Page contents

- Establish a reproducible baseline
- Build the source project
- Declare loader metadata and dependencies
- Verify the built artifact
- Coverage still needed
- Sources and provenance

Offline copy of Loader, project and metadata .

Slay the Spire: loader, project and metadata

Documentation status: In progress. **Source checked:** 2026-10-06. **In-game validation:** not recorded. This chapter expands the preserved first-mod guide for the original Slay the Spire. The loader and API documentation below come from the ModTheSpire project and BaseMod project; these are separate projects with separate responsibilities.

Establish a reproducible baseline

ModTheSpire loads external mod JARs and applies code patches. BaseMod is itself a mod that supplies game-content hooks and a development console. Install a ModTheSpire release compatible with the game installation, create the game's `mods/` directory, and place the BaseMod JAR there. The ModTheSpire installation guide shows the launcher beside `desktop-1.0.jar`; its README documents Java 8 as the launcher route. The BaseMod README also states Java 8 and a ModTheSpire dependency. Keep the version identifiers from the actual files you install; a linked wiki example may predate the current release.

Start once with only BaseMod selected. Record whether the loader lists it, whether the game reaches the menu, and the first error if it fails. Then add your own JAR with a single initializer log line. This order distinguishes an environment failure from a card

implementation failure. Keep a disposable run and a backup of saves for subsequent console tests.

Build the source project

Use the BaseMod `testmod` source or the maintained BasicMod example as a structure reference. Preserve your own source tree, asset files, localization JSON, build configuration, and dependency versions. Add references to the installed game, ModTheSpire and BaseMod for compilation as the chosen example specifies. The ModTheSpire README lists JDK 8 and Maven for its build; check the chosen example's build file for the exact project command instead of assuming every mod has the same Maven goals.

The distributable is a JAR, but a source archive is the preservation copy. A typical layout separates Java packages from packaged resources:

```
src/main/java/<your package>/
  YourMod.java
  cards/
  patches/
src/main/resources/
  ModTheSpire.json
  <your mod's localization and art>
```

This tree is a planning example, not a requirement that those exact package names or folders register content. Inspect the resulting JAR: `ModTheSpire.json` must be at its root, while resource paths used by Java code must match the packaged paths. Do not redistribute the game's JAR in the mod package.

Declare loader metadata and dependencies

The ModInfo reference defines `ModTheSpire.json` at the JAR root. Its required fields are `modid` and `name`; `dependencies` lists mod IDs that load first and are required, while `optional_dependencies` loads present mods first without requiring them. A BaseMod-dependent mod can start with:

```
{
  "modid": "your_unique_modid",
  "name": "Your Mod Name",
  "author_list": ["Your Name"],
  "description": "A short description of the actual content.",
  "version": "0.1.0",
  "dependencies": ["basemod"]
}
```

Replace the placeholder identity and description before distribution. Use the project reference for `sts_version` or `mts_version` if you need a deliberate compatibility constraint; do not paste dates or minimum versions from an old example without checking the installed build. An optional integration still needs guarded code paths when its dependency is absent. The older `ModTheSpire.config` format remains for legacy compatibility, but the ModInfo page explicitly says not to use it for new mods.

Verify the built artifact

After packaging, inspect the JAR as a ZIP: confirm the JSON file is at the top level, the initializer class is included, and every referenced localization/art file is present. Place a fresh copy of that JAR in `mods/`, select it and BaseMod in the launcher, and confirm one startup log line before registering content. Repeat the test with only declared dependencies. A build can succeed while a missing resource, wrong mod ID, or incompatible installed loader makes the release unusable.

Coverage still needed

Exact IDE setup, operating-system paths, Workshop upload steps, cross-version compatibility, and reproducible clean-machine testing still need versioned walkthroughs. Continue with subscribers and patches for behavior and content registration.

Sources and provenance

- ModTheSpire README and installation wiki — loader installation and build requirements; checked 2026-10-06.
- ModInfo — metadata fields, dependency meaning, and legacy format; checked 2026-10-06.
- BaseMod README, testmod, and BasicMod — API requirement and project examples; checked 2026-10-06.
- This chapter is original ModDocs prose. It does not include game code, binaries, art or a copy of a third-party example project.