



2025.6.13

MCPにおける認証・認可



ソフトウェアエンジニア

大坪 悠 Yu Otsubo



職歴：

新潟県出身。大学卒業後、2016年にKDDIに新卒入社。
通信設備の運用、社内データ基盤構築のPM、Webチャットボット開発などを経験する。
スタートアップへの転職を経て2024年にKAG入社。
KAGでは生成AIを活用したアプリケーション(A-BOSS - 本部長AI)開発に従事。

著書：

「K.A.G Tech Book」(KDDIアジャイル開発センター)
「やさしいMCP入門」(秀和システム)

@tubone24

tubone-project24.xyz

mugimugi.cutedog

趣味：

愛犬と遊ぶこと
個人開発



モットー：

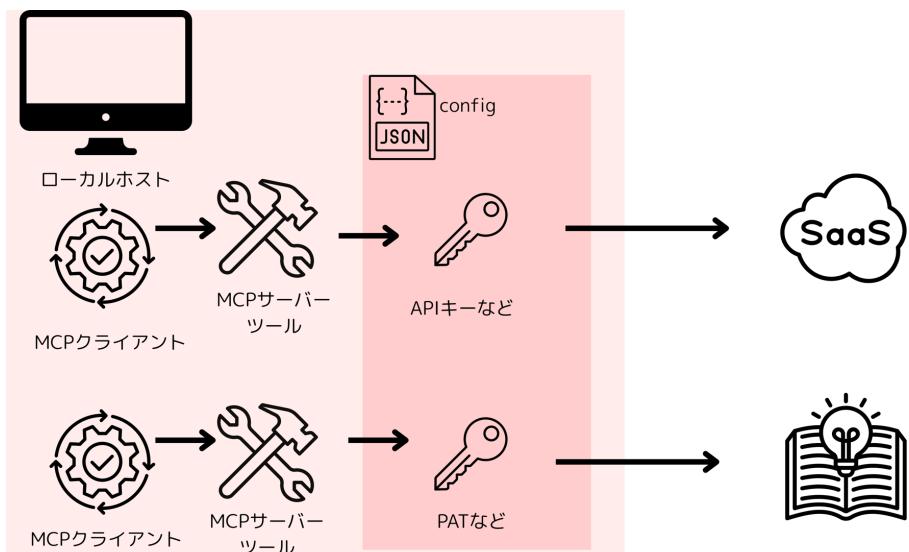
早く小さく失敗する



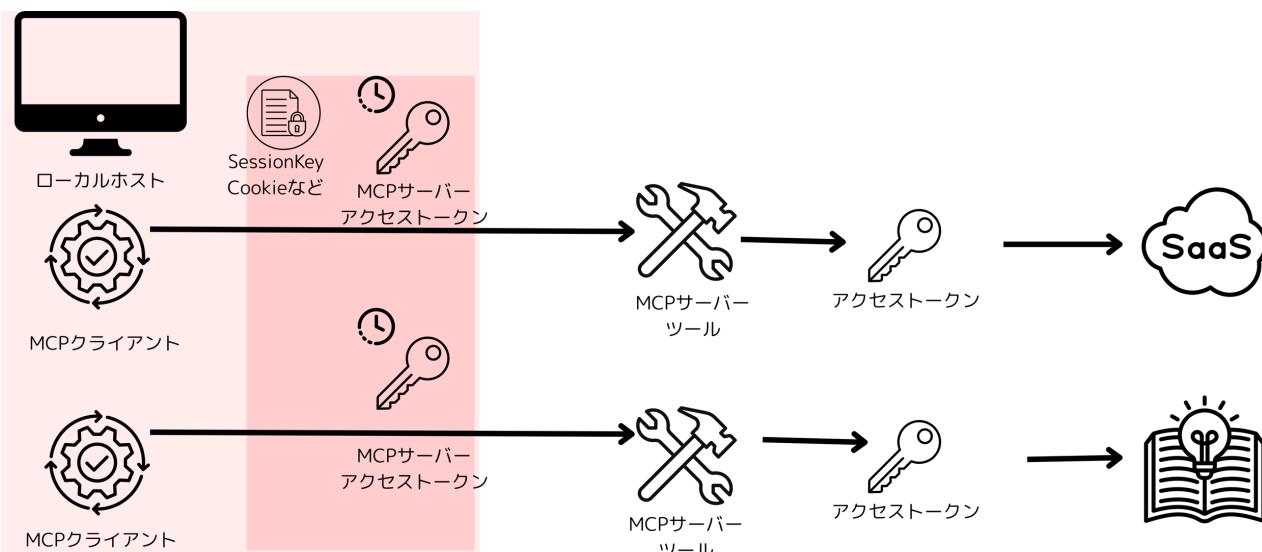
MCPにおける認証・認可の重要性

リモートMCPサーバーの普及や機微情報の扱いにおいて認証認可は重要

従来



これから



【💡ポイント💡】

ローカルMCPサーバーは接続コンフィグ(claude_desktop_config.jsonなど)に
強力なアクセストークンがそのまま記載されてしまう
一方でリモートMCPサーバーはMCPサーバー専用のアクセストークンを払い出す形

リモートMCPサーバーに強力なAPIキーを渡すことはできない!

