AI Deliberation & the Router

POST /v1/estimate

Find out what a question is before you commit to answering it — how hard, what kind, and whether a panel would help at all. Free on every call. The branch stays in your code.

What is in here

Sections 03 to 06 are the argument and the contract you are coding against. 07 is the branch itself. 08 to 11 are the router's working life, with code and data on every page. 12 and 13 are wiring and the loop. If you are short of time, read 05 and 07.

EXHIBITS

03

What Quorum is, in one page.

Quorum is a deliberation platform. One question goes to several frontier models from different labs at once. They answer independently, critique each other, and are judged — and what comes back includes **where they disagreed**.

If you already call one model well, the distinction that matters is this: calling the same model twice gives you the same reasoning again, more politely. Independent models fail in different places. That is the whole mechanism, and it is also the whole cost.

A SEAT

One frontier model on the panel. Three seats is the standard panel; **seat_responses** is a variable-length array. Quorum represents seats as shapes rather than naming vendors.

A MODE

The configuration of a deliberation — which engines sit, how many rounds, how deep. Which mode you call is one of your two server-side levers. §06.

A RECEIPT

The record of a completed deliberation: per-seat model and judge score, difficulty signal, cost, latency. It is what you tune the router on. §11.

WHERE IT RUNS

An OpenAI-compatible REST API at **quorum.dog/v1**, a hosted MCP server on the same key, and Quorum's own surfaces. §12.

THE CLAIM, STATED NARROWLY

Quorum does not claim a single model is bad. It claims that a single model, however capable, **cannot be its own second opinion** — it has one way of being wrong, and it applies that way consistently. Panels exist for the questions where that matters, and this paper is about finding those questions in traffic rather than by hand.

ONE THING TO BE EXACT ABOUT

The model your product already calls is **the caller, not a seat**. It stays where it is, holding your context, and hands a question to a panel of independent models — none of which is the one it was about to answer with. In the code that follows, `singleModel()` and `panel()` are two different destinations, not two parts of one panel.

WHAT ONE DELIBERATION ACTUALLY DOES — FOUR THINGS, INSIDE A SINGLE CALL

01 ANSWER

Each seat takes the question independently, without seeing the others.

02 CRITIQUE

Each seat then reads the others and argues against them.

03 JUDGE

The result is judged, and those scores land on the receipt you tune the router with.

04 SYNTHESISE

One answer is written — with the disagreement reported rather than smoothed away.

All four run before anything comes back — which is where the wait lives, and why the next page is about not spending it on every turn.

04

`deliberate()`

A panel is the wrong default for every turn.

A Quorum deliberation runs several frontier models, has them critique each other, judges the result and writes a synthesis. That is not free and it is not fast.

Put it behind every turn of a product and one user in ten waits a minute and a half, while you pay panel prices for **three models to agree that Paris is the capital of France**.

The fix is not a faster panel. It is not convening one unless the question earns it.

Which means the interesting number on this page is not any of the percentiles. It is **0.9 seconds** — roughly what it costs to ask what a question looks like before you decide whether to spend the other twenty-eight.

MEASURED ON PRODUCTION · END TO END

CALL	LATENCY	WHAT IT IS
`estimate`	~0.9 s	Classifies and prices a question. Does not run it. Free on every call.
`deliberate p50`	28.5 s	Median of 3,845 real deliberations.
`deliberate p90`	88.2 s	One turn in ten is at least this long.
`deliberate p99`	229.3 s	The tail you have to design a timeout and a UI around.

N=3,845 · MEASURED ON PRODUCTION · DATED 2026-08-20 · TRANSCRIBED, NOT ROUNDED

THIS IS NOT A COST ARGUMENT

For work that is genuinely judgement, the price of a call is not the deciding factor and nobody should pretend otherwise. Budget ceilings belong in the router because a router runs thousands of times a day, not because deliberation is expensive relative to being wrong.

IT IS A FIT ARGUMENT

Most traffic in most products is execution: lookups, extraction, formatting, recall. A panel has nothing to disagree about on those, so it returns consensus slowly. The waste is not the money. It is the ninety seconds spent manufacturing agreement.

SO: ONE FREE CALL FIRST

`POST /v1/estimate` classifies and prices a prompt without running it, and returns `billed_usd: 0` on every call — by design, not as a trial allowance. The next section is what it hands you.

Everything after this page answers one question: which turns are worth 28.5 seconds. Asking which ones deserve it costs nothing.

05

POST /v1/estimate

What the free call hands back.

