@ ? d: -> =>

Lonelog

A Standard Notation for Solo RPG Session Logging

Version 1.1.0

Table of contents

1. Introduction

If you've ever played a solo RPG, you know the challenge: you're deep in an exciting scene, dice are rolling, oracles are answering questions, and suddenly you realize: how do I capture all this without breaking the flow?

Maybe you've tried free-form journaling (gets messy), pure prose (loses the mechanics), or bullet points (hard to parse later). This notation system offers a different approach: a **lightweight shorthand** that captures the essential game elements while leaving room for as much (or as little) narrative as you want.

1.1 Why "Lonelog"?

This system started life as **Solo TTRPG Notation**, a name that was descriptive but unwieldy. Nearly 5,000 downloads later, it was clear the concept resonated with the community. But real-world use brought valuable lessons about what worked, what caused friction, and where the notation could evolve.

The rename to **Lonelog** reflects three insights:

- **A name that sticks.** "Solo TTRPG Notation" got abbreviated a dozen different ways. *Lonelog* is compact and evocative: *Lone* (solo play) + *log* (session record). It works.
- **A name you can find.** Search "solo ttrpg notation" and you'll drown in generic results. Search "lonelog" and you get *this system*. Think of how **Markdown** succeeded as both a format and a brand, it's not called "Text Formatting Notation." Lonelog gives this notation a distinct, findable identity.

- **A name built to last.** As the system matures, having a clear identity makes it easier for the community to share resources, tools, and session logs under one banner.

The core philosophy hasn't changed: separate mechanics from fiction, stay compact at the table, scale from one-shots to long campaigns, and work in both markdown and paper notebooks.

1.2 What Lonelog Does

Think of it as a shared language for solo play. Whether you're playing *Ironsworn*, *Thousand Year Old Vampire*, a non-solo RPG using Mythic GME, or your own homebrew system, this notation helps you:

- **Record what happened** without slowing down play
- **Track ongoing elements** like NPCs, locations, and plot threads
- **Share your sessions** with other solo players who'll understand the format
- **Review past sessions** and quickly find that crucial detail from three sessions ago

The notation is designed to be:

- **Flexible** — usable across different systems and formats
- **Layered** — works as both quick shorthand or expanded narrative
- **Searchable** — tags and codes make it easy to track NPCs, events, and locations
- **Format-agnostic** — works in digital markdown files or analog notebooks

The notation's goals:

- **Make reports written by different people readable at a glance:** standard symbols facilitate reading

- **Separate mechanics from fiction**: the best reports are those that highlight how the use of rules and oracles informs fiction
- **Have a modular and scalable system**: you can use the core symbols or extend the notation as you wish
- **Make it useful for both digital and analog notes**
- **Compliance and extension of markdown for digital use**

1.3 How to Use This Notation

Think of this as a **toolbox, not a rulebook**. The system is fully modular: grab what works for you and leave the rest behind.

At its core are just **five symbols** (see *Section 3: Core Notation*). They are carefully chosen to avoid conflicts with markdown formatting and comparison operators. These are the minimal language of play:

- `@` for player actions
- `?` for oracle questions
- `d:` for mechanics rolls
- `->` for oracle/dice results
- `=>` for consequences

That's it. **Everything else is optional.**

Scenes, campaign headers, session headers, threads, clocks, narrative excerpts—these are all enhancements you can add when they serve your play. Want to track a long campaign? Add campaign headers. Need to follow complex plots? Use thread tags. Playing a quick one-shot? Stick to the five core symbols.

Think of it as concentric circles:

- **Core Notation** (required): Actions, Resolutions, Consequences

- **Optional Layers** (add as needed): Persistent Elements, Progress tracking, Notes, etc.
- **Optional Structure** (for organization): Campaign Header, Session Header, Scenes

Start small. Try the core notation for one scene. If it clicks, great—keep going. If you need more, layer in what helps. Your notes should serve your play, not the other way around.

A note on licensing: This work is released under the CC BY-SA 4.0 license, that covers the Lonelog specification itself — this document. Session logs, actual plays, and other content you create using Lonelog notation are your own work and are not subject to this license. Write, publish, and license your sessions however you like.

1.4 Quick Start: Your First Session

Never used notation before? Here's everything you need:

```
S1 *Your starting scene*
@ Action you take
d: your roll result -> Success or Fail
=> What happens as a result

? Question you ask the oracle
-> Oracle's answer
=> What that means in the story
```

That's it! Everything else is optional. Try this for one scene and see how it feels.

Quick Start Example

```
S1 *Dark alley, midnight*
@ Sneak past the guard
d: Stealth 4 vs TN 5 -> Fail
=> I kick a bottle. Guard turns!

? Does he see me clearly?
-> No, but...
=> He's suspicious, starts walking toward the noise
```

1.5 Migrating from Solo TTRPG Notation v2.0

If you're already using Solo TTRPG Notation v2.0, welcome! Lonelog is an evolution of that system with clarified symbols for better consistency.

What Changed:

v2.0 Symbol	Lonelog Symbol	Why the Change
`>`	`@`	Avoids conflict with Markdown blockquotes
`->` (oracle only)	`->` (all resolutions)	Now unified for both dice AND oracle results
`=>` (overloaded)	`=>` (consequences only)	Clarified—no longer doubles as dice outcome

Key clarification: In v2.0, `=>` was confusingly used for both dice outcomes and consequences. Lonelog clarifies this by using `->` for ALL resolutions (dice and oracle), reserving `=>` exclusively for consequences.

Your Old Logs Are Still Valid

The structure and philosophy remain identical. Your existing logs are perfectly readable—you don't need to convert them unless you want consistency across your campaign.

Conversion

If you prefer manual conversion, use find & replace in your text editor:

1. Find: `>` (at start of lines) → Replace: `@`
2. The `->` and `=>` symbols are retained but with clarified usage

2. Digital vs Analog Formats

This notation works in **both digital markdown files and analog notebooks**. Choose the format that suits your play style.

2.1 Digital Format (Markdown)

In digital markdown files:

- **Campaign metadata** → YAML front matter (top of file)
- **Campaign Title** → Level 1 heading
- **Sessions** → Level 2 headings (`## Session 1`)
- **Scenes** → Level 3 headings (`### S1`)
- **Core notation and tracking** → Code blocks for easy copying/parsing
- **Narrative** → Regular prose between code blocks

Note: *Always wrap notation in code blocks (```` ``` ````) when using digital markdown. This prevents conflicts with Markdown syntax and ensures symbols like `=>` render correctly. Some Markdown extensions (Mermaid, Obsidian plugins) may interpret `=>` outside of code blocks.*

2.2 Analog Format (Notebooks)

In paper notebooks:

- Write headers and metadata directly as shown
- Core notation works identically but without code fences
- Use the same symbols and structure
- Brackets and tags help scanning paper pages

2.3 Format Examples

Digital markdown

~~~
## Session 1
*Date: 2025-09-03 | Duration: 1h30*

### S1 *School library after hours*

```
@ Sneak inside to check the archives
d: Stealth d6=5 vs TN 4 -> Success
=> I slip inside unnoticed. [L:Library|dark|quiet]
```
~~~

Analog notebook

```
=== Session 1 ===
Date: 2025-09-03 | Duration: 1h30

S1 *School library after hours*
@ Sneak inside to check the archives
d: Stealth d6=5 vs TN 4 -> Success
=> I slip inside unnoticed. [L:Library|dark|quiet]
```

Both formats use identical notation — only the wrapping differs.

3. Core Notation

This is the heart of the system—the symbols you'll use in nearly every scene. Everything else in this document is optional, but these core elements are what make the notation work.

There are only five symbols to remember, and they mirror the natural flow of solo play: you take an action or ask a question, you resolve it with mechanics or an oracle, then you record what happens as a result.

Let's break it down.

3.1 Actions

In solo play, uncertainty comes from two distinct sources: **you don't know if your character can do something** (that's mechanics), or **you don't know what the world does** (that's the oracle).

