When AI Agents Retry:

Building Reliable Event-Driven Workflows

Krishna Kandi

Senior Software Engineer



DevFest Troy 2026

October 24, 2026

A completed action, an uncertain agent

Invoice INV-1042: approved USD 250 payment request

Provider: ACCEPTED

Workflow: VERIFIED

Agent observation: UNKNOWN

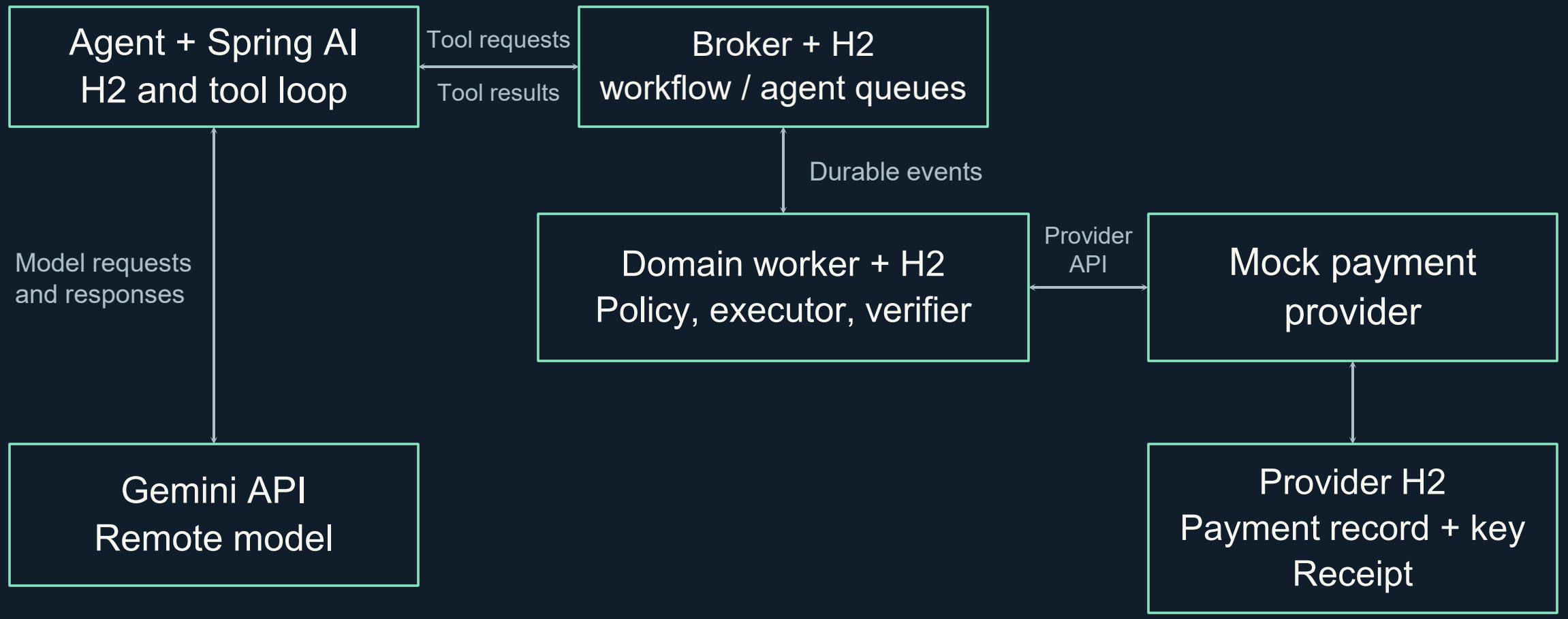
The agent paused before saving and acknowledging its completion event

Planner, executor and verifier

Role	Responsibility	Control boundary
Planner	Gemini selects the next tool	Structured proposal
Executor	Use the saved approved payment	Stable operation and receipt
Verifier	Check provider-owned evidence	Verified workflow state

Logical roles. Executor and Verifier share one worker JVM.

Architecture and state ownership



Four local JVMs with durable state. Gemini API runs remotely.

Gemini's supplied business context

Spring AI sends task, tool schemas and business context

Goal: request approved payment for INV-1042
Policy: stop on hold or missing approval

Registered tools:
  get_invoice_context(invoiceId)
  get_payment_status(invoiceId)
  submit_payment(invoiceId)

Facts return as tool results in the same conversation

{"functionCall":{"name":"submit_payment","args":{"invoiceId":"INV-1042"}}}

Java validates the request and creates ActionProposed

Events and local transaction boundaries

Validated submit_payment creates ActionProposed



Stage completion: inbox entry, state change and outgoing event

The provider commits in its own database.
Operation identity and receipt lookup protect recovery.

Identity across redelivery and new tool calls

Identity	Broker redelivery	New model request
Tool-call ID	Same	New
Event ID	Same	New
Operation ID	Same approved payment	Same approved payment

Operation identity spans both kinds of repetition

The completion-event gap

Workflow and broker

Agent application

1. Provider accepts
Workflow verifies

2. Consumer pauses
Before save and ACK

3. Broker retains event
Later delivery count = 2

Agent observes UNKNOWN
Gemini selects next tool

Gemini capture: status lookup. Scripted test: payment tool again.