Quorum can tell you what a question looks like — how hard, what kind, what it would cost, and whether deliberating on it is likely to help — before you commit to asking it. What you do with that is your call, not ours.

REQUEST, AND THE RESPONSE

```
curl https://www.quorum.dog/v1/estimate \
  -H "Authorization: Bearer $QUORUM_API_KEY" -H "Content-Type: application/json" \
  -d '{"model":"quorum-standard","messages":[{"role":"user",
       "content":"Should we migrate off the monolith this quarter?"}]}'

{ "request_id": "0f5c…", "model": "quorum-standard", "mode_key": "standard",
  "depth": "deep",
  "difficulty_score": 0.78,
  "task_type": "strategy",
  "deliberation_value": {
      "verdict": "likely_helps",
      "basis": "hypothesis",
      "evidence": "per-task pattern, n=36 — not a measured effect"
  },
  "estimated_price_usd": 0.15,
  "billed_usd": 0 }
```

Exhibit 01 — The classification call, and the response in full. Field names and the free-on-every-call contract are the API's; the identifier and figures shown are illustrative of one question.

BRANCH ON THIS ONE FIRST

deliberation_value — `likely_helps`, `likely_hurts` or `unknown`. It is a better gate than the score, because it answers a different question: not *how hard is this* but *does arguing about it help*. A hard question whose answer is a verbatim artifact — a quoted clause, a config block — is still the wrong thing to escalate, and `likely_hurts` is what fires when a synthesis pass would only rewrite it.

Read its own basis, though. It ships with `basis: "hypothesis"` and an evidence line saying n=36 — a per-task pattern, not a measured effect. Treat it as a strong prior that shows its working, and keep §08's override list for where being wrong is expensive.

AND THE FOUR THAT DO THE REST

difficulty_score — 0 to 1, the main dial. Set the threshold from your own distribution, not ours. §09.

task_type — what kind of question it is, which is how you exempt whole categories in one line.

depth — `light`, `medium` or `deep`.

estimated_price_usd — what the real call would cost, so a budget ceiling can sit in front of it.

06

Whose decision this actually is.

Worth being exact, because it determines what you have to build.

On the API, every deliberation runs at full depth. The request pins the deliberation tier internally. There is no per-call dial that makes Quorum deliberate less on an easy question, and you should not go looking for one.

You have two server-side levers: **which mode you call** — a mode's configuration decides how its panel behaves — and **quorum.max_cost_usd**.

Be precise about the second one. It is a ceiling on what you are *billed*, applied after the fact. The deliberation still runs. You still get the answer. The response comes back with `finish_reason: "cost_cap"` and Quorum absorbs the difference.

It protects your budget. It does not prevent the work and it does not save you the wait — so it is not, and cannot be made into, an escalation control.

THE BUDGET LEVER, AND WHAT IT DOES NOT DO

```
{ "model": "quorum-standard",
  "messages": [ … ],
  "quorum": { "max_cost_usd": 0.25 } }

-> the panel still convenes, you still wait, the answer still arrives
-> and if it went over:  "finish_reason": "cost_cap"
```

So the escalation decision is not a setting inside Quorum that you tune. It is a branch in your code: for this turn, do I call a single model, or do I call a panel? Quorum's job is to hand you the facts that branch needs, priced at nothing.

WHICH HAS THREE CONSEQUENCES

The threshold is yours. It belongs in your codebase, under review, next to your other product constants. And you should expect to move it as you learn — sections 09 to 11 are the three jobs that follow from owning it.

ON THE CONSUMER PRODUCT IT WORKS DIFFERENTLY

People using Chat, Panel or Studio choose Express, Foundation or Frontier next to the composer, and that choice does change how much deliberation happens per question. That control belongs to the person asking, not to a router. If you are building on the API, the equivalent control is the branch in §07 — and unlike theirs, yours has to make the call without a human looking at the question.

07

Write the branch once.

Classify, branch on your own rule, keep a budget ceiling. The thresholds below are **placeholders** — they are the shape of the decision, not values anyone is recommending.

Read it in order and the priority is the argument: does a panel help, then is it mechanical, then can we afford it, and only then how hard the classifier thought it was. The score is the last question, not the first.

