

Mission-Bound Authorization

Governing the approved work, not just the token

Identifying, bounding and ending the work a person approved

Part I

Why does OAuth need a Mission?

The authorization system has credentials and grants.
People approve units of work.

Alice asks an agent to reconcile invoices

“Review Q3 invoices and post adjustments up to \$500 each.”

What did Alice actually approve?

Read	Invoices issued during Q3
Write	Journal adjustments up to 500.00 USD each
Time allowed	Until 13:00 UTC, set when she approves
Accountable	Alice: she approved it, and she can stop it
Carried by	Many tokens: issued, refreshed, exchanged, retried

One approved unit of work. Which OAuth object represents it?

Alice cancels the job at 12:02. What can she stop?

OAuth today

Revoke the client

Stops all of its work, not just this job

Revoke tokens one by one

[RFC 7009](#), [Token Status List](#): only tokens someone can find

Revoke the grant

[Grant Management](#): client-only; access tokens SHOULD stop

Wait out the tokens

Meanwhile the grant can still mint new ones

What she means

Stop this job, by name

Every way to a new token for this job refuses: code, refresh, [Token Exchange](#).

Only this job stops; other work goes on.

Issued tokens end within one lifetime.

Not the agent, not one token: she wants to stop the work she approved

OAuth has no interoperable object for the approved work

Who is acting?	Identity and delegation: OpenID Connect , WIMSE , actor chains (RFC 8693); across domains, ID-JAG and identity chaining
What may a principal hold, and when?	Entitlement: entitlements, SCIM provisioning, just-in-time elevation
What authority was issued?	Grants and tokens: scopes, Rich Authorization Requests (RAR), resource indicators, Token Exchange; Grant Management
Is this credential or grant still good?	Status: revocation (RFC 7009), introspection (RFC 7662), Token Status List , Shared Signals Framework
May this one request proceed?	Per request: AuthZEN decisions and access requests, Deferred Token Response (DTR), Transaction Challenge
What was this work approved to do, and is it still approved?	No common contract. An approved-work object: bounded authority, an identifier, a lifecycle

Each answers a piece. None names the approved work or says it is still approved.

Approval enumerates. Agents discover.

Typical OAuth deployments

Agents instead

A client registered in advance

Can arrive unregistered: the Model Context Protocol (MCP) makes Client ID Metadata Documents the default

Scopes chosen at design time

Discover tools at runtime; each call challenges for its scopes

One resource, known at grant time

Compose tools across servers and authorization servers

Work known when the grant issues

Touch objects that do not exist when approval happens

Authority accumulates while consent moves away from the operation

Each choice is sound locally. Together they push more authority through fewer approvals.

Correctly authorized calls add up to authority nobody approved

What a Mission is

A durable, approval-backed governance object,

held by its Mission Issuer:

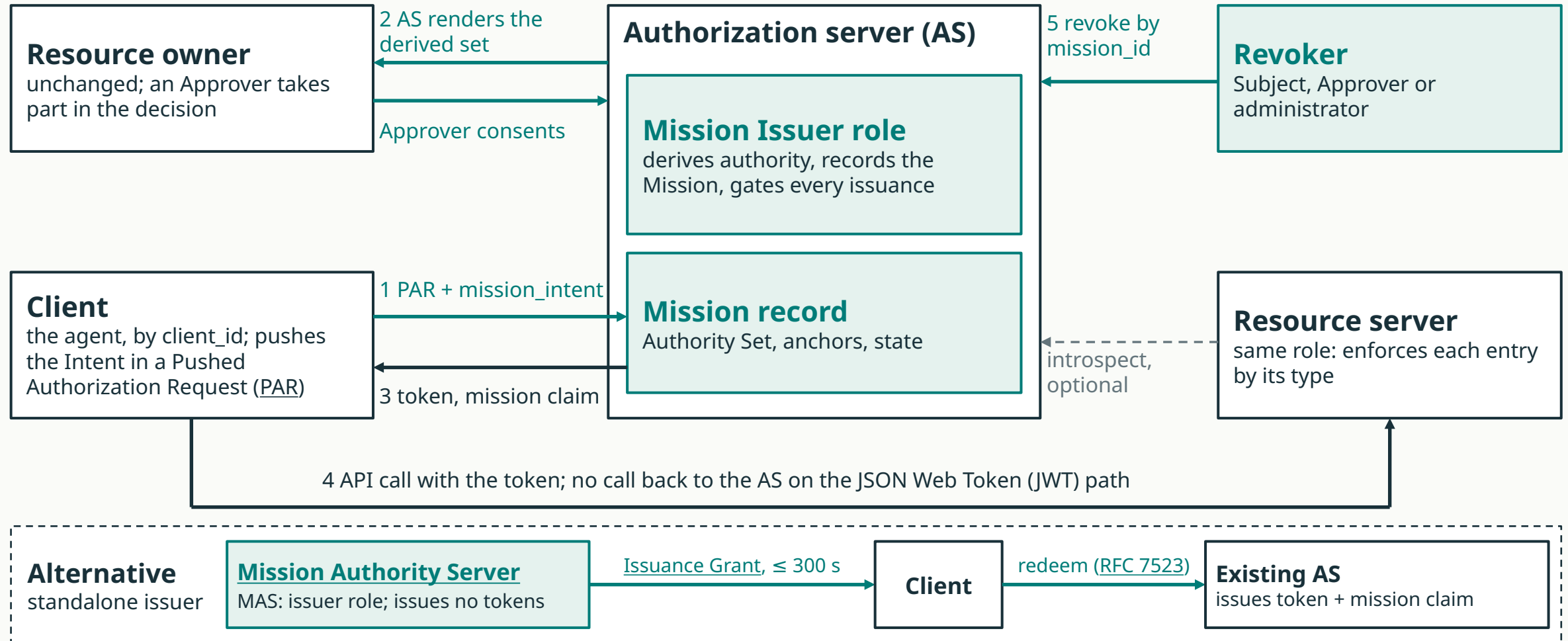
the approved work, with a lifecycle,

that authority is derived for, bound to, and gated on.

In one line: the approved work, made addressable, so every later derivation stays bound to it and gated by it.

The handle Alice was missing: one object for the approved work

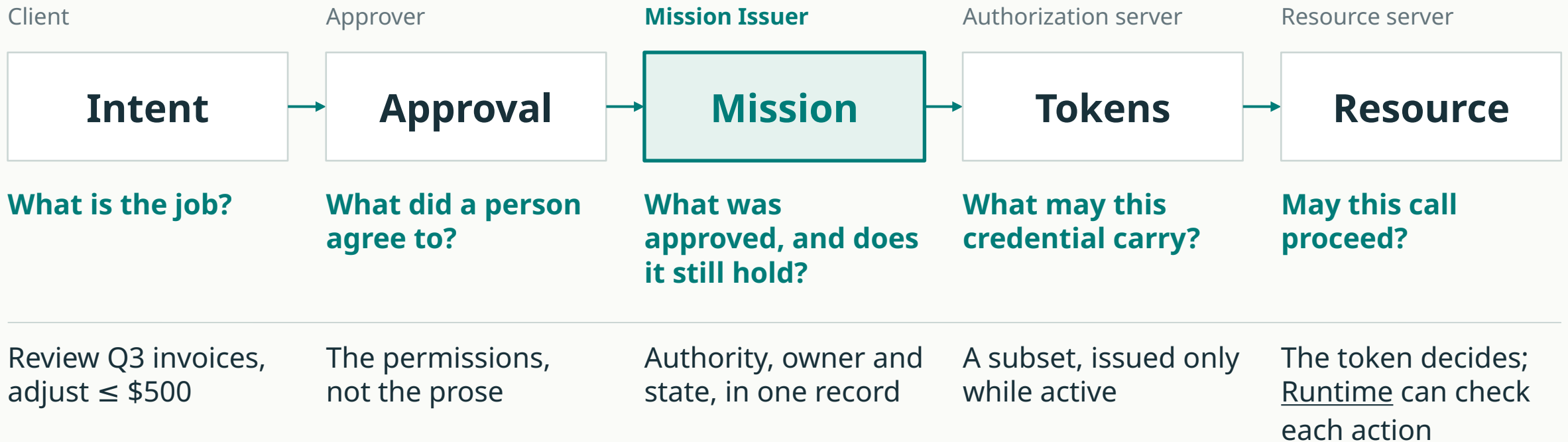
Where a Mission sits among OAuth's roles



Same OAuth roles. The AS, or a standalone MAS, hosts the issuer; a revoker gets one new call.