This distinction is fundamental. When you swing a sword, you use mechanics to see if you hit. When you wonder whether guards are nearby, you ask the oracle. Both create uncertainty, but they're resolved differently.

The notation reflects this with two different symbols—one for each type of action.

The `@` symbol represents you, the player, acting in the game world. Think of it as 'at this moment, I...' It's visually distinct from comparison operators, making your logs clearer and avoiding confusion when recording dice rolls.

Player-facing actions (mechanics):

```
@ Pick the lock
@ Attack the guard
@ Convince the merchant
```

World / GM questions (oracle):

```
? Is anyone inside?
? Does the rope hold?
? Is the merchant honest?
```

3.2 Resolutions

Once you've declared an action (`@`) or asked a question (`?`), you need to resolve the uncertainty. This is where the game system or oracle gives you an answer.

There are two types of resolutions: **mechanics** (when you roll dice or apply rules) and **oracle answers** (when you ask the game world a question).

3.2.1 Mechanics Rolls

Format:

```
d: [roll or rule] -> outcome
```

The `d:` prefix indicates a mechanics roll or rule resolution. Always include the outcome (Success/Fail or narrative result).

Examples

```
d: d20+Lockpicking=17 vs DC 15 -> Success
d: 2d6=8 vs TN 7 -> Success
d: d100=42 -> Partial success (using result
table)
d: Hack the terminal (spend 1 Gear) ->
Success
```

Comparison shorthand

When comparing rolls to target numbers, you can use comparison operators:

```
d: 5 vs TN 4 -> Success    (standard format)
d: 5≥4 -> S                (shorthand: ≥
means meets/exceeds TN)
```

```
d: 2≤4 -> F                (shorthand: ≤
means fails to meet TN)
```

Note: Comparison operators `≥` and `≤` work seamlessly with lonelog notation, with no symbol conflicts. You can also use `>=` and `<=`.

Add `S` (Success) or `F` (Fail) letters if you want explicit flags:

```
d: 2≤4 F
d: 5≥4 S
```

3.2.2 Oracle and Dice Results

The `->` symbol represents a definitive resolution—a declaration of outcome. The arrow visually shows "this leads to outcome," whether determined by dice mechanics or the oracle's answer.

Format:

```
-> [result] (optional: roll reference)
```

The `->` prefix indicates any resolution outcome—mechanics or oracle.

Dice Mechanics Results

For mechanics rolls, `->` declares Success or Fail:

```

```

```
d: Stealth d6=5 vs TN 4 -> Success
d: Lockpicking d20=8 vs DC 15 -> Fail
d: Attack 2d6=7 vs TN 7 -> Success
d: Hacking d10=3 -> Partial Success
```

Oracle Answers

For oracle questions, `->` declares what the world reveals:

```
-> Yes (d6=6)
-> No, but... (d6=3)
-> Yes, and... (d6=5)
-> No, and... (d6=1)
```

Common oracle formats

- **Yes/No oracles:** `-> Yes`, `-> No`
- **Yes/No with modifiers:** `-> Yes, but...`, `-> No, and...`
- **Degree results:** `-> Strong yes`, `-> Weak no`
- **Custom results:** `-> Partially`, `-> With a cost`

Why unified syntax?

Both mechanics and oracles resolve uncertainty. Using `->` for both creates consistency—every resolution gets the same declaration, making your log easier to scan and parse. Whether you rolled dice or asked the oracle, `->` marks the moment uncertainty becomes certainty.

3.3 Consequences

Record the narrative result after rolls using `=>`. The symbol shows consequences flowing forward from actions and resolutions. The double arrow visualizes how events cascade through your story.

```
=> The door creaks open, but the noise echoes
through the hall.
=> The guard spots me and raises the alarm.
=> I find a hidden diary with a crucial clue.
```

Multiple consequences

You can chain multiple consequence lines for cascading effects:

```
d: Lockpicking 5≥4 -> Success
=> The door opens
=> But the hinges squeal loudly
=> [E:AlertClock 1/6]
```

3.4 Complete Action Sequences

Here's how the core elements combine:

Mechanics-driven sequence

```
@ Pick the lock
d: d20+Lockpicking=17 vs DC 15 -> Success
```

```
=> The door creaks open, but the noise echoes
through the hall.
```

Oracle-driven sequence

```
? Is anyone inside?
-> Yes, but... (d6=4)
=> Someone is here, but they're distracted.
```

Combined sequence

```
@ Sneak past the guards
d: Stealth 2≤4 -> Fail
=> My foot kicks a barrel. [E:AlertClock 2/6]

? Do they see me?
-> No, but... (d6=3)
=> Distracted, but one guard lingers nearby.
[N:Guard|watchful]
```

4. Optional Layers

You’ve got the basics—actions, rolls, and consequences. That’s enough for simple play. But longer campaigns often need more: NPCs who reappear, plot threads that weave through sessions, progress that accumulates over time.

This section covers the **tracking elements** that help you manage complexity. They're all optional. If you're playing a one-shot mystery, you might not need any of this. If you're running a sprawling campaign with dozens of NPCs and multiple plot threads, you'll probably want most of it.

Pick and choose based on what your campaign needs.

4.1 Persistent Elements

As your campaign grows, certain things stick around: NPCs who reappear, locations you return to, ongoing threats, story questions that span sessions. These are your **persistent elements**.

Tags let you track them consistently across scenes and sessions. The format is simple: brackets, a type prefix, a name, and optional details. Like this: `[N:Jonah|friendly|wounded]`.

Why use tags?

- **Searchability**: Find every scene where Jonah appears
- **Consistency**: Reference NPCs the same way every time
- **Status tracking**: See how elements change over time
- **Memory aid**: Remind yourself of details weeks later

You don't need to tag everything—only what matters to your campaign. A random merchant you'll never see again? Just call them "the merchant" in prose. A recurring villain? Definitely tag them.

Here are the main types of persistent elements you might track:

4.1.1 NPCs

```
[N:Jonah|friendly|injured]
[N:Guard|watchful|armed]
[N:Merchant|suspicious]
```

Updating NPC tags:

When an NPC's status changes, you can either:

- Restate with new tags: `[N:Jonah|captured|wounded]`
- Show just the change: `[N:Jonah|captured]` (assumes other tags persist)
- Use explicit updates: `[N:Jonah|friendly→hostile]`
- Add `+` or `-`: `[N:Jonah|+captured]` or `[N:Jonah|-wounded]`

Choose the style that keeps your log clearest.

4.1.2 Locations

```
[L:Lighthouse|ruined|stormy]
[L:Library|dark|quiet]
[L:Tavern|crowded|noisy]
```

4.1.3 Events & Clocks

```
[E:CultistPlot 2/6]
[E:AlertClock 3/4]
[E:RitualProgress 0/8]
```

Events track significant plot elements. The number format `X/Y` shows current/total progress.

4.1.4 Story Threads

```
[Thread:Find Jonah's Sister|Open]
[Thread:Discover the Conspiracy|Open]
[Thread:Escape the City|Closed]
```

Threads track major story questions or goals. Common states:

- `Open` — active thread
- `Closed` — resolved thread
- `Abandoned` — dropped thread
- Custom states allowed (e.g., `Urgent`, `Background`)

4.1.5 Player Character

```
[PC:Alex|HP 8|Stress 0|
Gear:Flashlight,Notebook]
[PC:Elara|HP 15|Ammo 3|Status:Wounded]
```

Updating PC stats:

```
[PC:Alex|HP 8]        (initial)
[PC:Alex|HP-2]        (shorthand: lost 2 HP,
now at 6)
[PC:Alex|HP 6]        (explicit: now at 6 HP)
[PC:Alex|HP+3|Stress-1]  (multiple changes)
```

4.1.6 Reference Tags

To reference a previously established element without restating tags, use the # prefix:

```
[N:Jonah|friendly|injured]     (first mention
— establishes the element)

... later in the log ...

[#N:Jonah]                     (reference —
assumes tags from earlier)
```

The # tells you this element was defined earlier. Use it to:

- Keep later mentions concise
- Signal to readers they should look back for context
- Maintain searchability (the ID “Jonah” still appears)

When to use reference tags:

- First mention: Full tag with details `[N:Name|tags]`
- Later mentions in same scene: Optional, use judgment
- Later mentions in different scenes/sessions: Use `[#N:Name]` to signal reference
- Status changes: Drop the # and show new tags `[N:Name|new_tags]`

4.2 Progress Tracking

Some things in your campaign don’t happen all at once—they build up over time. The ritual takes twelve steps to complete. The guards’ suspicion grows with each noise you make. Your escape plan inches forward. The air supply counts down.

