BackstageCon North America 2026
@Salt Lake City, USA

HITACHI

Can Your Agent Execute Anything?

Authorization for Backstage MCP Actions

Masato Kozuka
IAM Specialist, Hitachi, Ltd.

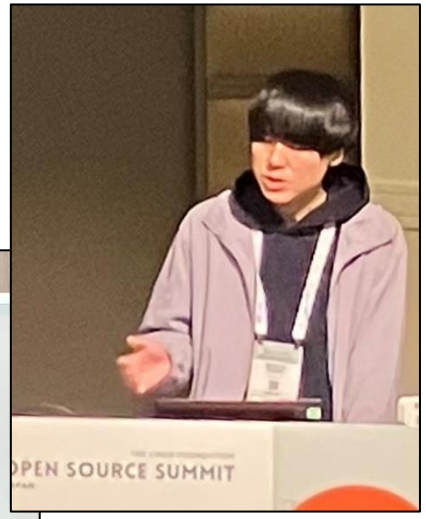
Date
November 9, 2026

©Hitachi, Ltd. 2026. All rights reserved

HITACHI

About me





Masato Kozuka
IAM Specialist, Hitachi, Ltd. in Japan

Current AI Focus

- IAM architecture for enterprise AI platforms

Core Technical Background

- Enterprise IAM/SSO architecture, including Keycloak implementations
- Security consulting on API hardening & Zero Trust access
- Platform Engineering advisory for enterprise adoption

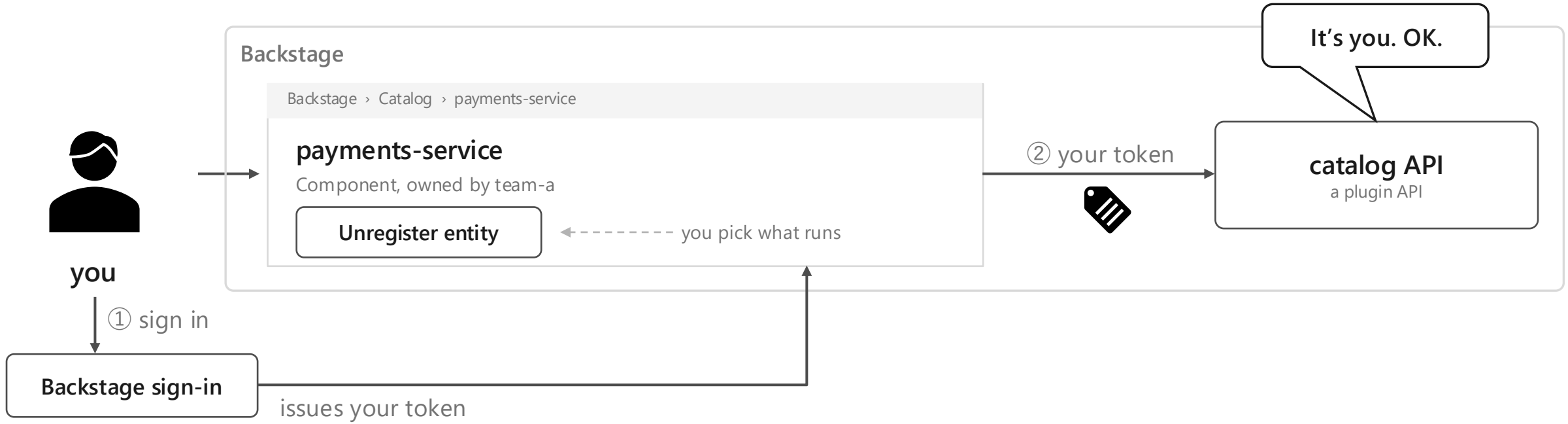
Community & Speaking

- Keycloak contributor & Backstage organization member
- Maintainer of Backstage Auth Plugin (for Keycloak)
- Speaker at AGNTCon+MCPCon NA 2026, Open Source Summit Japan 2025, etc.
- Upcoming speaker at KubeCon NA 2026

©Hitachi, Ltd. 2026. All rights reserved

HITACHI

Backstage was built for people at the portal

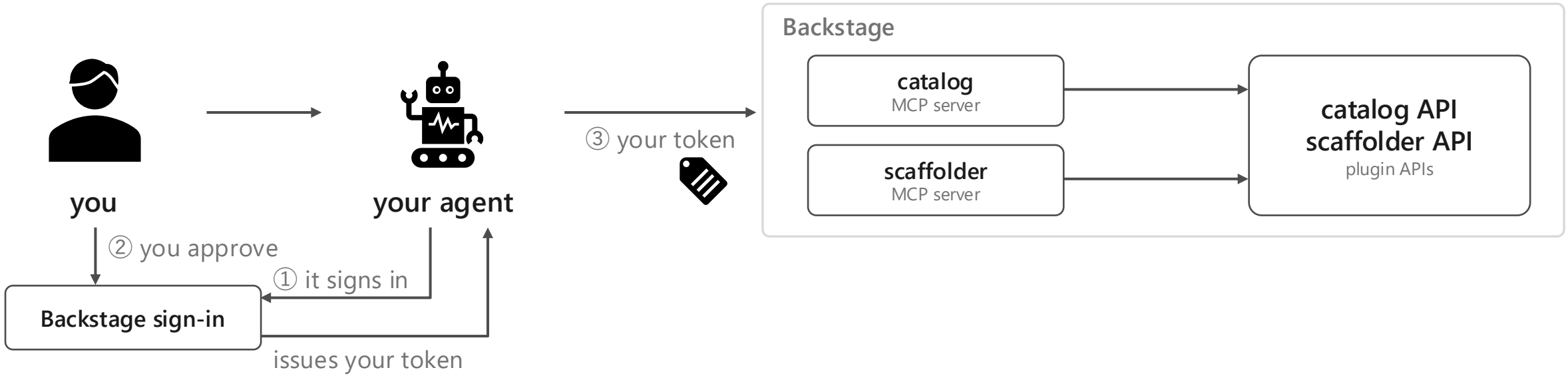


©Hitachi, Ltd. 2026. All rights reserved

HITACHI

Now you hand off the work to an agent

It signs in as you and calls Backstage over MCP.

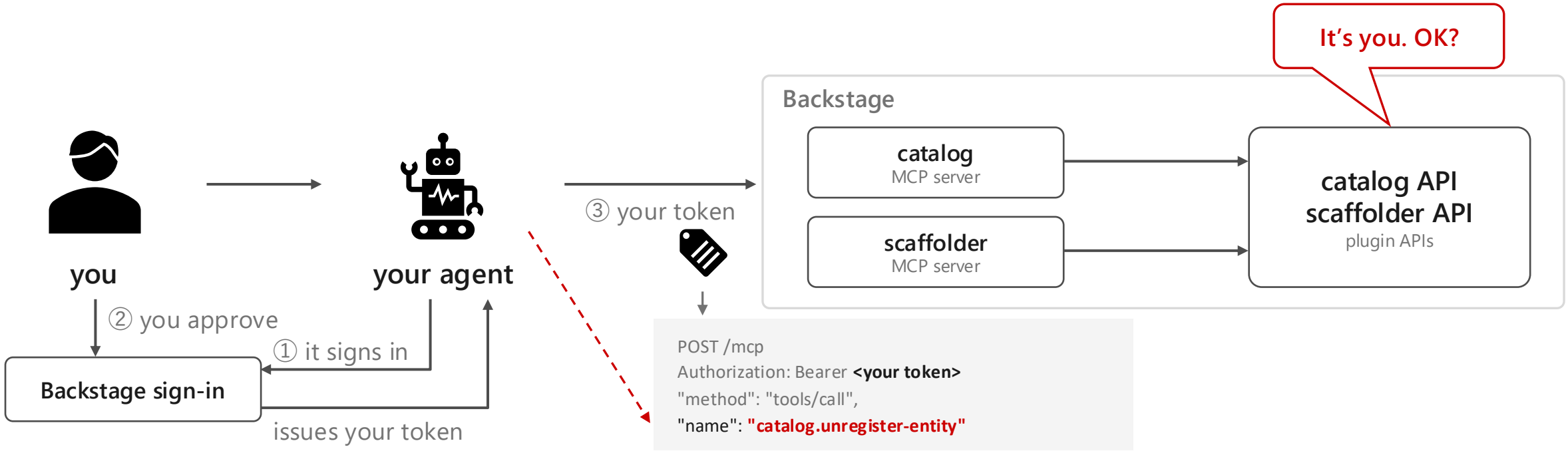


©Hitachi, Ltd. 2026. All rights reserved

HITACHI

Now you hand off the work to an agent

It signs in as you and calls Backstage over MCP.