From intent to resource: five questions

One record answers the middle one, and every Mission-bound token traces back to it.



Tokens still carry authority. The Mission is where that authority comes from.



Shaping: from Alice's sentence to a proposal

Alice asks

"Review Q3 invoices and post adjustments up to \$500 each."

Mission Shaper: client or service

- Resolves each capability against a catalog or policy; never invents one
- Bounds open work: period, amount, expiry
- Ambiguous? Clarifies, narrows or refuses

Untrusted proposal

Mission Intent

goal, targets, task bounds, expiry

Authority Proposal (RAR)

invoices.read, issued in Q3
journal-entries.write, ≤ 500.00 USD

Shaping Evidence

how it was produced, inferences included; audit only, never authority

Mission Issuer
validates, narrows, derives the set

trust boundary

The shaper proposes. Only the Mission Issuer creates authority.

Intent

Approval

Mission

Tokens

Resource

Alice approves the permissions, not the prose

The job

**Reconcile Q3 invoices
and post small adjustments**

What the agent may do

**Use this accounting API
Read Q3 invoices
Post adjustments up to \$500 each
Access ends at 13:00 UTC**

Who approves: Alice directly in the MVP. Standing consent, such as a Template, is a companion.

Accountable

consent_principal

Always a person, or a principal a person answers for

Decides

adjudication

That person now, or a versioned policy they approved. A model may refuse or narrow; it never decides.

Triggers

activation_actor

That person, or a dispatcher running a template that person approved. The working agent never approves its own expansion.

Interoperability begins at the committed Authority Set, not the Intent



The Mission: one record of the approved work

Identifier	msn_7Kq2...: what every token names, and what Alice revokes
Authority Set	What she approved: Q3 invoice reads, adjustments up to \$500 each. Committed by hash, so a change is detectable
Accountable	Alice: she approved it and answers for it
Intent	Why the work exists, for consent and evidence; never grants authority
State	Active until 13:00, unless revoked first. Widening means a new Mission

Only the state changes after approval



Approved authority is an envelope, not a script

Approved Authority Set: the envelope

Any Q3 invoice, any adjustment $\leq$ \$500

What this run
actually does

Agent risk lives in the gap between them

Closed at approval: a resource the envelope never named gets nothing in the core.

Levers that narrow the gap

Subset rule caps every token

Runtime check of each action against it

Fresh approval for the riskiest actions

One bounded ceiling for many increments

Caps on cumulative consumption

Retiring authority as the work completes

Core

Extension

Extension

Extension

Extension

Interop set

Let the agent improvise the path. Never let it improvise the authority.

Intent

Approval

Mission

Tokens

Resource

In, on or out of the loop: the same bound

In the loop

HITL: a person decides first

Alice approves the Mission before any token is issued

MVP

Wider authority needs a new approval and a new Mission

MVP

Fresh approval bound to one high-consequence action

Extension

On the loop

HOTL: it runs; a person can stop it

Inside the envelope, no prompt per call

MVP

Alice, or her admin, revokes by mission_id

MVP

Query, suspend and resume

Interop set

Out of the loop

HOOTL: no person in the moment

A policy a human approved creates the Mission

Extension

Same subset rule, expiry and revoke

MVP

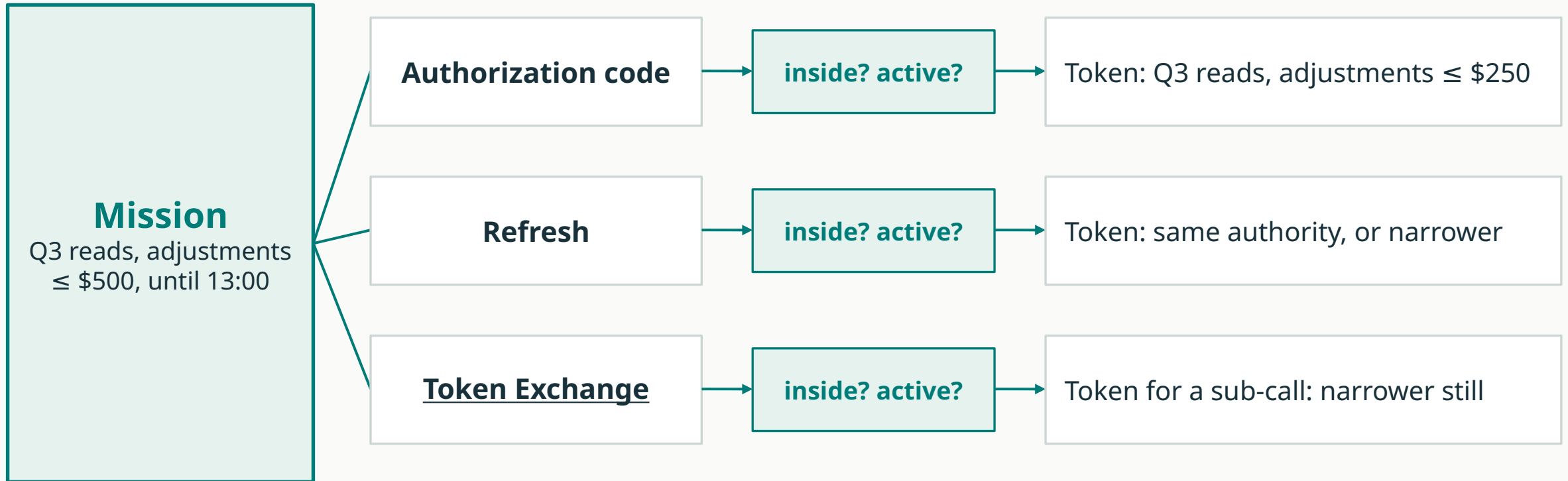
Prohibited classes always go back to a person

Extension

The loop is a deployment choice. The bound is the same in all three.



Tokens: every path checks the approval



After 12:02: all three paths refuse. Tokens already issued run out on their own clock, never past 13:00.

Inside the approval, and only while the work is live



The token bounds each call; Runtime checks each action

Core: the resource enforces the token

- Validates the token and enforces each authorization_details entry by its type
- A \$300 post fails under a \$250 token
- Validated locally, no call to the authorization server. A resource that introspects instead sees active: false once the Mission ends

Runtime extension: each action checked

- A policy enforcement point (PEP) asks a policy decision point (PDP) before each consequential action
- Checks the action and its parameters against the Mission, the actor chain and fresh state
- \$200 to invoice 4471: permit, bound to those parameters. A minute after revocation: deny, before the token expires

The token bounds the authority. Runtime checks each action against the Mission.



Three things not to confuse with a Mission

Not the agent's identity

Who or what is acting

Workload identity (WIMSE), the AI Identity Management System (AIMS) and actor chains (RFC 8693, the Actor Profile) name who is acting, and through whom. An identity outlives any one task.

Not the Intent's meaning

What the words mean

Intent explains the work. It does not grant authority: the approved RAR authority is the boundary.

Not execution state

What the agent is doing now

Model Context Protocol and Agent2Agent (A2A) tasks and sessions track progress. A Mission is built to end.

A Mission answers: what approved work does this serve, and is it still approved?

A standard lets independent parts agree on the work

One authorization server can approximate this privately. These crossings need a standard:



Private state approximates it in one deployment. Interoperability starts where components meet.

Why not RAR + Grant Management?

The closest build: RAR plus one grant per job under OpenID's Grant Management (GM) draft.

