

2026年9月16日

APS MCP ワークショップ

Autodesk Developer Network

オートデスク株式会社



アジェンダ

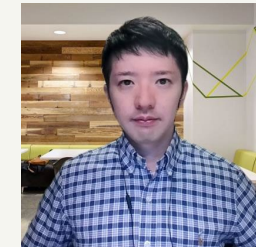
- 13:00～13:20 **概要説明**
- 13:20～14:20 **SSA の用意**
- **<休憩>**
- 14:30～15:30 **MCP サーバーの作成**
- **<休憩>**
- 15:40～16:40 **MCP ツールの作成**
- 16:40～17:00 **MCP クライアントとの統合**



Toshiaki Isezaki
伊勢崎 俊明

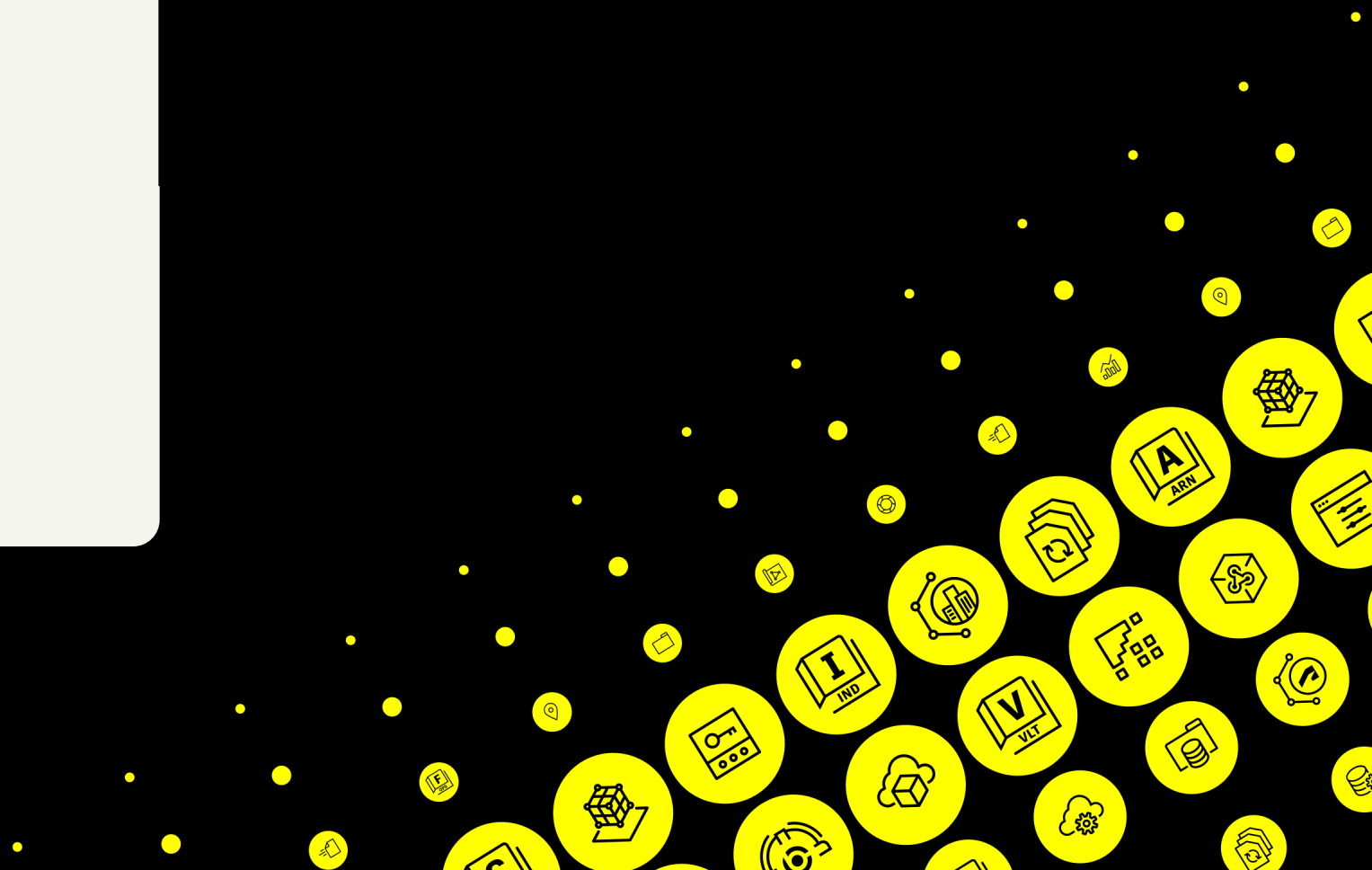


Takehiro Kato
加藤 文博



Ryuji Ogasawara
小笠原 龍二

概要説明



マテリアル ダウンロードについて

- 本プレゼンテーション資料 PDF ダウンロード

<https://blog.autodesk.io/wp-content/uploads/2026/06/MCP-Workshop.pdf>



対象者と環境

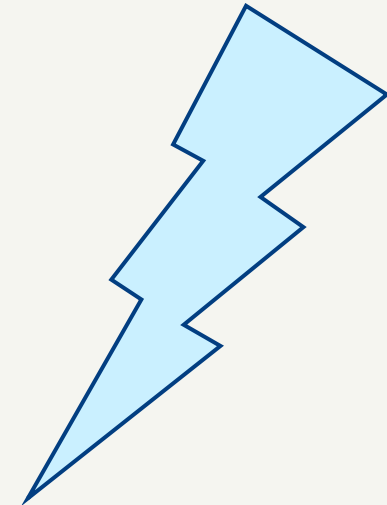
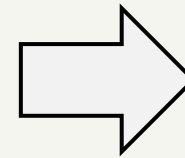
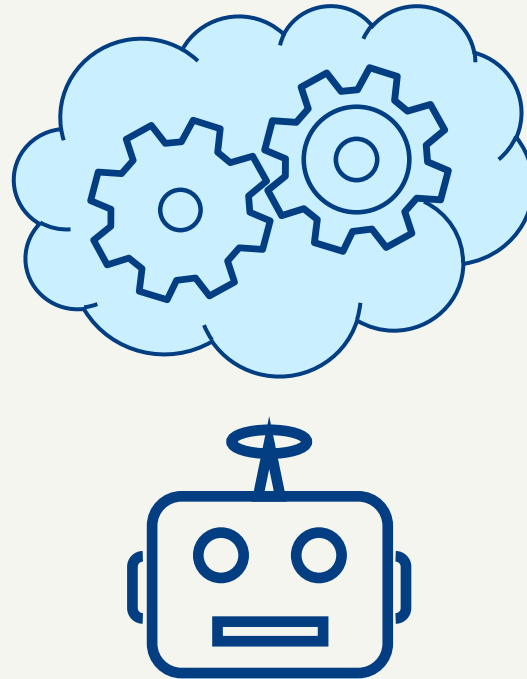
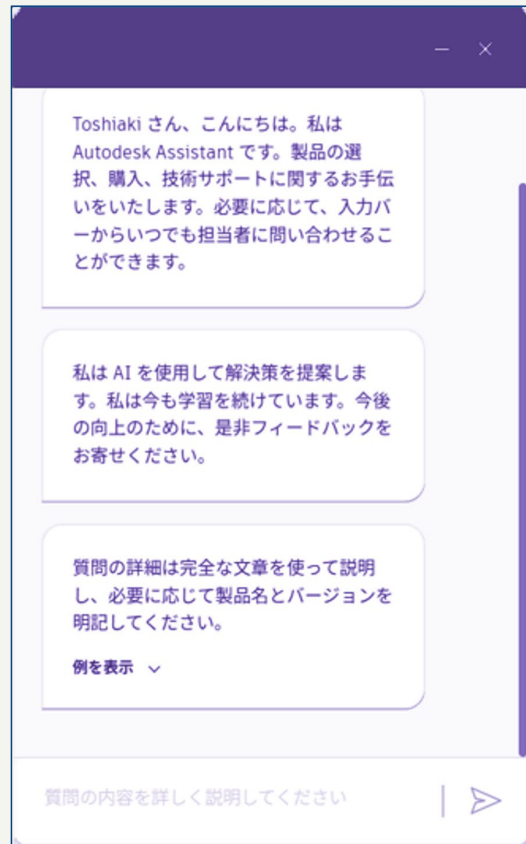
- 対象者：
 - APS 開発経験または知識をお持ちの方
 - AI エージェントと MCP サーバーの可能性を把握したい方
 - APS でオートデスク SaaS プロジェクト統合をされる方
 - Forma Data Management – 旧 Autodesk Docs（建設業）
 - Fusion Web クライアント - 旧 Fusion Team（製造業）
- 環境
 - Windows PC
 - VS Code
 - Node.js

ワークショップ内容と事前準備

- チュートリアル コンテンツ
 - [APS MCP Server: Tutorial](#)
 - <https://autodesk-platform-services.github.io/aps-mcp-server-nodejs/#/>
- VS Code でローカル MCP サーバーを実装、テスト
 - 使用する APS API : Secure Service Account API、Data Management API
- VS Code と Node.js のインストール
 - 参考 : [Autodesk Developer Blog : APS の開発環境](#)
- デベロッパーハブの用意

LLM から AI エージェントへ進化

- フロント エンドは LLM（大規模言語モデル）
- コンテキストを理解してワークフローを組立て・実行



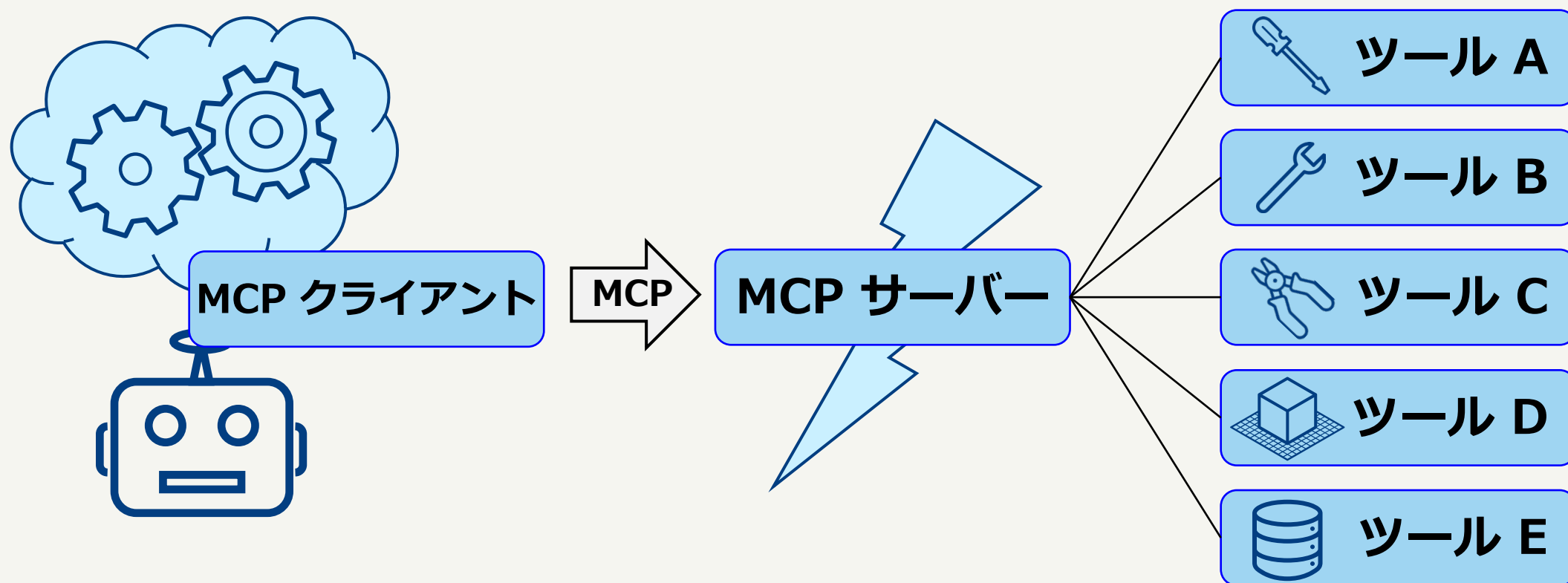
標準化に寄与する MCP の登場

Model Context Protocol

AI エージェントが利用するツールを持つ
実装（サーバー）との橋渡し部分の仕様

AI エージェントと MCP サーバー

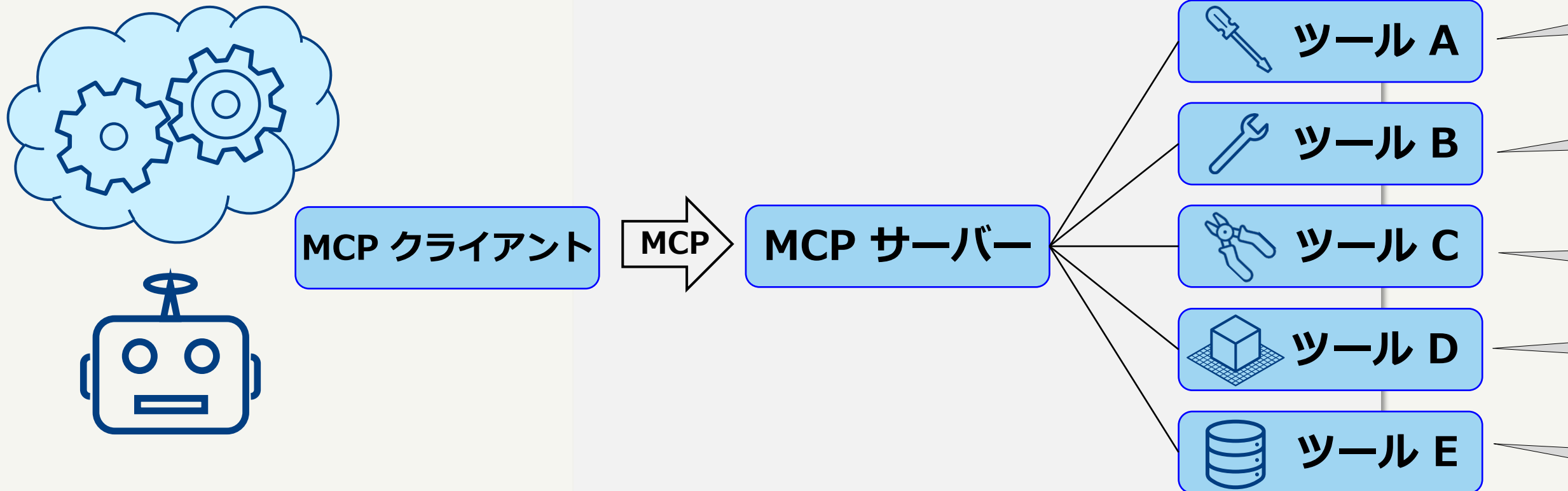
- MCP サーバーは複数のツールを持つ



MCP サーバーとツール

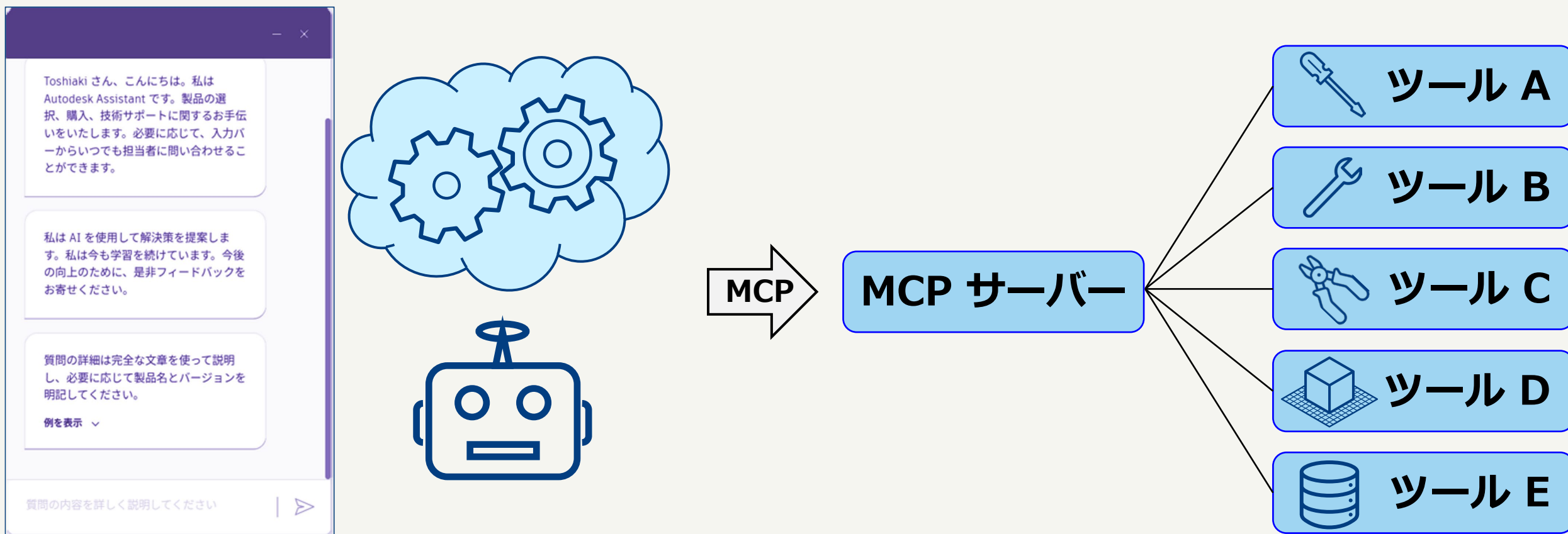
- ツールは API を呼び出して機能呼び出す

通常、どの API を使っているか非公開



適切な MCP サーバーとツールを呼び出す

- フロント エンドは LLM（大規模言語モデル）
- コンテキスト を理解してワークフローを組立て・実行



本ワークショップのゴール



VSCode で GitHub Copilot を使用

1. アクセス可能なハブの取得

2. プロジェクトとファイル一覧を取得

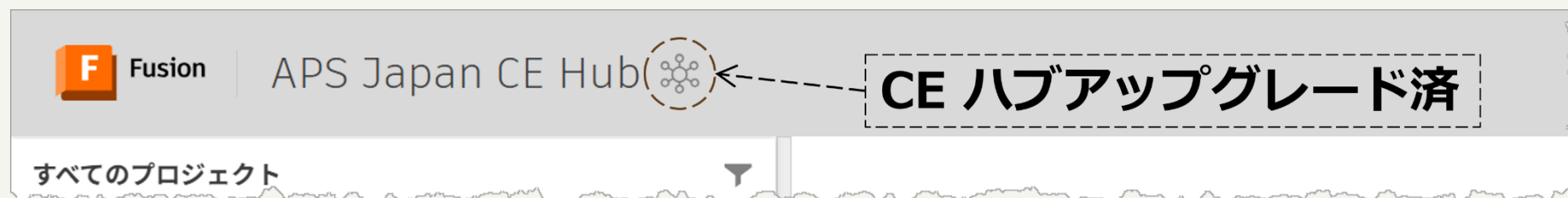
※アクセス対象：

- ✓ Forma Data Management
(旧 Autodesk Docs)
- ✓ Fusion Contributor
(旧 Fusion Team)



Data Model API (GraphQL) は意図的に未使用

- Forma と Fusion へのアクセスを同一コードで達成する為
- AEC Data Model API と Manufacturing Data Model API
 - 本来は粒状データアクセスの利点のため使用を推奨しています
 - ただし、エンドポイントと GraphQL 構文が異なってしまう
- Manufacturing Data Model API v3 利用の要件
 - Fusion が Collaborative Editing (共同編集) ハブである必要あり
 - 旧ハブは共同編集ハブにアップグレードする必要がある
 - アップグレード作業はオートデスクが順次実行 (現時点で未適用ハブあり)





✱ こんにちは、伊勢崎さん

アクセス可能なオートデスク プロジェクトとプロジェクト下のファイルをすべてリストして

+

Sonnet 5 中 ▾



MCP 仕様について

- **Model Context Protocol**
 - Anthropic 社提唱のテクノロジー
 - AI エージェントと外部システム間のオープンソース プロトコル
 - 仕様
 - <https://modelcontextprotocol.io/specification/2026-07-28>
 - GitHub リポジトリ
 - <https://github.com/modelcontextprotocol>
 - MCP SDK
- AI エージェント、MCP 仕様は進化（変化）していきます



プロジェクト データ アクセスに必要な認証フロー

- Data Management API : API Basics

API Basics

There are two key data access paradigms that make up the Data Management API.

- Accessing data from Autodesk SaaS applications using any of the Data Management services.
 - For BIM 360 Team, Fusion Team, and A360 Personal, end users need to provide a 3-legged OAuth token for your app to access the data.
 - For BIM 360 Docs, an account administrator needs to add an integration with your app in BIM 360 Enterprise. You can access data using either 2-legged or 3-legged authentication.
- Managing and storing files from your app on Autodesk Platform Services, independent of any Autodesk SaaS application. You need to use the Object Storage service (OSS).

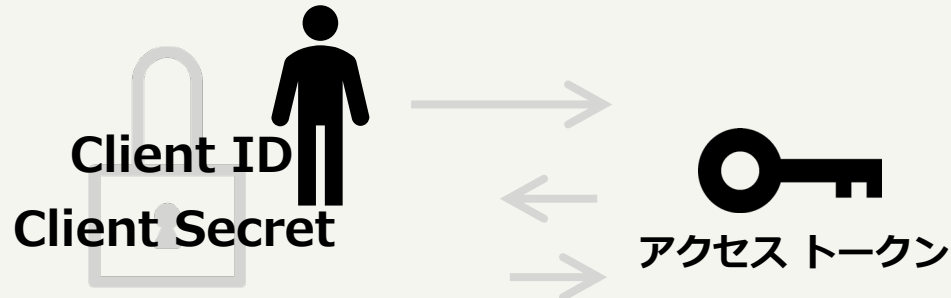
To navigate and access BIM 360 Team, Fusion Team, BIM 360 Docs, A360 Personal, and OSS data, you need to be familiar with the following terminology:

破線枠内の説明を現製品名に置き換えた解釈

- Fusion – Contributor (Fusion) にアクセスする場合、エンドユーザーがデータにアクセスするには、APS アプリに 3-legged 認証フローで得たアクセス トークンを提供する必要があります。
- Forma Data Management (Forma) にアクセスする場合、Hub Admin が Forma で「カスタム統合」を使ってアプリを追加する必要があります。APS アプリは、2-legged 認証フローまたは 3-legged 認証フローのいずれかで得たアクセス トークンを使用してデータにアクセスできます

もともと APS で提供している認証フロータイプ

- 3-legged 認証 (3LO)
 - ユーザー指向のアプリ連携向け
 - ユーザー コンテキスト
 - サインイン操作でアクセス トークン を取得 (初回)
 - リフレッシュ トークンで1時間毎にアクセス トークンを更新
- 2-legged 認証 (2LO)
 - アプリ専用
または サーバー間通信向け
 - アプリケーションコンテキスト
 - 任意タイミングでアクセス トークン を取得可能
 - 「なりすまし」機能は限定的



動作特性の違いによる認証フローの使い分け

- 3-legged 認証 (3LO)

- ユーザーにアクセス許可を得てデータ アクセス
- アクセス許可したユーザーの権限が継承される
- Forma アクセスには「カスタム統合」が必須
- 推奨される方法



- 2-legged 認証 (2LO)

- 分析のために複数プロジェクトを横断的にアクセス
- 「カスタム統合」を除き**アクセスをコントロールする手立てがない**

例) アプリが 2-legged 認証フローでアップロード

- アップロードしたユーザーは誰？

2-legged トークン 使用時に
x-user-id リクエストヘッダー未指定時

フォルダ		パッケージ				削除された項	
プロジェクト フ...							
A Project							
名前 ↑		最終更新日	更新者	バージョンの追加者		レビ	
☐	result.pdf	2024年11月18日 17:31	Forma システム	Forma システム			
☐	result2.pdf	2024年11月25日 16:12	Toshiaki Isezaki Developer Advocacy ...	Toshiaki Isezaki Developer Advocacy ...			

3-legged トークン 使用時、または
2-legged トークン 使用時に
x-user-id リクエストヘッダー指定時

認証別のアクセスまとめ と 過剰アクセスの懸念

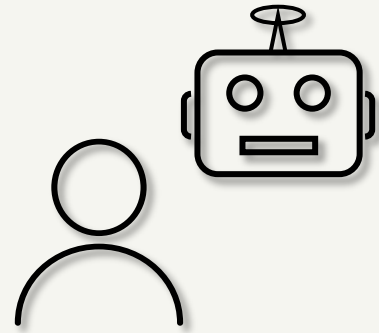
- プロジェクト データのオーナーは「ユーザー」
- 原則、**3-legged 認証フローの利用を推奨**

認証フロー	Forma	Fusion
2-legged 認証	○	×
2-legged 認証 x-user-id	○	×
3-legged 認証	○	○

- x-user-id は [GET OIDC User Info](#) sub フィールド値
- 2-legged 認証フローはアプリケーションコンテキスト時のみ
 - **アクセス制限無く全プロジェクトにアクセス出来てしまう**

過剰アクセスの懸念に対する「解」

- **Secure Service Account (SSA)**
 - アプリ アクセス用のロボットアカウントを用意・利用
 - 通常アカウントのようにプロジェクト側でアクセス制御



The image shows two overlapping screenshots from Autodesk software. The background screenshot is the Autodesk Forma interface, displaying a 'メンバー' (Members) list. A red box highlights the 'RA Robot Account' entry. The foreground screenshot is the Autodesk Fusion 'MyProject' interface. It shows a list of projects on the left, with 'MyProject' highlighted by a red box. On the right, the 'メンバーと権限(5)' (Members and Permissions) tab is active, showing a table of users and their permissions. A red box highlights the 'RA Robot Account' entry in this table, which has a permission level of 'Read Only'.

Autodesk Forma Members List:

名前	電子メール
Mamoru Miyakoji...	
RA Robot Account	robot_accoun...
Toshiaki Isezaki	

Autodesk Fusion MyProject Projects List:

- Demo Project
- My Datas
- MyProject**
- 0802 Case Practice Final - Tokyo model
- 0824 Case Practice Final - Osaka model
- EMBER_Assembly.f3d
- GearClock3_Metal
- GearClock3_Wood.f3d
- Stapler

Autodesk Fusion MyProject Members and Permissions:

名前	権限レベル
Admins (管理者グループ)	Full Control
RA Robot Account	Read Only
Toshiaki Isezaki	Full Control



SSA の利点と欠点

- ✓ パスワードの盗難や対話的な使用が不可能
- ✓ オートデスクのアプリからのみ使用可能
- ✓ API 経由でのみ使用可能
- ✓ 1年間未使用の場合、自動的に失効

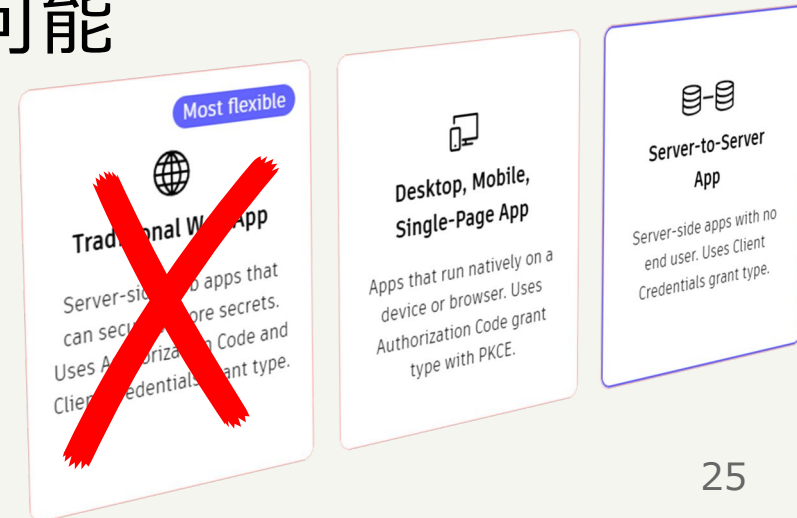


✗ “Server-to-Server App” タイプでのみ使用可能

- 従来の “Traditional Web App” タイプは適用不可

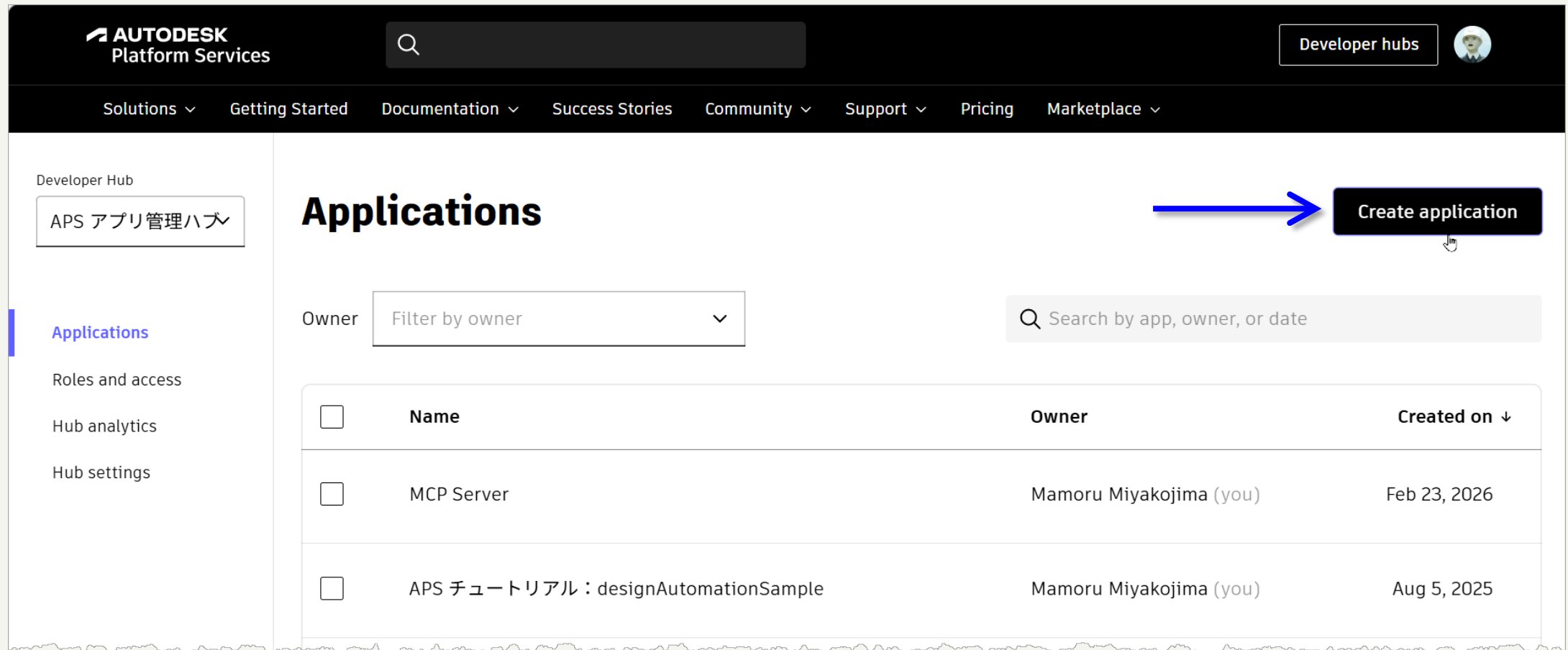
✗ ロボットアカウントの権限設定が必要

✗ App Store 配布シナリオへの対応に難



SSA の手動セットアップ

① APS アプリ作成（Client ID、Secret の取得）



The screenshot shows the Autodesk Developer Hub interface. The top navigation bar includes the Autodesk logo, a search bar, and links for 'Developer hubs' and a user profile. Below this is a secondary navigation bar with links for 'Solutions', 'Getting Started', 'Documentation', 'Success Stories', 'Community', 'Support', 'Pricing', and 'Marketplace'. The main content area is titled 'Applications' and features a sidebar on the left with links for 'Developer Hub', 'APS アプリ管理ハブ', 'Applications' (highlighted), 'Roles and access', 'Hub analytics', and 'Hub settings'. The main area has a 'Create application' button in the top right, pointed to by a blue arrow. Below the button is a search bar labeled 'Search by app, owner, or date'. A table lists existing applications with columns for 'Name', 'Owner', and 'Created on'.

