MIDI 1.0 ランタイム (MidiManager)

このページでは、`jp.kshoji.unity.midi.MidiManager` と `IMidiPlugin` バックエンドインターフェースを中心とした MIDI 1.0 ランタイム API について説明します。

> 注意: このプロジェクトは、複数のトランスポート（プラットフォーム固有のネイティブ MIDI、WebGL MIDI、RTP-MIDI、Nearby Connections など）をサポートしています。これらはすべて、`IMidiPlugin` と `MidiManager` を通じて共通の API を提供します。

インストールやビルドの要件については、以下を参照してください:

- スタートガイド
- ビルド後の処理とスクリプト定義シンボル
- プラットフォームと制限事項

主な役割

`MidiManager` は以下の役割を担います:

- アクティブな MIDI プラグインバックエンドの初期化と終了処理。
- MIDI 1.0 メッセージ (Note On/Off, CC, PC, アフタータッチ, ピッチホイール, SysEx, システムメッセージ) の受信。
- **Unity EventSystem** ハンドラインターフェースを介した Unity オブジェクトへのメッセージ配信。
- 現在のバックエンドを通じた MIDI 1.0 メッセージの送信。

初期化と終了 (推奨パターン)

`Awake` で初期化し、`OnDestroy` で終了します:

```
using UnityEngine;
using jp.kshoji.unity.midi;

public sealed class Midi1LifecycleExample : MonoBehaviour
{
    private void Awake()
    {
        // イベントを処理するオブジェクトを登録
        MidiManager.Instance.RegisterEventHandleObject(gameObject);

        // MIDI 初期化
        MidiManager.Instance.InitializeMidi(() =>
        {
#if (UNITY_ANDROID || UNITY_IOS || UNITY_WEBGL) && !UNITY_EDITOR
            // BLE MIDI のみ (サポートされている場合): 初期化完了後にスキャンを開始
            MidiManager.Instance.StartScanBluetoothMidiDevices(0);
#endif
        });
    }

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

> シーン内に既に EventSystem が存在し、重複の警告やエラーが表示される場合は、 マネージャーの初期化内での EventSystem 自動追加を削除してください。

バックエンドの抽象化: IMidiPlugin

`IMidiPlugin` は、プラットフォームやトランスポートに依存する実装を定義します:

- ライフサイクル:
  - `InitializeMidi(Action initializeCompletedAction)`
  - `TerminateMidi()`
  - (Editor 用) `PlayModeStateChanged(...)`
- オプションの BLE スキャン/アドバタイズ (コンパイルシンボルでプラットフォームを制限)。
- デバイスメタデータ:
  - `GetDeviceName(deviceId)`
  - `GetVendorId(deviceId)`
  - `GetProductId(deviceId)`
- MIDI 1.0 送信メソッド:
  - Note On/Off, CC, PC, アフタータッチ, ピッチホイール
  - SysEx および システム・リアルタイム / コモンメッセージ (プラットフォームにより利用可否が異なります)

具体的な実装は以下のファイルにあります:

- `MidiPlugin.Android.cs`, `MidiPlugin.Apple.cs`, `MidiPlugin.Linux.cs`, `MidiPlugin.Windows.cs`, `MidiPlugin.WebGL.cs`
- `MidiPlugin.RtpMidi.cs` (RTP-MIDI トランスポート)
- `MidiPlugin.Nearby.cs` (Nearby トランスポート)

デバイスメタデータ: 名前 / ベンダー ID / プロダクト ID

- `MidiManager.Instance.GetDeviceName(deviceId)`
- `MidiManager.Instance.GetVendorId(deviceId)`
- `MidiManager.Instance.GetProductId(deviceId)`

重要な注意点:

- 一部のプラットフォームやトランスポートはベンダー ID やプロダクト ID をサポートしていません。その場合、空文字列が返されます。
- デバイスが切断された後は、これらのメソッドは空文字列を返す可能性があります。

ベンダー ID / プロダクト ID の対応状況

プラットフォーム	Bluetooth MIDI	USB MIDI	ネットワーク MIDI (RTP-MIDI)	Nearby Connections MIDI
iOS	○	○	-	-
Android	○	○	-	-
Universal Windows Platform	-	○	-	-
Standalone macOS / Unity Editor macOS	○	○	-	-
Standalone Linux / Unity Editor Linux	-	-	-	-
Standalone Windows / Unity Editor Windows	-	○	-	-
WebGL	-	△ (ベンダー ID のみ)	-	-

ベンダー ID の例

基盤となるプラットフォーム API が異なるため、取得される `VendorId` 文字列は OS によって異なる場合があります。

プラットフォーム	Bluetooth MIDI	USB MIDI	ネットワーク MIDI (RTP-MIDI)	Nearby Connections MIDI
iOS	QUICCO SOUND Corp.	Generic	-	-
Android	QUICCO SOUND Corp.	1410	-	-
Universal Windows Platform	-	VID_0582	-	-

プラットフォーム	Bluetooth MIDI	USB MIDI	ネットワーク MIDI (RTP-MIDI)	Nearby Connections MIDI
Standalone macOS / Unity Editor macOS	QUICCO SOUND Corp.	Generic	-	-
Standalone Linux / Unity Editor Linux	-	-	-	-
Standalone Windows / Unity Editor Windows	-	1	-	-
WebGL	QUICCO SOUND Corp.	Microsoft Corporation	-	-

プロダクト ID の例

`ProductId` についても同様です。

プラットフォーム	Bluetooth MIDI	USB MIDI	ネットワーク MIDI (RTP-MIDI)	Nearby Connections MIDI
iOS	mi.1	USB2.0-MIDI	-	-
Android	mi.1	298	-	-
Universal Windows Platform	-	PID_012A	-	-
Standalone macOS / Unity Editor macOS	mi.1	USB2.0-MIDI	-	-
Standalone Linux / Unity Editor Linux	-	-	-	-
Standalone Windows / Unity Editor Windows	-	102	-	-
WebGL	mi.1	UM-ONE	-	-

イベント配信モデル (Unity EventSystem)

ハンドラインターフェース

MIDI 1.0 イベントハンドラインターフェースは以下で定義されています:

- `Assets/MIDI/Scripts/IMidiEventHandler.cs`
- `Assets/MIDI/Scripts/IMidiDeviceEventHandler.cs`

これらは「1 イベント 1 インターフェース」の小規模なインターフェースです:

- `IMidiNoteOnEventHandler`
- `IMidiControlChangeEventHandler`
- `IMidiSystemExclusiveEventHandler`
- ... 加えて Start/Stop/Clock, Song Select などのシステムメッセージ用ハンドラ

複数のイベントをまとめたインターフェースもあります:

- `IMidiPlayingEventsHandler` (ノート + チャンネルイベント)
- `IMidiSystemEventsHandler` (システムおよび SysEx)
- `IMidiAllEventsHandler` (すべてのイベント)

ハンドラの登録

`MidiManager` は、イベントハンドラオブジェクトを登録/解除するためのメソッドを提供します:

- Unity のコンポーネントやオブジェクトに 1 つ以上の `IMidi*EventHandler` インターフェースを実装します。
- そのオブジェクトを `MidiManager` に登録します。
- MIDI メッセージを受信すると、`MidiManager` は一致するすべてのハンドラインターフェースのコールバックを呼び出します。

C# デリゲートによるイベント受信 (オプション)

インターフェースの代わりに、以下のような名前の C# イベントを購読(Subscribe)することもできます:

- `MidiManager.Instance.OnMidiXXXXEvent`

例:

```
using UnityEngine;
using jp.kshoji.unity.midi;

public sealed class Midi1DelegateExample : MonoBehaviour
{
    private void OnEnable()
    {
        MidiManager.Instance.OnMidiNoteOnEvent += OnMidiNoteOn;
    }

    private void OnDisable()
    {
        MidiManager.Instance.OnMidiNoteOnEvent -= OnMidiNoteOn;
    }

    private void OnMidiNoteOn(string deviceId, int group, int channel, int note,
int velocity)
        => Debug.Log($"NoteOn デバイス:{deviceId} ch:{channel} note:{note} vel:
{velocity}");
}
```

Bluetooth MIDI ペリフェラルモード (Android のみ)

BLE MIDI デバイスをスキャン・接続するだけでなく、Android ではアプリを **BLE MIDI ペリフェラル** としてアドバタイズ（公開）できます。

API:

- アドバタイズ開始: `MidiManager.Instance.StartAdvertisingBluetoothMidiDevice()`
- アドバタイズ停止: `MidiManager.Instance.StopAdvertisingBluetoothMidiDevice()`

注意点:

- この機能は **Android 限定** です。
- アドバタイズはスキャンとは別個の機能です。相手が接続した後は、通常の MIDI 送受信が可能です。
- 高度な Android BLE ペアリングワークフローを使用している場合は、以下の `FEATURE_ANDROID_COMPANION_DEVICE` を参照してください: ビルド後の処理とスクリプト定義シンボル

MIDI 1.0 メッセージの送信

`MidiManager` は一般的なメッセージ用のヘルパーメソッドを提供しています:

- ノート・オン/オフ: `SendMidiNoteOn`, `SendMidiNoteOff`
- コントロール・チェンジ (CC): `SendMidiControlChange`
- プログラム・チェンジ (PC): `SendMidiProgramChange`
- アフタータッチ: `SendMidiPolyphonicAftertouch`, `SendMidiChannelAftertouch`
- ピッチホイール: `SendMidiPitchWheel`
- SysEx: `SendMidiSystemExclusive`
- システムメッセージ: クロック, 開始/継続/停止, リセットなど

パラメータの範囲 (一般的)

- `group`: 0–15 (MIDI 1.0 バックエンドでは無視されることが多いですが、API の一貫性のために含まれています)
- `channel`: 0–15
- `note`: 0–127
- `velocity`: 0–127
- `pressure`: 0–127
- `amount` (ピッチホイール): 0–16383
- SysEx: 通常 `0xF0` で始まり `0xF7` で終わるバイト配列

RTP-MIDI ヘルパーメソッド (有効な場合)

RTP-MIDI プラグインが使用されている場合、`MidiManager` は以下のメソッドを公開します:

- ポートごとの RTP-MIDI サーバーの開始/停止
- エンドポイントへの接続/切断
- リスナーが実行中かどうかの確認

参照:

- トランスポートとプラットフォームの注意点

ソースコード実装例 (MIDI 1.0 クイックスタート)

新しいスクリプトを作成するか、既存のものを使用してください:

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

これを GameObject にアタッチして Play モードに入り、MIDI デバイスを接続してください。 以下の動作が行われます:

- デバイスの接続/切断をログ出力
- 受信した Note On をログ出力
- (オプション) 最初の出力デバイスにテストノートを送信