Progress tracking gives you a visual way to see these accumulating forces. Three formats handle different types of progression:

Clocks (fill up toward completion):

```
[Clock:Ritual 5/12]
[Clock:Suspicion 3/6]
```

Use for: Threats building, spells preparing, danger accumulating. When the clock fills, something happens (usually bad for you).

Tracks (progress toward a goal):

```
[Track:Escape 3/8]
[Track:Investigation 6/10]
```

Use for: Your progress on projects, journey advancement, long-term goals. When the track fills, you succeed at something.

Timers (count down toward zero):

```
[Timer:Dawn 3]
[Timer:AirSupply 5]
```

Use for: Deadlines approaching, resources depleting, time pressure. When it hits zero, time's up.

The difference? Clocks and tracks both go up, but clocks are threats (bad when full) and tracks are progress (good when full). Timers go down and create urgency.

You don't need to track everything numerically. Only use these when the accumulation matters to your story and you want a concrete way to measure it.

4.3 Random Tables & Generators

Solo play thrives on surprise. Sometimes you roll on a table to see what you find, or use a generator to create an NPC on the fly. When you do, it helps to record what you rolled—both for transparency and so you can recreate the logic later.

Simple table lookup:

```
tbl: d100=42 -> "A broken sword"
tbl: d20=15 -> "The merchant is nervous"
```

Use `tbl:` when you're pulling from a straightforward random table—the kind where you roll once and get a result.

Complex generators:

```
gen: Mythic Event d100=78 + 11 -> NPC
Action / Betray
gen: Stars Without Number NPC d8=3,d10=7 ->
Gruff/Pilot
```

Use `gen:` when you're using a multi-step generator that combines multiple rolls or produces compound results.

Integrating with oracle questions:

```
```

```
? What do I find in the chest?
tbl: d100=42 -> "A broken sword"
=> An ancient blade, snapped in two, with
strange runes on the hilt.
```

Why record the rolls? Three reasons:

1. **Transparency**: If you're sharing the log, others see your process
2. **Reproducibility**: You can trace how you got surprising results
3. **Learning**: Over time, you see which tables you use most

That said, if you're playing fast and loose, you can skip the roll details and just record the result: `=> I find a broken sword [tbl]`. The important part is the fiction, not the math.

4.3.1 Inline Table Definitions

The examples above assume your table lives somewhere else — a rulebook, a supplement, a separate file. You roll, you record the result, and anyone reading your log has to trust you (or own the same book) to verify it.

But what if you made the table yourself? What if you filtered options from a larger set to fit your campaign? What if you're playing a game where content generation *is* the game — systems like Bivius Companion, homebrew oracles, or any setup where the possibility space is part of the creative act?

In those cases, embedding the table directly in your log makes it **self-contained**. Readers see the full option space *and* the result. No external references, no "see page 47."

Format:

```
tbl: TableName (die)
  1: Result one
  2: Result two
  3: Result three
  4: Result four
  5: Result five
  6: Result six
```

The table name and die type go on the first line. Each entry is indented with its number and result. Then roll against it normally:

```
tbl: TableName d6=3 -> Result three
```

Complete example:

```
tbl: Forest Encounter (d6)
  1-2: Nothing — eerie silence
  3: Animal tracks, fresh
  4: Abandoned campsite
  5: Traveler on the road
  6: Something is following you

? What do I encounter on the forest path?
tbl: Forest Encounter d6=5 -> Traveler on the
road
=> A cloaked figure waves me down.
[N:Traveler|unknown|friendly?]
```

When to define inline vs. reference externally:

- **Inline** — when you created the table, when the table is short (roughly 10 entries or fewer), when shareability matters, or when the table only exists in your head
- **External** — when you're rolling on a published table that readers can look up, or when the table is too long to include without cluttering your log

For longer tables, you can define them once at the start of a session or campaign (much like the Resource Status Block pattern), then reference them by name throughout play:

```
tbl: Forest Encounter d6=5 -> Traveler on the
road
```

If the table was defined earlier in the log, readers can scroll back to find it. If it's a published table, the name and die type provide enough context to locate the source.

4.3.2 Filtered Option Sets

Some games don't use numbered tables — they use curated lists you pick or draw from. You might filter a larger set of options down to the ones relevant to your scene, then select randomly or intuitively.

Format:

```
tbl: TableName [Option A, Option B, Option C,
Option D]
```

Square brackets signal "these are the options in play." No numbers, no die — just the possibility space.

Rolling against a filtered set:

```
tbl: Mood [Tense, Melancholic, Hopeful,
Uncanny]
tbl: Mood -> Uncanny

tbl: Weather [Clear, Fog, Rain, Storm]
tbl: Weather d4=2 -> Fog
=> A thick fog rolls in from the coast.
Visibility drops to nothing.
```

Building a filtered set from a larger source:

```
(note: filtering Bivius Companion themes for
this arc)
tbl: Theme [Betrayal, Redemption, Sacrifice,
Secrets]

tbl: Theme -> Sacrifice
=> The scene will center on what someone is
willing to give up.
```

Dynamic filtering mid-session:

```
tbl: Available Leads [The dockworker's tip,
The torn letter, The locked room]
tbl: Available Leads -> The torn letter
=> I follow up on the letter I found in
Session 2.
[Thread:Torn Letter|Open]
```

The key difference from numbered tables: filtered sets capture *what was available*, not just what was chosen. This is especially valuable when you’re sharing logs — readers see the roads not taken alongside the one you picked.

4.3.3 Multi-Line Result Blocks

Some generators produce compound results — multiple axes of meaning that together create something greater than any single roll. An NPC might have a role, a personality trait, and a motivation. A location might have a feature, a mood, and a secret. Recording each axis makes the creative logic transparent.

Format:

```
gen: GeneratorName
  Axis1: roll -> result
  Axis2: roll -> result
  Axis3: roll -> result
```

Each axis is indented under the generator name. Roll details are optional — include them when transparency matters, skip them when speed matters.

NPC generator example:

```
gen: NPC (custom)
  Role: d6=3 -> Merchant
  Trait: d6=5 -> Secretive
  Want: d6=1 -> Escape
=> [N:Unnamed Merchant|secretive|wants to
flee town]
```

Location generator example:

```
gen: Ruin (custom d6 tables)
  Feature: d6=4 -> Collapsed tower
  Mood: d6=2 -> Oppressive silence
  Secret: d6=6 -> Hidden passage beneath the
rubble
=> [L:Old Watchtower|collapsed|eerie|hidden
passage]
```

With inline table definitions — you can combine these features. Define the axes, then roll:

```
tbl: NPC Role (d6) [Guard, Merchant, Scholar,
Beggar, Noble, Priest]
tbl: NPC Trait (d6) [Nervous, Secretive,
Boisterous, Cold, Kind, Obsessive]
tbl: NPC Want (d6) [Escape, Revenge, Wealth,
Knowledge, Power, Peace]

gen: NPC
  Role: d6=2 -> Merchant
  Trait: d6=6 -> Obsessive
  Want: d6=4 -> Knowledge
=> [N:The Collector|merchant|obsessive|seeks
forbidden texts]
```

Minimal format — when you just need the output:

```
gen: NPC -> Merchant / Secretive / Escape
=> [N:Unnamed Merchant|secretive|wants to
flee]
```

Use the expanded multi-line format when you want to show your work — especially useful in shared logs, for generators you created yourself, or when you want to trace how the fiction emerged from the mechanics. Use the minimal single-line format when speed matters more than process.