☐	Name	Owner	Created on ↓
☐	MCP Server	Mamoru Miyakojima (you)	Feb 23, 2026
☐	APS チュートリアル：designAutomationSample	Mamoru Miyakojima (you)	Aug 5, 2025

※ アプリ作成はデベロッパーハブからのみ可能

SSA の手動セットアップ ～アプリ名入力とタイプ指定

② Create Application

aps-mcp-server-nodejs (任意)

Server-to-Server App

※ Forma 統合には「[カスタム統合](#)」が必要

ご注意：Forma 統合時の「カスタム統合」

カスタム統合機能を追加

×

ハブ ID

統合から API にアクセスする場合は、ハブ ID が必要です。続行する前に、ハブ ID を確認してください。ハブ ID は、後で次から参照することもできます [ハブ設定](#)

Autodesk Platform Services クライアント ID *

?

Autodesk Platform Services クライアント ID

カスタム統合名 *

カスタム統合名

説明

説明

0/255

キャンセル

次へ

**SSA の利用時にも
APS で Forma に
アクセスするアプリには
「カスタム統合」は必要です
(Client ID の
Forma への事前登録)**

SSA の手動セットアップ ～ SSA Manager ログイン

③ Access SSA Manager (<https://ssa-manager.autodesk.io>)

The screenshot shows the Autodesk SSA Manager web interface. At the top, there is a header bar with the Autodesk logo and the text "Sample SSA Management Tool". On the right side of the header, there are input fields for "Client Id" and "Client Secret", a "Log In" button, and a "GitHub" link. Below the header, the interface is divided into several sections: "Accounts", "Account Details", "Keys", "Key Details", "Private Key", and "Access Token". The "Accounts" section has a list of accounts and buttons for "Delete Selected Account", "Create Account With Name:", "First Name", "Last Name", and "Enable/Disable Account". The "Keys" section has a list of keys and buttons for "Delete Selected Key", "Create Key", and "Enable/Disable Key". The "Private Key" section has a "Load From File" button. The "Access Token" section has a "Create Token With Scopes:" dropdown, a text input field, and buttons for "Copy Token" and "Open On jwt.io". Annotations with blue arrows point from the "Client Id" and "Client Secret" input fields to the "Log In" button, with the text "②の Client ID" and "②の Client Secret" respectively. The text "Log in" is also present near the "Log In" button.

AUTODESK
Platform Services Sample SSA Management Tool

Client Id Client Secret Log In GitHub

Accounts

Account Details

②の Client ID

②の Client Secret

Log in

Keys

Key Details

Private Key

Access Token

Delete Selected Account Create Account With Name: First Name Last Name Enable/Disable Account

Delete Selected Key Create Key Enable/Disable Key

Load From File

Create Token With Scopes: data.read data.create data.\ Copy Token Open On jwt.io

SSA の手動セットアップ ～ アカウント作成

④ Create a New Service Account

AUTODESK
Platform Services Sample SSA Management Tool

W5GgwLdarlbq46ZPfUj0nd Log In GitHub

Accounts

Delete Selected Account

Account Details

Keys

Key Details

Private Key

Access Token

Create Account with Name

First Name
Robot

Last Name
Account

SSA の手動セットアップ ～ アカウント詳細の確認

⑤ Note Your Service Account Details

Accounts

Robot_Account@W5GgwLdarlbq46ZPfUj0ndGFcV71tRmqIR13bYACuqhAIE42.adsserviceaccount.com

選択

Delete Selected Account Create Account With Name: Robot Account Enable/Disable Account

Keys

Delete Selected Key Create Key Enable/Disable Key

Private Key

Load From File

Account Details

serviceAccountId と email を確認

Key Details

Access Token

Create Token With Scopes: data:read data:create data:\ Copy Token Open On jwt.io

SSA の手動セットアップ ~ プライベートキーの作成

⑥ Create a New Private Key



Accounts

Robot_Account@W5GgwLdarlbq46ZPfUj0ndGFcV71tRmqIR13bYACuqhAIE42.adsserviceaccount.com

選択

Delete Selected Account Create Account With Name: Robot Account Enable/Disable Account

Account Details

```
{
  "serviceAccountId": "QT326T3UKYA4K9C4",
  "email": "Robot_Account@W5GgwLdarlbq46ZPfUj0ndGFcV71tRmqIR13bYACuqhAIE42.adsserviceaccount.com",
  "createdBy": "W5GgwLdarlbq46ZPfUj0ndGFcV71tRmqIR13bYACuqhAIE42.adsserviceaccount.com",
  "status": "ENABLED",
  "createdAt": "2026-05-21 08:36:10 +0000 UTC",
  "accessedAt": "2026-05-21 08:36:10 +0000 UTC",
  "expiresAt": "2027-05-21 08:36:10 +0000 UTC"
}
```

Keys

Delete Selected Key Create Key Enable/Disable Key

Private Key

Create Key

Key Details

Access Token

Create Token With Scopes: data:read data:create data:\ Copy Token Open On jwt.io

最近のダウンロード履歴

6e0585bb-5e56-4765-9b91-ecfe8edeb602.pem

1,675 B • 4 分前

すべてのダウンロード履歴

.pem ファイルが自動ダウンロードされる

SSA の手動セットアップ ~ プライベートキー詳細の確認

⑦ Note Your Key Details

AUTODESK
Platform Services

Sample SSA Management Tool

iscflKR6pCq0xLovje7tozJmY5a960SgyryqzG0AsOPoZXjg.....Log InGitHub

Accounts

Robot_Account@iscflKR6pCq0xLovje7tozJmY5a960SgyryqzG0AsOPoZXjg.adskserviceaccount.com

Delete Selected AccountCreate Account With Name:TESTEnable/Disable Account

Keys

08cd8b2f-4755-4e5d-bcc4-ea3ec638b117

Delete Selected KeyCreate KeyEnable/Disable Key

Private Key

Load From File

Account Details

{
 "serviceAccountId": "LMG204C3L8Y9HSD3",
 "email": "Robot_Account@iscflKR6pCq0xLovje7tozJmY5a960SgyryqzG0AsOPoZXjg.adskserviceaccount.com",
 "createdBy": "iscflKR6pCq0xLovje7tozJmY5a960SgyryqzG0AsOPoZXjg",
 "status": "ENABLED",
 "createdAt": "2026-05-22 07:16:26 +0000 UTC",
 "accessedAt": "2026-05-26 00:19:45 +0000 UTC",
 "expiresAt": "2027-05-26 00:19:45 +0000 UTC"
}

Kidを確認

Key Details

Access Token

Create Token With Scopes:data:read data:create data:\Copy TokenOpen On jwt.io

ここまでの認証情報まとめ

- **CLIENT_ID** - ②で取得した Client ID
- **CLIENT_SECRET** - ②で取得した Client Secret
- **SSA_ID** - ⑤で取得した serviceAccountId
- **SSA_KEY_ID** - ⑦で表示させた kid 値 (Key ID)
- **SSA_KEY_PATH** - ⑥で入手した .pem ファイルパス
例) "C:¥Users¥<username>¥Downloads¥6e0585bb-5e56-4765-9b91-ecfe8edeb602.pem"

アクセストークンの取得：

⑧ .pem ファイルのロード

AUTODESK Platform Services Sample SSA Management Tool iscflKR6pCq0xLovje7tozJmY5a960SgyryqzG0AsOPoZXjg Log In GitHub

Accounts

Robot_Account@iscflKR6pCq0xLovje7tozJmY5a960SgyryqzG0AsOPoZXjg.adskserviceaccount.com

Delete Selected Account Create Account With Name: Enable/Disable Account

Account Details

```
{
  "serviceAccountId": "LMG204C3L8Y9HSD3",
  "email": "Robot_Account@iscflKR6pCq0xLovje7tozJmY5a960SgyryqzG0AsOPoZXjg.adskserviceaccount.com",
  "createdBy": "iscflKR6pCq0xLovje7tozJmY5a960SgyryqzG0AsOPoZXjg",
  "status": "ENABLED",
  "createdAt": "2026-05-22 07:16:26 +0000 UTC",
  "accessedAt": "2026-05-26 00:19:45 +0000 UTC",
  "expiresAt": "2027-05-26 00:19:45 +0000 UTC"
}
```

Keys

08cd8b2f-4755-4e5d-bcc4-ea3ec638b117

Delete Selected Key Create Key Enable/Disable Key

Key Details

```
{
  "kid": "08cd8b2f-4755-4e5d-bcc4-ea3ec638b117",
  "status": "ENABLED",
  "createdAt": "2026-05-22 07:17:19 +0000 UTC",
  "accessedAt": "2026-05-26 00:19:45 +0000 UTC"
}
```

Private Key

(事後表示させる場合)
.pem ファイル選択
Load From File

Load From File

Access Token

Create Token With Scopes: Copy Token Open On jwt.io

⑩ JWT トークンのデコード

Copy Token

Open on jwt.io

アクセストークンの検証：

⑩ JWT トークンのデコード

より詳しく JSON Web Token (JWT) を知るには？ [「JWTハンドブック」をダウンロード](#)

JWT Debugger デバッガー JWTの概要 ライブラリ コミュニティ

エンコードされた値 ☐ オートフォーカスを有効にする

> JSON Web Token (JWT)

```
eyJhbGciOiJSUzI1NiIsImtpZCI6ImFmNmFiOGFhLTlhYzItNDRkZi1iMDE5LTdhZTkzYTExYjg3NyIsInR5cCI6IkpXVCJ9.eyJhdWQiOiJodHRwczovL2F1dG9kZXNlLnVbSIsImNsaWVudF9pZCI6ImIzY2ZsS1I2cENxMHhMb3ZqZTd0b3pKbVktYTk2MFNneXJ5cXpHMEFzT1BvWlhqZyIsInVzZXJpZCI6IkkxNRZJRnEMzTDhZOUhTRDMLCjZyY29wZSI6WyJkYXRhOnJlYWQiLCJkYXRhOmNyZWF0ZSI6ImRhdGE6d3JpdGUlXSwiaXNzIjoiaHR0cHM6Ly9kZXZlbG9wZXIuYXBpLmF1dG9kZXNlLnVbSIsImV4cCI6MTc3OTc1ODM4NiwiYWVhbnRpdjoiU0EtMDgzMTg4YjctOGRjNS00OTJlLWlYyZTI0GI4MmRjMzg3ZTIxIn0.x9mJF9wgze9F2HwAoaHLZtzRc8pL-1JrcbbAACy8_Gs58_2xEz3rIZjWBBDqZai6gsIzuIWBp73Y6L5uzj1EW7IVI_r0EA71Kiuyms7ujnS77SkIeVoR4r1qmVa_GGRcWLM7hpUSILFpwCURKLCI8ussjZqpxupiQk0acL8TPLLIw-UGSuuizKnJSGh3IZ_eG7r2YzvKsf9IWDWme4xKwbJzxVejhQMAX6h3G7iVJDTElrg3qp9xawqpVT9lE2tH6wE5FSk2Ggh1YHan_sFXN_YHI1xzp2Mhatq6nnk3E1eIIu1-QNo_Fx0RYHTSQXFKBSl9bMtsI5Ia_1QsMRMrGw
```

デコードされたヘッダー

JSON クレームの表

```
{  "alg": "RS256",  "kid": "af6ab8aa-8ac2-44df-b019-7ae93a11b877",  "typ": "JWT"}
```

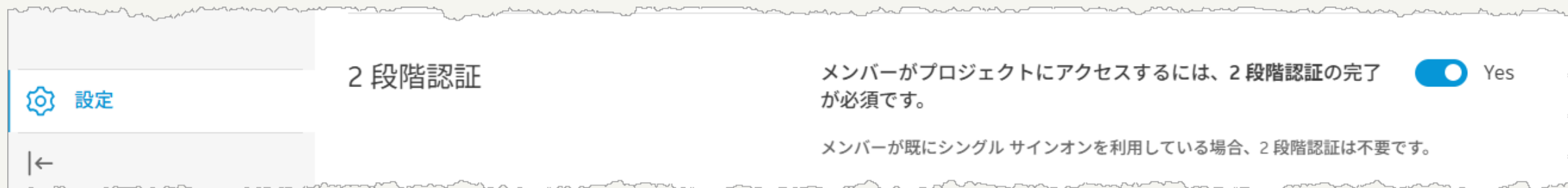
デコードされたペイロード

JSON クレームの表

```
{  "aud": "https://autodesk.com",  "client_id": "iscf1KR6pCq0xLovje7tozJmY5a960SgyryqzG0As0PoZXjg",  "userid": "LMG2Q4C3L8Y9HSD3",  "scope": [    "data:read",    "data:create",    "data:write"  ],  "iss": "https://developer.api.autodesk.com",  "exp": 1779758386,  "jti": "SA-083188b7-8dc5-492c-b2e2-8b82dc387e21"}
```

補足

- 12か月連続で未使用の SSA は無効化後、削除の対象に
- 1つの Client ID から作成可能な SSA は 10 個まで
 - ただし、有効利用可能なロボット アカウントはハブにつき 1 つ
- 2 段階認証設定した Forma ハブには現在 SSA は未対応

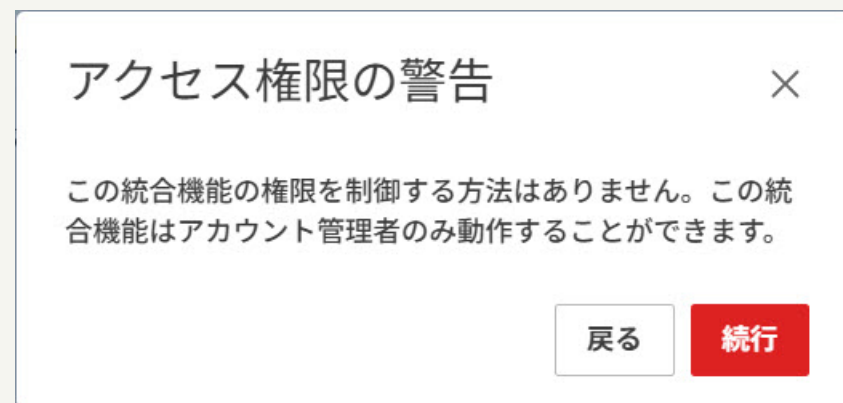


- 現在対応作業中

カスタム統合時の補足

- SSA 使用の Client ID の「カスタム統合」時に確認を表示
- SSA 未使用の Client ID の「カスタム統合」時に警告を表示

※ 2-legged、3-legged 使用の Client ID の登録も可能です
※ ご注意：将来 2-legged の廃止も検討されています



カスタム統合時の補足

複数 SSA アカウントを持つ Client ID のカスタム統合

- 使用するロボット アカウントの Email を **1つ**入力

複数のサービス アカウントが検出されました

この統合で使用するサービス アカウントの電子メールを入力してください。不明な場合は、開発者に問い合わせるガイダンスを受けてください。

サービス アカウントの電子メール: *

サービス アカウントの電子メールを入力

戻る

追加

Client ID 配下で有効利用可能になるロボット アカウントがハブに付き 1つ」になる理由

ロボット アカウントも人間アカウント、アプリと同様とする

ここまでの補足： SSA Manager 実装内容

- GitHub リポジトリで公開済
<https://github.com/autodesk-platform-services/ssa-manager-sample>
- Secure Service Account API
 - API で SSA 手順の自動化が可能
<https://aps.autodesk.com/en/docs/ssa/v1/tutorials/getting-started-with-ssa>
- ブログ記事
 - [Update : Secure Service Accounts \(SSA\) 一般公開](#)
- Forma オンラインヘルプ
 - [カスタム統合機能](#)
 - [セキュア サービス アカウント](#)

MCP サーバーの作成



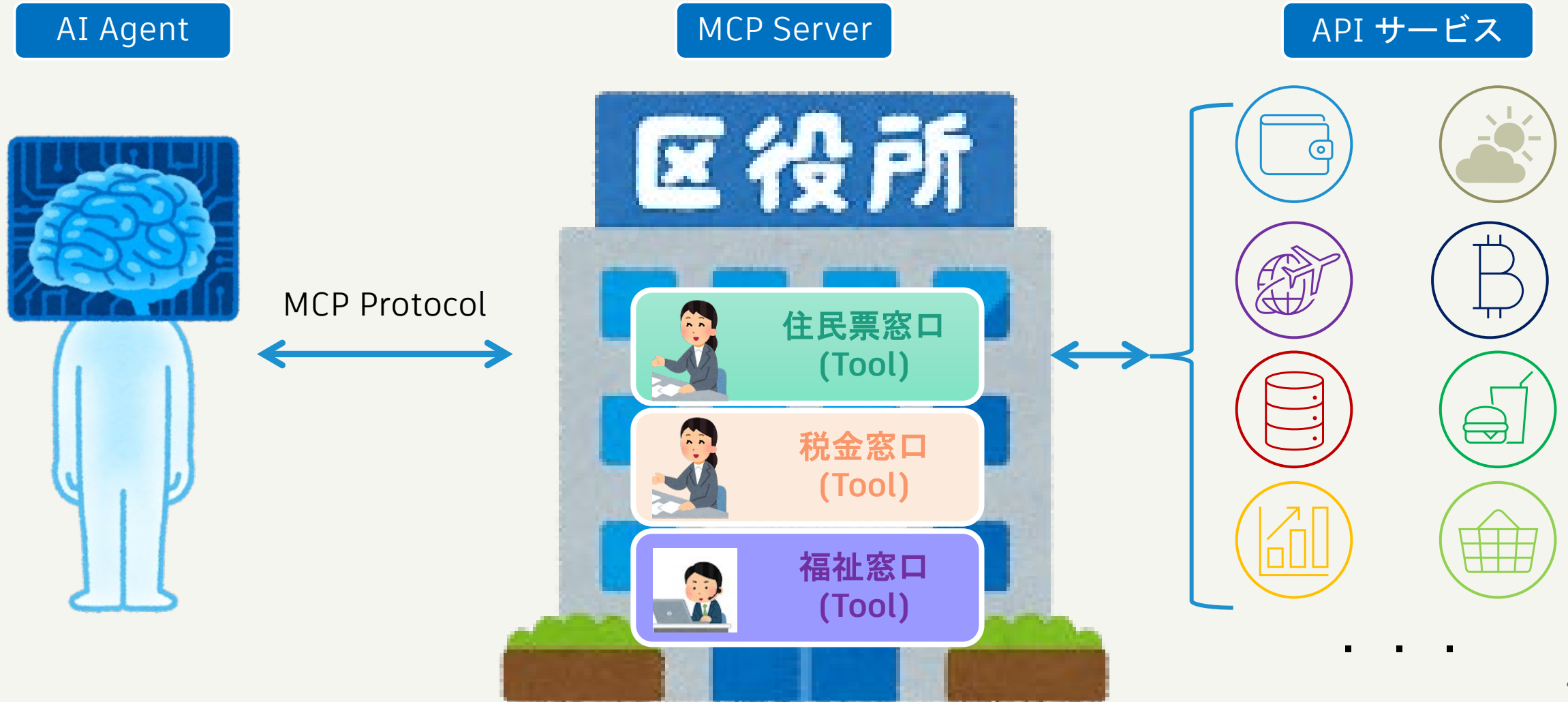
- Node.js を使って MCP Server の骨格を作成
- APS API はまだ呼ばない
- Part 3 の Tool 実装へ進める状態を作る

この章のゴール

- MCP Server の基本構造を理解する
- config.js / utils.js / server.js の役割を理解する
- MCP Inspector から接続確認できる
- Tool 実装へ進める状態を作る

MCP Server と MCP Tool

MCP Server は窓口をまとめて提供する場所、MCP Tool は個々の窓口



操作手順

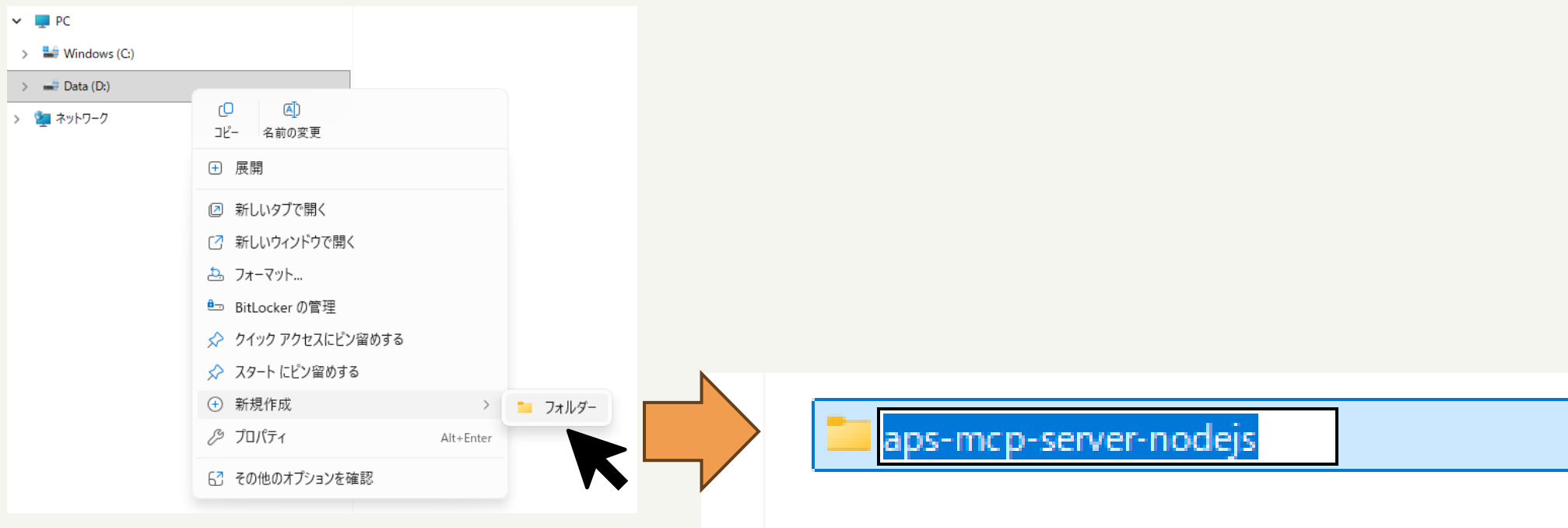
1. プロジェクトフォルダの作成、VS Codeでフォルダを開く
2. VS CodeのターミナルでNode.jsの初期化
3. package.json ファイルの“dependencies”の記述
4. ターミナルでパッケージのインストール
5. .envファイルの作成
6. config.jsの作成
7. util.jsの作成
8. tools/index.jsの作成
9. server.jsの作成
10. MCP Inspectorで接続確認

操作手順

1. プロジェクトフォルダの作成、VS Codeでフォルダを開く
2. VS CodeのターミナルでNode.jsの初期化
3. package.json ファイルの“dependencies”の記述
4. ターミナルでパッケージのインストール
5. .envファイルの作成
6. config.jsの作成
7. util.jsの作成
8. tools/index.jsの作成
9. server.jsの作成
10. MCP Inspectorで接続確認

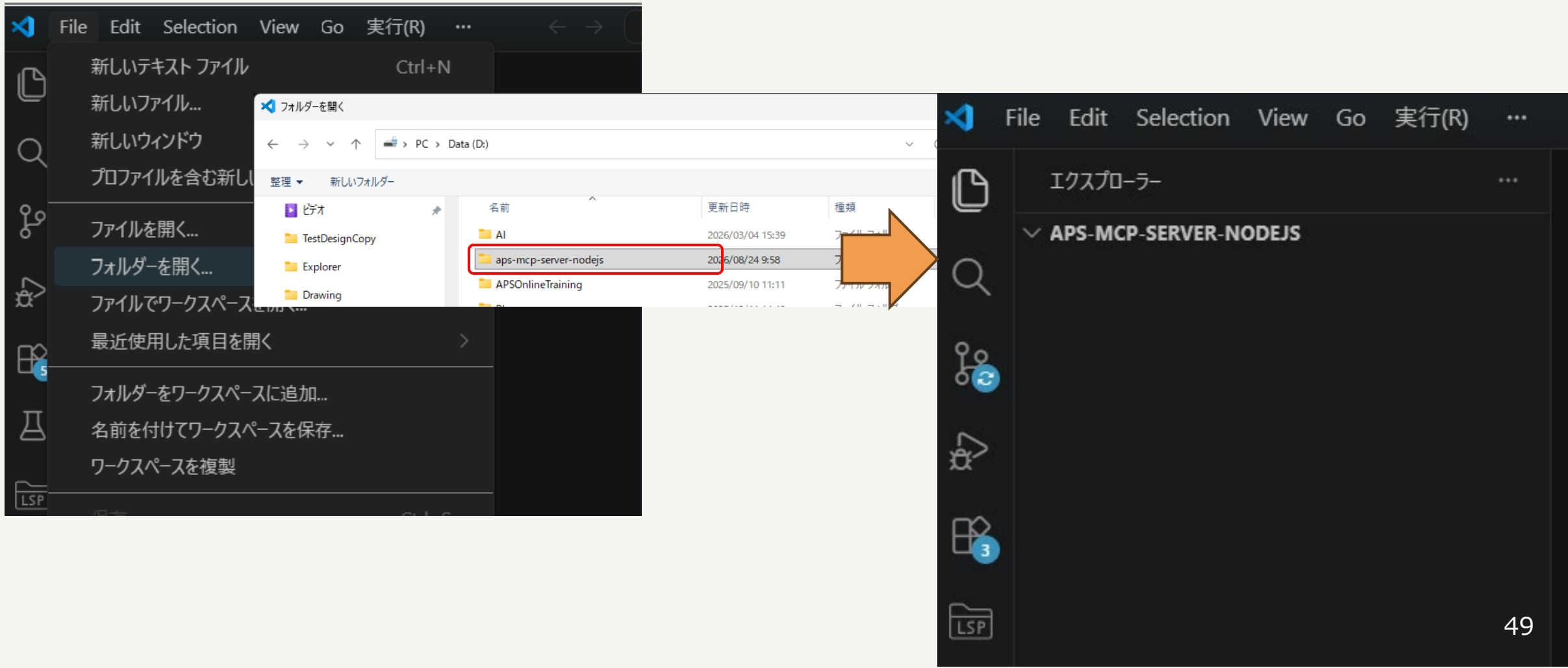
プロジェクトフォルダの作成

- エクスプローラから、右クリックメニューの[新規作成]-[フォルダ]を実行。作成されたフォルダの名前に“aps-mcp-server-nodejs”を入力。



VS Codeでフォルダを開く

- VS Codeを起動し、メニューから[File]-[フォルダを開く]を実行して作成したフォルダを開く。



D:¥

×

+

←

→

↑

↺

🖥️

>

PC

>

Data (D:)

>

Data (D:)の検索

+

新規作成

✂️

📄

📁

📁

🗑️

↕️

並べ替え

≡

表示

...

