Author: Jayke Paver
Module Type: Additive Module
Compatible With: Frontiers Beta 2026-2 and any system using the default Gradient Resolution and Flow of Play rules.
Version: Beta 2026-2 Revision
Original Release: February 13, 2026
Current Revision: April 24, 2026

This document is a **modular addition** to the Frontiers Engine.

The core Frontiers rules do not assume combat as a required pillar of play. A system built on Frontiers may focus on exploration, politics, survival, intrigue, social maneuvering, or entirely nonviolent interaction. Nothing in the engine mandates structured combat.

This module exists for systems that choose to include it.

The Basic Combat Module provides an endorsed, example implementation of structured conflict using the existing engine architecture. It is not a core requirement. It does not redefine baseline mechanics. It does not alter the Gradient Resolution System, the Flow of Play structure, or the Attributes and Derivatives framework. It layers on top of them.

In other words, this is not "how Frontiers does combat." This is "a way Frontiers can do combat."

This module serves three purposes:

First, it offers a clear and adaptable framework for resolving violent or forceful conflict using the existing engine structure.

Second, it demonstrates how a subsystem attaches cleanly to the Frontiers Engine without modifying its foundation.

Third, it provides a stable reference implementation that system designers may adopt, modify, expand, or replace.

Use of this module is optional. It is supported and endorsed, but not mandatory.

Systems built on Frontiers may:

- Adopt this module as written
- Modify portions of it
- Expand it with additional combat layers
- Replace it entirely with a different combat framework
- Omit structured combat altogether

This module assumes use of the default Gradient Resolution and Encounter Play rules. If a system alters those foundational systems, adjustments may be required.

Combat in Frontiers is not a separate engine. It is a deliberate application of the existing one under pressure.

Combat in Frontiers always occurs within **Encounter Play**.

This means combat inherits all timing and structure from the Flow of Play system:

- Time is divided into Rounds
- Participants act in Turns
- Actions cost ✦Action Points
- Movement uses ⇉Movement Points
- ➤Initiative determines order (higher acts first)

This module does not redefine any of those mechanics. Combat is simply a context in which those mechanics are applied under heightened stakes.

When an encounter escalates into violence or structured forceful opposition, the game shifts into Encounter Play. From that point forward, all standard Encounter rules apply as written in the core engine. Combat does not introduce a new timing layer. It intensifies the existing one.

Combat in this module rests on five principles.

First, **attacks are simply ✦Actions**. They follow the same declaration and resolution structure as any other mechanically significant effort in Encounter Play.

Second, **Defense values function as Difficulty Ratings**. An attack compares its Resolution result against either Physical Defense or Mental Defense, depending on the nature of the harm.

Third, **success and harm are distinct**. The Resolution Roll determines whether an attack connects. Damage determines how much harm is inflicted. These are related, but they are not the same question.

Fourth, **Gradient shapes the result, not the outcome**. A positive or negative Gradient does not reverse a hit or a miss. It determines whether the result is a normal exchange, a graze, or a critical strike. Most attacks resolve as standard hits or misses; the extremes of the Gradient bell curve are where combat takes on dramatic weight.

Fifth, **pressure compounds**. The action economy permits two attacks per Turn without strain, but pushing for a third or stumbling on a follow-up swing can generate ⊖Fatigue. The engine's existing

harm tracks (⊗Wounds for incoming physical damage, ⊖Fatigue for self-imposed strain) handle the pressure.

This approach keeps combat fully integrated with the core design philosophy of the Frontiers Engine rather than introducing a separate mechanical identity.

An attack is a standard ✦Action performed during Encounter Play.

Unless otherwise stated, an attack costs **1 ✦AP**.

When a participant chooses to attack, the following procedure is used.

Step 1 — Declare Intent

The attacker declares:

- The target
- The method of attack (the weapon, Ability, or improvised approach being used)
- The general approach (overhead strike, careful aim, psychic intrusion, etc.)

Step 2 — Determine Governing Attribute

As with all Resolution Rolls, the task determines the governing Attribute. A heavy overhead strike might use ❖Vigor. A precise ranged shot might use Finesse. A psychic intrusion might rely on Acuity or ☼Resolve. The Game Master confirms the final Attribute selection.

Step 3 — Identify the Defense

The attacker identifies which Defense the attack targets:

- **Physical harm** compares against Physical Defense
- **Psychological, emotional, or cognitive harm** compares against Mental Defense

These Defense values function as Difficulty Ratings.