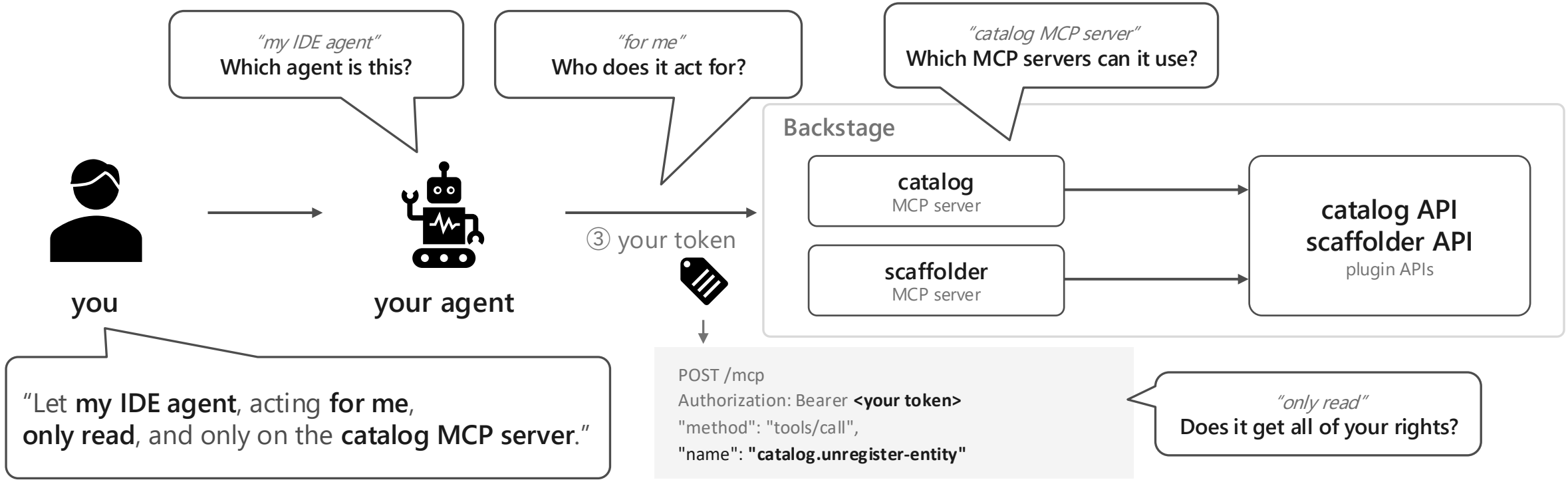


©Hitachi, Ltd. 2026. All rights reserved

HITACHI

Backstage needs four answers about your agent

To do what you want, Backstage needs an answer for each part.

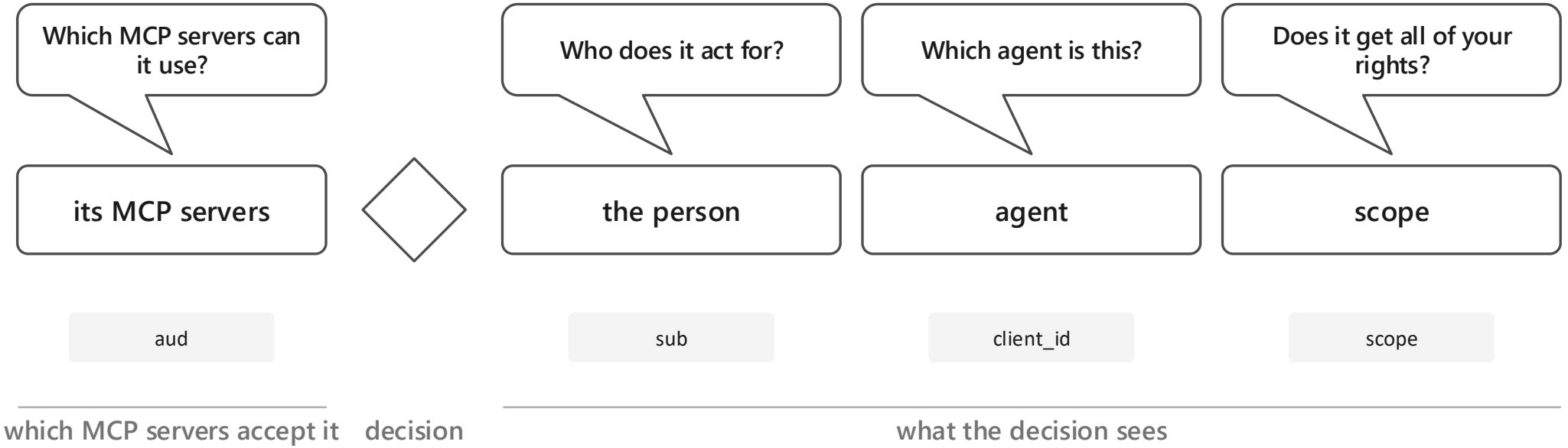


©Hitachi, Ltd. 2026. All rights reserved

HITACHI

Each question about your agent has a token claim

MCP builds authorization on OAuth: the token carries these claims, and each MCP server checks it.

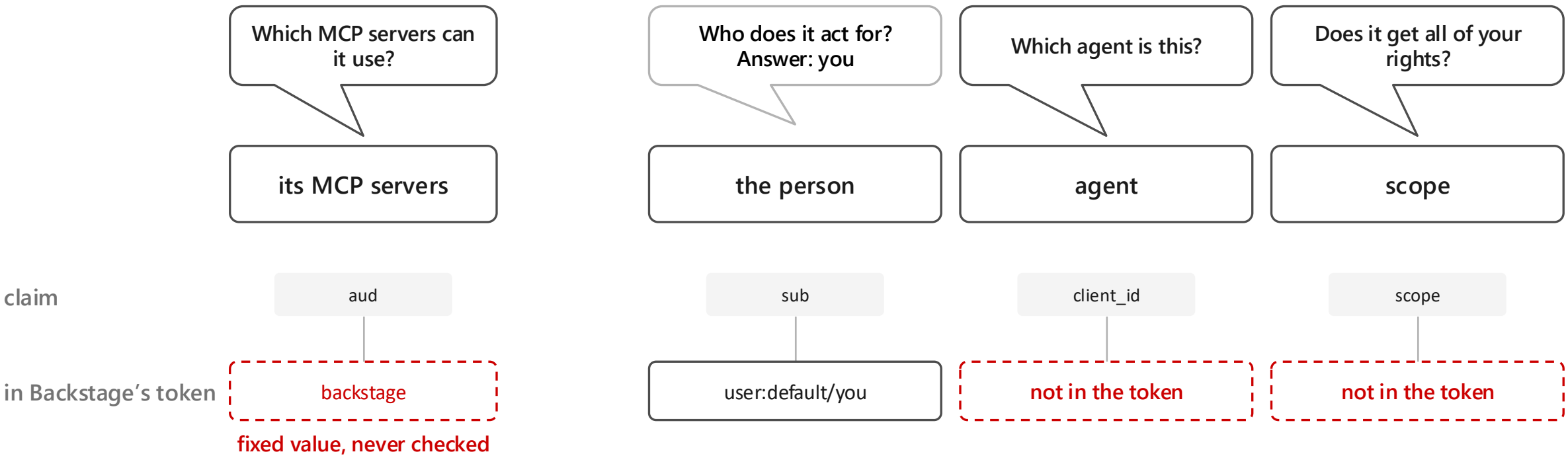


©Hitachi, Ltd. 2026. All rights reserved

HITACHI

Backstage’s token answers only who it acts for

The other three have standard claims; the token leaves them out or unchecked.

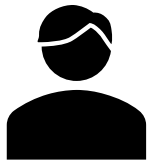


©Hitachi, Ltd. 2026. All rights reserved

HITACHI

Today: It can run anything you can

Even with permissions on, Backstage treats your agent as you.

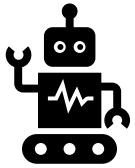


you

=

Backstage sees the same caller

user:default/you



your agent

With only you in the token, you can't narrow one agent:

your agent still gets it; you'd take it away

By action

chat | register-entity

IDE | **register-entity**

client_id: not in the token

By MCP server

chat | catalog | scaffolder

IDE | catalog | **scaffolder**

aud: fixed, never checked

By scope

chat | catalog.read | catalog.write

IDE | catalog.read | **catalog.write**

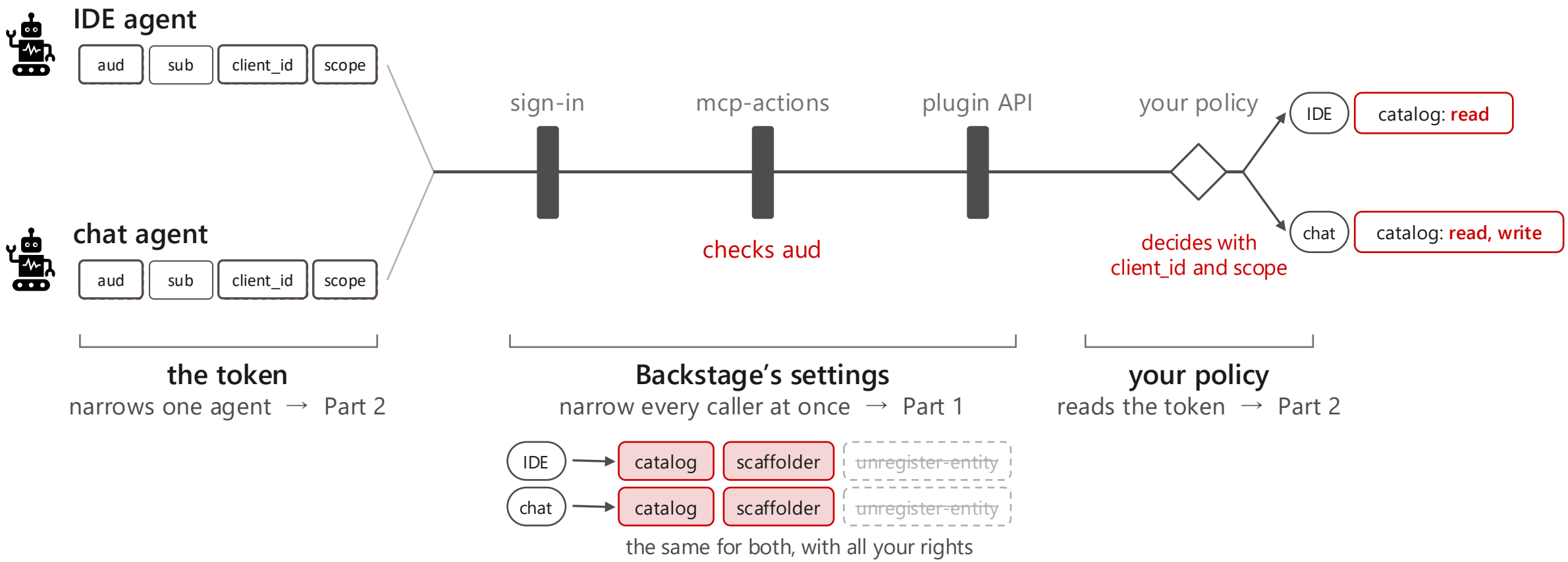
scope: not in the token

©Hitachi, Ltd. 2026. All rights reserved

HITACHI

Two places narrow your agent: settings, then the token

Settings work with today's token, but leave both agents the same tools and all your rights.

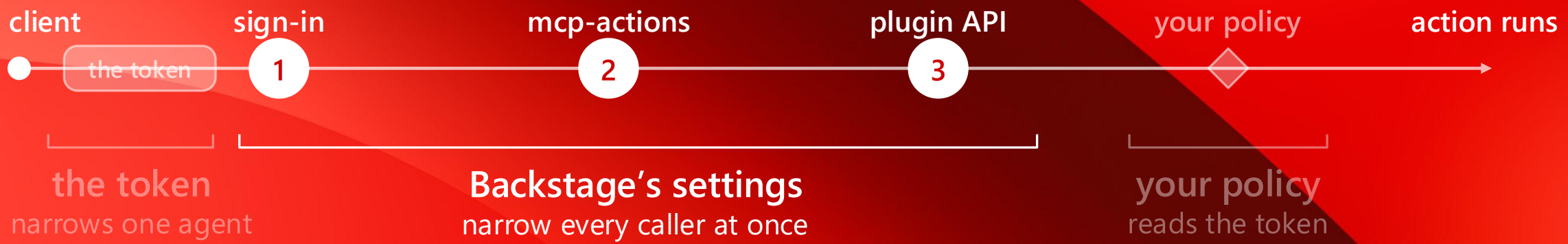


©Hitachi, Ltd. 2026. All rights reserved

HITACHI

Part 1

Three checks you can run today

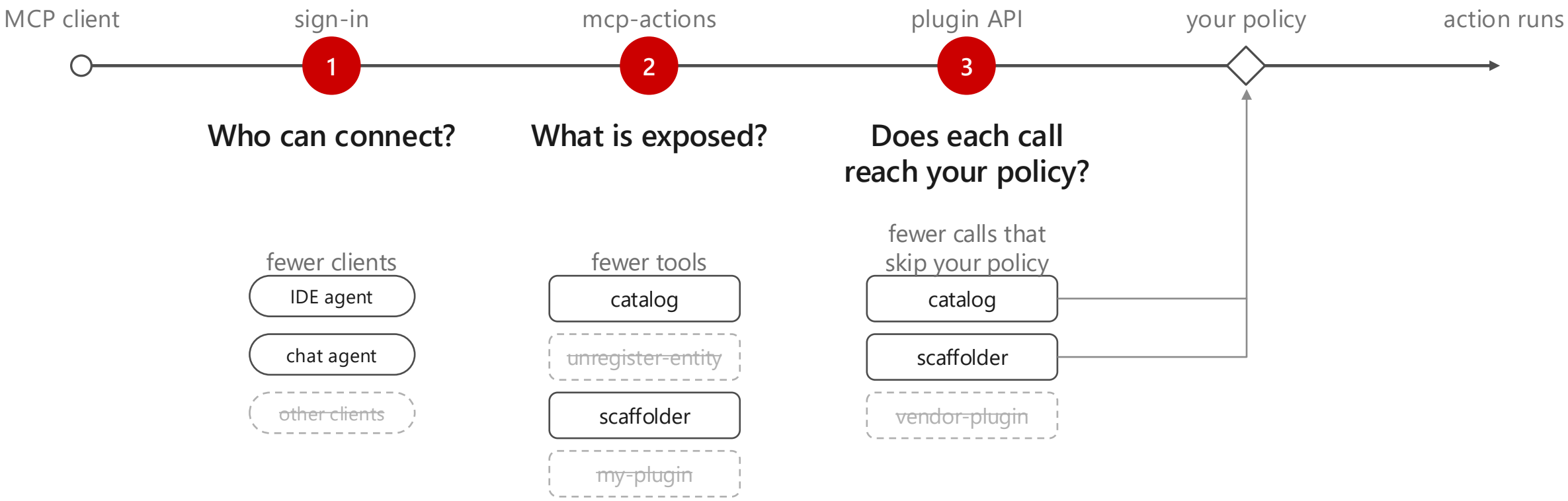


©Hitachi, Ltd. 2026. All rights reserved

HITACHI

The three checks follow the path of a call

Each check narrows the path for every caller.



©Hitachi, Ltd. 2026. All rights reserved

Clients you didn’t approve sign in, and static tokens skip sign-in

Once CIMD* is on, four clients are allowed by default.

Clients that can call mcp-actions today

signed in through CIMD, as you	
Claude	●
VS Code	●
ChatGPT Codex	●
Backstage CLI	●

static token you set in app-config, as itself
chat agent
ci-job

4 clients you didn’t approve

They are on the default list.

- connects without your approval

2 clients you set up

They use static tokens from app-config.

They skip sign-in, so your policy never runs for them.

* CIMD: Client ID Metadata Documents, how an MCP client tells Backstage who it is, without registering first.

©Hitachi, Ltd. 2026. All rights reserved

Narrow sign-in to the clients you approve

Inspect Which clients and static tokens can call mcp-actions?

app-config.yaml

```
auth:
  clientIdMetadataDocuments:
    enabled: true
    allowedClientIdPatterns:
      - 'https://ide.example.com/*'
      - 'https://chat.example.com/*'
backend:
  auth:
    externalAccess:
      - type: static
        options:
          token: ${CHAT_TOKEN}
          subject: chat-agent
      - type: static
        options:
          token: ${CI_TOKEN}
          subject: ci-job
        accessRestrictions:
          - plugin: mcp-actions
          - plugin: catalog
```

Set 1 Set 2 Set 3

Clients that can call mcp-actions, after

signed in through CIMD, as you

~~Claude~~
~~VS Code~~
~~ChatGPT Codex~~
IDE agent
chat agent
Backstage CLI — built in, always kept

Set 1

static token you set in app-config, as itself

~~chat agent~~ — acts for you, moved to CIMD — Set 2
ci-job — Out of scope — Set 3

Set 1 Replace the default clients with the ones you approve.

Set 2 Move clients that act for a user from static tokens to CIMD.

Set 3 Restrict the rest to the plugins they need.

From here on, **every call for a user goes through sign-in.**

©Hitachi, Ltd. 2026. All rights reserved

Every action from the plugins in your config becomes a tool

By default, every caller sees all of them.

tools/list, as a signed-in user (excerpt)

catalog.get-catalog-entity	
catalog.register-entity	●
catalog.unregister-entity	●
catalog.refresh-catalog-entity	●
scaffolder.list-scaffolder-tasks	
scaffolder.execute-template	●
my-plugin.restart-service	●
vendor-plugin.sync-everything	●
vendor-plugin.unregister-source	●

9 tools, from 2 built-in and 2 example plugins

7 of them make changes
And your agent can call them, as you.