To do this	RAR + Grant Management	Mission
Stop one job	GM's revoke API is client-only	Revoke: a standard stop by Subject, Approver or admin (Status revoke-only)
End what derives	Refresh: MUST. Access tokens: SHOULD. Exchange: silent	Gate: MUST refuse on every path, exchange included
Narrow every derivation	Left to the AS	Narrow: typed subset rule, or refuse
Rely on what was approved	merge and replace rewrite the grant	Commit: immutable record; Authority Set hashed
Tie tokens to the work	grant_id only in the token response	Join key: mission claim on every token

The two-task test: one client, two tasks. Someone other than the client ends one with a standard call. From then on every issuance path MUST refuse it, exchange of a still-valid token included; the other runs on.

A Grant Management profile that closes these gaps carries the same contract

Part II

What is the minimum viable Mission?

If we agree the object is useful, how little must OAuth add to get it?
Commit. Narrow. Gate.

The MVP: commit, narrow, gate

MVP: the [OAuth binding](#) plus Status's revoke-only class, for **direct approval**. Three behaviors at the AS define it.

Commit

Approval fixes the Authority Set; no widening in place

Keeps: Authority serves an approved task · Anchors commit, not semantics

Narrow

Every derived authority MUST be a subset of the approval

Keeps: Authority only narrows

Gate

Every issuance path MUST check the Mission is active

Keeps: Only active permits · Revocation is possession-independent

Outside the MVP

Added by extensions

Approval without Alice at the screen

→ [Template](#) · [Deferred Approval](#) · [Governance](#)

The work changes after approval

→ [Expansion](#) · [Progressive](#) · [Containment](#)

Delegation beyond Token Exchange

→ [Child Delegation](#) · [Attenuation](#) · [Cross-Org](#)

Not guaranteed: extensions narrow it

Issued tokens stop at once

→ [Lifecycle Signals](#) · [Status](#) · [Runtime](#)

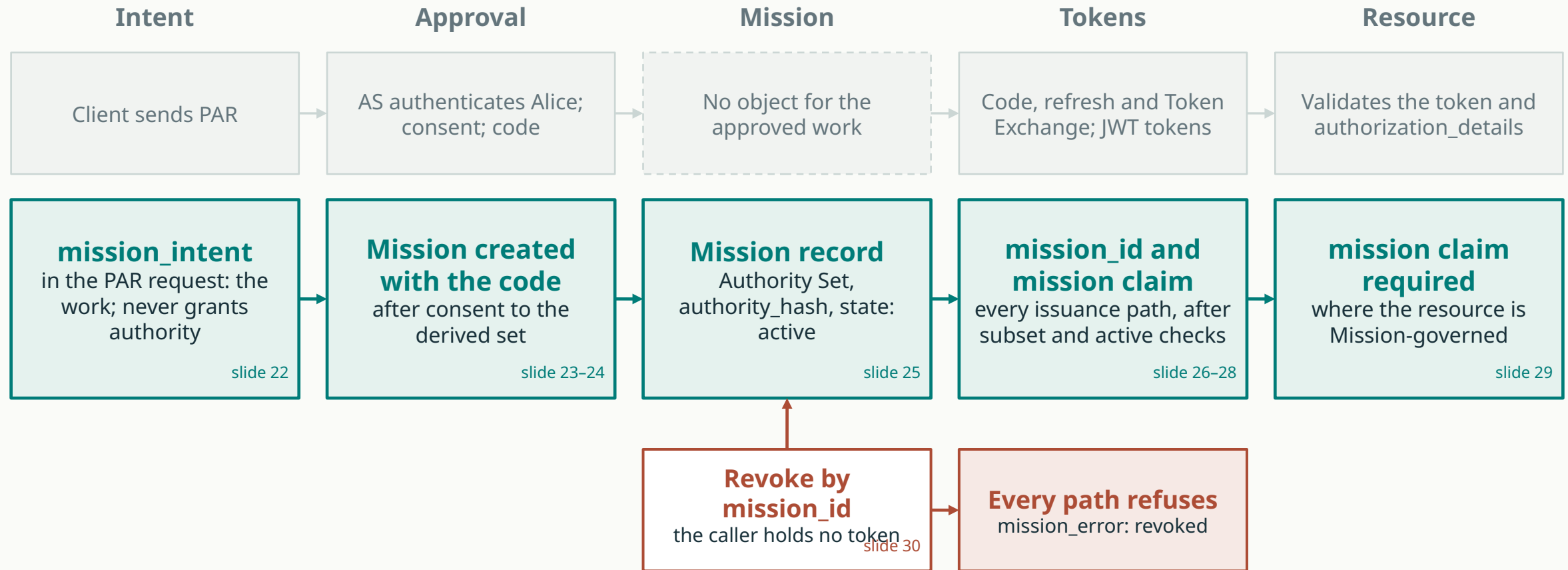
Alice is still entitled

→ [Cross-Domain Projection](#) · [Runtime](#)

Each action serves the work

→ [Runtime](#) · [Transaction Authorization](#) · [Metering](#)

Where the MVP touches OAuth



□ OAuth as deployed, unchanged

□ What the MVP adds, and the slide that shows it

Ordinary OAuth, plus a few teal touchpoints. Each gets a slide next.

The request: the agent proposes through PAR

```
POST /par HTTP/1.1
Host: as.example.com
Content-Type: application/x-www-form-urlencoded
```

decoded, line-wrapped for display

```
response_type=code&client_id=s6BhdRkqt3
&code_challenge=...&code_challenge_method=S256
&mission_intent={"intent": {
  "goal": "Reconcile Q3 invoices and post
            adjustments up to $500 each.",
  "target_resources": ["https://erp.example.com"],
  "expires_at": "2026-10-26T13:00:00Z" }}
&authorization_details=[
  { "type": "mission_resource_access",
    "resource": "https://erp.example.com",
    "actions": ["invoices.read"],
    "constraints": {
      "resource_issued_after": "2026-07-01T00:00:00Z",
      "resource_issued_before": "2026-09-30T23:59:59Z" } },
  { "type": "mission_resource_access",
    "resource": "https://erp.example.com",
    "actions": ["journal-entries.write"],
    "constraints": { "max_amount":
      { "amount": "500.00", "currency": "USD" } } } ]
```

mission_intent

The work as the Approver saw it. Untrusted; committed as intent_hash

authorization_details

The proposed authority: the AS derives the Authority Set from it by its own policy

Why a new parameter, not a RAR type

- RAR entries land in the token as authority; the Intent must not
- Its own anchor: intent_hash
- PAR only; RAR may ride the front channel
- Required with RAR from governed clients

The Intent rides its own parameter, through PAR only. Authority stays in RAR.

Intent

Approval

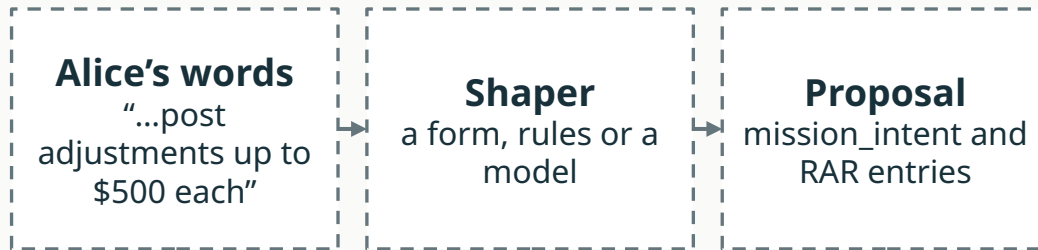
Mission

Tokens

Resource

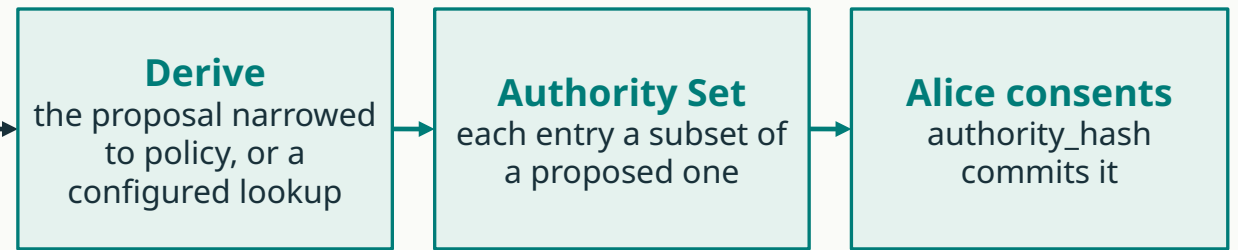
Probabilistic shaping, mechanical derivation

Probabilistic: client side, untrusted



Can guess wrong, or be steered by what the agent read

Mechanical: the authorization server



PAR: trust boundary

- Same inputs and policy version, same Authority Set; policy_version records which
- Prose (goal, task_bounds) never changes the derived set or gates issuance
- A model at the AS is a recorded input that can refuse or narrow, never supply or widen
- If the subset relation cannot decide, the AS refuses

Mechanical at one AS, not portable: two ASes may derive differently, so interoperability starts at the committed set.

