Page contents

- Consumables and Tarot, Planet, Spectral sets
- Starting decks and challenge rules
- Blinds and vouchers
- Editions, enhancements and seals
- Connect objects without key collisions
- Sources and limits

Offline copy of Consumables, decks, blinds and card modifiers .

Balatro: consumables, decks, blinds and card modifiers

Documentation status: In progress. **Sources checked:** 2026-10-12. **In-game validation:** not recorded. The first Joker guide demonstrates one scoring object, and the object-model chapter explains IDs and shared fields. This chapter maps the other requested Steamodded object families to their own registration and test paths. Use the installed Steamodded release and its matching game-object API for exact signatures.

Consumables and Tarot, Planet, Spectral sets

`SMODS.Consumable` needs a unique `key`, a `set`, and localization. The Consumable reference identifies the built-in sets as `'Tarot'`, `'Planet'` and `'Spectral'`; a modded consumable type uses its own set key. Define `can_use(self, card)` to control availability and `use(self, card, area, copier)` for the use effect. `calculate` is a separate callback for contexts where a held consumable participates in scoring or other events. Do not put a one-time use effect only in `calculate` and expect the Use button to run it.

For a test consumable, assign a namespaced key, text, cost and atlas frame, then implement a reversible effect on a disposable run. Confirm the item appears in the intended pool, can be selected from a pack, obeys its `can_use` conditions, performs

the effect once, and is removed or retained as intended. The reference's `select_card`, `keep_on_use`, `in_pool` and `on_select` methods describe those distinct phases. Test full consumable slots and save/reload, not just direct creation from a debug helper. A Planet card that modifies a poker hand needs a hand-level check as well as a collection entry.

Starting decks and challenge rules

`SMODS.Back` is the deck/back object used at run setup. Its API page documents `apply(self, back)` for starting modifiers, but says starting-deck card changes require events. Test the back on a fresh run with its intended stake and seed, then verify card count, starting money, hand/discard modifiers, and any assets or localization. A back that looks correct in selection can still fail when the run initializes.

`SMODS.Challenge` is a separate object, with `rules.custom`, `rules.modifiers`, starting Jokers and consumables described in the Challenge reference. Challenge names use `misc.challenge_names` localization. Treat a challenge as a controlled game-mode composition: validate each starting object key, rule value, UI text, ban or restriction, and win/fail path. Its custom rule table is not a replacement for a mod dependency declaration.

Blinds and vouchers

`SMODS.Blind` owns an encounter, including selection-time `set_blind`, calculation contexts and defeat/disable behavior in the Blind reference. The page explicitly says the active blind is `G.GAME.blind` inside its methods; the prototype `self` is not the active instance. Test normal, boss and skipped routes as applicable, score target changes, debuffs, save/reload and cleanup after a blind is defeated. An effect that remains after the blind ends is a lifecycle bug.

`SMODS.Voucher` belongs to the shop/progression path. Its reference uses full keys in `requires`; a modded voucher prerequisite therefore needs the class and mod prefix, not the short local key. Test appearance, purchase, unlock prerequisite, effect application, duplicate prevention and a resumed run. Use object-level `dependencies` only for optional cross-mod content; a loader-level requirement belongs in mod metadata.

Editions, enhancements and seals

These alter cards in different ways. An Edition can supply scoring values and a shader (or `shader = false`), weight, extra cost and per-card behavior. Its label has a localization location distinct from the full display name. Test whether it appears in shops/packs, how its score stacks with base card effects, and whether the shader survives different card sizes and save/reload.

An Enhancement can change a playing card's rank/suit semantics. Flags such as `replace_base_card`, `no_rank`, `no_suit` and `any_suit` affect hand evaluation as well as drawing. Test at least high-card and multi-card hands, scoring and non-scoring cards, copy effects, conversion, and challenge restrictions. A Seal has its own `calculate` and `get_p_dollars` methods; test both triggered score and end-of-round money where used. Seals are not simply another edition layer.

Connect objects without key collisions

For every family, keep a small register in the mod source: short key, full registered key, object class, localization category, atlas frame, acquisition/pool, dependency, and test case. Register a single object first, then add relationships: voucher prerequisite, consumable target, back starting content or challenge item. Use the full key when an API calls for it. When changing a vanilla object, read taking ownership and test with other mods that touch the same field.

Sources and limits

- Steamodded game objects, Consumable, Back, Challenge, Blind and Voucher — SMODS Team references; checked 2026-10-12.
- Edition, Enhancement and Seal — SMODS Team references; checked 2026-10-12.
- This is original ModDocs prose. The object families need compiled, in-game examples before this archive can claim tested coverage.