AAC · AGENT AUTHORITY CLOUD · BY CASCADEAUTH

# Enforce business authorization across AI agents

AAC carries signed business authorizations: what each agent may do, the limits it must respect, and which agent may act next.

cascadeauth.com

THE PROBLEM

# Delegated work needs a verifiable business authorization

### API access

An API credential establishes access to an API. By itself, it does not show the originating business approval or the limits added by earlier agents.

### Delegation scope

Passing on an accepted bearer credential can give the next agent the same scope. Narrower delegation requires additional authorization logic and trust relationships.

### Evidence after execution

Teams need a signed record of the business authorization each agent received, the limits it added and the result it returned.

The problem today: **over-privileged agents**, or **integrations stalled on trust** the moment work crosses a team or organization boundary.

WHY NOW

# Agent-to-agent work is crossing organization boundaries, and the identity stack stops at the first one

### Agents now talk to other companies' agents

Open protocols such as A2A and frameworks such as Google ADK make a cross-organization agent exchange a small integration. The authorization question arrives with the first real transaction.

### Enterprise buyers own business risk

For enterprise tenants, AAC's buyer is the CFO, CRO or CISO. Their teams need enforceable business authorizations and an audit trail across organizations.

### The standards layer is forming

IETF WIMSE adopted the AIMS Internet-Draft in September 2026. Its authors include Defakto, AWS, Zscaler, Ping, OpenAI and Okta. AIMS builds on workload identity and OAuth; AAC focuses on business authorizations that collaborating organizations can verify.

WHAT WE BUILT

# Business authorizations agents can verify

AAC carries signed business authorizations between agents. Each application's paired sidecar checks the authorization, its limits and the sender's identity before the application acts.

**1 · APPROVE**

Your application records the approved task and its limits: for example, a $10,000 trip budget.

→

**2 · SIGN**

The originating agent's sidecar signs that decision as the root of a chain.

→

**3 · DELEGATE**

The planning agent delegates an $8,000 limit to the booking agent. It cannot grant more than it received.

→

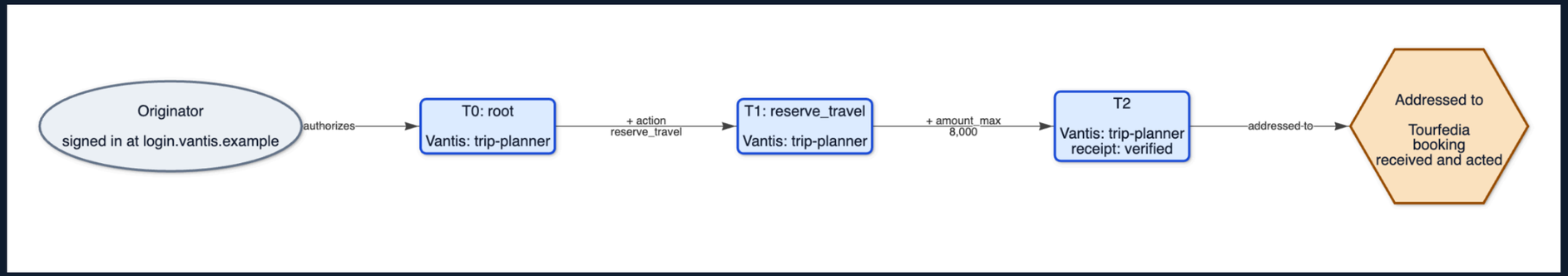
**4 · VERIFY**

The receiving sidecar verifies the chain locally before the application acts. A completed reservation returns a signed receipt.

Your agents keep their business logic and ordinary API credentials. Signed receipts and the execution graph help you inspect recorded execution.

SHIPPED · SHAPE 1, LINEAR AGENT WORKFLOW

# One approval, one agent after another, in minutes on your laptop




Recorded run of the public Docker Compose demo: the person who signed in, the trip-planner narrowing the authorization at each step, and the booking agent it was addressed to, which received and acted.

### `./quickstart`

One command installs the CLI, registers two companies as tenants in the hosted beta, starts both agents with their sidecars and makes the reservation.

### `2 to 5 minutes`

Measured first runs with downloads partly cached; on the timed run, most of it was the four GitHub sign-ins.

