应用间 MIDI — 跨平台说明 (Android, iOS/macOS, Linux)

本文档介绍了本 Unity MIDI 插件中“**应用间 MIDI (Inter-App MIDI)**”在各平台上的工作方式。

由于“应用间 MIDI”在不同操作系统上并没有一个通用的标准名称，本文档在**实践意义**上使用该术语：

> **应用间 MIDI = 在同一台机器/设备上的不同应用程序之间发送/接收 MIDI 数据**，通常通过操作系统提供的虚拟终端/端口或系统服务来实现。

1) 各平台快速总结

Android（明确的应用间 MIDI 集成）

- 通过 Android AAR 实现，并由 Unity 通过 Android 插件封装类进行访问。
- 位置：
  - `Assets/MIDI/Plugins/Android/inter-app-midi-0.0.5.aar`
  - `Assets/MIDI/Scripts/MidiPlugin.Android.cs` (MIDI 1.0)
  - `Assets/MIDI/Scripts/Midi2Plugin.Android.cs` (MIDI 2.0 / UMP)
- 除了 USB MIDI 和 BLE MIDI 外，您还会获得一个“应用间 (Inter-App)”传输层。

iOS / macOS (CoreMIDI：通过虚拟终端进行应用间路由)

- 在苹果平台上，应用间 MIDI 通常指 **CoreMIDI 终端**（应用可以暴露虚拟 MIDI 目标/源，操作系统在它们之间路由消息）。
- 在本项目中，苹果平台的支持实现在：
  - `Assets/MIDI/Scripts/MidiPlugin.Apple.cs` (MIDI 1.0)
  - `Assets/MIDI/Plugins/iOS/libMIDIPlugin.a` 以及 `Assets/MIDI/Plugins/macOS/MIDIPlugin.bundle`
- 从 Unity 的角度来看，您仍然使用 `MidiManager` 和设备 ID；如果另一个应用暴露了 CoreMIDI 端口，它会显示为一个设备，并以“应用间 MIDI”的方式工作。

Linux (ALSA/JACK：通过 MIDI 端口进行应用间路由)

- 在 Linux 上，应用间 MIDI 路由通常通过 **ALSA MIDI 端口** 和/或 **JACK MIDI** 实现（取决于原生插件的构建/配置方式）。
- 在本项目中，Linux 的支持实现在：
  - `Assets/MIDI/Scripts/MidiPlugin.Linux.cs`
  - `Assets/MIDI/Plugins/Linux/MIDIPlugin.so`
- 同样，Unity 的用法是一致的：`MidiManager` 向对应于操作系统/协议栈暴露端口的设备 ID 发送/接收数据。

2) 在插件架构中的位置

MIDI 1.0

- 您的应用通过 `MidiManager` 进行以下操作：
  - 初始化（`InitializeMidi`）
  - 设备枚举（通过 `DeviceIdSet`，`InputDeviceIdSet`，`OutputDeviceIdSet`）
  - 事件接收（Unity EventSystem 处理器接口）
  - 发送 MIDI 1.0 消息（Note On/Off, CC, SysEx 等）

iOS/macOS/Linux 上的应用间 MIDI 实际上只是操作系统提供的"另一组 MIDI 终端"，并通过平台插件显现出来。

MIDI 2.0 / UMP

- Android 拥有针对 MIDI 2.0 的明确应用间 MIDI 路径（`Midi2Plugin.Android.cs`）。
- 对于 iOS/macOS/Linux，MIDI 2.0 的支持取决于平台插件和操作系统协议栈所暴露的内容。在本仓库中，MIDI 2.0 的插件实现与 MIDI 1.0 的 Apple/Linux 插件是分开的，因此除非该平台的 MIDI 2.0 插件后端提供支持，否则"应用间 MIDI 2.0"可能无法以与 Android 相同的方式使用。