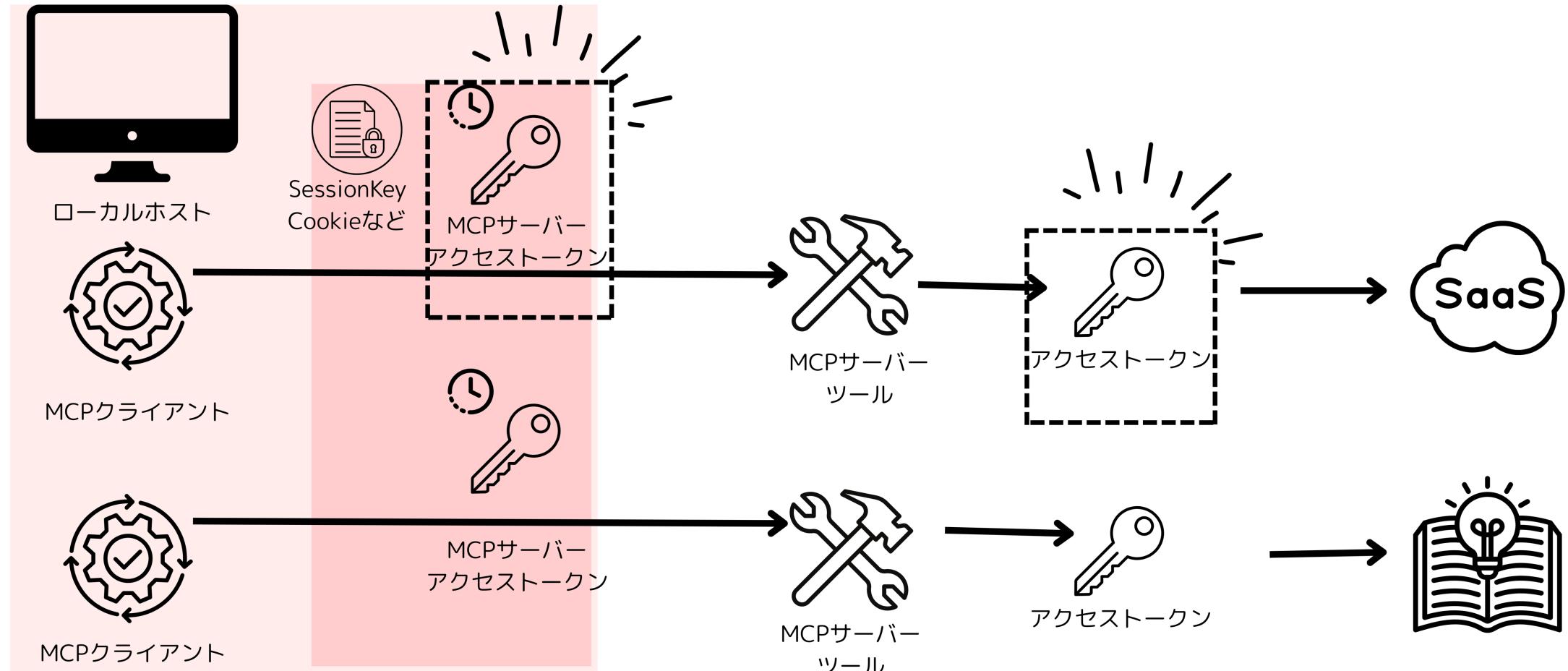
APIは静的トークン = 漏洩時の影響が大きすぎる

- 定期的なキーローテをユーザーに求める必要がある
 - 誰もやらない
- (外部の)MCPサーバー内にキーを抱え込みにくい
 - DBのID/PWとかとてもじゃないけど持てない
- SaaS権限とは別にリソースサーバーとしてのMCPサーバーの権限制御ができない
 - SharePointを読み込むけど、既存の記事は編集できない、など。

<https://github.com/modelcontextprotocol/modelcontextprotocol/discussions/234>

このあたりで議論されていました。

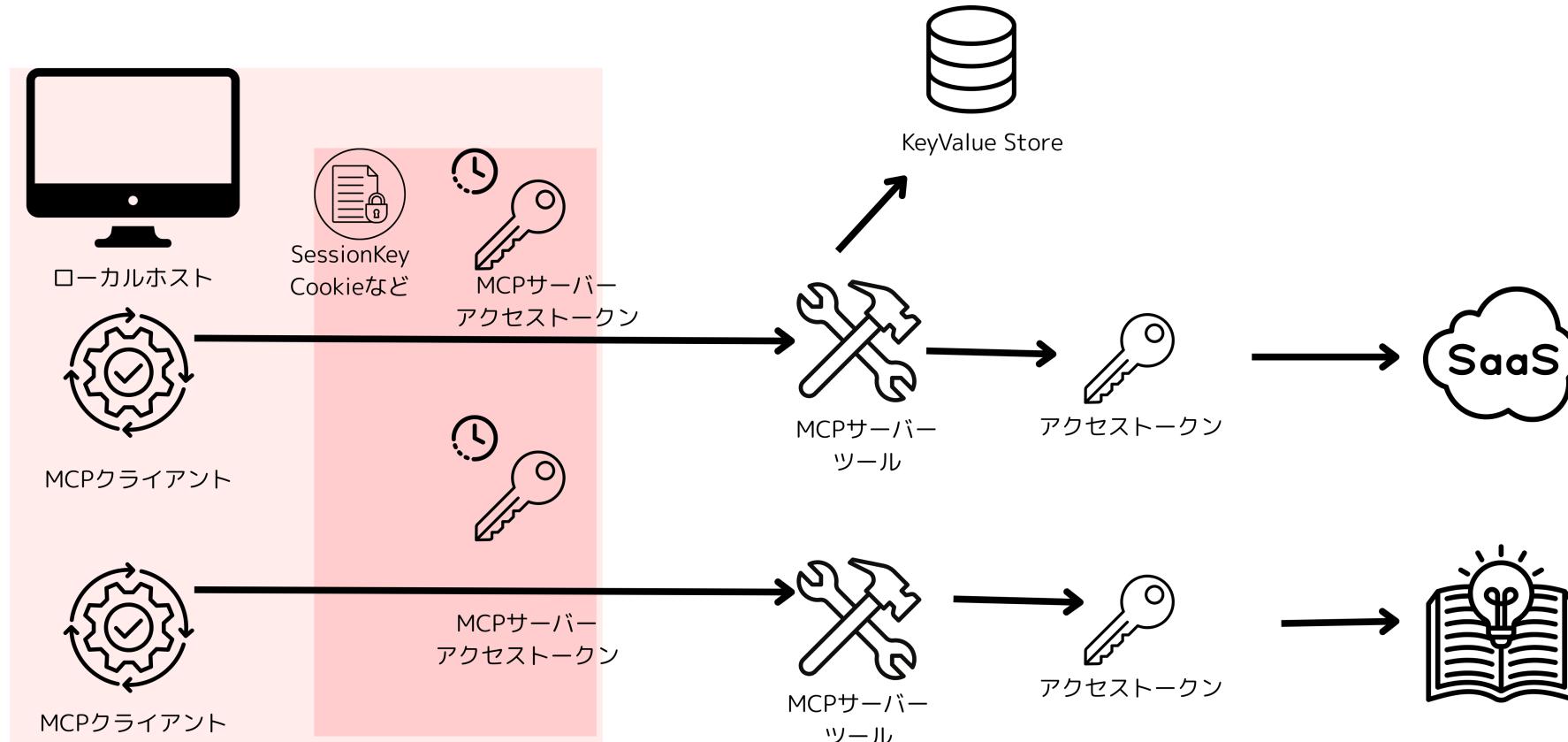
大きな特徴は、MCPクライアント vs サーバー間の認可と MCPサーバー vs 外部プロダクト間の認可の二つがあること



https://modelcontextprotocol.io/specification/draft/basic/security_best_practices#token-passthrough にて規定されます

MCPにおける認証・認可の重要性

なので、認証認可が完了すると、二つのアクセストークンが出来上がる状態。
加えて言うと、二つのアクセストークンを紐づける仕組みが必要になる。



よくある手法としてKeyValueでそれぞれの認証・認可のフローで必要な情報を保存しておく。
DBとしてTTLが設定できるとなおセッション感が出てよい。

MCP(OAuth2.1)を支える技術

PKCE(ピクシーと読む)を用いた認可フロー(MUST)

Proof Key for Code Exchange by OAuth Public Clients

- 認可コードを横取りされない仕組み
 - 認可リクエスト者とトークンリクエスト者が同じである証明
 - 乱数とそのハッシュをそれぞれで突合させる
- パブリックなクライアントでも使える
 - クライアントを証明する方法=>クライアントクレデンシャル
 - Webページやスマホには埋め込めない欠点
 - クライアントを証明する代わりの方法

