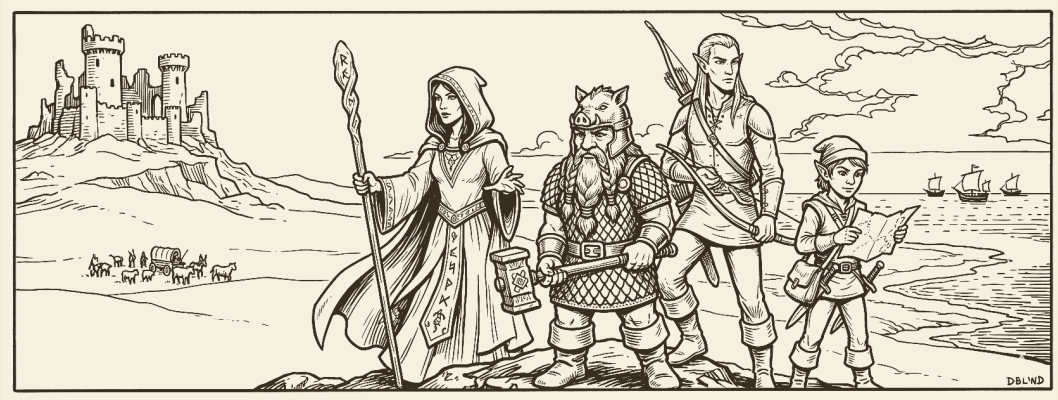
WILDERSTEP: INFINITE ABYSS



DUNGEON MASTER'S MANUAL

A Guide to Building Adventures in Wilderstep

The Author's Companion to the Player's Manual

Wilderstep: Infinite Abyss is not just a game — it is a platform for building your own adventures. Where the Player's Manual teaches you how to *play* Wilderstep, this manual teaches you how to *make* it: to design worlds, populate them with monsters and treasure, weave quests through their towns and dungeons, and hand the result to other players. This is the Dungeon Master's craft.

This manual is a living document. Right now it is deliberately sparse — the bones of a guide that will grow as the toolset matures and as we learn what authors most need explained. Expect it to fill in over time.

Contents

- Introduction
- What You Can Build
- How This Manual Is Organized
- Getting Started
- Entering Dungeon Master Mode
- Modules: How Content Is Organized
- Inheritance: Building on the Default Module
- The Building Blocks of an Adventure
- The Author's Loop: Edit, Test, Publish
- Where to Go Next

INTRODUCTION

Every adventure in Wilderstep is data. The monsters you fight, the maps you walk, the shops you barter in, the quests you chase, the very races and classes your heroes belong to — all of it lives in plain, editable records rather than being baked into the game's code. Dungeon Master mode is the toolset that lets you create and shape that data. If you have ever wanted to run players through a dungeon of your own design, this is where you do it.

You do not start from a blank page. Wilderstep ships with a complete, playable baseline — the **Default module** — containing a full catalog of races, classes, spells, items, monsters, and the rules that tie them together. Your adventure builds *on top of* that baseline: you inherit everything Default provides, then add, change, or replace only the pieces that make your story your own. A new author can have a working adventure in minutes, because the hard parts already exist.

What You Can Build

An adventure in Wilderstep is assembled from a handful of content types, each of which you can author:

- **Maps** — the overworld, towns, building interiors, and battle arenas players explore tile by tile.
- **Monsters** — the creatures that inhabit your world, with their stats, abilities, and on-hit effects.
- **Encounters** — curated groups of monsters that appear as a single fight, whether triggered by a quest or rolled at random while exploring.
- **Quests** — the threads that give players purpose: objectives, steps, and the rewards for completing them.
- **NPCs and dialog** — the people who populate your towns and give the world its voice.
- **Items, shops, and treasure** — the gear players find, buy, and loot.
- **The rules themselves** — races, classes, spells, abilities, and effects, all of which you can tune or extend.

You don't have to touch all of these to make something worth playing. A satisfying first adventure might be nothing more than a new town, a dungeon beneath it, a monster or two, and a quest to tie them together — everything else inherited, unchanged, from Default.

How This Manual Is Organized

The rest of this manual is, for now, a single **Getting Started** chapter that orients you in the toolset and the core concepts you'll rely on no matter what you build. As the manual grows, it will branch into focused guides — designing maps, balancing encounters, structuring quests, and publishing for others — each going deeper than this overview can. Until then, treat Getting Started as your map of the territory.

GETTING STARTED

Wilderstep's authoring toolset is built around one central idea: you assemble an adventure as a **module** that inherits from a shared baseline and overrides only what it needs. Understand that idea and the rest of the toolset falls into place. This chapter walks you from signing in, through the module and inheritance model, to the loop of editing, testing, and publishing your work.

Entering Dungeon Master Mode

Dungeon Master mode is unlocked by signing in to your account. Once signed in, the **Editor** becomes available from the landing page — this is Wilderstep's game development kit, where every piece of authorable content can be browsed and changed. The Editor is the home base for everything in this manual.