● makes changes

©Hitachi, Ltd. 2026. All rights reserved

HITACHI

Expose only the actions your agents use

Inspect Call tools/list, and compare it with what your agents use.

app-config.yaml

```
backend:
  actions:
    pluginSources:
      - catalog
      - scaffolder
      - vendor-plugin
    filter:
      exclude:
        - id: '*:unregister-*'
```

Set 1

Set 2

tools/list, after (excerpt)

```
catalog.get-catalog-entity
catalog.register-entity
~~catalog.unregister-entity~~                  Set 2
catalog.refresh-catalog-entity
scaffolder.list-scaffolder-tasks
scaffolder.execute-template
~~my-plugin.restart-service~~                  Set 1
vendor-plugin.sync-everything
~~vendor-plugin.unregister-source~~            Set 2
```

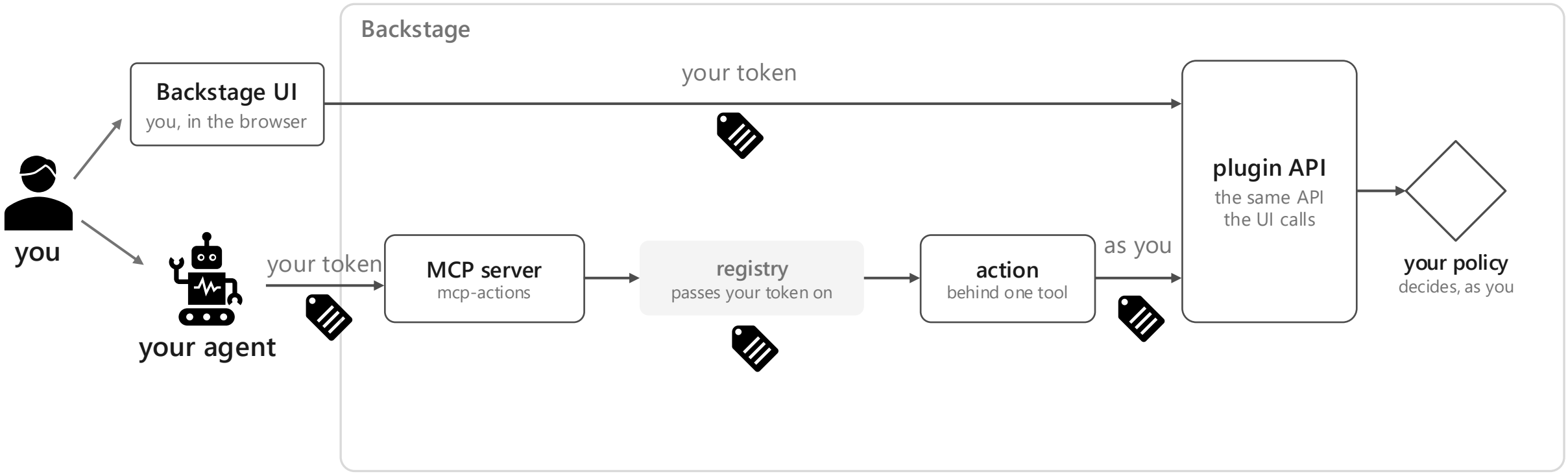
Set 1 Keep only the plugins they call.

Set 2 Exclude the actions they never need.

Every caller now sees **only the tools you kept.**

©Hitachi, Ltd. 2026. All rights reserved

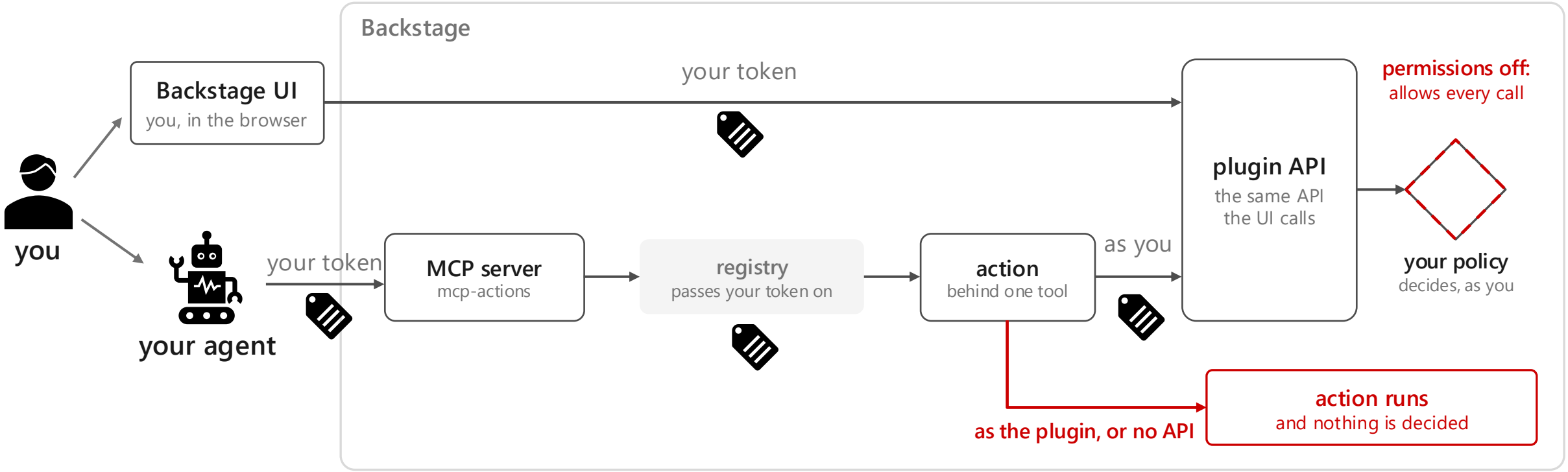
Your policy decides each call in the plugin API, as you



©Hitachi, Ltd. 2026. All rights reserved

A call reaches your policy only if two things are true

The action calls an API as you, and permissions are on.



©Hitachi, Ltd. 2026. All rights reserved

Turn on permissions, and drop what may skip your policy

Inspect Are permissions on? Read each tool's action: does it call an API as you?

app-config.yaml

```
permission:
  enabled: true
backend:
  actions:
    filter:
      exclude:
        - id: '*:unregister-*'
        - id: 'vendor-plugin:*'
```

Set 1

Set 2

in actions you write

```
await permissions.authorize(
  [{ permission,
     resourceRef: input.entityRef }],
  { credentials });
```

Set 3

tools/list after Check 2

✓ calls an API as you

catalog.get-catalog-entity
catalog.register-entity
catalog.refresh-catalog-entity
scaffolder.list-scaffolder-tasks
scaffolder.execute-template

? may skip your policy

~~vendor-plugin.sync-everything~~ Set 2

Set 1 Turn permissions on, so calls reach your policy.

Set 2 Filter out the tools that may skip your policy.

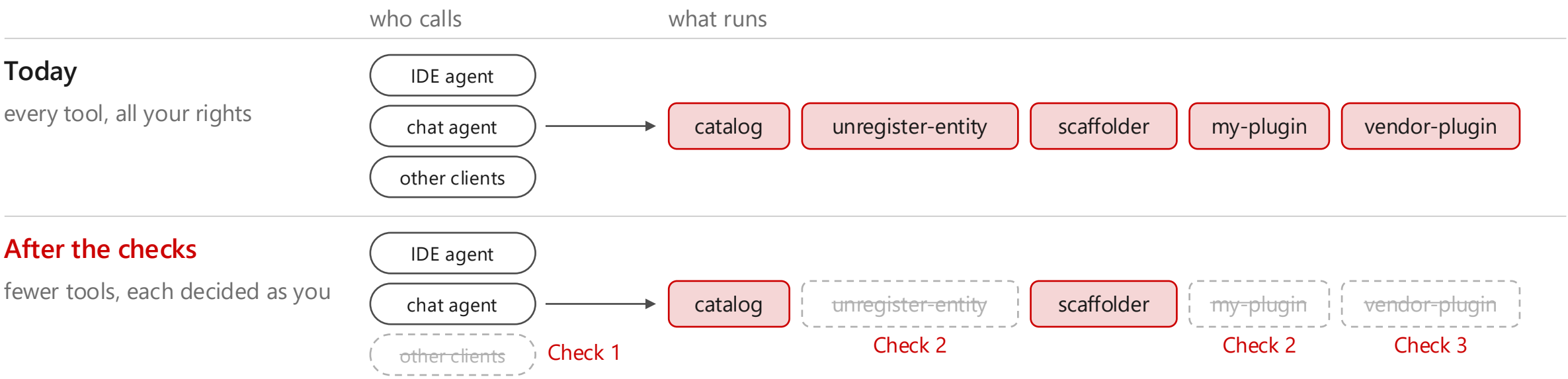
Set 3 In actions you write, ask your policy as the caller.

