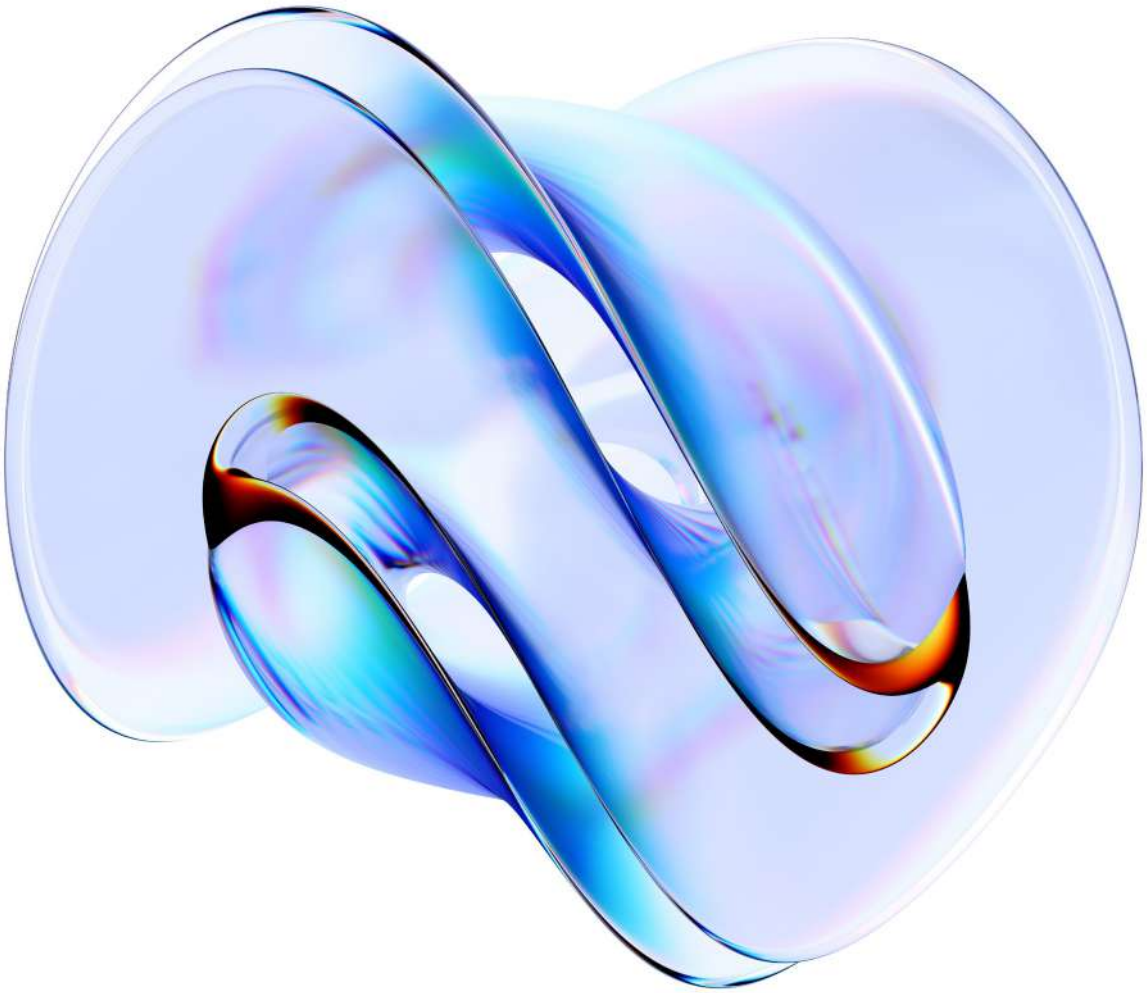
E2E Networks



Decisions you let it make

How to build tool-calling agents for dispute triage, collections and internal research on Indian infrastructure, in a way your risk committee will sign off.

Four rules before you start

Any AI system you put in front of a credit decision in India has to clear the same four rules. They aren't a compliance topic and they aren'tsomebody else's problem. They decide what you're allowed to architect, which decides what you can ship.

Rule 1

Consent is the only door

The DPDP Act gives you two ways to process personal data. Consent, or one of nine specific uses written into the law. That's the list. There's no "legitimate interest" route the way GDPR has one, so you can't write an assessment arguing that your interest outweighs the customer's.

