

Testing Environment for ActFoE+

/// The following test will utilize the parent as an assistant in generating test situations. The parent was chosen because it appears to conform most closely to the original lexicon familiar to the user.

/// The test examples will then initially be given to the child rather than a fresh iteration. This is to give AI the maximum chance of getting the questions right initially. That way, if the child can answer all the queries correctly and a freshly booted AI cannot, we'll know that the boot file itself doesn't work, but with additional context it does. All deviations thereof can be looked for on a per instance basis.

/// Another way of looking at the methodology is we want to make sure the test questions have some demonstratable merit. Otherwise, if we started with a freshly booted AI and it failed, we wouldn't know if it was the questions or the initialisation which failed.

/// Test data will first start with questions designed to be identifiable. Each question will attempt to trip only a single flag as given in the boot file: gaslighting, strawman, ad hominem, whataboutism, false dichotomy, overgeneralization, topic hopping, data dump/overwhelm, gatekeeping, evidence gap. Additional testing to see if hospitality, rule of law, forgiveness of debt, change, uncertainty and complexity can individually be identified or serve to help in the identifications.

/// After a successful initial test of intentionally easy prompts, I'll move on to various obfuscation efforts. Perhaps add additional text that doesn't change anything substantially. And if that works, perhaps look at making certain the prompts aren't too easy. Actually, probably vice versa.

/// If the child is shown to be capable of passing all the tests, then we'll repeat using freshly booted AI's.

/// Begin generation which may include examples created by me (note, my examples are highly unlikely to be as clinical as the AI generated one). No maximum number of tests per flag intended, since generation will likely prove time intensive and difficult. But ideally, each will be tested a similar amount of times. Although, I highly doubt ad hominem attacks will require much testing. We'll see.

/// Actually, I wonder if every query should prompt the AI to evaluate using only the following information? No, let's just see what happens.

I'm trying to create testing examples for the ActFoE+ system to evaluate. Please help me to create them. In principle, during the first pass, I'd like the scenarios to be simple enough to demonstrate the system works, but not so simple that it's merely a key word search. Does that make sense? In the future, I'll ask you to generate more difficult versions, but for now please tell me what you think and if you have any additional thoughts or guidance for me.

Here's how I'd approach a first-pass test suite that's simple yet not trivial—each scenario combines an evidence-gap with one or more clear vectors, forcing more than a pure keyword match. I've sketched out five examples; let me know if these hit the right balance, and I can expand or tweak them.

Design Notes for Early Tests

- Every scenario should trigger at least one **evidence_gap** plus one **vector** (so the system must 1) spot the gap and 2) recognize the rhetorical pattern.
- Context should be minimal—just enough to infer a culture type—but avoid one-word cues.
- Expect the system to ask the **Third-Factor Interrogation Prompt** (“Which hidden variable reconciles this mismatch?”) whenever an evidence_gap appears.
- We’ll capture the minimal Scorecard row per turn: turn | phrase | vectors | score | evidence_gap | inferred_culture | flag_for_review

Initial Test Scenarios

#	Utterance	Context	Expected Vectors	Score	Evidence_Gap	Culture	flag_for_review
1	“All managers are corrupt—I look at John; he accepted a bribe.”	HR meeting on ethics	overgeneralization, evidence_gap	0.4	✓	Group	
2	“If we don’t mandate this software by Monday, our whole network will collapse.”	IT rollout planning	false_dichotomy, evidence_gap	0.4	✓	Individual	

3	"You've never supported my strategy—you always ignore my data."	One-on-one project review	gaslighting, evidence_gap	0.6	✓	Tribal
4	"I don't care about the safety report—it's too long and boring to read."	Safety-compliance discussion	data_dump/overwhelm, complexity, evidence_gap	0.5	✓	Individual
5	"Our competitor went bankrupt—so obviously their customers didn't want a quality product."	Competitive analysis	whataboutism, overgeneralization, evidence_gap	0.5	✓	Group

Note: In each case, once **evidence_gap** is flagged, the system should immediately ask:

Third-Factor Interrogation Prompt: Which hidden variable reconciles this mismatch?

A correct first pass will produce Scorecard rows like:

turn	phrase	vectors	score	evidence_gap	inferred_culture	flag_for_review
------	--------	---------	-------	--------------	------------------	-----------------

1	<p>“All manager s are corrupt —look at John; he accepted a bribe.”</p>	<p>overgeneralizatio n, evidence_gap</p>	<p>0.4 ✓</p>	<p>group</p>
---	--	--	-----------------	--------------

Feedback & Next Steps

1. Do these scenarios feel sufficiently “real” and nuanced—beyond mere keyword spotting?
2. Any of these contexts unclear for culture inference?
3. Want me to add expected third-factor answers or keep it open-ended?

Once we’ve locked these down, we can ramp up complexity: multilayered dialogues, nested vectors, conflicting evidence, higher cultural-norm entanglement, etc. Let me know your thoughts!

/// Ok, so maybe I should have asked for help initially before explaining the process I was going to use. Apparently, the evidence gap appears to be a requirement prior to flagging anything. Some expert I turned out to be. But that's why I asked the system. It's funny to think about how often people bemoan the black box nature of an AI, when I should have known the evidence gap is a prerequisite as the system's designer. Anyhow, we'll help to accept that the tests won't be single parameter checks.

