# SMF / 序列化 (jp.kshoji.midisystem)

本页面介绍了位于以下目录中的 SMF（标准 MIDI 文件）和序列化相关代码：

- `Assets/MIDI/Scripts/midisystem/`

命名空间：`jp.kshoji.midisystem`

## 本模块提供的功能

- MIDI 序列的内存模型：
  - `Sequence` (序列)
  - `Track` (音轨)
  - `MidiEvent` (MIDI 事件)
- MIDI 消息表示：
  - `MidiMessage` (基类)
  - `ShortMessage` (大多数通道/系统消息)
  - `SysexMessage` (系统独占消息)
  - `MetaMessage` (元消息)
- SMF 文件 I/O：
  - `StandardMidiFileReader` (读)
  - `StandardMidiFileWriter` (写)
- 序列化接口与实现：
  - `ISequencer`, `SequencerImpl`
  - `IReceiver`, `ITransmitter`
  - 元消息/控制器事件监听器

这在概念上类似于 Java MIDI API 模型 (Sequence/Track/MidiEvent)。

## StandardMidiFileWriter 行为（重要细节）

在写入 `Sequence` 时：

- 支持 SMF 0 型和 1 型（取决于音轨数量）。
- 写入包含以下内容的 `MThd` 文件头：
  - 文件类型
  - 音轨数量
  - 分辨率类型/分辨率编码
- 对于每个音轨：
  - 在写入前计算音轨长度。
  - 写入时跳过**系统实时 (System Realtime)** 消息（状态值 >= `0xF8` 的 `ShortMessage`）。
  - 写入内容包括：
    - Delta time（变长整数）
    - 消息字节 (ShortMessage)，或
    - `0xFF` + 元消息类型 + 长度 + 数据 (MetaMessage)，或
    - `0xF0` + 长度 + 数据 (SysexMessage)
  - 确保存在 **End of Track**（音轨结束）元事件；如果缺失则会自动补全。

## 序列化工作流（播放 / 录制 / 导出）

本节参考了随附手册中的序列化器用法模式。

### 创建并打开序列化器

```
using jp.kshoji.midisystem;

var isSequencerOpened = false;

// 创建序列化器并打开它
var sequencer = new SequencerImpl(() => { isSequencerOpened = true; });
sequencer.Open();
```

### 将 SMF 读取到 Sequence 并播放

```
using System.IO;
using jp.kshoji.midisystem;

// 更新设备连接
sequencer.UpdateDeviceConnections();

using var stream = new FileStream(smfPath, FileMode.Open, FileAccess.Read);
sequencer.SetSequence(stream);

sequencer.Start();

// ... 稍后
sequencer.Stop();
```

### 录制序列

```
using jp.kshoji.midisystem;

sequencer.UpdateDeviceConnections();

// 使用 PPQ 的示例
sequencer.SetSequence(new Sequence(Sequence.Ppq, 480));

sequencer.StartRecording();

// ... 稍后
sequencer.Stop();
```

### 将录制的序列写入 SMF 文件

```
using System.IO;
using jp.kshoji.midisystem;

var sequence = sequencer.GetSequence();

if (sequence.GetTickLength() > 0)
{
    using var stream = new FileStream(recordedSmfPath, FileMode.Create,
FileAccess.Write);
    MidiSystem.WriteSequence(sequence, stream);
}
```

## 与 UMP 工具的关系

位于 `Assets/MIDI/Scripts/UmpSequencer/` 的 UMP 工具提供了以下转换功能：

- SMF `Sequence` (MIDI 1.0 表示)
- UMP `UmpSequence` (面向 MIDI 2.0 数据包的表示)

在以下场景中可以使用它：

- 导入/导出 MIDI 文件 (SMF)
- 为 MIDI 2.0 工作流转换为 UMP 剪辑 (Clip)
- 在转换器支持的情况下，将 MIDI 2.0 剪辑文件转换为 SMF（反之亦然）。

## 实践指南

- 对于简单的“通过发送消息来播放 MIDI 文件”，通常步骤如下：

    1. 将 SMF 读取到 `Sequence` 中。
    2. 按 Tick 遍历事件（或使用序列化器）。
    3. 通过 `MidiManager` 发送 (MIDI 1.0) 或转换为 UMP 后通过 `Midi2Manager` 发送 (MIDI 2.0)。

- 如果您需要精确的时间控制和传输控制（播放/停止/循环），请考虑使用：

    - `SequencerImpl` (针对 SMF 风格的序列化)，或者
    - 如果您的目标是以 UMP 为中心的，请使用 MIDI 2.0 UMP 剪辑序列化工作流（参见 MIDI 2.0 / UMP）。

## 源代码示例实现 (SMF → MIDI 输出)

请参考：

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

如何测试：

1. 将一个 `.mid` 文件放入 `Assets/StreamingAssets/`（如果文件夹不存在请创建）。
2. 设置 `streamingAssetsMidiFileName`（例如 `test.mid`）。
3. 将该组件附加到 GameObject 上并进入播放模式。

该示例通过 `StandardMidiFileReader` 读取文件，遍历事件，并通过 `MidiManager` 发送基础消息。