OAUTH WG VIRTUAL INTERIM

Where OAuth hurts for general-purpose agents

Anthropic & OpenAI agentic use cases

Paul Carleton (Anthropic) · Nick Steele (OpenAI)

BEFORE WE START

Who are we?

Nick Steele

Member of Technical Staff, OpenAI

Agent Security and Identity

Paul Carleton

Member of Technical, Anthropic

Model Context Protocol and Authorization

BEFORE WE START

Why bother listening to us?

OAuth **LLMs**

We make a lot of OAuth connections

connections are what make LLM apps useful

Standardized is simpler use and simpler to secure

much easier to make connections safe when everyone does it the same way

BEFORE WE START

Practical problems only

Things we actually hit today. No philosophy, no hypotheticals.

BEFORE WE START

Listing and defining problems, not solutions

We'll focus on presenting problems. Please also ask questions about the problems. We'll mention some candidate solutions, and feel free to raise others, but this call isn't the place to pick one.

BEFORE WE START

Three sections in one hour

01

“Boring” problems

02

Principal selection

03

“Scope” issues

Discussion break after each section.

01

“Boring” problems

e.g. OBO

We're a general-purpose OAuth client, so…

Any integration friction is very visible
e.g. registration

User ↔ client ↔ server coordination is slow
and often opaque to all parties

Discovery is more important
zero pre-configuration much more common

Client registration: who does the integrating?

Then: the client developer

developer
1. Picks a specific server to integrate

→

developer ↔ server
2. Registers a client, tests, fixes, repeats

→

developer
3. Ships with a known-good client_id

→

user
4. Signs in. The pair has already been tested.

Now: the user

developer
1. Ships a general-purpose client. No server in mind.

→

user
2. Plugs in a server the client has never seen

→

client ↔ server
3. Discovers and registers at runtime.

→

user
4. Hits any failure first. Developer isn't in the loop.

Client registration

CIMD takes the registration step away

draft-ietf-oauth-client-id-metadata-document

But: “does my client work?”

private_key_jwt + no user account = no way to check. A DCR-style GET (RFC 7592) might help.

Consent screens: a lot of them, many different places

Too many consent screens

Corp ↔ personal crosstalk

Current work: ID-JAG

draft-ietf-oauth-identity-assertion-authz-grant

01 · “BORING” PROBLEMS

Refresh tokens: “do I want one? why didn't I get one?”

CIMD: no DCR feedback
ironically makes this harder

Hard to remediate
do the whole flow again

01 · “BORING” PROBLEMS

Leaked tokens are valuable → we want less durable tokens

Sender-constraining
e.g. DPoP

Shorter lived tokens / no refresh token
e.g. ID-JAG

Workload identity federation

DISCUSSION · 01

“Boring” problems

- General-purpose client: visible friction, slow coordination, discovery
- Client registration: CIMD, “does my client work?”
- Consent screens: too many, corp ↔ personal crosstalk, ID-JAG
- Refresh tokens: no feedback, hard to remediate
- Less durable tokens: sender-constraining, ID-JAG, workload identity federation

02

Principal selection

sub vs. act: the four setups

1 · user token, no actor

`sub = user · no act`

Plain delegated OAuth. What most chat integrations get today; the "agent" is only visible as the OAuth client

2 · user token, agent as actor

`sub = user · act = agent`

Explicit OBO. User in chat, or a user's cron job: their authority, agent named as actor. (tbd: which agent id to put in)

3 · agent token, user as actor

`sub = agent · act = user`

agent is the principal; a specific user triggered this call.

4 · agent token, no actor

`sub = agent · no act`

Agent on its own. Multiplayer with no single user to name, or fully autonomous runs.

blue = user · **orange** = agent · card colour follows sub · filled = act present, dashed = no act

What's a client? Multi-surface clients

One product, many surfaces

Codex: Codex app, Workspace Agents, Presence, Web

Claude: Claude code, Claude Tag, Claude web, Desktop

What's a client? Class vs. instance

Principal selection: new principal, or just metadata?