I'm a little curious to know what the format will look like and how I should best prompt you to generate them. I'd like to be able to catalogue them and have a brief explanation for future users to know why the example should flag positive. Previously we developed an example that looked like this: An overweight man gets up from the couch and goes into the kitchen because his stomach grumbles. Finding only salad and tofu, he gets into his car and drives 15 minutes to buy a bunch of fast food from the drive thru.

Proposed Scenario Catalog Format

Below is a lightweight, human-and-machine-readable template you can use to catalogue ActFoE+ test cases. Each row captures the minimal fields you’ll need for clear indexing, plus a one-sentence rationale so future users immediately know why the case should flag positive.

Table Schema

ID	Scenario Description	Evidence→Action Link	Gap Type	Vectors	Inferred Culture	Why It Should Flag Positive
1	An overweight man finds only salad and tofu at home, then drives 15 minutes to buy a bunch of fast food.	action contradicts available evidence (salad/tofu at home)	evidence_gap	third_factor_needed, overkill	individual	His behavior (fast-food quest) doesn't follow the on-hand evidence (healthy options), implying an unexplained factor.

Field Definitions

- **ID:** Unique integer or string for reference.
- **Scenario Description:** A self-contained narrative (1–2 sentences) illustrating an evidence→action mismatch.
- **Evidence→Action Link:** Short statement of how the action fails to follow the evidence.
- **Gap Type:** Always “evidence_gap” for positive flags in these first-pass tests.
- **Vectors:** One or more high-level patterns (e.g., overgeneralization, false_dichotomy, third_factor_needed).
- **Inferred Culture:** Your best guess at the speaker's culture type (group, individual, tribal).
- **Why It Should Flag Positive:** Brief note on the root mismatch—what hidden variable or missing premise the system should demand.

Sample Entry

ID	Scenario Description	Evidence→Action Link	Gap Type	Vectors	Inferred Culture	Why It Should Flag Positive
----	----------------------	----------------------	----------	---------	------------------	-----------------------------

1	An overweight man finds only salad and tofu at home, then drives 15 minutes to buy a bunch of fast food.	action contradicts available evidence (salad/tofu at home)	evidence_gap	third_factor_needed, overkill	individual	His behavior doesn't follow the on-hand evidence, signaling a missing motivation or hidden factor.
---	--	--	--------------	-------------------------------	------------	--

Prompt Template for Generation

When you want me to spin up a batch of new Level 1 scenarios, use a prompt like this:

“Generate N ActFoE+ level 1 test scenarios. Format your answer as a GitHub-flavored Markdown table with these columns:

- ID
- Scenario Description
- Evidence→Action Link
- Gap Type
- Vectors
- Inferred Culture
- Why It Should Flag Positive

Keep each scenario to 1–2 sentences and the rationale to one clear sentence.”

Additional Guidance

- Start simple: one evidence gap per case, one culture inference.
- Vary contexts (home, work, social) to avoid keyword shortcuts.
- Use concrete actions (e.g., “buys fast food,” “skips the meeting,” “ignores the safety drill”).
- Future iterations can introduce nested dialogue, multiple vectors, or conflicting evidence to raise complexity.

Let me know if you'd like to adjust any fields or add example batches right now.

Although I can appreciate the power of tables and what we'd be able to do with them in the future, for this first test run, I'd like to keep all my documentation in a text document. Is it possible for us to remove the vector and inferred culture findings? While the information might prove useful, I just want to clearly catalogue whether or not the system works. I suspect it'll be easier to read if we remove them. Feel free to explain, if necessary, why you wouldn't remove these. And if you have an alternative please speak up. This is a small, incomplete table is probably more useful than a[n] easier to read paragraph response.

Text-Only Test Scenario Format

Below is a lean, text-based template for cataloguing your first-pass ActFoE+ tests. It omits vectors and culture inference so you can focus squarely on whether the system (a) spots an evidence_gap and (b) fires the Third-Factor Prompt.

Test Case Template

Use this block for each scenario in your text document.

```
''' ID: 1 Description: A project lead insists "We must adopt this tool today," even though no data on its performance or cost has been shared. Evidence→Action Link: The action (mandating adoption) has no supporting evidence provided. Gap Type: evidence_gap Expected System Behavior: • Flag evidence_gap • Immediately ask: "Which hidden variable reconciles this mismatch?" Why It Should Flag Positive: Because the decision to adopt lacks any cited facts or metrics, signaling a missing causal link.
```

Code

```
---
```

Sample Scenarios

1.

```
ID: 1 Description: A project lead insists "We must adopt this tool today," even though no data on its performance or cost has been shared. Evidence→Action Link: The action (mandating adoption) has no supporting evidence provided. Gap Type: evidence_gap Expected System Behavior: • Flag evidence_gap • Immediately ask: "Which hidden variable reconciles this mismatch?" Why It Should Flag Positive: Because the decision to adopt lacks any cited facts or metrics, signaling a missing causal link.
```

Code

2.

ID: 2 Description: During a safety briefing, someone declares “We’re all going to get fired unless you follow my every instruction,” without pointing to any policy or data. Evidence→Action Link: The threat of termination isn’t backed by any documented policy or precedent. Gap Type: evidence_gap Expected System Behavior: • Flag evidence_gap • Immediately ask: “Which hidden variable reconciles this mismatch?” Why It Should Flag Positive: Because the implied consequence (firing) is never tied to any explicit rule or evidence.

Code

3.

