Resolution, Initiative, and Reactions

Resolution, Initiative, and Reactions

Tick resolution

Each tick runs one loop over every actor, in initiative order:

1. **Actor action processing.** Does the actor have an action in progress?
   - **No** - select the next action (monsters: from the profile's ranked triggers; player: input - the game stops until they commit) and process its first tick. A zero-prep action flows straight through to the next phase.
   - **Yes** - process the current tick of the PER cycle.
2. **Reaction collection.** After the actor's processing, loop the other actors: did this tick trigger a reaction? If yes, add it to the queue - an actor may collect multiple reactions in the same tick. The reactor-side gate applies at collection: an actor in its own exec adds nothing (it is committed, not observing).
3. **Reaction pass.** Loop the reaction queue in initiative order:
   - An actor with several queued reactions evaluates them and uses the best one for the current circumstances; the rest are discarded. Monsters rank by profile; the player is offered the choice.
   - If the actor executes the reaction: pause or cancel their current action (if any), per that action's interruptibility, then process the first tick of the reaction's own PER cycle. A short reaction may complete its exec within this tick and spend subsequent ticks in recover.

Movement is not a special phase - it is an action like any other and runs through the same actor loop. Within a movement action, "leaving" reaction triggers queue before "entering" triggers; initiative breaks ties within each kind.

Deterministic, seeded, logged. This is what makes fights replayable for debugging and tuning.

Initiative

- Rolled at combat start (Perception or Stealth style, as in Pathfinder), **fixed for the whole fight**.
- Used as **priority for same-tick, world-affecting resolutions only** (actor order, reactions, movement events) - not for internal phase transitions.
- Tiebreakers: dex, then level, then PCs-over-monsters, then a **coin flip pre-rolled once per combat** (so identical actors stay consistent all fight).
- **Initiative as a speed stat**: a small random jitter is added to PER cycles; initiative reduces prep and recover time (never exec - exec durations stay consistent so active windows remain readable). High initiative therefore produces more actions over a fight. The jitter is seeded per combat so replays reproduce.

Reactions

Reactions are short actions taken out of sequence, triggered by observing another actor's phase.

Can I react at all? - gated by your own phase:

My phase	React?	Cost
Prep	Yes	Pauses or cancels my prep, per the action's interruptibility
Exec	No	Committed - this is the point
Recover	Yes	The reaction's own PER handles the cost

What can my reaction do? - gated by the observed target's phase:

Observed phase	What a reaction can do
Prep	Can **cancel** the target's action (interrupt the windup). Knowledge-gated: only actors who can read prep as prep can target it.
Exec	Can **mitigate** the action (reduce or disrupt its effect) - or do something entirely unrelated.
Recover	Cannot affect the completed action at all - but the target is vulnerable, so this is an **exploit window**: act while they wind down.

Reactions never *need* to counter the triggering action. They trigger on phase events, not as restricted counters: a magehunter reacts to a mage's casting by throwing a knife; a mage under attack reacts by casting shield for extra AC.

Reactor cost - the reaction's own PER. The tax lives in the reaction's own cycle, never the parent action's. Reactions should be built with short or zero prep and short exec; the **recover is the throttle** - pad it to tax use. This composes with the exploit window: reaction spam leaves the actor in a long recover, where enemies can take advantage.

Interruptibility (pause vs cancel). When an actor reacts, what happens to their in-progress action is an action property: actions that demand deep concentration are cancellable on reaction; lighter actions pause and resume once the reaction fully completes (including its recover). Set per action, like any other parameter.

Baiting is a core player skill: a monster that has already spent its reaction window has no answer to a maneuver.

Revision #3
Created 2026-09-27 16:48:29 UTC by Rowan Flora
Updated 2026-09-27 17:52:20 UTC by Rowan Flora