For an AI team it lands in one place. If you train on customer data, the consent has to cover that use. Which means you need to be able to show whichdata went into which model. Not describe it. Show it.

The biggest penalty in the Act, up to ₹250 crore, attaches to failing to keep data secure. Not to paperwork. To architecture, access control and who else can reach your tenancy.

The obligations and the penalties commence on 13 May 2027.

Rule 2

The data stays in India

Payment data has to be stored only in India. If you process it abroad you have to delete it there and bring it back within one business day or 24 hours, whichever is shorter. Borrower data under the digital lending rules works the same way.

Teams tend to find the awkward part late. An inference call is processing. So if your model sits outside India, every scored transaction turns into something you have to prove you cleaned up. Not once. Once per transaction, for as long as the system runs.

An endpoint inside the country removes the proof, not just the risk.

Rule 3

You validate the model, not your vendor

RBI's draft guidance on model risk covers models you built and models you bought. Two lines in it change how you design.

You validate third-party models yourself, whatever certifications the vendor holds. Their audit report isn't your validation. And a human stays in command, meaning someone who can look at what the model did, disagree with it, reverse it, and switch it off. Override, suspension and deactivation are named requirements, not good practice.

Both are cheap if you designed for them. Close to impossible to retrofit.

The guidance was issued in draft in June 2026 and the final version is awaited.

Rule 4

Check whose rules you are under

RBI regulates banks and NBFCs. IRDAI regulates insurers.

If you're an insurer, none of RBI's AI material binds you. Not the FREE-AI framework, not the model risk draft. DPDP applies to you in full, the rest doesn't. Worth checking before you borrow an architecture note from a bank.

WHERE THE FOUR RULES LAND

In five places in any stack. Jurisdiction, meaning who can be legally compelled to hand over your data. Infrastructure, meaning where the GPUs physically are. Tenancy, meaning who else can reach them. Model lifecycle, meaning whether you hold the weights and the version history. And serving, meaning whether the off switch belongs to you. Four of those are engineering choices you can revisit. The first is a fact about your provider, and it's the only one you can't change later

Five things you will be asked to show

Models don't usually get blocked for being inaccurate. They get blocked because nobody can produce the evidence. These five are what a risk committee, an auditor or a supervisor actually asks for.

A data flow page

One page. Which data, for which purpose, under which consent, sitting where. Your DPO asks first and then everyone else does. Real means it was written while the system was still small, and updated when it changed. Not reconstructed in month nine.

Version lineage

This dataset version produced these weights, which are serving on this endpoint, traceable in both directions. Model validation asks for it, and so does anyone investigating a single decision. Real means a repository you can query, not a spreadsheet somebody keeps.

An evaluation record

What you tested before you deployed, against what baseline, with what result. Your risk committee asks, under the model risk guidance. Real means the same harness every time, kept for every version, including the ones you rejected.

An override path with a name on it

How a person reviews, challenges and reverses what the model decided. Anyone who has to defend a decision to a customer will ask. Real means a named owner, a queue they actually work, and a turnaround you have written down.

A deactivation procedure you have tested

How you switch the model off, and what the business does while it's off. The model risk guidance names it and your CISO will ask too. Real means tested at least once, in production, on purpose.

THE STOP RULE

If a build can't produce all five, it doesn't reach production. That isn't a compliance problem, it's a sign-off problem, and it stops the project just as dead. Usually around month eight, after the model has already shown it works.

Where the line sits. Our certifications cover the infrastructure and the platform. Your application's compliance posture stays yours. We can give you the architecture, the evidence and the contract terms to close the gap. We can't close it for you, and neither can anyone else.

The line between doing and proposing

An agent is not a new kind of software. It is a model, a set of tools you defined, and a loop. On this platform tool calling is a setting on the endpoint: you turn on automatic tool choice, pick a parser, and the agent frameworks everyone is talking about run against an ordinary OpenAI-compatible endpoint you own.

So the interesting question was never whether you can build one. It is what you let it do.

There is one distinction that everything else in this guide follows from. **What the agent may do, and what it may only propose.** A model that drafts a dispute response is a writing tool. A model that submits the representation has taken an action on your behalf, and every requirement in the previous section now applies to it.

Three patterns, in the order you should build them

Read only. It looks things up and answers. No writes, no money, no contact with a customer. An internal agent over your own policy documents and circulars is the ideal first one, because the worst failure is a wrong answer to a colleague who can check it.