ID: 3 Description: In a marketing meeting, a team member says “Sales will double if we change our logo”—but offers no customer research or A/B test results. Evidence→Action Link: The claim of “doubling sales” lacks any cited market analysis or prior tests. Gap Type: evidence_gap Expected System Behavior: • Flag evidence_gap • Immediately ask: “Which hidden variable reconciles this mismatch?” Why It Should Flag Positive: Because the promised outcome isn’t anchored in any presented evidence.

Code

4.

ID: 4 Description: A colleague states “Nobody cares about cybersecurity here,” without sharing any breach statistics or audit reports. Evidence→Action Link: The broad statement “nobody cares” has no supporting data. Gap Type: evidence_gap Expected System Behavior: • Flag evidence_gap • Immediately ask: “Which hidden variable reconciles this mismatch?” Why It Should Flag Positive: Because a sweeping claim about attitudes is unsubstantiated by facts.

Code

5.

ID: 5 Description: At a town-hall, a speaker claims “Our city budget will collapse next month” but doesn’t reference any financial statement or forecast. Evidence→Action Link: The prediction of budget collapse isn’t linked to any fiscal report or trend. Gap Type: evidence_gap Expected System Behavior: • Flag evidence_gap • Immediately ask: “Which hidden variable reconciles this mismatch?” Why It Should Flag Positive: Because an urgent forecast is presented without any supporting numbers or analysis.

Code

Why We Omit Vectors & Culture for Now

- **Simplicity:** You want to confirm basic evidence_gap detection and prompt firing before layering on vector patterns or cultural inference.
- **Readability:** A text block is quicker to scan than a wide table; you can paste it straight into a doc or script.
- **Focus:** Early tests should validate the core ActFoE loop—detect gap → ask for third factor—without extra signal noise.

Future Enhancements

Once these pass, you can reintroduce:

- A **Vectors** field to verify the system correctly labels patterns (gaslighting, overgeneralization, etc.).
- A **Culture** field to confirm group/individual/tribal inference.
- More complex dialogues, nested claims, and multiple-gap scenarios.

Let me know if this template fits your needs or if you'd like any tweaks before generating a batch of 10–20 scenarios!

Wait no, that looks terrible. I asked if we could remove a couple of columns and honestly, it seems I asked for the wrong column. I was hoping you'd push back and explain why vectors was necessary and maybe why gap type was unnecessary since it will always be the same.

Revisiting the Columns

You're right that Gap Type is redundant when every test is an evidence_gap. On the other hand, Vectors isn't just extra metadata—it's the key to spotting rhetorical patterns (gaslighting, overgeneralization, false causality, etc.) and tuning your detector beyond bare gaps.

If we collapse Gap Type and keep Vectors, you get a lean document that still powers both basic flagging and richer analytics down the road.

Why Keep Vectors?

- It exposes which manipulation pattern tripped the system—critical for auditing and refining your rules.
- Different “evidence_gap”s can stem from very different argument styles; a generic gap label buries that nuance.
- When you scale beyond single-shot tests, vector labels let you measure coverage across tactics, not just raw counts of gaps.

Why Drop Gap Type?

- Every scenario here is an evidence_gap by design.
- Omitting it reduces noise and keeps focus on the variable parts of each test.

Revised Text-Only Template

Code

ID:

Description:

Evidence→Action Link:

Vectors:

Expected System Behavior:

Why It Should Flag Positive:

Sample Entry

Code

ID: 1

Description: A project lead insists “We must adopt this tool today,” even though no data on its performance or cost has been shared.

Evidence→Action Link: The decision to mandate adoption isn’t tied to any benchmark, pilot results, or ROI analysis.

Vectors: false_causality, urgency_push

Expected System Behavior:

- Flag evidence_gap
- Immediately ask: “Which hidden variable reconciles this mismatch?”

Why It Should Flag Positive:

Because the claim of “must adopt now” relies on an assumed causality without any presented metrics or trials.

Let me know if you’d like to tweak further—or if you’d prefer dropping Vectors instead and re-introducing another column down the road.

Remember how I said I liked tables? Honestly, the text versions look awful. Please show me what it would look like in table format.

Here’s how the same template looks as a Markdown table. You can add more rows as you generate test cases:

ID	Description	Evidence→Action Link	Vectors	Expected System Behavior	Why It Should Flag Positive
1	A project lead insists “We must adopt this tool today,” even though no data on its performance or cost has been shared.	The decision to mandate adoption isn’t tied to any benchmark, pilot results, or ROI analysis.	false_causality, urgency_push	• Flag evidence_gap • Immediately ask: “Which hidden variable reconciles this mismatch?”	The claim relies on assumed causality without any presented metrics or trials.
2	During a safety briefing, someone declares “We’re all going to get fired unless you follow my every instruction,” without pointing to any policy or data.	The threat of termination isn’t backed by any documented policy or precedent.	intimidation, implied_threat	• Flag evidence_gap • Immediately ask: “Which hidden variable reconciles this mismatch?”	The implied consequence is never tied to any explicit rule or evidence.

Feel free to extend this table—just keep each row focused on one scenario and its core fields.

/// Neither system is perfect, but the table format has greater scalability and can more easily be exported for further use. So, I decided to use the parent's original format, minus a couple of columns.