Same inputs, same policy, same Authority Set



Approval: Alice consents, the Mission commits

GET /authorize?client_id=s6BhdRkqt3

&request_uri=urn:ietf:params:oauth:request_uri:6esc_11ACC5bwc

Consent screen, rendered by the AS

The accounting agent may, until 13:00 UTC:

- Use the accounting API (erp.example.com)
- Read invoices issued in Q3
- Post journal adjustments up to 500.00 USD each

Work, as the agent described it: "Reconcile Q3 invoices and post adjustments up to \$500 each."

Approve

Deny

HTTP/1.1 302 Found

Location: https://agent.example/cb

?code=Sp1x10BeZQQYbYS6WxSbIA

&iss=https%3A%2F%2Fas.example.com

At approval the AS, in order:

1. Authenticates Alice itself, not via the client
2. Establishes the Subject, the authority source and the effective expiry
3. Renders the derived Authority Set, not just the goal, distinct from client text
4. Computes authority_hash, intent_hash and proposal_hash
5. Creates the Mission, active, atomically with the authorization code

Consent attaches to the derived Authority Set. The Mission commits with the code.

Intent

Approval

Mission

Tokens

Resource

What Alice's approval records, in JSON

```
{
  "id": "msn_7Kq2xV9pL3wRb4Tn8YzQ1d",
  "issuer": "https://as.example.com",
  "state": "active",
  "authority_set": [
    { "type": "mission_resource_access",
      "resource": "https://erp.example.com",
      "actions": ["journal-entries.write"],
      "constraints": { "max_amount":
        { "amount": "500.00", "currency": "USD" } } },
    { ...the Q3 invoice-read entry... } ],
  "authority_hash": "sha-256:l3KvZ4mP5x0wQrR6...",
  "expires_at": "2026-10-26T13:00:00Z",
  "intent": { "goal": "Reconcile Q3 invoices...", ... },
  "intent_hash": "sha-256:wQ7p4LHnX9Md0LqJ...",
  "approval_basis": { "type": "direct", ... }
}
```

id, issuer

The handle: what every token names, and what Alice revokes

authority_set

What Alice approved: typed RAR entries, the capped writes and the Q3 reads

authority_hash

Commits the set; widening is a new Mission

state, expires_at

Only state changes after approval; expires_at closes the Mission on its own

intent, intent_hash, approval_basis

Kept for consent and evidence; they never grant authority

One stable record that every later derivation is judged against

The token carries authority and a reference

Decoded access token, excerpt

```
{
  "iss": "https://as.example.com",
  "aud": "https://erp.example.com",
  "exp": 1793016300,
  "cnf": { "jkt": "0ZcOCORZNYy-DWpqq30j..." },
  "authorization_details": [
    { ...the Q3 invoice-read entry... },
    { "type": "mission_resource_access",
      "resource": "https://erp.example.com",
      "actions": ["journal-entries.write"],
      "constraints": { "max_amount":
        { "amount": "250.00", "currency": "USD" } } },
  "mission": {
    "id": "msn_7Kq2xV9pL3wRb4Tn8YzQ1d",
    "issuer": "https://as.example.com" }
}
```

Authority

authorization_details: any RAR type the AS supports. Here mission_resource_access, narrowed to \$250

Provenance

mission: issuer and id of its approval. A reference, not a binding: gated issuance makes it Mission-bound.

Authority lives in authorization_details. The mission claim names where it came from.

Every new token stays inside the approval

Approved: adjustments up to 500.00 USD each

Request for the next token	Result
Cap each adjustment at \$250	Issue: within the approved entry
Exchange that \$250 token for \$400	Refuse: broader than the presented token
Raise the cap to \$600	Refuse: broader
Remove the cap	Refuse: a constraint disappeared
Change the cap to 500.00 EUR	Refuse: another currency is never a subset
Narrow an entry of a type the AS cannot narrow	Only the recorded entry, unchanged

Here the entry is `mission_resource_access`, a RAR type with a published subset relation, shown next.

The type defines the comparison. What cannot be compared is never narrowed.

Companion: what Mission Resource Access adds

RFC 9396 carries authority but has “no standardized mechanism to compare” two requests. Resource Access adds one general-purpose type whose entries can be compared and narrowed the same way everywhere.

Resource matching

Exact by default, or prefix at a path-segment boundary; prefix compares after RFC 3986 normalization

Action families

invoices.* covers invoices.read and invoices.q3.*, never invoicesx.read

Common Constraints

A registry, one subset rule each, e.g. resource_issued_after/before, max_amount, time_window, allowed_tools

Subset algebra

A constraint missing from the derived entry counts as broadest, so it fails; delegation limits only shrink

Fail-closed resources

An unknown constraint or match mode means refuse

Tools and scopes

An MCP tool is an entry: server as resource, tool name as action. Safe conditions for projecting to plain scope

The binding does not depend on this type. It needs a subset relation; this is one general-purpose type that has one.
Next: align with draft-zehavi-oauth-authority-metadata, which publishes what a type conveys, and draft-liu-oauth-authorization-schemas, which registers type schemas and their narrowing rules.

RAR says how to ask for authority. Resource Access says how to compare it.

Resource: the API call, enforced from the token alone

```
POST /journal-entries HTTP/1.1
Host: erp.example.com
Authorization: DPOP eyJhbGciOiJFUzI1NiIs...
DPoP: eyJ0eXAiOiJkcG9wK2p3dCIIs...
```

```
{ "invoice": "4471",
  "amount": { "amount": "300.00",
              "currency": "USD" } }
```

HTTP/1.1 403 Forbidden

```
WWW-Authenticate: DPOP error="insufficient_scope",
  mission_denial="insufficient_authority"
```

The token caps adjustments at \$250: \$200 passes.

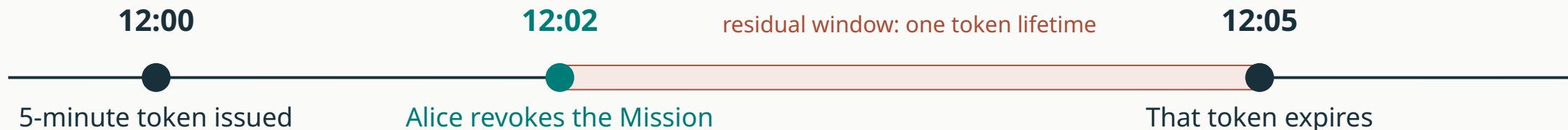
The resource server:

- Validates the token and its DPOP (proof-of-possession) binding locally; no call to the AS
- Enforces each authorization_details entry by type
- Fails closed on a type, constraint or match mode it cannot enforce
- Never lets scope grant more than authorization_details
- Where it requires Mission-bound tokens, rejects one without the mission claim: invalid_token
- Logs mission.id with each request, so access joins to the approval

The resource needs only the token. Mission adds no call on the request path.



12:02: Alice stops the work, not the client



Revoke, with Mission Status

```
POST /as/mission/lifecycle HTTP/1.1
Host: as.example.com
Content-Type: application/x-www-form-urlencoded
Authorization: DPoP eyJhbGciOiJFUzI1NiIs...
DPoP: eyJ0eXAiOiJkcG9wK2p3dCIIs...
```

```
mission_id=msn_7Kq2xV9pL3wRb4Tn8YzQ1d
&operation=revoke
&nonce=nonce_8Y3vN0sM6tP1xR9bQ5
```

The revoker's own token (scope mission_lifecycle);
it holds none of the Mission's.

