Editor & Lifecycle Notes

This page documents a few “Unity-specific” behaviors that often explain why MIDI works differently in Editor vs builds.

Manager lifetime and `DontDestroyOnLoad`

`MidiManager` is created as a `DontDestroyOnLoad` GameObject the first time `MidiManager.Instance` is accessed (during Play Mode).

Practical implications:

- You generally initialize once per play session.
- Don’t call into the manager from Editor tooling while Unity is not playing.

Unity Editor play mode transitions

The project includes Editor hooks so transports can start/stop cleanly as you enter/exit Play Mode.

If you see:

- MIDI still arriving after you stop Play Mode, or
- devices not closing properly,

make sure:

- you terminate MIDI on shutdown (`TerminateMidi` / `TerminateMidi2`), and
- your scripts don't keep static references alive across play sessions.

Main-thread dispatching

Some transports receive messages on background threads (e.g., network transports). Incoming events are forwarded to Unity in a main-thread-safe way.

Practical implication:

- Your event handlers should assume callbacks occur on Unity's main thread.
- Heavy processing should still be avoided in callbacks; queue work if needed.

EventSystem note

If your scene already contains a Unity `EventSystem`, avoid creating duplicates.
If you hit duplication warnings/errors, adjust initialization accordingly (see Getting Started).