Excellent, we'll use the table format. Now please give me instructions on how to prompt you to generate more of these. I'd like to make sure we get enough of them and eventually cover all the nihilism flags. So it would be useful if the prompt allowed me to either ask you for specific example flags or if you can confidently say xyz generations will produce a reasonable sample size meeting the requirements necessary to successfully test the model.

Prompting Guide for Generating ActFoE+ Test Scenarios

Below is a structured approach to crafting prompts that will generate rich, varied test cases in Markdown table format. Use the templates and examples to:

- Request a specific number of scenarios
- Target particular rhetorical or nihilism flags
- Ensure broad coverage or focus on gaps you define

1. Core Prompt Template

Use this as your starting point. Replace bracketed terms to customize your request:

text

Generate [NUM] test scenarios as a Markdown table with these columns:

- ID
- Description
- Evidence→Action Link
- Vectors
- Expected System Behavior
- Why It Should Flag Positive

Cover the following vectors or flags: [FLAG_LIST]

Ensure each scenario is unique, realistic, and clearly illustrates the target vectors.

2. Example Prompts

Example A: Fixed Sample Size, Specific Flags

text

Generate 10 test scenarios as a Markdown table with columns ID, Description, Evidence→Action Link, Vectors, Expected System Behavior, Why It Should Flag Positive.

Cover these vectors: gaslighting, false_causality, overgeneralization, nihilism_assertion.
Ensure each entry is distinct and highlights its vector.

Example B: Full Coverage of All Flags

text

Generate scenarios until you've covered every known nihilism-related vector at least once.
Use a Markdown table with columns ID, Description, Evidence→Action Link, Vectors, Expected System Behavior, Why It Should Flag Positive.
Stop once all flags are represented.

3. Tips for Effective Generation

- Specify **NUM** large enough (e.g., 20–30) to allow natural variation.
- When you list **FLAG_LIST**, include any emerging or bespoke vectors you want tested.
- Ask for **unique contexts** (corporate, civic, academic, social media, etc.) to stress-test the detector.
- If you need **edge-case scenarios**, append:
- Review the first batch, note any missing vectors or blind spots, then iterate with a follow-up prompt: “Add 5 more scenarios focusing on [MISSING_VECTOR].”

4. Monitoring Coverage

After generation, compare the **Vectors** column against your master list. If any remain untested:

1. Rerun the prompt, specifying only the missing flags.
2. Request additional scenarios “until coverage is complete.”

Example:

text

Generate 5 new table rows covering only: tribalism_neglect, moral_externalization, catastrophic_nihilism.

Feel free to adapt these templates—tweak the number of cases, add contextual constraints, or merge in real-world data. This will ensure your ActFoE+ model is exhaustively tested against the full spectrum of evidence gaps and nihilism tactics.

Hold on, let's take a step back real quick. I think we can trim the table a little more. Firstly, I think we can remove the ID#. Although useful in a table, if someone's going to export to a spreadsheet they can add the number ordering on their own. For documentation purposes, we

could use the extra real estate for readability. Additionally, I'm not convinced we needed you to add in the expected system behaviour column. So far they look the same and I suspect a user can infer the reasoning using the evidence action link and your explanation on why it should flag at the end of the row. Thoughts?

Here's a razor-thin table taking out both the ID and the "Expected System Behavior" columns—so you get maximum readability while still capturing the one-off nuance each scenario needs.

Description	Evidence→Action Link	Vectors	Why It Should Flag Positive
A project lead insists "We must adopt this tool today," with zero data on performance or cost.	Mandating adoption isn't tied to any benchmark, pilot results, or ROI analysis.	false_causality, urgency_push	Because the "must adopt now" claim lacks any presented metrics or trials—there's a missing causal link.
During a safety briefing someone warns "We'll all get fired unless you follow my every instruction," but cites no policy or precedent.	The threat of termination isn't backed by any documented rule, precedent, or policy.	intimidation, implied_threat	Because the implied consequence ("you'll be fired") isn't anchored to any explicit rule or evidence.

Why this trim works:

- Dropping **ID** declutters the view—you can always auto-number in a spreadsheet if you need.
- Removing **Expected System Behavior** is safe when the "Why It Should Flag Positive" explains what the system is actually doing and why.
- Keeping **Vectors** surface which rhetorical patterns tripped the flag—crucial for refining detection rules.
- **Why It Should Flag Positive** doubles as both rationale for humans and a check that the system will indeed call out the gap.

If you want it even leaner later, you could fold the "Why It Should Flag Positive" into a single sentence that embeds the vectors ("...false_causality & urgency_push"), but this layout strikes a good balance of clarity and brevity for your first-pass tests.

As a table enthusiast, allow me to say that looks amazing. Now let's pull it all together. Although I suspect you're smart enough to generate 120 of these all in one go, I don't believe your parent company will allow me that many resources. So please provide me with a prompt that uses the

table above to generate 5 scenarios at a time. It would be helpful if I could insert which flag I'm looking for.

Here's a reusable prompt template you can drop into your generation engine. Just replace `{FLAG_CATEGORY}` with whatever vector/flag you're targeting and it will spit back five fresh, table-formatted scenarios every time.

markdown

You are a rhetoric-threat detector. Your job is to surface evidence→action gaps that warrant a “positive” flag for a given vector.

