity-MIDI-Plugin-supports/issues>

关于插件作者

- Kaoru Shoji/庄司 薫 : 0x0badc0de@gmail.com
- github: <https://github.com/kshoji>

使用的开源软件由我创建

- 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>

其他人使用的示例 MIDI 数据

指定为 UnityWebRequest 的 URL 源。不包括 SMF 二进制文件。

- 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.