Page contents

- Package a sprite atlas deliberately
- Move growing text into localization
- Keep preferences separate from run state
- Build a player-installable package
- Version record and remaining checks
- Sources and provenance

Offline copy of Atlases, localization, config and release .

Balatro: atlases, localization, config and release

Documentation status: In progress. **Sources checked:** 2026-10-12. **In-game validation:** not recorded. This chapter closes the release path after the first Joker, content families and scoring/patching chapters.

Package a sprite atlas deliberately

Steamodded's Atlas reference registers an atlas with a key, image path and frame dimensions (`px`, `py`). A card-like object references that atlas and a zero-based `pos`. The first Joker guide shows a 71 × 95 frame. Test the real image with the loader release used for packaging: Steamodded's docs note that newer versions can generate a missing resolution, while older releases may need both `assets/1x` and `assets/2x`. Verify the frame in collection, shop, in-run animation and enlarged view. A valid PNG can still point to the wrong atlas key or frame.

For an edition shader, follow the Edition reference and Shader object together. `shader = false` is supported when a new edition needs no shader. If it does use one, test all card faces and scale changes and confirm the packaged shader file is present. Sound and UI changes have their own Steamodded objects; do not treat them as atlas metadata. Only include art, audio and shader code you can distribute.

Move growing text into localization

`loc_txt` is useful for a first object, but the Localization reference documents `localization/en-us.lua` or JSON and fallback `localization/default.lua`. The category must match the object type, and the lookup key includes the registered prefix. Make a short inventory for each line: object full key, language, category, name, description and any variable placeholders. Test a second language and the fallback, then inspect the actual packaged directory. A collection label and a shop tooltip can fail independently.

The metadata specification describes the mod's `id`, `prefix`, `main_file`, version, dependencies and conflicts. Keep the JSON file valid and at the same layer as its Lua and asset paths. A legacy Steamodded header is not needed when using the JSON method and can cause duplicate loading if mixed with it. If a compatibility object is conditional on another mod, distinguish the mod-level dependency from the individual object's `dependencies` in the game-object API.

Keep preferences separate from run state

The Mod Object reference allows an optional `config.lua` at the mod root; it returns a table of defaults. `SMODS.current_mod.config` exposes that table to the mod. A configuration tab in the Mods menu is a separate UI definition through `config_tab`; a config file alone does not create a useful settings screen. Decide which values can be changed during a run and which should require restart, then test the change, save, relaunch and load behavior. Run-specific Joker or card values belong on the card instance, usually under `card.ability.extra`, not in global mod preferences.

For error reports, capture the first Lua exception and the Lovely/Steamodded logs, the game and loader versions, enabled mod list, profile/save state and shortest reproduction steps. The logging reference and Lovely README identify the relevant log paths. Separate failures into discovery, entry-file execution, object registration, localization/art lookup, gameplay callback, persistence or source-patch match. Keep the original save backup from before mod installation so removing a mod does not make a test run irrecoverable.

Build a player-installable package

Make a release ZIP from the mod folder containing the Steamodded JSON metadata, `main.lua`, every required source file, localization, original assets and Lovely patch files if used. Install that exact ZIP into a clean mod profile with only declared requirements and walk the quick-start test again. Check a fresh run, a reload, all unlock/pool paths, a second mod that touches the same object, and a missing optional integration. Record the Balatro, Lovely and Steamodded versions actually tested; do not infer compatibility from a successful Lua syntax check.

For Thunderstore, the Steamodded packaging guide distinguishes Thunderstore's `manifest.json` from Steamodded's mod metadata: both may coexist. It requires Thunderstore files such as `manifest.json`, README and icon at the ZIP root and says Lovely files must also be at the root. It notes published package versions are immutable, so any fix needs a new version. Inspect the archive layout before upload, then install the published version as a user would and verify it loads. This page records the workflow; ModDocs has not uploaded or installed a Thunderstore package.

Version record and remaining checks

Keep a release record with date, game build, Lovely build, Steamodded release, mod version, declared dependencies, file hash, clean-install result, save/reload result, source-patch targets and known conflicts. Steamodded's release notes must be checked when moving that record to a new loader version. The overall guide remains **In progress** until the examples are run in-game and the packaging path is independently validated.

Sources and provenance

- Steamodded Atlas, Localization, Mod Object, Mod Metadata and Logging — SMODS Team references; checked 2026-10-12.
- Steamodded Thunderstore guide, release notes and Lovely README — package and compatibility references; checked 2026-10-12.

- This chapter is original ModDocs prose. It does not bundle Balatro, Lovely or Steamodded binaries, images or source text.