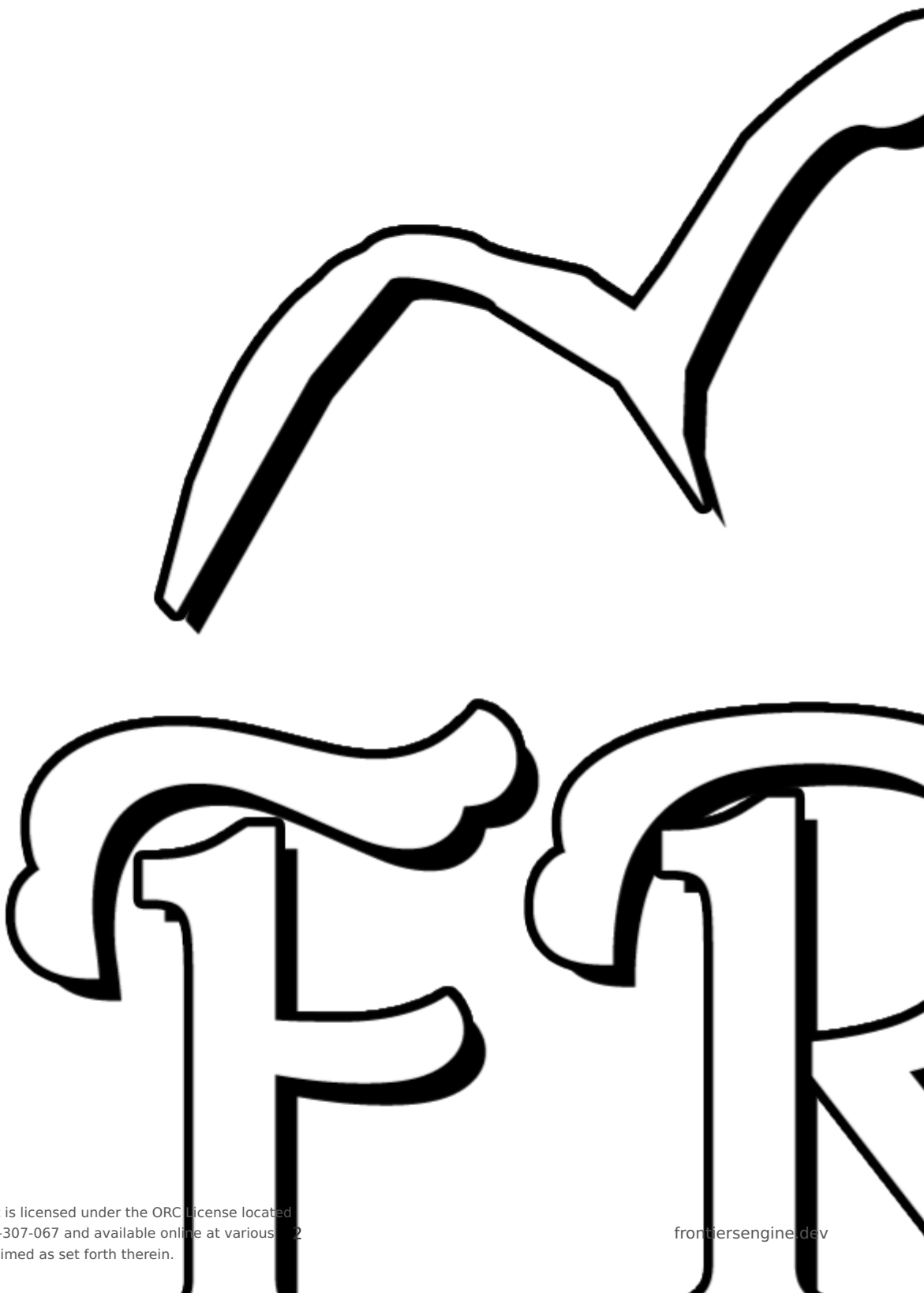

Beta 2026-2

This document is the **front door** of the Frontiers Tabletop Engine. It introduces the engine's design philosophy, walks through the broad strokes of how to design a tabletop roleplaying game, and explains how Frontiers fits into that work.

It pairs with the **Frontiers Overview**, the condensed working ruleset. The Overview covers what the rules are; this document covers how to design with them.

It is also a teaching document about TTRPG design as a craft, and about the questions every designer runs into when building a system, whether or not they start from Frontiers. Frontiers is opinionated, and its opinions are one workable set of answers among several. I have tried to be honest about that throughout.