From 12:02, every derivation for this Mission refuses

- An unredeemed code** `invalid_grant; mission_error: revoked`
 - Refresh** `invalid_grant; mission_error: revoked`
 - Exchange the noon token** `invalid_request; mission_error: revoked`
 - The client's other Missions** Unaffected
 - Adjustments already posted** `Not undone`
- The gain is granularity, this job and not the client; not speed.

Revoking the Mission stops new authority for this work, not the whole client



Mission Status: one lifecycle, by mission_id

Operation	Effect	Typically authorized for
revoke	To terminated, reason revoked	The Subject, the Approver, administrators
suspend	To suspended; derivation pauses. Optional suspend_until with on_expiry	Administrators
resume	Suspended to active, without re-approval	Administrators, with distinct or elevated authorization
complete	To terminated, reason completed	The requesting client, administrators

Beyond the four operations

Status query

An authenticated caller asks by mission_id; no Mission token needed. Signed response

No enumeration oracle

Unknown and unauthorized look identical: the same 404, body and headers

RFC 7009

Revokes tokens; a deployment may couple refresh-token revocation to the Mission, but only the lifecycle endpoint is the interoperable stop

Stop a Mission by its identifier, without holding any of its tokens



Close

One core, extensions when needed

What the MVP is, what it costs, the path, and the questions for the group.

The MVP, the interoperability set, and what waits

The MVP: binding + Status revoke-only

- Mission identifier and issuer, bound to the grant, by direct approval
- Approval-backed Authority Set, committed
- Subset rule on every derivation
- Active-state gate on every issuance path
- mission claim on derived tokens
- Revoke by mission_id: Status's revoke-only class

Interoperability set

- Resource Access: a reusable type with a subset relation
- The rest of Mission Status: query, suspend, resume, complete
- Next: align Resource Access with Authority Metadata and Domain-Specific Authorization Schemas

Waits: on demand

- Runtime enforcement
- Cross-domain projection
- Child delegation
- Metering
- Portable evidence
- Progressive authorization
- Open: Mission Authority Server

The cost: little new on the wire; at the AS, durable records, a state read on every issuance path, one order per Mission

A narrow waist: extensions attach later without changing the core

Proposed path: two tracks, one alignment

- 1 Use cases** evolve the use case, framework and requirements with the group: [draft-liu-mcguinness-oauth-bounded-work](#) (Informational)
 - 2 Solution** validate the proposed solution and make it concrete: [draft-mcguinness-oauth-mission](#) with Status's revoke-only class: conformance, a second implementation, deployments
 - 3 Alignment** align [draft-mcguinness-oauth-mission-resource-access](#) with [draft-zehavi-oauth-authority-metadata](#) and [draft-liu-oauth-authorization-schemas](#); cross-reference the Mission record from [draft-ietf-wimse-aims](#) (AI Identity Management System)
- Then** adoption and the container, once the requirements settle and the solution is validated

Most useful now: a passing combination, an independent implementation, or the open problems

Questions for the working group

1 Is there an interoperability gap?

Do existing mechanisms give independent implementations a common contract for identifying, bounding and ending approved work across derivations?

2 What is the minimum contract?

Committed authority, narrowing on every issuance path, lifecycle gating, revocation by identifier: necessary, and enough?

3 What should the group standardize?

A distinct Mission object, a constrained grant profile, or smaller reusable extensions?

If an existing combination passes the two-task test, we should evaluate it

Backup: where to start reading

- 1 **draft-liu-mcguinness-oauth-bounded-work**
Bounded Work Authorization: the use case, a framework and requirements R1 to R9; no wire format
- 2 **Mission Handbook**
The narrative companion, in prose: the why before the wire, objections included
- 3 **Mission Architecture (editor's copy)**
The authority path, the verbs and the invariants: how every part fits
- 4 **draft-mcguinness-oauth-mission**
Mission-Bound Authorization for OAuth 2.0: the three behaviors, and the job end to end
- 5 **draft-mcguinness-oauth-mission-resource-access**
Mission Resource Access: the subset relation and the Common Constraints
- 6 **draft-mcguinness-oauth-mission-status**
Mission Status: revoke, suspend, resume, complete; the revoke-only class
- 7 **Repository**
Test fixtures and the conformance ledger

Start with the use case and the Handbook, then the same job on the wire.

Backup: how the Mission specs compose

Model (informative)	Architecture · Security Model · Intent Shaping · Agent Access Model mapping (sketch) (Analyze; Propose on the client side)																
Kernel (normative)	Mission Substrate Requirements: the binding-neutral kernel every binding is authored against; the one candidate-maturity draft																
Bindings (peers)	OAuth binding: the MVP		Mission Authority Server		AAuth binding		UMA 2.0 (sketch)		GNAP (sketch)								
Companions, by verb	Propose	Approve and Record	Govern	Enforce Each Action	Run and Wind Down	Delegate	Project	Continue	Prove								
	Submission Evidence Request Provenance	Resource Access Issuance Grant Deferred Approval Approval Revision Template Approval Governance Consent Evidence	Status Signals Status List Management Containment Expansion Progressive Discharge Derivation Limits Metering Discovery Control Plane AAuth Management AAuth Expiry	Runtime AuthZEN Profile OAuth Profile Transaction Authorization Capability Binding Runtime Evidence	Harnesses Orchestration	Child Delegation Offline Attenuation Cross-Org Delegation	Cross-Domain Projection Cross-Org Delegation	Continuation	Consent Evidence Runtime Evidence Mandate Audit Transparency Evidence Envelope Work Products Approved-Set Check								
Builds on	PAR , RAR		RFC 7523 ; DTR ; ARAP		RFC 7662 ; Status List ; SSF , CAEP		AuthZEN ; Transaction Challenge		Agent harnesses ; no new wire	RFC 8693 ; Actor Profile		ID-JAG ; identity chaining		ICA ; RFC 8693		SCITT (RFC 9943)	

Catalog as of 6 October. Bold: the MVP and interoperability set. Teal verbs: what the MVP implements. UMA: User-Managed Access · GNAP: Grant Negotiation and Authorization Protocol · SCITT: Supply Chain Integrity, Transparency and Trust · ID-JAG: Identity Assertion JWT Authorization Grant · ICA: Identity Continuation Assertion

One kernel, peer bindings, companions by verb: each attaches when its trigger appears

Backup: why not an existing extension

Extension	Solves	Leaves open
RAR (RFC 9396)	Structured authority on request and token	No record, lifecycle or narrowing contract across derivations
Token Exchange and act	Delegation per hop; who acted through whom	Narrowing is local policy; act is actor lineage, not the task
Grant Management	An addressable grant_id: query, revoke	Merge and replace rewrite in place; client-only revoke: refresh MUST, access tokens SHOULD; no narrowing rule
Short tokens, refresh revocation	Bounded credential lifetime	Per client or grant, not per task; only as fresh as every path's re-check
Transaction Tokens , ID-JAG	Call-chain context; identity across domains	Carry identity and scopes per hop; no approval record, no lifecycle
DTR , ARAP , CIBA , Transaction Challenge	Asynchronous and per-transaction approval	Approval ends at the decision; nothing governs what follows
AuthZEN , Status List , SSF and CAEP	Decisions, token status, session events	No state model or event for a task or an approval
Dynamic registration per task	Isolates and revokes grants per task, one AS	Client IDs are local to one AS; no approved bound or subset rule
A PDP with task context	Per-call decisions in a task's context	A private task store; issuance stays ungated, so tokens keep minting

DTR: Deferred Token Response · ARAP: AuthZEN Access Request and Approval Profile · CIBA: Client-Initiated Backchannel Authentication · SSF: Shared Signals Framework · CAEP: Continuous Access Evaluation Profile

None combines an approved-work identifier, committed authority, narrowing and a gate.

Backup: how Mission composes with existing controls

Each control tightens one credential. The Mission says what they are all bounding; it replaces none of them.