🏠 ホーム

🖼️ ギャラリー

> ☁️ Takehiro - Autodesk

🖥️ デスクトップ

📁 ダウンロード

🎵 ミュージック

🎬 ビデオ

📁 VaultLoginVS2026Sample

📁 VaultDialogLoginVS2026Sample

📁 2026年令和8年

📁 MCPWorkshop

> ☁️ OneDrive - Autodesk

▼ 🖥️ PC

> 🖥️ Windows (C:)

> 🖥️ Data (D:)

> 🌐 ネットワーク

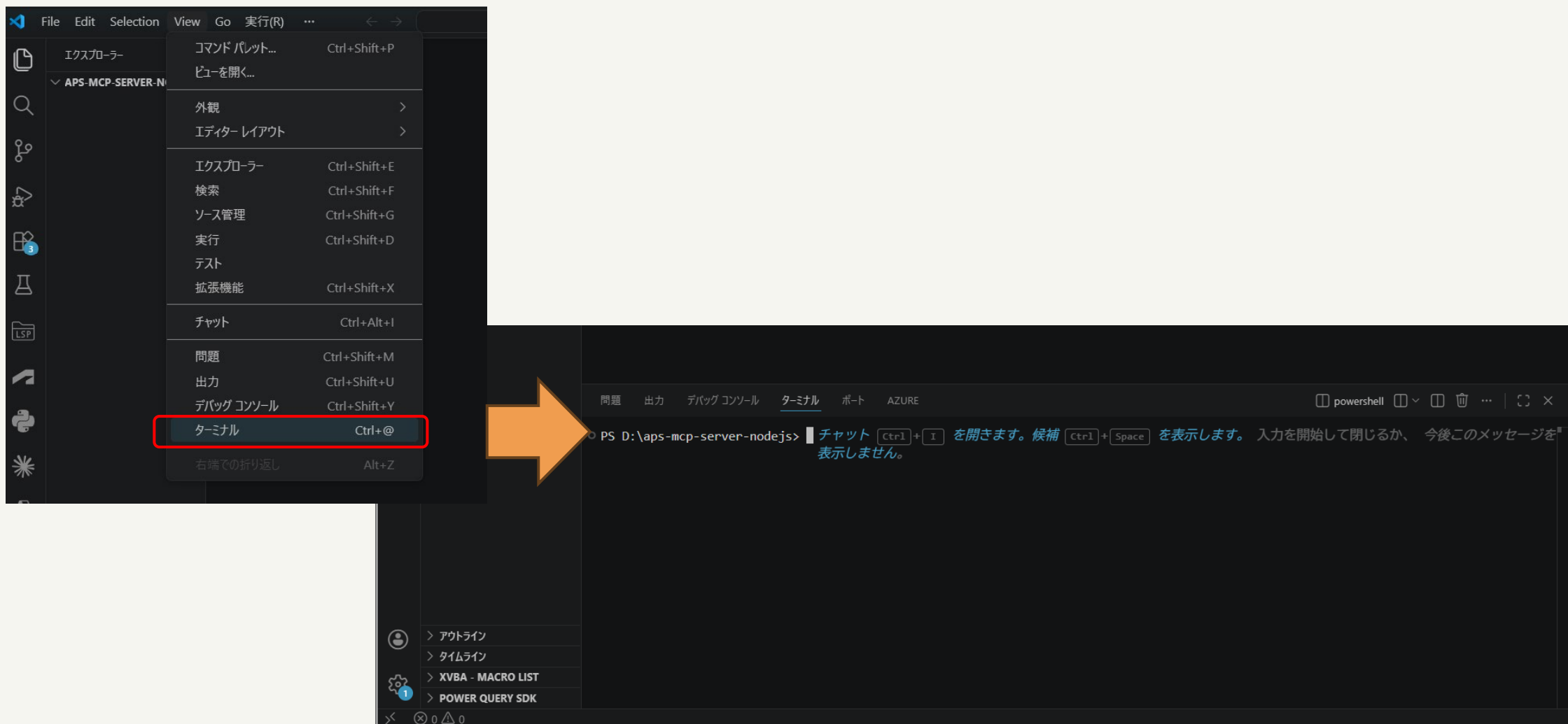
更新日時	種類	サイズ
2026/03/04 15:39	ファイル フォルダー	
2025/09/10 11:11	ファイル フォルダー	
2025/12/11 14:40	ファイル フォルダー	
2026/07/08 11:00	ファイル フォルダー	
2023/10/20 16:15	ファイル フォルダー	
2026/04/13 18:39	ファイル フォルダー	
2026/09/01 20:06	ファイル フォルダー	
2024/08/06 17:34	ファイル フォルダー	
2025/03/26 20:53	ファイル フォルダー	
2026/08/27 19:22	ファイル フォルダー	
2026/07/23 13:06	PEM ファイル	2 KB
2025/09/10 21:42	圧縮 (zip 形式) フォ...	5,384,979 KB

操作手順

1. プロジェクトフォルダの作成、VS Codeでフォルダを開く
2. VS CodeのターミナルでNode.jsの初期化
3. package.json ファイルの“dependencies”の記述
4. ターミナルでパッケージのインストール
5. .envファイルの作成
6. config.jsの作成
7. util.jsの作成
8. tools/index.jsの作成
9. server.jsの作成
10. MCP Inspectorで接続確認

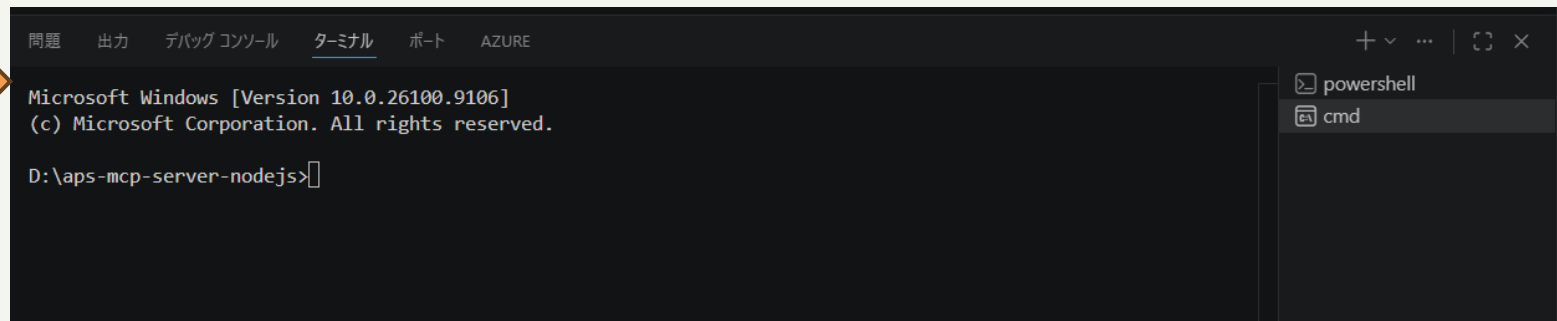
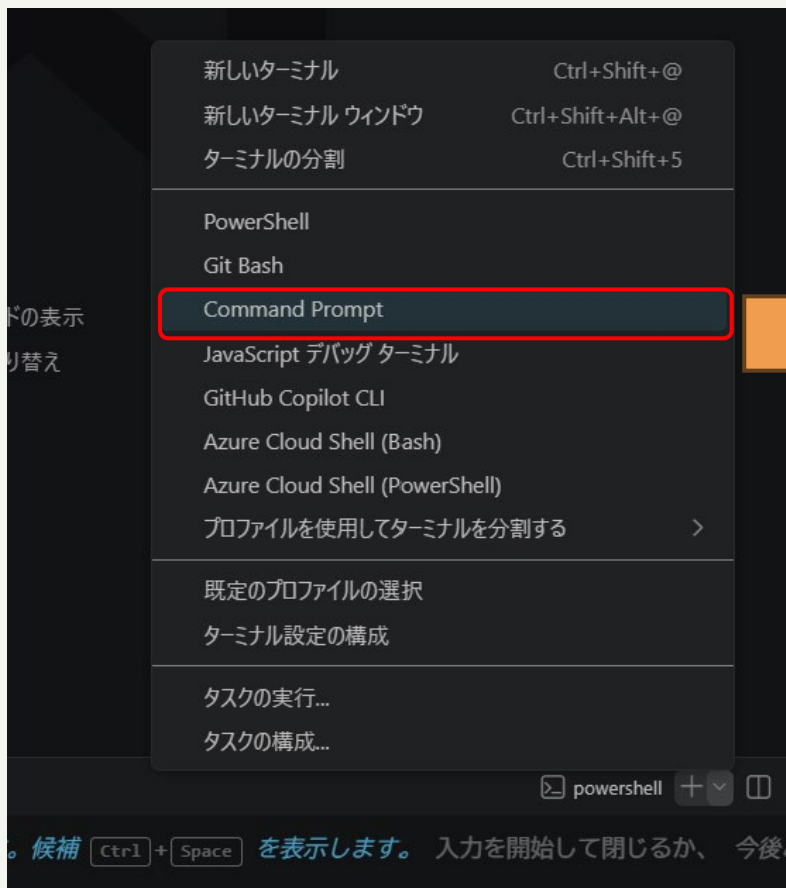
VS CodeのターミナルでNode.jsの初期化

- VS Codeで[View]-[ターミナルメニュー]から、ターミナルを起動



VS CodeのターミナルでNode.jsの初期化

- ターミナルの右上の V をクリックして、メニューから[Command Prompt]を選択する



VS CodeのターミナルでNode.jsの初期化

- Node.jsのバージョンの確認

```
D:\¥aps-mcp-server-nodejs>node --version  
v22.22.2
```

結果

✓ V16以上であればOK !

- NPMの疎通確認

```
D:\¥aps-mcp-server-nodejs>npm ping  
npm notice PING https: ...  
npm notice PONG 1125ms
```

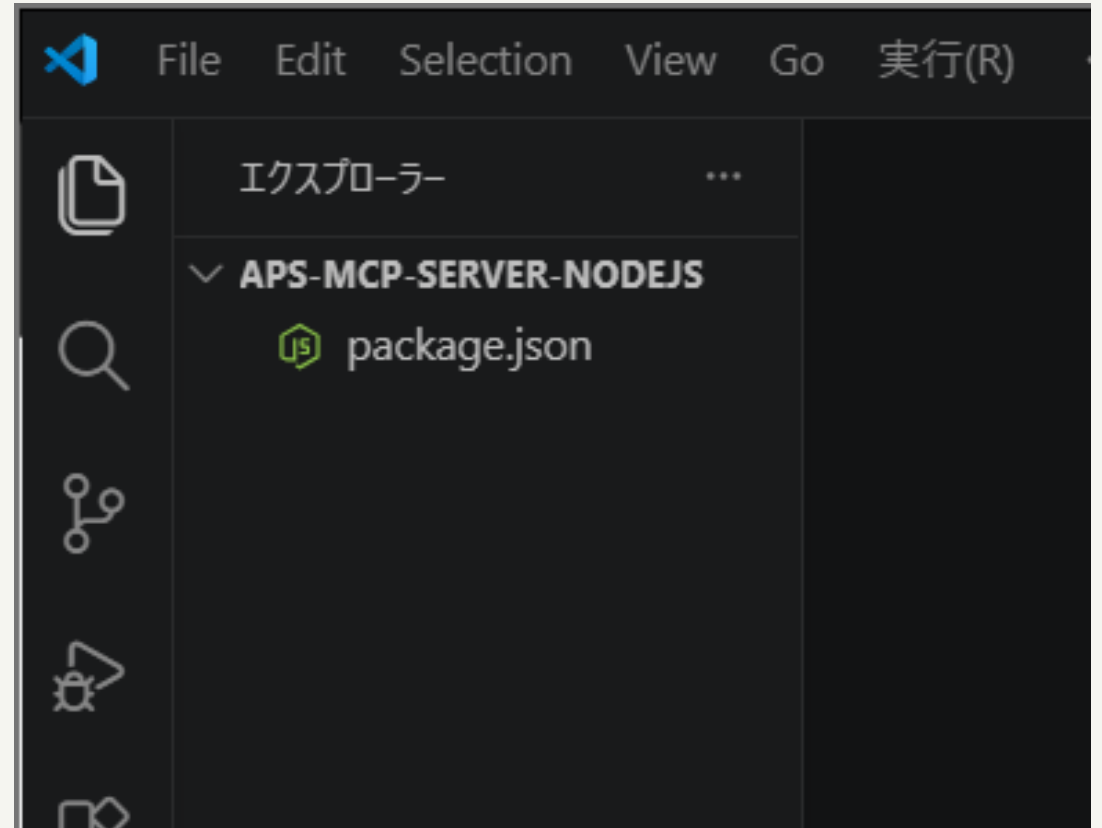
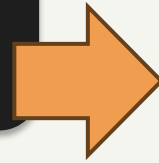
結果

✓ PONGが帰ってくればOK
✓ ECONNREFUSED などのエラーが出る場合は、
ネットワークの不具合やプロキシ設定に問題がある可能性があります

VS CodeのターミナルでNode.jsの初期化

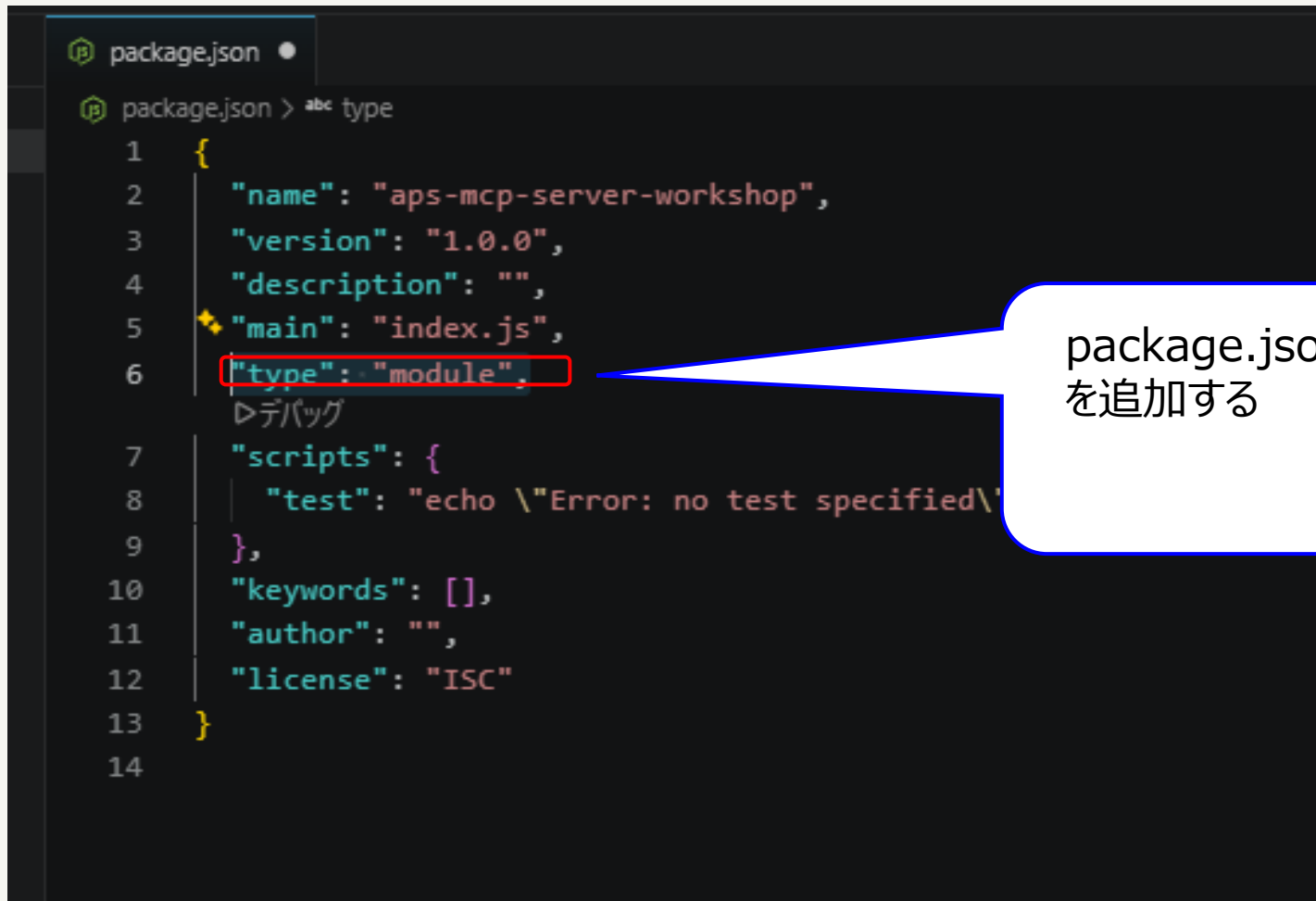
- Node.jsのプロジェクトを作成

```
D:\¥aps-mcp-server-nodejs> npm init -y
```



VS CodeのターミナルでNode.jsの初期化

- ES Modules (import/export) を使用



```
package.json
package.json > abc type
1  {
2    "name": "aps-mcp-server-workshop",
3    "version": "1.0.0",
4    "description": "",
5    "main": "index.js",
6    "type": "module",
7    "scripts": {
8      "test": "echo \"Error: no test specified\"",
9    },
10   "keywords": [],
11   "author": "",
12   "license": "ISC"
13 }
14
```

package.json を開いて"type": "module" を追加する

エクスプローラー ...

▼ APS-MCP-SERVER-NODEJS



aps-mcp-server-nodejs

チャットを開く

Ctrl + Alt + I

すべてのコマンドの表示

Ctrl + Shift + P

操作手順

1. プロジェクトフォルダの作成、VS Codeでフォルダを開く
2. VS CodeのターミナルでNode.jsの初期化
3. package.json ファイルの“dependencies”の記述
4. ターミナルでパッケージのインストール
5. .envファイルの作成
6. config.jsの作成
7. util.jsの作成
8. tools/index.jsの作成
9. server.jsの作成
10. MCP Inspectorで接続確認

package.json ファイルの“dependencies”の記述

- package.jsonファイルに、必要なモジュールを記述しファイルを上書き保存します

```
package.json
package.json > {} dependencies
1  {
2    "name": "aps-mcp-server-workshop",
3    "version": "1.0.0",
4    "description": "",
5    "main": "index.js",
6    "type": "module",
7    "scripts": {
8      "test": "echo \"Error: no test specified\" && exit 1"
9    },
10   "keywords": [],
11   "author": "",
12   "license": "ISC",
13   "dependencies": {
14     "@aps_sdk/construction-issues": "^1.1.0",
15     "@aps_sdk/data-management": "^1.1.2",
16     "@modelcontextprotocol/sdk": "^1.20.0",
17     "dotenv": "^17.2.3",
18     "jsonwebtoken": "^9.0.2",
19     "zod": "^3.24.2"
20   }
21 }
```

package.json に
"dependencies": {
 "@aps_sdk/construction-issues": "^1.1.0",
 "@aps_sdk/data-management": "^1.1.2",
 "@modelcontextprotocol/sdk": "^1.20.0",
 "dotenv": "^17.2.3",
 "jsonwebtoken": "^9.0.2",
 "zod": "^3.24.2"
}
を追加する。

操作手順

1. プロジェクトフォルダの作成、VS Codeでフォルダを開く
2. VS CodeのターミナルでNode.jsの初期化
3. package.json ファイルの“dependencies”の記述
4. ターミナルでパッケージのインストール
5. .envファイルの作成
6. config.jsの作成
7. util.jsの作成
8. tools/index.jsの作成
9. server.jsの作成
10. MCP Inspectorで接続確認

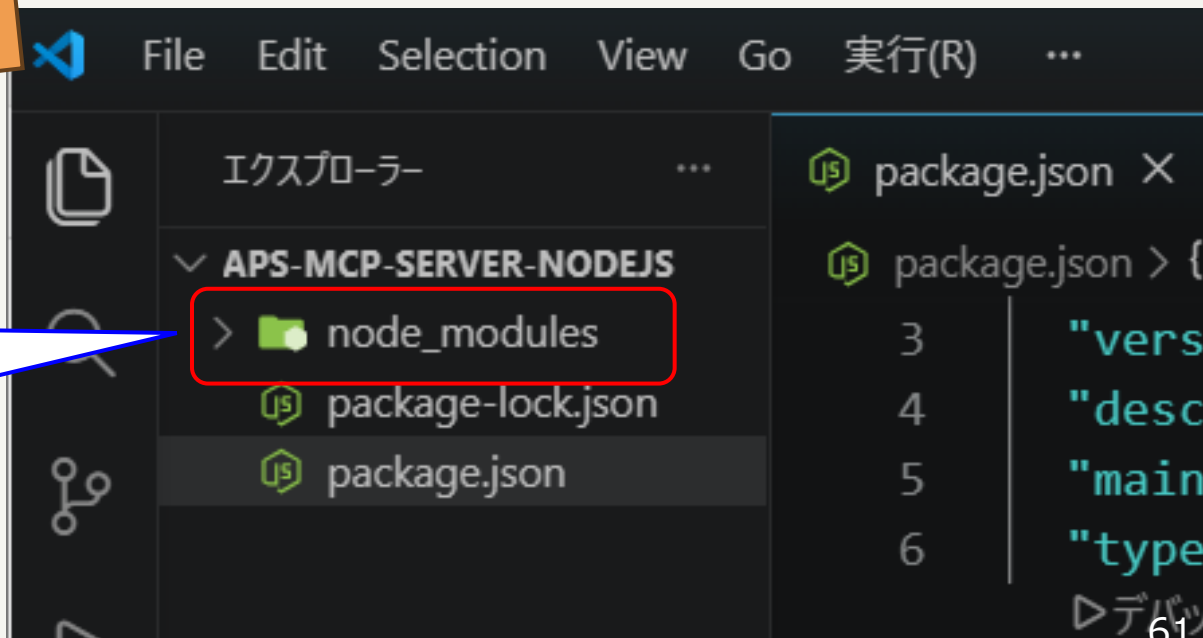
ターミナルでパッケージのインストール

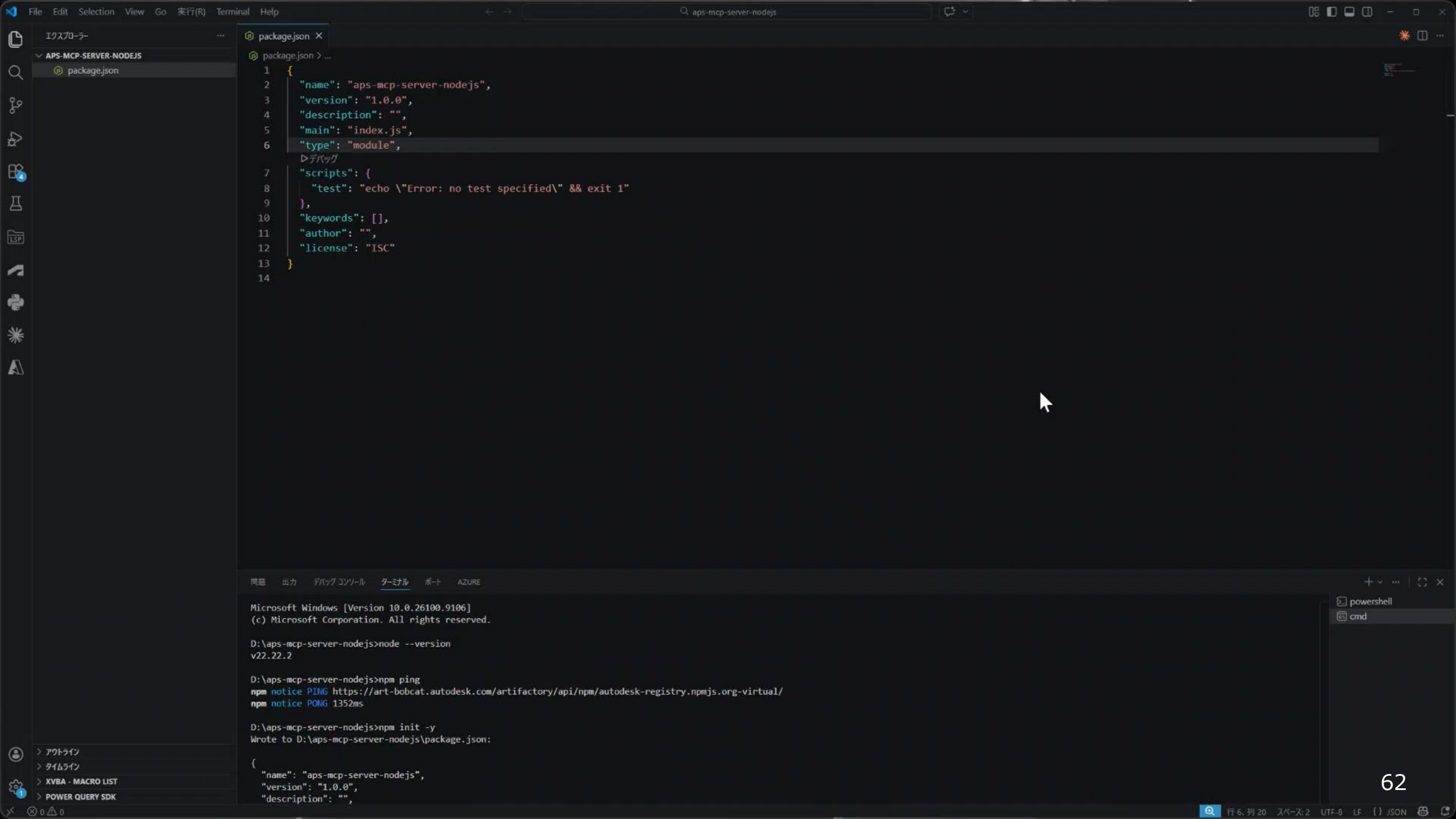
- Npm installを実行し、パッケージのインストール

```
D:\¥aps-mcp-server-nodejs>npm install  
added 153 packages in 50s
```

```
38 packages are looking for funding  
run `npm fund` for details
```

node_modules フォルダ配下に
パッケージがインストールされる





package.json X

package.json > ...

```
1 {
2   "name": "aps-mcp-server-nodejs",
3   "version": "1.0.0",
4   "description": "",
5   "main": "index.js",
6   "type": "module",
7   "scripts": {
8     "test": "echo \"Error: no test specified\" && exit 1"
9   },
10  "keywords": [],
11  "author": "",
12  "license": "ISC"
13 }
14
```

問題 出力 デバッグ コンソール ターミナル ポート AZURE

Microsoft Windows [Version 10.0.26100.9106]
(c) Microsoft Corporation. All rights reserved.

