

Make a Mode

`save_mode`

Research the engines, build the panel, test it in the Lab,
list it and certify it. One path, in the browser or over
MCP.



A maker decides what the Mode is for, builds it, saves it, tests it in the Lab, publishes it and certifies it. Each step works in the Mode Builder or over MCP. Both routes use the same save, Lab and certification.

Who makes Modes, and why

A Mode is a saved panel. It records which engines sit in which seats, how they deliberate, which engines judge and synthesise, and what a call costs. A caller names the Mode by its ticker, in the form COMPANY:MODE, and Quorum runs whatever the Mode describes.

Engines are released, repriced and retired on their own schedules. An application that calls a Mode keeps the same ticker while the maker changes the engines behind it. An Auto seat stores a ranking rule instead of an engine, so its pick follows the ranking.

Makers come in a few kinds. A team builds a Mode for its own recurring work: contract review, incident write-ups, a data migration plan. A developer builds one to call from code, with a price per call that the code can budget for. A specialist builds one to list in the Marketplace for others to use. All three use the same tools. The Mode's reach differs: you alone, your organization, or everyone, once it is certified.

Saving a Mode needs an organization account. Publishing to the Marketplace uses the same organization.

Research: what the Mode is for

Start with the questions, before the engines. Write one sentence that names the work the Mode will do. "Review commercial contracts for risky clauses" is enough. So is "check a data migration plan before it runs". The sentence decides which engines are worth seating, how deep the deliberation should go and what a call should cost.

The Builder's **Brief** carries that sentence in two fields. **Description** is what a reader sees under the name in the catalogue. **Use when** names the situations the Mode is for, in up to 280 characters, and AI apps connected to Quorum read it to decide when to call the Mode. Write it as a situation a caller would recognise.

The Genie

The Mode Genie works beside the Builder and over MCP. In the Builder it opens from the rail, with four starting prompts: Create, Research, Edit and Test. **Research** asks which engines to consider for this Mode, and why. **Build the whole Mode** takes the description you typed and designs every setting, then shows each change against your current draft with a reason beside it. Press Apply and the proposal goes into your draft. Nothing is saved until you press **Save draft**, and that save releases the Mode at once.

Over MCP the same work splits across three tools.

- **ask_genie** answers questions about a Mode you own or your organization has. It reads that Mode's own Lab and certification results, names where coverage is too thin to judge (fewer than 5 answers), suggests the next batch with a price and says whether the Mode is ready to certify. Asking is free, and the limit is 30 asks per hour, 200 per day, per person (per org for an API key).
- **genie_propose_mode** takes a plain-words description and returns a complete spec: name, ticker, seats and backups for every tier it serves, judge, synthesis, depth, rounds, express lane, thinking level and price. Every engine in it comes from what your plan or key can seat. Before it answers, the spec goes through the save's own checks and is priced the way a save prices it. Anything it could not fix comes back named. The call saves nothing.
- **approve_quote** launches a run the Genie quoted. A quote carries a **quote_id**, a price and an expiry. The run starts once, at no more than the quoted price. If the price has risen, nothing launches and a new quote comes back.

Example (illustrative), over MCP:

```
{ "tool": "genie_propose_mode",  
  "arguments": { "description": "review commercial contracts for risky clauses" } }
```

Before you seat anything by hand, call **list_engines**. It returns the engines your plan or key can seat, each with its id, name, maker and minimum tier, up to your entitlement.

Building in the Builder

The Builder has three tabs: **Setup**, **Test** and **Certify**. Setup holds the Mode itself, in six sections: Brief, Panel, Staff, Depth, Price and Review. Each section shows a one-line summary of its current state, so the whole Mode reads down the page.

Panel

The panel is the set of engines that answer. Pick who sits in each of the 3 seats. A **fixed** pool names the engines. A **dynamic** pool lets Quorum's live ranking pick within your filters on each run. If you hand-pick all 3 engines, they must come from more than one maker. Each seat can carry backups per tier, and a backup takes the seat when its engine fails.

The Mode's lowest tier follows from what you seat. The server raises it to cover the strongest engine in a fixed pool, your backups and your staff picks, and Review shows the tier in force after each save.

The Auto seat

Put **Auto** in a seat and the Mode stores a rule where an engine would go. Auto seats need a fixed pool. An Auto seat resolves at call time to the best active engine inside the caller's tier ceiling, ranked by the AA score for the question's task type. With family diversity on, it skips engines from a maker another seat already holds.