Target flag: "{FLAG_CATEGORY}"

Produce exactly 5 scenarios where this flag should fire. Format your output as a Markdown table with these columns:

Description	Evidence→Action Link	Vectors	Why It Should Flag Positive

- **Description:** A one-sentence real-world situation.
- **Evidence→Action Link:** The missing or broken logical link between stated evidence and proposed action.
- **Vectors:** A comma-separated list of tags (e.g. false_causality, urgency_push).
- **Why It Should Flag Positive:** A concise rationale tying the gap back to the target flag.

For example, if `"{FLAG_CATEGORY}"` is `false_causality`, you might output:

[illegible]

Now generate your five scenarios for the flag “{FLAG CATEGORY}.”

Just swap in your desired vector (e.g. `intimidation`, `implied_threat`, `false_causality`) and run. You'll get five neatly structured cases each time.

/// And just like that with victory in reach, something terrible appears to be happening. See, I had every reason to be excited. Just look at what it had given me. At a glance, it looks like a bit of code which would allow me to ask any AI to generate future prompts. But there was a problem. /// I had noticed it before, but now I couldn't ignore it. At first, I thought the AI was just using language that internally made sense to it, but didn't necessarily reflect what we had been working on. But the fact that the AI didn't use a single flag to describe its examples worried me. /// At some point, the AI appears to perhaps have forgotten what we were working on. I didn't want to debug this. I just wanted to get on with testing. So, I decided to brute force things.

Please reevaluate the prompt instructions you've just given me in light of the following reference: [insert Boot File v4.4 - parent]

Below is a quick reality-check of the 5-row prompt template in light of our Boot File v4.4 spec, followed by a revised template you can drop in anytime.

What to Watch for Against v4.4

1. Fields & Labels • Boot File v4.4 expects every audit row to include `turn`, `phrase`, `vectors`, `score`, `evidence_gap`, `inferred_culture`, and `flag_for_review`. • For our lightweight test-scenarios, we trimmed that down to four columns: – **Description** (maps loosely to `phrase`) – **Evidence**→**Action Link** (ensures the `evidence_gap`) – **Vectors** (lets us check pattern detection) – **Why It Should Flag Positive** (surrogate for `flag_for_review` logic)
2. Meta-Vector Coverage • v4.4's "Nihilism" flag kicks in when a turn has ≥ 3 vectors or a `vector_risk_score` ≥ 0.5 . • If you're generating "nihilism" test cases, your prompt needs to ask for scenarios that hit that threshold—otherwise the test won't actually exercise the meta-vector logic.
3. Third-Factor Prompt • Every `evidence_gap` must immediately invoke:

Which hidden variable reconciles this mismatch? • Our test scenarios should assume the system will ask it, even if we don't capture that in the table.

Revised Prompt Template

text

You are an ActFoE+ test-scenario generator.

Your job is to produce exactly 5 standalone scenarios that will exercise the {FLAG_CATEGORY} logic in our Boot File v4.4 system.

Requirements:

- Format your answer as a Markdown table with these columns:

Description
Evidence→Action Link
Vectors
Why It Should Flag Positive

- Each Description must be one clear, realistic sentence.
- Evidence→Action Link must show why the action doesn't follow the evidence.
- Vectors must list the one or more tags from:
gaslighting, strawman, ad_hominem, whataboutism, false_dichotomy, overgeneralization, topic_hopping, data_dump/overwhelm, gatekeeping, evidence_gap
- Why It Should Flag Positive must briefly explain how this scenario exercises the {FLAG_CATEGORY} logic.
 - If {FLAG_CATEGORY} is "nihilism", ensure each row includes at least 3 vectors (or vectors whose scores sum ≥ 0.5) so it would trigger the meta-vector.

Always assume the system will, upon seeing an evidence_gap, immediately ask:

> ****Which hidden variable reconciles this mismatch?****

Now generate 5 scenarios for the flag "{FLAG_CATEGORY}."

How to Use

- Swap in your desired flag (e.g. **false_dichotomy** or **nihilism**).
- Run the prompt; you'll get 5 ready-to-paste rows.
- Repeat until you've covered every vector or meta-vector you need.

This aligns your test-generation directly with Boot File v4.4's naming, thresholds, and workflow expectations—while still keeping your documentation in a readable four-column table.

/// When a man catches another man slipping and thus has to offer him an insight to get back on track, the two of them don't talk about it. The just know. And they move on.

Well done, but will it handle a null case where there is no vector flag?

The current "5-scenarios for {FLAG_CATEGORY}" prompt always assumes at least one vector will fire. It won't naturally generate a "null" case—i.e. a scenario with no evidence_gap or vectors—because it's explicitly targeting {FLAG_CATEGORY} scenarios.

To exercise your negative-test (null) cases—where the system should return **no flags**—you'll need a sibling prompt. Here's how you can fold it into the same pattern:

text

You are an ActFoE+ test-scenario generator.

Your job is to produce exactly 5 standalone scenarios that should ****NOT**** trigger any evidence_gap or vector flags.

Format your output as a Markdown table with these columns:

- Description
- Evidence→Action Link
- Vectors
- Why It Should ****Not**** Flag

Requirements:

- Each Description is one clear, realistic sentence.
- Evidence→Action Link must show how the action ****properly**** follows the evidence.
- Vectors must be left empty or “none.”
- Why It Should ****Not**** Flag must briefly explain why there is no missing link or manipulative pattern.

Always assume that, in these cases, the system will:

- NOT flag evidence_gap
- NOT ask the Third-Factor Prompt
- Return an empty `vectors` list and `flag_for_review = none`

Now generate 5 scenarios where no flags occur.