- Application publisher organization (Anthropic, OpenAI)
- Surface it’s in (e.g. previous slide)
- Longer lived project scope
- Task / Objective
- Conversation (e.g. slack thread)
- Turn (e.g. 1 message)

Cardinality problem, similar to DCR

Provisioning friction for “agent on its own” (OIO)

Borrow someone's login

leaks their private context

One shared API key

audit log: svc-agents ×40

A ticket per agent, per tool

nobody files 300 tickets

Want: just-in-time

One answer: Workload Authorization Grant

DISCUSSION · 02

Principal selection

- sub vs. act: chat (OBO), user-driven agent, multiplayer
- What's a client: class vs. instance, many surfaces, cardinality
- Provisioning friction: client creds everywhere vs. just-in-time (WAG)

03

“scope” issues

Scoping across OAuth integrations

- Scoping is currently too broad
  - Often has varying levels of support by providers
  - Connected platforms can offer additional scopes
    - Additionally confusing

- As a result, often used in conjunction with other tools, platforms, security boundaries

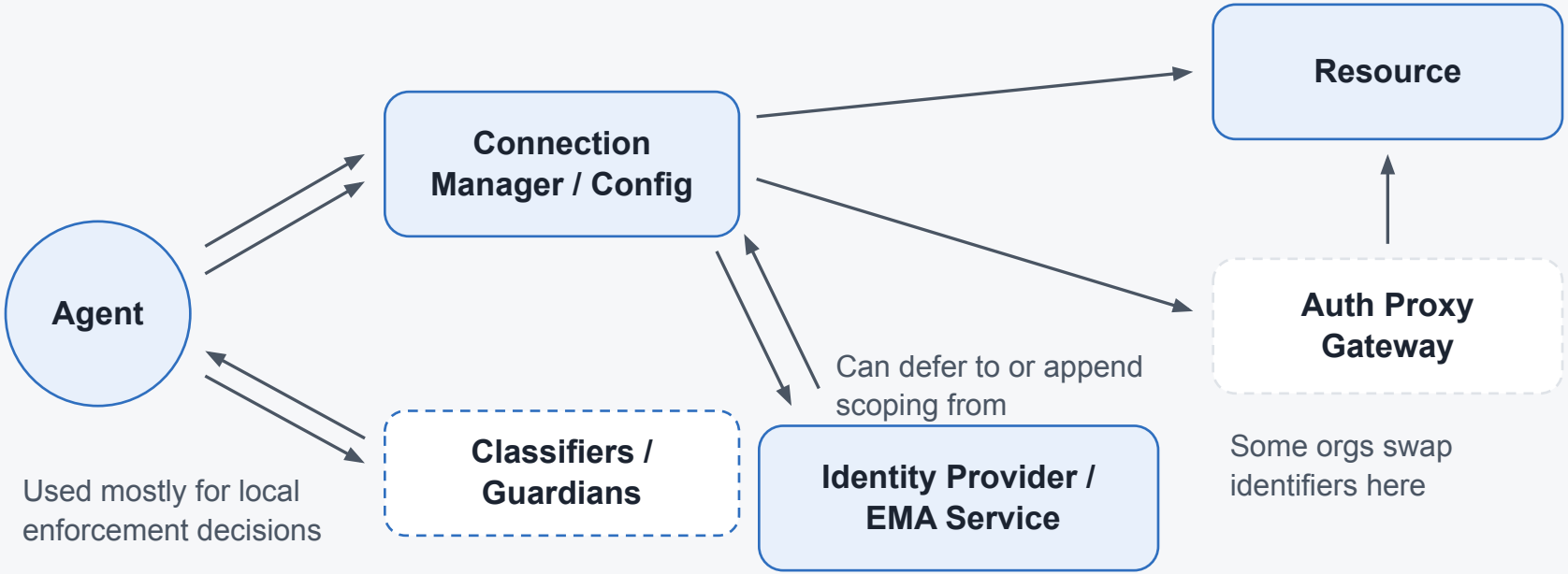
- Hard to centralize enforcement, most of which is client side

One agent can require multiple configurations

Example:
Review this change, resolve findings, and update the discussion.