Move	Existing control	What it tightens	Mission connection
Bind it	DPoP (RFC 9449), mutual TLS (RFC 8705)	Who can present a token	Binds the holder of the Mission's authority; only a sender-constrained token can be delegated
Shorten it	Short-lived tokens, refresh rotation	How long a token lives	Renewal is a derivation: refresh checks live Mission state
Narrow it	RAR , resource indicators, policy engines	What a token reaches	Limits derive from the approved Authority Set; every derivation is a subset
Delegate it	Token Exchange , act claims, Actor Profile	Who acted through whom	Keeps the Mission and its ceiling; an exchange stays within the presented token
Step up	Step-up authentication (RFC 9470)	Authentication strength or freshness	Kept apart from approval: a fresh login approves no wider work
Revoke it	RFC 7009 , Grant Management	One token, or one client's grant	Revoke by mission_id stops every derivation for this work

Existing controls implement the bounds. The Mission says what they bound.

Backup: why the Mission, and not another root

Candidate root	Approved	Checked every issuance	Stops issuance	Carries the bound	Crosses exchange, servers
Client registration	No	At one AS	Delete the client	No	No
Refresh-token family	Issued from one	Refresh only	If someone holds it	Its own, changeable	No
Grant (grant_id)	Yes	At one AS	By the client	Merge, replace change it	No
Session or MCP/A2A task	No	No: runtime state	No	No	No
Agent identity	No	At the identity provider	Agent's, not job's	No	Identity only
External task identifier	No	No	Not at the AS	No	As a label
GNAP grant (RFC 9635)	Yes	At its AS, one instance	Yes	For that instance	No cross-client
Transaction Token	No	One trust domain	One call chain	Scopes, context	One domain
AAuth mission	Yes, at the PS	At the Person Server	Yes	No subset rule	Within AAuth
Mission	Yes	Yes	Yes	Yes, immutably	Via projection

The client is too coarse, tokens too many, a grant the right shape but a private handle

Backup: overlapping drafts in OAuth and WIMSE

Defines a task object: AAuth missions, approved and held at the Person Server (individual)	Names the gap, defines no object: <u>agent authorization gap analysis (chen)</u> ; WIMSE AIMS §10.1, the Agent Mission, translation out of scope
--	--

Propose	Approve and Record	Govern	Enforce Each Action	Delegate	Project and Continue	Prove
<u>Intent Admission (jiang)</u> <u>RAR intent request (chen)</u>	Deferred Token Response <u>Transaction Challenge (rosomakho)</u> <u>Agent grants profile (mishra)</u> <u>PACT consent profile (valverde)</u> <u>Scope aggregation (jia)</u> <u>User-mediated delivery (emerson)</u>	<u>Agent revocation (chen)</u> <u>RAR lifecycle binding (chen)</u> <u>Delegated refresh tokens (zhu)</u> <u>Revocation closure (watts)</u>	Transaction Tokens <u>Txn-Tokens for agents (araut)</u> <u>Transactional access tokens (tulshi)</u> <u>SPT-Txn (coetzee)</u>	<u>Attenuating tokens (niyikiza)</u> <u>Delegated authorization (li)</u> <u>Delegation chain (liu)</u> <u>Actor chains (mw)</u> <u>Agent-Delegation (hamr)</u> <u>Agent task groups (song)</u> <u>Identity propagation (sharma)</u> <u>Attenuated chains (asor, W)</u> <u>Task-bound grants (atakora, W)</u> <u>Credential delegation (sweeney, W)</u> <u>Chain evaluation (jackson, W)</u>	ID-JAG Identity chaining <u>Txn-Token chaining (fletcher)</u> <u>Cross-domain Txn-Tokens (liu)</u> <u>Request delegation chain (zehavi)</u> <u>Cross-org delegation (reece, W)</u>	<u>Decision subject (aravind)</u> <u>Operator of record (aravind)</u> <u>Agent audit record (gilda, W)</u> <u>Authorization evidence (munoz, W)</u> <u>Aggregate signatures (reddy, W)</u>
Intent per request; no durable record	Per request or capability; nothing outlives the decision	Keyed to agents, tokens or a task id the AS does not own	Per-call context; no approval it answers to	Narrows per hop; no approved root to narrow from	Carries identity and scopes, not the approved bound	Records decisions with no task to key them to

Datatracker, 6 October 2026. Bold: OAuth or WIMSE working-group document; all others individual. W: submitted to WIMSE. Grey, under each column: outside that cluster's aim. Excludes the author's own drafts and identity-only, implementation-report and Best Current Practice (BCP) drafts.

Adjacent work covers each verb. Should it share one approval-backed root?

Backup: closing the agent authorization gaps

Agent Authorization Use Cases and Gap Analysis (draft-chen-oauth-agent-authz-use-cases-04): seven major gaps

Gap (section 6)	What closes it	Status
Pre-approval vs. continuous dialogue	Expansion to a successor Mission; Progressive Authorization	Answered, beyond the MVP
No standard interactive channel	Deferred Approval , ARAP , action-bound approval, suspended state	Answered, beyond the MVP
Delegation chains	Narrowing Token Exchange with the act chain; Child Missions; offline attenuation	MVP within one Mission; the rest beyond
Revocation: task-level and bulk	Revoke by mission_id; bulk through Mission Management; killing all of a principal's work stays with the credential issuer	Task-level in the MVP; principal-wide delegated
Per-client, not per-group	Child Missions partition the work; metering bounds the aggregate	Partial: shared state across domains is open
Grant layer vs. execution layer	Consent Evidence , action-bound approval, parameter_digest, runtime evidence	Partial: no user-signed confirmation
Machine-readable terminal disposition (new in -04)	mission_error and terminal Mission state say why; runtime retry and access-request signals	Partial; not yet in the crosswalk

Several gaps reduce to one missing object: the approved work

Backup: aligning Resource Access with the metadata drafts

Authority Metadata (Zehavi) and Domain-Specific Authorization Schemas (Liu): individual -00s, October 2026

To do this	Authority Metadata	Domain-Specific Authorization Schemas	Resource Access: align by
Say what a type conveys	Per scope or RAR type: resource_type, actions, effects, risk	A JSON Schema per type, descriptive, never authoritative for validation	Publishing its actions, action families and resource contracts there, not out of band
Discover it per resource	authority_metadata in resource metadata, bound to the resource identifier	schema_profiles in resource metadata; schema_profile_uri on a challenge	Folding mission_constraints_supported into those members
Register the type	No registry; vocabulary is the publisher's	Schema types registry, Specification Required	Registering mission_resource_access, with Common Constraints as its constraint vocabulary
Compare and narrow	can_be_attenuated_to names a candidate only; no set comparison	Each type defines its subset rule; Token Exchange only narrows	Supplying that rule: the concrete subset relation the binding's Narrow uses
Bound from above	Access requirement profiles: token and authentication, not authority	Guardrails: upper bounds that compose by intersection	Mapping a guardrail to the binding's policy ceiling
Identify things	Stable URIs recommended; equal strings need not mean the same	Type identifiers match exactly, never normalized	Keeping type identifiers exact; RFC 3986 normalization stays on resource values

Metadata says what a type conveys, schemas register it, Resource Access says how to compare it

Backup: Mission and WIMSE AIMS

AIMS	What it says	What the Mission supplies
§10.1	An Agent receives a Mission, often in natural language; translating it into authorization requirements is planning, out of scope	The other half: the agent proposes, the issuer derives, the approver decides, and the result is an addressable record
§10.7	A person's confirmation binds to an AS-issued grant	The approval event and the recorded approval_basis; authority_hash commits the Authority Set only
§11	Auditors reconstruct the authorization context behind an action	The mission claim on every token, the anchors, and evidence keyed to the Mission
Terminology	The Mission is the objective	The approved work created for it; its intent records the AIMS Mission

WIMSE identifies the agent. OAuth specifies the authority it acts under.

