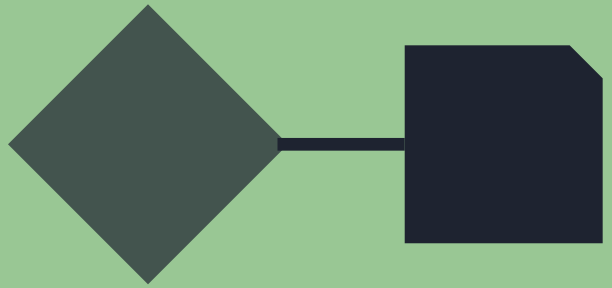


Stop registering, Start linking!

Moving MCP auth from **Dynamic Client Registration** to **Client ID Metadata Documents**

The problem

Agents want in , and they act *on a user's behalf*.

Why this is hard

OAuth assumes you've met before.

You can't pre-register a client you've never *seen*.

Before any token, a handshake.

01	**Request with no token**	→ 401 · WWW-Authenticate
02	**Discover Protected Resource & Auth Server Metadata**	RFC 9728 · 8414
03	**Get a `client_id`**	**DCR | CIMD**
04	**OAuth 2.1 + PKCE + resource**	→ Access token

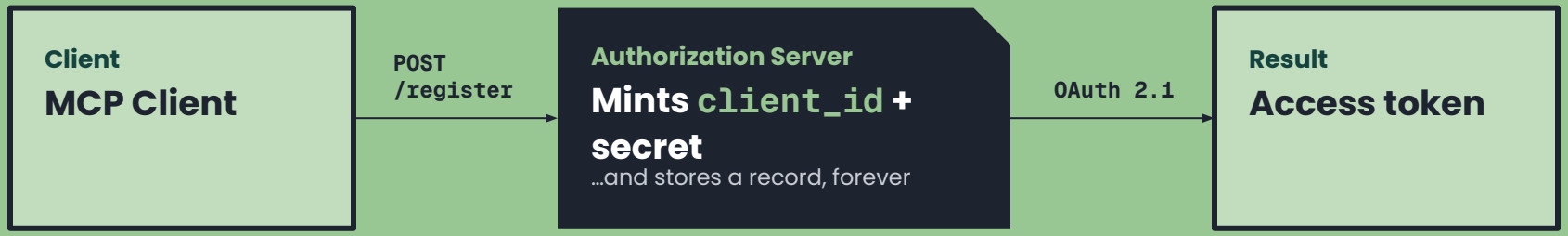
Before any token, a handshake.

01	Request with no token	→ 401 · WWW-Authenticate
02	Discover Protected Resource & Auth Server Metadata	RFC 9728 · 8414
03	**Get a `client_id`**	**DCR | CIMD**
04	OAuth 2.1 + PKCE + resource	→ Access token

This one step, how the client gets an id, is this whole talk.

Dynamic Client Registration · RFC 7591

Register on the fly.



One request, one identity.

REQUEST

```
POST /register
{
"client_name": "MCP Inspector",
"redirect_uris": ["…/callback"],
"grant_types": ["authorization_code"]
}
```

RESPONSE

```
201 Created
{
"client_id": "tpc_8SXWY6j3afl2…",
"client_secret": "••••••••••"
}
```

↳ opaque · server-minted · stored on both sides

Where DCR hurts.

Client-ID sprawl
A record per client, forever.

No verification
An open endpoint trusts anyone.

Per-server secrets
Store, rotate, revoke — everywhere.

Built for stable apps
MCP clients live for minutes.

Client ID Metadata Documents

What if the ID was the client?

The client publishes itself.

```
// client.json — hosted by the client
{
"client_id": "https://app.dev/client.json",
"redirect_uris": ["https://app.dev/cb"],
"token_endpoint_auth_method": "none"
}
```

01 **client_id = the URL**

02 **Server GETs + validates it**

03 **OAuth → access token**

The client publishes itself.

```
// client.json — hosted by the client
{
"client_id": "https://app.dev/client.json",
"redirect_uris": ["https://app.dev/cb"],
"token_endpoint_auth_method": "none"
}
```

01 `client_id` = the URL

02 Server GETs + validates it

03 OAuth → access token

No `/register`. Nothing stored.

The rules that keep it safe.

- HTTPS URL, with a path
- `client_id` is the URL
- No shared secrets
- Server fetches defensively
- Hostname shown at consent
- `redirect_uris` validated

What linking gives you.

The URL is the identity
One id, every server.

Domain ownership
A real signal of who's calling.

Stateless by design
Nothing to store or rotate.

Zero-setup onboarding
No trace left behind.

DCR vs CIMD.

	DCR	CIMD
Creates the id	Server, at runtime	Client, ahead of time
Proof of identity	None	Domain ownership
Server state	A record, forever	Nothing to store
Best fit	Long-lived apps	Ephemeral agents
MCP spec	`MAY`	`SHOULD`

Demo

Same server. Two identities.

Demo - DCR

Register.

```
Inspector connects → POST /oidc/register

token.client_id = tpc_8SXWY6j3afl2…
```

Demo - CIMD

Register.

```
Inspector uses client_id = https://…/client.json
```

```
token.client_id = https://…/client.json
```

CIMD is the default.

CIMD is the default.

DCR stays for long-lived, managed clients where a registration record is a feature, not a liability.

CIMD is the default.

DCR stays for long-lived, managed clients where a registration record is a feature, not a liability.

But for agents? Stop registering. Start linking.

In practice

Tomorrow morning.

01 Auth0 supports both DCR and CIMD* already.

02 Run them side by side, same API, same tokens.

03 Publish your doc at a stable HTTPS URL.

04 Keep DCR as the fallback.

*Manual Client ID Metadata Document (CIMD) registration only for now.

CIMD

It's not magic.

- It’s still an IETF draft, things could change.
- Proves origin, not trust.
- Confidential clients still need a key.
- Enterprise governance looks different.

MCP

The complete story.

- OAuth 2.1
- Client ID Metadata Documents
- On-Behalf-Of Token Exchange
- Resource identifiers

Resources.

MCP Authorization spec	`modelcontextprotocol.io`
CIMD draft	`draft-ietf-oauth-client-id-metadata-document`
Auth0 MCP docs	`auth0.com/ai/mcp`
Demo repo	`github.com/Sambego/auth0-xmcp-cimd`

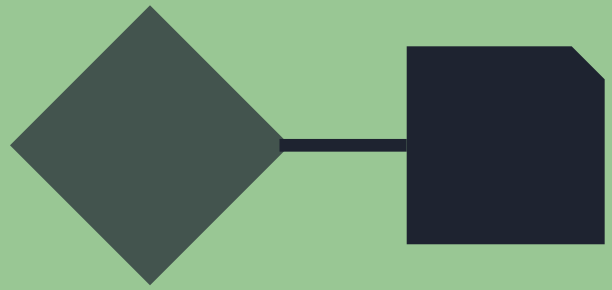
Sam Bellen

Principal Developer Advocate at Auth0

@sambego · sambego.tech

Slides are available

slides.sambego.tech/start-linking



Thank you.