Step 4 — Roll the Attack

The base attack roll is:

> **Attack Roll = d20 + governing Attribute Modifier + Weapon bonus (if any) + situational modifiers**

If the system layers in additional modifiers (such as Skill Category modifiers, Skill Favor, or other system-defined bonuses), apply them per that system's rules. The base math above is the engine-level minimum.

If the **Modified Result ≥ target's Defense**, the attack is a hit. If the **Modified Result < target's Defense**, the attack is a miss.

There are no automatic successes or failures unless a system built on Frontiers explicitly introduces them. The final modified result compared to Defense determines the binary outcome.

Step 5 — Roll Gradient

After the hit/miss is determined, roll the **Gradient Die (1d10)** to determine how the result manifests.

The interpretation of the Gradient result depends on whether the attack hit or missed. (See the next section, **Gradient in Combat**, for the full table.)

Step 6 — Apply Damage and Effects

If the attack hits, damage is calculated based on the Gradient result and applied to the target's ✚HP.

If the attack misses, apply any narrative or mechanical consequence determined by the Gradient result.

Combat is the most common context where Gradient produces sharp mechanical effects. Most attacks resolve as standard hits or misses; the bell-curve extremes (Major Negative, Major Positive) are what create the dramatic moments.

Use the following interpretation table for the default combat Gradient:

Gradient Result	If the Attack Hit	If the Attack Missed
1 — Major Negative	**Graze** — half damage (round down)	**Critical Miss** — see below
2-3 — Minor Negative	Normal hit	Normal miss
4-7 — Standard	Normal hit	Normal miss
8-9 — Minor Positive	Normal hit	Normal miss
10 — Major Positive	**Critical Hit** — double damage	**Graze** — half damage (round down)

The bell curve produces standard hits and misses 80% of the time when Gradient is rolled. Major bands at the extremes produce the dramatic outcomes: a Graze (the attack barely connects), a Critical Hit (a clean and devastating strike), or a Critical Miss (a serious failure that creates an opportunity for the target).

Critical Hit

A Critical Hit deals **double damage**.

When doubling damage: - Roll the standard damage dice - Double the resulting total

(Alternatively, the table may roll the standard number of damage dice twice and total the results. Both methods are mathematically identical in the long run; pick the one the table prefers.)

Additional bonuses gained specifically from a Critical Hit (Abilities, Equipment effects) are not doubled unless they explicitly state so.

Critical Miss

A Critical Miss is a serious mechanical failure beyond a normal miss. The default engine does not assign a specific consequence to a Critical Miss; the consequence is **system-defined**.

Common system implementations include: - The attacker provokes a free Reaction from the target (if the target has ✦AP available) - The attacker drops or fumbles their weapon - The attacker exposes themselves and grants Favor to the target's next attack - The attacker triggers a Multi-Attack Fatigue check (see **Multiple Attacks per Turn**, below)

A system designer should pick one consistent interpretation and apply it across the system. The engine deliberately leaves this open because the right answer changes depending on the system's tone (gritty horror wants serious consequences; heroic fantasy wants only minor narrative complications).

Graze

A Graze is a successful or near-successful attack that lands with reduced effect. A Graze deals **half damage (round down)** and applies no other on-hit effects (no triggered conditions, no riders).

Graze can occur on either a Hit (with Major Negative Gradient) or a Miss (with Major Positive Gradient). In both cases, the mechanical effect is the same: half damage, no riders.

Damage represents reduction of a target's ✚Health Pool.

Weapons, Abilities, and harmful effects define their own damage expressions. This module does not prescribe specific dice sizes or damage tables. Instead, it establishes the structural rules for how damage interacts with the engine.

Damage Calculation

The default damage formula is:

> **Damage = weapon damage dice (or Ability damage)**

By default, no Attribute Modifier is added to damage. Damage comes from the weapon or effect itself.

Some system designers may want to add Attribute Modifiers to damage. This is supported as an extension to the module (see **Optional Extensions**, below) but is not part of the default engine layer. Adding Attribute Modifiers to damage significantly shifts lethality and pacing, and should be a deliberate choice.

Weapon Damage Specification

A weapon should clearly state:

- Its damage dice (e.g., 1d6, 1d8, 2d6)
- Its range in Units
- Any Competency required to use it (e.g., Light Weapons, Heavy Weapons)
- Any traits or special properties

Damage values, dice sizes, and scaling are system-defined. The engine does not prescribe damage tables.