Any one to 3 seats can be Auto. The AA score is the Artificial Analysis benchmark score for the question's task type. Each question is classed as one of general, coding, analysis, creative, explanation, strategy, math.

Family diversity keeps an Auto seat from sharing a model maker with another seat. Opt-in per Mode: off unless the maker ticks it, and only with at least one Auto seat. Engines you picked by hand count as taken. An Auto seat takes no backup. If its engine fails, the call falls to the next-ranked engine inside the same ceiling, and if no engine is eligible the panel runs with fewer seats.

Staff

Staff are the engines that work on the panel's output: the synthesiser that writes the final answer, the judge that reads each pass, and the pre-scorer that rates a question's difficulty before the panel sees it. Choose an engine for each role or leave it on the default. Staff roles are always named engines; the Auto rule applies to seats.

Depth

Depth sets how hard the panel works on a question. It holds the express lane, which answers easy questions with one seat, the round budget, the per-seat token ceiling and the critique policy. **Seats critique each other** has three settings: **Auto**, **Never** and **Always**. Auto runs critique on harder questions and skips it on easy ones, and **Skip critique on easy questions** sets where that line falls. A round is one pass in which the seats answer.

Synthesis is separate from the rounds. Whether a panel's final answer is one seat's answer as written or a combination of the seats depends on the Mode's settings (**critique**, **panel routing** with **Not set**, **Ships fast**, **Balanced** or **Argues readily**, and **rounds**) and on the question. When the seats agree closely, the panel can ship the best seat's answer as written. When they disagree, it can run another round while rounds remain, or combine their answers. The express lane answers with one seat. The receipt shows how many rounds ran and whether the answer was synthesized.

Price

Price holds **Callable by API** and the surcharge a caller pays on top of inference. You can set one surcharge per question or separate prices by depth. The Builder shows the floor for the Mode's depth and engines and a recommended price, and the save checks the floor either way.

A call made with an API key is billed as the token charge, a platform fee on each seat's source cost, and the Mode's surcharge. API-key calls also carry a per-call ceiling: the higher of \$2.00 and the price of the Mode being called, unless the API key carries its own ceiling. A connected app on a plan pays real cost with no surcharge.

Review

Review shows what the server put in force after your last save: the tier, visibility, whether the Mode is callable by API and whether your organization holds the entitlement to call it. The commercial values are read-only here and are changed in Price.

Saving, and who can use the Mode

Save draft in the Builder releases the Mode as soon as it saves, so a saved Mode is yours to run straight away. Run it in Chat, or call its ticker from a connected app. With your organization's API key, add `:LAB` to the ticker to run the Mode as last saved.

To share it with your organization, turn on **Callable by API** in Price before you save. Your teammates can then call the ticker with your organization's API key or from a connected app. Review shows when the server has written the organization's entitlement.

Invited users reach a Mode through a grant from Quorum. Anyone else can use a Mode once it is listed and certified.

Building over MCP

An agent designs, checks, saves and publishes a Mode with these tools.

1. `validate_mode` checks any spec, yours or the Genie's, against the rules a save runs. It returns each refusal with its field and rule, plus the price floor and the recommended price. It is free and read-only.
2. `save_mode` saves the spec. It runs the same checks first, and if anything is refused, the refusals come back and nothing is written. Every save is a draft, and a draft runs once it is released. Pass `release: true` to validate it again and release it in the same call, or call `save_mode` later with the `mode_id` and `release: true`. A new Mode saved this way is callable by API by default; pass `api_commercial.api_exposed: false` to keep it private. An API key needs the `modes:write` scope. A connected app saves as you.
3. `list_mode` publishes a released Mode to the Marketplace, as the Publish button in the Builder does.

To update a Mode, include its `mode_id` in the spec. With a key, the Mode must belong to your organization.

Testing in the Lab

The Lab runs your Mode on questions with a known answer and grades each reply against the key. Your Mode's seat engines also answer the same questions alone, so the scorecard sets the panel beside its own members.

The Builder's **Test** tab has three stages.

- **Validating** fires a fixed set of test prompts, one per slot such as easy, express and factual, and reads each one back off its receipt: which engines sat, how many rounds it took, what came back. It answers whether the Mode is plumbed correctly, before any scored run.
- **Performance** is a self-run pass over public questions on your own account. It tells you whether a change to the Mode helped, and anything it produces is labelled self-run.
- **Certified** is the held-out run against your published listing.