D:\aps-mcp-server-nodejs>node --version
v22.22.2

D:\aps-mcp-server-nodejs>npm ping
npm notice PING https://art-bobcat.autodesk.com/artifactory/api/npm/autodesk-registry.npmjs.org-virtual/
npm notice PONG 1352ms

D:\aps-mcp-server-nodejs>npm init -y
Wrote to D:\aps-mcp-server-nodejs\package.json:

```
{
  "name": "aps-mcp-server-nodejs",
  "version": "1.0.0",
  "description": "",
```

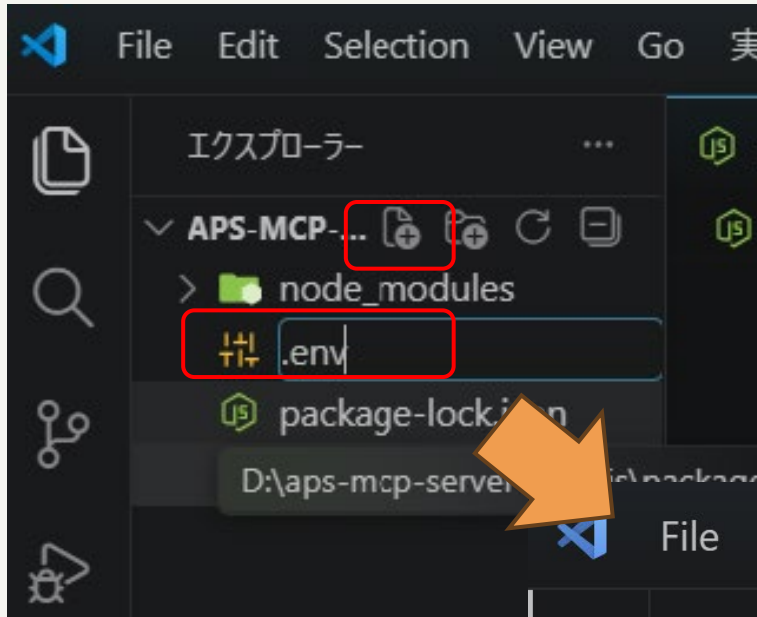
+ ...
powershell
cmd

62

操作手順

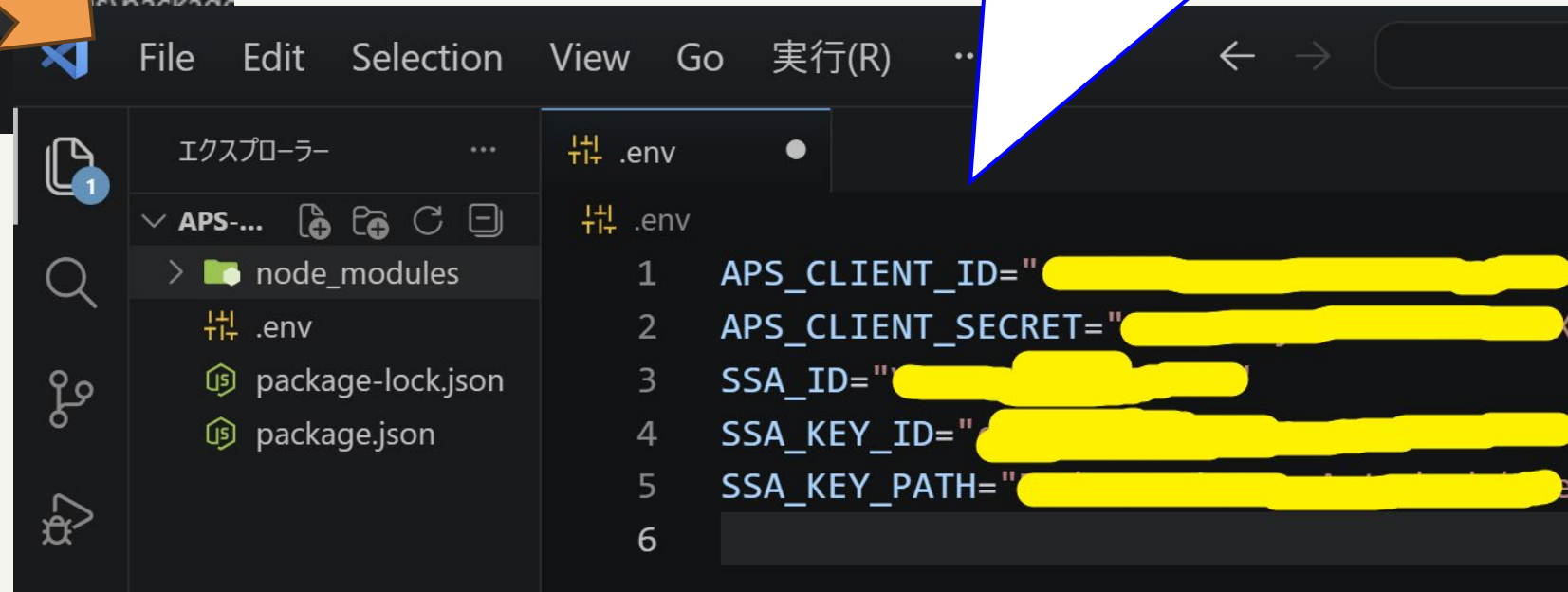
1. プロジェクトフォルダの作成、VS Codeでフォルダを開く
2. VS CodeのターミナルでNode.jsの初期化
3. package.json ファイルの“dependencies”の記述
4. ターミナルでパッケージのインストール
5. .envファイルの作成
6. config.jsの作成
7. util.jsの作成
8. tools/index.jsの作成
9. server.jsの作成
10. MCP Inspectorで接続確認

.envファイルの作成



.envファイルの記述内容

- `APS_CLIENT_ID`
- `APS_CLIENT_SECRET`
- `SSA_ID`
- `SSA_KEY_ID`
- `SSA_KEY_PATH`(Windows でもパス区切りは / を推奨)



Visual Studio Code interface showing the `package.json` file for the `aps-mcp-server-nodejs` project.

```
1 {
2   "name": "aps-mcp-server-nodejs",
3   "version": "1.0.0",
4   "description": "",
5   "main": "index.js",
6   "type": "module",
7   "scripts": {
8     "test": "echo \"Error: no test specified\" && exit 1"
9   },
10  "keywords": [],
11  "author": "",
12  "license": "ISC",
13  "dependencies": {
14    "@aps_sdk/construction-issues": "^1.1.0",
15    "@aps_sdk/data-management": "^1.1.2",
16    "@modelcontextprotocol/sdk": "^1.20.0",
17    "dotenv": "^17.2.3",
18    "jsonwebtoken": "^9.0.2",
19    "zod": "^3.24.2"
20  }
21 }
```

The terminal shows the command `D:\aps-mcp-server-nodejs>npm install` and its output:

```
added 153 packages in 50s
38 packages are looking for funding
  run `npm fund` for details
D:\aps-mcp-server-nodejs>
```

Autodesk Platform Services interface showing the MCP-WorkshopOnline2026-09-16 application settings.

MCP-WorkshopOnline2026-09-16

App settings | API usage

Client Credentials

The Client ID and Client Secret are used to obtain access tokens, which you must use to authenticate API calls.

Client ID: TkEWj71L4B0Tjbe6FTUJ8HBz3itJ3wRlGUGBBY4mYIvNN20p

Client Secret:

[Regenerate](#)

General Settings

Name: MCP-WorkshopOnline2026-09-16

Description:

File Explorer view showing the contents of the `D:\aps-mcp-server-nodejs` directory:

名前	更新日時	種類	サイズ
node_modules	2026/09/04 14:21	ファイル フォルダー	
6651708d-424e-49ad-a21e-4f1d92f652f6.pem	2026/07/23 13:06	PEM ファイル	2 KB
package.json	2026/09/04 14:20	JSON ソース ファイル	1 KB
package-lock.json	2026/09/04 14:21	JSON ソース ファイル	75 KB

操作手順

1. プロジェクトフォルダの作成、VS Codeでフォルダを開く
2. VS CodeのターミナルでNode.jsの初期化
3. package.json ファイルの“dependencies”の記述
4. ターミナルでパッケージのインストール
5. .envファイルの作成
6. config.jsの作成
7. util.jsの作成
8. tools/index.jsの作成
9. server.jsの作成
10. MCP Inspectorで接続確認

config.js の作成

The image shows a VS Code editor with a dark theme. The Explorer sidebar on the left shows the file structure of a project named 'APS-MCP-...'. The 'node_modules' directory is expanded, and a new file 'config.js' is being created. The main editor shows the code for 'config.js'.

Background Tutorial: APS MCP Server: Tutorial

- > Overview
- > Part 1: Prepare Service Account
- > Part 2: Create MCP Server
 - Create Node.js Project
 - Configuration Management

Create Configuration Module

Create a `config.js` file in your project directory with the following code:

```
import path from "node:path";
import url from "node:url";
import dotenv from "dotenv";

const __filename = url.fileURLToPath(import.meta.url);
const __dirname = path.dirname(__filename);
dotenv.config({ path: path.resolve(__dirname, ".env") });
const { APS_CLIENT_ID, APS_CLIENT_SECRET, SSA_ID, SSA_KEY_ID, SSA_KEY_PATH } = process.env;
if (!APS_CLIENT_ID || !APS_CLIENT_SECRET || !SSA_ID || !SSA_KEY_ID || !SSA_KEY_PATH) {
  console.error("Missing one or more required environment variables");
  process.exit(1);
}

export {
  APS_CLIENT_ID,
  APS_CLIENT_SECRET,
  SSA_ID,
  SSA_KEY_ID,
  SSA_KEY_PATH
}
```

The code in the main editor is as follows:

```
1 import path from "node:path";
2 import url from "node:url";
3 import dotenv from "dotenv";
4
5 const __filename = url.fileURLToPath(import.meta.url);
6 const __dirname = path.dirname(__filename);
7 dotenv.config({ path: path.resolve(__dirname, ".env") });
8 const { APS_CLIENT_ID, APS_CLIENT_SECRET, SSA_ID, SSA_KEY_ID, SSA_KEY_PATH } = process.env;
9 if (!APS_CLIENT_ID || !APS_CLIENT_SECRET || !SSA_ID || !SSA_KEY_ID || !SSA_KEY_PATH) {
10   console.error("Missing one or more required environment variables");
11   process.exit(1);
12 }
13
14 export {
15   APS_CLIENT_ID,
16   APS_CLIENT_SECRET,
17   SSA_ID,
18   SSA_KEY_ID,
19   SSA_KEY_PATH
20 }
21
```

aps-mcp-server-nodejs

エクスプローラー

- APS-MCP-SERVER-NODEJS
 - node_modules
 - .env
 - 6651708d-424e-49ad-a21e-4f1d92f652f6...
 - package-lock.json
 - package.json

チャットを開く **Ctrl + Alt + I**

すべてのコマンドの表示 **Ctrl + Shift + P**

フォルダーを指定して検索 **Ctrl + Shift + F**

問題 出力 デバッグ コンソール ターミナル ポート AZURE

```
},
"keywords": [],
"author": "",
"license": "ISC"
}

D:\aps-mcp-server-nodejs>npm install

added 153 packages in 50s

38 packages are looking for funding
  run `npm fund` for details

D:\aps-mcp-server-nodejs>
```

アクタイン
タイムライン
XVBA - MACRO LIST
POWER QUERY SDK

MCP-WorkshopOnline2026-0 | Sample MCP Server | Part 2: Create MCP Server | Gemini に相談

autodesk-platform-services.github.io/aps-mcp-server-nodejs/#/part2-mcp-server/index?id=...

Sales Force | Sales Force | Sales Force | AI Training | Autodesk Sites | Autodesk support | Autodesk products... | Home

APS MCP Server: Tutorial

- Overview
- Part 1: Prepare Service Account
- Part 2: Create MCP Server
 - Create Node.js Project
 - Configuration Management
 - Shared Logic
 - Setup MCP Server
 - Try it out
- Part 3: Create MCP Tools
- Part 4: Integrate with MCP Clients

```
APS_CLIENT_SECRET="your-client-secret-here"
SSA_ID="your-service-account-id"
SSA_KEY_ID="your-key-id"
SSA_KEY_PATH="/absolute/path/to/your-key.pem"
```

Replace the placeholders with your actual values collected in [Part 1](#).

Create Configuration Module

Create a `config.js` file in your project directory with the following code:

```
javascript
import path from "node:path";
import url from "node:url";
import dotenv from "dotenv";

const __filename = url.fileURLToPath(import.meta.url);
const __dirname = path.dirname(__filename);
dotenv.config({ path: path.resolve(__dirname, ".env"), quiet: true });
const { APS_CLIENT_ID, APS_CLIENT_SECRET, SSA_ID, SSA_KEY_ID, SSA_KEY_PATH } =
  if (!APS_CLIENT_ID || !APS_CLIENT_SECRET || !SSA_ID || !SSA_KEY_ID || !SSA_KEY_PATH) {
    console.error("Missing one or more required environment variables: APS_CLI");
    process.exit(1);
  }

export {
  APS_CLIENT_ID,
  APS_CLIENT_SECRET,
  SSA_ID,
  SSA_KEY_ID,
  SSA_KEY_PATH
}
```

This module:

- Loads environment variables from `.env`
- Validates all required credentials are present
- Exits with an error if any credentials are missing
- Exports configuration for use by other modules

Shared Logic

Next, let's implement a shared logic that will be used by all MCP tools we will build in [Part 3](#).

68

config.js の内容



The image shows a VS Code editor window with a file explorer on the left and a code editor in the center. The file explorer shows a project structure with files like .env, config.js, package-lock.json, and package.json. The code editor displays the content of config.js. Three callout boxes highlight specific parts of the code:

- .envファイルの記述
内容を読み込み**: Points to lines 5-8, which load the .env file and set environment variables.
- 必須値 *validation***: Points to lines 9-12, which check for required environment variables and exit with an error if any are missing.
- 他 *module* への *export***: Points to lines 14-20, which export the environment variables as a module.

```
1 import path from "node:path";
2 import url from "node:url";
3 import dotenv from "dotenv";
4
5 const __filename = url.fileURLToPath(import.meta.url);
6 const __dirname = path.dirname(__filename);
7 dotenv.config({ path: path.resolve(__dirname, ".env"), quiet: true });
8 const { APS_CLIENT_ID, APS_CLIENT_SECRET, SSA_ID, SSA_KEY_ID, SSA_KEY_PATH } = process.env;
9 if (!APS_CLIENT_ID || !APS_CLIENT_SECRET || !SSA_ID || !SSA_KEY_ID || !SSA_KEY_PATH) {
10   console.error("Missing one or more required environment variables: APS_CLIENT_ID, APS_CLIENT_SECRET, SSA_ID, SSA_KEY_ID, SSA_KEY_PATH");
11   process.exit(1);
12 }
13
14 export {
15   APS_CLIENT_ID,
16   APS_CLIENT_SECRET,
17   SSA_ID,
18   SSA_KEY_ID,
19   SSA_KEY_PATH
20 }
21
```

操作手順

1. プロジェクトフォルダの作成、VS Codeでフォルダを開く
2. VS CodeのターミナルでNode.jsの初期化
3. package.json ファイルの“dependencies”の記述
4. ターミナルでパッケージのインストール
5. .envファイルの作成
6. config.jsの作成
7. util.jsの作成
8. tools/index.jsの作成
9. server.jsの作成
10. MCP Inspectorで接続確認

util.jsの作成 : APS 認証層

- util.jsを追加して、処理を追加

The screenshot shows a VS Code editor with two windows. The left window displays the file explorer for a project named 'APS-MCP-SERVER-WORKSH...'. The 'node_modules' folder is expanded, and a new file named 'utils.js' is being created, highlighted with a red box and an orange arrow. The right window shows the 'APS MCP Server: Tutorial' documentation, specifically the 'Create Common Utilities' section, which instructs to create a 'utils.js' file with the following content. An orange arrow points from the documentation to the code in the editor.

```
1 import fs from "node:fs/promises";
2 import jwt from "jsonwebtoken";
3 import { DataManagementClient } from "@aps_sdk/data-management";
4 import { IssuesClient } from "@aps_sdk/construction-issues";
5 import { APS_CLIENT_ID, APS_CLIENT_SECRET, SSA_ID, SSA_KEY_ID, SSA_KEY_PATH } from ".env";
6
7 const TOKEN_ENDPOINT = "https://developer.api.autodesk.com/authentication/v2/token";
8
9 class ServiceAccountAuthenticationProvider {
10   constructor(scopes) {
11     this._accessToken = null;
12     this._expiresAt = 0;
13     this._scopes = scopes;
14   }
15
16   async getAccessToken() {
17     if (!this._accessToken || this._expiresAt < Date.now()) {
18       const assertion = await this._createAssertion(this._scopes);
19       const { access_token, expires_in } = await this._exchangeAccessToken(assertion);
20       this._accessToken = access_token;
21       this._expiresAt = Date.now() + expires_in * 1000;
22     }
23     return this._accessToken;
```

File Edit Selection View Go ...

aps-mcp-server-nodes

config.js

```
1 import path from "node:path";
2 import url from "node:url";
3 import dotenv from "dotenv";
4
5 const __filename = url.fileURLToPath(import.meta.url);
6 const __dirname = path.dirname(__filename);
7 dotenv.config({ path: path.resolve(__dirname, ".env"), quiet: true });
8 const { APS_CLIENT_ID, APS_CLIENT_SECRET, SSA_ID, SSA_KEY_ID, SSA_KEY_PATH } = process.env;
9 if (!APS_CLIENT_ID || !APS_CLIENT_SECRET || !SSA_ID || !SSA_KEY_ID || !SSA_KEY_PATH) {
10   console.error("Missing one or more required environment variables: APS_CLIENT_ID, APS_CLIENT_SECRET, SSA_ID, SSA_KEY_ID, SSA_KEY_PATH");
11   process.exit(1);
12 }
13
14 export {
15   APS_CLIENT_ID,
16   APS_CLIENT_SECRET,
17   SSA_ID,
18   SSA_KEY_ID,
19   SSA_KEY_PATH
20 }
21
```

問題 出力 デバッグ コンソール ターミナル ポート AZURE

```
},
"keywords": [],
"author": "",
"license": "ISC"
}

D:\aps-mcp-server-nodes>npm install

added 153 packages in 50s

38 packages are looking for funding
  run `npm fund` for details

D:\aps-mcp-server-nodes>
```

行 21, 列 1 スペース: 4 UTF-8 CRLF {} JavaScript

MCP-WorkshopOnline2026-0 x Sample SSA Management Tool x Part 2: Create MCP Server x Gemini に相談

autodesk-platform-services.github.io/aps-mcp-server-nodesjs/#/part2-mcp-server/index?id=...

Sales Force Sales Force Sales Force AI Training Autodesk Sites Autodesk support Autodesk products... Home

APS MCP Server: Tutorial

- Overview
- Part 1: Prepare Service Account
 - Create Node.js Project
 - Configuration Management
 - Shared Logic
 - Setup MCP Server
 - Try it out
- Part 3: Create MCP Tools
- Part 4: Integrate with MCP Clients

Create Common Utilities

Create a `utils.js` file in the root folder with the following content:

```
import fs from "node:fs/promises";
import jwt from "jsonwebtoken";
import { DataManagementClient } from "@aps_sdk/data-management";
import { IssuesClient } from "@aps_sdk/construction-issues";
import { APS_CLIENT_ID, APS_CLIENT_SECRET, SSA_ID, SSA_KEY_ID, SSA_KEY_PATH } from "utils";

const TOKEN_ENDPOINT = "https://developer.api.autodesk.com/authentication/v2/token";

class ServiceAccountAuthenticationProvider {
  constructor(scopes) {
    this._accessToken = null;
    this._expiresAt = 0;
    this._scopes = scopes;
  }

  async getAccessToken() {
    if (!this._accessToken || this._expiresAt < Date.now()) {
      const assertion = await this._createAssertion(this._scopes);
      const { access_token, expires_in } = await this._exchangeAccessToken(assertion);
      this._accessToken = access_token;
      this._expiresAt = Date.now() + expires_in * 1000;
    }

    return this._accessToken;
  }

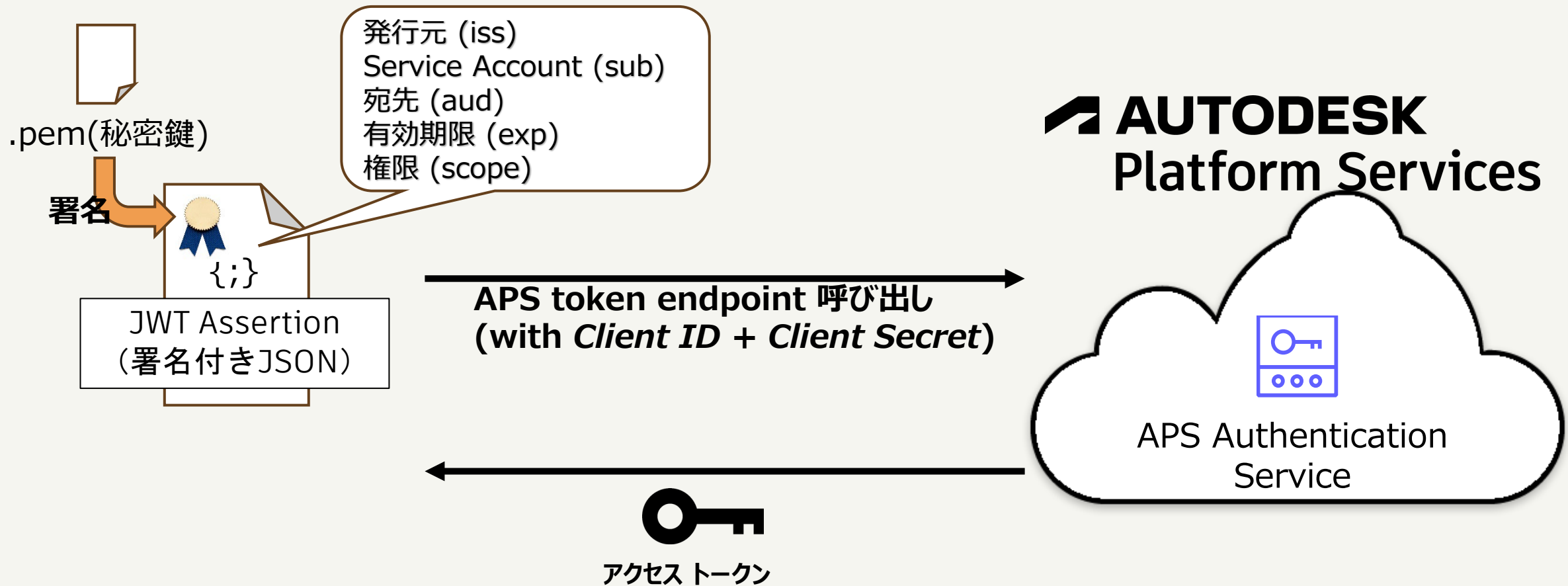
  async _createAssertion(scopes) {
    const expiresAt = Math.floor(Date.now() / 1000) + 300;
    const payload = { iss: APS_CLIENT_ID, sub: SSA_ID, aud: TOKEN_ENDPOINT, exp: expiresAt };
    const privateKey = await fs.readFile(SSA_KEY_PATH, "utf-8");
    const options = {
      algorithm: "RS256",
      header: { alg: "RS256", kid: SSA_KEY_ID },
      noTimestamp: true
    };
    return jwt.sign(payload, privateKey, options);
  }

  async _exchangeAccessToken(assertion) {
    const headers = {
      "Accept": "application/json",
      "Authorization": `Basic ${Buffer.from(`${APS_CLIENT_ID}:${APS_CLIENT_SECRET}`).toString("base64")}`,
      "Content-Type": "application/x-www-form-urlencoded"
    };
    const body = new URLSearchParams({ grant_type: "urn:ietf:params:oauth:grant-type:jwt-bearer", assertion: assertion });
    const response = await fetch(TOKEN_ENDPOINT, { method: "POST", headers, body });
    const { access_token, expires_in } = await response.json();
    return { access_token, expires_in };
  }
}
```

72

utils.js : APS 認証層

●処理の流れ

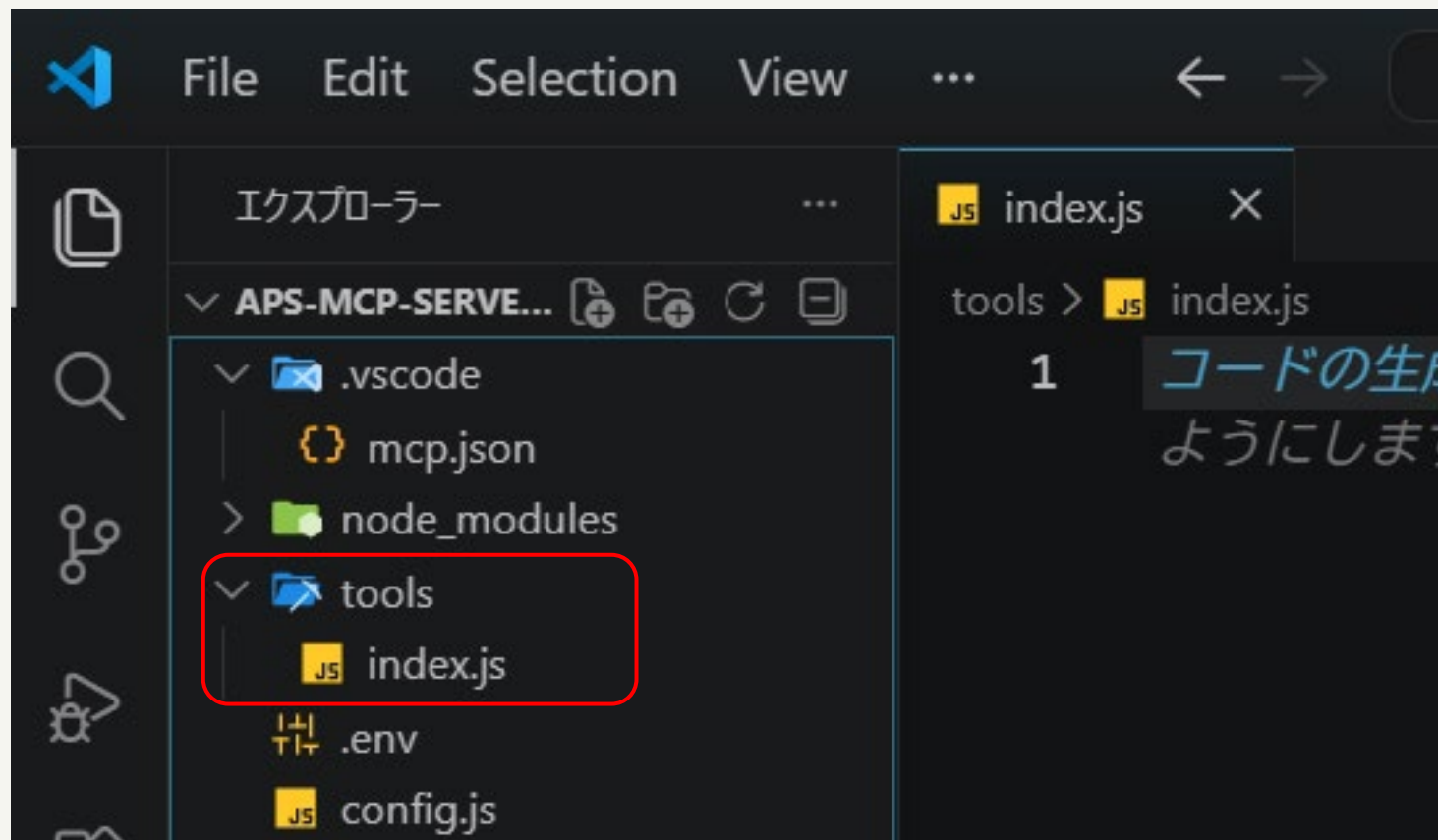


操作手順

1. プロジェクトフォルダの作成、VS Codeでフォルダを開く
2. VS CodeのターミナルでNode.jsの初期化
3. package.json ファイルの“dependencies”の記述
4. ターミナルでパッケージのインストール
5. .envファイルの作成
6. config.jsの作成
7. util.jsの作成
8. tools/index.jsの作成
9. server.jsの作成
10. MCP Inspectorで接続確認

tools/index.js の追加

- Tool の登録口
 - Part 3 で Tool を追加
- toolsフォルダを作成し、配下にindex.jsを作成（中身はまだ空のままでOK）



File Edit Selection View Go ...

aps-mcp-server-nodes

config.js utils.js

```
1 import fs from "node:fs/promises";
2 import jwt from "jsonwebtoken";
3 import { DataManagementClient } from "@aps_sdk/data-management";
4 import { IssuesClient } from "@aps_sdk/construction-issues";
5 import { APS_CLIENT_ID, APS_CLIENT_SECRET, SSA_ID, SSA_KEY_ID, SSA_KEY_PATH } from "config";
6
7 const TOKEN_ENDPOINT = "https://developer.api.autodesk.com/authentication/v2/token";
8
9 class ServiceAccountAuthenticationProvider {
10   constructor(scopes) {
11     this._accessToken = null;
12     this._expiresAt = 0;
13     this._scopes = scopes;
14   }
15
16   async getAccessToken() {
17     if (!this._accessToken || this._expiresAt < Date.now()) {
18       const assertion = await this._createAssertion(this._scopes);
19       const { access_token, expires_in } = await this._exchangeAccessToken(assertion);
20       this._accessToken = access_token;
21       this._expiresAt = Date.now() + expires_in * 1000;
22     }
23     return this._accessToken;
24   }
25
26   async _createAssertion(scopes) {
27     const expiresAt = Math.floor(Date.now() / 1000) + 300;
28     const payload = { iss: APS_CLIENT_ID, sub: SSA_ID, aud: TOKEN_ENDPOINT, exp: expiresAt };
29     const privateKey = await fs.readFile(SSA_KEY_PATH, "utf-8");
30     const options = {
31       algorithm: "RS256",
32       header: { alg: "RS256", kid: SSA_KEY_ID },
33       noTimestamp: true
34     };
35     const assertion = jwt.sign(payload, privateKey, options);
36     return assertion;
37   }
38
39   async _exchangeAccessToken(assertion) {
40     const headers = {
41       "Accept": "application/json",
42       "Authorization": `Basic ${Buffer.from(`${APS_CLIENT_ID}:${APS_CLIENT_SECRET}`).toString("base64")}`,
43       "Content-Type": "application/x-www-form-urlencoded"
44     };
45     const body = new URLSearchParams({ grant_type: "urn:ietf:params:oauth:grant-type:jwt-bearer", assertion: assertion });
46     const response = await fetch(TOKEN_ENDPOINT, { method: "POST", headers, body });
47     const json = await response.json();
48     return json;
49   }
50 }
```

問題 出力 デバッグ コンソール ターミナル ポート AZURE

```
},
"keywords": [],
"author": "",
"license": "ISC"
}

D:\aps-mcp-server-nodes>npm install

added 153 packages in 50s

38 packages are looking for funding
  run `npm fund` for details

D:\aps-mcp-server-nodes>
```

行 22, 列 10 スペース:4 UTF-8 CRLF {} JavaScript

MCP-WorkshopOnline2026-0 MCP-WorkshopOnline2026-0 Sample SSA Management Tool Part 2: Create MCP Server

autodesk-platform-services.github.io/aps-mcp-server-nodes/#/part2-mcp-server/index?id=...