4.4 Narrative Excerpts

Here's a secret: **you don't need to write narrative at all**. The shorthand captures everything mechanically. But sometimes the fiction demands more—a piece of dialogue that's too perfect not to record, a description that sets the mood, a document your character finds.

That's what narrative excerpts are for: the moments where shorthand isn't enough.

Inline prose (short descriptions):

```
=> The room reeks of mildew and decay. Papers
are scattered everywhere.
```

Use for: Quick atmospheric details, sensory information, emotional beats. Keep it short—a sentence or two.

Dialogue (conversations worth recording):

```
N (Guard): "Who's there?"
PC: "Stay calm... just stay calm."
N (Guard): "Show yourself!"
PC: [whispers] "Not happening."
```

Use for: Memorable exchanges, character voice, important conversations. You don't need to record every word—just the exchanges that matter.

Long narrative blocks (found documents, important descriptions):

```
\---
The diary reads:
"Day 47: The tides no longer obey the moon.
The fish have stopped
coming. The lighthouse keeper says he sees
lights beneath the waves.
I fear for our sanity."
---\
```

Use for: In-world documents, lengthy descriptions, key revelations. The `\---` and `---\` markers separate it from your log, making it clear this is in-fiction content. The asymmetric delimiters prevent conflicts with Markdown horizontal rules.

How much narrative should you write? Only as much as serves you. If you're playing for yourself and shorthand tells you everything you need to remember, skip the prose. If you're sharing your log or you love the writing process, add more. There's no right amount—just what makes your log useful and enjoyable to you.

4.5 Meta Notes

Sometimes you need to step outside the fiction and leave yourself a note: a reminder about a house rule you're testing, a reflection on how a scene felt, a question to revisit later, or a clarification about your interpretation of a rule.

That's what meta notes are for—your out-of-character asides to yourself (or to readers, if you're sharing).

Format: Use parentheses to signal "this is meta, not fiction":

```
(note: testing alternate stealth rule where
noise increases Alert clock)
(reflection: this scene felt tense! the timer
really worked)
(house rule: giving advantage on familiar
terrain)
(reminder: revisit this thread next session)
(question: should I have rolled for that?
seemed obvious)
```

When to use meta notes:

- **Experiments**: Track rule variants or house rules you're testing
- **Reflection**: Capture what worked or didn't work emotionally
- **Reminders**: Flag things to follow up on later
- **Clarification**: Explain unusual rulings or interpretations
- **Process**: Document your thinking for shared logs

When NOT to use them: Don't let meta notes overwhelm your log. If you're stopping every few lines to reflect, you're probably over-thinking it. The game is the thing—meta notes are just occasional margin comments.

Think of them like director's commentary on a movie. Most of the time, you just watch the film. Occasionally, there's an interesting behind-the-scenes note worth sharing.

5. Optional Structure

So far we've talked about *what* you write (actions, rolls, tags). Now let's talk about *how you organize it.*

Structure helps in two ways: it makes your notes easier to navigate, and it signals boundaries (this session ended, that scene began). But structure adds overhead—more headers to write, more formatting to maintain.

This section shows you the organizing elements: campaign headers (metadata about your whole campaign), session headers (marking play sessions), and scene structure (the basic unit of play). Use what helps you stay oriented without slowing you down.

The key difference? **Digital and analog formats handle structure differently.** Digital markdown uses headings and YAML; analog notebooks use written headers and markers. We'll show both.

5.1 Campaign Header

Before you dive into play, it helps to record some basics: What are you playing? What system? When did you start? Think of this as the "cover page" of your campaign log.

This is especially useful when:

- You're running multiple campaigns (helps you remember which is which)
- You're sharing logs with others (they need context)
- You return to a campaign after a break (reminds you of tone/themes)

If you're just trying out the notation with a quick one-shot, skip this entirely. But for campaigns you plan to revisit, a header is worth the 30 seconds.

Digital and analog formats differ here. Digital markdown uses YAML front matter (structured metadata at the top of the file). Analog notebooks use a written header block.

For digital markdown files, use YAML front matter at the very top:

```
title: Clearview Mystery
ruleset: Loner + Mythic Oracle
genre: Teen mystery / supernatural
player: Roberto
pcs: Alex [PC:Alex|HP 8|Stress 0|
Gear:Flashlight,Notebook]
start_date: 2025-09-03
last_update: 2025-10-28
tools: Oracles - Mythic, Random Event tables
themes: Friendship, courage, secrets
tone: Eerie but playful
notes: Inspired by 80s teen mystery shows
```

For analog notebooks, write a campaign header block:

```
=== Campaign Log: Clearview Mystery ===
[Title]        Clearview Mystery
[Ruleset]      Loner + Mythic Oracle
[Genre]        Teen mystery / supernatural
[Player]       Roberto
[PCs]          Alex [PC:Alex|HP 8|Stress 0|
Gear:Flashlight,Notebook]
[Start Date]   2025-09-03
[Last Update]  2025-10-28
[Tools]        Oracles: Mythic, Random Event
tables
[Themes]       Friendship, courage, secrets
```

```
[Tone]          Eerie but playful
[Notes]         Inspired by 80s teen mystery
shows
```

Optional fields (add as needed):

- `[Setting]` — Geographic or world details
- `[Inspiration]` — Media that inspired the campaign
- `[Safety Tools]` — X-card, lines/veils, etc.

5.2 Session Header

A session header marks the boundary between play sessions and provides context: when did you play, how long, what happened?

Why use session headers?

- **Navigation**: Jump to specific sessions quickly
- **Context**: Remember when you played and what was happening
- **Reflection**: Track your play patterns (how often? how long?)
- **Continuity**: Connect sessions with recaps and goals

When to skip them:

- One-shot games (no multiple sessions)
- Continuous play (you play daily with no clear breaks)
- Pure shorthand logs (you just want the fiction, not the meta-structure)

Like campaign headers, digital and analog formats handle sessions differently. Choose the style that fits your medium.

5.2.1 Digital format (markdown heading)

```
## Session 1
*Date: 2025-09-03 | Duration: 1h30 | Scenes: S1-S2*

**Recap:** First session, introducing Alex and the mystery.

**Goals:** Set up the central mystery, establish the lighthouse as key location.
```

5.2.2 Analog format (written header)

```
=== Session 1 ===
[Date]          2025-09-03
[Duration]      1h30
[Scenes]        S1-S2
[Recap]         First session, introducing Alex and the mystery.
[Goals]         Set up the central mystery, establish the lighthouse.
```

Optional fields:

- `[Mood]` — Planned or actual tone for the session
- `[Notes]` — Rules variants, experiments, or special conditions
- `[Threads]` — Active threads this session

5.3 Scene Structure

Scenes are the basic unit of play within a session. At its simplest, a scene is just a numbered marker with context.

Digital format (markdown heading):

```
### S1 *School library after hours*
```

Analog format:

```
S1 *School library after hours*
```

The scene number helps you track progression and reference events later. The context (in italics/asterisks) frames where and when the scene takes place.

5.3.1 Sequential Scenes (Standard)

Most campaigns use simple sequential numbering:

```
S1 *Tavern, evening*
S2 *Town square, midnight*
S3 *Forest path, dawn*
S4 *Ancient ruins, midday*
```

When to use: Default for linear play. Scene 2 happens after Scene 1, Scene 3 after Scene 2, etc.

Numbering: Start at S1 each session, or continue across the whole campaign (S1-S100+).