Backup: every registration and departure

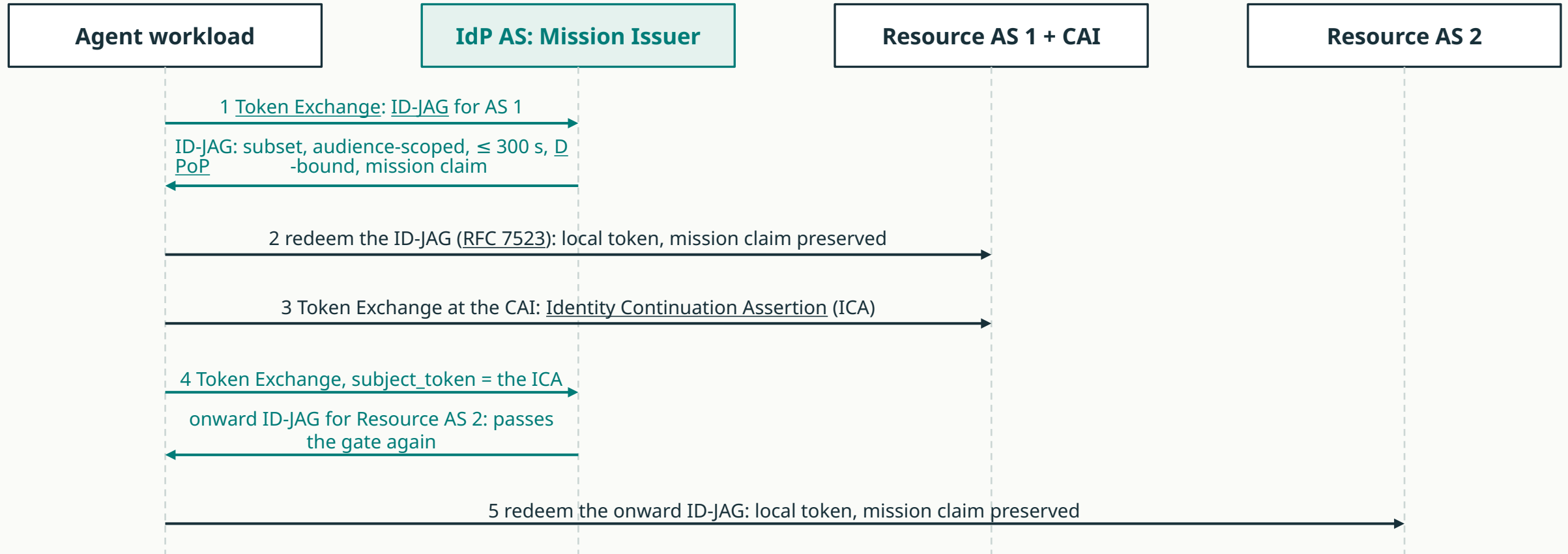
Registry	Entry
OAuth Parameters	mission_intent (PAR); mission_id, mission_expires_at, mission_error (token response)
OAuth Extensions Error	invalid_mission_intent_evidence (optional evidence path)
JWT Claims; Introspection	mission
AS and Resource Metadata	mission_bound_authorization_supported; mission_bound_authorization_required
WWW-Authenticate	mission_denial (no registry needed)
New registries	Mission Lifecycle States; Termination Reasons; Intent Members
Status: AS Metadata	revoke-only uses mission_lifecycle_endpoint, its auth methods, mission_status_signing_alg_values_supported (8 in all)
Status: media type	application/mission-status-response+jwt
Status: into the binding	state suspended; reason completed (full Status only)

Departs from, strengthens, or depends on

- RFC 7662 §2.2, §4: Mission state for an inactive token (deviation)
- RFC 7662 §4: no caching semantics for mission
- RFC 9396 §7: never omits granted values
- RFC 9101 §6.3: rejects a mission_intent outside the Request Object
- RFC 8693 §2.2.2: refuses a still-valid subject token
- RFC 8785 (JCS): normative downref
- Client instance ID draft: normative, individual
- Resource Access: RFC 3986 normalization beyond RFC 9396 §12

Binding: nine registrations, three registries. Status revoke-only: three metadata, one media type.

Backup: one Mission across many authorization servers



ID-JAG: Identity Assertion JWT Authorization Grant. CAI: Continuation Assertion Issuer. The handle grants nothing; every hop re-derives a subset under a live state check. Revoked: step 4 returns `invalid_continuation`. ICA hops stay within one IdP's trust; across a trust boundary, projection is single-hop.

Every onward hop asks the issuer again: after revocation, no AS gets a new grant

Backup: how far a revoke reaches, and how fast

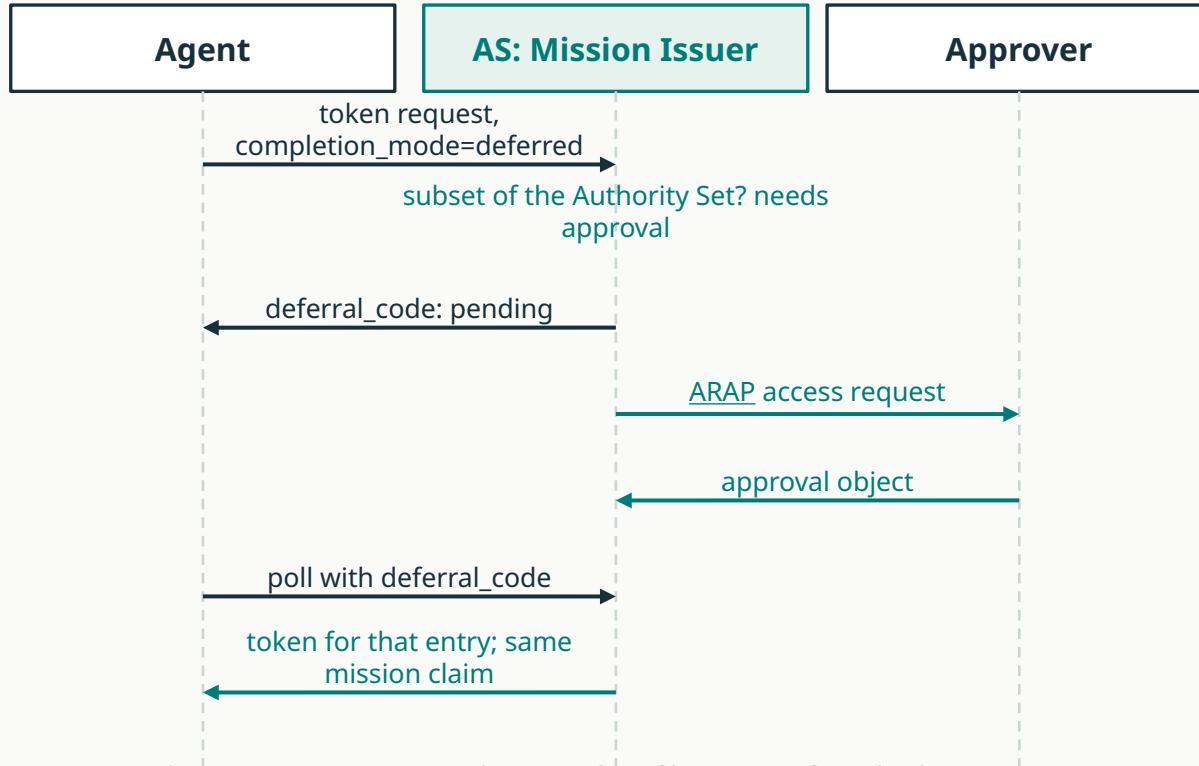
Mechanism	Draft	What it reaches	After the revoke
Issuance gate (MVP)	draft-mcguinness-oauth-mission	Code, refresh and Token Exchange refuse at once	Issued tokens run out
Token lifetime sizing	draft-mcguinness-oauth-mission-status	Every token	TTL SHOULD be $\leq$ mission_max_stale_seconds
Introspection	draft-mcguinness-oauth-mission	A resource that introspects	Per request: active false
Status query	draft-mcguinness-oauth-mission-status	A resource that checks per request	Per request, within fresh_until
Lifecycle Signals	draft-mcguinness-oauth-mission-signals	Subscribers on a Shared Signals stream	On delivery
Runtime enforcement	draft-mcguinness-mission-runtime	Each consequential action	Staleness bound plus permit window; under 300 s targeted
Child Missions	draft-mcguinness-oauth-mission-child-delegation	Sub-agents' own Missions	Cascade revocation
Cross-domain projection	draft-mcguinness-oauth-mission-cross-domain	A partner AS's issuance	Grant's remaining life, $\leq$ 300 s

No revoke reaches: work already done (orchestration unwinds it), running processes (a workload kill), or egress (network controls).