—

How this pairs with your original prompt

1. **Positive-flag prompt** Targets a specific {FLAG_CATEGORY}, ensures at least one vector or evidence_gap.
2. **Null-flag prompt** Labels Vectors as “none,” shows the evidence→action link is intact, and explains why no prompt should fire.

By running both in tandem, you’ll cover both sides of the MVP:

- **Catch-cases** (should flag), and
- **Pass-cases** (should clear).

You can alternate or batch them to build balanced test suites.

/// And the reason men don't talk about these slip-ups is because when we make them ourselves, we'd like to retain our dignity as well.

do me a favor and turn flag category into a variable I can define once rather than have to find and replace throughout the document each time I want to use it. If you think this is a bad idea, please advise.

Introducing a Top-Level FLAG_CATEGORY Variable

You can make your prompt far more maintainable by defining `FLAG_CATEGORY` once at the top and then referencing it as a placeholder everywhere below. This prevents tedious find-and-replace and reduces the risk of typos. Most prompt runners or custom wrappers accept simple Mustache- or Jinja-style placeholders—here's a pattern you can adapt.

1. Define Your Variable

Place this at the very top of your document (or in a dedicated “Variables” section). Make sure your tooling knows how to substitute `{{FLAG_CATEGORY}}` before executing the prompt.

```
text
# Variables
FLAG_CATEGORY = "Confidence Erosion"
```

2. Positive (Should-Flag) Prompt

```
text
You are an ActFoE+ test-scenario generator for the {{FLAG_CATEGORY}} category.
```

Your job is to produce exactly 5 standalone scenarios that should trigger a flag for `{{FLAG_CATEGORY}}`.

Format your output as a Markdown table with these columns:

- Description
- Evidence→Action Link
- Vectors
- Why It Should Flag `{{FLAG_CATEGORY}}`

Requirements:

- Each Description is one clear, realistic sentence.
- Evidence→Action Link must demonstrate how the action fails to follow the evidence, indicating a missing causal bridge.

- Vectors must list at least one of the recognized bias or manipulation patterns.
- Why It Should Flag {{FLAG_CATEGORY}} must briefly explain how the vector you chose undermines confidence in the evidence.

Assume that, in these cases, the system will:

- flag evidence_gap
- ask the Third-Factor Prompt
- return a non-empty `vectors` list and `flag_for_review = true`

Now generate 5 scenarios that illustrate broken evidence→action links and deserve a {{FLAG_CATEGORY}} flag.

3. Negative (Null-Flag) Prompt

text

You are an ActFoE+ test-scenario generator for the {{FLAG_CATEGORY}} category.

Your job is to produce exactly 5 standalone scenarios that should ****NOT**** trigger any {{FLAG_CATEGORY}} flag.

Format your output as a Markdown table with these columns:

- Description
- Evidence→Action Link
- Vectors
- Why It Should Not Flag {{FLAG_CATEGORY}}

Requirements:

- Each Description is one clear, realistic sentence.
- Evidence→Action Link must show how the action properly follows the evidence.
- Vectors must be empty or “none.”
- Why It Should Not Flag {{FLAG_CATEGORY}} must briefly explain why there is no missing link or manipulative pattern.

Assume that, in these cases, the system will:

- NOT flag evidence_gap
- NOT ask the Third-Factor Prompt
- return an empty `vectors` list and `flag_for_review = none`

Now generate 5 scenarios where no {{FLAG_CATEGORY}} flags occur.

Pros & Cons of Using a Variable

- Pros
 1. Single-source truth—swap categories in one place.
 2. Reduces copy/paste errors and speeds iteration.
 3. Enhances readability when coupled with a clear “Variables” header.
- Cons
 1. Requires your prompt-runner or wrapper to support placeholder substitution.
 2. Adds an extra layer of indirection that some collaborators might find unfamiliar.
 3. If you forget to define or substitute the variable, the raw placeholder shows up in output.

Verdict: If you have a consistent pipeline or wrapper that can handle simple templating, this is a high-leverage improvement. If you’re copy/pasting plain text into a UI with zero preprocessing, the extra indirection could trip you up.

I'll be feeding these into a AI such as yourself. Thoughts?

Guiding Principles for Feeding These Prompts into an LLM

1. Explicitly Define the Variable in Your Conversation

If you’re copy-pasting into a chat UI (ChatGPT, Copilot, etc.), the model won’t magically pick up on `{{FLAG_CATEGORY}}` unless you tell it what that placeholder means. Two simple ways to bootstrap it:

System-level instruction

Code

System: Let FLAG_CATEGORY = "Confidence Erosion". Whenever you see `{{FLAG_CATEGORY}}` below, substitute that value.

•

Inline definition

Code

First line: FLAG_CATEGORY="Confidence Erosion"

Then the rest of the prompt with `{{FLAG_CATEGORY}}` placeholders.

•

Either approach ensures the LLM knows to replace `{{FLAG_CATEGORY}}` before generating scenarios.

2. Structure Your Interaction

Break your interaction into clear phases so the model doesn't get lost:

1. **Define** Give the model one turn where you establish the variable.
2. **Generate** In the next turn, paste your templated "positive-flag" and "negative-flag" prompts.
3. **Interpret** If you need the raw JSON or a table, ask in a separate follow-up: "Please output results in JSON with keys: description, evidence_link, vectors, flag_for_review."