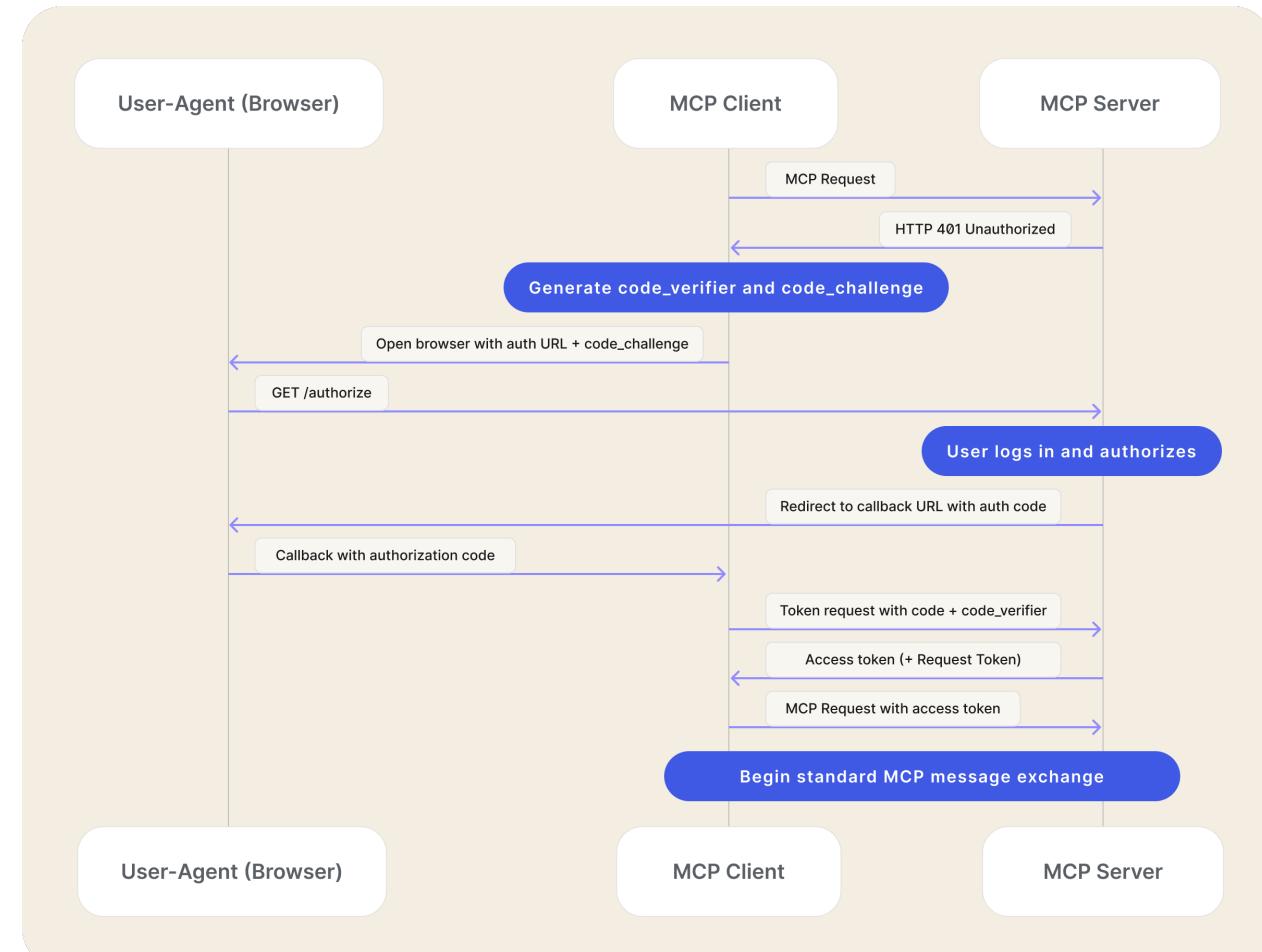


図: <https://auth0.com/blog/an-introduction-to-mcp-and-authorization/>

メタデータディスカバリー(Authorization Server Metadata)

MCPサーバーにいざアクセスしても、どこを叩いて認証・トークン取得すればいいかわからない。

なので、well-knownディレクトリとして、認証に関する情報をまとめておくエンドポイントを用意することが推奨される(ServerはSHOULD、ClientはMUST)