Github	**Slack/Notion/Jira**
Read and modify code Repositories and files	Read and post updates Spaces and threads

One assignment must be mapped into multiple resources and *must* resolve.


Resource
Connection Manager / Config
Agent
Auth Proxy Gateway
Can defer to or append scoping from
Some orgs swap identifiers here
Classifiers / Guardians
Identity Provider / EMA Service
Used mostly for local enforcement decisions

3 Phases of Trust Building

Always Ask

- Consistent with early ent deployments
- Tends to increase toil and decrease trust
- Additional friction when used in conjunction with existing authorization tools

Approval Patterns in Place

- Organizations now have framework in place
- Decreases need for human intervention but still often requires multiple consent flows / MFA

Patterns + Agent Guards / Automode

- Helps deal with broad scoping by putting trust/risk within another source. Highly contextual
- Can be used effectively with MFA, moving high-assurance decisions

High Trust / Low Assurance Models

Blanket Approval / YOLO

- Useful for an extremely limited range of deployments
- Requires effective containment to exist elsewhere, so not great with RAG/Resources

Synthetic/Assumed Identity

- The agent impersonates the delegating user/token
- Offloads the access problem to IdP and human identity controls
- Doesn't strongly identify the agent

Agent Guards only

- The agent impersonates the delegating user
- Offloads the access problem to IdP and human identity controls
- Unclear what the agent might be able to request

Scope names are service-specific and broad

Scopes often aren’t resource identifier *specific* in all scenarios in order to offset the per-resourcing problem.

Operations included

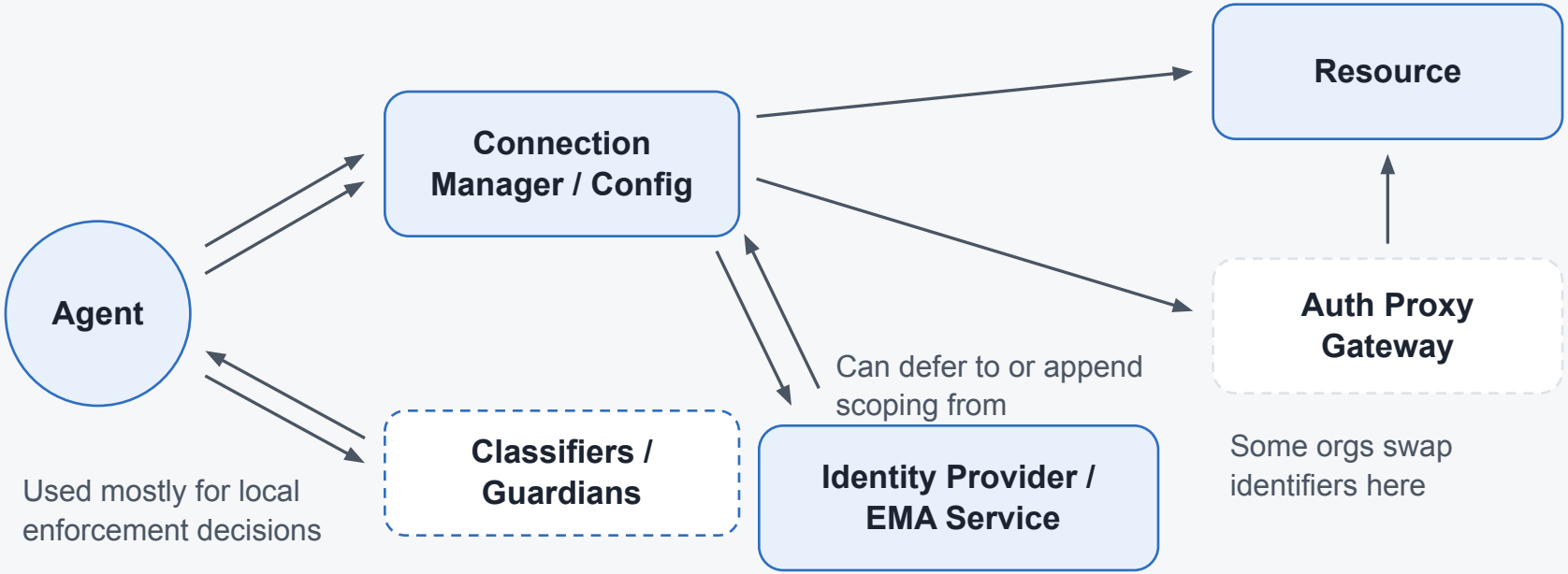
Read, search, export?