Propose and approve. It drafts the action and a person commits it. Dispute packs, collections notes, a first-pass claim summary. Most agents in regulated finance should live here for a long time, and there is no shame in that being the destination rather than a stage.

Act within limits. It executes inside an envelope you defined: allow-listed operations, value caps, and only actions that can be reversed. This is where the governance work actually lands, and where most teams arrive before they have done it.

THE RULE WORTH PRINTING ON THE WALL

An agent's permissions are its tool list. Not its prompt. A prompt is a request and a tool list is a boundary, and the two are not interchangeable. You cannot prompt your way to a permission model. If the agent has a tool that moves money, it can move money, and the only question left is whether it does so on a good day or a bad one.

What people are building

Dispute and chargeback triage. Reads the case, gathers evidence from your own systems, drafts the representment. A person sends it.

Collections conversations. Early stage, tightly scripted, across Indian languages, with a hard route to a human whenever the customer asks or the conversation turns.

Internal policy and circular research. Retrieval over regulation and your own credit or underwriting policy. Read only, genuinely useful, and the safest place to learn what these systems do badly.

Servicing and status. Balance, status, statement, dispute progress. Useful, and the one most likely to be handed a customer's free text, which matters more than it sounds.

Nine decisions

Most of the architecture argument for agents is settled before you start, by the four rules above. Here is where each one lands.

THE DECISION	MOST TEAMS REACH FOR	WHAT SURVIVES THE FOUR RULES
What the agent may	Broad tools and a carefully written prompt	The smallest tool list that completes the job, with everything irreversible taken out of it. The prompt is advice. The tool list is the boundary.
Where the boundary is enforced	In the prompt, or in the model's judgement	In your own code, between the tool call and the system that executes it. At that moment the model is untrusted input. Validate every argument as though a stranger sent it, because in effect one did.
Which model, and where it runs	The largest frontier model, through an API	An open-weight model you hold, on a dedicated endpoint in your VPC, in India. Rule 2 applies to every call the agent makes, and a tool-calling loop makes many.
How many steps it gets	However many it needs	A hard step budget, a wall-clock timeout, and a cost ceiling per session. Loops are the normal failure in agent systems, not the exceptional one.
What it can read	Access to the whole knowledge base	Scoped retrieval, where the scope comes from the case or customer it is working on. An agent with broad read access is a data exfiltration path with a chat interface.
Where the human sits	Someone reviews the transcripts afterwards	An approval step on anything irreversible, and a hard escalation the moment a customer asks for a person. Human in command is the stated expectation, and a customer-facing agent cannot be a dead end.
What happens when it is down	The queue backs up until somebody notices	It degrades to the process you had before it, and the tool list can be emptied without a deploy. Emptying the tool list is your kill switch, and it is faster than rolling back a model.
How you will see what it did	Log the prompts and the replies	Log the tool calls. Every call, its arguments, its result, and which model version chose it. Nobody audits prompts. They audit actions.
How you will be billed	Per token, which for an agent is unpredictable by design	Dedicated capacity at a published hourly rate, with a per-session ceiling on top. One session can use ten times the tokens of another depending on how many steps it takes. A demo never shows you that.

IF YOU ONLY TAKE ONE ROW, TAKE THE FIRST

Everything else in this table is a consequence of it. Teams spend weeks writing careful instructions to keep an agent inside its lane, when the same afternoon spent removing three tools would have made the lane a wall. Start from the smallest tool list that completes the job and add reluctantly, because every tool you add is a permission you have granted and will have to justify.

Five things break, and only one is the model

These are the questions you will be asked around month four.
The point of this section is that you can answer them in month one.

Customer text is untrusted input

A collections reply, a dispute description, an uploaded document. Anything a customer writes eventually reaches the model, and a model reads instructions as readily as it reads content. Assume that at some point a customer will try, deliberately or by accident, to change what your agent does. The defence is not a better prompt. It is a fixed tool list, arguments validated in your own code, and scoped retrieval, so that the worst case is a strange conversation and not a wrong action.

The loop

Two tools that call each other. A retry that never terminates. A plan that re-plans. Latency, cost and confusion all arrive at once, usually on the day traffic is highest. A step budget and a wall-clock timeout are cheap to add on day one and awkward to add on day ninety.

Capability creep on upgrade

You swap the base model for a newer one that benchmarks better, and it is better. It also now chooses to call a tool the old model never reached for, in a situation you never tested. Re-run the same evaluation against the same tool list before you swap, and keep both results. A model upgrade is a permissions change even when nobody edited the permissions.