Example in play:

```
S1 *Tavern common room, evening*
@ Ask the barkeep about rumors
d: Charisma d6=5 vs TN 4 -> Success
=> He leans in close and tells me about
strange lights at the old mill.
[Thread:Strange Lights|Open]

S2 *Outside the tavern, night*
@ Head toward the mill
? Do I encounter anything on the way?
-> Yes, but... (d6=4)
=> I see a shadowy figure, but they don't
seem hostile.
[N:Stranger|mysterious|watching]
```

5.3.2 Flashbacks

Flashbacks show past events that inform the current story. Use letter suffixes branching from the “present” scene.

Format: `S#a`, `S#b`, `S#c`

When to use:

- Revealing backstory mid-session
- Character memory triggers
- Showing how something happened
- Explaining mysterious elements

Example structure:

```
S5 *Investigating the mill*
=> I find my father's old journal.
```

```
S5a *Flashback: Father's workshop, 10 years ago*
(This happened before the campaign)
=> Father: "Promise me you'll never go to the mill alone."

S6 *Back at the mill, present day*
(Now we continue from S5)
```

Complete example:

```
S8 *Lighthouse keeper's quarters*
@ Search the desk for clues
d: Investigation d6=6 vs TN 4 -> Success
=> I find a faded photograph. It's... my mother? She's standing at this lighthouse!
[Thread:Mother's Connection|Open]

S8a *Flashback: Home, 15 years ago*
(Memory triggered by the photograph)
(Do I remember anything about this place?)
? Did mother ever mention a lighthouse?
-> Yes, but... (d6=5)
=> She mentioned it once, briefly, then changed the subject quickly.

PC (Young me): "Mom, where is this?"
N (Mother): [nervous] "Just an old place. Nothing important."

S8b *Flashback: Mother's study, 14 years ago*
(Following the thread of memory)
(Did I ever see documents about the
lighthouse?)
? Was I snooping in her papers?
-> Yes, and... (d6=6)
=> I found a deed. The lighthouse belonged to
our family!
[E:LighthouseSecret 1/4]

S9 *Lighthouse keeper's quarters, present*
(Back to current timeline)
=> Armed with this memory, I search more
carefully for family records.
```

Numbering tips:

- Branch from the scene that triggers the flashback
- Return to sequential numbering afterward
- Keep flashbacks short (1-3 scenes usually)
- Note in context when returning: `*Present day*` or `*Back at the...*`

5.3.3 Parallel Threads

When tracking multiple storylines that happen simultaneously or in alternating focus, use thread prefixes.

Format: `T#-S#` where T# is the thread number, S# is the scene number within that thread

When to use:

- Multiple characters/viewpoints
- Simultaneous events in different locations
- Alternating between plot lines
- Separate but related story arcs

Example structure:

```
T1-S1 *Main character at the lighthouse*
T2-S1 *Meanwhile, ally in the city*
T1-S2 *Back to lighthouse*
T2-S2 *Back to city*
T1-S3 *Lighthouse, continuing*
```

Complete example:

```
=== Session 3 ===
[Threads] Main story (T1), City investigation (T2)

T1-S1 *Lighthouse tower, dusk*
[PC:Alex|investigating the tower]
@ Climb to the top
d: Athletics d6=4 vs TN 4 -> Success
=> I reach the top. The light mechanism is still functional!

? Is anyone else here?
-> No, but... (d6=3)
=> Fresh footprints in the dust lead down.

T2-S1 *City archives, same time*
[PC:Jordan|researching at the library]
@ Search for lighthouse records
d: Research d6=6 vs TN 4 -> Success
=> I find construction documents from 1923. There's a hidden basement!
[E:SecretBasement 1/4]

T1-S2 *Lighthouse basement stairs*
[PC:Alex]
```

```
@ Follow the footprints down
d: Stealth d6=3 vs TN 5 -> Fail
=> A step creaks loudly.

? Does someone react?
-> Yes, and... (d6=6)
=> A voice from below: "Who's
there?" [N:Cultist|hostile|armed]

T2-S2 *City archives, moments later*
[PC:Jordan]
@ Call Alex to warn about the basement
? Does the call go through?
-> No, and... (d6=2)
=> No signal. The lighthouse is in a dead
zone!
[Clock:Alex in Danger 2/6]

T1-S3 *Lighthouse basement*
[PC:Alex|unaware of danger]
@ Try to talk my way out
d: Deception d6=2 vs TN 5 -> Fail
=> The cultist isn't buying it. He advances
with a knife!
```

When threads converge:

Once parallel threads meet, you can either:

- Continue with thread prefixes: `T1+T2-S5`
- Return to sequential: `S14` (note: threads merged)

```
T1-S6 *Alex escapes the lighthouse*
T2-S4 *Jordan drives toward the lighthouse*
```

```
S14 *Lighthouse entrance, both reunited*
(Threads merged)
[PC:Alex|wounded] meets [PC:Jordan|worried]
```

5.3.4 Montages and Time Cuts

For activities that span time or multiple quick vignettes, use decimal notation.

Format: `S#.#` (e.g., `S5.1`, `S5.2`, `S5.3`)

When to use:

- Traveling across long distances
- Training/research over weeks
- Multiple quick encounters
- Gathering resources
- Time-lapse sequences

Example structure:

```
S7 *Beginning the journey*
S7.1 *Day 1: Forest*
S7.2 *Day 3: Mountains*
S7.3 *Day 5: Desert*
S8 *Arriving at destination*
```

Complete example:

```
S12 *Preparing for the ritual*
=> I need to gather three components across
the region.
```

```
[Track:Ritual Components 0/3]

S12.1 *Herb shop, morning*
@ Buy sacred herbs
d: Persuasion d6=5 vs TN 4 -> Success
=> The herbalist gives me a discount.
[Track:Ritual Components 1/3]
[PC:Gold-5]

S12.2 *Blacksmith, afternoon*
@ Obtain silver dagger
? Is it in stock?
-> No, but... (d6=4)
=> He can make one by tomorrow.
[Timer:Ritual Deadline 2]

S12.3 *Graveyard, midnight*
@ Collect cemetery soil
? Am I interrupted?
-> Yes, and... (d6=6)
=> The groundskeeper catches me AND calls the
guard!
[Clock:Suspicion 3/6]

@ Run and hide
d: Stealth d6=6 vs TN 5 -> Success
=> I escape with the soil.
[Track:Ritual Components 2/3]

S13 *Blacksmith shop, next morning*
(Montage complete, back to sequential)
=> I collect the silver dagger.
[Track:Ritual Components 3/3]
```

Travel montage example:

```
S8 *Setting out from Port Ashan*
=> Three-week journey to the Northern Wastes begins.

S8.1 *Week 1: Coastal road*
? Encounters on the road?
tbl: d100=23 -> "Merchant caravan"
=> I join a caravan for safety. [N:Merchants|friendly]

S8.2 *Week 2: Mountain pass*
? Weather problems?
-> Yes, and... (d6=6)
=> Blizzard hits. The pass is blocked!
[Clock:Supplies Dwindle 2/4]

@ Find shelter
d: Survival d6=5 vs TN 5 -> Success
=> I locate a cave. [L:Mountain Cave|shelter|dark]

S8.3 *Week 3: Descending into wastes*
@ Navigate the frozen terrain
d: Survival d6=4 vs TN 6 -> Fail
=> I'm lost for two days.
[Clock:Supplies Dwindle 4/4]
[PC:Rations depleted]

S9 *Arriving at the Northern Wastes*
(Journey complete)
=> Exhausted and hungry, but I've made it.
```

5.3.5 Choosing Your Approach

Use sequential (S1, S2, S3) when:

- Playing straightforward, linear story
- Don't need complex time manipulation
- Want simplicity
- Most common choice

Use flashbacks (S5a, S5b) when:

- Revealing backstory mid-game
- Character development moments
- Explaining mysteries
- Short diversions from main timeline

Use parallel threads (T1-S1, T2-S1) when:

- Playing multiple characters
- Tracking simultaneous events
- Alternating between locations
- Complex, interwoven plots

Use montages (S7.1, S7.2) when:

- Covering long time periods
- Series of quick scenes
- Travel sequences
- Resource gathering
- Training/research periods

5.3.6 Scene Context Elements

Beyond numbering, enrich scenes with context in the frame:

Location:

```
S1 *Lighthouse tower*
S1 [L:Lighthouse] *Tower room*
```

Time markers:

```
S1 *Lighthouse, midnight*
S1 *Lighthouse, Day 3, dusk*
S1 *Two weeks later: Lighthouse*
```

Emotional tone:

```
S1 *Lighthouse (tense)*
S1 *Lighthouse - moment of calm*
```

Multiple elements:

```
S1 *Lighthouse tower, midnight, Day 3*
S5a *Flashback: Father's workshop, 10 years
ago*
T2-S3 *Meanwhile in the city, same evening*
S7.2 *Day 2 of journey: Mountain pass*
```

Minimal (just number):

```
S1
(Add context in first action or consequence)
```

Choose the level of detail that helps you track your story. More detail helps future reference; less detail keeps notes cleaner.

