Page contents

- Complete the first card path
- Relics and powers
- Potions, events and characters
- Monsters and encounter ownership
- A repeatable content checklist
- Sources and limits

Offline copy of Cards and content families .

Slay the Spire: cards and content families

Documentation status: In progress. **Source checked:** 2026-10-06. **In-game validation:** not recorded. Start with the loader/project and subscriber/patch chapters. This page turns BaseMod's separate object references into a repeatable design and test sequence; examples must be compiled against the installed BaseMod version.

Complete the first card path

The BaseMod Custom Cards reference uses `CustomCard`, a namespaced ID, a packaged image, localized `CardStrings`, and `BaseMod.addCard` inside `receiveEditCards`. Before writing effects, make the card load with a unique ID such as `your_unique_modid:FirstCard` and confirm that the image path resolves inside the JAR. Register its strings in `receiveEditStrings` using `BaseMod.loadCustomStringsFile(CardStrings.class, path)` as shown by the Hooks reference. An ID mismatch between Java, strings and registration is a common reason a compiled card fails at load.

Build in this order: (1) constructor with cost, type, color, rarity and target; (2) base damage/block/magic values as needed; (3) `use` that queues a game action; (4) `upgrade` that changes the matching value and updates its description; (5) an independent copy path. The BaseMod card example demonstrates damage and

Vulnerable together, but start with one effect to isolate failures. Load the card through the development console, then play, upgrade, copy, save/reload and find it through the intended reward or pool. Console creation alone does not prove normal acquisition.

Relics and powers

The Custom Relics page places registration in `receiveEditRelics`. It distinguishes a shared/vanilla pool from a custom character's color pool through `addRelic` and `addRelicToCustomPool`; pick the intended pool explicitly. `RelicStrings` must load before the relic: the page warns that missing strings can crash startup. Check pickup, combat triggers, counters, description, save/reload and pool appearance separately. A relic art path or class alone does not insert it into a pool.

A power is a combat state object. Model its owner, amount, ID, localization and icon before adding behavior; queue an action that applies it from a card or relic. The BasicMod power examples show `PowerStrings`, amount changes, description updates and copying. Test stacking, reduction, removal, turn transitions and reload where persistence matters. Do not assume a power needs the same registration callback as a card or relic; its creation path is the effect or action that applies it.

Potions, events and characters

BaseMod's Custom Potions reference describes potion class, strings, registration and pool behavior. Trace potency, target selection, use effect, rendering and copy/upgrade behavior on the installed version. Validate both console access and normal potion rewards.

For events, the Custom Events guide registers the class with `BaseMod.addEvent` during `receivePostInitialize`. The event ID must agree with its localization, and an optional dungeon ID limits the pool. The newer `AddEventParams.Builder` supports conditions and replacement behavior. Test each dialogue branch, reward, leave path and save/reload; for an event with combat, use the guide's `PhasedEvent` path or its documented reward handling rather than assuming a simple dialogue class covers it.

A custom character is a larger package: Custom Characters and Custom Colors describe the character registration, card color, starter deck/relics, strings and visuals.

Validate new-run selection, starting content, character-specific reward pools, animations, death/restart and loading an existing save. Do not add a character as an incidental extension of the first-card example.

Monsters and encounter ownership

The BaseMod wiki includes Custom Monsters, but a monster class alone is not an encounter. Document where it is registered or injected, its encounter/dungeon pool, move state, intent, animations, death behavior and rewards. Run an encounter in a disposable save and confirm the same fight survives save/reload. This path may need direct patches or additional APIs depending on the desired placement; it is a separate implementation from `EditCards`.

A repeatable content checklist

For every family, keep a table in the mod repository with **ID**, Java class, localization key, asset path, registration callback, acquisition path, save behavior and tested build. Verify JAR contents first, then minimal-loader startup, then console creation, then normal gameplay, then cross-mod behavior. Record the first failing stage instead of rewriting unrelated components.

Sources and limits

- BaseMod Custom Cards, Custom Relics, Custom Potions, Custom Events, Custom Characters, Custom Monsters, and Hooks — project-maintained API examples; checked 2026-10-06.
- BasicMod powers — sample implementation path; checked 2026-10-06.
- This is an original workflow map. The object families beyond the first-card sequence need complete compiled examples and independent in-game validation before the guide can claim comprehensive tested coverage.