ROUTER.JS · THE WHOLE THING

```
const QUORUM = 'https://www.quorum.dog/v1';

async function estimate(messages, model = 'quorum-standard') {
  const r = await fetch(`${QUORUM}/estimate`, { method: 'POST',
    headers: { 'Authorization': `Bearer ${process.env.QUORUM_API_KEY}`,
               'Content-Type': 'application/json' },
    body: JSON.stringify({ model, messages }) });
  if (!r.ok) return null;                // fail OPEN -- §10
  return r.json();
}

async function route(messages, { maxUsd = 0.25 } = {}) {
  const est = await estimate(messages);
  // Classification unavailable: do not block the user on our router.
  if (!est) return singleModel(messages);

  // Does a panel HELP -- a different question from "is this hard".
  const dv = est.deliberation_value?.verdict;
  if (dv === 'likely_hurts') return singleModel(messages);

  // Questions that cannot really be disagreed about.
  const mechanical = ['lookup','formatting','extraction'].includes(est.task_type);
  const tooExpensive = (est.estimated_price_usd ?? 0) > maxUsd;

  // Only now, the score -- and 0.65 is a PLACEHOLDER, not a recommendation.
  const worthIt = dv === 'likely_helps' || est.difficulty_score >= 0.65;
  if (worthIt && !mechanical && !tooExpensive) return panel(messages);
  return singleModel(messages);
}
```

Exhibit 02 — The router branch, complete. Threshold, ceiling and exempt task types are illustrative placeholders.

SEND THE SPECIFICS, NOT A SUMMARY

Pass the real payload through to `panel()`. A panel argues about what it is given: a summary in, three summaries out.

SURFACE THE SPLIT

Where the seats disagreed is the finding. Do not flatten it into one paragraph on the way to your UI — that discards what you waited for.

LOG THE REQUEST_ID

Store it against the turn. Without that join you have receipts and no way to tell which ones mattered, and §11 does not work.

08

Overrides, and where a router does not pay.

The classifier is a model reading a prompt. It will occasionally call something light that deserved a panel. Where being wrong is genuinely expensive, escalate on **task_type or your own signal** rather than on the score alone — and write those rules above the score, not below it.

01	**Always escalate, whatever the score**	Anything irreversible, anything a user will be asked to defend to someone else, anything you would want a receipt for six months later. These are properties of your product that a classifier reading one prompt cannot see.
02	**Never escalate, whatever the score**	Your own mechanical task types, verbatim retrieval, anything where the correct answer is an artifact rather than a judgement. This list is short and you will already know what is on it.
03	**Let people ask**	A user-invoked panel bypasses the router entirely. It is also the cheapest labelled data you will get about where your threshold is wrong — every manual escalation is somebody telling you the branch missed one.
04	**Nothing else**	Resist a third and fourth signal. Every override you add is a rule you now have to keep true, and the tuning loop in §11 gets harder to read with each one.

WHERE THIS DOES NOT PAY — STATED PLAINLY, BECAUSE A GUIDE THAT ONLY ARGUES ONE WAY IS MARKETING

YOU ARE LATENCY-BOUND

Classification is a round trip. Far cheaper than a deliberation, but not zero — and on a stream-every-token chat UI it is a real addition.

TRAFFIC IS UNIFORMLY HARD

If nearly everything scores above the threshold, the router adds a hop and diverts nothing. Measure first; it costs nothing to find out.

TRAFFIC IS UNIFORMLY EASY

Then you do not need a panel at all, and Quorum would rather you knew that than paid for one.

BEING WRONG IS COSTLY

Then the score alone is not enough. Rule 01 above exists for exactly this case.

09

THRESHOLD → FALLBACK → FEEDBACK

Pick it from your own distribution.

Run `estimate` across a week of real traffic with no branch attached. It costs nothing and changes nothing, and at the end of it you have the only distribution that matters.

Quorum publishes its own for comparison. It is not a recommendation and it is not a benchmark — if yours looks nothing like it, that is information about your product, not a sign you are holding it wrong.

BAND	WHAT IT USUALLY IS	QUORUM N=3,947	ONE PRODUCT N=4,102
< 0.35	Lookups, formatting, recall, mechanical transforms	45%	71%
0.35 – 0.65	Real questions with a defensible single answer	33%	20%
> 0.65	Judgement, trade-offs, ambiguity — where panels earn their cost	22%	9%
mean		0.456	0.284

Exhibit 03 — Two difficulty distributions. The Quorum column is measured across 3,947 classified questions on Quorum's own traffic, dated 2026-08-20. **The right-hand column is illustrative** — the shape a support-and-drafting product might find in its first week, shown so the comparison has something to be a comparison against.

