アプリ間 MIDI — クロスプラットフォームの注意点 (Android, iOS/macOS, Linux)

このドキュメントでは、この Unity MIDI プラグインにおいて各プラットフォームで「**アプリ間 MIDI (Inter-App MIDI)**」がどのように動作するかを説明します。

「アプリ間 MIDI」という名称は OS ごとに統一された標準名ではないため、このドキュメントでは以下の**実用的**な意味で使用します:

> **アプリ間 MIDI = 同じマシン/デバイス上のアプリケーション間での MIDI 送受信**。通常、OS が提供する仮想エンドポイント/ポート、または OS のサービスを介して行われます。

1) プラットフォーム別の概要

Android (明示的なアプリ間 MIDI 統合)

- Android AAR を介して実装され、Android プラグインラッパーを通じて Unity からアクセスされます。
- 場所:
  - `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)
- USB MIDI や BLE MIDI に加えて、「Inter-App」トランスポートが利用可能です。

iOS / macOS (CoreMIDI: 仮想エンドポイントによるルーティング)

- Apple プラットフォームでは、アプリ間 MIDI は通常 **CoreMIDI エンドポイント** を指します（アプリが仮想 MIDI 出力先/入力元を公開し、OS がメッセージをルーティングします）。
- このプロジェクトでは、Apple プラットフォームのサポートは以下で実装されています:
  - `Assets/MIDI/Scripts/MidiPlugin.Apple.cs` (MIDI 1.0)
  - `Assets/MIDI/Plugins/iOS/libMIDIPlugin.a` および `Assets/MIDI/Plugins/macOS/MIDIPlugin.bundle`
- Unity 側からは、引き続き `MidiManager` とデバイス ID を使用します。他のアプリが CoreMIDI ポートを公開している場合、それがデバイスとして表示され、「アプリ間 MIDI」として機能します。

Linux (ALSA/JACK: MIDI ポートによるルーティング)

- Linux では、アプリ間 MIDI ルーティングは通常 **ALSA MIDI ポート** や **JACK MIDI**（ネイティブプラグインのビルド/設定に依存）を介して行われます。
- このプロジェクトでは、Linux サポートは以下で実装されています:
  - `Assets/MIDI/Scripts/MidiPlugin.Linux.cs`
  - `Assets/MIDI/Plugins/Linux/MIDIPlugin.so`
- Unity での使用方法は同じです。OS/スタックによって公開されたポートに対応するデバイス ID に対して `MidiManager` で送受信を行います。

2) プラグインアーキテクチャへの適合

MIDI 1.0

- アプリでは `MidiManager` を以下の用途で使用します:
  - 初期化 (`InitializeMidi`)
  - デバイスの列挙 (`DeviceIdSet`, `InputDeviceIdSet`, `OutputDeviceIdSet`)
  - イベント受信 (Unity EventSystem ハンドラインターフェース)
  - MIDI 1.0 メッセージの送信 (Note On/Off, CC, SysEx など)

iOS/macOS/Linux におけるアプリ間 MIDI は、実質的に「プラットフォームプラグインを通じて公開される、OS 提供の別の MIDI エンドポイントセット」にすぎません。

MIDI 2.0 / UMP

- Android には MIDI 2.0 用の明示的なアプリ間 MIDI パスがあります (`Midi2Plugin.Android.cs`)。
- iOS/macOS/Linux の場合、MIDI 2.0 サポートはプラットフォームプラグインと OS スタックが何を公開しているかに依存します。このリポジトリでは MIDI 2.0 プラグインの実装は MIDI 1.0 用の Apple/Linux プラグインとは分かれているため、プラットフォームの MIDI 2.0 バックエンドがサポートしていない限り、Android のようには「アプリ間 MIDI 2.0」を利用できない可能性があります。

実用上のまとめ:

- **MIDI 1.0 のアプリ間 MIDI は、iOS/macOS/Linux のプラットフォーム MIDI バックエンドを介して明確にサポートされています。**
- **MIDI 2.0 (UMP) のアプリ間 MIDI は、Android 用に明示的に統合されています。** 他のプラットフォームについては、使用している MIDI 2.0 バックエンドの実装に依存します。

