

入门指南（安装、初始化、发送/接收）

安装

1. 从 Asset Store 导入 [.unitypackage](#)。
 - 如果是在旧版本基础上覆盖更新，请务必检查：[Assets/MIDI/Plugins/Android](#) 并手动删除任何旧的或重复的 [.aar](#) 文件。
2. 切换到您的目标平台（iOS / Android / Standalone / WebGL / UWP）。
3. 构建并运行示例场景：
 - [Assets/MIDI/Samples/Scenes/](#)

如果您之前安装了可选依赖项（例如 Nearby），请将它们更新到最新版本。

MIDI 1.0 快速入门（接收 + 可选的 BLE 扫描）

推荐的生命周期：

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

public sealed class Midi1QuickStart : MonoBehaviour, IMidiDeviceEventHandler,
IMidiNoteOnEventHandler
{
    private void Awake()
    {
        // 注册事件处理对象
        MidiManager.Instance.RegisterEventHandleObject(gameObject);

        // 初始化 MIDI
        MidiManager.Instance.InitializeMidi(() =>
        {
#ifndef UNITY_ANDROID || UNITY_IOS || UNITY_WEBGL && !UNITY_EDITOR
            // 仅限支持 BLE MIDI 的平台：初始化完成后开始扫描。
            MidiManager.Instance.StartScanBluetoothMidiDevices(0);
#endif
        });
    }

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

    // 设备连接事件
    public void OnMidiInputDeviceAttached(string deviceId)
        => Debug.Log($"输入设备已连接: {deviceId}");

    public void OnMidiInputDeviceDetached(string deviceId)
        => Debug.Log($"输入设备已断开: {deviceId}");

    public void OnMidiOutputDeviceAttached(string deviceId)
        => Debug.Log($"输出设备已连接: {deviceId}");

    public void OnMidiOutputDeviceDetached(string deviceId)
        => Debug.Log($"输出设备已断开: {deviceId}");

    // MIDI 事件示例：Note On
    public void OnMidiNoteOn(string deviceId, int group, int channel, int note,
int velocity)
        => Debug.Log($"NoteOn 设备:{deviceId} 组:{group} 通道:{channel} 音符:{note}
力度:{velocity}");
}
```

MIDI 1.0 发送

```
// 发送 Note On
MidiManager.Instance.SendMidiNoteOn(
    "deviceId",
    0 /* group (组) */,
    0 /* channel (通道) */,
    60 /* note (音符) */,
    127 /* velocity (力度) */
);
```

您可以从以下位置获取输出设备 ID：

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

MIDI 2.0 快速入门

```
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 输入已连接: {deviceId}");

    public void OnMidi2InputDeviceDetached(string deviceId)
        => Debug.Log($"MIDI2 输入已断开: {deviceId}");

    public void OnMidi2OutputDeviceAttached(string deviceId)
        => Debug.Log($"MIDI2 输出已连接: {deviceId}");

    public void OnMidi2OutputDeviceDetached(string deviceId)
        => Debug.Log($"MIDI2 输出已断开: {deviceId}");

    public void OnMidi2NoteOn(string deviceId, int group, int channel, int note,
int velocity, int attributeType, int attributeData)
        => Debug.Log($"MIDI2 NoteOn 设备:{deviceId} 组:{group} 通道:{channel} 音符:{note} 力度:{velocity} 属性类型:{attributeType} 属性数据:{attributeData}");
}
```

```
{note} 力度:{velocity}"");  
}
```

后续阅读内容

- [平台与限制](#)
- [构建后处理与脚本定义符号](#)
- [传输协议与平台说明](#)
- [示例项目](#)