Sales Force Sales Force Sales Force AI Training Autodesk Sites Autodesk support Autodesk products... Home

APS MCP Server: Tutorial

- Overview
- Part 1: Prepare Service Account
 - Create Node.js Project
 - Configuration Management
 - Shared Logic
 - Setup MCP Server
 - Try it out
- Part 3: Create MCP Tools
- Part 4: Integrate with MCP Clients

Create Common Utilities

Create a `utils.js` file in the root folder with the following content:

```
import fs from "node:fs/promises";
import jwt from "jsonwebtoken";
import { DataManagementClient } from "@aps_sdk/data-management";
import { IssuesClient } from "@aps_sdk/construction-issues";
import { APS_CLIENT_ID, APS_CLIENT_SECRET, SSA_ID, SSA_KEY_ID, SSA_KEY_PATH } from "config";

const TOKEN_ENDPOINT = "https://developer.api.autodesk.com/authentication/v2/token";

class ServiceAccountAuthenticationProvider {
  constructor(scopes) {
    this._accessToken = null;
    this._expiresAt = 0;
    this._scopes = scopes;
  }

  async getAccessToken() {
    if (!this._accessToken || this._expiresAt < Date.now()) {
      const assertion = await this._createAssertion(this._scopes);
      const { access_token, expires_in } = await this._exchangeAccessToken(assertion);
      this._accessToken = access_token;
      this._expiresAt = Date.now() + expires_in * 1000;
    }
    return this._accessToken;
  }

  async _createAssertion(scopes) {
    const expiresAt = Math.floor(Date.now() / 1000) + 300;
    const payload = { iss: APS_CLIENT_ID, sub: SSA_ID, aud: TOKEN_ENDPOINT, exp: expiresAt };
    const privateKey = await fs.readFile(SSA_KEY_PATH, "utf-8");
    const options = {
      algorithm: "RS256",
      header: { alg: "RS256", kid: SSA_KEY_ID },
      noTimestamp: true
    };
    const assertion = jwt.sign(payload, privateKey, options);
    return assertion;
  }

  async _exchangeAccessToken(assertion) {
    const headers = {
      "Accept": "application/json",
      "Authorization": `Basic ${Buffer.from(`${APS_CLIENT_ID}:${APS_CLIENT_SECRET}`).toString("base64")}`,
      "Content-Type": "application/x-www-form-urlencoded"
    };
    const body = new URLSearchParams({ grant_type: "urn:ietf:params:oauth:grant-type:jwt-bearer", assertion: assertion });
    const response = await fetch(TOKEN_ENDPOINT, { method: "POST", headers, body });
    const json = await response.json();
    return json;
  }
}
```

76

操作手順

1. プロジェクトフォルダの作成、VS Codeでフォルダを開く
2. VS CodeのターミナルでNode.jsの初期化
3. package.json ファイルの“dependencies”の記述
4. ターミナルでパッケージのインストール
5. .envファイルの作成
6. config.jsの作成
7. util.jsの作成
8. tools/index.jsの作成
9. server.jsの作成
10. MCP Inspectorで接続確認

server.jsの作成 : MCP Server 本体

- server.jsを追加して、処理を追加

The image shows a VS Code workspace for creating an MCP server. The left sidebar shows the file explorer with the following structure:

- APS-MCP-...
 - node_modules
 - tools
 - index.js
 - server.js (highlighted with a red box and an orange arrow pointing to the code editor)
 - .env
 - config.js
 - package-lock.json
 - package.json
 - utils.js

The right sidebar shows the "APS MCP Server: Tutorial" with the following sections:

- Overview
- Part 1: Prepare Service Account
- Part 2: Create MCP Server
 - Create Node.js Project
 - Configuration Management

The main editor shows the "Create Server File" section, which instructs to create `server.js` in the root directory. A "Copy to clipboard" button is visible. The code in the editor is as follows:

```
1 import { McpServer } from "@modelcontextprotocol/sdk/server/mcp.js";
2 import { StdioServerTransport } from "@modelcontextprotocol/sdk/server/stdio.js";
3 import * as tools from "./tools/index.js";
4
5 const server = new McpServer({ name: "aps-mcp-server", version: "0.0.1" });
6 for (const [name, { title, description, inputSchema, outputSchema, callback }] of Object.entries(tools)) {
7   server.registerTool(name, { title, description, inputSchema, outputSchema, callback });
8 }
9
10 try {
11   await server.connect(new StdioServerTransport());
12 } catch (err) {
13   console.error("Server error:", err);
14 }
15
```

aps-mcp-server-nodejs

index.js

```
1 コードの生成( Ctrl+I ) または 言語の選択 ( Ctrl+K M )。入力を開始して閉じるか、今後は 表示しない ようにします。
```

tools > index.js

問題 出力 デバッグ コンソール ターミナル ポート AZURE

```
},
"keywords": [],
"author": "",
"license": "ISC"
}

D:\aps-mcp-server-nodejs>npm install

added 153 packages in 50s

38 packages are looking for funding
  run `npm fund` for details

D:\aps-mcp-server-nodejs>
```

アクタイン
タイムライン
XVBA - MACRO LIST
POWER QUERY SDK

autodesk-platform-services.github.io/aps-mcp-server-nodejs/#/part2-mcp-server/index?id=...

APS MCP Server: Tutorial

- > Overview
- > Part 1: Prepare Service Account
- > Part 2: Create MCP Server
 - Create Node.js Project
 - Configuration Management
 - Shared Logic
 - Setup MCP Server
 - Try it out
- > Part 3: Create MCP Tools
- > Part 4: Integrate with MCP Clients

Setup MCP Server

Now let's setup the MCP server itself.

Create Server File

Create `server.js` in the root directory:

```
import { McpServer } from "@modelcontextprotocol/sdk/server/mcp.js";
import { StdioServerTransport } from "@modelcontextprotocol/sdk/server/stdio.js";
import * as tools from "../tools/index.js";

const server = new McpServer({ name: "aps-mcp-server-nodejs", version: "0.0.1" });
for (const [name, { title, description, inputSchema, callback }] of Object.entries(tools)) {
  server.registerTool(name, { title, description, inputSchema }, callback);
}

try {
  await server.connect(new StdioServerTransport());
} catch (err) {
  console.error("Server error:", err);
}
```

This code:

- Creates an MCP server instance
- Iterates through all tools and registers them
- Connects to `stdio` transport for communication with MCP clients

Try it out

Before we start implementing individual MCP tools, let's make sure that our server runs as expected.

Start MCP Inspector

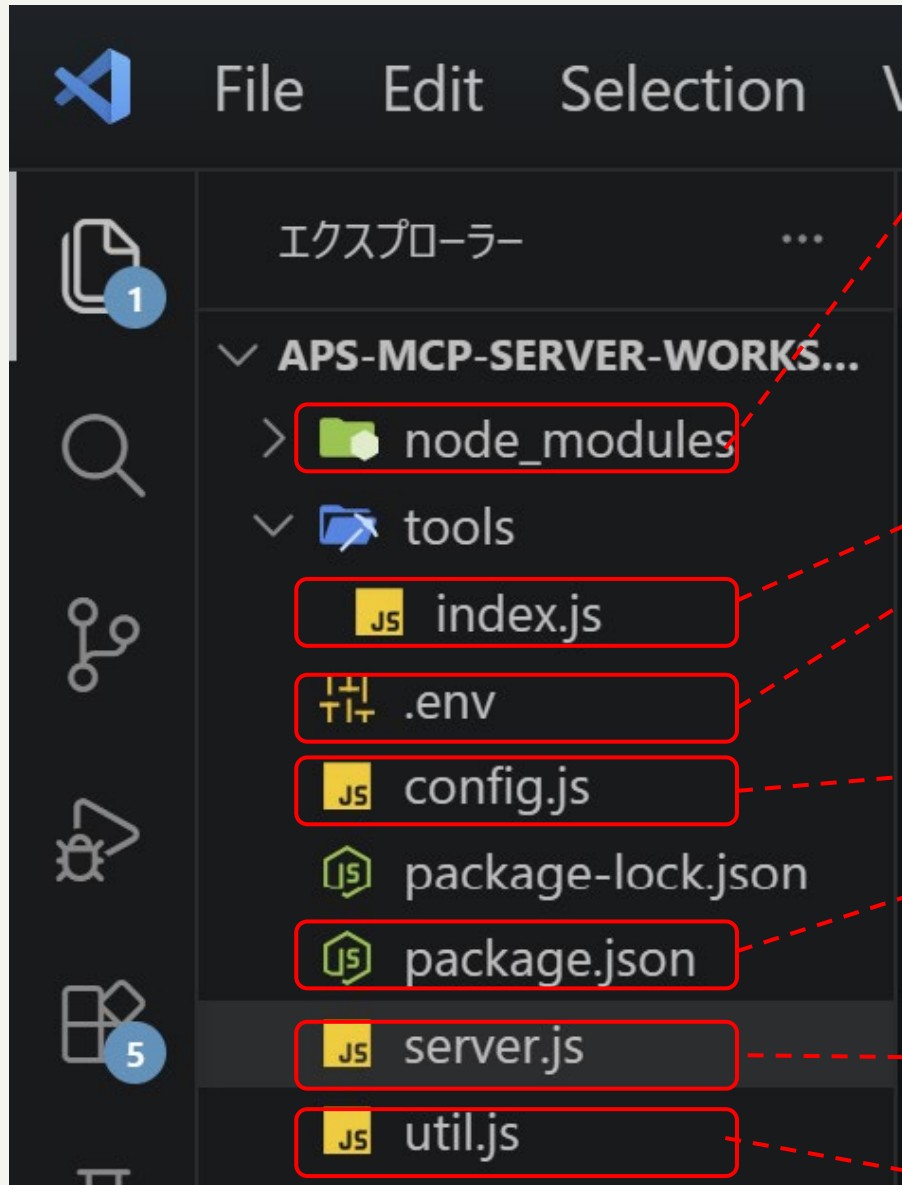
Open the terminal in your project folder, and start the [MCP Inspector](#):

```
npx @modelcontextprotocol/inspector
```

This will start an MCP Inspector server locally, and automatically open it in the browser.

79

完成形のプロジェクト構成



node_modules

インストールしたパッケージ

tools/index.js

MCP サーバで公開するToolの一覧

.env

認証情報等の設定

config.js

設定読み込み

package.json

必要なパッケージの設定 等

server.js

MCP Server 本体

utils.js

APS 認証

操作手順

1. プロジェクトフォルダの作成、VS Codeでフォルダを開く
2. VS CodeのターミナルでNode.jsの初期化
3. package.json ファイルの“dependencies”の記述
4. ターミナルでパッケージのインストール
5. .envファイルの作成
6. config.jsの作成
7. util.jsの作成
8. tools/index.jsの作成
9. server.jsの作成
10. MCP Inspectorで接続確認

MCP Inspector で接続確認

- MCP Inspectorの起動

```
D:\aps-mcp-server-nodejs>npx @modelcontextprotocol/inspector
```

```
D:\aps-mcp-server-nodejs>npx @modelcontextprotocol/inspector
Starting MCP inspector...

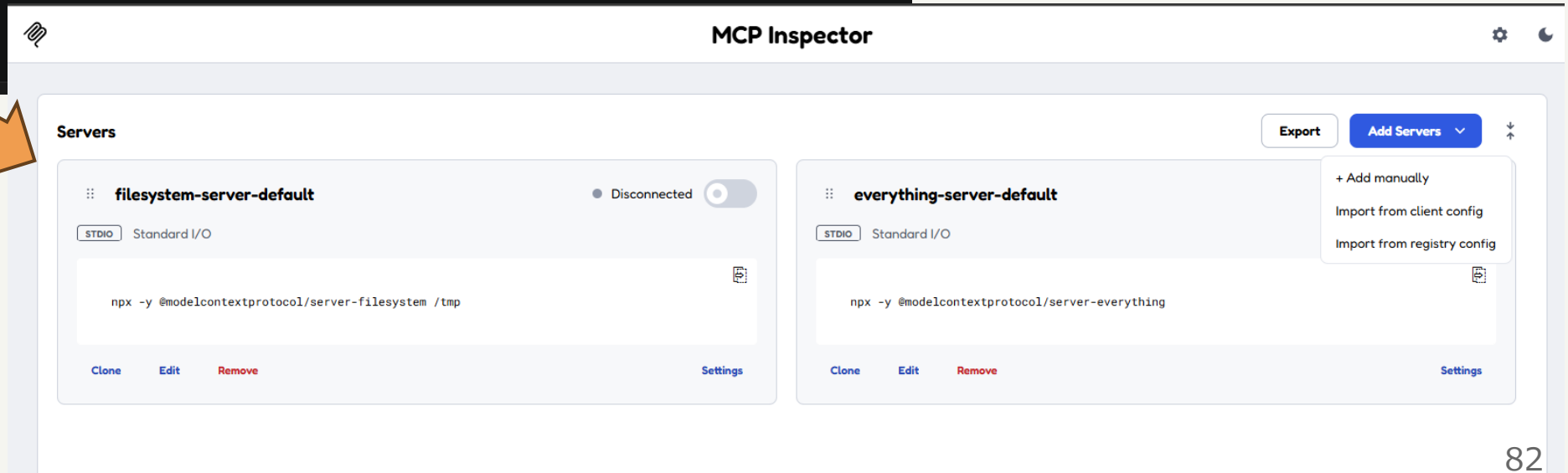
MCP Inspector Web is up and running at:
  http://127.0.0.1:6274?MCP_INSPECTOR_API_TOKEN=c7da6dc23e2a722061d20eb5ffe7718c804321d178ba5544024eaf004b2c8ff2

Sandbox (MCP Apps): http://127.0.0.1:6275/sandbox

Auth token: c7da6dc23e2a722061d20eb5ffe7718c804321d178ba5544024eaf004b2c8ff2

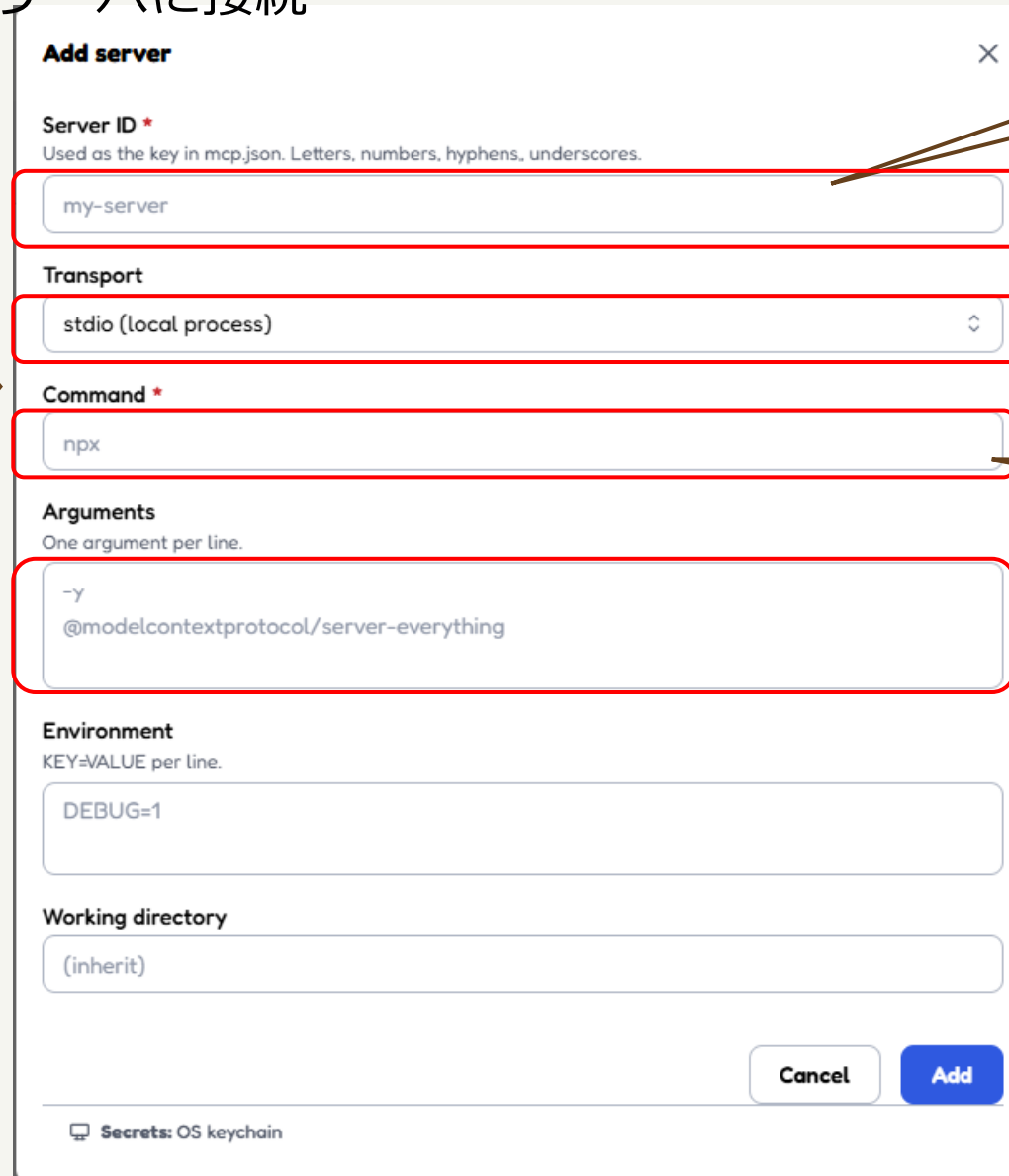
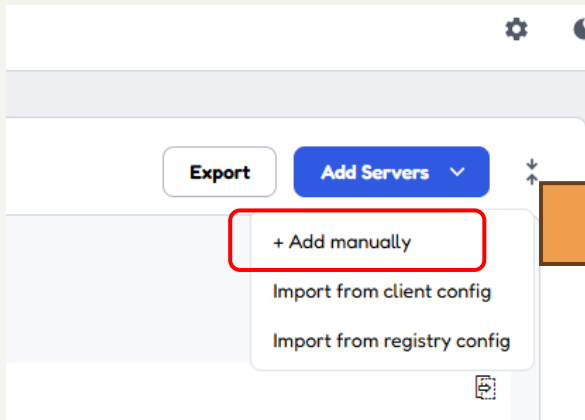
Secrets: OS keychain

Opening browser...
```



MCP Inspector で接続確認

- MCP InspectorからMCPサーバに接続

A screenshot of the 'Add server' dialog box in MCP Inspector. The dialog has a title bar with a close button. It contains several input fields and sections: 'Server ID' with a red asterisk and a note 'Used as the key in mcp.json. Letters, numbers, hyphens, underscores.'; 'Transport' with a dropdown menu; 'Command' with a red asterisk; 'Arguments' with a note 'One argument per line.'; 'Environment' with a note 'KEY=VALUE per line.'; and 'Working directory'. At the bottom are 'Cancel' and 'Add' buttons. A red box highlights the 'Add' button, with a callout bubble pointing to it that says 'Click'. Another red box highlights the 'Server ID' field, with a callout bubble pointing to it that says 'Server ID: aps-mcp-server-nodejs'. A third red box highlights the 'Transport' dropdown, with a callout bubble pointing to it that says 'Transport: STDIO'. A fourth red box highlights the 'Command' field, with a callout bubble pointing to it that says 'Command: node'. A fifth red box highlights the 'Arguments' field, with a callout bubble pointing to it that says 'Arguments: server.js'. A sixth red box highlights the 'Environment' field, which contains 'DEBUG=1'. A seventh red box highlights the 'Working directory' field, which contains '(inherit)'. A small icon and text 'Secrets: OS keychain' are at the bottom left of the dialog.

Server ID:
aps-mcp-server-nodejs

Transport:
STDIO

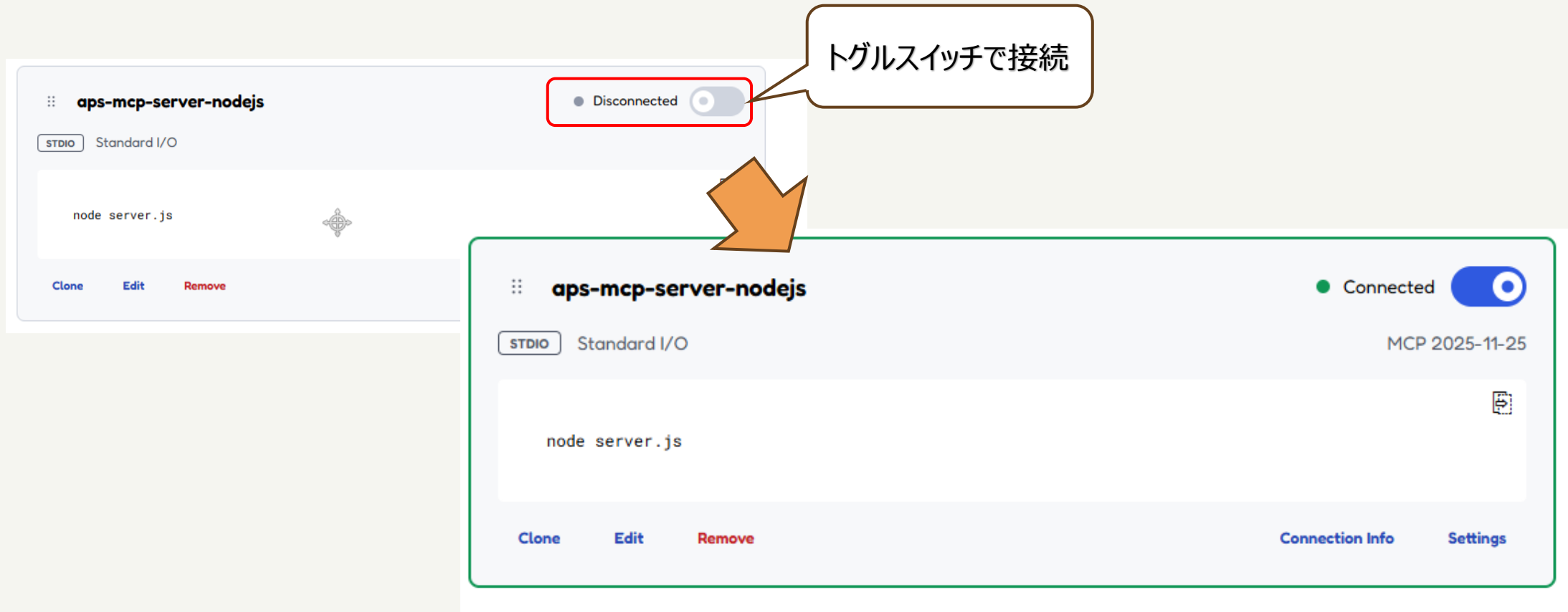
Command:
node

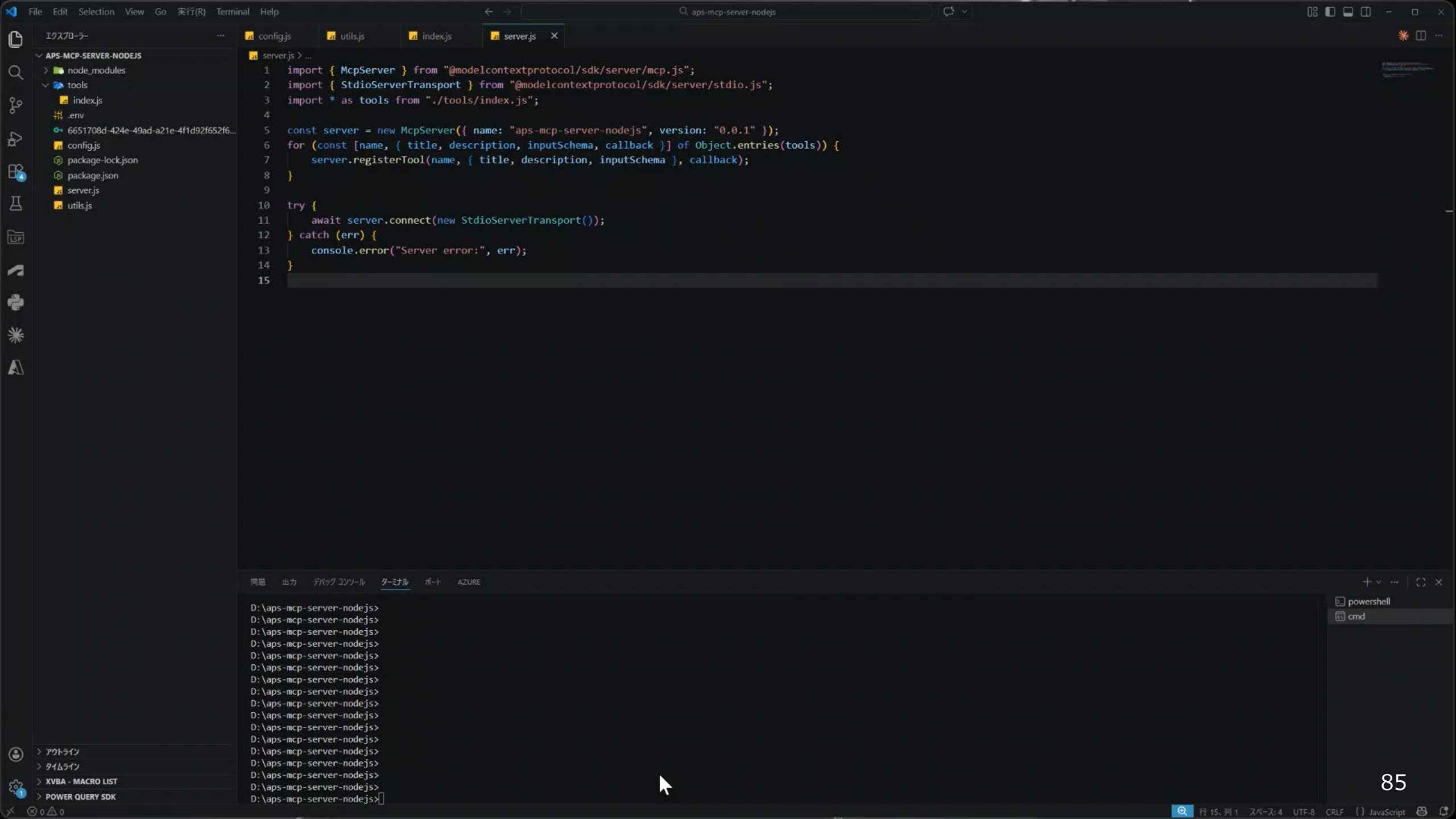
Arguments:
server.js

Click

MCP Inspector で接続確認

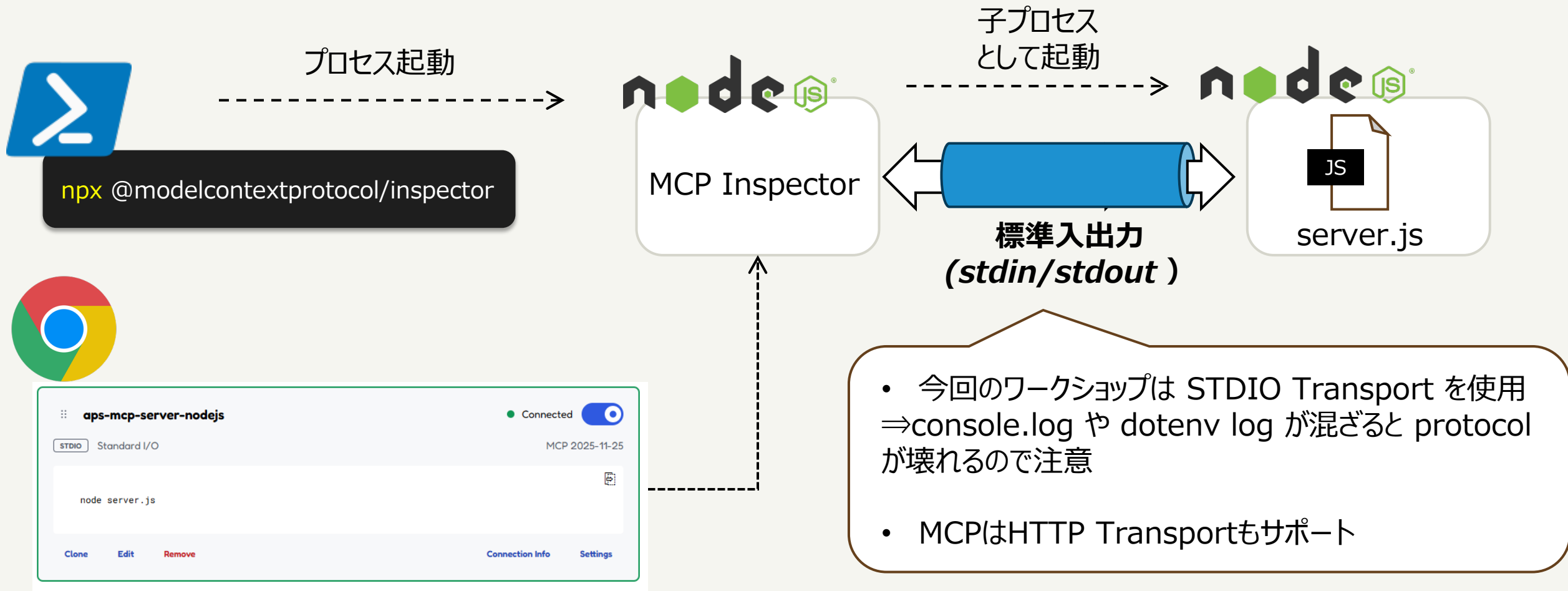
- MCP InspectorからMCPサーバに接続





MCP Inspector と stdio transport

Inspector が server.js を子プロセスとして起動



.vscode/mcp.json : VSCode連携設定

APS MCP Server: Tutorial

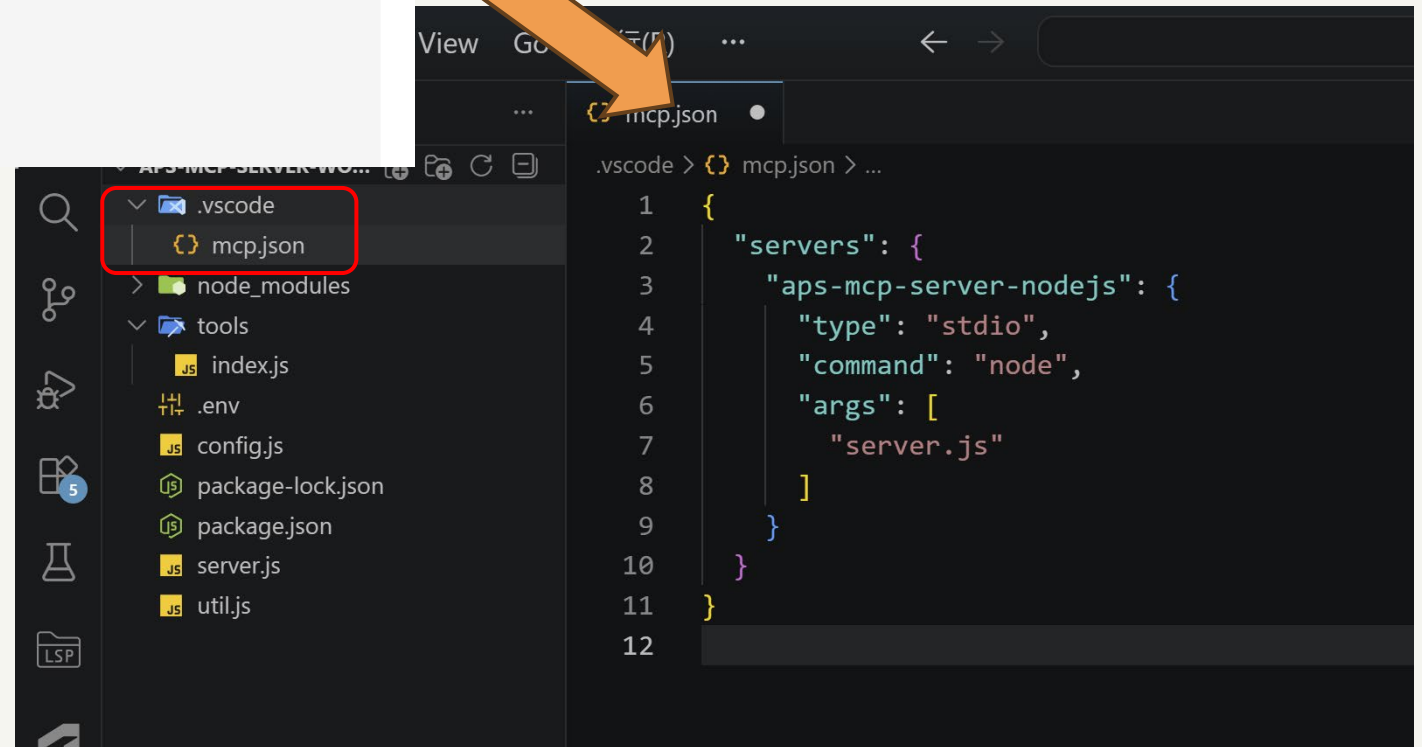
- > Overview
- > Part 1: Prepare Service Account
- ✓ Part 2: Create MCP Server
 - Create Node.js Project
 - Configuration Management
 - Shared Logic
 - Setup MCP Server
 - Try it out
- > Part 3: Create MCP Tools
- > Part 4: Integrate with MCP Clients

test it as you're building it:

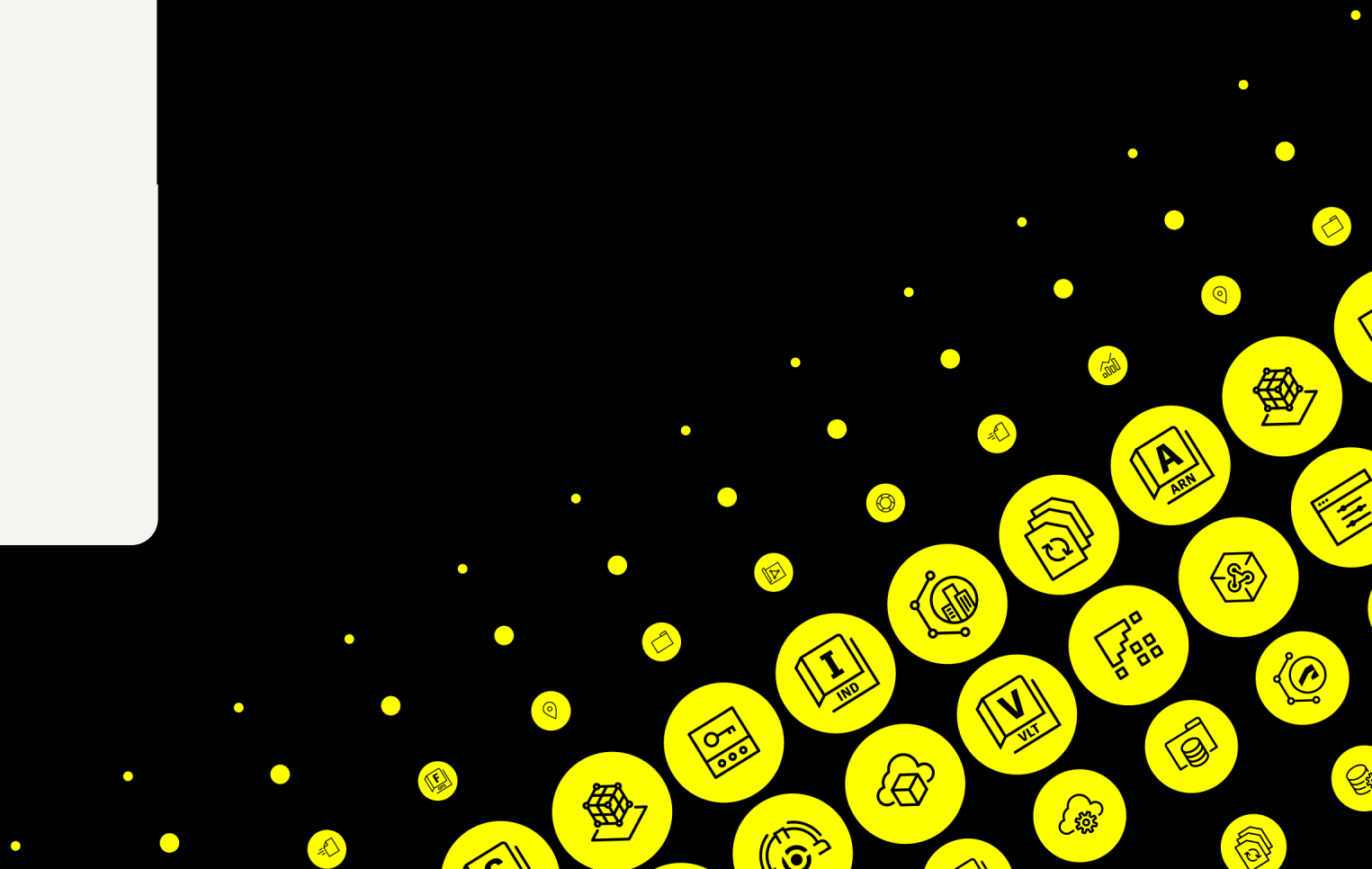
- Make sure you have [enabled MCP servers in Visual Studio Code](#)
- Create `.vscode/mcp.json` file in the root folder of your project, and add the following JSON to it:

```
{
  "servers": {
    "aps-mcp-server-nodejs": {
      "type": "stdio",
      "command": "node",
      "args": [
        "server.js"
      ]
    }
  }
}
```

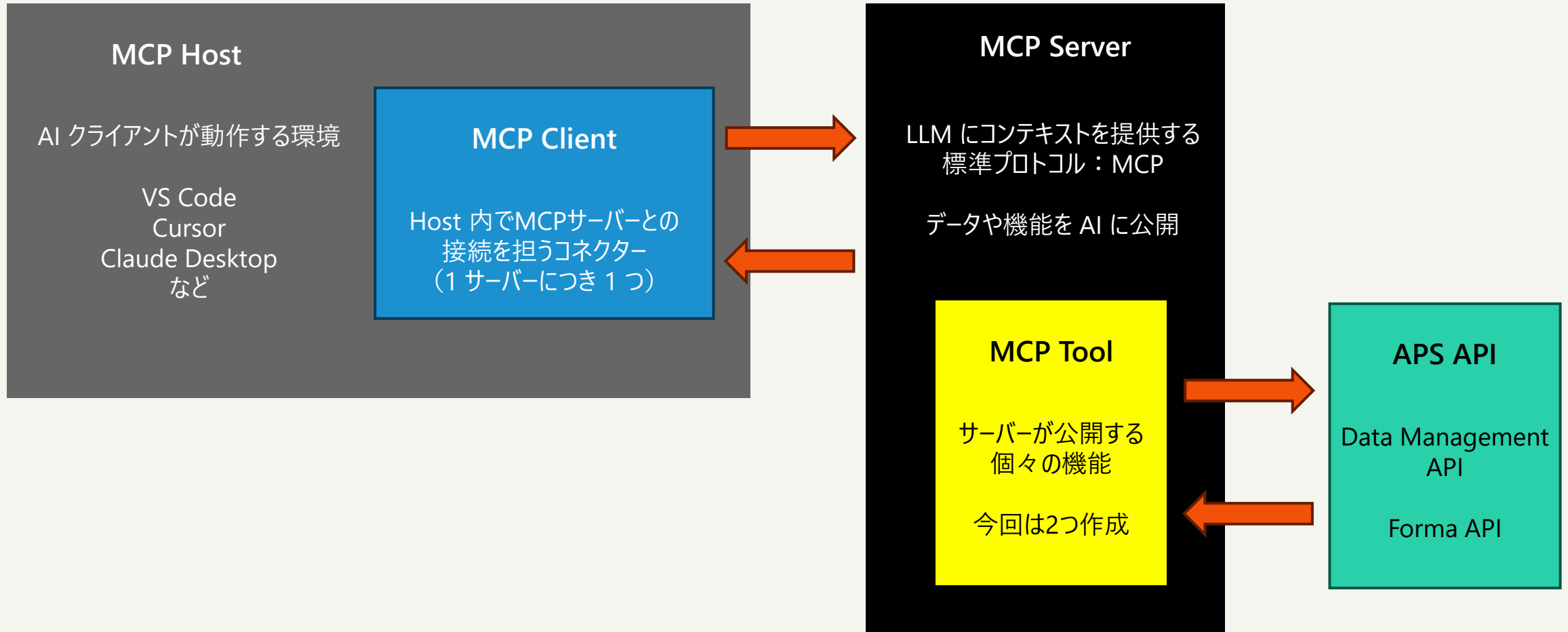
Copy to clipboard



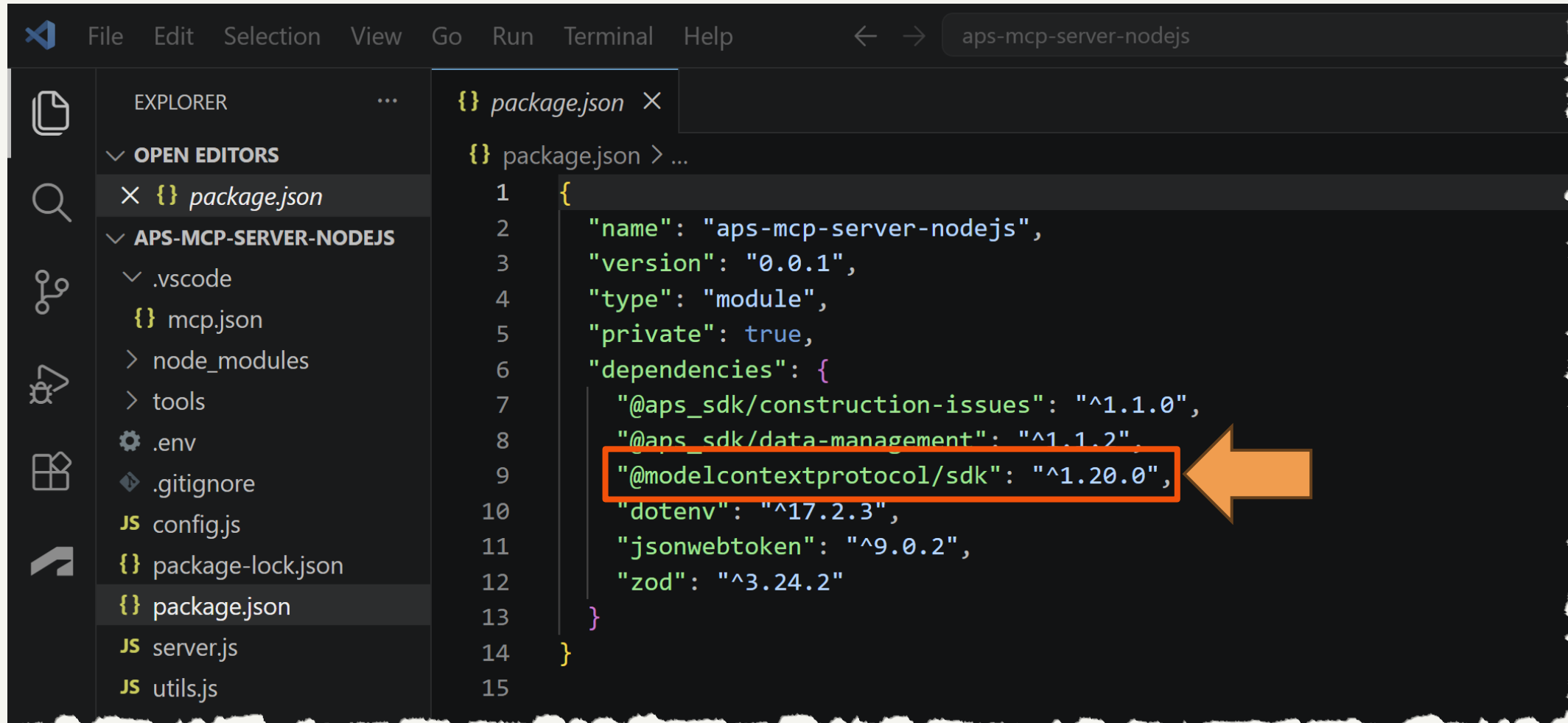
Part 3: MCP ツールの作成



MCP サーバーの構成要素



MCP SDK (旧仕様)

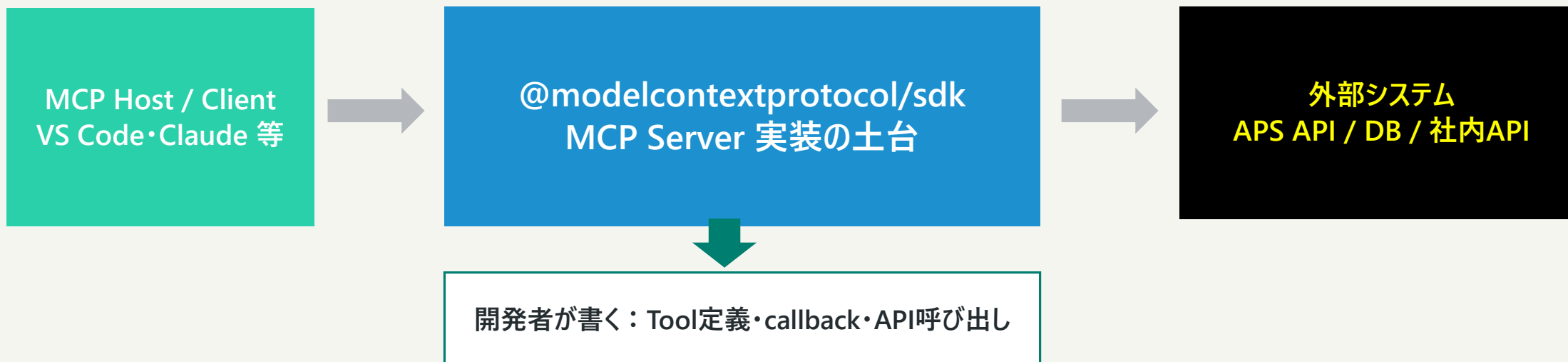


The screenshot shows the Visual Studio Code interface with the Explorer sidebar on the left and the Code editor in the center. The Explorer sidebar shows the file structure of the project 'aps-mcp-server-nodejs', including files like 'package.json', 'server.js', and 'utils.js'. The Code editor displays the contents of 'package.json', which is a JSON object with fields for 'name', 'version', 'type', 'private', 'dependencies', 'dotenv', 'jsonwebtoken', and 'zod'. The 'dependencies' field is a JSON object containing three entries: '@aps_sdk/construction-issues', '@aps_sdk/data-management', and '@modelcontextprotocol/sdk'. The entry for '@modelcontextprotocol/sdk' is highlighted with an orange box, and an orange arrow points to it from the right.

```
{
  "name": "aps-mcp-server-nodejs",
  "version": "0.0.1",
  "type": "module",
  "private": true,
  "dependencies": {
    "@aps_sdk/construction-issues": "^1.1.0",
    "@aps_sdk/data-management": "^1.1.2",
    "@modelcontextprotocol/sdk": "^1.20.0",
    "dotenv": "^17.2.3",
    "jsonwebtoken": "^9.0.2",
    "zod": "^3.24.2"
  },
  "scripts": {
    "start": "node server.js"
  }
}
```

MCP SDK の役割

公式 TypeScript SDK が、MCPサーバーの「配管」を引き受ける



SDKが主に代行すること

- JSON-RPCメッセージ処理とメソッド振り分け
- Toolの公開、呼び出し、スキーマ検証
- stdio / Streamable HTTP などのTransport接続

開発者側に残る責務

- APIアクセスやDB処理などの業務ロジック
- 認証・認可、アクセス制御、セッション設計
- AIに渡す結果の最小化とエラー文言設計

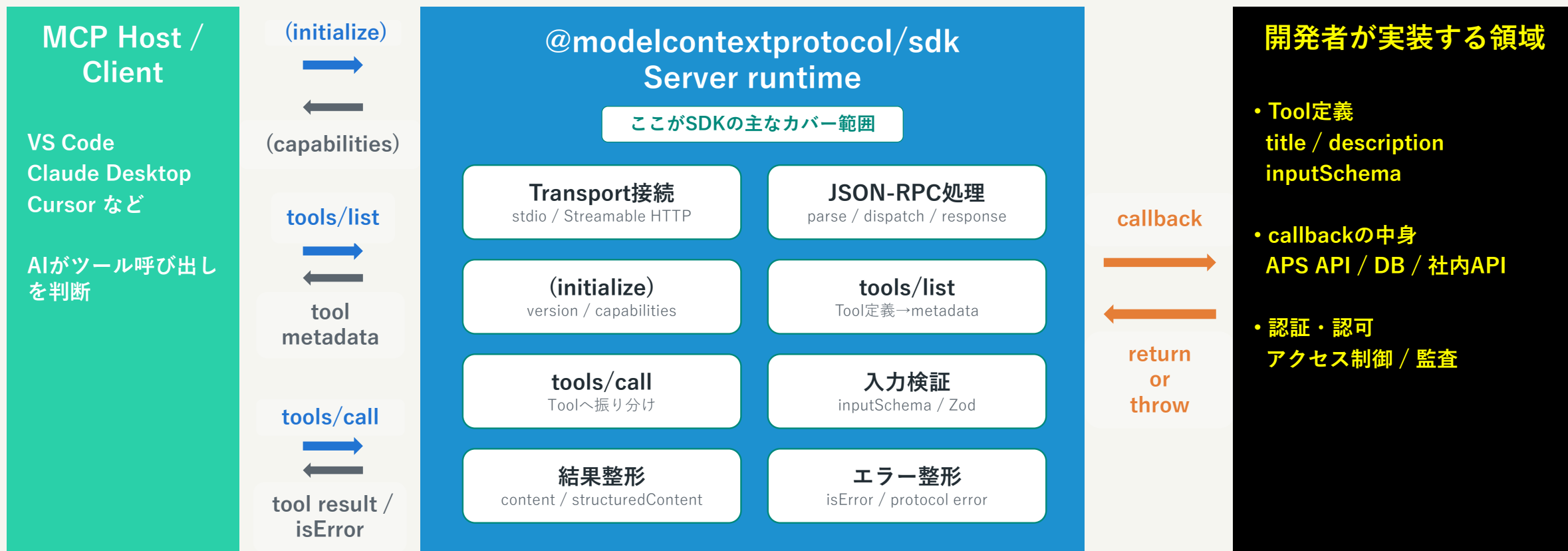


覚えるべき境界線

SDKはプロトコルの面倒を見る。MCPサーバーとして「何をAIに公開するか」は開発者が設計する。

SDK が代行するプロトコル処理

MCP SDK の担当範囲：JSON-RPCの受信・振り分け・応答整形。業務ロジックは開発者の callback に残る。

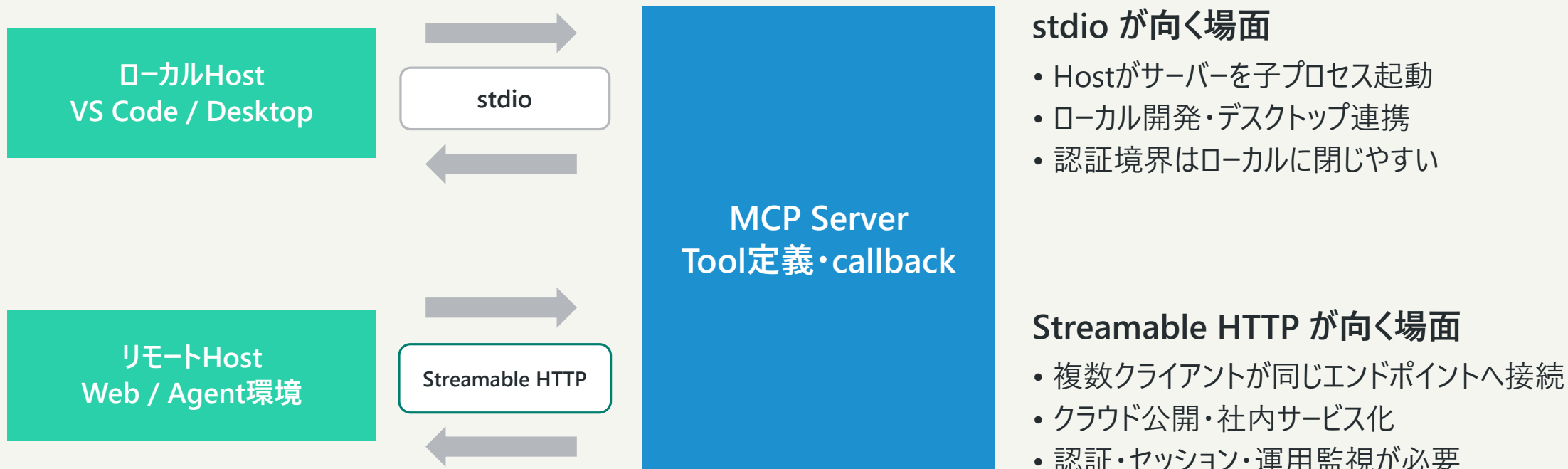


設計上のメリット

tools/list と tools/call を自前で実装しなくてよい。toolの中身は callback に閉じ込め、プロトコル層と業務ロジックを分離できる。

Transport : stdio と Streamable HTTP

tool実装は保ちつつ、接続方式は実行環境に合わせて選ぶ



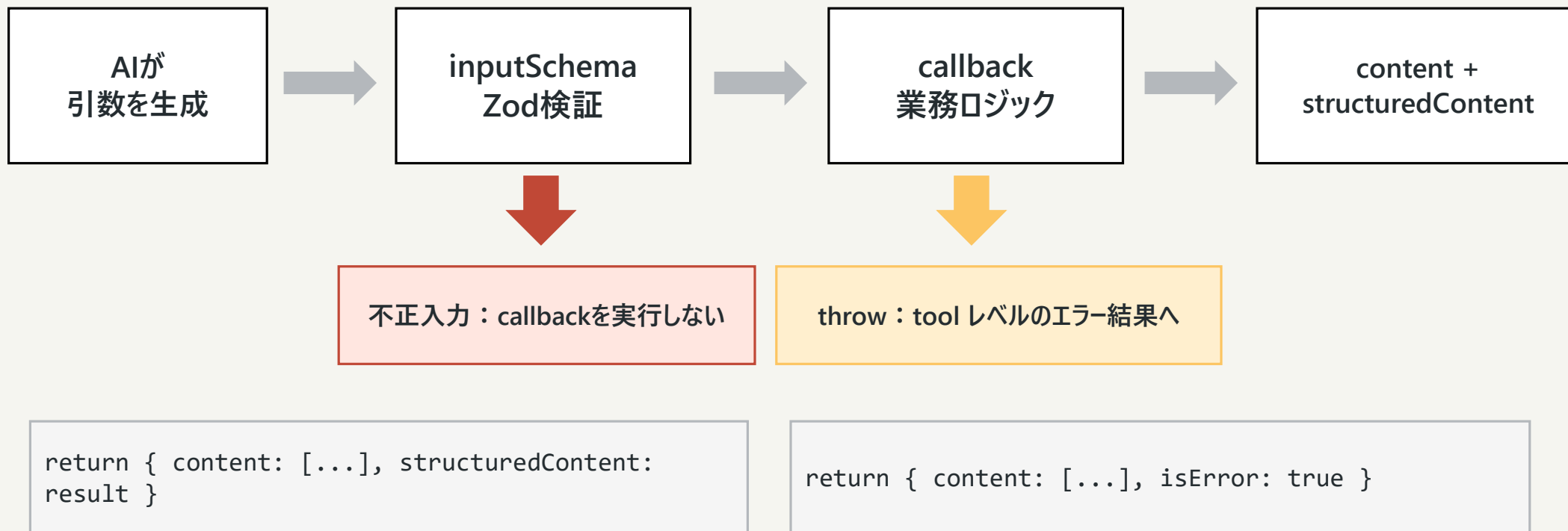
「差し替えるだけ」は半分正しい

⚠ Transportの抽象化によりTool実装は大きく変えずに済む。ただしHTTP公開では、HTTPサーバー、認証・認可、セッション管理、Origin/Host検証を別途設計する。

補足：旧式の HTTP + SSE Transport は後方互換用。新規リモート実装では Streamable HTTP を前提にする。

tool 実装：入力・出力・エラー処理を SDK に任せる

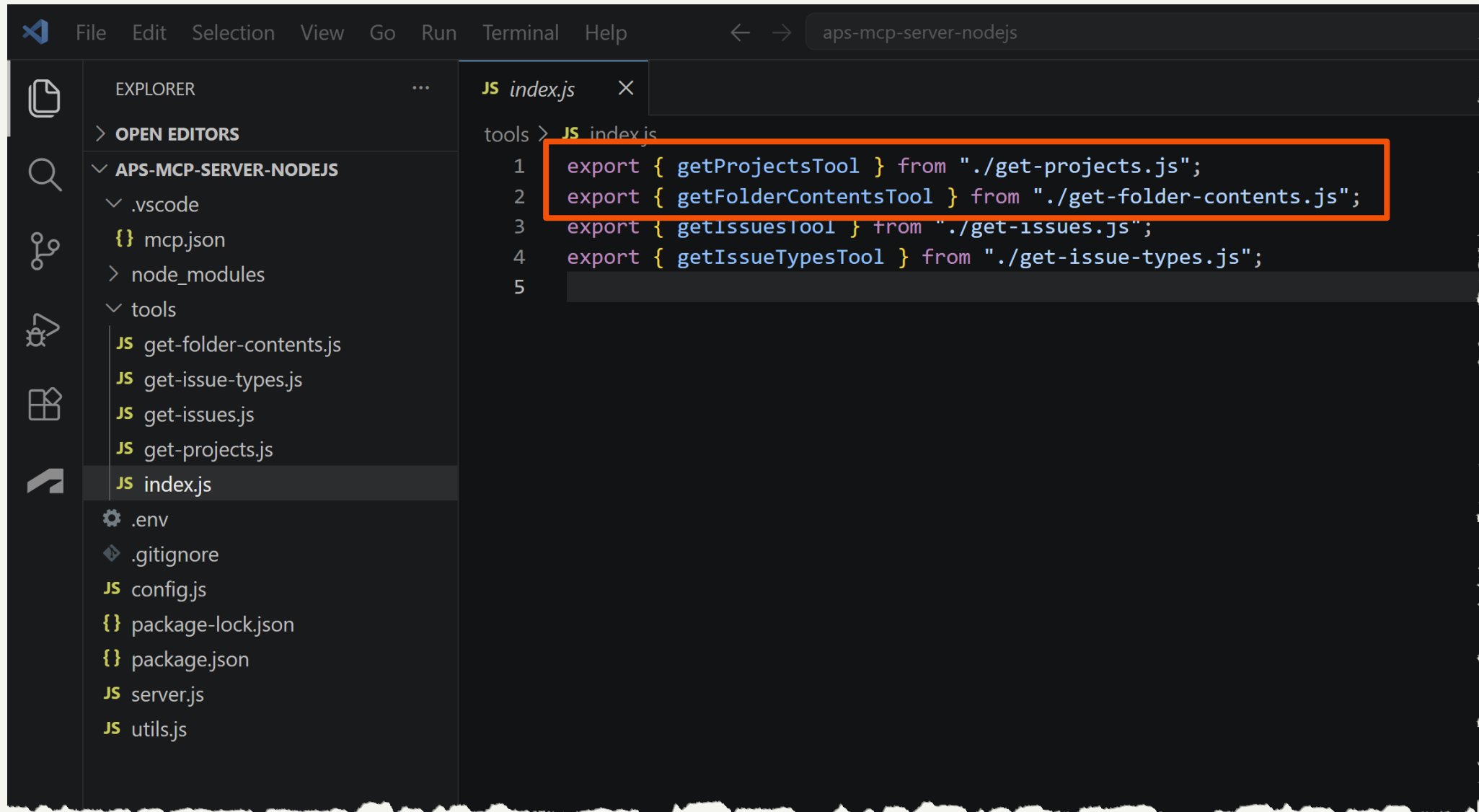
Zodスキーマで呼び出し前に検証し、結果はcontent / structuredContentで返す