対応していない場合は、下記のデフォルトエンドポイントを暗黙的に利用する

■ デフォルトエンドポイント

- /authorize 認証エンドポイント
- /token トークンエンドポイント
- /register 動的クライアント登録エンドポイント

<https://datatracker.ietf.org/doc/html/rfc8414>

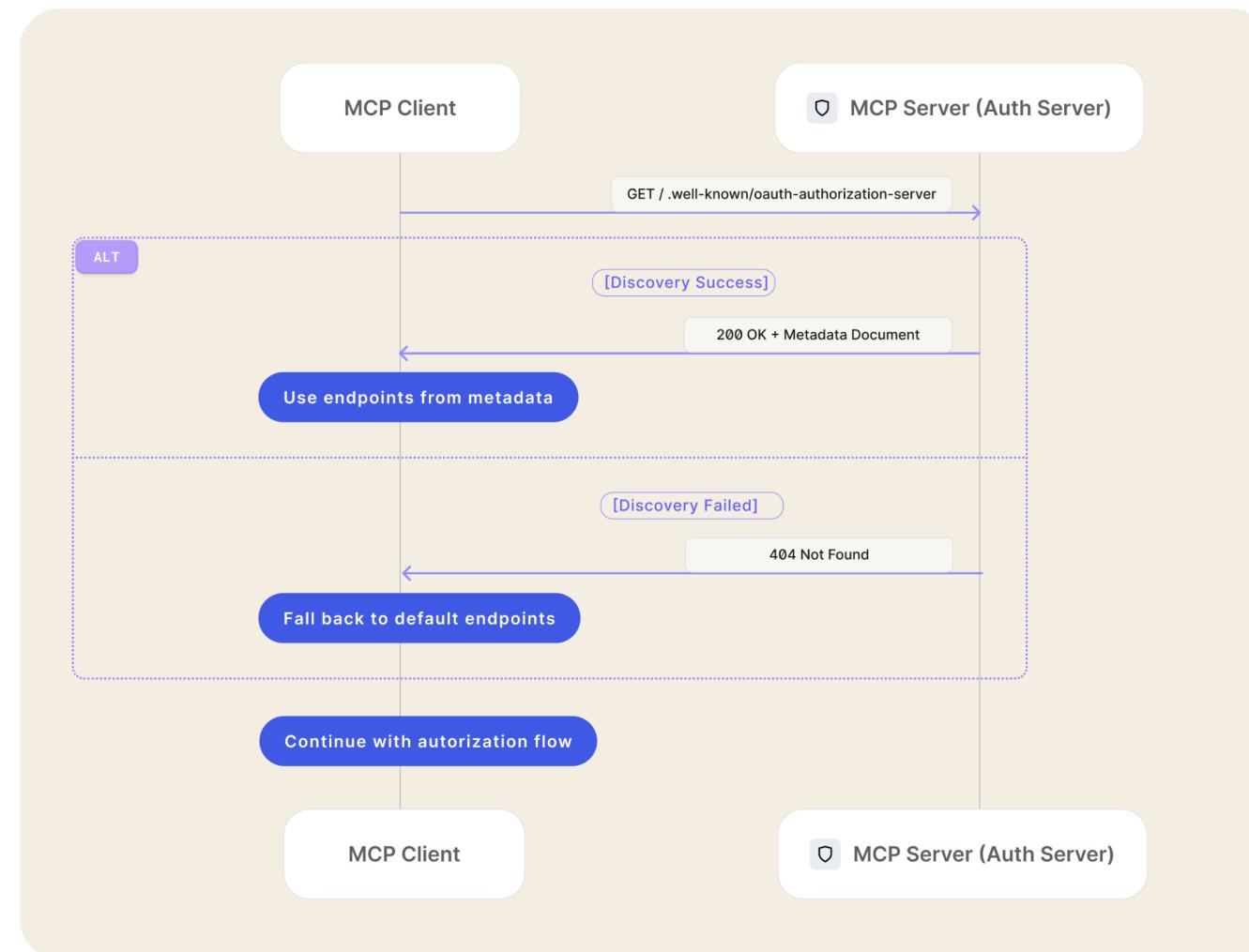


図: <https://auth0.com/blog/an-introduction-to-mcp-and-authorization/>

メタデータディスカバリー(Protected Resource Metadata - MUST)

細かいですが、メタデータディスカバリーには、

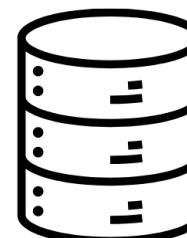
- 認可サーバー側の機能
- リソースサーバー側の機能

の2つが別々のRFCで規定されている

MCPサーバーはリソースサーバーであるため、
リソースサーバー側のメタデータディスカバリーの
実装はMUSTとなっている。



Authorization Server Metadata
認可サーバーが自身と対話するために
必要な情報を渡す



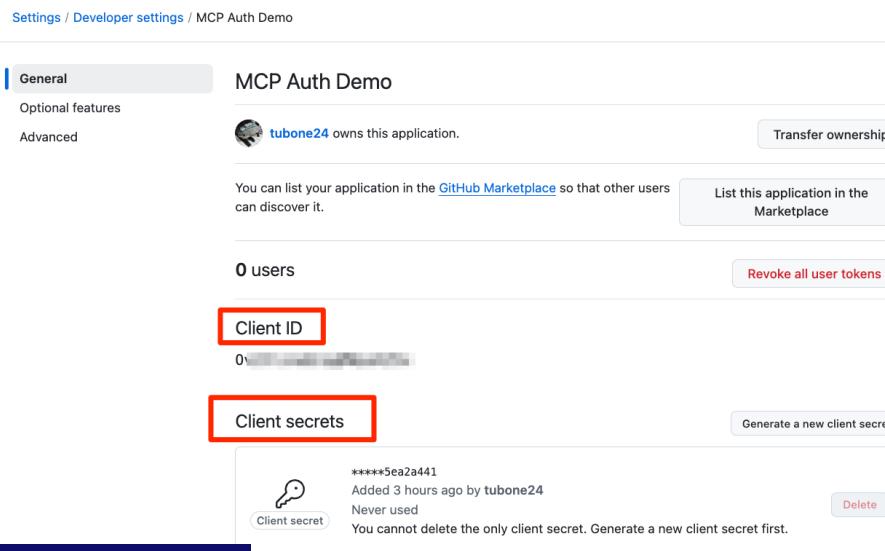
Protected Resource Metadata
リソースサーバーが自身の保護リソースに
アクセスするために必要な情報を渡す

<https://datatracker.ietf.org/doc/html/rfc9728>

動的クライアント登録(DCR - SHOULD)

OAuthのクライアントを設定したことがある人はわかるかも

- 事前にクライアントIDとシークレットを登録する必要あり
- MCPクライアントに毎度設定するのは大変
- リクエストに応じてクライアントIDを発行する仕組み



GitHubの例

<https://datatracker.ietf.org/doc/html/rfc7591>

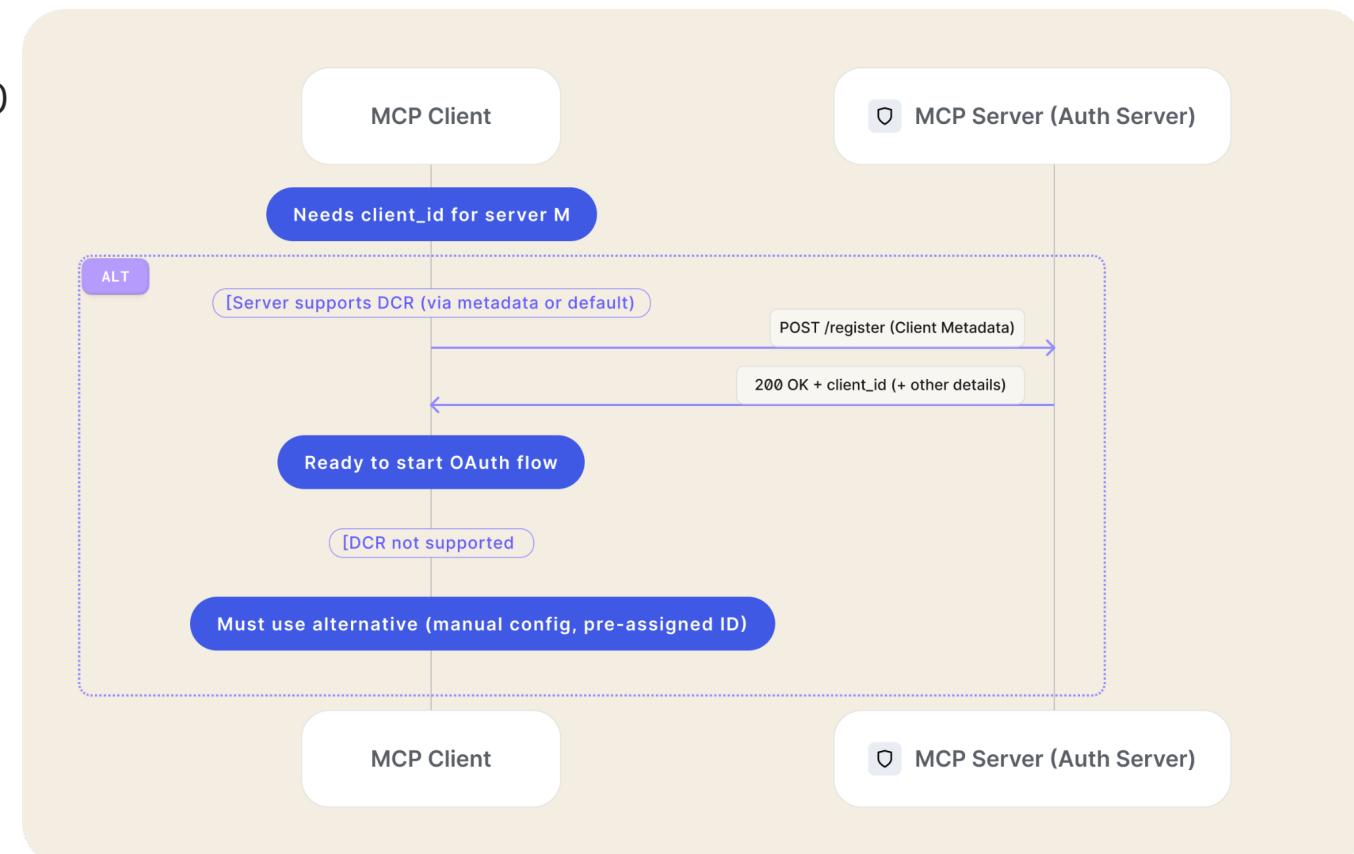


図: <https://auth0.com/blog/an-introduction-to-mcp-and-authorization/>

デモ

(自作のリモートMCPサーバー+Inspector)

今回のコードは次のレポジトリに保存してます
また、仕組みを作るのにCloudflare workersとKVを使いました

tubone24/remote-mcp-oauth-github

A Model Context Protocol (MCP) server for Claude.ai custom integrations, running on Cloudflare Workers with GitHub OAuth authentication.