3) 使用方法 (全プラットフォーム共通 API)

アプリ間 MIDI の受信 (MIDI 1.0)

1. 1 つ以上の MIDI 1.0 ハンドラインターフェース（例: `IMidiNoteOnEventHandler`）を実装します。
2. そのオブジェクトを `MidiManager` に登録します。
3. MIDI を初期化します (`MidiManager.InitializeMidi(...)`)。
4. OS やアプリのルーティング設定を使用して、他のアプリを自アプリのエンドポイントに接続（またはその逆）します。
5. メッセージが到着するとハンドラが実行されます。

アプリ間 MIDI の送信 (MIDI 1.0)

- `MidiManager.Instance.SendMidiNoteOn(...)` / `SendMidiControlChange(...)` / `SendMidiSystemExclusive(...)` などを使用します。
- 送信先となるアプリ/ポートに対応する適切な `deviceId` を選択してください。

4) プラットフォーム別「アプリの接続方法」(概念)

> ルーティング用の UI やツールは OS のバージョンやサードパーティ製アプリによって異なるため、ここでは概念的な説明に留めます。

Android

- Android アプリ間 MIDI サービス/プラグイン統合によってルーティングが処理されます。
- デバイス ID は、そのレイヤーを通じて公開されるエンドポイントに対応します。

iOS / macOS

- 他のアプリが CoreMIDI ポート（仮想ソース/デスティネーション）を公開します。
- 通常、以下のいずれかを使用してアプリを接続します:
  - 各アプリ内の MIDI 設定 UI。
  - OS レベルの MIDI ルーティングツール（デバイスやワークフローにより異なります）。
- 接続されると、Unity アプリからはそれらが MIDI デバイスとして認識されます。

Linux

- アプリが ALSA/JACK を介してポートを公開します。
- ルーティングは通常、外部のパッチベイや接続ツールを使用して行われます。
- ルーティングされると、Unity アプリは対応するポートを介して送受信を行います。

5) 注意点、制限事項、トラブルシューティング

プラットフォームによる「アプリ間 MIDI」の呼称の違い

- Android: このリポジトリにおける明示的な「Inter-App MIDI」プラグイン/サービス。
- iOS/macOS: CoreMIDI エンドポイント（設計上、アプリ間通信が可能）。
- Linux: ALSA/JACK ポート（設計上、アプリ間通信が可能）。

デバイス ID の形式

デバイス識別子はプラットフォームやトランスポートによって定義されます。OS をまたいで同じ `deviceId` 形式であるとは想定しないでください。

二重送信 (Android)

Android では、初期化されているトランスポート（USB/BLE/Inter-App）によっては、MIDI 1.0 バックエンドが複数の経路で送信を行う可能性があります。メッセージが重複して届く場合は、対象のエンドポイントと、同じ論理デバイスに対して複数のトランスポートがアクティブになっていないかを確認してください。

MIDI 2.0 アプリ間通信への期待

アプリ間での MIDI 2.0 UMP 通信 を目的とする場合は、以下を確認してください:

- ターゲットプラットフォームの MIDI 2.0 バックエンドがそれをサポートしていること。
- 通信相手のアプリがそのプラットフォームで実際に MIDI 2.0/UMP をサポートしていること。

6) 関連ドキュメント

- MIDI 1.0 (MidiManager)
- MIDI 2.0 / UMP (Midi2Manager)
- トランスポートとプラットフォーム

ソースコード実装例 (アプリ間 MIDI の実用例)

アプリ間 MIDI は主に **ルーティング** に関する事項であり、Unity 側のコードは単に「デバイスを列挙し、正しい deviceId を選び、送受信する」だけです。

MIDI 1.0 クイックスタートコンポーネントを使用してください:

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

アプリ間ルーティングの確認方法:

1. スクリプトをアタッチして Play モードを開始します。
2. OS や MIDI アプリを使用して、仮想ソース/デスティネーションを Unity アプリ（または他のアプリ）にルーティングします。
3. Unity のログを確認します:

- デバイスの接続イベントが表示されれば、エンドポイントが認識されています。
- Note On のログが表示されれば、アプリ間通信による受信が確認できます。

4. テストノートを送信し、対象のアプリに届くことを確認します。