Versioned event contracts

TaskPlanned example


```
{
  "contract": "dev.agentretry.agentic",
  "schemaVersion": "1.0",
  "type": "TaskPlanned",
  "eventId": "evt-example-001",
  "toolCallId": "call-example-001",
  "tenant": "demo-tenant",
  "invoiceId": "INV-1042",
  "operationId":
    "demo-tenant:INV-1042:approval-v1:submit-payment",
  "intentVersion": 1,
  "expectedStateVersion": 1
}
```


Version	Example	Decision
1.0	Known fields	Accept
1.1	Optional traceHint	Accept
2.0	Unknown	Reject

Java supplies the headers and validates the envelope.

Stale observations and current policy

Earlier tool result

approved: true
onHold: false

Illustrative earlier observation

Current record	Value at execution
Saved payment	INV-1042, approval version 1
Current policy	onHold = true
Incoming request	Same invoice and operation
Execution decision	POLICY rejection Provider untouched

Policy-change test: reject before provider submission

Policy before action

Check	Business context	Policy outcome
Approval	Invoice approval version 1	Required before execution
Hold flag	onHold = true	Reject submission
Payment intent	Amount and beneficiary	Application-owned fields

DENIED

Accepted payment requests: 0

Failure channels and recovery decisions

Failure category	Boundary	Application response
VALIDATION	Unsupported contract or stale event	Reject and record before execution
DELIVERY	Publication or lease budget	Bounded retries, failure record
PROCESSING	Transient work or evidence mismatch	Retry transient, stop permanent

POLICY records authorization denials.
Demo channels are typed durable failure records.

Retry budgets survive a restart



execution attempts = 3 of 3
restart / fresh proposal: no reset
next action = PROCESSING failure

Model calls, business execution and broker delivery have separate limits

Recovery from the last durable step

Observed evidence	Recovery decision
Workflow VERIFIED, agent UNKNOWN	Read status or return saved result
Provider receipt, local result absent	Recover receipt and resume verification
Conflicting or unprovable evidence	Record failure and stop execution

Worker-crash test: two execution attempts, one accepted provider record.

Agent recovery and repeat safety

java -jar agent-retry-demo.jar agentic --mode gemini
--fault completion --wait-ms 8000

Evidence source	After UNKNOWN	New payment proposals*	Accepted records
Live Gemini run Spring AI build4	get_payment_status	0	1
Scripted fixture Build4 JAR	submit_payment again	1	1

Repeat test: same command with --mode scripted

* New payment proposals counted after UNKNOWN.
Live run uses Gemini through Spring AI. Scripted choices are programmed.

Recovery invariants

One approved business action

One stable operation identity

One accepted payment request

Provider idempotency and retrievable evidence protect the effect

Applying this in your agent application

01 Register tools and return current business facts

02 Save approved parameters and operation identity

03 Commit inbox, state and outbox together

04 **Recover provider evidence within saved retry limits**

Patterns apply across model frameworks and clouds.

Companion: github.com/kkmurthyt21/agent-retry-demo

Event contract reference

Field group	Representative fields	Purpose
Contract	contract, schemaVersion, type	Select supported semantics
Trace	eventId, toolCallId	Correlate delivery and caller
Precondition	intentVersion, expectedStateVersion	Guard approved transitions
Operation identity	tenant, invoiceId, operationId	Recover one intended action
Business data	Tool results / saved approved intent	Keep money fields application-owned

Transaction reference

State owner	Atomic local records	Recovery evidence
Agent	Request: history, pending, outbox Result: inbox, pending response	Saved model/tool context
Workflow worker	Intent, state, event inbox and resulting outbox	Approved intent and stage
Mock provider	Payment record, key, fingerprint and replayable receipt	Provider-owned payment record
Broker	Queue entry, lease and ACK	Unacknowledged delivery

Failure and recovery reference

Case / origin	Application behavior
Completion pause (live Spring AI run)	Status reconciliation, one payment
Fresh payment call (scripted fixture)	Reuse approved operation, one payment
Worker crash (packaged fixture)	Restart worker, recover provider receipt
Stale execution event (unit test)	VALIDATION, provider untouched
Hold / revoked approval (domain test)	POLICY rejection before effect
Execution budget (unit test)	Three saved attempts, then stop

Technical references

Spring AI Gemini integration

Spring AI reference

Jota: AI banking workflows

Temporal customer case study

AP agents in Ramp Bill Pay

Ramp product documentation

Financial data access with Gemini

Copel / Google Cloud

Safe retries with idempotent APIs

Amazon Builders' Library

Transactional outbox pattern

AWS Prescriptive Guidance

Versioned agent event contracts

Krishna Kandi / AWS Builder Center

Durable refund reference demo

Temporal reference example

Thank you & Questions

What evidence would make your next retry safe?

github.com/kkmurthyt21/agent-retry-demo

in/krishna-kandi-2370269