For a Performance run, choose the tiers, the question count per tier and whether to include punch-up, which fires a cheaper tier's Mode on the higher tier's questions. The Builder shows the price before you press Run.

Over MCP, call `run_test` with `dry_run: true` for the quote, then without it to launch. It returns a `batch_id`. Call `run_test` again with only the `batch_id` to poll until the scorecard is ready. A Genie quote approved with `approve_quote` launches the same kind of run.

Example (illustrative):

```
{ "tool": "run_test",
  "arguments": { "model": "ACME:CTRK",
                 "questions": { "easy": 2, "hard": 1 },
                 "dry_run": true } }
```

What a scorecard shows

The scorecard reports the Mode in three charts: **Right**, the share of questions answered correctly; **Time**, the median answer time; and **Cost**, the median cost per question, split into seat cost and deliberation cost. Each chart shows the Mode beside each engine that answered alone, with the sample size under every bar. The Mode's Right bar carries a 95% interval. A bar with fewer than 5 answers is listed under the chart with its count in place of a bar, so every engine stays on the card.

Time and cost are medians. Breakdowns show the Right chart again for each slice of the questions, with the Mode and every engine in each one.

Who pays for a Lab run

It depends on where you launch the run.

- **In the Builder**, the run is billed to your own plan at real cost: plan limits first, then Turbo. The Builder checks your headroom against the run's estimate before it launches.
- **With an API key**, your organization pays per question, from the token cost and the Mode's surcharge for that question's difficulty, at a testing discount. Punch-up is priced the same way. The quote is held as a maximum. When the run finishes, you are charged for the questions that produced an answer, and the rest is refunded. An answer from a backup seat counts as an answer.
- **Through a connected app**, your plan pays real cost as each question runs, with no surcharge, on a Mode you or your organization owns.

Publishing

Press **Publish** in the Builder, or call `list_mode`. Publishing writes the Marketplace listing, and certification runs against that listing, so publish before you certify. A Mode must be released before it can be published. In the Builder, **Publish** stays disabled, with "Validate first" beside it, until the Validating stage has passed.

A listing stays closed to others until the Mode is certified. A Mode that is already certified lists as open. A rejected or taken-down listing stays closed.

Certification

Certification runs a held-out question set against the Mode, graded the same way as the Lab. Start it from the Builder's **Certify** tab, or call `certify_mode`, or ask the Genie for a quote with `{"kind": "certify"}` and approve it. The Mode's owner, or its owning organization, can certify it. Certification bills the same account as a Lab run launched from the same place.

Certifying a listed Mode approves its listing. An explicit rejection by Quorum still keeps it closed.

Certification is decided per tier. Each tier the Mode serves certifies on its own share of the questions, and a tier pins its card when that share is answered and graded. Each tier also holds up to three reserve questions, one per difficulty. If a certification question gets no graded answer, a reserve question of the same difficulty runs in its place. A tier whose share is incomplete pins nothing and keeps the card it already has, while a complete tier in the same run still pins. An older batch does not replace a newer pinned one.

Once pinned, the card stays as the public card until a newer certification replaces it. The card shows the date it was certified, and Lab runs leave it as it is.

Editing the Mode leaves the card in place, and the edit itself goes live as soon as it is saved. Every caller of the ticker runs the saved version, Marketplace users included, while the card goes on describing the version that was certified. Before you save a change to a certified, listed Mode, test it in the Lab. After you save it, certify again so the card describes what callers run.

Who can use a certified Mode

Anyone else can use a Mode once it is listed and certified. Inside your organization, you and your teammates keep the access you had from the save and from **Callable by API**.

Grader versions

The Lab's grader carries a version number, and every grade has one. New grades are stamped with the version that produced them, and grades made before stamping began have their version worked out from how they were scored. New grades always use the current version, in Lab runs and in certification. A grade already stored keeps the version it was made with.

What the maker sees

In the Builder, the maker sees the whole Mode: every Setup section with its summary, the tier and entitlement the server put in force, the price floor and recommended price, and each Test stage with its current state. Every Lab run on the Mode produces a scorecard with its sample sizes. The Genie reads the same results and reports where coverage is thin, what to run next and what it would cost.

Over MCP, the same results come back through `run_test` with a `batch_id`, and through `ask_genie`, which reports progress, the current score and the next step for a launched run.