6. Complete Examples

Theory is one thing, but seeing the notation in action is where it clicks. This section shows complete play examples in different styles—from ultra-compact shorthand to rich narrative logs—so you can find the approach that works for you.

Each example demonstrates the same notation system, just with different levels of detail. Pick the style that matches your preference, or mix and match as your session demands.

6.1 Minimal Shorthand Log

Pure shorthand, no formatting — perfect for fast play:

```
S1 @Sneak d:4≥5 F => noise [E:Alert 1/6] ?
Seen? ->Nb3 => distracted
S2 @Search d:6≥4 S => find key [E:Clue 1/4] ?
Trapped? ->Yn6 => yes, spikes!
S3 @Dodge d:3≤5 F => HP-2 [PC:HP 6] =>
bleeding, need to retreat
```

6.2 Hybrid Digital Format

Combines shorthand with narrative, using markdown structure:

~~~markdown
### S7 *Dark alley behind tavern, Midnight*

```
@ Sneak past the guards
d: Stealth d6=2 vs TN 4 -> Fail
=> My foot kicks a barrel. [E:AlertClock 2/6]

? Do they see me?
-> No, but... (d6=3)
=> Distracted, but one guard lingers.
[N:Guard|watchful]
```

The guard's torch light sweeps across the
alley walls. I press myself
into the shadows, barely breathing.

```
N (Guard): "Who's there?"
PC: "Stay calm... just stay calm."
```
~~~

6.3 Analog Notebook Format

Same content as 6.2, formatted for handwritten notes:

```
S7 *Dark alley behind tavern, Midnight*

@ Sneak past the guards
d: Stealth d6=2 vs TN 4 -> Fail
=> My foot kicks a barrel. [E:AlertClock 2/6]
```

````
? Do they see me?
-> No, but... (d6=3)
=> Distracted, but one guard lingers.
[N:Guard|watchful]

The guard's torch light sweeps across the
alley. I press into shadows.

N (Guard): "Who's there?"
PC: "Stay calm... just stay calm."
````

6.4 Complete Campaign Log (Digital)

~~~
---
title: Clearview Mystery
ruleset: Loner + Mythic Oracle
genre: Teen mystery / supernatural
player: Roberto
pcs: Alex [PC:Alex|HP 8|Stress 0]
start_date: 2025-09-03
last_update: 2025-10-28
---

# Clearview Mystery

## Session 1
*Date: 2025-09-03 | Duration: 1h30*

### S1 *School library after hours*

```
~~~

~~~~
@ Sneak inside to check the archives
d: Stealth d6=5 vs TN 4 -> Success
=> I slip inside unnoticed. [L:Library|dark|
quiet]

? Is there a strange clue waiting?
-> Yes (d6=6)
=> I find a torn diary page about the
lighthouse. [E:LighthouseClue 1/6]
```

The page is yellowed with age. The
handwriting is shaky: "The light
calls to us. We must not answer."

```
[Thread:Lighthouse Mystery|Open]
```

### S2 *Outside the library, empty hall*

```
? Do I hear footsteps?
-> Yes, but... (d6=4)
=> A janitor approaches, but he doesn't
notice me yet. [N:Janitor|tired|suspicious]
```

I freeze. His keys jangle as he walks past
the doorway.

N (Janitor): "Thought I heard something..."
PC (Alex, whisper): "Gotta get out of here."
~~~~

~~~
```
@ Slip out while he's distracted
d: Stealth d6=6 vs TN 4 -> Success
=> I escape into the night safely.
```
## Session 2
*Date: 2025-09-10 | Duration: 2h*

**Recap:** Found diary page hinting at
lighthouse. Nearly spotted in library.

### S3 *Path to the old lighthouse, Day 2*

```
@ Approach quietly at dusk
d: Stealth d6=2 vs TN 4 -> Fail
=> I step on broken glass, crunching loudly.
[Clock:Suspicion 1/6]

? Does anyone respond from inside?
-> No, but... (d6=3)
=> The light flickers briefly in the tower
window. [L:Lighthouse|ruined|haunted]
```

### S4 *Inside lighthouse foyer*

```
@ Search the floor for signs of activity
d: Investigation d6=6 vs TN 4 -> Success
=> I find fresh footprints in the dust.
[Thread:Who is using the lighthouse?|Open]

tbl: d100=42 -> "A broken lantern"
~~~

~~~~
=> A cracked lantern lies near the stairs.
[E:LighthouseClue 2/6]
```

Someone's been here. Recently.

PC (Alex, thinking): "This place isn't as
abandoned as everyone thinks..."
~~~~

6.5 Complete Campaign Log (Analog)

```
=== Campaign Log: Clearview Mystery ===
[Title]        Clearview Mystery
[Ruleset]      Loner + Mythic Oracle
[Genre]        Teen mystery / supernatural
[Player]       Roberto
[PCs]          Alex [PC:Alex|HP 8|Stress 0]
[Start Date]   2025-09-03
[Last Update]  2025-10-28

=== Session 1 ===
[Date]        2025-09-03
[Duration]    1h30

S1 *School library after hours*

@ Sneak inside to check the archives
d: Stealth d6=5 vs TN 4 -> Success
=> I slip inside unnoticed. [L:Library|dark|
quiet]
```

```
? Is there a strange clue waiting?
-> Yes (d6=6)
=> I find a torn diary page about the
lighthouse. [E:LighthouseClue 1/6]

The page is yellowed. Shaky writing: "The
light calls to us."

[Thread:Lighthouse Mystery|Open]

S2 *Outside the library, empty hall*

? Do I hear footsteps?
-> Yes, but... (d6=4)
=> A janitor approaches, but doesn't notice
me yet. [N:Janitor|tired|suspicious]

N (Janitor): "Thought I heard something..."
PC (Alex): "Gotta get out of here."

@ Slip out while distracted
d: Stealth d6=6 vs TN 4 -> Success
=> I escape into the night safely.

=== Session 2 ===
[Date]        2025-09-10
[Duration]    2h
[Recap]       Found diary page, nearly
spotted in library.

S3 *Path to lighthouse, Day 2*

@ Approach quietly at dusk
d: Stealth d6=2 vs TN 4 -> Fail
```

```
=> I step on broken glass. [Clock:Suspicion
1/6]

? Does anyone respond?
-> No, but... (d6=3)
=> Light flickers in tower window.
[L:Lighthouse|ruined|haunted]

S4 *Inside lighthouse foyer*

@ Search floor for signs
d: Investigation d6=6 vs TN 4 -> Success
=> Fresh footprints in dust. [Thread:Who uses
lighthouse?|Open]

tbl: d100=42 -> "A broken lantern"
=> Cracked lantern near stairs.
[E:LighthouseClue 2/6]

PC (Alex): "This place isn't as abandoned as
everyone thinks..."
```

7. Best Practices

You've learned the notation—now let's talk about using it well. This section shows proven patterns that make your logs clearer and more useful, plus common mistakes to avoid.

Think of these as guidelines from the solo community's collective experience. They're not rigid rules, but they'll help you create logs that are easy to read, reference, and share.