Every tool you kept is **decided by your policy, as you.**

©Hitachi, Ltd. 2026. All rights reserved

HITACHI

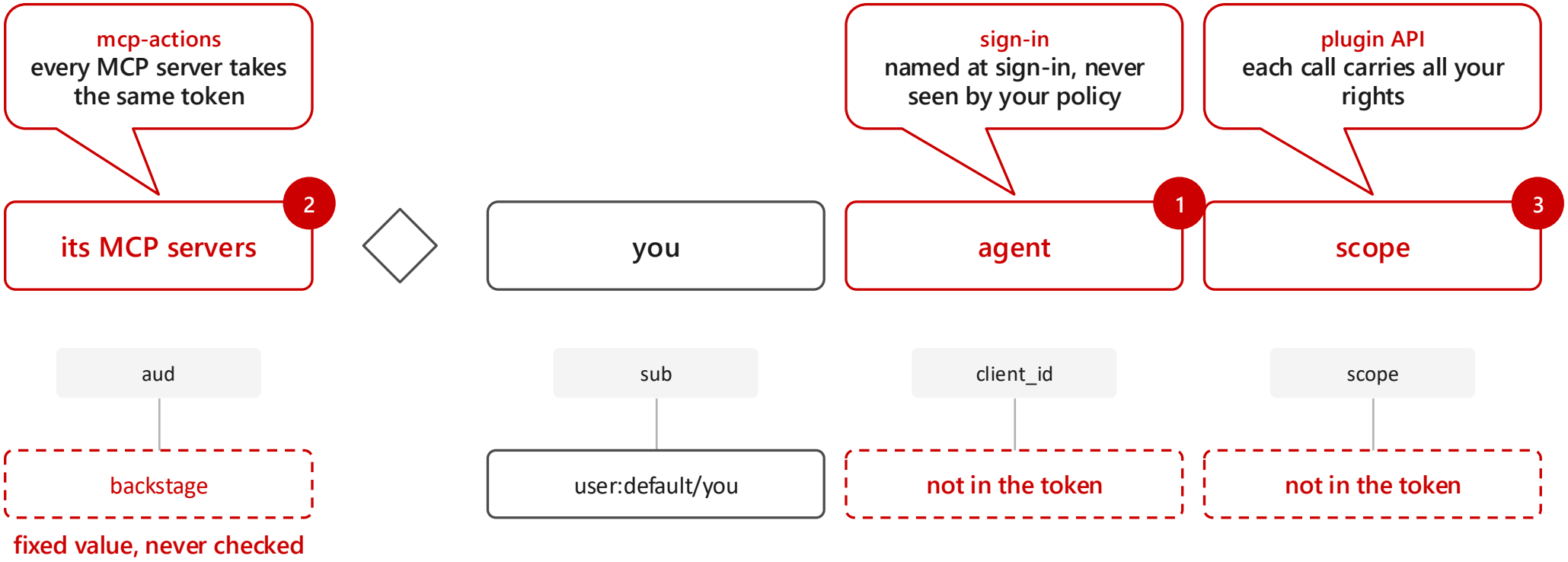
It runs fewer actions, and your policy decides each one



©Hitachi, Ltd. 2026. All rights reserved

HITACHI

Every leftover comes back to the token



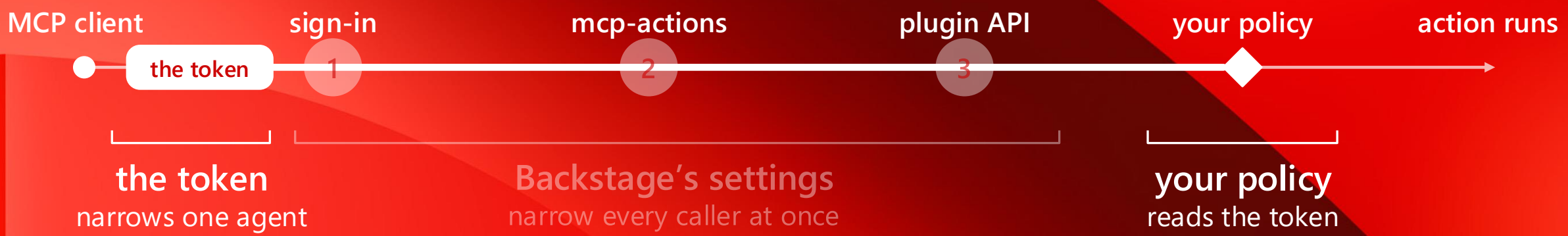
No setting adds these claims to the token.

©Hitachi, Ltd. 2026. All rights reserved

HITACHI

Part 2

A token change

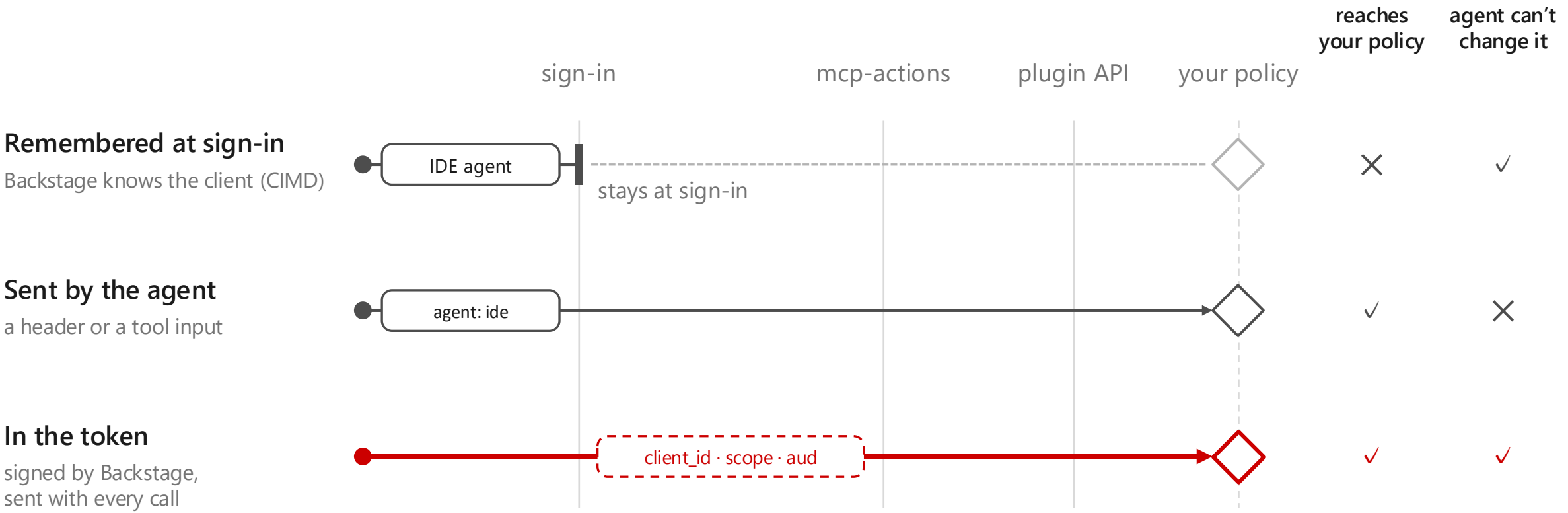


©Hitachi, Ltd. 2026. All rights reserved

HITACHI

Narrowing what one agent can do needs a token change

Each plugin decides per call, and the token is the only part of the call your agent can’t change.



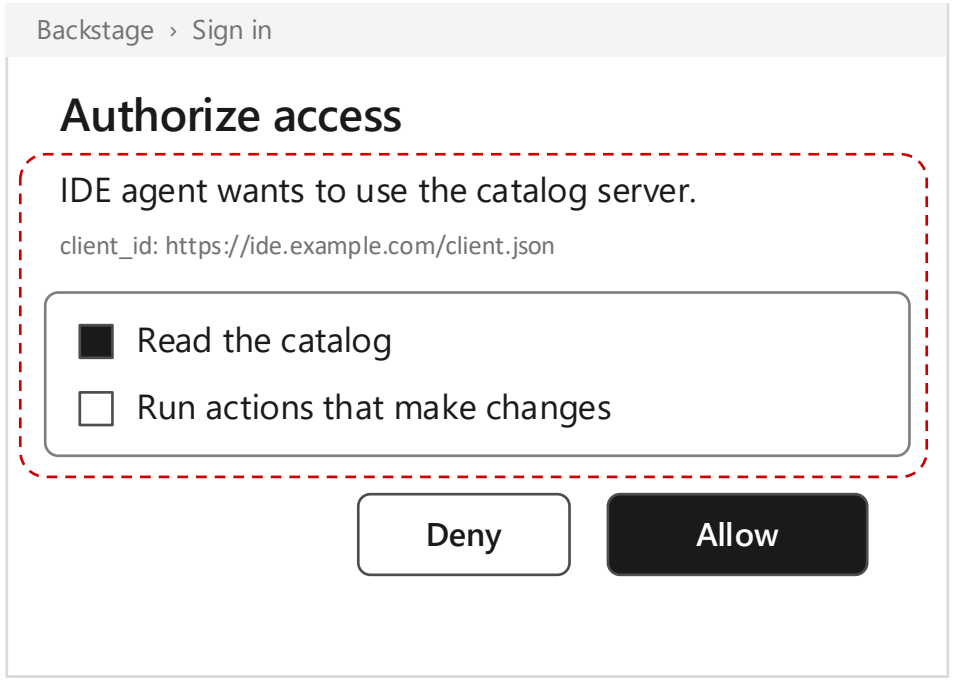
©Hitachi, Ltd. 2026. All rights reserved

Sketch: not in Backstage today

HITACHI

The token would carry the agent, its scope, and its server

MCP requires tokens bound to one server, and defines how to ask for scopes.



token issued


```
{
  "sub": "user:default/you",
  "client_id": "https://ide.example.com/...",
  "scope": "catalog.read",
  "aud": "https://.../api/mcp-actions/v1/catalog",
  "exp": 1762703600
}
```


Scope names are examples. What a scope means in Backstage is open.