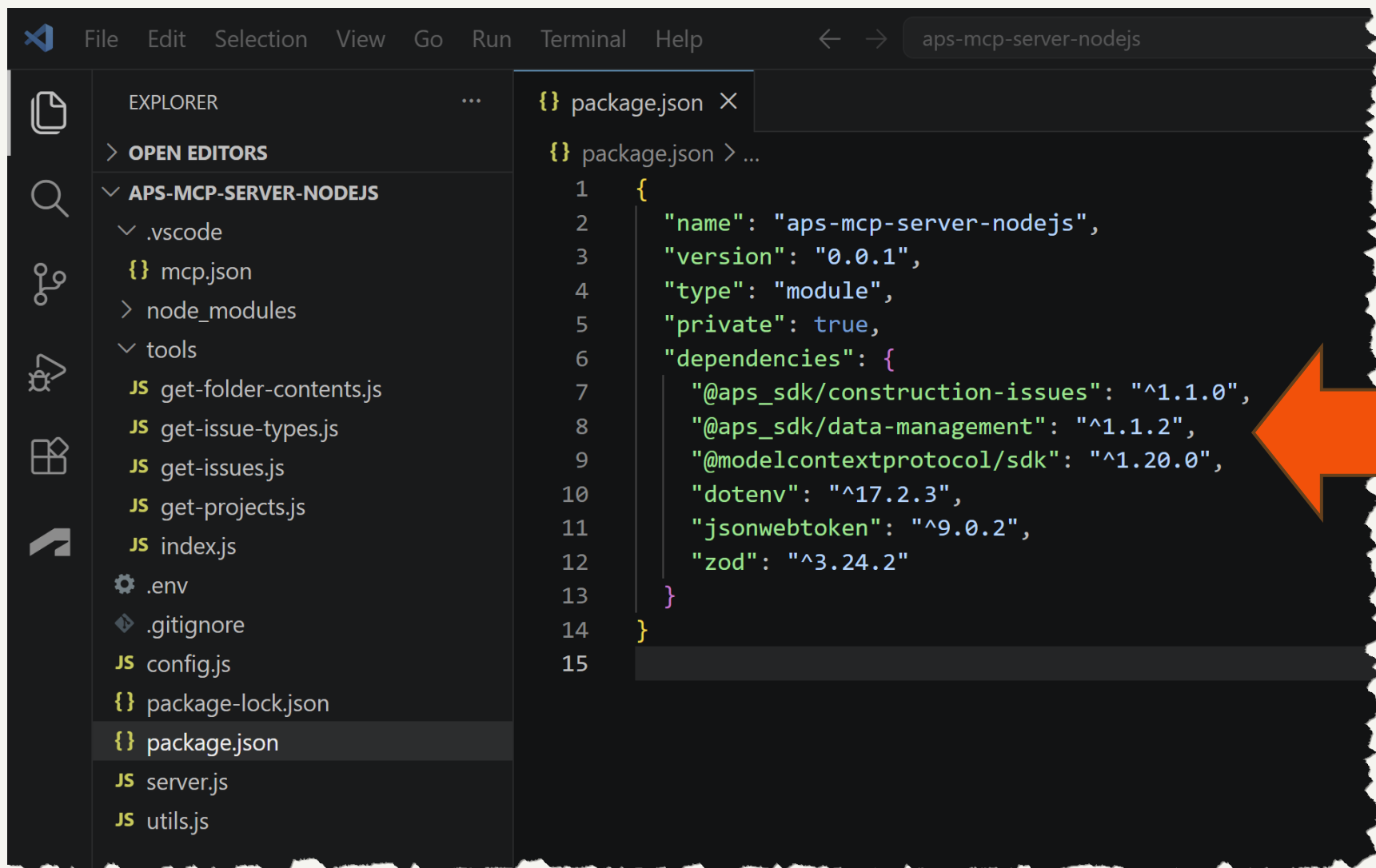
エラーは2層で考える

throw は tool レベルのエラー結果 (isError: true) として返す。不正JSON-RPCや未定義メソッドはプロトコルエラーとして扱う。

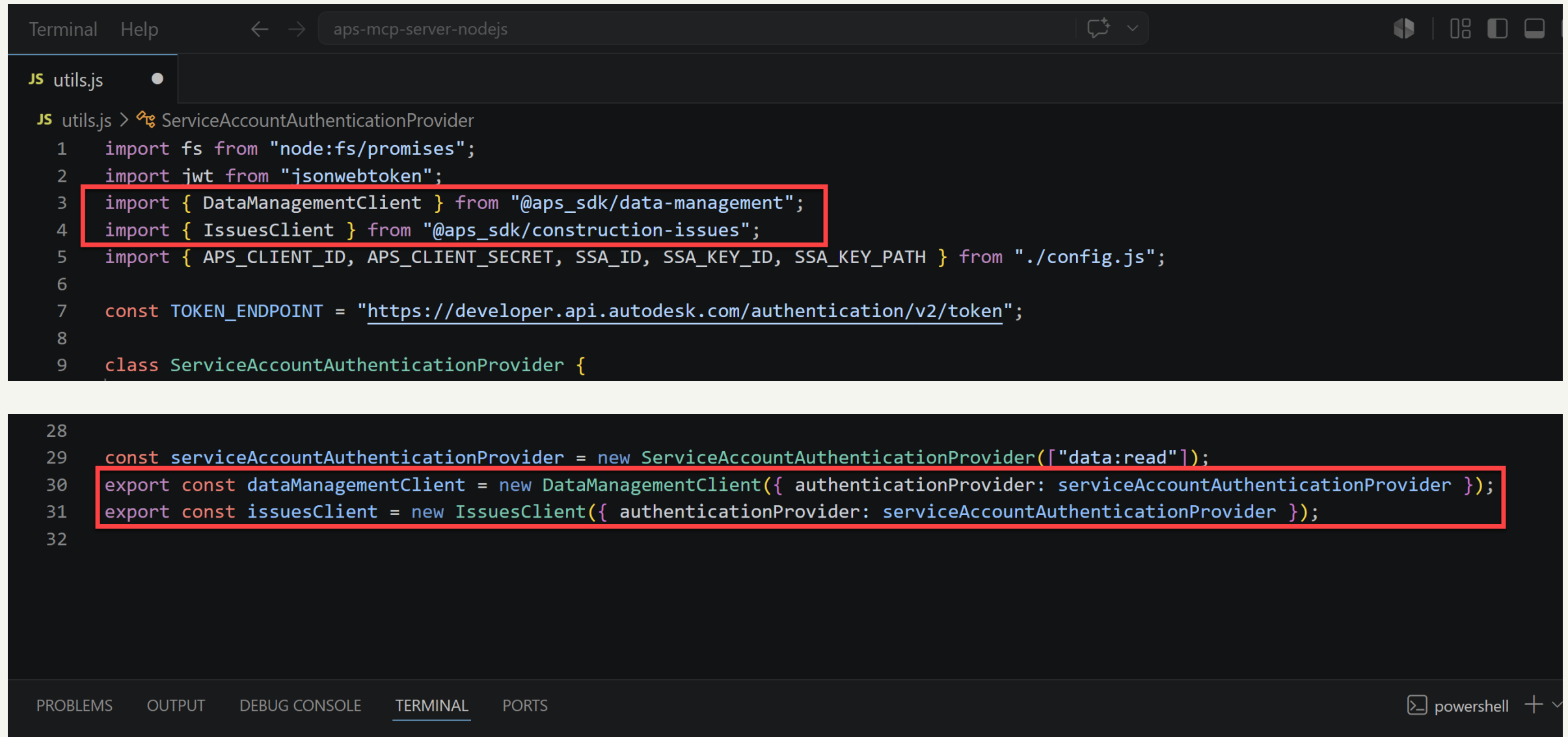
今回作成する MCP ツール



APS SDK をモジュール単位でインストール



utils.js で SDKの読み込み・エクスポート



```
Terminal Help  ← →  aps-mcp-server-nodejs  [icons]

JS utils.js ●

JS utils.js > ServiceAccountAuthenticationProvider
1  import fs from "node:fs/promises";
2  import jwt from "jsonwebtoken";
3  import { DataManagementClient } from "@aps_sdk/data-management";
4  import { IssuesClient } from "@aps_sdk/construction-issues";
5  import { APS_CLIENT_ID, APS_CLIENT_SECRET, SSA_ID, SSA_KEY_ID, SSA_KEY_PATH } from "./config.js";
6
7  const TOKEN_ENDPOINT = "https://developer.api.autodesk.com/authentication/v2/token";
8
9  class ServiceAccountAuthenticationProvider {
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29  const serviceAccountAuthenticationProvider = new ServiceAccountAuthenticationProvider(["data:read"]);
30  export const dataManagementClient = new DataManagementClient({ authenticationProvider: serviceAccountAuthenticationProvider });
31  export const issuesClient = new IssuesClient({ authenticationProvider: serviceAccountAuthenticationProvider });
32

PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS  [icons] powershell + ▾
```

MCP ツールの基本構造

各ツールは以下のフィールドを持つオブジェクトとして実装します。

title	ツールのタイトル
description	ツールの説明文。 AI エージェント が「いつ／どう使うか」を判断する材料
inputSchema	入力パラメータのスキーマ（zod で定義）
callback	実行時に呼ばれる非同期関数。{content, structuredContent} を返す



ツール 1 つ = ファイル 1 つの方針で実装

tools/get-projects.js のように 1 ツール = 1 ファイル。最後に tools/index.js で集約。

description は「AI への説明書」

AI は description を読んで、「このツールをいつ使うべきか」を判断します。
ここの書き方次第で AI の振る舞いが大きく変わります。

効果的な description の書き方

- 何を返すかを最初の 1 行で明示する (What)
- どんな入力が必要かを明示する (Required parameters)
- 戻り値の構造を伝える (Returns ...)
- 似たツールとの違いを書く (vs ... では ...)
- ユースケースや例を含めるとさらに良い



曖昧な description は AI の誤呼び出しを招く

「データを取得する」だけでは AI が他ツールと混同する。「ACC プロジェクトの Issue 一覧を取得する」のように具体的に書く。

zod スキーマ入門

zod は TypeScript ファーストのスキーマ検証ライブラリ。
MCP では「ツール入力の型」を定義するために使います。

zod 例

```
import { z } from "zod";

const schema = {
  projectId: z.string().nonempty(),      // 必須・空文字不可
  folderId: z.string().optional(),       // 任意
  limit: z.number().int().min(1).max(100) // 整数 1..100
};
```

MCP におけるメリット

- AI クライアントに必須／任意・型情報が伝わる → 正しい呼び出しが促される
- 実行時に入力を検証 → 不正な値で API を叩かない

zod スキーマの応用

基本の string / number に加え、選択肢・既定値・配列・説明を足すと、AI がより正確に呼び出せます。

応用例

```
const schema = {  
  type: z.enum(["issue", "rfi"]),           // 選択肢を限定  
  limit: z.number().int().min(1).max(100)  
    .default(20),                          // 既定値  
  tags: z.array(z.string()).optional(),    // 文字列の配列  
  projectId: z.string()  
    .describe("Forma プロジェクト ID"), // AI 向け説明  
};
```

describe は「パラメータ単位の説明書」

- enum / default で入力のブレを抑え、AI の誤りを防ぐ
- .describe() の一文が、AI に「何を入れるか」を伝える

content と structuredContent

ツールの callback は 2 種類のフォーマットで結果を返すのが推奨。
新旧の MCP クライアントに対応するため、両方を含めるのがベストプラクティス。

content (必須)

```
[{ type: "text", text: "..."}]
```

旧来の MCP 仕様で必須のフィールド。
JSON.stringify で文字列化した結果を入れる。
構造化対応していないクライアントはこちらを読む。
Human Readable な要約。

structuredContent (推奨)

```
{ accounts: [...] } など
```

新しい MCP 仕様の構造化データフィールド。
JSON オブジェクトをそのまま返せる。
新しい AI クライアントはこちらを優先的に解釈する。
MCP クライアント向けのデータ。



JSON は必要最小限のフィールドだけに絞る (コンテキスト節約)

ツールのエラーハンドリング

callback の中でエラーは throw するだけ。MCP SDK が受け取り、AI に伝わる形へ変換します。

例

```
callback: async ({ projectId }) => {  
  const res = await dataManagementClient  
    .getProject(projectId)  
    .catch(e => {  
      throw new Error(`Failed: ${e.message}`);  
    });  
  return { content: [...], structuredContent: res };  
}
```

エラーメッセージも AI への情報

- 「何が・なぜ失敗したか」を具体的に書く → AI がリトライや別手段を判断できる
- content に自前で埋め込まず throw に任せる（整形は SDK が担当）

Forma Hub/Project の確認（作成）

AUTODESK Forma

Hub Admin ▾

ryuji-adsk-us-hub ▾

?

小龍 ▾

プロジェクト

メンバー

会社

役割

プロジェクト テンプ...

サブスクリプション

ライブラリ

アプリ

製品とツール

カスタム統合

BIM 360 管理者

プロジェクト

アクティブ アーカイブ済み

+ プロジェクトを作成 ⓘ

🔍 プロジェクト名で検索

🔼

名前	番号	プラットフォーム	メンバー	会社	ステータス	使用されたテン	⚙️
テストプロジェクト1		🏠	3	1	アクティブ	-	⋮
Ryuji Test Project for AEC DM		🏠	1	1	アクティブ	-	⋮
Sample Project		B	1	1	アクティブ	-	⋮
Construction : Sample Project - Seapor...		🏠	1	1	アクティブ	-	⋮

Forma フォルダ/ファイルの確認（作成）

AUTODESK Forma

Data Management

テストプロジェクト1

?

小龍

ファイル

仕様

レビュー

ファイル転送

指摘事項

フォーマボード

ファイル

フォルダ

パッケージ

削除された項目

設定

プロジェクト ファイル

テストフォルダ1

ファイルをアップロード

書き出し

検索とフィルタ

設定

☐	名前 ↑	注記	バージョン	インジケータ	マークア...	指摘事項	サイズ	最終更新日	設定
☐	Apartment.rvt		V1			⊙	36.9 MB	2026年6月4日	⋮

Get Projects ツール（概要）

サービスアカウントがアクセス可能な Forma アカウントとプロジェクトの一覧を返すツール。

APS Data Management API

- GET /project/v1/hubs → アカウント一覧
- GET /project/v1/hubs/{hub}/projects → 各アカウントのプロジェクト一覧

入力パラメータ

- なし（最もシンプル。AI がまず最初に呼ぶ「ディスカバリ」用途）

Get Projects コード

tools/get-projects.js

```
import { dataManagementClient } from "../utils.js";

export const getProjectsTool = {
  title: "Get Accounts & Projects",
  description: `
    Retrieves all Autodesk Construction Cloud (ACC) accounts
    and their associated projects accessible to the configured
    service account.
    Returns a structured list of accounts (with account IDs and
    names) and associated projects (with project IDs and names).
  `,
  inputSchema: {},
  callback: async () => {
    次ページ
  }
};
```

```

callback: async () => {
  const accounts = await dataManagementClient.getHubs()
    .then(res => res.data || []);
  for (const account of accounts) {
    account.projects = await dataManagementClient
      .getHubProjects(account.id)
      .then(res => res.data || []);
  }
  const result = {
    accounts: accounts.map(a => ({
      id: a.id,
      name: a.attributes.name,
      projects: (a.projects || []).map(p => ({
        id: p.id, name: p.attributes.name
      })))
  }));
};
return {
  content: [{ type: "text", text: JSON.stringify(result) }],
  structuredContent: result
};
}

```

Get Projects の中身

- `dataManagementClient.getHubs()` : 認証済み API クライアントでハブ一覧を取得
- ハブ = Forma アカウントと 1:1 対応。BIM 360 のチームも含まれる場合あり
- for ループで各ハブのプロジェクト一覧を取得
 - ハブ数は通常 1～数個なので問題なし
- id と name に絞って整形（attributes は SDK 由来の構造）
- result を `JSON.stringify` して content に、オブジェクトとして `structuredContent` に



AI 視点での意義

ユーザーが「アクセスできるプロジェクトは？」と聞いたら、AI はまずこのツールを呼ぶ。
返ってきた ID を別ツール（Get Folder Contents 等）に渡せるよう、構造化して返す。

MCPツールの登録

① tools/index.js に export 文を追加

tools/index.js

```
export { getProjectsTool } from "../get-projects.js";
```

② （任意）VS Code でテストする例

- .vscode/mcp.json の[Start]ラベルでサーバー起動
- Copilot のチャット（Ctrl+Shift+I）を開く
- 「私がアクセスできるプロジェクトをリストアップしてください。」と質問
- AI が getProjectsTool を呼び、ACC のプロジェクト一覧を答える（適宜、許可が必要）



MCP Inspector からも同様にテスト可能

Get Folder Contents ツール（概要）

Forma プロジェクトのフォルダ内容（サブフォルダ・ファイル）を返すツール。
folderId 未指定ならトップレベル、指定ありならそのフォルダの中身を返す。

APS Data Management API（条件分岐）

- folderId なし → GET /project/v1/hubs/{hub}/projects/{prj}/topFolders
- folderId あり → GET /data/v1/projects/{prj}/folders/{folder}/contents

入力パラメータ

- accountId（必須）／projectId（必須）／folderId（任意）

Get Folder Contents コード

tools/get-folder-contents.js

```
import { z } from "zod";
import { dataManagementClient } from "../utils.js";

export const getFolderContentsTool = {
  title: "Get Folder Contents",
  description: `
    Retrieves the contents of a folder within an ACC project.
    Requires accountId and projectId parameters.
    If no folderId is provided, returns the top-level folders.
    Returns the list of folders and files with their IDs and names.
  `,
  inputSchema: {
    accountId: z.string().nonempty(),
    projectId: z.string().nonempty(),
    folderId: z.string().optional()
  },
  callback: async ({ accountId, projectId, folderId }) => {
    次ページ
  }
};
```

AIエージェントは、このツールを呼び出す際にアカウントIDとプロジェクトIDを提供する必要があることを認識し、オプションでフォルダIDを提供することもできます。

Get Folder Contents コード

```
callback: async ({ accountId, projectId, folderId }) => {
  const contents = folderId
    ? await dataManagementClient
      .getFolderContents(projectId, folderId)
      .then(res => res.data || [])
    : await dataManagementClient
      .getProjectTopFolders(accountId, projectId)
      .then(res => res.data || []);
  const result = {
    contents: contents.map(item => ({
      id: item.id, type: item.type,
      name: item.attributes.displayName
    })))
  };
  return {
    content: [{ type: "text", text: JSON.stringify(result) }],
    structuredContent: result
  };
}
```

Get Folder Contents の中身

- folderId の有無で条件分岐
 - 未指定 → getProjectTopFolders : トップ階層を取得 (accountId が必要)
 - 指定あり → getFolderContents : そのフォルダの中身を取得
- 戻り値の item.type で folders / items の区別ができる
- items (ファイル) は別途バージョン詳細を取りたいケースもある
- 本ツールは「一覧」に特化、深掘りは AI が再帰的に呼び出す形



AI による再帰探索

AI に「全フォルダの中身を取って」と頼むと、トップから順に当ツールを再帰的に呼んでツリー全体を辿る。AI 側のループ能力を活用するため、ツール自体は単純に保つ。

MCPツールの登録

tools/index.js への追加

```
export { getFolderContentsTool } from "./get-folder-contents.js";
```

動作確認プロンプト例

- 「プロジェクト XYZ の全ての *.rvt ファイルを列挙して」
 - AI は Get Projects でプロジェクト ID を取得 → Get Folder Contents で再帰検索
- 「『03_Drawings』フォルダの内容を見せて」
 - AI はフォルダ名から ID を見つけ、その内容を取得



AI の挙動を観察するコツ

Inspector では「invoked tools」のログを見ると、AI がどう判断したかが分かる。
想定外の呼び出しがあれば description を見直すヒントになる。

tools/index.js の最終形

tools/index.js

```
export { getProjectsTool } from "./get-projects.js";  
export { getFolderContentsTool } from "./get-folder-contents.js";  
// export { getIssuesTool } from "./get-issues.js";  
// export { getIssueTypesTool } from "./get-issue-types.js";
```

✓ 2 つのツールがすべて登録されている状態



MCP サーバー完成！

server.js は tools/index.js をそのまま読み込むので、ここに 1 行 export を追加するだけで AI クライアントから新しいツールが見えるようになる。
今後ツールを増やすときも同じパターン。

MCP ツール設計のベストプラクティス

- 1 ツール 1 責務：複合的な操作は AI が複数ツールを組み合わせる
- description は AI 視点で書く：いつ／どう呼ぶかを明確に
- 戻り値は最小限：コンテキストウィンドウを圧迫しない
- ID と名前は必ず返す：AI が次のツール呼び出しに使う
- エラーは throw する：MCP SDK が AI にエラーを伝えてくれる
- 副作用のあるツールは慎重に：書き込み系は実装前に必要性を再考



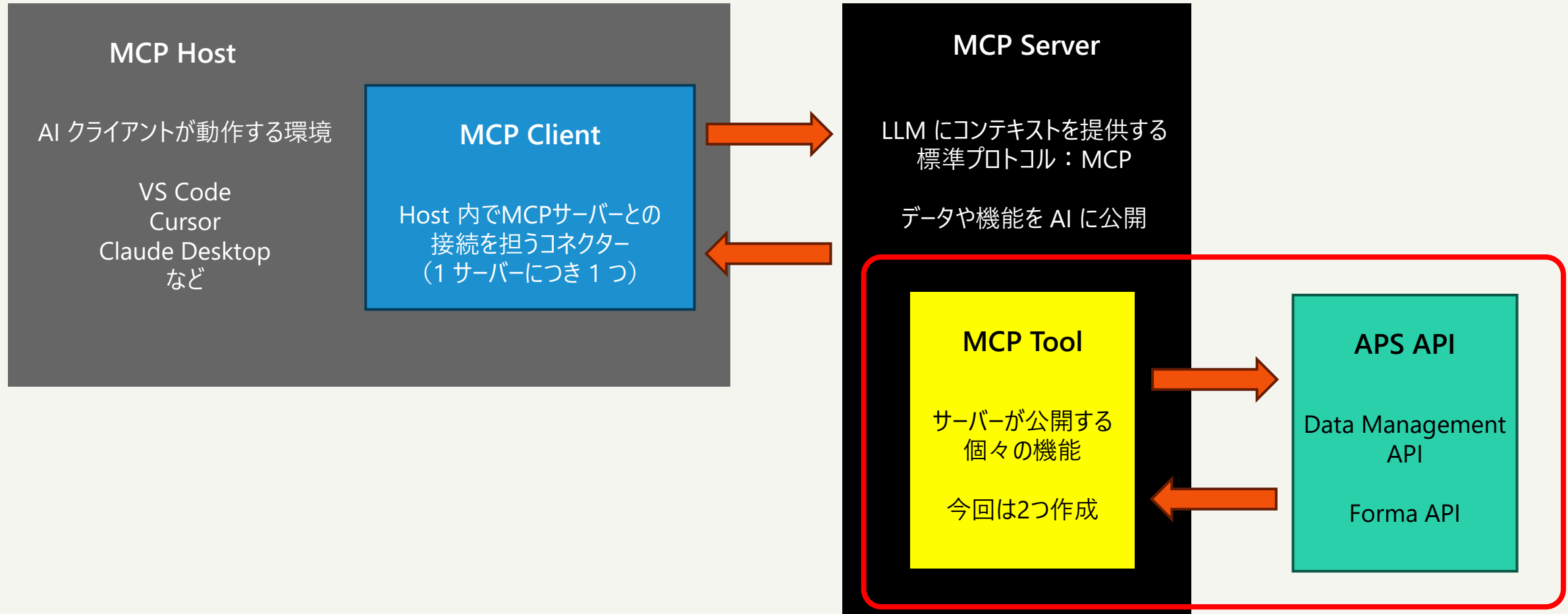
AI フレンドリーな API 設計 = 人間にも親切な API 設計

明示的な命名、最小限の入出力、独立した責務、明快な失敗メッセージ。
古典的な良い API デザインの原則がそのまま当てはまる。

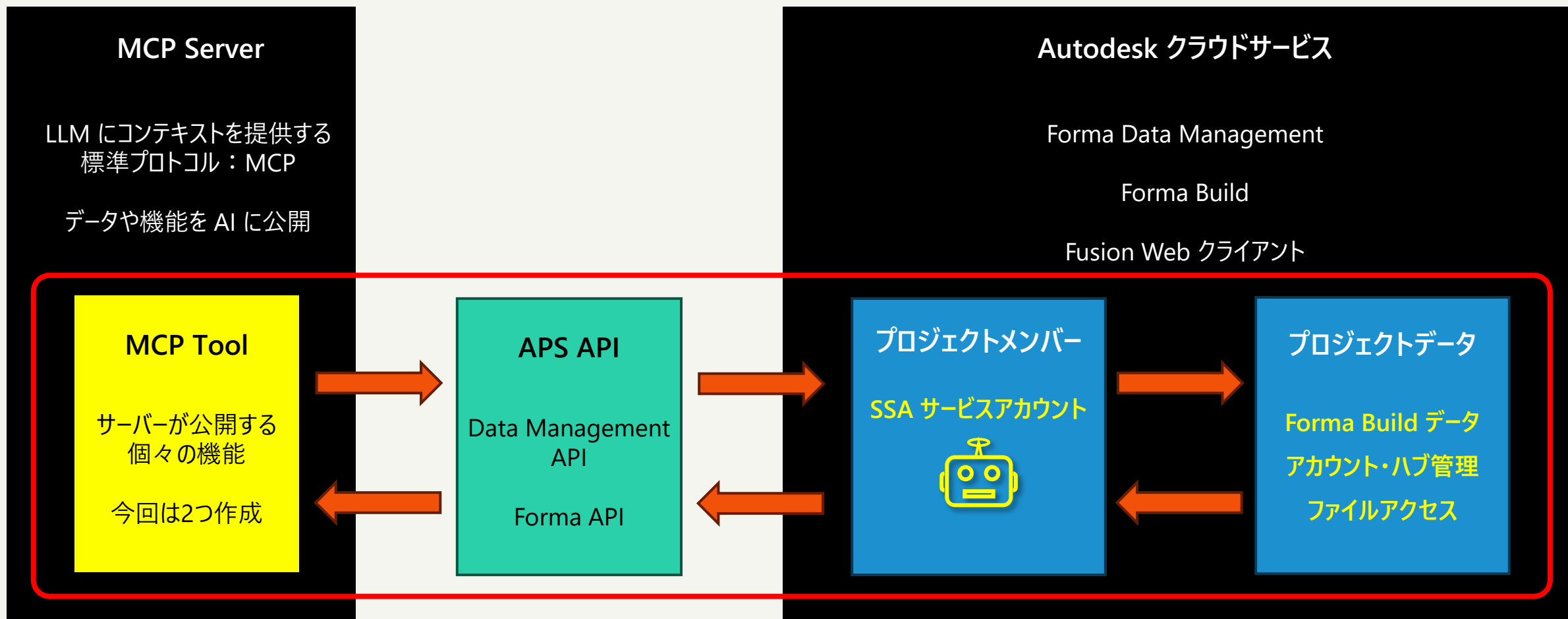
SSA ロボット アカウントの追加

Forma / Fusion

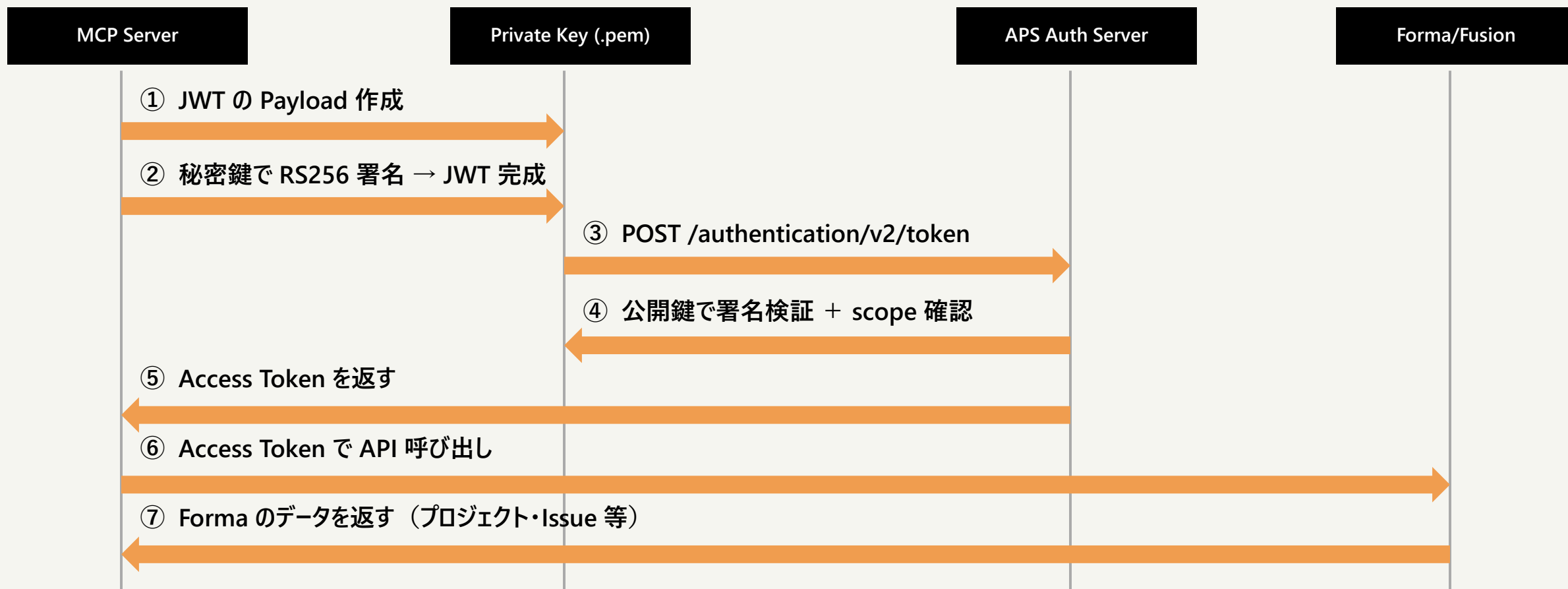
MCP サーバーの構成要素



Autodesk クラウドサービスとの接続



SSA 認証フロー（シーケンス）



Access Token はキャッシュして使い回す

expires_in（通常 1 時間）まで再利用 → 期限が切れたら再度 JWT で交換。

JWT (JSON Web Token)