Applying Damage

After damage is rolled, the total is subtracted from the target's ✚Health Pool.

If this reduction brings the target to 0 ✚HP: - The target gains 1 ⊗Wound and falls **Unconscious** (per the default rules in Flow of Play) - If the new Wound would exceed the target's Maximum ⊗Wounds, the target dies (per the engine's death condition)

This module does not alter the Wound system, the death threshold, or recovery mechanics.

A character may attempt up to **2 attacks per Turn** without penalty, paying the standard ✦AP cost for each.

A **3rd attack** within the same Turn may be made, but introduces strain. If the 3rd attack misses, the attacker gains **1 ⊖Fatigue**.

In addition, any **Critical Miss after the 1st attack** in the same Turn also gains the attacker **1 ⊖Fatigue**.

This rule mirrors the engine's broader pressure philosophy: pushing past your competence creates strain. Wounds come from things hitting you. Fatigue comes from things you do to yourself.

A character who has reached their Maximum ⊖Fatigue cannot make additional attacks until they recover.

Multi-Attack Examples

Example 1. A character with 3 ✦AP attacks once (1 AP), attacks again (1 AP), then attacks a third time (1 AP). The 3rd attack is a hit. No Fatigue gained.

Example 2. Same character, same sequence. The 3rd attack misses. The attacker gains 1 ⊖Fatigue.

Example 3. A character makes their 1st attack and rolls a Critical Miss. No Fatigue gained (Critical Miss on the 1st attack of the Turn does not trigger the rule).

Example 4. A character makes their 1st attack (hit), then their 2nd attack rolls a Critical Miss. The attacker gains 1 ⊖Fatigue.

Example 5. A character makes their 1st attack (hit), 2nd attack (hit), 3rd attack rolls a Critical Miss. The attacker gains 2 ⊖Fatigue (1 for the 3rd attack missing, 1 for the Critical Miss after the 1st attack). Both effects stack.

Reactions are effects triggered outside a participant's Turn.

Reactions cost ✦AP from the participant's current pool, which persists until their next Turn refresh. (See the Flow of Play for the full Reaction rules.)

Common combat-relevant Reactions include:

- **Counterstrike** (Ability) — react to an adjacent enemy's attack with a melee strike of your own
- **Parry** (Ability) — react to an incoming melee attack to grant yourself a temporary +2 to Physical Defense against that attack
- **Disengage** (Ability) — react to an enemy moving away to follow them, maintaining engagement

This module does not define the specific Reaction Abilities a system uses. Those are defined by the Abilities the system grants its characters. (See the Abilities design doc.)

The module does establish that Reactions are integral to combat tactics. Players who spend all 3 ✦AP on their Turn cannot React until their next Turn refresh. Holding back AP is a tactical choice.

Combat positioning uses □Units (1 □Unit = 5 feet, or 1 grid square).

Weapons specify their effective range. The default range bands are:

Range Band	Distance	Examples
Melee	1 □Unit (adjacent)	Daggers, fists, short swords
Reach	2 □Units	Polearms, whips, long spears
Short	up to 4 □Units	Thrown daggers, hand pistols
Medium	up to 12 □Units	Bows, crossbows, longarms
Long	up to 24 □Units	Sniper rifles, longbows at extreme range

A weapon used outside its effective range is either ineffective or imposes Disfavor on the attack roll, depending on system design.

Systems built on this module may define their own range bands. The defaults above exist as a starting point. The key engine principle is that range is measured in □Units and weapons should clearly state their range.

Cover and concealment modify how attacks resolve against a defender.

Cover

Cover represents physical material between the attacker and the target that interrupts incoming attacks.

Cover Type	Effect
Partial Cover	Target gains +2 to relevant Defense against ranged attacks
Full Cover	Target cannot be hit by ranged attacks; attacker must move, change angle, or use Abilities that bypass cover

Cover applies to ranged attacks. Melee attacks are generally unaffected by cover unless the target is fully behind a barrier the attacker cannot reach across.

Concealment

Concealment represents conditions that obscure the target without physically blocking the attack: darkness, fog, smoke, dim lighting, illusion.

A target with concealment imposes **Disfavor on the attacker's roll**.

Concealment and cover can stack. A target in dim lighting behind a low wall benefits from both.

Flanking

When two allies attack the same target from roughly opposite sides (or, on a grid, from opposite squares), both attackers gain **+2 to their attack rolls** against that target.