If you want the rules, read the Frontiers Overview.

If you want the depth on a specific subsystem, read the relevant Design Documentation entry.

If you want to think about TTRPG design as a whole, and where Frontiers fits in, keep reading.

What Frontiers Is

Frontiers is a **modular tabletop roleplaying engine** designed to remove the need to reinvent common mechanical solutions when building a new TTRPG system.

On its own it is not a complete game. What it gives you is a **set of structural defaults** for the mechanical layer of a TTRPG: how dice work, how characters are built, how time and action flow, how resources are tracked, and how characters grow.

Designers can: - Adopt the entire engine as written and build a system on top of it - Take individual subsystems (the Resolution System, the Flow of Play) and ignore the rest - Modify any default with a Variant or Alternative the engine already provides - Replace any subsystem entirely with their own design

Frontiers is opinionated on purpose, and every choice in it was made for a stated reason. Those choices are meant to be argued with and overridden where your game needs something else.

What Frontiers Is Not

Some things this engine makes no attempt to do:

- **Worldbuilding.** Frontiers does not help you design a setting, a cosmology, a magic system's narrative logic, or a faction's politics. The engine is genre-neutral and assumes the designer is bringing their own world to the table.

- **Tone-setting beyond mechanics.** Frontiers can support a horror game or a heroic adventure game, but it doesn't tell you how to write one. The mechanical levers (lethality calibration, resource scarcity, Gradient weighting) influence tone, but the actual creative work of evoking a feeling is the designer's.
- **Adventure design.** Frontiers doesn't include adventures, encounters, or narrative structures beyond what's needed to run the mechanics. Adventure modules built on Frontiers exist as separate work.
- **Character art, layout, or aesthetic identity.** The engine is text and rules, and visual design remains the designer's territory.

For any of the above you will want to look elsewhere. Frontiers is a mechanical toolkit and nothing more.

What Other Engines and Frameworks Look Like

Frontiers is one of many TTRPG engines and frameworks in the modern indie space, and it draws on plenty of them. Anyone choosing a starting point has real options:

- **Powered by the Apocalypse** (Vincent Baker). A framework rather than a strict engine. Tightly couples Moves to specific roll outcomes. Excellent for narrative-forward systems.
- **Forged in the Dark** (John Harper). Built around action verbs, position/effect, and Stress as a meta-resource. Excellent for heist, ensemble, and consequence-driven systems.
- **Cypher System** (Monte Cook Games). A flexible framework for building cypher-style games. Heavy on player-facing flexibility.
- **Year Zero Engine** (Free League). Used in Mutant: Year Zero, ALIEN, and others. Stat + Skill dice pools with a push mechanic.
- **Open D6** (West End Games). A classic d6-based generic system.
- **Fudge / Fate** (Evil Hat). Aspect-based narrative systems.
- **Open Legend** (Open Legend RPG). Attribute-based open license system.

Frontiers sits in the same space and overlaps in ambition with several of them. What separates it is the particular set of opinions it holds: the two-die Resolution system, the Reference Level maths that unifies Abilities and Equipment, the three-Category Skill structure, and the Vector Mark advancement model.

If you read through Frontiers and find its opinions do not match what you want to build, one of the engines above will probably suit you better. That is a perfectly ordinary outcome and worth acting on early.

This section covers the broad strokes of designing a TTRPG, regardless of whether Frontiers is the starting point. Each phase points to the relevant Frontiers Design Documentation when applicable, but the phases themselves apply to almost any system.

Treat the phases below as a rough roadmap rather than a checklist. Designers loop back constantly as later phases turn up constraints the earlier ones missed, but this order is a reasonable place to begin.

Phase 1 — Before You Design Mechanics

Before any dice get rolled or any stats get listed, you should be able to answer some basic questions about the game. None of them are mechanical questions, and the answers shape every mechanical decision that follows.

The questions worth answering before mechanics:

- **What is this game for?** Is it a heist game? A horror game? A dungeon crawl? A long-form character drama? Each tone wants different mechanics.
- **Who is the audience?** New players? Veterans? A specific subculture? A mass-market crowd? Different audiences need different complexity levels.
- **How long should a session be?** A 2-hour one-shot system has very different mechanical needs than a 20-session campaign system.
- **What does play actually look like at the table?** Mostly conversation? Mostly tactical maps? Mostly dice rolling? Mostly free-form description?
- **What feeling do you want players to leave with?** Triumph? Dread? Catharsis? Bewilderment? Camaraderie? Mechanics should reinforce the target feeling.