If you have only ever played Wilderstep, the Editor will feel like looking behind the curtain: the same races, classes, monsters, and maps you met as a player are all here as editable records, laid out in browsable tables you can sort, filter, and open.

Modules: How Content Is Organized

A **module** is a self-contained adventure — a named bundle of all the content types listed above. The game's play menu lists the modules a player can choose from; each one is a separate world to step into.

There are a few flavors of module, distinguished by their role:

- **Core** — the **Default module**. It holds the canonical record set every other module builds on. It is editable but is hidden from the play menu; you don't play Default directly, you play something built from it.
- **Playable** — a runnable adventure, shown in the play menu. This is what you are usually making.
- **Library** — an importable collection of reusable content (a pack of maps, a set of NPCs) that other modules can pull records from on demand, without inheriting the whole thing.

Every module declares its own identity and its relationships to others. The two relationships that matter most are `extends` — the single parent module it inherits from — and `uses` — an optional palette of library modules it can import individual records from.

Inheritance: Building on the Default Module

This is the most important concept in the toolset, so it's worth getting right.

When your module `extends` Default, it does **not** copy Default's content. Instead, at the moment a player loads your adventure, the game layers your module's records on top of Default's, **matching by**

record id:

- A record id that exists only in Default is **inherited** — your module uses Default's version as-is.
- A record id your module also defines **overrides** the inherited one — your version wins, completely.
- A record id that exists only in your module is **new** — it's added to the world.

The practical payoff is that you only ever author the *difference* between your adventure and the baseline. Want the Default goblin, but tougher? Override just the goblin. Want a brand-new boss? Add it as a new record. Everything you don't touch — hundreds of items, dozens of monsters, every spell and class — comes along for free, and stays current with the baseline.

The Editor surfaces this provenance directly: as you browse a content type, each record is badged as **inherited**, **overridden**, or **new**, so you always know whether you're looking at the baseline, your change to it, or something only you have. Editing an inherited record turns it into an override (a copy-on-write you own); reverting an override drops your copy and falls back to the inherited version.

A consequence worth internalizing early: because inheritance is resolved live at load time, your adventure tracks the parent. If the baseline changes, the inherited parts of your adventure change with it. That is usually exactly what you want — but if you ever need an adventure that is frozen and immune to baseline changes, the move is to override (own a copy of) every record it depends on, so nothing is left inherited.

The Building Blocks of an Adventure

Each content type is browsed and edited the same way — a table of records you can open and modify — but each plays a distinct role in an adventure:

- **Maps** are the stage. The overworld is the open world players cross; towns and interiors are the places they visit; arenas are the boards fights play out on. Maps reference the other content types — a town's shop points at items, a dungeon's rooms draw monsters from encounters.
- **Monsters** are individual creatures: hit points, armor, attacks, and special traits like regeneration, resistances, or on-hit effects (the kind of thing that lets a creature swallow a hero whole).
- **Encounters** bundle monsters into a fight. An encounter has a roster, a difficulty, and an *area* that determines which random-spawn pool it belongs to (dungeon, overworld, and so on). Encounters are used two ways: pinned to a quest step by id, or rolled at random from their area pool as players explore.
- **Quests** give the adventure shape. A quest is a sequence of steps — defeat a named encounter, recover an item, reach a place — each with conditions for completion and rewards for finishing.
- **NPCs and dialog** make the world feel inhabited: the townsfolk who hand out quests, sell goods, or simply have something to say.
- **Items** are everything players wield, wear, drink, and carry — and the stock that fills your shops and the loot that fills your chests.
- **Races, classes, spells, abilities, and effects** are the rules layer. Most adventures inherit these unchanged, but they're fully yours to retune or expand when your story calls for a new kind of hero or a new kind of magic.

You will rarely build all of these at once. A good first adventure leans heavily on what's inherited and adds just enough new content — a place, a foe, and a reason to go there — to feel like its own world.

The Author's Loop: Edit, Test, Publish

Authoring in Wilderstep is iterative. The rhythm looks like this:

1. **Edit.** Open a content type in the Editor and make a change — add a monster, place it in an encounter, write a quest step that hunts it. Your work-in-progress is held as a draft so you can experiment freely before committing.
2. **Test.** Play your adventure to see the change in context. The Editor includes tools for trying out battles and dungeons in isolation, so you can check an encounter's balance or a map's flow without replaying your whole adventure to reach it.
3. **Publish.** When a change is ready, publish it. Publishing writes your module's records out as the live version and refreshes the catalog so the adventure — and your update to it — is available to play.

Then you go back to step one. Most of building an adventure is small loops like this: a change, a playtest, a tweak.

Where to Go Next

For now, the best next step is to open the Editor and explore. Browse the Monsters table to see how a creature is defined; open an Encounter to see how monsters are grouped into a fight; trace a Quest from its first step to its reward. Seeing how the Default module is built is the fastest way to learn how to build your own — every adventure that ships with Wilderstep is made of exactly the same parts you have in front of you.

As this manual grows, this is where the deeper guides will live: designing maps, tuning encounters, scripting quests, and sharing finished adventures with other players. Until then — welcome behind the curtain. The world is yours to make.