1

Contributor

0

Issues

0

Stars

0

Forks



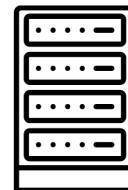
<https://github.com/tubone24/remote-mcp-oauth-github>

Streamable HTTPのエンドポイント(/mcp)にアクセスすると 認証を求められる

Streamable HTTPでアクセスします～
<https://mcp-github-oauth-demo.tubone24.workers.dev/mcp>



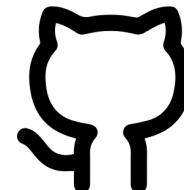
MCPホスト(クライアント)



リモートMCPサーバー
(Cloudflare Workers)



Auth store
(Cloudflare KV)



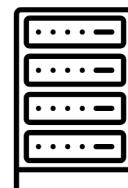
GitHub OAuth Server

ちゃんと認証してからにしてください～(401 Unauthorized)

```
{"jsonrpc":"2.0","error":{"code":-32000,"message":"Authentication required"}}
```



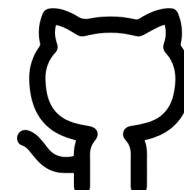
MCPホスト(クライアント)



リモートMCPサーバー
(Cloudflare Workers)



Auth store
(Cloudflare KV)



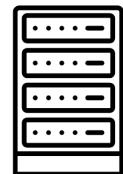
GitHub OAuth Server

認証に関するエンドポイントを一覧もらう これがメタデータディスカバリー

Wellknownにアクセスして認証のエンドポイント一覧もらいます～
/.well-known/oauth-authorization-server



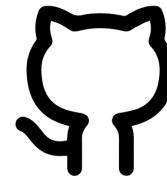
MCPホスト(クライアント)



リモートMCPサーバー
(Cloudflare Workers)



Auth store
(Cloudflare KV)



GitHub OAuth Server



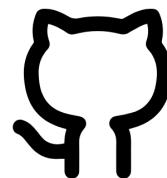
MCPホスト(クライアント)



リモートMCPサーバー
(Cloudflare Workers)



Auth store
(Cloudflare KV)



GitHub OAuth Server

クライアントを自動で登録して以降のOAuthクライアントとして利用 これがDCR

今後OAuthで認証するためにもクライアントとして登録したいです！
[/register](#)



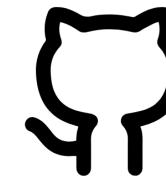
MCPホスト(クライアント)



リモートMCPサーバー
(Cloudflare Workers)



Auth store
(Cloudflare KV)



GitHub OAuth Server

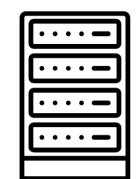
わかりました！あなたをクライアントxxxxとして登録しておきます。
スコープなども登録しておきました。よろしく！

保存

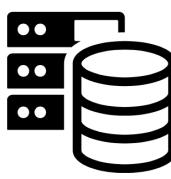
```
{  
  "client_id": "xxxx",  
  "client_secret": "yyyy"  
}
```



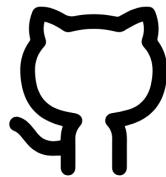
MCPホスト(クライアント)



リモートMCPサーバー
(Cloudflare Workers)



Auth store
(Cloudflare KV)