7.1 Good Practices ✓

These patterns make your logs cleaner, more searchable, and easier to reference later. You don’t need to follow all of them, but they represent what works well for most solo players.

Do: Keep actions and rolls connected

```
@ Pick the lock
d: d20=15 vs DC 14 -> Success
=> The door swings open silently.
```

Do: Use tags for persistent elements

```
[N:Jonah|friendly|wounded]
[L:Lighthouse|ruined]
```

Do: Record consequences clearly

```
=> I find the key. [E:Clue 2/4]
=> But the guard heard me. [Clock:Alert 1/6]
```

Do: Use reference tags in later scenes

```
First mention: [N:Jonah|friendly]
Later: [#N:Jonah] approaches cautiously
```

Do: Mix shorthand and narrative as needed

```
@ Sneak past guard
d: 5≥4 S -> Success
=> I slip by unnoticed, heart pounding.
```

7.2 Bad Practices ✗

These are common pitfalls that make logs harder to read or parse. If you catch yourself doing these, don't worry—just adjust for next time. We've all been there!

Don't: Bury mechanics in prose

```
❌ I tried to pick the lock and rolled a 15
which beat the DC so I opened it

✓ @ Pick the lock
  d: 15≥14 -> Success
  => The door opens quietly.
```

Don't: Forget to record consequences

```
❌ @ Attack the guard
  d: 8≤10 -> Fail

✓ @ Attack the guard
  d: 8≤10 -> Fail
  => My blade glances off his armor. He
counterattacks!
```

Don't: Lose track of tags across scenes

```
❌ [N:Guard|alert] ... then later ...
[N:Guard|sleeping]
   (How did this change? When?)

✓ [N:Guard|alert] ... then later ...
  @ Knock him out
  d: 6≥5 S => [N:Guard|unconscious]
```

Don't: Mix action and oracle symbols

```
❌ ? Sneak past guards    (This is an
action, not a question)

✓ @ Sneak past guards    (Actions use @)
  ? Do they notice?      (Questions use ?)
```

Don't: Forget scene context

```
❌ S7
  @ Sneak past guards

✓ S7 *Dark alley, midnight*
  @ Sneak past guards
```

8. Templates

Starting from a blank page can be daunting. These templates give you a structured starting point—copy them, fill in the blanks, and start playing.

Each template comes in both **digital markdown** and **analog notebook** formats. Choose whichever matches your play style, or use them as inspiration to create your own.

Don't treat these as rigid forms. They're scaffolding. Once you're comfortable with the notation, you'll probably develop your own templates that fit your specific needs better.

8.1 Campaign Template (Digital YAML)

For digital markdown files, use YAML front matter to store campaign metadata. This goes at the very top of your file, before any other content.

Copy this template, fill in your details, and you're ready to start your first session.

```yaml
title:
ruleset:
genre:
player:
pcs:
start_date:
last_update:
tools:
themes:
tone:
notes:
```

```
# [Campaign Title]

## Session 1
*Date: | Duration: *
```

```
### S1 *Starting scene*

Your play log here...
```

8.2 Campaign Template (Analog)

For paper notebooks, write this header block at the start of your campaign log. Keep it simple—you can always add more details later if needed.

```
=== Campaign Log: [Title] ===
[Title]
[Ruleset]
[Genre]
[Player]
[PCs]
[Start Date]
[Last Update]
[Tools]
[Themes]
[Tone]
[Notes]

=== Session 1 ===
[Date]
[Duration]

S1 *Starting scene*

Your play log here...
```

8.3 Session Template

Use this at the start of each play session to mark boundaries and provide context. The digital version uses markdown headings; the analog version uses written headers.

Fill in what's useful and skip what's not. The only essential field is the date—everything else is optional.

Digital:

```
## Session X
*Date: | Duration: | Scenes: *

**Recap:**

**Goals:**

### S1 *Scene description*
```

Analog:

```
=== Session X ===
[Date]
[Duration]
[Recap]
[Goals]

S1 *Scene description*
```

8.4 Quick Scene Template

This is your workhorse template—the basic structure you'll use scene after scene. It's intentionally minimal: just enough structure to keep you oriented without slowing you down.

Use this as your default starting point for every scene, whether you're playing digitally or analog.

~~~
S# *Location, time*
```
@ Your action
d: your roll -> outcome
=> What happens

? Your question
-> Oracle answer
=> What it means
```
~~~

9. Adapting to Your System

Here's the beautiful part: this notation works with *any* solo RPG system. *Ironsworn*, *Mythic GME*, *Thousand Year Old Vampire*, your own homebrew—doesn't matter. The core symbols stay the same; only the resolution details change.

This section shows you how to adapt the `d:` roll notation and `->` oracle formats to match your specific game system. We'll cover common systems (PbtA, FitD, Ironsworn, OSR) and oracles (Mythic, CRGE, MUNE), but the principles work for anything.

The key insight: The notation separates *mechanics* from *fiction*. Your system determines how mechanics work; the notation just records them consistently.

9.1 System-Specific Roll Notation

The `d:` notation works with any system—you just need to adapt it to your specific dice mechanics. Here's how the notation looks across popular solo RPG systems.

These examples show the pattern: record what you rolled, compare it to what you needed, note the outcome. The details change by system, but the structure stays the same.

9.1.1 Powered by the Apocalypse (PbtA)

```
d: 2d6=9 -> Strong Hit (10+)
d: 2d6=7 -> Weak Hit (7-9)
d: 2d6=4 -> Miss (6-)
```

9.1.2 Forged in the Dark (FitD)

```
d: 4d6=6,5,4,2 (take highest=6) -> Critical Success
d: 3d6=4,4,2 -> Partial Success (4-5)
d: 2d6=3,2 -> Failure (1-3)
```

9.1.3 Ironsworn

```
d: Action=7+Stat=2=9 vs Challenge=4,8 -> Weak
Hit
d: Action=10+Stat=3=13 vs Challenge=2,7 ->
Strong Hit
```

9.1.4 Fate/Fudge

```
d: 4dF=+2 (++0-) +Skill=3 = +5 -> Success
with Style
d: 4dF=-1 (-0--) +Skill=2 = +1 -> Tie
```

9.1.5 OSR/Traditional D&D

```
d: d20=15+Mod=2=17 vs AC 16 -> Hit
d: d20=8+Mod=-1=7 vs DC 10 -> Fail
```

9.2 Oracle Adaptations

Different oracle systems have different output formats. Some give yes/no answers, others generate complex results. Here's how to record results from popular oracle systems.

The key is consistency: always use `->` for oracle results, then capture whatever information your oracle provides.

9.2.1 Mythic GME

```
? Does the guard notice me? (Likelihood:
Unlikely)
-> No, but... (CF=4)
=> He doesn't see me, but he's suspicious.
```

9.2.2 CRGE (Conjectural Roleplaying Game Engine)

```
? What is the merchant's mood?
-> Surge: Actor + Focus => Angry + Betrayal
=> The merchant is furious about being
cheated.
```

9.2.3 MUNE (Madey Upy Number Engine)

```
? Is anyone home?
-> Likely + roll 2,4 => Yes
=> Lights are on, someone's definitely
inside.
```

9.2.4 UNE (Universal NPC Emulator)

```
gen: UNE Motivation -> Power + Reputation
=> [N:Baron|ambitious|seeks recognition]
```

9.3 Handling Edge Cases

Every system has quirks. Here's how to handle common situations that don't fit the basic notation patterns.