READ THE DIFFERENCE, NOT THE COLUMN

The same threshold of 0.65 escalates 22% of Quorum's traffic and 9% of the illustrative product's. Same number, different products, an entirely different bill and an entirely different latency profile. That is the whole reason to measure rather than copy a value out of a guide — including this one.

A DEFENSIBLE WAY TO CHOOSE THE FIRST ONE

Decide what share of turns may wait, then read the threshold off your own curve at that share — rather than picking a number that sounds principled and discovering the share afterwards. Then hold it still long enough to collect receipts against it.

MEASURING IT · THE WHOLE JOB

```
// One-off and offline: costs nothing, changes nothing, ships nothing.
const hist = new Array(10).fill(0);
for (const messages of replayLastWeek()) {
  const est = await estimate(messages);
  if (est) hist[Math.floor(est.difficulty_score * 10)]++;   // decile
}
```

10

THRESHOLD → **FALLBACK** → FEEDBACK

Fail open, not closed.

If the classification call fails, answer the question with your single model rather than erroring or defaulting to the expensive path. **A router that breaks your product when one endpoint has a bad minute is a worse router than no router.**

Failing *closed* to the panel is the tempting mistake, because it looks like caution. It is not: it converts a classifier outage into a bill and a ninety-second wait on every turn, on exactly the traffic that least deserved one.

THREE FAILURE MODES, ONE BRANCH

```
const ctl = new AbortController();
const deadline = setTimeout(() => ctl.abort(), 1500);  // your number
try {
  const r = await fetch(`${QUORUM}/estimate`, { ...init, signal: ctl.signal });
  if (!r.ok) return null;          // non-2xx
  return await r.json();                // unparseable throws -> caught below
} catch {
  return null;                     // abort, DNS, TLS, parse -- all the same
} finally { clearTimeout(deadline); }
```

THE SAME MINUTE IN THE REQUEST LOG

```
14:22:07.118  estimate  200  0.91s  d=0.71  likely_helps   -> panel
14:22:31.402  estimate  200  0.88s  d=0.12  likely_hurts   -> single
14:22:44.900  estimate  200  0.94s  d=0.55  unknown        -> single
14:23:02.771  estimate  ---  deadline 1.50s exceeded       -> single  [open]
14:23:03.014  estimate  503  0.06s                         -> single  [open]
14:23:03.660  estimate  200  0.90s  d=0.81  likely_helps   -> panel
```

Exhibit 04 — The degraded path, in the request log. Illustrative lines; the estimate latencies around them are consistent with the ~0.9 s measured on production 2026-08-20.

PUT YOUR OWN DEADLINE ON IT

The call is roughly 0.9 s. Pick a ceiling you are willing to add to every turn, and treat exceeding it as a failure like any other. 1500 ms above is a placeholder.

NO RETRIES IN THE HOT PATH

A retry doubles the worst case to save a call you were about to skip anyway. The fallback is already correct; retrying is how a 0.9 s hop becomes a 3 s one.

UNKNOWN IS NOT A FAILURE

`unknown` is a verdict, not an outage — the classifier declining to guess. Decide once what it means for you and write it down; falling through to `difficulty_score` is the honest default.

The router is an optimisation, and a degraded turn is simply your product without it.

11

THRESHOLD → FALLBACK → **FEEDBACK**

Move it with receipts.

Every deliberation returns a `request_id`. GET `/v1/receipts/{id}` tells you which engines ran, how much they disagreed, and what it cost. That is the evidence the threshold moves on.

ONE RECEIPT

request_id	rq_8f2c41a9
mode	quorum-standard
difficulty	0.71
seat 1	judge 0.83
seat 2	judge 0.81
seat 3	judge 0.79
cost	billed, per call
latency	26.4 s

SHAPE OF THE RESPONSE · ILLUSTRATIVE VALUES

WHAT IT SAYS, AND WHAT YOU DO

Judge scores inside four hundredths of each other are the convergence tell. The seats did not fight; they agreed, slowly, at panel price. One receipt like that is noise. A run of them is a threshold set too low.

LAST 200 ESCALATIONS	N	SHARE
Panel split on at least one point	38	19%
Converged, no dissent recorded	**162**	**81%**

threshold	0.62 → 0.70
escalation rate	14% → 9%

ILLUSTRATIVE FIGURES

Exhibit 05 — A receipt, and the threshold change it drove. Receipt fields are the API's; every value here is illustrative of the pattern, not a record of a specific call. Quorum names seats by shape rather than by vendor on anything shareable.