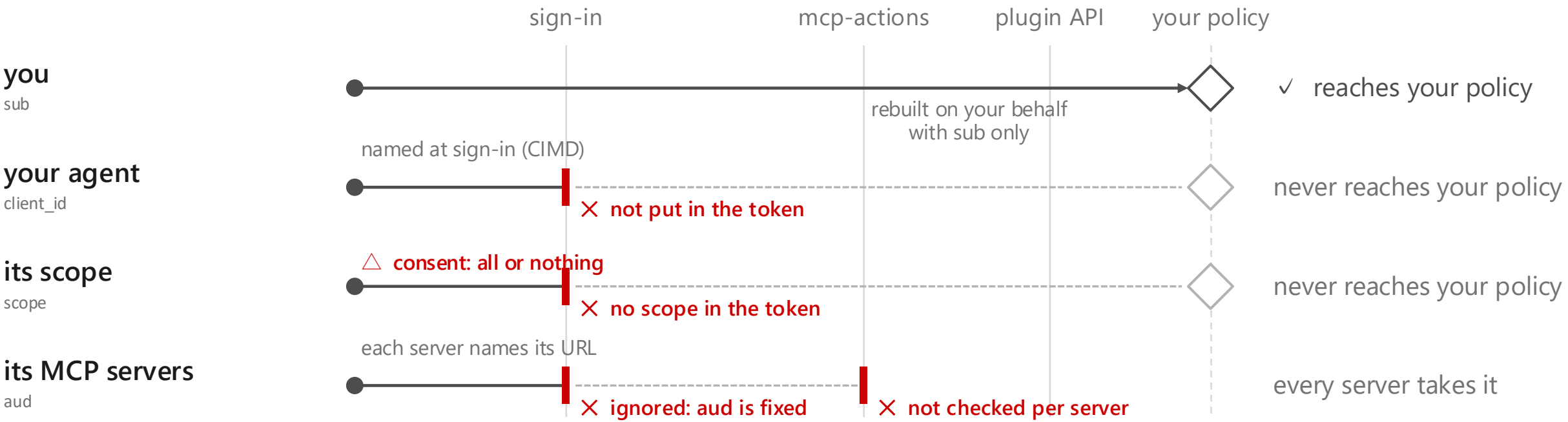
The permission framework would have to **carry them to your policy.**

©Hitachi, Ltd. 2026. All rights reserved

HITACHI

Today only you reach your policy

From Backstage's code (main, October 2026).



The path is already there; **the agent's facts never get on it.**

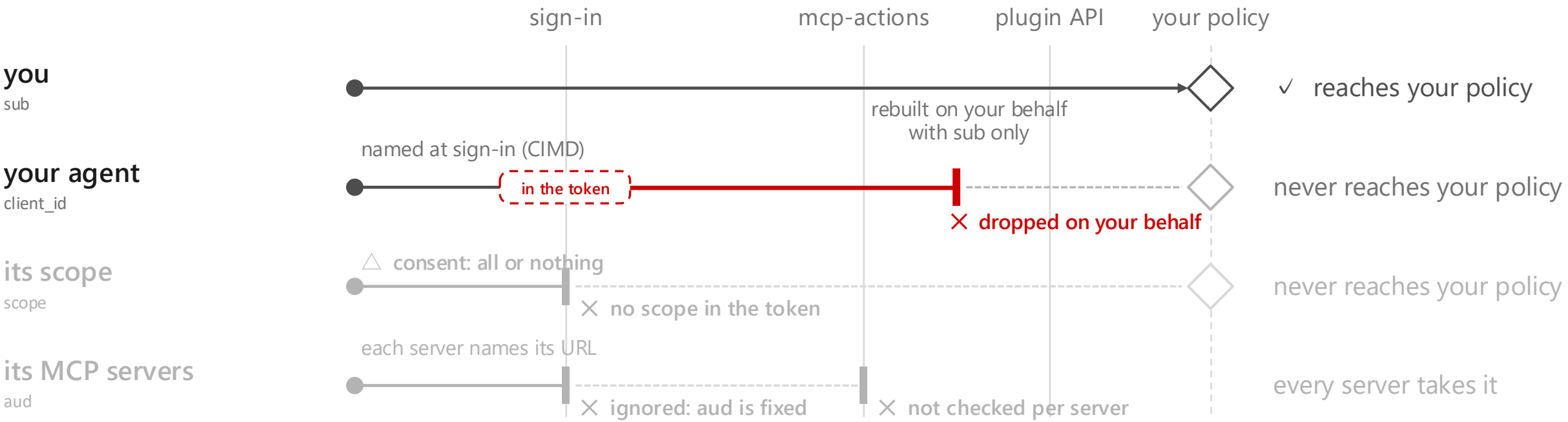
©Hitachi, Ltd. 2026. All rights reserved

Sketch: not in Backstage today

HITACHI

Every hop would have to carry the agent's facts

auth-backend (sign-in): issue the token with the client_id it already knows. It's dropped at the next hop.



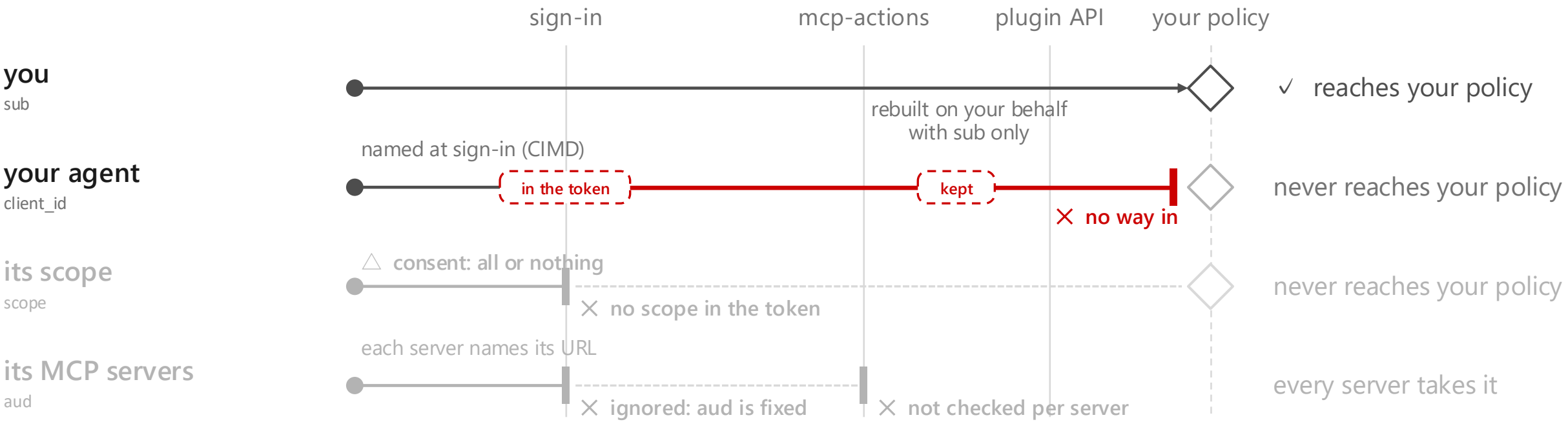
©Hitachi, Ltd. 2026. All rights reserved

Sketch: not in Backstage today

HITACHI

Every hop would have to carry the agent's facts

Backend auth service: keep client_id in on-behalf-of tokens. It reaches the plugin, not your policy.



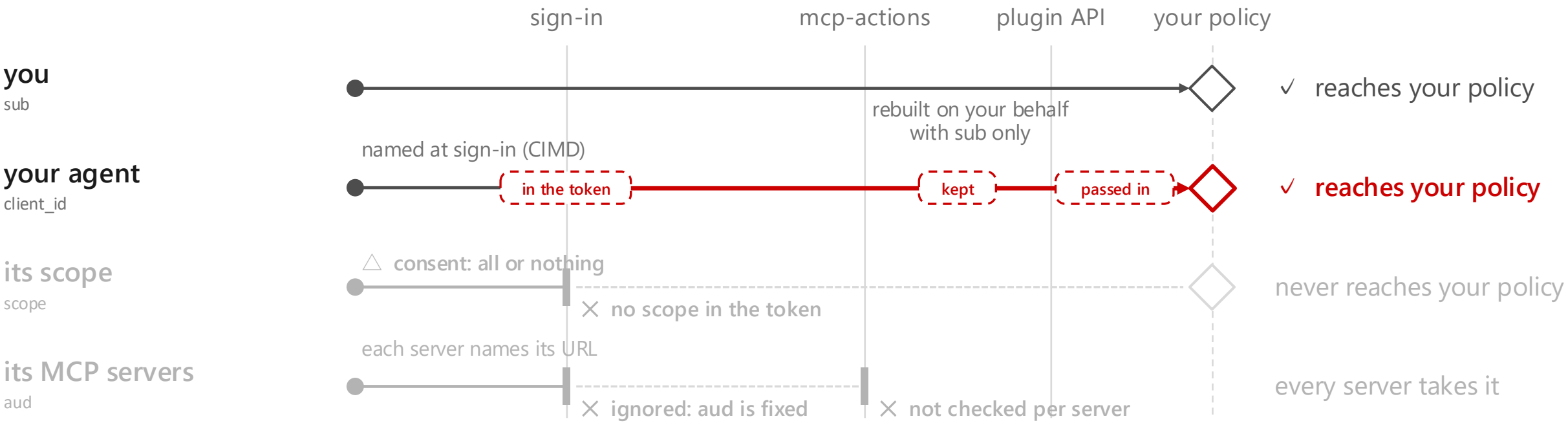
©Hitachi, Ltd. 2026. All rights reserved

Sketch: not in Backstage today

HITACHI

Every hop would have to carry the agent's facts

Permission framework: hand client_id to your policy. Now your policy sees which agent is calling.