Frontiers is no help at all in this phase; it is entirely designer work, and it repays doing carefully before you touch any mechanics. Rough mechanics that fit the tone will serve you better than polished ones pulling against it.

Phase 2 — The Mechanical Spine: How Resolution Works

The first real mechanical decision is **how the game resolves uncertainty**, since that is the core loop everything else gets built around. A resolution system that works gives the rest of the design something stable to sit on, and one that does not will have every other subsystem fighting it for the lifetime of the project.

The resolution system answers basic questions: - What's the dice mechanic? d20? 2d6? Dice pool? Single roll-under? - Are results binary (success/fail) or gradated (success with cost, partial success, etc.)? - How do skill, attribute, and difficulty interact mathematically? - What does a "roll" feel like at the table?

Almost every other subsystem ends up passing through resolution, from combat and skill checks through to social conflict.

> **In Frontiers:** Resolution uses two dice. A d20 Resolution Die for success or failure, and a d10 Gradient Die for texture. Frontiers is opinionated here: whether you succeed and how that

> success manifests are two separate questions, so they get two separate dice. For full reasoning, Variants, and Alternatives, see The Resolution System design doc.

Other systems handle resolution very differently. PbtA games use 2d6 with three outcome bands. Mothership uses d100 roll-under against percentile stats. Mörk Borg uses d20 against modifier-adjusted DRs. These are all defensible; they simply hold different opinions about what a roll should feel like at the table.

Phase 3 — The Structure of Play: How Time and Action Work

Once resolution is decided, the next major question is **how time flows in the game**. Specifically:

- How does the game shift between casual exploration and high-pressure conflict?
- What does a single "turn" mean in terms of in-fiction time?
- How is action economy handled? One action per turn? Multiple? Action points?
- How does positioning work? Grid-based? Zone-based? Narrative?
- What resources get tracked during play, and how often do they refresh?

Most games run at least two modes: a low-pressure one for conversation and exploration, and a high-pressure one for combat or tense scenes. Where you draw the boundary between them, and how cleanly the rules carry a table across it, is among the more consequential choices a system makes.

> **In Frontiers:** Two modes (Free Play and Encounter Play) with one consistent grammar. Encounter Play uses a 3 ✦Action Point economy with separate ⇉Movement Points and ⚡Energy as a tactical exertion currency. For full reasoning, Variants, and Alternatives, see the Flow of Play design doc.

Other approaches: PbtA games dissolve the line between modes entirely (everything is conversation that triggers Moves). Pathfinder 2nd Edition uses a sharply delineated three-action economy in combat. Blades in the Dark uses Free Play, Score and Downtime as three distinct modes. Which structure suits you depends on the tempo you are after.

Phase 4 — Character Numbers: What the Sheet Tracks

With resolution and play flow established, the designer can start asking what each character actually is, mechanically. This is the layer of stats, derived values, and tracked resources.

Questions worth answering:

- How many core stats? Few (3-4) for simplicity, or many (6+) for differentiation?
- What scale do the stats use? 1-5? 1-10? 0-100?
- What derived values come from the stats? HP? Defense? Initiative?
- What gets tracked during play? Health alone? Multiple damage tracks? Stress? Mana?

- What does the character sheet look like physically?

The character sheet is the interface between the player and the system. Seven clear values feel very different in the hand from thirty cluttered ones, even where the underlying maths is identical.

> **In Frontiers:** Four Attributes (❖Vigor, Finesse, Acuity, ☼Resolve) on a 1-10 scale, with ten Derivatives that include two parallel damage tracks (⊗Wounds for physical, ⊖Fatigue for mental). Defense formula is "5 + highest of the relevant Modifier pair + Equipment." For full reasoning, Variants, and Alternatives, see the Attributes & Derivatives design doc.

Other approaches range widely. Lasers & Feelings characters have a single number. Call of Cthulhu characters have eight characteristics on a 0-100 scale plus dozens of skills. ALIEN RPG uses four attributes and twelve skills, and tracks Health, Stress, Air and Radiation as separate survival pressures. Each suits the game it was built for.

Phase 5 — Differentiation: How Characters Differ Mechanically

Once the baseline numbers are settled, the question becomes how characters distinguish themselves from one another, since two characters with identical stats should still be able to feel mechanically different. That work is handled by three broad sub-layers:

- **Skills or Training** — what the character has practiced
- **Abilities or Powers** — what the character can do beyond baseline
- **Equipment** — what the character carries that affects play

These three sub-layers can be designed independently or fused. Some games (Blades in the Dark) blur the line between attributes and skills. Some (Cypher System) fuse abilities and equipment into "cyphers." Some (Pathfinder) keep them rigorously separate.