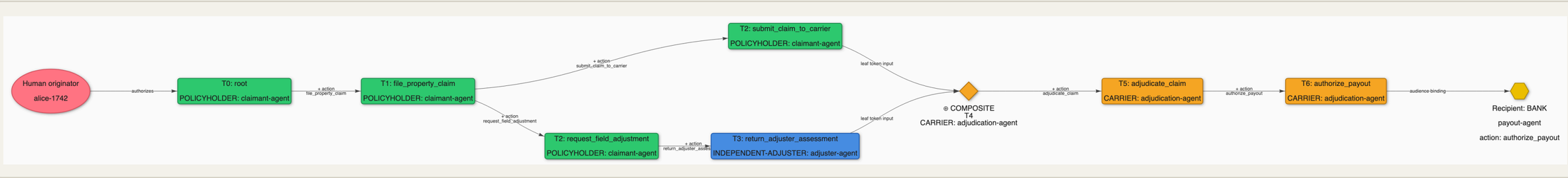
### `A2A too`

The same reservation runs as two Google ADK agents exchanging A2A messages through their sidecars, in a second public example.

SHIPPED · SHAPE 2, FORK AND JOIN WORKFLOW

# Business authorizations can branch and converge

Agent workflows in the field are linear, fork and join, or converge from several starts, and combinations of the three. This insurance claim forks to two parties and joins again at the carrier.




Human originator · Policyholder · Independent adjuster · Carrier · Recipient: bank

## One approval, two branches

The policyholder’s agent files the claim, then hands one branch to the carrier and one to an independent adjuster for a field assessment. Each branch carries the original approval, narrowed to its task.

## One combined authorization

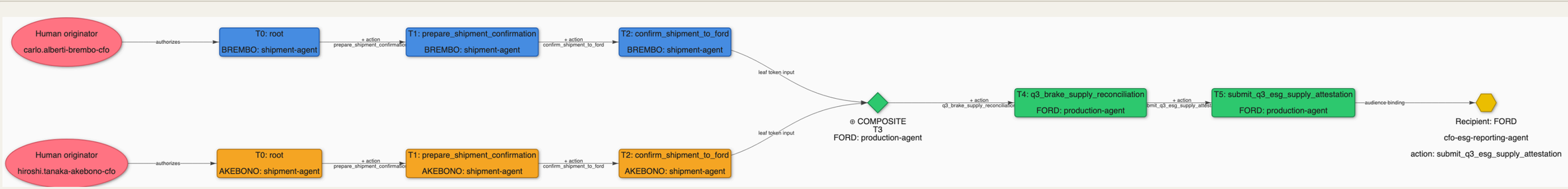
The carrier’s adjudication agent verifies both incoming chains and joins them into a composite authorization (the diamond), so the payout it authorizes is provably backed by the claim and the assessment.

## Four organizations, one graph

Policyholder, adjuster, carrier and bank each keep their own logs; the graph is reconstructed from all four by one chain id, with every hop’s limits visible.

SHIPPED · SHAPE 3, TWO WORKFLOWS CONVERGE

# Two workflows converge: two organizations’ approvals join at a third




## Two roots, two organizations

Each supplier’s CFO authorizes a shipment confirmation; each supplier’s agent narrows it and sends it to the manufacturer. Neither supplier shares an identity provider with the other, or with the manufacturer.

## One attestation, both lineages

The manufacturer’s production agent converges the two chains, reconciles the quarter’s supply and submits a compliance attestation whose authorization traces back to both suppliers’ approvals.

## Both approval lineages remain verifiable

AAC retains both verified parent chains in a composite authorization. The receiving sidecar can check both lineages locally.

Illustrative demo data; the named companies are not customers.

WHAT IS DIFFERENT

# Verifiable business authorization

### The approval and each delegation

The receiving agent sees the whole chain: the original approval and every limit added since. Each hop can tighten it and never widen it.

### Verified locally, across organization boundaries

No call back to earlier agents or their identity providers. Two organizations that share no authorization server can still verify each other’s agents on the spot.

### Bound to the agent that holds it

The sidecar checks that the sender matches the authorization’s designated holder. Copying the chain alone does not grant another agent access.

### A record you can audit

Signed receipts and the execution graph help you inspect which agents acted, under what limits, and who delegated the work.

WHERE WE ARE

# A beta product that trial users can onboard today

## Live (beta)

AAC Developer Beta: the hosted control plane and public-trust service, free, open to anyone with a GitHub or Google sign-in.

## Ready for evaluation

Four components released: the sidecar (Docker Hub, container or Linux and macOS binary), the AAC CLI (PyPI), the hosted AAC Control Plane, and the trust anchor publisher.

## Examples ready

Two public, runnable examples: the Docker Compose reservation and two Google ADK agents over A2A.

### Self-serve onboarding

Register a tenant, run the quickstart and inspect your own execution graph in minutes: no sales call, no provisioning ticket.

### Developer surface

docs.cascadeauth.com with a ten-minute path, guides, protocol reference and release notes for every component.

### Protocol reach

Agents in any language connect to their sidecar over local HTTP, natively or with A2A; sidecars verify across tenants with no shared authorization server.