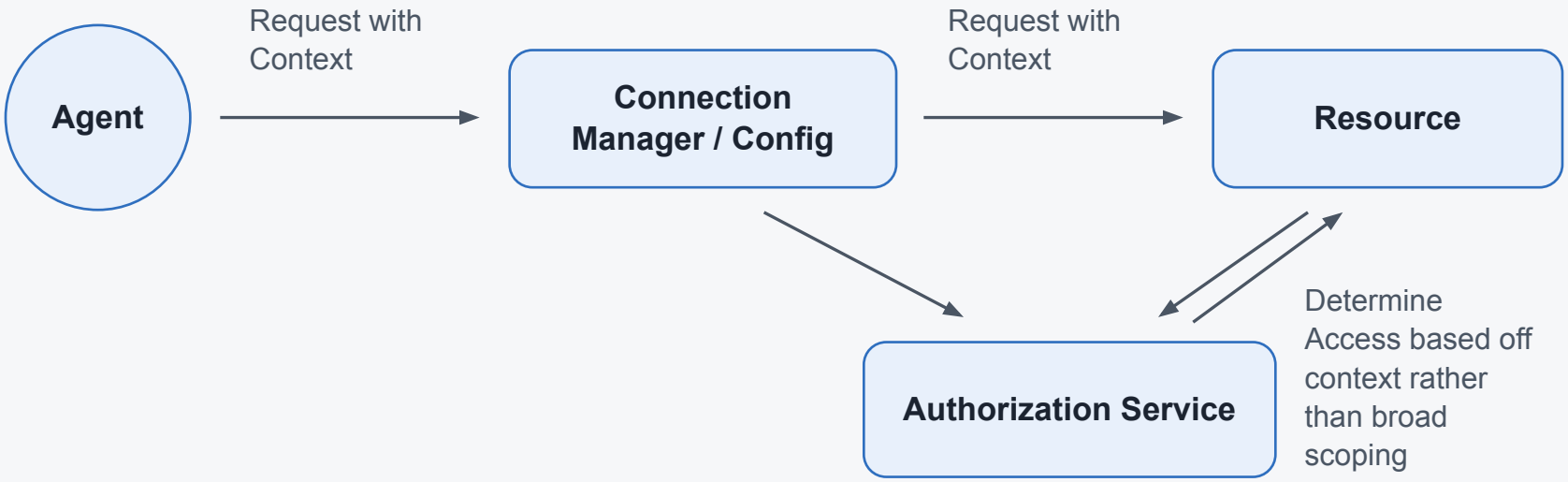
Objects included

One object, collection, account?

What must the client learn to interpret this service’s access?

Most Enforcement is Client side




Agent
Request with Context
Connection Manager / Config
Request with Context
Resource
Authorization Service
Determine Access based off context rather than broad scoping

DISCUSSION · 03

“Scope” issues

- Scoping is too broad and enforcement too narrow
- Currently favors client side enforcement due to weight of context
- One agent can require multiple configurations, often needs to be “provisioned”
    - Requires building specific “roles” for agents to meet permission set
- Would like to move more context and enforcement capabilities towards the resource