RAISE IT WHEN

Receipts on your escalated traffic keep showing the panel converging immediately with no dissent. **You are paying for agreement.**

LOWER IT WHEN

Your users keep pushing back on answers that were routed to a single model. That is the same signal arriving from the other direction.

AND WHEN IT NEEDS TO LEAVE

A completed deliberation exports from Quorum as a full transcript PDF, a carousel, social cards or a short video — for when the argument has to travel outside your product.

STATED AS DIRECTION, NOT AS A FEATURE

Quorum does not run this loop for you. There is no hosted controller that watches your receipts and moves your threshold, and no store that links a deliberation to what your users did next — that join lives in your system, keyed on the `request_id` you logged in §07. It is named here because the receipt is what would make such a loop possible, not because there is something to switch on.

12 Connect it.

The router in §07 talks to the OpenAI-compatible REST API at **quorum.dog/v1**. The same key works against the hosted MCP server, which is how an agent reaches the same six tools without you writing the fetch.

Authentication is a static bearer token — your API key in the Authorization header. No OAuth, no app registration, no redirect URL, on either surface.

BOTH ENDPOINTS · AND A CHECK YOU CAN RUN BEFORE WRITING ANY CODE

```
https://www.quorum.dog/v1     REST, OpenAI-compatible
https://www.quorum.dog/mcp    MCP, Streamable HTTP

curl -sS -X POST https://www.quorum.dog/mcp \
  -H "Content-Type: application/json" \
  -H "Authorization: Bearer qk_..." \
  -d '{"jsonrpc":"2.0","id":1,"method":"tools/list"}'
```

SIX TOOLS ON THE HOSTED SERVER

TOOL	COST	WHAT IT DOES
`estimate`	Free	Classifies and prices a question without running it. The router's whole input.
`list_modes`	Free	Which panels your key is allowed to convene — the menu behind your first lever.
`deliberate`	Paid	Convenes the panel. The one that costs, and the one you wait for.
`get_receipt`	Free	Per-seat model and judge score, difficulty signal, cost, latency. §11.
`run_test`	—	Mode testing. On the hosted server since 2026-08-22.
`certify_mode`	—	Mode certification. Also hosted since 2026-08-22 — guides that still call these two local-only are out of date.

TWO THINGS THAT LOOK LIKE FAULTS AND ARE NOT

A **GET** to the MCP URL returns 405 — the server is Streamable HTTP, POST only, and that is the correct response rather than a broken endpoint. And the connection is stateless: every call stands alone, there is no session to keep warm and nothing to reconnect.

WHICH SURFACE FOR A ROUTER

REST, almost always. A router is code you own, on a path you are timing to the millisecond, and §07 is thirty lines. Reach for MCP when the caller is an agent choosing tools for itself rather than a branch you wrote.

13

The loop is the product.

Nothing in this paper is a setting. The classification call is free, the two server-side levers are a mode and a billing ceiling, and everything that decides whether a question gets a panel is **thirty lines in your repository**, under review, with a number in it that you are expected to move.

Which is the honest version of the product: not *always use a panel*, but **know which questions deserve one**.

Start before you write the branch. Replay a week of your own logs through `estimate` with nothing wired to the result. It costs nothing, it changes nothing, and it answers the only question worth answering first: whether your traffic has a hard tail at all.

If it does not, you have saved yourself a router. That is a good outcome and this guide will not argue with it.

THE THREE JOBS, IN THE ORDER YOU WILL DO THEM

01	**Threshold**	Measure your own distribution, decide what share of turns may wait, read the threshold off your curve at that share. Not off anybody else's. **§09.**
02	**Fallback**	Every failure of the classifier — non-2xx, deadline, unparseable — collapses to the single model. Never fail closed to the panel. **§10.**
03	**Feedback**	Pull receipts on what you escalated. Immediate convergence with no dissent means raise it; users pushing back on single-model answers means lower it. **§11.**

WHERE TO START

TODAY

One key, one real question out of your logs, one **estimate** call. Read the response.

THIS WEEK

Replay a week of traffic, plot the deciles, pick a threshold you can justify.

THEN

Ship the branch, log every **request_id** beside its turn, come back for the receipts.

Ship it with the threshold too high. An escalation you did not make is a slightly worse answer; an escalation you should not have made is ninety seconds of a user's attention spent watching three models agree.

WHERE TO GO NEXT

Pricing for what a call costs · API reference for the full endpoint contracts · Testing for how the efficacy numbers are produced · Help for what Express, Foundation and Frontier do on the consumer product.