The GM determines flanking when not on a grid, based on the relative positioning of the attackers and target.

Flanking is a positional reward for tactical movement. Characters who can maneuver to set up flanks gain a small consistent bonus that doesn't escalate into critical-hit territory but rewards good play.

Not all combat interactions are resolved against static Defense values.

When two participants directly oppose one another in a struggle (such as grappling, disarming, overpowering, or resisting mental domination), the interaction may use **Contested Resolution**.

In a Contested Combat exchange: - Both participants roll a Resolution Die - Add their relevant Attribute Modifier and any situational modifiers - The higher Modified Result wins the contest

If degree matters, one or both participants may roll Gradient to determine how decisively the exchange resolves.

Contested combat follows the same structure as other contested interactions in the engine. No additional combat-specific framework is required.

The Multi-Attack Fatigue rule does not apply to Contested Combat; a single contested exchange is one ✦Action, regardless of the back-and-forth in the fiction.

Abilities operate exactly as defined in the Abilities design doc.

A combat-oriented Ability may: - Require a Resolution Roll to determine effect - Apply damage directly without a roll - Compare a defined value against a target's Defense - Impose narrative or mechanical effects without rolling to hit

The presence of combat does not alter the universal Ability structure.

If an Ability compares against Defense without using a standard Resolution Roll, it should clearly state how its effect scales when the target's Defense exceeds or falls below its defined threshold. This scaling remains consistent with the principle that Defense functions as a passive Difficulty Rating.

All ✦AP costs, ⚡Energy costs, Overspending rules, and timing constraints from the core engine remain unchanged in combat.

The Multi-Attack Fatigue rule applies to attacks made via Abilities that involve attack rolls, the same way it applies to weapon attacks. An Ability that grants two attacks in a single ✦Action counts as two attacks for the purpose of Multi-Attack Fatigue.

A participant is removed from active combat when they are rendered unable to act.

This most commonly occurs when:

- Their ✚Health Pool is reduced to 0 and they gain a ⊗Wound (becoming Unconscious per Flow of Play rules)
- They exceed their Maximum ⊗Wounds (death condition)
- They exceed their Maximum ⊖Fatigue (Unconscious until Fatigue drops below maximum)
- A specific effect renders them incapacitated

This module does not introduce additional defeat states such as death saves, bleed-out timers, or stabilization subsystems beyond what the Flow of Play already provides. Designers who wish to include such mechanics may create supplemental modules that build on this one.

The Flow of Play **Stabilize Action** (an adjacent ally spending 2 ✦AP to restore an Unconscious character to 1 ✚HP) applies normally during combat.

The auto-revive rule (Unconscious characters at Encounter end reawaken at 1 ✚HP if an ally can reach them) also applies normally.

Recovery from harm follows the standard Downtime rules. Combat does not alter recovery pacing unless another module specifies otherwise.

Equipment interacts with combat through damage expressions, Defense bonuses, range definitions, and ⚡Energy strain when used above the user's Reference Level. (See the Equipment design doc for full Equipment rules.)

This module does not define weapon lists, armor categories, or mandatory equipment scaling. It establishes expectations for clarity.

Weapons

A weapon should clearly communicate: - The Attribute commonly used for attack rolls (typically ❖Vigor or Finesse) - The Competency required to use it (Light Weapons, Heavy Weapons, Ranged Weapons, etc.) - Damage dice (e.g., 1d6, 1d8, 2d6) - Range in Units - Any traits or special interactions

A weapon's Level (if Enhanced) follows the standard Equipment Level rules: above the user's Reference Level reduces Maximum ⚡Energy while wielded.

Armor

Armor adds to Physical Defense per its design. It may also: - Impose Disfavor on stealth or movement-based rolls (per the system's design) - Require a specific Competency to wear without penalty - Carry an Equipment Level interaction if Enhanced

Multiple armor pieces stacking, layered armor systems, and shield interactions are system-defined. The engine does not prescribe stacking rules.

Items Granting Combat Abilities

Some items grant Abilities to their user (a magical sword granting Cleaving Strike, a focus granting Magic Missile). The granted Ability follows the rules in the Equipment design doc:

- The Ability's Level equals the item's Level
- The user must be able to Equip the item to access the Ability
- ⚡Energy cost follows normal Ability rules

These granted Abilities work in combat the same way as any other Ability.

The lethality and pacing of combat in a system built on this module are not determined by this module alone.