实践要点：

- **iOS/macOS/Linux 上的应用间 MIDI 1.0 通过其平台 MIDI 后端得到明确支持。**
- **Android 上明确集成了应用间 MIDI 2.0 (UMP)**；其他平台上的可用性取决于您所使用的 MIDI 2.0 后端实现。

3) 用法（各平台通用的 API）

接收应用间 MIDI (MIDI 1.0)

1. 实现一个或多个 MIDI 1.0 处理器接口（例如 `IMidiNoteOnEventHandler`）。
2. 将处理器对象注册到 `MidiManager`。
3. 初始化 MIDI (`MidiManager.InitializeMidi(...)`)。
4. 使用操作系统/应用的路由 UI 或工具，将另一个应用连接到您的应用暴露的终端（或将您的应用连接到另一个应用的终端）。
5. 消息到达时，您的处理器将触发。

发送应用间 MIDI (MIDI 1.0)

- 使用 `MidiManager.Instance.SendMidiNoteOn(...)` / `SendMidiControlChange(...)` / `SendMidiSystemExclusive(...)` 等方法。
- 选择对应于目标应用/端口的正确 `deviceId`。

4) 各平台“如何连接应用？”（概念性）

> 本节内容主要是概念性的，因为路由 UI 和工具随操作系统版本及第三方应用而异。

Android

- 应用间 MIDI 路由由 Android Inter-App MIDI 服务/插件集成处理。
- 设备 ID 对应于通过该层暴露的终端。

iOS / macOS

- 其他应用会暴露 CoreMIDI 端口（虚拟源/目标）。
- 您通常使用以下方式连接应用：
  - 应用自带的 MIDI 设置 UI，和/或
  - 操作系统级别的 MIDI 路由工具（因设备和工作流而异）。
- 连接后，您的 Unity 应用会将这些终端视为 MIDI 设备。

Linux

- 应用通过 ALSA/JACK 暴露端口。
- 路由通常使用外部配线架 (Patchbay) 或连接工具完成。
- 路由完成后，您的 Unity 应用即可通过对应的端口进行发送/接收。

5) 注意事项、限制与故障排除

“应用间 MIDI”在各平台命名不同

- Android：本仓库中明确的“Inter-App MIDI”插件/服务。
- iOS/macOS：CoreMIDI 终端（设计上即支持应用间）。
- Linux：ALSA/JACK 端口（设计上即支持应用间）。

设备 ID 在各平台可能不同

设备标识符由平台/传输层定义。请勿假设所有操作系统上的 `deviceId` 格式都相同。

重复发送 (Android)

在 Android 上，MIDI 1.0 后端可能会通过多个传输层（USB/BLE/Inter-App，取决于初始化了哪些）进行发送。如果您看到重复的消息，请检查您发送的目标终端，以及是否对同一个逻辑设备激活了多个传输层。

MIDI 2.0 应用间预期

如果您的目标是实现**应用间的 MIDI 2.0 UMP**，请确认：

- 您目标平台的 MIDI 2.0 后端支持该功能，并且
- 对端应用在该平台上确实支持 MIDI 2.0/UMP。

6) 相关文档

- MIDI 1.0 (MidiManager)
- MIDI 2.0 / UMP (Midi2Manager)
- 传输协议与平台

源代码示例实现（应用间 MIDI “实践”示例）

应用间 MIDI 的核心在于**路由**；Unity 端的代码非常简单，即“枚举设备、挑选正确的 deviceId、发送/接收”。

使用 MIDI 1.0 快速入门组件：

- `Assets/MIDI/Samples/Scripts/DocumentationExamples/Midi1QuickStartExample.cs`

如何验证应用间路由：

1. 附加脚本并启动播放模式。
2. 使用您的操作系统 / MIDI 应用将虚拟源/目标路由到 Unity 应用（或另一个应用）。
3. 查看 Unity 日志：

- 设备连接 (Attach) 事件表明终端可见。
- Note On 日志确认收到应用间流量。

4. 发送测试音符并确认目标应用已接收。