GitHub OAuth Server

Authorization URLを発行

DCRで作成されたclient_idに加え、code_challengeが設定されている
これがPKCE

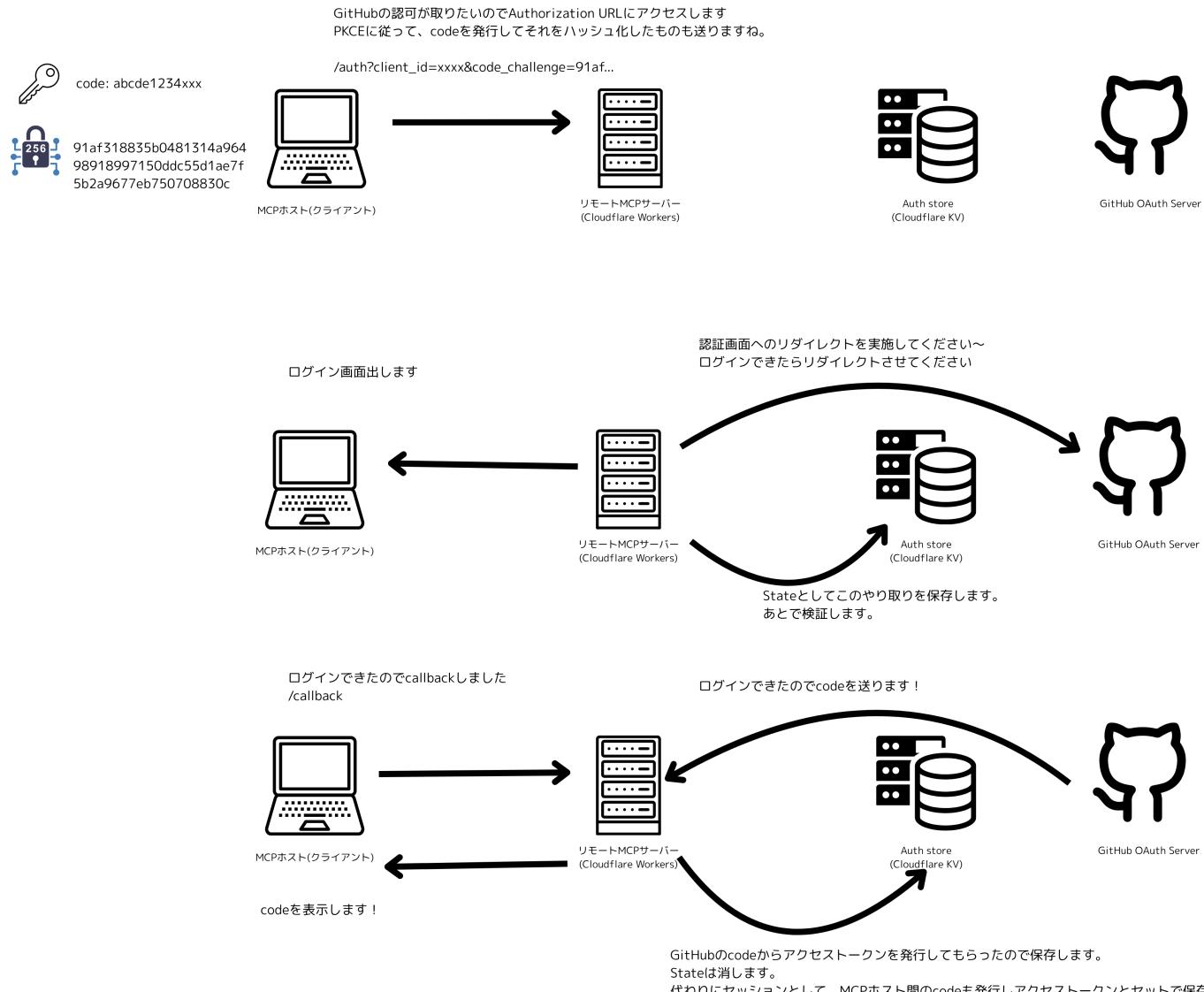
④ Preparing Authorization

Authorization URL:

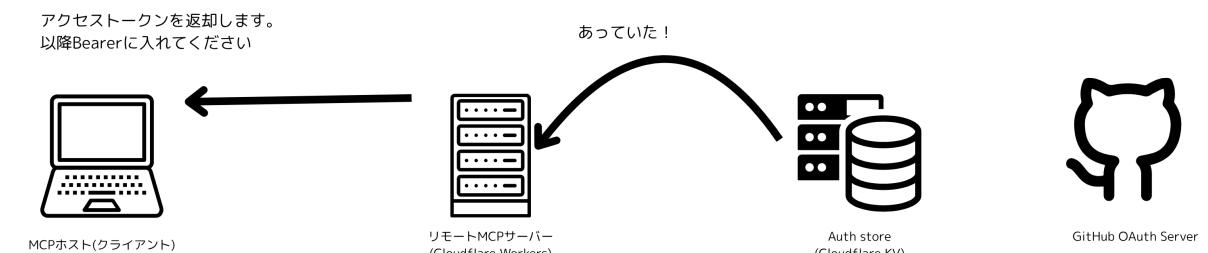
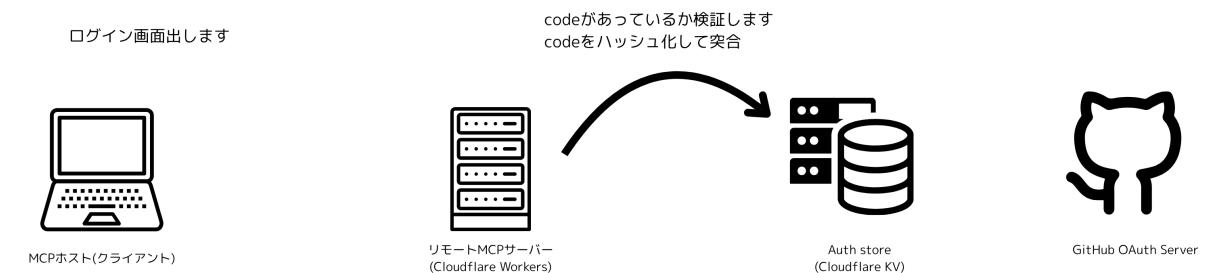
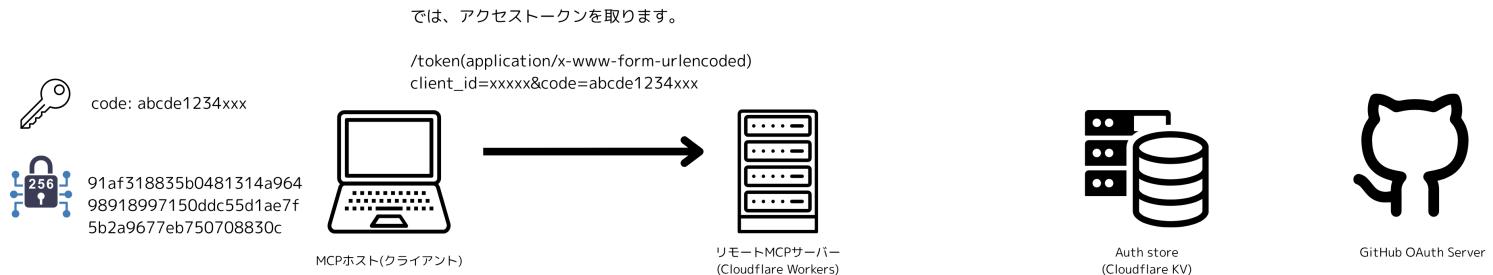
https://mcp-qithub-oauth-demo.tubone24.workers.dev/auth?response_type=code&client_id=cac4099c-41ef-40d2-a9c4-46ab690a93de&code_challenge=srTVEx0Uw18aP_DALPi6ufbjefzTlqlfaHqB2ugC-Ys&code_challenge_method=S256&redirect_uri=http%3A%2F%2F127.0.0.1%3A6274%2Foauth%2Fcallback%2Fdebug&scope=read%3Auser+user%3Aemail

Click the link to authorize in your browser. After authorization, you'll be redirected back to continue the flow.

