Getting Started (Install, Initialize, Send/Receive)

Install

1. Import the `.unitypackage` from the Asset Store.
    - If the package was updated over an older version, check: `Assets/MIDI/Plugins/Android` and remove any older/duplicate `.aar` files if present.
2. Switch to your target platform (iOS / Android / Standalone / WebGL / UWP).
3. Build and run the sample scenes:
    - `Assets/MIDI/Samples/Scenes/`

If you previously installed optional dependencies (e.g., Nearby), update them to latest.

MIDI 1.0 quick start (receive + optional BLE scan)

Recommended lifecycle:

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

public sealed class Midi1QuickStart : MonoBehaviour, IMidiDeviceEventHandler,
IMidiNoteOnEventHandler
{
    private void Awake()
    {
        MidiManager.Instance.RegisterEventHandleObject(gameObject);

        MidiManager.Instance.InitializeMidi(() =>
        {
#if (UNITY_ANDROID || UNITY_IOS || UNITY_WEBGL) && !UNITY_EDITOR
            // BLE MIDI only (where supported): start scan after init completes.
            MidiManager.Instance.StartScanBluetoothMidiDevices(0);
#endif
        });
    }

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

    public void OnMidiInputDeviceAttached(string deviceId)
        => Debug.Log($"Input attached: {deviceId}");

    public void OnMidiInputDeviceDetached(string deviceId)
        => Debug.Log($"Input detached: {deviceId}");

    public void OnMidiOutputDeviceAttached(string deviceId)
        => Debug.Log($"Output attached: {deviceId}");

    public void OnMidiOutputDeviceDetached(string deviceId)
        => Debug.Log($"Output detached: {deviceId}");

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

MIDI 1.0 send

```
// Send Note On
MidiManager.Instance.SendMidiNoteOn(
    "deviceId",
    0 /*group*/,
    0 /*channel*/,
    60 /*note*/,
    127 /*velocity*/
);
```

You can get output device IDs from:

- `MidiManager.Instance.OutputDeviceIdSet` (type: `HashSet<string>`)

MIDI 2.0 quick start

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

public sealed class Midi2QuickStart : MonoBehaviour, IMidi2DeviceEventHandler,
IMidi2NoteOnEventHandler
{
    private void Awake()
    {
        MidiManager.Instance.RegisterEventHandleObject(gameObject);
        MidiManager.Instance.InitializeMidi2(() => { });
    }

    private void OnDestroy()
    {
        MidiManager.Instance.TerminateMidi2();
    }

    public void OnMidi2InputDeviceAttached(string deviceId)
        => Debug.Log($"MIDI2 In attached: {deviceId}");

    public void OnMidi2InputDeviceDetached(string deviceId)
        => Debug.Log($"MIDI2 In detached: {deviceId}");

    public void OnMidi2OutputDeviceAttached(string deviceId)
        => Debug.Log($"MIDI2 Out attached: {deviceId}");

    public void OnMidi2OutputDeviceDetached(string deviceId)
        => Debug.Log($"MIDI2 Out detached: {deviceId}");

    public void OnMidi2NoteOn(string deviceId, int group, int channel, int note,
int velocity, int attributeType, int attributeData)
        => Debug.Log($"MIDI2 NoteOn dev:{deviceId} g:{group} ch:{channel} note:
{note} vel:{velocity}");
}
```

Where to look next

- Platforms & Limitations
- Build PostProcessing & Scripting Define Symbols
- Transports & Platform Notes
- Samples