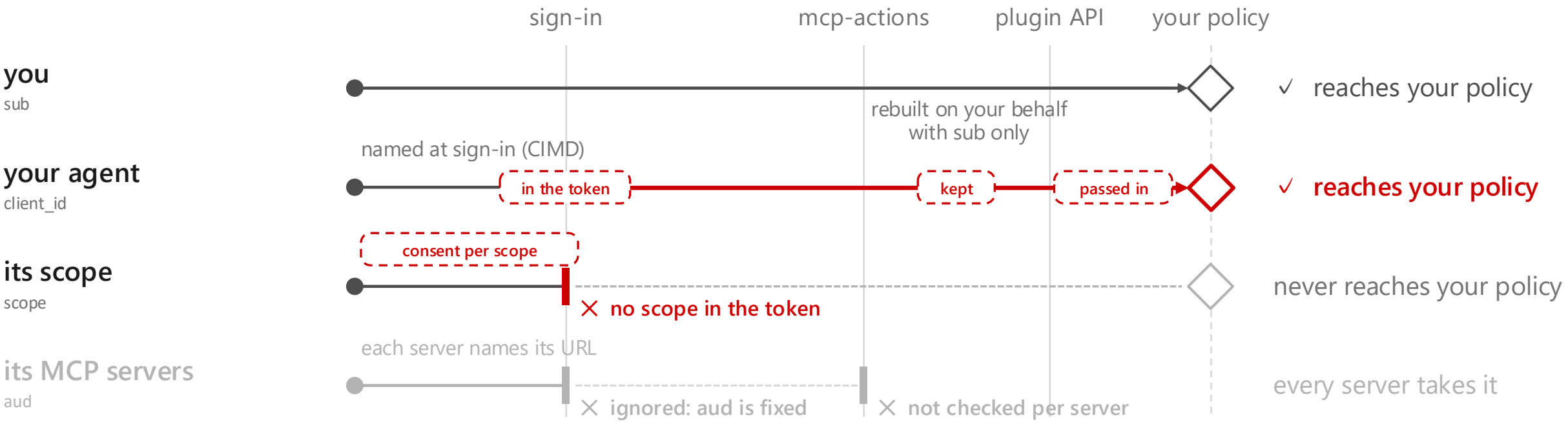
©Hitachi, Ltd. 2026. All rights reserved

Sketch: not in Backstage today

HITACHI

Every hop would have to carry the agent's facts

auth-backend: let the consent page list each scope for you to pick. The token still has no scope.



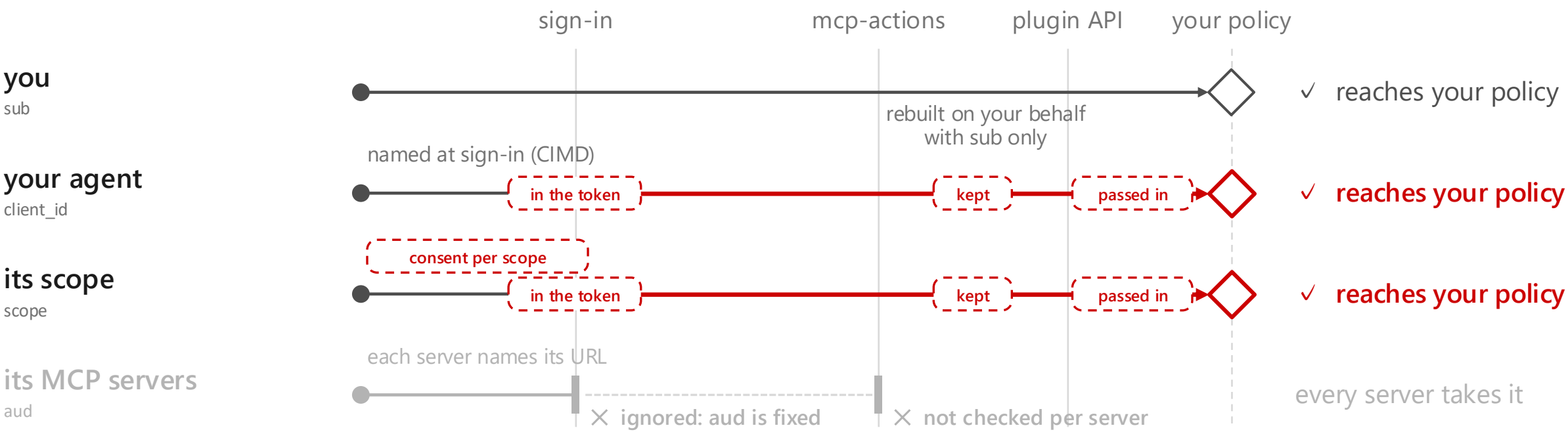
©Hitachi, Ltd. 2026. All rights reserved

Sketch: not in Backstage today

HITACHI

Every hop would have to carry the agent's facts

auth-backend, backend auth service, permission framework: the same three changes carry the scope.



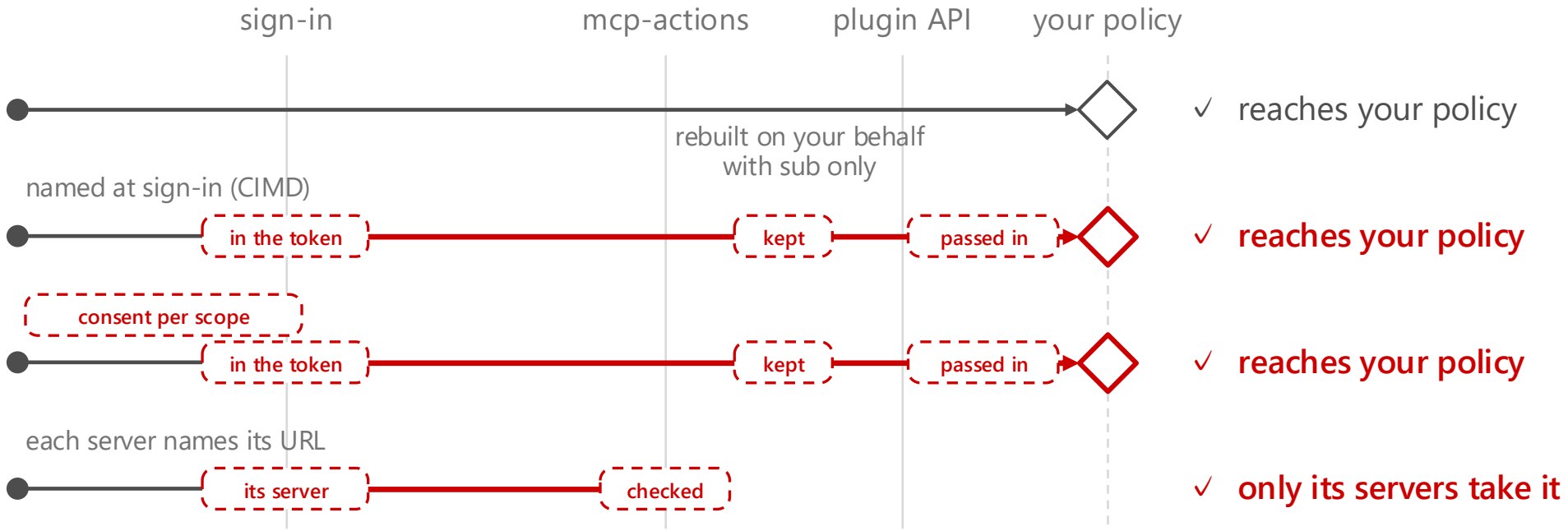
©Hitachi, Ltd. 2026. All rights reserved

Sketch: not in Backstage today

HITACHI

Every hop would have to carry the agent's facts

auth-backend: bind each token to a server you approved at consent.
mcp-actions: reject tokens for other servers.



