Page contents

- Read the entity-component tree
- Load and inspect an entity
- Separate a spell action from its held card
- Avoid false positives during testing
- Sources and limits

Offline copy of Entities, components, spells and wands .

Noita: entities, components, spells and wands

Documentation status: In progress. **Source checked:** 2026-10-06. **In-game validation:** not recorded. Start with the mod lifecycle chapter. This page describes the boundaries between an entity, a projectile, a spell action and the wand or inventory item that holds it, using the Noita Wiki's modding basics and custom spell guide.

Read the entity-component tree

An entity XML file defines an entity and its components. It may inherit a base entity, hold children, and have tags. The basics page uses a simple banner to show `VelocityComponent`, `SimplePhysicsComponent` and `SpriteComponent`. These each supply different behavior: a sprite path is not a physics body, and a physics component does not imply health or AI. Begin by inspecting a close vanilla example in the installed extracted data. Copy only the structure needed for an original mod-owned entity under `mods/<your-id>/files/`, then change one component value at a time.

Entity tags and component tags are different attributes (`tags` and `_tags` in the wiki reference). Give your own tags distinct names. The tag system reference notes limits on allocated tags; avoid generating a new unique tag for every entity instance. Use the installed `tools_modding/component_documentation.txt` and

`lua_api_documentation.txt` for exact component fields and API signatures, as the developer's Lua scripting pointer directs.

Load and inspect an entity

The Noita Lua API documents `EntityLoad(path, x, y)` for spawning XML, `EntityGetAllComponents` for inspection, and `EntityKill` for removal. Call them in a callback where the world and coordinates exist, such as after player spawn, rather than at top-level load time. Use a disposable save, a harmless test location and a mod-owned XML path. Check the return ID and the game's log; if nothing appears, inspect XML validity and every referenced image/script path before changing spawning logic. Remove the test entity or gate it so it is not spawned on every load.

Projectiles, enemies and items are different entity designs. A projectile needs its own movement, collision and lifetime behavior; an enemy adds damage/AI/death and encounter placement; an item needs pickup/inventory behavior. Use the closest vanilla XML as an API map, then replace its art and logic with your own. The custom enemy guide is a research lead, while a functioning enemy additionally needs spawn tables, intent/attack checks and save/load tests. An entity spawned by console or script has not yet been integrated into world generation.

Separate a spell action from its held card

The custom spell guide explains Noita's internal terminology: the action is the spell behavior; a card is the inventory/wand object that references an `action_id`. The installed `data/scripts/gun/gun_actions.lua` is the action list. Place a mod-owned Lua file, for example `mods/<your-id>/files/actions.lua`, and append it from `init.lua` using `ModLuaFileAppend` on `data/scripts/gun/gun_actions.lua`. Define a unique uppercase action ID and a test action using a known projectile path from the installed build. Check the reference's fields for type, sprite, mana, price, spawn levels/probabilities and uses rather than copying an old example wholesale.

Test the action in two steps: first confirm the action is present and invokes its effect; then confirm a wand can hold and cast the corresponding card. A spell appearing in a list is not proof that its projectile XML, sprite, translations or cast delay behave correctly. Test repeated casts, low mana, limited uses, modifiers and save/reload. If creating a custom wand, inspect how its entity, ability component, deck and sprite

connect in the installed game data. Do not assume adding an action also adds a wand or a world drop.

Avoid false positives during testing

A stale `data/` override can make the new entity appear even if the append hook failed. Temporarily disable other mods and remove unrelated overrides, then restart and inspect the log. Keep a test matrix with mod version, game build, enabled mods, entry callback, entity/action ID, expected result and observed result. Test new run and reload separately, because `OnPlayerSpawned` can run on both according to the Lua API callback reference.

Sources and limits

- Noita Wiki modding basics, Lua API, tag system, and custom spell guide — community-maintained technical references; checked 2026-10-06.
- Nolla Games Lua scripting pointer — directs authors to the installed API documentation; checked 2026-10-06.
- This page is original workflow guidance. It does not bundle extracted Noita data, a complete custom projectile, or an in-game-tested spell.