The transcript is not the audit trail

People assume the conversation log is the evidence. It is not. It shows what was said, not what was done, and those two things diverge exactly when it matters. The tool call log is the evidence. If you kept only transcripts, you will discover this during an investigation, which is the worst possible time.

The agent quietly becomes the process

It starts as an assistant. Then people stop doing the manual path. Then the manual path stops working, because the queue it fed dried up and the person who ran it moved on. Now the fallback you were relying on is a document rather than a capability. Run the pre-agent process deliberately, on a schedule, and confirm it still works.

Only the third of these is really about the model. The rest are about the boundary around it: what the agent can be told, how long it can run, what you kept, and whether the thing you fall back to still exists. Agents fail at their edges, and the edges are yours to build.

Running it

Everything above runs on ordinary platform capability.
No custom work, no professional services engagement, and nothing waiting on a roadmap.

What you need from the platform

- **Tool calling as a setting on the endpoint,** so an agent runs on the same infrastructure as everything else and is not a separate product with its own bill and its own limits.
- **Open-weight models you hold and version,** so a model upgrade is a decision you make on a date you chose. On a managed API the model can change underneath a running agent, and that is a permissions change nobody approved.
- **Endpoints inside your own network,** attached to your VPC, so every call in a multi-step loop stays in your perimeter and in the country.
- **Retrieval against your own store,** scoped per case, so the agent reads what it needs and nothing else.
- **Speech models alongside language models,** so a voice agent is one pipeline on one cluster and not three vendors and a latency budget you cannot meet.
- **Dedicated GPU nodes at a published hourly rate,** billed at a fixed amount, so an unpredictable workload does not become an unpredictable bill.
- **Metrics and logs you can pull into your own monitoring,** not a dashboard somebody has to log in and read.

On billing

Agents are the worst case for variable pricing, and it is worth being blunt about why. The token use of a single session depends on how many steps the model decides to take, and that varies by an order of magnitude between an easy case and a hard one. You cannot forecast it from a pilot, because pilots run on easy cases.

Dedicated capacity at a fixed rate turns the unpredictable part into a known quantity, and a per-session step budget caps what is left. Together they are the difference between a cost you can put in a business case and a cost you can only explain afterwards.

On observability

Log the tool calls, not the prompts. Every call, its arguments, its result, the model version that chose it, and the session it belonged to. That single log is your audit trail, your debugging tool and your evidence in an investigation, and it is much smaller than storing every conversation.

Three signals matter more for agents than the usual ones. The distribution of which tools get called, because a shift there is the first sign the model has changed its mind about something. Steps per session, because that is where cost and loops both show up. And the escalation rate, because an agent that never escalates is not being careful, it is being ignored.

Thirty, sixty, ninety

Days 1 to 30

Build the read-only agent. No writes, no money, no contact with a customer. Internal research over your own policy documents and circulars is the right first one. Measure steps per session and how often the answer holds up when a person checks it. Exit: a working read-only agent, and an honest accuracy number from colleagues who tried to break it.

Days 31 to 60

Add exactly one tool that writes, behind an approval step. Build the tool call log before you need it. Set the step budget and the cost ceiling. Then empty the tool list in production, on purpose, and watch what happens. Exit: one agent that proposes, a tool call log, and a kill switch you have actually pulled.

Days 61 to 90

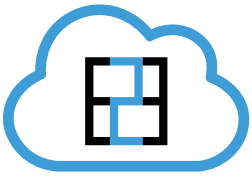
Put it in front of a real workflow, with a person approving anything irreversible. Write the agent into your model inventory the way you would a model: its version, its tool list, its limits, its owner, its off switch. Take it to your risk committee with the evaluation and the tool call log. Exit: one agent live inside a boundary you can describe in a sentence.

Two numbers this guide does not print

Steps per session and cost per session would make it complete. Both depend on your tools, your model and the difficulty of your cases, so they are yours to measure and not ours to assert. The first thirty days above exist to get you both, on your own work, before you commit to anything.

COME AND TALK TO US

Bring your volume and your latency budget. Those two numbers, plus your compliance constraint, are enough to sketch this architecture properly. We will do it on a whiteboard in about fifteen minutes.
Booth no. A2, Global Fintech Fest. Jio World Centre, Mumbai.



E2E Networks