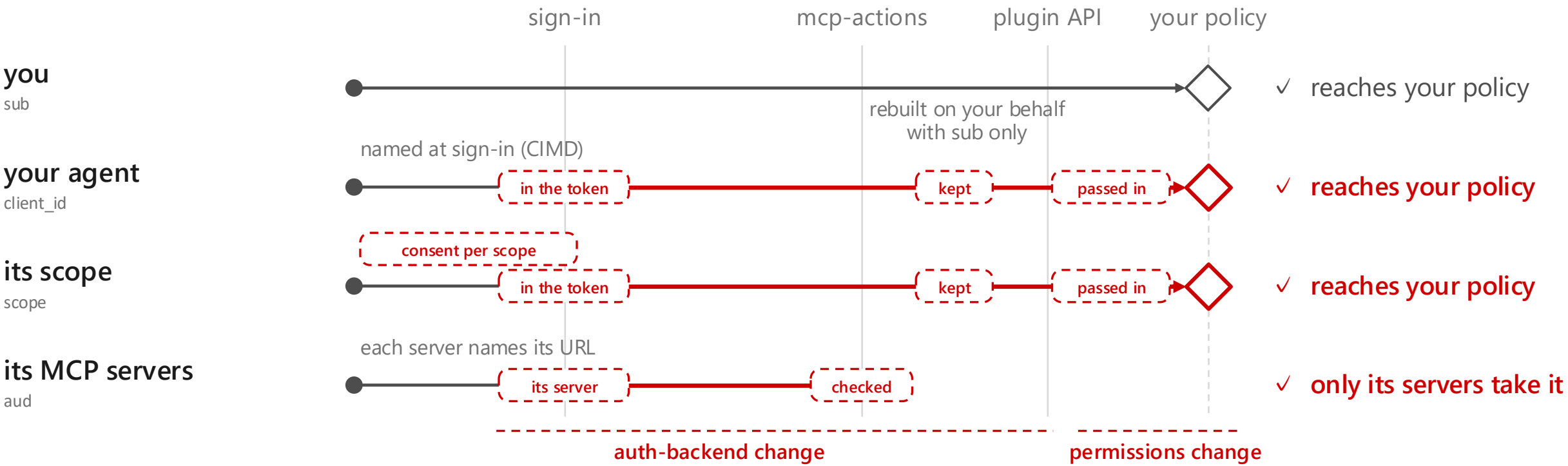
©Hitachi, Ltd. 2026. All rights reserved

Sketch: not in Backstage today

HITACHI

Every hop would have to carry the agent's facts

This needs two upstream changes: one for the token and each hop, and one for your policy.



Once they arrive, **your policy can use them.**

©Hitachi, Ltd. 2026. All rights reserved

Sketch: not in Backstage today

HITACHI

Your policy would decide with the agent and its scope

A write needs both: an agent your policy allows, and a scope you granted.

```
async handle(request, user) {
  const { action } = request.permission.attributes;
  const { agent, scope } = user.credentials.principal;
  if (agent && action !== 'read'
      && !(mayWrite(agent)
          && scope.includes(writeScopeFor(request.permission)))) {
    return { result: AuthorizeResult.DENY };
  }
  // your other rules, unchanged
}
```

— already there: **what the call does**
— new: agent and scope beside **who you are**

— mayWrite: **which agents may write** (your policy)
— **writeScopeFor:** the scope a call needs (not defined yet)

each row calls
catalog.register-entity (create)

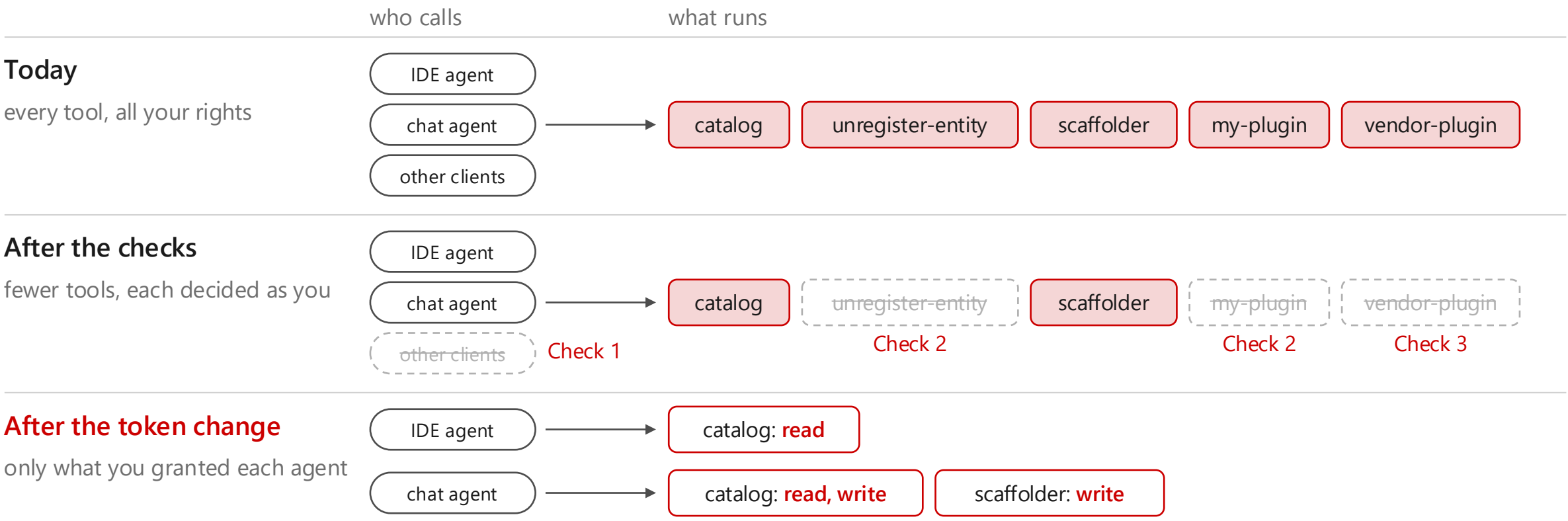
		agent your policy	**scope** you, at consent		
IDE agent	catalog.write	✗	✓	→ ◇	✗ **refused**
chat agent	catalog.read	✓	✗	→ ◇	✗ **refused**
chat agent	catalog.write	✓	✓	→ ◇	✓ **allowed**

©Hitachi, Ltd. 2026. All rights reserved

HITACHI

It runs only what you granted your agent

Once the token carries the agent, its scope, and its server, on top of the three checks.

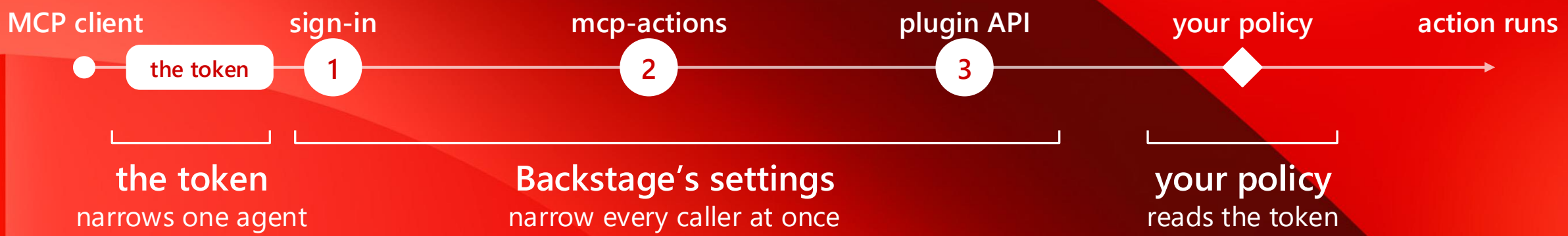


©Hitachi, Ltd. 2026. All rights reserved

HITACHI

Takeaways

Settings, then the token



©Hitachi, Ltd. 2026. All rights reserved

HITACHI

Settings narrow every caller. Only the token can narrow one agent.



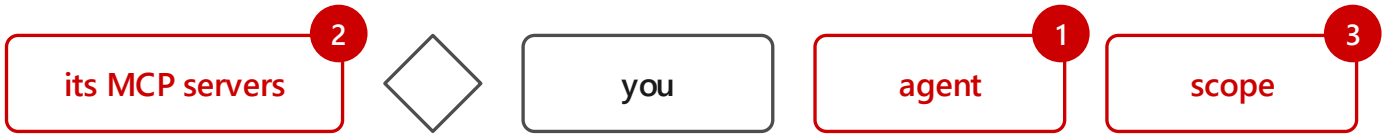
this week, in settings

Allow only approved clients
Restrict static tokens

Expose only the actions
your agents use

Turn on permissions
Drop what may skip your policy

left for the token



If you run agents against Backstage,
bring **your use case** to the upstream issue.

Upstream issue #36669

©Hitachi, Ltd. 2026. All rights reserved

Thank you

HITACHI

BackstageCon North America 2026
@Salt Lake City, USA

Can Your Agent Execute Anything?

Authorization for Backstage MCP Actions



Session feedback



LinkedIn

Masato Kozuka
IAM Specialist, Hitachi, Ltd.

Date
November 9, 2026

©Hitachi, Ltd. 2026. All rights reserved

HITACHI

Trademarks

- Model Context Protocol (MCP) is a trademark of LF Projects, LLC.
- Backstage is a trademark of The Linux Foundation.
- Claude is a trademark of Anthropic, PBC.
- ChatGPT and Codex are trademarks of OpenAI.
- VS Code (Visual Studio Code) is a trademark of Microsoft Corporation.
- GitHub is a trademark of GitHub, Inc.
- Other product names, company names, and logos mentioned in this presentation are trademarks or registered trademarks of their respective owners.

©Hitachi, Ltd. 2026. All rights reserved