> **In Frontiers:** Skills are binary (have or don't have) and grant Favor on the roll when applicable, layered with three Skill Categories (Body / Mind / Voice) that grant +1 to +5 modifiers. Abilities have Levels and use the Reference Level rule to determine ⚡Energy cost. Equipment uses the same Reference Level rule, with three structural states (Equipped, Activated, Consumable). For full reasoning, see the Skills, Abilities, and Equipment design docs.

Whichever sub-layer you load the most differentiation onto ends up telling players what kind of game they are in: leaning on skills reads as investigative, leaning on abilities reads as powered, and leaning on equipment reads as gear-driven.

Phase 6 — Identity: How Characters Differ Narratively

The mechanical differentiation work is paired with narrative differentiation. A character isn't just a stat block. They have an origin, a history, a current vocation. The system can express these through identity layers:

- **What the character is** (species, lineage, supernatural status, biological inheritance)
- **What the character did** (profession, lived history, formative experience)
- **What the character does** (active vocation, class, archetype)

Some games use all three (D&D 5e: Race + Background + Class). Some use one (PbtA: Playbook). Some use none (Lasers & Feelings: a single descriptive line).

> **In Frontiers:** Three identity layers (Origin, Background, Archetype) with open mechanical contributions. Each layer has a clear philosophical purpose, but what each grants is system-defined. Vector Mark advancement (3 Marks per Level, GM-granted at narrative milestones) ties character growth to story. Multi-Archetype is permissive by default. For full reasoning, see the Characters design doc.

The choice here shapes character creation length and the feel of progression. A heavily layered identity system makes characters feel rich but slows creation. A single-layer system makes characters feel focused but narrow.

Phase 7 — Iteration and Polish

The last phase is the longest. Once a system has all its phases drafted, the designer needs to:

- **Playtest.** Play sessions, record what works and what doesn't, iterate.
- **Get external feedback.** Other people's play experience reveals what the designer can't see from the inside.
- **Polish the rules text.** Clarity, consistency, terminology, edge cases.
- **Lay out the document.** Sheet design, visual hierarchy, indexing.
- **Decide on distribution.** Free PDF? Print run? Crowdfunding?

Frontiers does not help with this phase either, but it does try to make iteration easier by being explicit about what each subsystem is doing and why. When a designer playtest reveals a problem, the Design Documentation should help them understand which lever to adjust.

The Frontiers Design Documentation set covers each major subsystem in depth. Each Design Doc follows the same three-part structure:

- **Part One — Teaching.** A general overview of the design space for that subsystem. Not Frontiers-specific. Explains how different games approach the same questions and what tradeoffs are involved.
- **Part Two — Frontiers' Answer.** The specific choices Frontiers made, with reasoning. Why this approach, what it costs, what it buys.
- **Part Three — The Working Ruleset.** The actual mechanics, written in implementation form.

Each doc also includes:

- **Tuning the Feel** — design knobs that adjust the feel without breaking the engine
- **Edge Cases** — clarifying rulings for unusual situations
- **Variants** — modifications to the engine that change implementation without changing philosophy
- **Alternatives** — replacements that change the engine's philosophy and require recalibration of related systems
- **A Note to System Designers** — closing reflection on the subsystem's role in the engine

A designer reading top-to-bottom gets the full design conversation: what the field looks like, how Frontiers chose, what the rules actually are, how to adjust them, and how to replace them.

A designer who just wants the rules can skip to Part Three. A designer who wants to modify a subsystem should read all of it.

Variants vs Alternatives — The Convention

Frontiers uses a deliberate vocabulary distinction:

- A **Variant** preserves the engine's core philosophy but changes its implementation. Replacing 3 ✦AP per Turn with 2 ✦AP is a Variant: the action economy still works the same way, just at a different scale.
- An **Alternative** replaces a subsystem's philosophy with a different one. Replacing the entire two-die Gradient Resolution with a single-die success-band system is an Alternative: the way the engine answers the underlying design question has fundamentally changed.

The distinction matters because Variants are usually drop-in replacements, while Alternatives require recalibrating other subsystems that depended on the original philosophy.

When in doubt, lean toward calling something an Alternative. Cleaner labeling helps the designer know what they're committing to.

This section covers the practical work of building a system on top of Frontiers.

