# 传输协议与平台说明

本页面解释了在 `Assets/MIDI` 中 MIDI 数据是如何跨平台和跨网络协议进行传输的。

有关各平台的功能矩阵以及操作系统/版本要求，请参阅：

- 平台与限制

## 管理器如何选择传输协议（重要）

在大多数平台上，`MidiManager` / `Midi2Manager` 会**并行初始化多个后端**（例如：平台原生 MIDI + RTP-MIDI，如果启用还包括 Nearby）。

实践中的影响：

- **发送可能会扇出**：单个 `SendMidi...()` 调用可能会被转发到支持该功能的每个已初始化后端。
- **目标选择至关重要**：务必向您打算触达的特定 `deviceId` 发送消息。如果同一个逻辑终端出现在多个后端中，请多加留意。
- **设备连接/断开事件是聚合的**：来自所有活动后端的设备 ID 都会汇总到管理器的设备集合中。

这是设计使然：所有传输协议都被视为共享同一个 API 表面的“提供者 (Providers)”。

## 平台原生 MIDI (Plugins/ + MidiPlugin.*.cs)

大多数平台使用原生插件（或 WebGL JS 插件）加上实现 `IMidiPlugin` 的 C# 封装类。

位置：

- 原生二进制文件：`Assets/MIDI/Plugins/<Platform>/...`
- C# 封装类：`Assets/MIDI/Scripts/MidiPlugin.<Platform>.cs`

### Windows

- 封装类：`MidiPlugin.Windows.cs`
- 辅助工具：`Assets/MIDI/Scripts/win32/` (P/Invoke 和端口抽象)

### macOS / iOS / Android / Linux

- 每个平台都有对应的封装类，并调用各自的原生库。

### Android 设备（包括 Meta Quest）

基于 Android 的头戴式设备（如 Meta Quest）使用相同的 Android 传输层，但有一些实际的设置项：

- **Meta Quest 上的 USB MIDI** 可能需要在构建后处理期间启用 USB 设备 Intent 过滤器。
- **Meta Quest 上的蓝牙 MIDI 设备配对** 通常通过 Android 的伴生设备 (Companion Device) 工作流完成（可选功能标志）。

参见：

- 构建后处理与脚本定义符号 (Meta Quest USB + `FEATURE_ANDROID_COMPANION_DEVICE`)
- 平台与限制 (Android API 级别说明)

### WebGL

- 封装类：`MidiPlugin.WebGL.cs`
- JS 库：`Assets/MIDI/Plugins/WebGL/*.jslib`

注意（重要）：

- 浏览器权限和 WebMIDI 支持情况会影响功能的可用性。
- WebGL 无法使用原始 UDP/TCP 套接字，因此 **RTP-MIDI** 和 **UDP MIDI 2.0** 不可用。
- WebGL 目前无法处理 USB MIDI 2.0 设备及网络 MIDI 2.0；MIDI 2.0 实际上仅限于文件工作流（例如 MIDI 剪辑读写）。
- 受浏览器/CORS 规则限制，WebGL 可能无法通过 `UnityWebRequest` 访问其他服务器的资源；请将 `.mid` 等文件放入 `StreamingAssets`。

**WebGL 模板要求：暴露 `unityInstance`**

某些 WebGL 流程要求从 JS 可以访问到名为 `unityInstance` 的 Unity 运行时实例。

您应当修改 WebGL 模板的 `index.html`，将创建的实例赋值给全局变量：

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

或者：

- 将 `Assets/MIDI/Samples/WebGLTemplates` 复制到 `Assets/WebGLTemplates`。
- 然后在以下位置选择 `Default-MIDI` 或 `Minimal-MIDI`：`Project Settings > Player > Resolution and Presentation > WebGL Template`

（参考 Unity 手册：WebGL 模板。）

# RTP-MIDI (网络 MIDI 1.0)

RTP-MIDI（苹果网络 MIDI 风格）通过纯 .NET 实现提供支持：

- 模块：`Assets/MIDI/Scripts/RTP-MIDI-for-.NET/`
- 适配器：`Assets/MIDI/Scripts/MidiPlugin.RtpMidi.cs` (实现 `IMidiPlugin`)

主要能力：

- 启动一个或多个监听 UDP 端口的 RTP-MIDI 服务器。
- 连接/断开远程终端。
- 接收并发送 MIDI 1.0 消息。

限制：

- 不支持纠错协议 (**RTP MIDI Journaling**)。

线程处理：

- 传入的 RTP-MIDI 事件会通过主线程安全的分发机制转发给管理器。

# UDP MIDI 2.0 (网络 UMP)

UDP MIDI 2.0 支持由以下提供：

- `Assets/MIDI/Scripts/Midi2Plugin.Udp.cs`
- 发现助手/依赖项：`Assets/MIDI/Scripts/UdpMidi2Discovery/`

暴露的能力（通过 `Midi2Manager`）：

- 运行 UDP MIDI 2.0 服务器。
- 在局域网内发现服务器。
- 连接/断开客户端。
- 发送/接收 UMP 消息。
- 处理会话级命令（Ping/重传/重置/NAK/Bye）。

另请参阅：

- MIDI 2.0 / UMP (Midi2Manager)

# “Nearby” 传输协议 (Google Nearby Connections)

包含一个 `nearby/` 模块及对应的插件封装：

- `Assets/MIDI/Scripts/nearby/`
- `Assets/MIDI/Scripts/MidiPlugin.Nearby.cs`

此传输协议是同一个 MIDI 1.0 API 下的另一种 MIDI 链路层。

设置说明：

- 需要添加 Nearby 依赖包（参见 构建后处理与脚本定义符号）。
- Android 上有 API 级别限制（参见 平台与限制）。

运行时说明：

- 启用后，Nearby 后端需要 Unity 播放循环 (Player loop) 的定期更新。 当传输协议被编译进项目时，项目会将此逻辑连接到管理器中。

# 源代码示例实现 (RTP-MIDI 传输)

请参考：

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

将其附加到 GameObject 上并进入播放模式。它会启动一个 RTP-MIDI 服务器。
然后使用您的操作系统/设备的 RTP-MIDI 路由工具连接到它（具体 UI 取决于平台）。