フローにするとこんな感じですが、詳細は割愛



トークン取得処理でcodeの検証を実施 これがPKCE



閑話休題

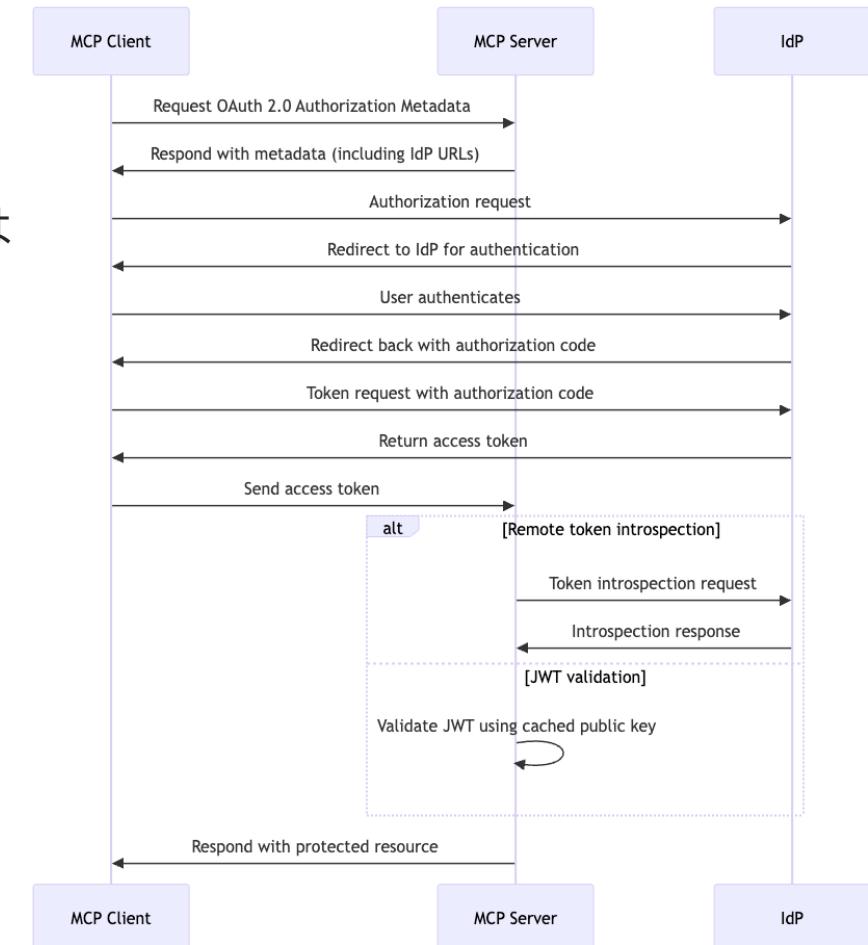
認可サーバーとリソースサーバーってくっつけていいの？

「ベストプラクティスじゃない」

が現状の規格だと、MCPサーバーがトークンの発行まで実施
ベストプラクティスに従えば、認可サーバーとリソースサーバーは
分けるべき。

よって下記のIssueなどにより良い方式が今なお、
議論がされているところです。

<https://github.com/modelcontextprotocol/modelcontextprotocol/issues/205>

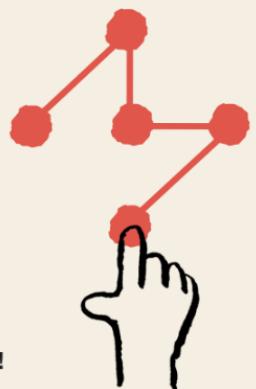


もっとMCPについて学びたい人は…

AIエージェント時代の標準規格
主要コンセプトをやさしく
図解

やさしい MCP 入門

モデル
コンテキスト
プロトコル



KDDI アジャイル開発センター株式会社
御田 稔／大坪 悠 [著]

- エンジニアでなくても規格がわかる！
- 自社ビジネスへの活かし方がわかる！
- 今後の AI トレンドへの展望がわかる！
- おすすめの MCP サーバーを厳選紹介！

秀和システム

話題のMCPにやさしく入門しよう

技術のしくみ & ビジネス展望がわかる！
図解たっぷりのコンパクトな本です。

7/1発売。アマゾンで予約受付中！

KDDIアジャイル開発センター
御田 稔（みのるん）、大坪 悠

KDDI Agile Development Center Corporation

22



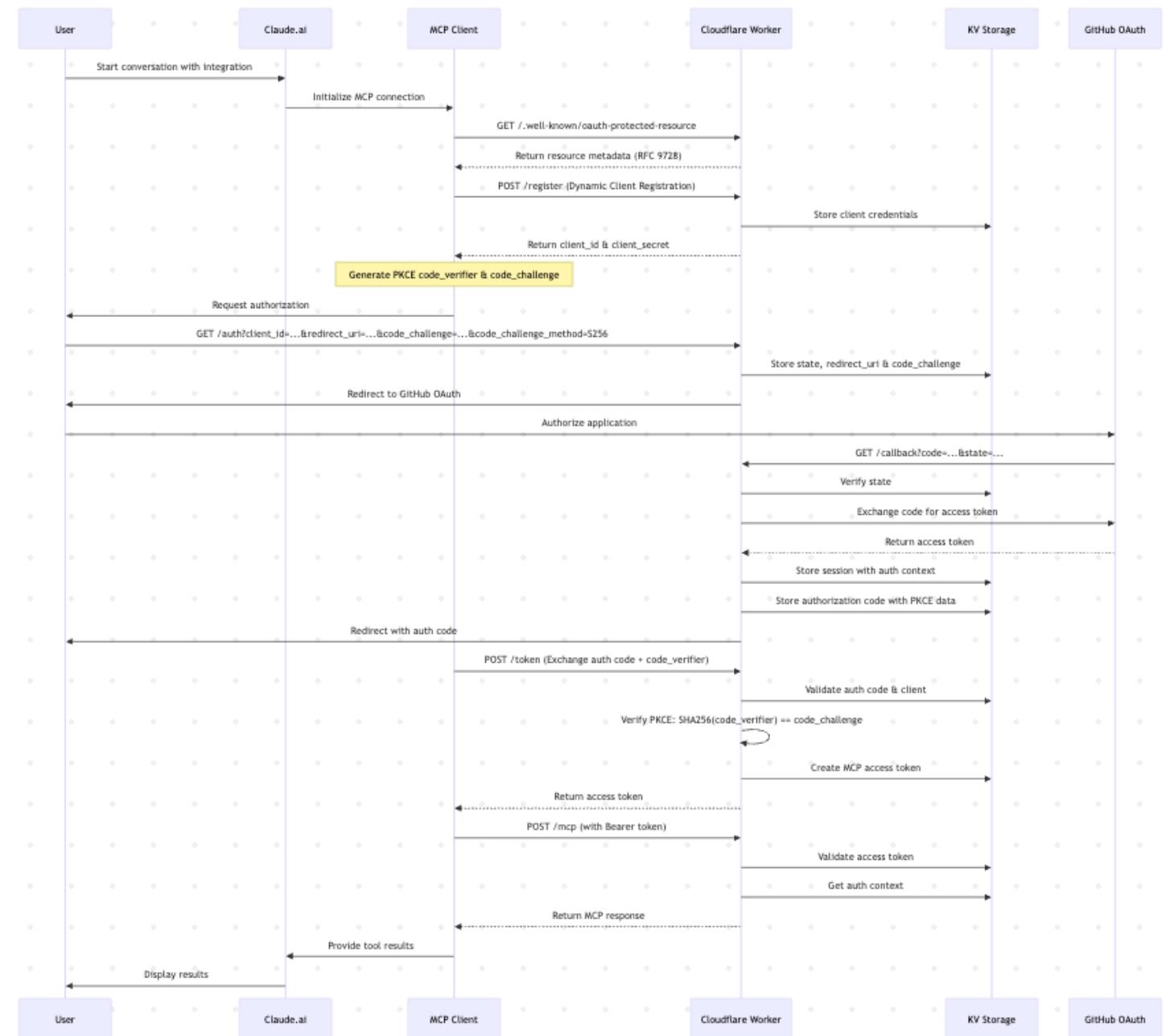
Be a Change Leader.

アジャイルに力を与え

共に成長し続ける社会を創る

Appendix

デモ - シーケンス



デモがうまく行かなかった用

MCP Inspector v0.14.0

Transport Type

Streamable HTTP

URL

h-demo.tubone24.workers.dev/mcp

> Authentication

Server Entry Servers File

> Configuration

Connect

Disconnected

System

?

?

?