They emerge from:

- The size and scaling of ✚Health Pools
- The average damage per successful attack
- Whether Attribute Modifiers add to damage (default: no)
- The frequency of Critical Hits and Grazes (driven by Gradient interpretation)
- The availability of healing and Downtime
- The availability and strain of ⚡Energy

A heroic fantasy system may allow characters to endure multiple Rounds of sustained combat. A survival horror system may make each successful attack catastrophic. A science-fiction system may emphasize equipment strain and ⚡Energy management over raw durability.

This module provides structure. The system designer determines tone.

The following extensions are not part of the default Basic Combat Module. They are common modifications a system designer may want to add. Each is a deliberate tonal shift.

Attribute Modifiers to Damage

Default: Damage = weapon damage dice only.
Extension: Damage = weapon damage dice + governing Attribute Modifier.

Adding Attribute Modifiers to damage substantially increases lethality, particularly at high Attribute scores. A ❖Vigor 8 character with a 1d8 sword does an average of 8.5 damage per hit instead of 4.5. Combat ends faster, characters feel more impactful, but defensive options need to scale to match.

Use when: Building a heroic or high-impact system where character growth should translate directly to combat effectiveness.

Damage Reduction (Armor as DR)

Default: Armor adds to Physical Defense.
Extension: Armor also reduces incoming damage by a flat amount before it's applied to ✚HP.

Armor-as-DR makes armor feel more durable and shifts the math: low-damage attacks become trivial against heavy armor, while high-damage attacks still threaten armored targets.

Use when: Building a system where armor should feel like physical protection, not just an evasion bonus.

Lethal Critical Hits

Default: Critical Hits deal double damage.
Extension: Critical Hits deal double damage AND inflict 1 ⊗Wound directly, regardless of remaining ✚HP.

This makes Critical Hits feel genuinely dangerous and creates lasting consequences from individual rolls.

Use when: Building a lethal or grimdark system where every Critical Hit should carry weight beyond the damage number.

Aimed Attack

Default: All attacks resolve identically.
Extension: A character may spend an additional 1 ✦AP before attacking to "Aim." An Aimed attack gains Favor on the attack roll.

Aimed Attack adds a tactical option for characters who want to commit to a single decisive strike rather than multiple smaller ones. It interacts cleanly with the Multi-Attack Fatigue rule (an Aimed Attack is still one attack for purposes of multi-attack counting).

Use when: Building a system that wants a clear "all-in" tactical option for commit-and-strike playstyles.

Differentiated Critical Miss Consequences

Default: Critical Miss has a system-defined consequence (or none).
Extension: Critical Miss consequences vary by weapon type. Heavy weapons may drop on a Critical Miss; ranged weapons may jam or break a string; magical implements may discharge wildly.

This adds tonal flavor and makes weapon choice carry more identity beyond raw damage.

Use when: Building a system where weapon variety should feel meaningful beyond stat blocks.

To adopt this Combat Module within a Frontiers-based system:

1. **Confirm engine compatibility.** This module assumes the default Gradient Resolution and Encounter Play rules of Frontiers Beta 2026-2. If your system modifies those, adjustments may be required.
2. **Define your weapons.** Determine damage dice, ranges, Competencies, and any traits for each weapon in your system.
3. **Define your armor.** Determine Defense bonuses, Competencies, and any associated penalties.
4. **Decide on damage scaling.** Default is weapon dice only. If you want Attribute Modifiers to apply, adopt the Extension.
5. **Decide on Critical Miss consequences.** Pick one consistent interpretation for your system.
6. **Decide on extensions.** Pick which optional extensions (Attribute damage, armor as DR, lethal crits, Aimed Attack, weapon-specific Critical Miss) fit your tone.
7. **Define your combat-relevant Abilities.** Use the universal Ability anatomy (see Abilities design doc) for combat Abilities, Reactions, and signature moves.
8. **Playtest.** Combat tone is emergent from the interaction of HP scale, damage scale, Critical frequency, and recovery pacing. Adjust based on actual play.

Attribution Line for Systems Using This Module:

> Includes material from the **Basic Combat Module** by Jayke Paver, Beta 2026-2 Revision.

This module may stand alone as the default combat structure for a system, or it may serve as a foundation for additional genre-specific expansions.

For the engine's broader design philosophy, modularity, and the Variant and Alternative convention, see **Designing With Frontiers**.

For the condensed working ruleset, see the **Frontiers Overview**. This module is released under the **Open RPG Creative (ORC) License**.