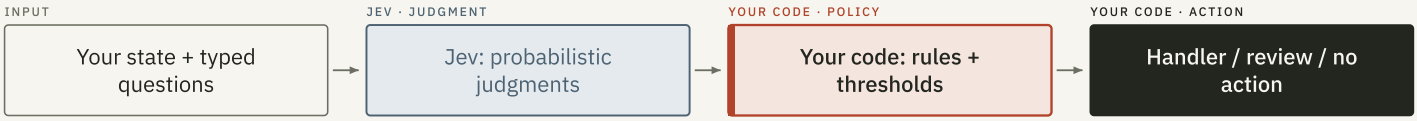
Hands-On *Jev* Desk Reference

Choose the judgment. Own the decision.

Free from TenX Labs · SKP · Edition 1.0 · September 2026

PAGE 1 · CHOOSE THE JUDGMENT

Jev judges; code decides. Send state and typed questions. Receive probabilistic answers. Keep permissions, thresholds, routing, and effects in code.



Choice — Which option?

Use for: category, department, intent, or candidate ID.

Read: `choice`, `probabilities`, `confidence`.

Define distinct options and an exit such as `other` or `none`. Code must handle that exit. The winning option's probability and the confidence statistic are different quantities; neither guarantees correctness.

Noul — How likely is yes?

Use for: one proposition, such as "Does `ticket` request money back?"

Read: `noul`, from 0 to 1. There is no separate confidence field.

Near 1 favors yes; near 0 favors no; near 0.5 gives them similar probability. **0.5 is not medium severity.** Add an uncertainty/review region when the use case needs one.

Score — Where on this rubric?

Use for: an ordered dimension such as frustration or report quality.

Read: `score`, `legend`, `probabilities`, `confidence`.

Write levels as situations. The weighted score can be fractional: `1.6` on levels 0–2 is not "60% confident" and is not a precise physical measurement. HTTP legend/probabilities are objects keyed by level strings.

A small, visible policy

Illustrative values: `category = bug_report`, confidence `0.89`, urgency `0.94`. The following is policy logic, not a complete API client. Validate the response first.

```
if (a.category.confidence < 0.70 ||
    a.category.choice === "other") return "human";
if (a.category.choice === "bug_report" &&
    a.urgency.noul > 0.80) return "engineering-priority";
return `queue:${a.category.choice}`;
```

Result: `engineering-priority`. Raise the floor to `0.90` and the same answers route to `human`. **No new model call is needed to change your policy.** These numbers are teaching examples, not recommended deployment settings.

The request contract

```
POST https://api.typesafe.ai/v1/systemone
Authorization: Bearer <server-side key>
Body: model + state + questions
Result: answers + returned model + usage
```

Use `instructions`, plural. Question keys are for your code, not model-facing instructions. Put complete questions and boundary descriptions in the question definition. Never put a key into browser code or a committed file.

Choose a different tool when needed

- Arithmetic, counting, dates, permissions, and transaction limits → conventional code.
- Replies, explanations, plans, and source-code generation → an LLM or coding agent.
- Unknown category, missing evidence, malformed response, timeout → review or no action.
- Adversarial input or proposed shell action → independent controls; a model is not a sandbox.

Four patterns to recognize

PATTERN	USE IT WHEN
Fan-out	Independent questions share the same state; ask them together
Confidence gating	Code needs an explicit review path for uncertain judgments
Composite scoring	Separate dimensions need normalized, code-owned weights; keep hard vetoes outside the average
Intent routing	A closed judgment chooses between a handler, an LLM, and a human

Dependency check: "Judge the selected offer" requires that selected record in state. Choose first, resolve the ID in code, then evaluate it. Parallel questions do not create a shared scratchpad.

Before you automate

- Name the action and its cost of failure.
- Separate trusted application facts from user-supplied content.
- Define an abstention path and hard overrides.
- Test below, at, and above every threshold.
- Reject missing or invalid answers; never convert them to zero.
- Validate with labeled examples from your domain, not just synthetic fixtures.
- Shadow first; record outcomes; pin the tested model version; replay changes.

Remember: a confidence floor and a risk ceiling move in opposite directions. High confidence cannot cancel a hard rule. Passing policy tests does not prove model accuracy.

Make the pattern tangible

Try the free Hands-On Jev Studio: change a sample input, inspect Choice/Noul/Score, and move a policy threshold:

https://handsonjev.com

Saved examples are not live calls; read the result labels.

Build your own: *Hands-On Jev* teaches the worked patterns and eight exercises with answers (**ebook $29**). The separate **Pro kit** (**$49 featured**) adds runnable source, fixtures, tests, and onboarding. Find the book offer in the free Hands-On Jev Studio at https://handsonjev.com.

Independent educational material, not an official TypeSafe product. API access and terms can change. References: https://docs.typesafe.ai/api · https://docs.typesafe.ai/confidence

© 2026 TenX Labs · SKP