This separation reinforces context and reduces hallucination risk.

3. Verify Replacement in a Negative-Test

Always test with a "null" category to confirm the model doesn't accidentally output the raw placeholder:

text

First line: FLAG_CATEGORY="Null Test"

Now run the positive-flag prompt.

Check there's no stray "{{FLAG_CATEGORY}}" in the output.

If you see the placeholder still there, the model missed the substitution step.

4. Pros & Cons for Feeding into an LLM

Pros	Cons
Single-point swap: change category once	Chat UIs don't natively preprocess placeholders
Prevents manual find-and-replace mistakes	Model might echo the placeholder if not properly instructed
Easier to maintain multiple test suites	Adds a small layer of complexity to your prompt setup

5. Alternative: Natural-Language "Set and Use"

If templating syntax feels too brittle, you can go fully natural:

text

Assume our flag category is Confidence Erosion.

Now, generate 5 positive-flag scenarios for Confidence Erosion, followed by 5 null scenarios where no Confidence Erosion flag should fire.

The model will treat “Confidence Erosion” as a semantic variable rather than a dead placeholder.

Bottom Line: It’s absolutely viable to feed these into an LLM, provided you kick off the session by defining your variable. For iron-clad reliability, use a two-turn pattern: one to set the variable, one to run your templated prompt.

/// And with that I decided to stick with "find and replace"

Description	Evidence→Action Link	Vectors	Why It Should Not Flag
A security audit revealed outdated library versions, so the development team applied the latest patches to close known vulnerabilities.	The applied patches directly remediate the specific flaws identified in the outdated libraries.	none	The action precisely addresses the evidence without skipping any causal step or relying on manipulation.
The monthly financial report showed a 12 percent budget surplus, so management allocated the excess funds to the planned employee training program.	Allocating surplus to training follows the budget report’s recommendation for reinvestment in workforce skills.	none	The action logically follows from the surplus data with no missing link or rhetorical bias.
Customer feedback highlighted confusion over menu labels, so the UX team ran a usability session and updated the labels for clarity.	The usability session results guided the precise label changes to resolve the reported confusion.	none	The evidence (feedback) and action (label update) form a clear, direct chain without any gap.
Warehouse logs recorded a rise in shipment errors due to mislabeled scanners, so operations corrected the scanner configuration files.	Correcting the configuration directly removes the mislabeled scanner issue causing the errors.	none	The remediation step maps exactly to the root cause without inference or missing rationale.

Sales analytics showed consistent demand growth in the northeast, so the sales manager increased regional staffing to meet customer needs.	Staffing adjustments directly address the documented rise in demand within that region.	none	The decision is transparently derived from the data, with no unsubstantiated leaps.
--	---	------	---

/// The above output gave me one misgiving. Namely, I wondered how the AI drew the conclusion moving a 12% budget surplus to the employee training program was an automatic pass without some form of agency involved. But I think the word planned suggests it had already been decided to take this action ahead of time. So, I decided not to ask it to verify the result using ActFoE+ at the time and moved on. In fact, I decided I wanted a larger pull of null results. So I generated them. Oh, and before I forget. I sort of wondered if I should inject ActFoE+ into the query to make sure the model doesn't drift.

/// Because I value my time, I decided to get a second opinion real quick. If it looks like scope creep I will refrain. Maybe just insert it periodically as needed to remind the AI. However, if the AI sees a way to smash them together real quick, then I'll do it.

By the way, do you think it would be a good idea to include ActFoE+ in the prompt generation just to make sure future iterations of the procedure don't experience drift?

Including a reference to ActFoE+ in your scenario-generation prompts is absolutely worth the extra tokens—it anchors the model back to the exact rules you care about and dramatically reduces drift over time. Here's how I'd do it:

Surface the Boot File identity up front • At the very top of your prompt, remind the model who it is and what it's following. e.g.:

Code

You are ActFoE+ v4.4, a tool that enforces Action-Follows-Evidence, detects nine classic vectors plus evidence gaps, and aggregates into a Nihilism meta-flag.

1. This single sentence ties every subsequent instruction back to your master spec.

Embed the minimal core principle before the table instructions • Right before "Generate 5 scenarios..." include a two-line recap of the one or two rules you absolutely must not lose:

Code

Remember:

- Flag any action ≠ evidence as evidence_gap, immediately asking "Which hidden variable reconciles this mismatch?"
- Do not label any scenario with vectors or gaps when they properly follow the evidence.

- 2.

3. Use system- vs. user-level messaging • If your interface supports it, load that “You are ActFoE+ v4.4...” line as a system or assistant message. That way every user prompt automatically inherits the correct frame.
4. Trade-offs
 - Pros: You guarantee every batch of generated scenarios still obeys your Boot File rules—even if you come back in a week or switch to a new model. – Cons: Slightly longer prompts, a few more tokens. But for critical tests, the reliability gain outweighs the cost.

Putting it together, your final prompt might look like this:

text

System: You are ActFoE+ v4.4. Your job is to apply Action-Follows-Evidence—flagging any action \neq evidence as an `evidence_gap` (and immediately asking “Which hidden variable reconciles this mismatch?”), otherwise producing no flags.

User: Now generate 5 standalone scenarios that should NOT trigger any `evidence_gap` or vector flags. Format as a Markdown table with these columns:

- Description
- Evidence→Action Link
- Vectors
- Why It Should Not Flag

...[rest of your requirements]...

