Page contents

- The mod boundary
- Choose a lifecycle event deliberately
- Content APIs and compatibility
- A controlled development and release loop
- Coverage still needed
- Sources and provenance

Offline copy of SMAPI: events and game content .

Stardew Valley SMAPI: events and game content

Documentation status: In progress. **Source checked:** 2026-10-06. **Target:** the Stardew Valley wiki's .NET 6 SMAPI mod setup and current event/content API. **Compiled or in-game validation:** not recorded. This path is for gameplay logic in C#; use the Content Patcher chapter for declarative asset edits and the quick start for the first pack.

The mod boundary

A SMAPI C# mod is a compiled class library. Its `manifest.json` identifies the mod and names its entry DLL; `ModEntry` subclasses `StardewModdingAPI.Mod`, and SMAPI calls `Entry(IModHelper)` when loading it. The maintained getting-started guide currently targets `net6.0` and uses `Pathoschild.Stardew.ModBuildConfig` to find game references and prepare output across operating systems. Check that guide and the build package notes before pinning a package version. The build result and `manifest.json` belong together inside one mod folder; a source `.cs` file alone cannot load as a SMAPI mod.

`Entry` is setup time. It is a good place to register event handlers, not to assume a save or player exists. The event that runs after a save loads is different from one that

runs after the game launches. Choose an event for the state you need, then verify it with a log message before changing gameplay.

Choose a lifecycle event deliberately

Event	When it is useful	State or cost to consider
`GameLoop.GameLaunched`	Integrate with another loaded mod once per session	It does not mean a save is loaded.
`GameLoop.SaveLoaded`	Initialize state for a loaded save	The world is ready, but day-start setup follows.
`GameLoop.DayStarted`	Do work once after a new day is initialized	It also runs after connecting to a multiplayer world.
`GameLoop.Saving` / `Saved`	Prepare or react around a save	Do not assume the same timing or mutation safety for both.
`GameLoop.UpdateTicked`	Observe changes without a dedicated event	It runs about 60 times per second, so expensive work here can slow the game.
`Content.AssetRequested`	Supply or edit an asset as the content pipeline loads it	It can be raised more than once for a target. Keep edits repeatable.

The SMAPI events reference lists event arguments, timing and other categories such as input, display and multiplayer. Register with `helper.Events.GameLoop.DayStarted += this.OnDayStarted;` in `Entry`, with a handler signature matching the current API. Keep logic that needs a save out of `GameLaunched` and check world-readiness for generic input/update handlers. Event handlers run synchronously; long work delays the game.

Content APIs and compatibility

The SMAPI content API distinguishes the files private to your mod from **game assets** in the shared content pipeline. To change a game image, map or data asset in C#, register `Content.AssetRequested` and examine `e.Name` or `e.NameWithoutLocale` with `IsEquivalentTo` rather than comparing a guessed disk path. `e.LoadFromModFile` can supply a file; `e.Edit` changes an asset after it loads. Select a priority only when the order relative to other edits is part of your mod's contract.

Asset requests can occur more than once, and several mods can edit the same asset. A handler should apply the same intended change each time, avoid mutating global state as a side effect of a request, and log enough context to diagnose conflicts. For content that should change with game state, invalidate a specific cached asset only when needed; invalidating every tick is expensive. A localized asset may require attention to the localized cache key. Do not patch the installed `.xnb` in place to accomplish a runtime edit.

The C# content API and Content Patcher can both affect the same game asset. Test their order with the actual installed mod set. If a change is only a static image/data/map edit, a Content Patcher pack often leaves a simpler, inspectable compatibility surface. Use C# when the desired behavior requires logic beyond the pack schema.

A controlled development and release loop

1. Create the class-library project and manifest from the maintained SMAPI getting-started guide. Compile for its stated target framework; do not infer that a newer SDK means the runtime target can be newer.
2. Add one log line in `Entry` and launch through SMAPI. Confirm the manifest ID, DLL path and entry class are accepted.
3. Register one event, log once at its intended trigger, and test both at the title screen and inside a disposable save. A successful load does not prove the handler ran.
4. If editing content, record the exact asset name and inspect the result in game. Recheck after a save/reload and with another mod that edits that asset.
5. Package the built DLL, manifest and needed assets in one release folder. The mod build package can automate installing/debugging and release ZIP creation; inspect its output before publishing. Record exact game/SMAPI/dependency versions and preserve source code.

The manifest API defines identity, dependencies and update keys. The test and troubleshoot and release guides cover logs and distribution. They remain necessary because this chapter is a conceptual path, not a compile-tested sample project.

Coverage still needed

SMAPI documents configuration, data storage, input, logging, multiplayer, translations, console commands, mod integrations and Harmony patching. Those APIs, full signatures, lifecycle edge cases, and tested example projects need separate chapters before this documentation can serve as a complete reference. The overall Stardew documentation remains **In progress**.

Sources and provenance

- SMAPI getting started — Stardew Valley wiki modder guide; read 2026-10-06; project/runtime setup.
- SMAPI events and content API — wiki API reference; read 2026-10-06; lifecycle and asset behavior.
- SMAPI mod build package — framework maintainer documentation; read 2026-10-06; build and release behavior.
- SMAPI source repository — LGPL-3.0 repository; this chapter is original ModDocs prose rather than a redistribution of upstream code or game assets.