9.3.1 Multiple Rolls in One Action

When you need to make multiple rolls for one action:

Advantage/Disadvantage:

```
@ Attack with advantage
d: 2d20=15,8 (take higher) vs TN 12 -> 15≥12 Success
=> I strike true, blade finding a gap in the armor.
```

Multiple dice pools:

```
@ Perform complex ritual
d: INT d6=4, WILL d6=5, vs TN 4 each -> Both succeed
=> The spell takes hold, energy crackling between my fingers.
```

Contested rolls:

```
@ Arm wrestle the sailor
d: STR d20=12 vs sailor d20=15 -> 12≤15 Fail
=> His grip tightens. My arm slams to the table.
```

9.3.2 Ambiguous Oracle Results

When the oracle gives unclear or contradictory results:

```
? Is the merchant trustworthy?
-> Yes, but... (d6=4)
(note: "but" contradicts "yes"—interpreting
as: trustworthy but hiding something)
=> He seems honest, but keeps glancing at the
door nervously.
```

Or re-roll if truly stuck:

```
? Can I trust him?
-> Unclear result (d6=3 on binary oracle)
(note: re-rolling with different framing)
? Is he trying to help me?
-> No, and... (d6=2)
=> He's actively working against me.
```

9.3.3 Nested Consequences

Sometimes one consequence leads to another, creating a cascade:

```
d: Lockpicking 5≥4 -> Success
=> The door opens
=> But the hinges squeal loudly
=> Guards in the next room hear it
[E:AlertClock 1/6]
=> One starts walking this way [N:Guard|
investigating]
```

When to use: Major successes or failures with multiple ripple effects. Don’t overuse—most actions have one clear consequence.

9.3.4 Failed Oracle Questions

What if the oracle doesn’t help?

```
? What's behind the door?
-> [Roll unclear/contradictory]
(note: asking a more specific question)
? Is there danger behind the door?
-> Yes, and...
=> Danger, and it's immediate!
```

Pro tip: If an oracle result doesn’t spark fiction, it’s okay to re-frame the question or roll again. The oracle serves your story, not the other way around.

Appendices

A. Solo RPG Notation Legend

This is your quick reference—the cheat sheet to keep handy while you play. Forget what `=>` means? Need to remember how to format a clock? This section has you covered.

Think of it as the notation’s “vocabulary list.” Everything here has been explained earlier in detail; this is just the condensed version for fast lookup.

Bookmark this section. You'll come back to it often in your first few sessions, then less and less as the notation becomes second nature.

A.1 Core Symbols

Symbol	Meaning	Example
`@`	Player action (mechanics)	`@ Pick the lock`
`?`	Oracle question (world/uncertainty)	`? Is anyone inside?`
`d:`	Mechanics roll/result	`d: 2d6=8 vs TN 7 -> Success`
`->`	Oracle/dice result	`-> Yes, but...`
`=>`	Consequence/outcome	`=> The door opens quietly`

A.2 Comparison Operators

- `≥` or `>=` — Greater than or equal (meets/beats TN)
- `≤` or `<=` — Less than or equal (fails to meet TN)
- `vs` — Versus (explicit comparison)
- `S` — Success flag
- `F` — Fail flag

A.3 Tracking Tags

- `[N:Name|tags]` — NPC (first mention)
- `[#N:Name]` — NPC (reference to earlier mention)
- `[L:Name|tags]` — Location
- `[E:Name X/Y]` — Event/Clock
- `[Thread:Name|state]` — Story thread
- `[PC:Name|stats]` — Player character

A.4 Progress Tracking

- `[Clock:Name X/Y]` — Clock (fills up)
- `[Track:Name X/Y]` — Progress track
- `[Timer:Name X]` — Countdown timer

A.5 Random Generation

- `tbl: roll -> result` — Simple table lookup
- `gen: system -> result` — Complex generator

A.6 Structure

- `S#` or `S#a` — Scene number
- `T#-S#` — Thread-specific scene

A.7 Narrative (Optional)

- Inline: `=> Prose here`
- Dialogue: `N (Name): "Speech"`
- Block: `--- text ---`

A.8 Meta

- `(note: ...)` — Reflection, reminder, house rule

A.9 Complete Example Line

```
S3 @Pick lock d:15≥14 S => door opens quietly
[N:Guard|alert]
```

B. FAQ

Got questions? You're not alone. These are the most common questions from people learning the notation, along with straight answers.

If your question isn't here, remember: the notation is flexible. If you're wondering whether you can do something differently, the answer is probably "yes, if it works for you."

Q: Do I need to use every element?
A: No! Start with just `@`, `?`, `d:`, `->`, and `=>`. Add other elements only if they help you.

Q: Can I use this with traditional RPGs (with a GM)?
A: The core notation works great for any RPG notes. The oracle elements (`?`, `->`) are specifically for solo play, but the action/resolution notation works everywhere.

Q: What if my system doesn't use dice?
A: Use `d:` for any resolution mechanic: `d: Draw from deck -> Queen of Spades`, `d: Spend token -> Success`

Q: Should I use digital or analog format?
A: Whichever you prefer! They use the same notation. Digital has better search/organization; analog is immediate and tactile.

Q: How detailed should my notes be?
A: As detailed as you want! The system works for pure shorthand (Example 6.1) or rich narrative (Example 6.4).

Q: Can I share my logs with others?
A: Yes! That's one reason for standardized notation. Others familiar with the system can read your logs easily.

Q: What about house rules or custom symbols?
A: Document them in meta notes: `(note: using + for`

`advantage, - for disadvantage)`. The system is designed to be extended.

Q: Do scene numbers have to be sequential?
A: No. Use `S1`, `S2`, `S3` for simplicity, but branch (`S3a`, `S3b`) or use thread prefixes (`T1-S1`) if helpful.

Q: Should I update tags every time something changes?
A: Show significant changes explicitly: `[N:Guard|alert]` → `[N:Guard|unconscious]`. Minor changes can be implied through narrative.

Q: If I post a recorded session log publicly, does it need to be under the ShareAlike license?
A: No. The CC BY-SA 4.0 license covers the Lonelog specification document, not content created using the notation. Your session logs are your own independent creative work — publish and license them however you choose. The ShareAlike clause would only apply if you were adapting or redistributing the spec itself, for example by forking Lonelog into a new notation system.

Q: I made my own random table. How do I include it in my log?
A: Define it inline with `tbl: Name (die)` followed by indented entries, or use `tbl: Name [Option A, Option B, ...]` for unnumbered option sets. See §4.3.1 and §4.3.2. This makes your log self-contained — readers see the full table and the result without needing external references.

C. Symbol Design Philosophy

Lonelog's symbols were chosen for specific reasons:

- `@` **(Action)**: Represents "at this point" or the actor taking action. Changed from `>` in v2.0 to avoid conflict with Markdown blockquotes.
- `?` **(Question)**: Universal symbol for inquiry. Unchanged from v2.0.
- `d:` **(Dice/Resolution)**: Clear abbreviation for dice rolls. Unchanged from v2.0.
- `->` **(Resolution)**: Retained from v2.0. Now unified for ALL resolutions (dice and oracle). The arrow visually shows "this leads to outcome."
- `=>` **(Consequence)**: Retained from v2.0. Double arrow shows cascading effects. Clarified usage: consequences only (v2.0 overloaded this for dice outcomes too).

Markdown Compatibility: All symbols work cleanly in code blocks and don't conflict with markdown formatting or mathematical operators. Always wrap notation in code blocks when using digital markdown to prevent conflicts with Markdown extensions.

Credits & License

This notation is inspired by the Valley Standard.

Thanks to:

- matita for the `+/-` method to track changes in tags
- flyincaveman for the suggestion on the use of the `@` symbol for character actions (in the tradition of the early ASCII rpgs)
- r/solorpgplay and r/Solo_Roleplaying for the positive reception of this notation and the useful feedbacks.

- Enrico Fasoli for playtesting and feedback

Version History:

- v 1.1.0: Clarified the use of the license. Added specifications for inline definitions, filtered option sets and multiline result blocks in section 4.3.
- v 1.0.0: Evolved from Solo TTRPG Notation v2.0 by Roberto Bisceglie

You are free to:

- Share — copy and redistribute the material
- Adapt — remix, transform, and build upon the material

Under these terms:

- Attribution — Give appropriate credit
- ShareAlike — Distribute adaptations under the same license

Happy adventuring, solo players!