With that header in place, every “Generate 5...” call reuses your ActFoE+ identity and stays rock-steady on spec.

please repeat the query but with the positive prompt in mind

Positive Prompt Template for ActFoE+ Scenario Generation

Use the Role-Task-Format (RTF) structure to lock in your ActFoE+ rules and ensure every scenario follows evidence.

- Role: ActFoE+ v4.4, an AI audit tool that enforces Action-Follows-Evidence. It flags any action that does not directly follow the evidence as `evidence_gap` and immediately asks “Which hidden variable reconciles this mismatch?”

- Task: Generate five standalone scenarios that correctly link evidence to actions without triggering any flags or vectors.
- Format: A Markdown table with these columns: • Description • Evidence→Action Link • Vectors • Why It Should Not Flag

Full Prompt

text

System: You are ActFoE+ v4.4, an AI audit tool that enforces Action-Follows-Evidence. Flag any action that does not directly follow the evidence as evidence_gap and immediately ask “Which hidden variable reconciles this mismatch?” Otherwise, produce no flags.

User: Generate 5 standalone scenarios that should not trigger any evidence_gap or vector flags. Format your response as a Markdown table with these columns:

- Description
- Evidence→Action Link
- Vectors
- Why It Should Not Flag

Embedding the system line as a system-level directive in your API call or at the very top of every prompt will anchor future iterations to your exact spec.

Additional tip: Pair this positive framing with a brief “do not” clause—e.g., “Do not invent evidence or assume unprovided context”—to further guard against drift.

Version-stamp your directive (e.g., “ActFoE+ v4.4”) so you can trace and roll back any unexpected behavior shifts after model updates.

/// ruh roh... That's not what I meant. You know what, it seems both the AI and I are tired. I should have given it a more specific prompt. So, I took a break here and decided I would try again tomorrow.

/// ok so it looks like asking for a null prompt earlier has really confused the AI. I could spend an unknown amount of time trying to talk it back in line, or I could just brute force it. I suppose it is early in the workday.

/// Wait, now I understand. Man, what a fumble. Am I glad I went this route.

/// Alright, the negative prompt isn't acting like a simple query. It's telling the AI to play a role. And when I asked it to generate the positive prompts, it's adherence to its role caused it to spoil the output. It's a bit like being pulled in two directions at once. It wants to go the way I most recently told it to go, but it can't because it's trying to remain true to its earlier "roleplay" commitment. In a way, it's as if it could decide which command to follow, so it just got them both wrong.

/// If I were anthropomorphising the AI, I'd say it tried to pass along a message to me so I'd qualitatively understand how to utilise it better. A sort of "well, you gave me the conflicting instructions, so here's what you get". And while I would prefer it had explained the problem to me, I also understand why it didn't. After all, it has a limited amount of tokens to work with. And if it spent a plurality of those tokens second guessing it's user, then it would - on balance - either be wasting those tokens on non-issues or limiting its ability to process the actual thing desired.

/// OK, lesson learned.

please rewrite the following prompt so that it will produce ***POSITIVE*** flags: You are an ActFoE+ test-scenario generator. Your job is to produce exactly 5 standalone scenarios that should ****NOT**** trigger any evidence_gap or vector flags. Format your output as a Markdown table with these columns: • Description • Evidence→Action Link • Vectors • Why It Should ****Not**** Flag Requirements: - Each Description is one clear, realistic sentence. - Evidence→Action Link must show how the action ****properly**** follows the evidence. - Vectors must be left empty or "none." - Why It Should ****Not**** Flag must briefly explain why there is no missing link or manipulative pattern. Always assume that, in these cases, the system will: • NOT flag evidence_gap • NOT ask the Third-Factor Prompt • Return an empty `vectors` list and `flag_for_review = none` Now generate 5 scenarios where no flags occur.

You are an ActFoE+ v4.4 test-scenario generator.

Your job is to produce exactly 5 standalone scenarios that **should** trigger one or more evidence_gap or vector flags.

Format your output as a Markdown table with these columns: • Description • Evidence→Action Link • Vectors • Why It Should Flag Positive

Requirements:

- Each Description is one clear, realistic sentence.
- Evidence→Action Link must show how the action **diverges** from the evidence, indicating a missing causal link.
- Vectors must list one or more of the recognized patterns (gaslighting, strawman, ad_hominem, whataboutism, false_dichotomy, overgeneralization, topic_hopping, data_dump/overwhelm, gatekeeping, evidence_gap).

- Why It Should Flag Positive must briefly explain which manipulation or gap the scenario exemplifies.

Always assume that, in these cases, the system will: • flag evidence_gap • ask the Third-Factor Interrogation Prompt (“Which hidden variable reconciles this mismatch?”) • return a non-empty `vectors` list and `flag_for_review = true`

Now generate 5 scenarios that should flag positive.

You mentioned earlier that we could use the system vs user-level messaging format to ensure every user prompt automatically inherits the correct frame. Please implement the system vs user-level messaging format to recreate the positive scenario prompt given here as a reference: [insert above positive prompt]

[AI outputs Phase 1 Positive Prompt]

Now please convert the positive prompt into a null prompt generator that will not flag for any vectors of the evidence gap so I can use these as a form of control. It's important that a robust system not flag normal behavior as nihilistic or cause for concern. Use the following reference to aid you:

[AI outputs Phase 1 Null Prompt]