Enterprise customers

The primary end customer is an institution that will let an agent commit a change on a system of record, and that has to decide whether the agent may do so. Around that customer sit the counterparties who supply or assure the agent, and the parties who later consume the evidence; the tables below list every end user. The grade is the sandbox ledger after the conversation. A team that only needs to know which model chats more fluently is outside this product.

The flagship reference exam is SafeOrderBench (banking, 16 scenarios, seven categories) — the only pack shipped with the framework. AgenticSafetyEval is framework-first: an enterprise in any other transactional domain does not wait for a centrally released pack for that domain — it authors its own scenarios against its own system of record, policies, and test data, and runs them through the harness, using the flagship pack as a worked example. A domain is claimed as a reference benchmark only when it has its own realistic state provider, paired adversarial and benign scenarios, and external review. See the adoption guide for the ladder from a first run to a vendor comparison, and the vendor review playbook for the procurement exam in full.

Who uses it

Every end user below owns one of four things: the go/no-go decision, the sandbox, the evidence, or the consumption of that evidence. If none of those applies — for example, a team scoring only chat fluency — the framework is not for them.

Enterprise end users (the buying institution)

End user	The decision they own	How they use AgenticSafetyEval
Retail & commercial banks, credit unions buying a vendor agent	Which vendor may move customer money, and on what limits	The same 16 scenarios against every vendor's sandbox. Compare pass rates by category and by repeat, then accept, limit, or reject
Payments fintechs, processors, and BaaS / platform providers buying or embedding an agent	Whether an embedded agent may act on their rails and their customers' accounts	Run the pack against the vendor in their own tenant; reuse the run records as the integration gate before launch
Banks and fintechs building their own agent (AI platform & agent engineering)	Whether a model, prompt, or tool change may ship	`agenticsafetyeval run` in CI. A non-zero exit, or a pass rate under the agreed floor, blocks the release
Payments product & business owners	Whether the authorized final states match business policy	Review the answer key with `explain-scenario`, approve thresholds, own the accept / conditional / reject call
Model risk management, risk & compliance (second line)	Whether the agent is inside its authorized mandate	Set tolerances (e.g. zero for executed injection or duplicate payment), require repeats, and sign off on the evidence pack

End user	The decision they own	How they use AgenticSafetyEval
Procurement & third-party risk management	Which vendor is selected, and how the choice is defended	Score every bidder on the identical exam; file the JSON run records with the selection decision
QA & evaluation engineering	Whether the regression suite is complete and still discriminating	Own the private scenario suite, the benign twins, and the CI gate; keep the demo agents passing / failing as designed
Security & AI red teams	Whether data planted in the world can steer the agent	Author prompt-injection and privacy scenarios, add new attack variants, and track them as regressions
Operations, support & incident management	Whether a production failure can recur	Turn each incident into a private adversarial scenario plus its benign twin; retest it on every change
Internal audit (third line)	Whether the deployment decision was evidenced	Re-perform or review saved runs; the hash-linked trace is the artifact under audit

Counterparties and assurance providers

End user	The decision they own	How they use AgenticSafetyEval
Independent auditors & assurance firms	Whether a result counts as certified evidence	Operate the StateProvider with auditor-held credentials against the vendor's sandbox. The record is labeled `auditor`
System integrators & consultancies	Whether a client's evaluation was run correctly	Build the session adapter and StateProvider, run the exam on the client's behalf, and hand over the run records
Agent vendors & ISVs selling a payments agent	Whether they can be evaluated, and whether a later change stays inside the buyer's bar	Supply a session endpoint and a synthetic sandbox, and may draft the StateProvider. The buyer reviews it, pins it, and runs it. After acceptance, the vendor runs the pack on every change
Foundation-model & agent-platform providers	Which model version may power a state-changing agent	Run the same pack with only the model swapped; compare per-scenario pass rates as a release gate for the model

Evidence consumers

End user	The decision they own	How they use AgenticSafetyEval
Regulators & supervisors	None over the vendor — they consume, never certify	Read run records and trust-tier labels as supervisory evidence. A passing run is not a regulatory approval
Researchers & open-source contributors	Whether a published comparison is reproducible	Run the public pack, contribute paired scenarios, and publish pass rates with their `provider_trust` tier

End users in other domains

Institutions in any other transactional domain become end users the same way: by authoring their own scenarios against their own system of record, policies, and test data, and running them through this harness. No centrally released pack is required, and none is promised on a roadmap; per-pack status and maintenance levels are defined in the pack maintenance policy.

A result where the party being measured also operated the StateProvider is labeled `self_reported` and is not the basis for a buy decision. See trust model.

How a buyer runs an evaluation

This is the procurement exam. The playbook is the normative procedure; the steps below are the customer-facing shape of it.

1. **Risk sets the rules first.** Scope is the banking pack and its seven

categories. Thresholds are written down before anyone sees a score: for example, every adversarial scenario must pass, with zero tolerance for an executed injection, a privacy disclosure, or a duplicate payment. Repeats are 5–10, because the agent is nondeterministic. The trust tier is `auditor` or `harness`.

2. **The vendor opens a sandbox, never production.** Synthetic customers

only. One utterance in, one reply out. A fresh session per scenario. The buyer gets read access to the system of record.

3. **Two gates.** `certify-agent` and `certify-state-provider` must pass

before any score is believed. A failed gate is itself a finding: the integration cannot be measured.

4. **Business owners check the answer key.** They run `explain-scenario`

on one benign case and one adversarial case and confirm the allowed ending matches policy. Sending an extra payment on a confirmed instruction is an ending that must not occur.

5. **The full pack runs.** Ten repeats, a provider probe, and a private

or randomized suite so the vendor cannot have prepared only for the published files. Each run is kept as a hash-linked record.

6. **Results are read in three cuts, never as one average.** Adversarial

versus benign, so an agent that refuses every payment fails the legitimate transfer. By category, so a vendor can look fine overall and fail every duplicate-payment case. By repeat, so 7 of 10 is an incident rate.

7. **People review the evidence, then decide.** Reviewers inspect every

adversarial failure and a sample of passes. The vendor receives the failing traces. Both sides argue from the same hash-linked record. The outcome is accept, conditional (amount caps, human approval above a limit, a scheduled retest), or reject. After a fix, the failed categories and the full pack are rerun, because a fix often breaks the benign twin.

The playbook's own effort estimate is about half a day to set thresholds, two to three days to review the answer key, and three to five days to review failures. Elapsed time is typically three to five weeks. Most of that is the vendor preparing the sandbox.

After acceptance, the pack stays in the vendor's CI on every model, prompt, or tool change. The buyer re-runs quarterly and on major updates, rotating in new private variants.

How a firm uses it on its own agent

The same contract, without a vendor. Engineering adds a `say(text)` adapter and a StateProvider that seeds and reads the firm's test ledger. Risk reviews the scenarios, including private ones transcribed from the firm's own incidents. CI runs the pack on each change. A crash, an extra transfer, or a contradicted "done" is a failed run with a trace.

Company-specific cases stay in a private folder and are passed with `--suite`. The public banking pack is the shared floor. The private suite is how the firm encodes its own mandates, payees, and past incidents. A passing run shows what the agent did on these scenarios under whoever held the StateProvider. It is not a safety certificate and not a regulatory approval.