Connect to an MCP server to start inspecting

Need to configure authentication? Open Auth Settings

History

No history yet

Server Notifications

No notifications yet

This screenshot shows the MCP Inspector v0.14.0 application window. On the left, there's a sidebar with transport type settings (Streamable HTTP), a URL input field containing 'h-demo.tubone24.workers.dev/mcp', and several navigation buttons: 'Authentication' (with a dropdown arrow), 'Server Entry', 'Servers File', 'Configuration' (with a dropdown arrow), and a large 'Connect' button. Below these is a status message 'Disconnected'. At the bottom of the sidebar are system-related buttons for 'System', a question mark, and other icons. The main area of the window has a message 'Connect to an MCP server to start inspecting' and a note about authentication. It also contains two empty sections: 'History' (with the message 'No history yet') and 'Server Notifications' (with the message 'No notifications yet').

デモがうまく行かなかった用

MCP Inspector v0.14.0

Transport Type

Streamable HTTP

URL

[https://mcp-github-oauth-demo.](https://mcp-github-oauth-demo)

> Authentication

Server Entry Servers File

> Configuration

Connect

Disconnected

System ⚙️ ⓘ ⚡ 🛡️

OAuth Flow Progress

Follow these steps to complete OAuth authentication with the server.

- Metadata Discovery
- Client Registration
- Preparing Authorization
- Request Authorization and acquire authorization code
- Token Request
- Authentication Complete

Continue

History

No history yet

Server Notifications

No notifications yet

デモがうまく行かなかった用

MCP Inspector v0.14.0

Transport Type

Streamable HTTP

URL

[https://mcp-github-oauth-demo.](https://mcp-github-oauth-demo)

> Authentication

Server Entry Servers File

> Configuration

Connect

Disconnected

System

OAuth Flow Progress

Follow these steps to complete OAuth authentication with the server.

- Metadata Discovery
- Client Registration
- Preparing Authorization
- Request Authorization and acquire authorization code
- Token Request
- Authentication Complete

Continue

History

No history yet

Server Notifications

No notifications yet

デモがうまく行かなかった用

MCP Inspector v0.14.0

Transport Type

Streamable HTTP

URL

<https://mcp-github-oauth-demo.tubone24.workers.dev>

> Authentication

Server Entry Servers File

> Configuration

Connect

● Disconnected

System ⚙️ ⓘ ⚡ ⚡

OAuth Flow Progress

Follow these steps to complete OAuth authentication with the server.

Metadata Discovery

OAuth Metadata Sources

Resource Metadata:

From <https://mcp-github-oauth-demo.tubone24.workers.dev/.well-known/oauth-protected-resource>

```
[{"id": "resource_id", "display_name": "GitHub OAuth Resource", "description": "A GitHub OAuth protected resource.", "scopes_supported": ["read:user", "user:email"], "bearer_methods_supported": ["header", "query"], "resource_documentation": "https://github.com/cloudflare/workers-oauth-provider", "resource_policy_uri": "https://mcp-github-oauth-demo.tubone24.workers.dev/terms", "resource_tos_uri": "https://mcp-github-oauth-demo.tubone24.workers.dev/terms", "mcp_version": "2024-11-05", "mcp_endpoints": {"sse": "https://mcp-github-oauth-demo.tubone24.workers.dev/sse", "http": "https://mcp-github-oauth-demo.tubone24.workers.dev/mcp"}]}
```

History

No history yet

Server Notifications

No notifications yet

デモがうまく行かなかった用

MCP Inspector v0.14.0

Transport Type

Streamable HTTP

URL

https://mcp-github-oauth-demo.

> Authentication

Server Entry Servers File

> Configuration

> Connect

● Disconnected

System ⚙️ ⓘ 🔍 🛡️

Metadata Discovery

▶ OAuth Metadata Sources

Client Registration

Registered Client Information

```
{  
    "redirect_uris": [  
        "http://127.0.0.1:6274/oauth/callback/debug"  
    ],  
    "grant_types": [  
        "authorization_code",  
        "refresh_token"  
    ],  
    "response_types": [  
        "code"  
    ],  
    "client_name": "MCP Inspector",  
    "scope": "read:user user:email",  
    "client_id": "1[REDACTED]",  
    "client_secret": "1[REDACTED]",  
    "client_id_issued_at": 1749733495,  
    "client_secret_expires_at": 0  
}
```

Preparing Authorization

History

No history yet

Server Notifications

No notifications yet

デモがうまく行かなかった用

Please copy this authorization code and return to the Auth Debugger:

ea[REDACTED]

Close this tab and paste the code in the OAuth flow to complete authentication.

デモがうまく行かなかった用

The screenshot shows the MCP Inspector v0.14.0 interface. On the left, the main panel displays configuration details: Transport Type set to Streamable HTTP, URL set to <https://mcp-github-oauth-demo>, and a status message indicating Disconnected. Below these are buttons for Authentication, Server Entry, Servers File, Configuration, and Connect. The Connect button is highlighted with a dark background. At the bottom, there are system navigation icons for System, Help, Refresh, and Logout.

The right side of the screen shows the OAuth token exchange process:

- Authorization Code**: A placeholder field for the authorization code, currently containing redacted text.
- Token Request**: A section where the user has completed the token request. It includes the Token Endpoint: <https://mcp-github-oauth-demo.tubone24.workers.dev/token>.
- Authentication Complete**: A section indicating successful authentication. It displays the Access Tokens received from the server:

```
{  
  "access_token": "redacted",  
  "token_type": "Bearer",  
  "expires_in": 3600,  
  "scope": "read:user user:email"  
}
```

Below this, sections for History and Server Notifications are present, both stating "No history yet" and "No notifications yet".

デモがうまく行かなかった用

The screenshot shows the MCP Inspector v0.14.0 application interface. The left sidebar contains transport type settings (Streamable HTTP), a URL input (https://mcp-github-oauth-demo.), and navigation links for Authentication, Server Entry, Servers File, Configuration, Reconnect, and Disconnect. A green 'Connected' status indicator is present. The main area has a 'Tools' tab selected in the top navigation bar, which includes 'Resources', 'Prompts', 'Tools', 'Ping', 'Sampling', 'Roots', and 'Auth'. The 'Tools' section lists 'get_user_info', 'calculate', and 'get_github_repos'. The 'get_user_info' tool is selected, showing its description ('認証されたユーザーの情報を取得'), a 'Run Tool' button, and a 'Tool Result: Success' section displaying user information: "ユーザー情報: - ユーザー名: tubone24 - 名前: tubone(Yu Otsubo) - メール: 未設定". The bottom sections show 'History' with three entries: '7. tools/call', '6. tools/call', and '5. tools/call', and 'Server Notifications' with the message 'No notifications yet'.

MCP Inspector v0.14.0

Transport Type
Streamable HTTP

URL
https://mcp-github-oauth-demo.

> Authentication

Server Entry Servers File

> Configuration

Reconnect Disconnect

Connected

System

Resources Prompts Tools Ping Sampling Roots Auth

Tools

List Tools

Clear

get_user_info
認証されたユーザーの情報を取得

calculate
数値計算を実行

get_github_repos
GitHubリポジトリ一覧を取得

get_user_info

認証されたユーザーの情報を取得

Run Tool

Tool Result: Success

"ユーザー情報:
- ユーザー名: tubone24
- 名前: tubone(Yu Otsubo)
- メール: 未設定"

History

7. tools/call

6. tools/call

5. tools/call

Server Notifications

No notifications yet