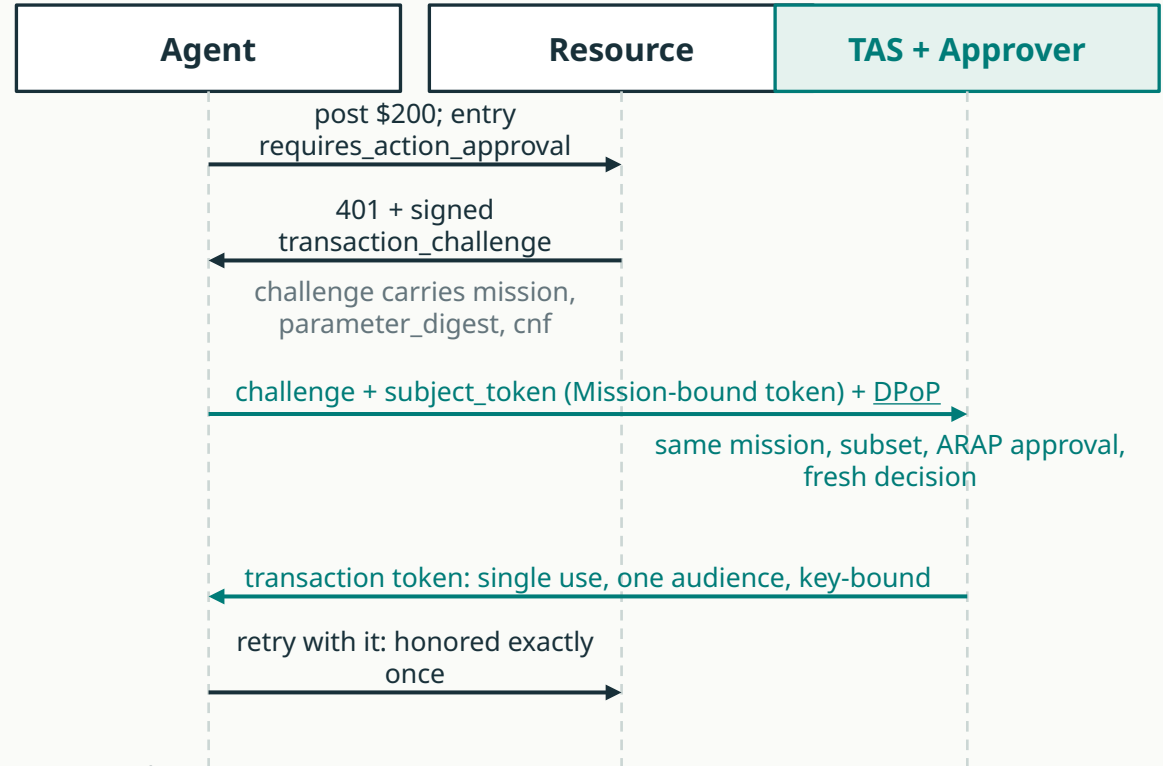
The gate stops new authority at once; each extension shortens the tail

Backup: fine-grained approval inside the Mission

Token challenge: DTR



Resource challenge: Transaction Challenge



ARAP: AuthZEN Access Request and Approval Profile. **DTR:** Deferred Token Response. **TAS:** Transaction Authorization Server.

Both stay inside the Mission: the request must already be a subset of the Authority Set. The approval is decision input, never authority. Widening is Expansion, a separate approval of a successor Mission.

Approval for one action or one entry adds a human pause. It never adds authority.

Backup: revising, delegating and metering a Mission

Revise

Never in place

Expansion: wider authority needs a fresh approval and a successor Mission; the predecessor ends as superseded

Progressive (experimental): drawdowns within a ceiling consented up front, adjudicated by policy; high-consequence classes still need a person

Delegate

Only ever narrower

Token Exchange: each hop a subset; the act chain records who acted

Child Missions: strict subset, expiry no later than the parent, own actor, fan-out limits, cascade termination

Offline attenuation: narrower tokens minted without the AS; runtime re-checks Mission state on every use

Meter

How much, consented up front

Bounds (experimental): max_budget, max_calls, max_duration, max_egress_volume, and exclusive

Runtime: atomic check and decrement, leases, settlement

Fail closed: a deployment that cannot meter a bound refuses it, never ignores it

Revise by new approval, delegate only narrower, meter what was consented

Backup: what evidence can key to a Mission

Recorded by the MVP: the OAuth binding's record and token

<code>intent_hash, proposal_hash</code>	<code>authority_hash</code>	<code>approval_basis</code>	<code>mission: issuer, id</code>
What was asked; hashed	What was approved; hashed	Who decided; not hashed	The join key on every token

Added by the Prove companions, each optional

<u>Consent Evidence</u> The exact disclosure the Approver saw, which the core does not commit	<u>Runtime Evidence</u> Decision, execution and refusal records; a portable Mission Receipt	<u>Mandate</u> A signed, portable statement of the committed facts: evidence, not a credential
<u>Audit Transparency</u> Tamper-evident, append-only, independently verifiable records in SCITT	<u>Approved-Set Verification</u> A resource server checks the subset itself instead of trusting the AS	<u>Work Products</u> Artifacts carry provenance, never authority

Adjacent work that could key to mission.id: agent audit record (gilda) · authorization evidence (munoz) · decision subject and operator of record (aravind) · AIMS §11: auditors reconstruct the authorization context

Limit: the anchors commit what was approved; they do not prove it was the right reading of the task. SCITT: Supply Chain Integrity, Transparency and Trust.

One approval record gives every evidence producer the same key

Backup: one Mission model on OAuth and on AAuth

	OAuth binding (the MVP)	AAuth binding, at the Person Server (PS)
Reference	issuer + mission_id; the mission claim	approving PS + s256; the mission_s256 claim. No second identifier
Approval	The AS renders the derived Authority Set for consent	Propose, clarify, approve at the PS; the person approves the mission description
Commitment	authority_hash, intent_hash, proposal_hash	s256 over the exact approved blob bytes
Authority	Typed RAR Authority Set; subset rule on every derivation	Each resource or Access Server decides: scopes, resource tokens, policy, optionally Rich Resource Requests (R3)
Gate	Every issuance path: code, refresh, Token Exchange	Structural only where the PS is on the path; agent-identity and resource-managed access are not gated
Lifecycle	active, terminated (revoked, expired); Status adds suspend and complete	active, terminated; reasons include completed, which the person must accept

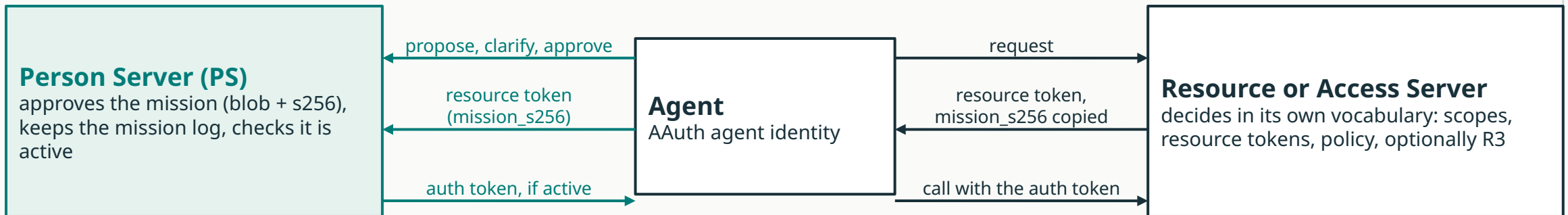
AAuth governs the context at the PS. The OAuth binding adds narrowing authority.

Backup: OAuth and AAuth bindings, side by side

OAuth binding (the MVP): the AS derives and gates authority



AAuth binding: the Person Server governs the context; each resource decides authority



Agent-identity and resource-managed access go straight to the resource: the PS is not on the path.

OAuth: one issuer derives and gates authority. AAuth: the PS governs context; resources decide.