Page contents

- Use callbacks according to their lifetime
- Settings and translations
- Sprites and sound
- Debugging and compatibility
- Release and offline preservation
- Remaining validation
- Sources and provenance

Offline copy of Settings, assets, debugging and distribution .

Noita: settings, assets, debugging and distribution

Documentation status: In progress. **Source checked:** 2026-10-06. **In-game validation:** not recorded. This chapter covers the release side of the Noita mod lifecycle and content/world workflow.

Use callbacks according to their lifetime

The Noita Lua API callback reference lists `OnModPreInit`, `OnModInit`, `OnModPostInit`, world callbacks and `OnPlayerSpawned`. It notes that the player-spawn callback runs on a new game and on load, and world-update callbacks can run before `OnWorldInitialized`. Put one-time file append/registration in initialization, use spawn logic only when a player exists, and guard effects that must occur once per run. A log line in `OnModInit` proves loading but does not prove that world entities, perks or spell actions are valid.

Settings and translations

The same Lua API reference documents `settings.lua` callbacks, `ModSettingGet` and `ModSettingSet`. The setting ID is normally `mod_name.setting_id`. Define a stable default, read it at the appropriate callback, and decide whether it should take effect immediately or after restart. Test main-menu and paused-game changes, then restart and reload to check persistence. The reference flags that `OnModSettingsChanged` may not reliably fire, so use a controlled re-read path rather than depending on it alone.

Translate every player-facing string through the game's expected CSV/localization path. The custom perk guide explains that perk labels must be compatible with the translation system. Preserve translation keys in a source file and test at least one non-English language, missing key behavior and clean installation. Do not ship a copied full vanilla translation table when a small appended set will work.

Sprites and sound

Keep image files under your mod-owned path and refer to them from the entity, spell, perk or UI definition that displays them. The Noita Wiki spritesheet guide describes generated UV data and the developer repository has a character sprite guide. Check sprite dimensions and generated paths with the installed tools. Test in world, inventory and menu surfaces separately; a valid image file can still have the wrong origin, animation or XML reference.

Noita uses FMOD banks for audio, as the Noita audio guide explains. The installed `tools_modding/noita-fmod-project` and `mods/example` are the local starting points for original sounds. Define a mod-owned event/bank, package the bank and any required mapping, and test both event trigger and audible output in a clean installation. Replacing a vanilla bank is a broad override that can conflict with other audio mods; record every replacement path and avoid distributing extracted original audio.

Debugging and compatibility

Read `logger.txt` after each controlled change. The modding basics describes a developer executable in `tools_modding` with a console; check what ships in the installed build before relying on it. When content fails, isolate the stage: mod enabled, init callback, append/override path, XML/Lua syntax, resource path, registration list, spawn/acquisition and save/load. Test with only your mod, then with a known

interacting mod. `ModGetActiveModIDs` in the Lua API can report active IDs in load order; record that order in a bug report.

For a world change, compare fresh seeds and loaded saves separately. For a setting, compare menu, paused state and restarted state. For a spell, compare action registration, wand cast and projectile. For an entity, compare direct spawn with normal biome placement. This staged matrix prevents a successful manual spawn from being mistaken for a functioning world-generation feature.

Release and offline preservation

Nolla Games' publishing guide describes a Steam-only Workshop path: `workshop.xml`, a 16:9 `workshop_preview_image.png`, upload exclusions, and an uploader shipped with the game. It also lists file types eligible for upload and warns that mods requesting unrestricted Lua API access do not work through Workshop. Check those rules against the installed uploader before publishing; the online guide may be older than the game build. Then install the Workshop item as a subscriber to confirm the uploaded package contains every Lua/XML/asset/bank file. For non-Steam distribution, package the mod folder and a README without extracted game data.

Preserve source, a manifest of paths, the exact game build, tested mod order, screenshots or logs of outcomes, and a ZIP of your own release folder. Document which files were appended and which vanilla paths were overridden. This makes future migration and conflict repair possible even if an online link or Workshop listing disappears.

Remaining validation

The site now describes the requested Noita surfaces, but no local Noita installation was available for an independent run. Exact Lua/XML examples for each content family, verified settings/UI behavior, FMOD build output, biome generation and a Workshop subscriber install remain open. The guide stays **In progress** until those are exercised.

Sources and provenance

- Noita Lua API, modding basics, spritesheets, audio, and custom perk — community-maintained technical references; checked 2026-10-06.
- Nolla Games publishing guide and modding repository — developer-maintained publication and local-tool references; checked 2026-10-06.
- This is original ModDocs prose. No game files, copied audio or third-party document text are included.