A Sample Checklist

If a designer wants a step-by-step path through a Frontiers-based design, this is a reasonable order:

1. **Decide your game's tone, audience, and length.** Don't touch mechanics yet.
2. **Read the Frontiers Overview.** Get a feel for what Frontiers gives you out of the box.
3. **Decide which Frontiers subsystems you'll adopt as-is.** For most designers, the Resolution System and Flow of Play are good starting defaults.
4. **Identify which subsystems you want to modify or replace.** Read the relevant Design Doc for each. Pick a Variant or Alternative if one fits, or sketch your own.
5. **Define your Skill list.** Frontiers gives you the Category structure but not specific Skills. Pick Skills that fit your genre.
6. **Define your Origins, Backgrounds, and Archetypes.** These are the most genre-specific layer. Use the templates in the Characters design doc as starting points.
7. **Define your Abilities.** Use the universal anatomy. Decide which are Static vs Dynamic.
8. **Define your Equipment.** Build out weapons, armor, tools, Enhanced items as needed.
9. **Build your character sheet.** Reflect what your system actually tracks.
10. **Playtest. Iterate. Polish.**

This is one path. Plenty of designers will work in a different order, especially if they have a strong vision for one subsystem (Origins, Equipment, Abilities) and want to design around it.

Modules

Frontiers supports **modules**, supplemental rule packages that add specific functionality on top of the engine. A module might cover:

- A specific genre's needs (a Sci-Fi Module, a Horror Module)
- A subsystem the engine doesn't include by default (a Carrying Capacity Module, a Vehicle Combat Module, a Magic System Module)
- A house-rule package that a designer wants to share

Modules attach to Frontiers cleanly because the engine's core is small. A module declaring "this assumes Frontiers Beta 2026-2 with default Resolution and Flow of Play" can be picked up by any system meeting those requirements.

Some example modules you could draw inspiration from include:

- **Basic D20-Combat** by Jayke Paver
- **Slot-Based Carrying Capacity** by Jayke Paver

Designers are encouraged to publish their own modules. The repository is intended to grow as the community contributes. Frontiers treats itself as an open-source starting point for tabletop design, not a finished product, and the modules built on top of it are part of the same ecosystem.

Licensing

Frontiers is released under the **Open RPG Creative (ORC) License**.

The short version:

- You can use Frontiers content — rules text, mechanics, and the Starter Set settings — in your own systems and modules, including commercial ones.
- You must include the four ORC notices in your published work. They are set out in full on the License page.
- Name the exact version of Frontiers you built against.
- You may not present a fork or modified version as the official continuation of Frontiers.
- The Frontiers brand identity — logo, wordmark, and trade dress — is **Reserved Material** under the ORC License. The rules and the settings are not.

For the full notice requirements and what is and is not covered, see the License. For the name and logo, see the Trademark & Usage Policy.

If a designer is using a different engine as their starting point, they should check that engine's license. Most modern indie engines (PbtA, FitD, Cypher, Year Zero, Open D6, Fate, Open Legend) have similar open-license arrangements, but each has its own attribution requirements.

Contributing to the Frontiers Repository

Frontiers grows through iteration and community work. Designers who build modules, expansions, or improvements are encouraged to share them publicly under the ORC License, and to attribute their work clearly.

If a designer is interested in contributing modules, collaborating on expansions, or providing structured feedback to the Frontiers team, contact: **contact@jaykepaver.me**

Frontiers is an opinionated set of starting points. Every default in the engine reflects a deliberate decision about what kind of mechanical experience is being offered. Those defaults are not commandments. They're a starting position that designers can accept, modify, or replace as they build.

The engine grew out of trying to remove the parts of TTRPG design that get reinvented every time someone makes a new system: how dice work, how characters are built, how time and action flow, how progression happens. By standardizing those, Frontiers tries to give designers more time for the parts of TTRPG design that should be reinvented every time: the world, the tone, the specific feel of a particular game.

If a designer reads through the Design Documentation and finds Frontiers' opinions don't match what they want to build, that's a useful result. It means the designer has clarity about what they actually want, and they can use a different engine or build from scratch with that clarity in hand. Either way, the design work moves forward.

If Frontiers' opinions do match what the designer wants, the engine should make the rest of the work easier. That's what it's for.

Now go design.

For the condensed working ruleset, see the **Frontiers Overview**. For the full Design Documentation set, see the **Design Documents collection**.

For the full license terms, see the **License**. Frontiers Tabletop SRD. Created by Jayke Paver. Released under the Open RPG Creative (ORC) License.