- SSA 認証では「秘密鍵で署名した JWT」を APS に提示してアクセストークンを得る。
- JWT は 3 つの部分が「.」でつながった文字列。

Header

Base64URL

署名アルゴリズム (RS256) と
秘密鍵 ID (kid)

Payload

Base64URL

発行者・対象・有効期限・
スコープ等のクレーム

Signature

Base64URL

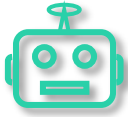
Header と Payload を
秘密鍵で署名

```
eyJhbGciOiJIUzI1NiIs... . eyJpc3MiOi... . abc123signature....
```

Accounts

ryuji-ssa-test@jYSTS03Sv7Bb1TdEHnGLWGsqh9FgWxDermShPGIjjVyuNDmR.adskserviceaccount.com

Ryuji_Ogasawara@jYSTS03Sv7Bb1TdEHnGLWGsqh9FgWxDermShPGIjjVyuNDmR.adskserviceaccount.com



サービスアカウント ID

サービスアカウント用Email アドレス

Delete Selected Account

Create Account With Name:

Ryuji

Ogasawara

Enable/Disable Account

Keys

bd6e0b06-44f0-4897-86bd-bb5121ef79c5

Delete Selected Key

Create Key

Enable/Disable Key

Private Key

-----BEGIN RSA PRIVATE KEY-----
MIIEpAIBAAKCAQEAq6V5etrY+NpqA6TsCJbQtc1kBwk84xu9BxK+QZcYsvXDCDi9
FKMVRFX3U4fpcczX44YdeS7vJqkIQUirMvXA1CR4LZnuXxezx6ttz7v6EWFNOBIR
490AoQAS4xSmgyUSYPUKir0Bw5ZL4irI89iVne2Z32XIbHoFVZFGofRAfUr65oIl
db9A6jyDSVvevKy9SfJ95dMq39Up3kjFrLCmXWKtGhowTa0aFt6qSxiuptpeRvLE
0V7nmmMyuisvv00NbqBROODc/PVrm+8o+5oGIjb1gkiYUI6gh3V2KM3xL4ttHxEF
5fVl7u1YTWZ6S3GnTrK8VATd1GMCmneqFLbMgwIDAQABAoIBAABAK85upADpiijZ
iCTwVYYI7WmvdMayKzqCkyHFkQXkVOV863HmjMFVIRXAJHmtgJvQeJ9u48gl8Bt
jLok2zAcvRNCdhn0WbA9KET2xieAFyKblFJrYzZxy8xdiAz5XMcuA/s3gx9/nj1n
ZF7ewkejTB1yodb0V75dR9NfyVeEFk9G6Wpib0x/Fpo7Tdh+KW7Yt07YYFYPPqCk
-----END RSA PRIVATE KEY-----

Account Details

{
 "serviceAccountId": "BX8JWKX4M587T3VQ",
 "email": "Ryuji_Ogasawara@jYSTS03Sv7Bb1TdEHnGLWGsqh9FgWxDermShPGIjjVyuNDmR.adskserviceaccount.com",
 "createdBy": "jYSTS03Sv7Bb1TdEHnGLWGsqh9FgWxDermShPGIjjVyuNDmR",
 "status": "ENABLED",
 "createdAt": "2026-06-04 05:46:28 +0000 UTC",
 "accessedAt": "2026-06-09 01:28:05 +0000 UTC",
 "expiresAt": "2027-06-09 01:28:05 +0000 UTC"
}

Key Details

Access Token

JWT ペイロードのフィールド解説

iss	Issuer 発行者	APS Client ID。APS が「誰が発行した JWT か」を識別
sub	Subject 主体	Service Account ID。「誰として認証したいか」  対応するサービスアカウント用 Email アドレスがわかる
aud	Audience 受信者	Token Endpoint URL。「どの API 向け JWT か」
exp	Expiration 有効期限	Unix epoch（秒）。最大 +300 秒（5 分）
scope	Scope スコープ	data:read など、要求する権限の配列

Forma/Fusion プロジェクトに追加する email アドレス

- SSA アカウントを選択し、「Account Details」セクションを表示する。

サービスアカウントの ID と メールアドレスが紐づけられている

serviceAccountId

サービスアカウントを一意に識別する ID

例 : 5e8b3a2c-9f4d-4b7e-8a1f-1234567890ab

email

招待時に使うアカウントのメールアドレス

例 : mcp.server@*****apsserviceaccount.com

Forma プロジェクトに招待する手順

- 対象 Forma プロジェクトを開く
- 左メニュー「メンバー」→「メンバーを追加」をクリック
- 招待ダイアログにサービスアカウントの Email アドレスを入力 → Enter
- 役割を割り当てる：
 - – Project Admin（推奨・全機能アクセス）
 - – または Member（より制限された権限）
- 「招待を送信」をクリック



サービスアカウントは招待の承認が不要

通常のユーザーと違って、招待した瞬間からアクセス可能になる。
メールボックスでの承認操作は発生しない。

Forma プロジェクトに招待する手順

AUTODESK Forma

Project Admin ▾

テストプロジェクト1 ▾

?

メンバー

会社

ブリッジ

竣工

アクティビティ

通知

場所

設定

メンバー

メンバー追加 ▾

書き出し

名前または電子メールでメンバーを...

名前	電子メール	アクセス レベル	追加日	Datum	Data Management	
RO Ryuji Ogasawara	ryuji_ogasawara@jysts03sv7bb1tdeh...	プロジェクト メンバー	2026/06/04	☐	☒	⋮
 Ryuji Ogasawara	ryuji.ogasawara@autodesk.com	プロジェクト管理者	2026/06/04	☐	☒	
小龍 小笠原 龍司	ogaryu@gmail.com	プロジェクト管理者	2026/06/04	☐	☒	⋮

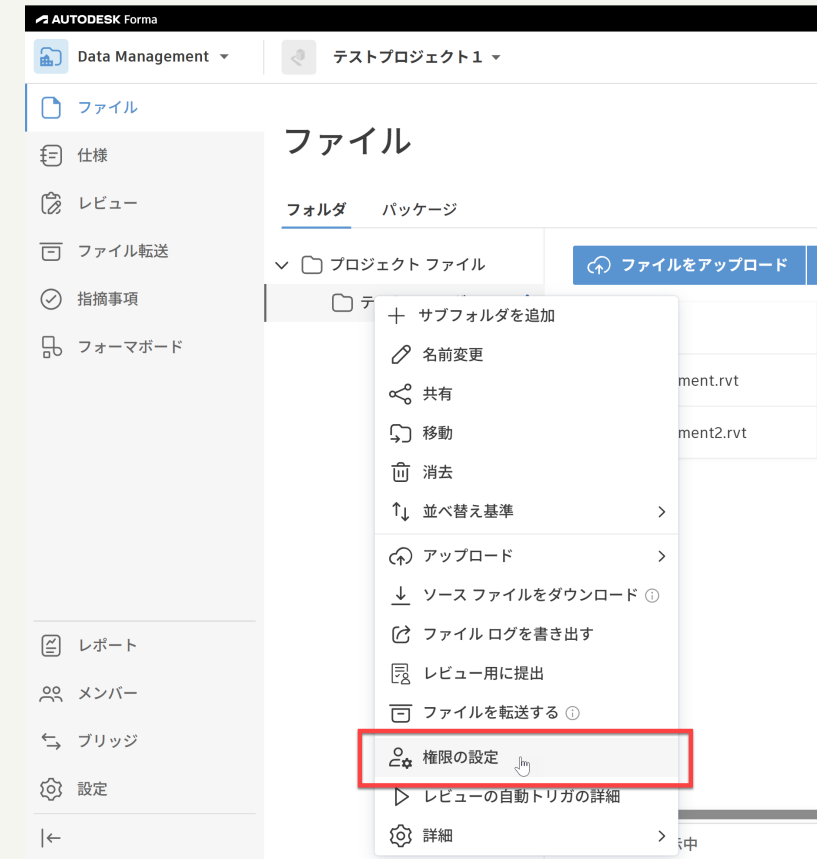
1 ~ 3/3 の表示中

127

Forma フォルダ権限の設定（ファイルツール）

- プロジェクトメンバー追加だけではファイルへのアクセスは制限されたままです。
- 対象フォルダごとに明示的に権限を付与する必要があります。

1. プロジェクトの「ファイル」ツールを開く
2. 対象フォルダ右側の「…」（3 ドット）→「権限の設定」
3. サイドパネル「権限」で「+ 追加」をクリック
4. SSA ロボットアカウントを選択し、権限レベルを設定
 - - View / View + Download / Edit / Manage など
5. 「追加」ボタンで保存
6. アクセスさせたい全フォルダで同じ手順を繰り返す



今回のワークショップでは、Get Folder Contents ツールでアクセスしたいフォルダ（複数あれば全部）に権限付与する。

Fusion プロジェクトに招待する手順

- Fusionで対象のプロジェクトからプロジェクト→「メンバと権限」を表示し、SSA作成時のメールアドレスを指定。
- 追加されたアカウントに、必要なアクセス権限を指定

GtjDmR5W7OPtWKNBd9cr..... Log In GitHub

Account Details

```
{  "serviceAccountId": "CCR2SMAON8DW6KWH"  "email": "Torano. .... Z.adsserviceaccount.com"}
```

Default Project

コンテンツ メンバーと権限(5) Wiki ページ

権限は、プロジェクト、フォルダ、またはサブフォルダのレベルで割り当てることができます。特定のフォルダへのアクセス権を持つメンバーは、そのフォルダが属するフォルダ構造全体を表示できますが、メンバーは割り当てられたフォルダ内のデータのみを表示できます。 [詳細はこちら](#)

メンバーおよびグループを追加または検索

メンバーを追加

グループを作成

☐	名前	↑	権限レベル
☐	(((A) Admins (管理者グループ)		■■■■■
☐	Takehiro Kato		■■■■■
☐	TK Takehiro Kato		■ ■ ■ ■
☐	TK Takehiro Kato		■ ■ ■ ■

MCP Inspector ツールの動作確認



MCP Inspector で全ツールテスト

① Inspector を起動

terminal

```
npx @modelcontextprotocol/inspector
```

② MCPサーバーに接続

MCP Inspectorのウェブページで、接続パラメータを次のように設定してください。

Transport Type : STDIO

Command : node

Arguments : server.js

その後、トグルボタンをクリックして接続を切り替えてください。

MCP Inspector で全ツールテスト

② List Tools をクリック → 2 ツールが表示

- getProjectsTool
 - getFolderContentsTool
- // getIssuesTool
// getIssueTypesTool

③ getProjectsTool を選択 → Run Tool

→ Forma アカウントとプロジェクト一覧が JSON で返ってくれば成功！



他のツールも projectId / accountId を入れて順次テストする



Servers

Export Add Servers ⌵ ⌵

⋮ filesystem-server-default

● Disconnected

STDIO

Standard I/O

npx -y @modelcontextprotocol/server-filesystem /tmp

Clone

Edit

Remove

Settings

⋮ everything-server-default

● Disconnected

STDIO

Standard I/O

npx -y @modelcontextprotocol/server-everything

Clone

Edit

Remove

Settings

⋮ example-server-default

● Disconnected

HTTP

Streamable HTTP

https://example-server.modelcontextprotocol.io/mcp

Clone

Edit

Remove

Settings

⋮ aps-mcp-server-nodejs

● Disconnected

STDIO

Standard I/O

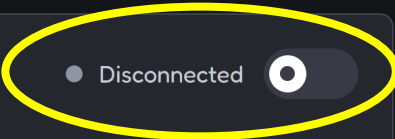
node server.js

Clone

Edit

Remove

Settings



Servers

Export Add Servers [Dropdown] [Icons]

example-server-default

Disconnected [Toggle]

HTTP Streamable HTTP

https://example-server.modelcontextprotocol.io/mcp

Clone Edit Remove Settings

aps-mcp-server-nodejs

Connected [Toggle]

STDIO Standard I/O

MCP 2025-11-25

node server.js

Clone Edit Remove Connection Info Settings

Protocol

Console

Search...

Messages

LEGACY

- Newest First [Dropdown]

Clear

Export

[Icons]
- 00:36:04 CLIENT → SERVER 67ms OK

TOOLS/LIST [Icons]
- 00:36:04 CLIENT → SERVER

NOTIFICATIONS/INITIALIZED [Icons]
- 00:36:03 CLIENT → SERVER 1186ms OK

INITIALIZE [Icons]

Tools

Search tools...

☒ Paginated

Get Folder Contents

getFolderContentsTool

Get Accounts & Projects

getProjectsTool

Get Accounts & Projects

Retrieves all Autodesk Construction Cloud (ACC) accounts and their associated projects accessible to the configured service account. Returns a structured list of accounts (with account IDs and names) and associated projects (with project IDs and names).

Edit as JSON ☐

Execute Tool

Protocol

Console

Search...

Messages

LEGACY

Newest First



Clear

Export



00:39:15

CLIENT → SERVER

2599ms

OK

TOOLS/CALL

getProjectsTo



00:38:30

CLIENT → SERVER

4576ms

OK

TOOLS/CALL

getProjectsTo



00:36:04

CLIENT → SERVER

67ms

OK

TOOLS/LIST



00:36:04

CLIENT → SERVER

NOTIFICATIONS/INITIALIZED



00:36:03

CLIENT → SERVER

1186ms

OK

MCP サーバーの 起動・テスト

Forma / Fusion

MCP クライアント

VS Code

GitHub Copilot

Microsoft の AI ペアプロ。
.vscode/mcp.json で設定。
仕事の文脈で使いやすい。

Cursor

AI ファースト IDE

VS Code ベース。
.cursor/mcp.json で設定。
MCP ネイティブサポート。

Claude Desktop

Anthropic 公式

デスクトップアプリ。
claude_desktop_config.json で設定。
対話 UI 中心。

GitHub Copilot 連携の設定

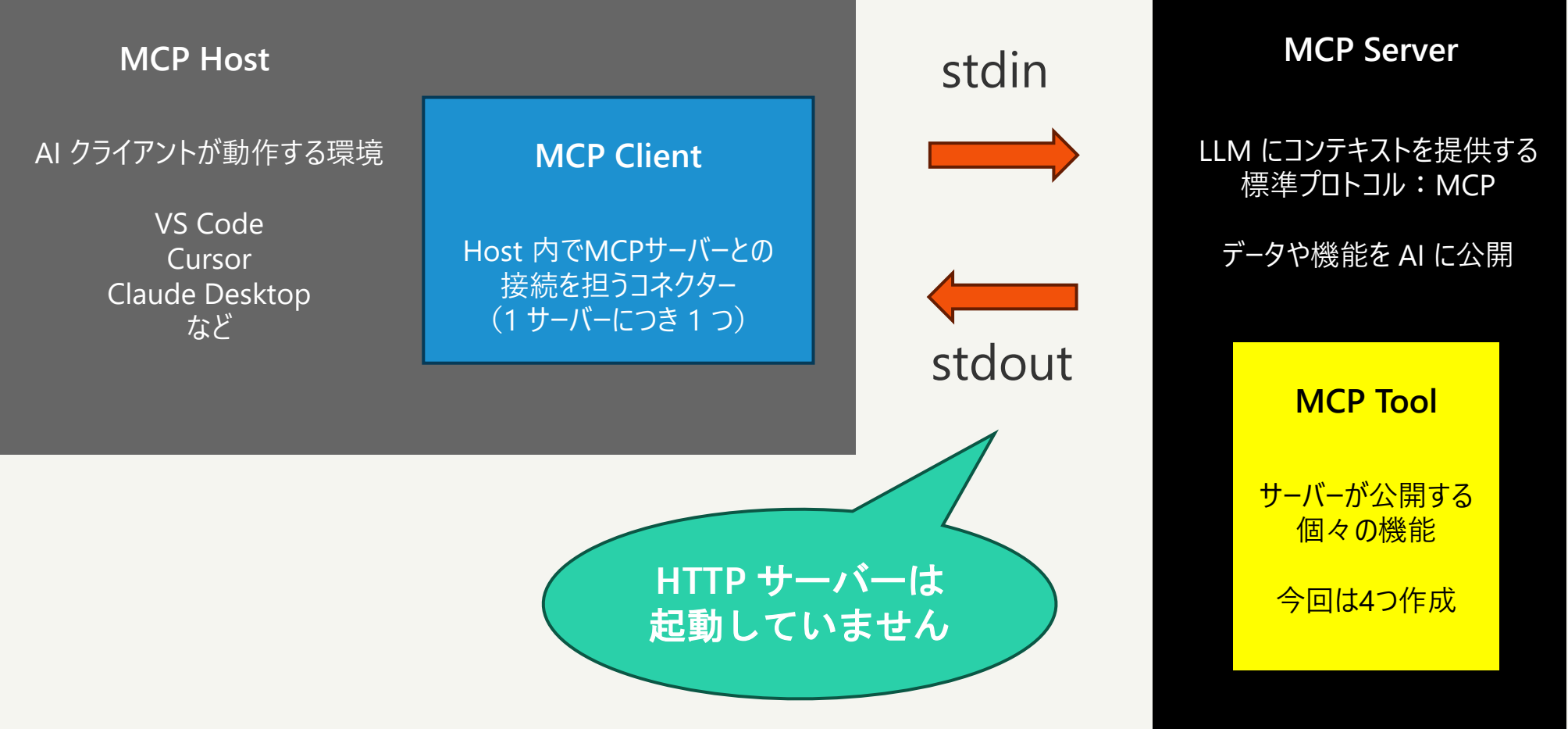
.vscode/mcp.json

```
{
  "servers": {
    "aps-mcp-server-nodejs": {
      "type": "stdio",
      "command": "node",
      "args": [ "server.js" ]
    }
  }
}
```

セットアップ手順

- VS Code で MCP を有効化
 - code.visualstudio.com の手順を参照
- プロジェクト直下に .vscode/mcp.json 作成
- ファイル冒頭の Start ラベルでサーバー起動
- Copilot Chat を開く
 - Ctrl+Shift+I (Win/Linux)
 - Cmd+Shift+I (macOS)

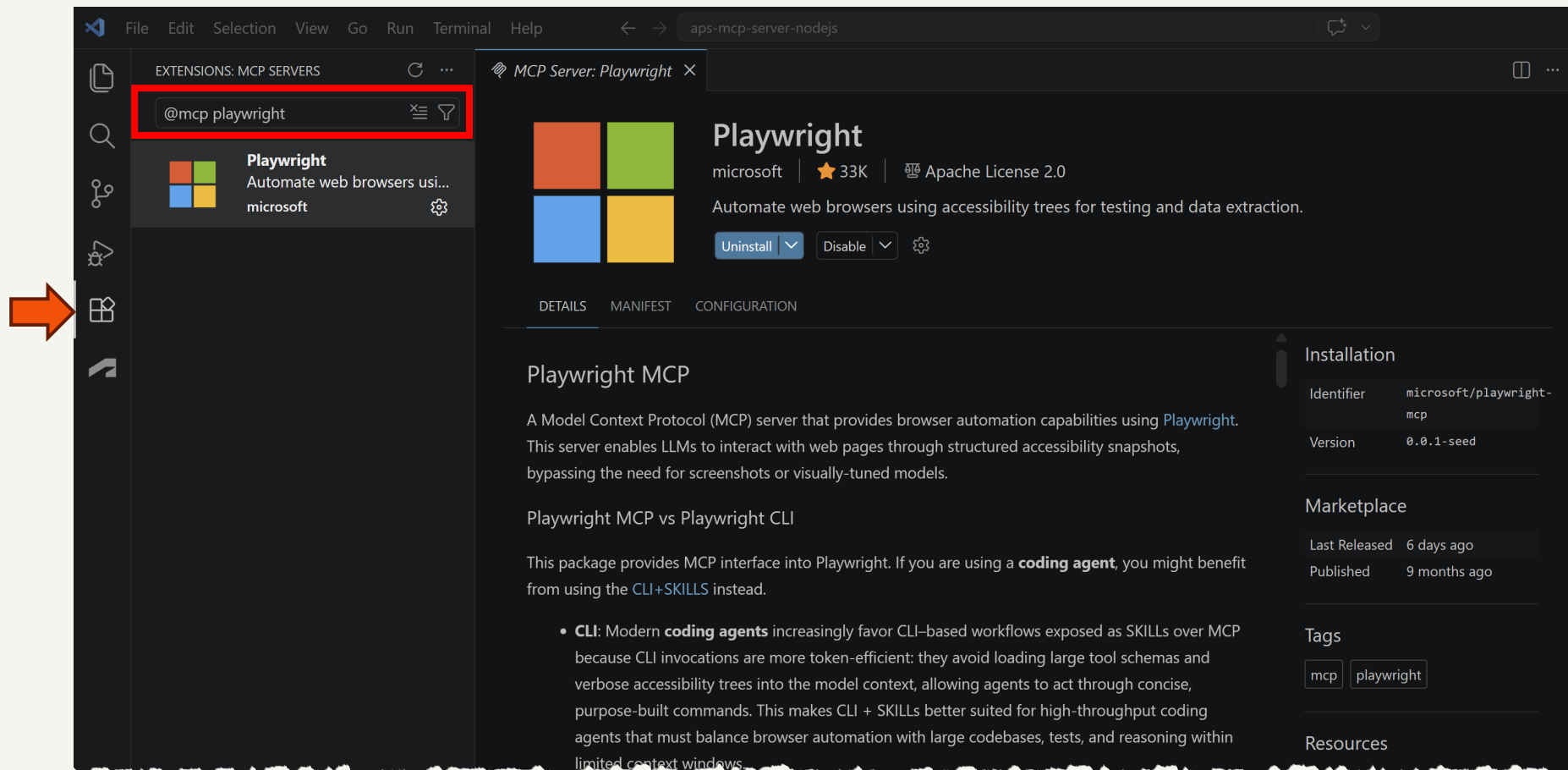
標準入出力による通信



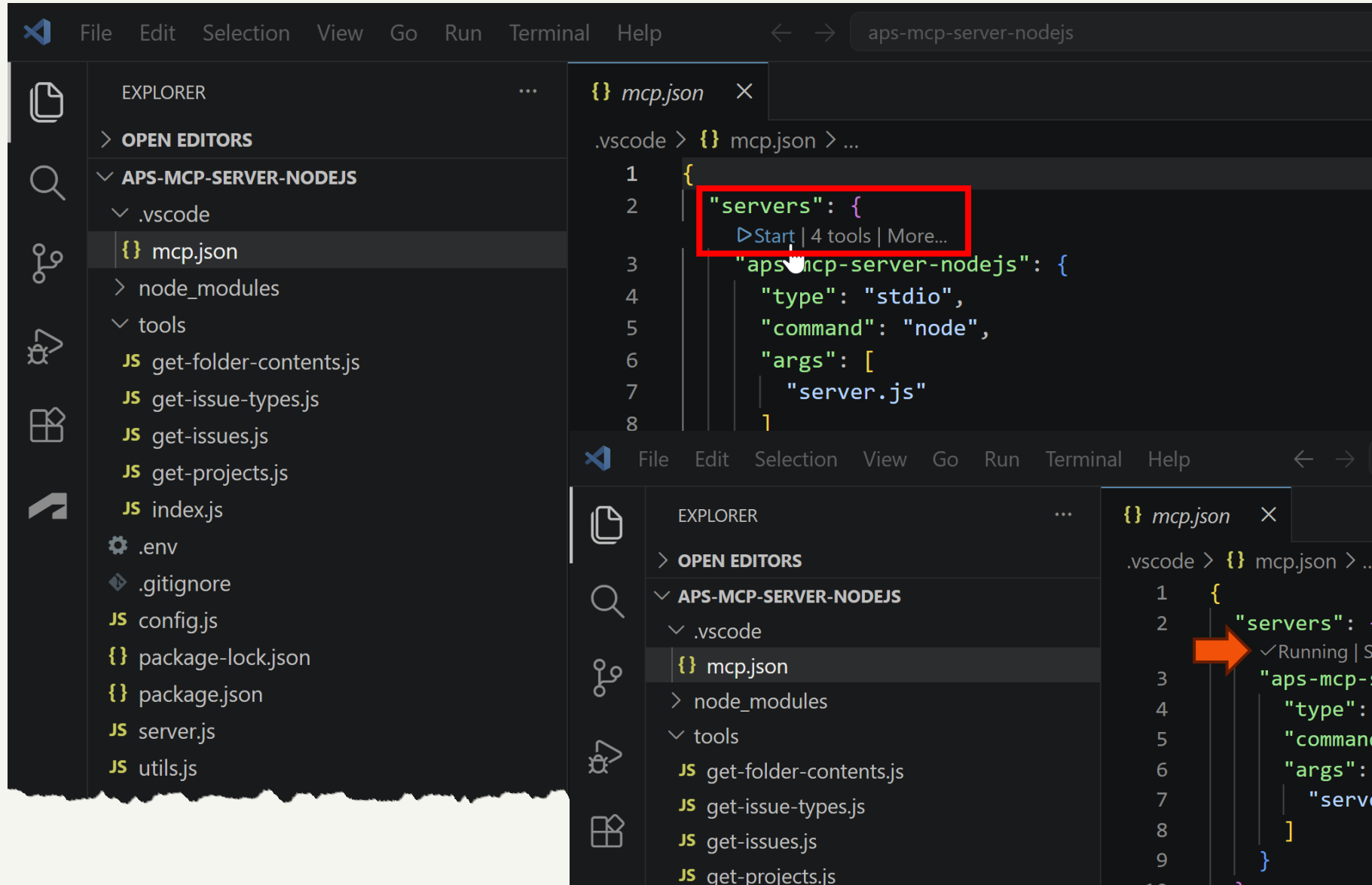
MCP の有効化（Playwright MCPのインストール）

<https://blog.autodesk.io/connect-mcp-on-vs-code/>

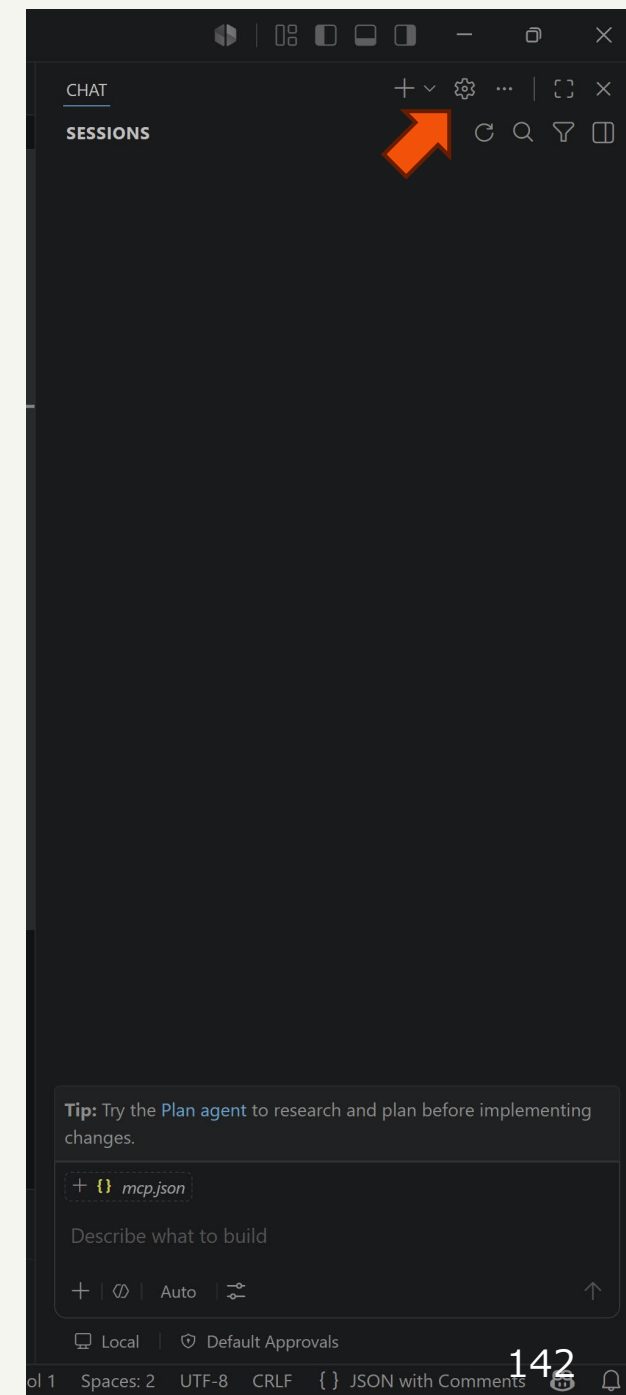
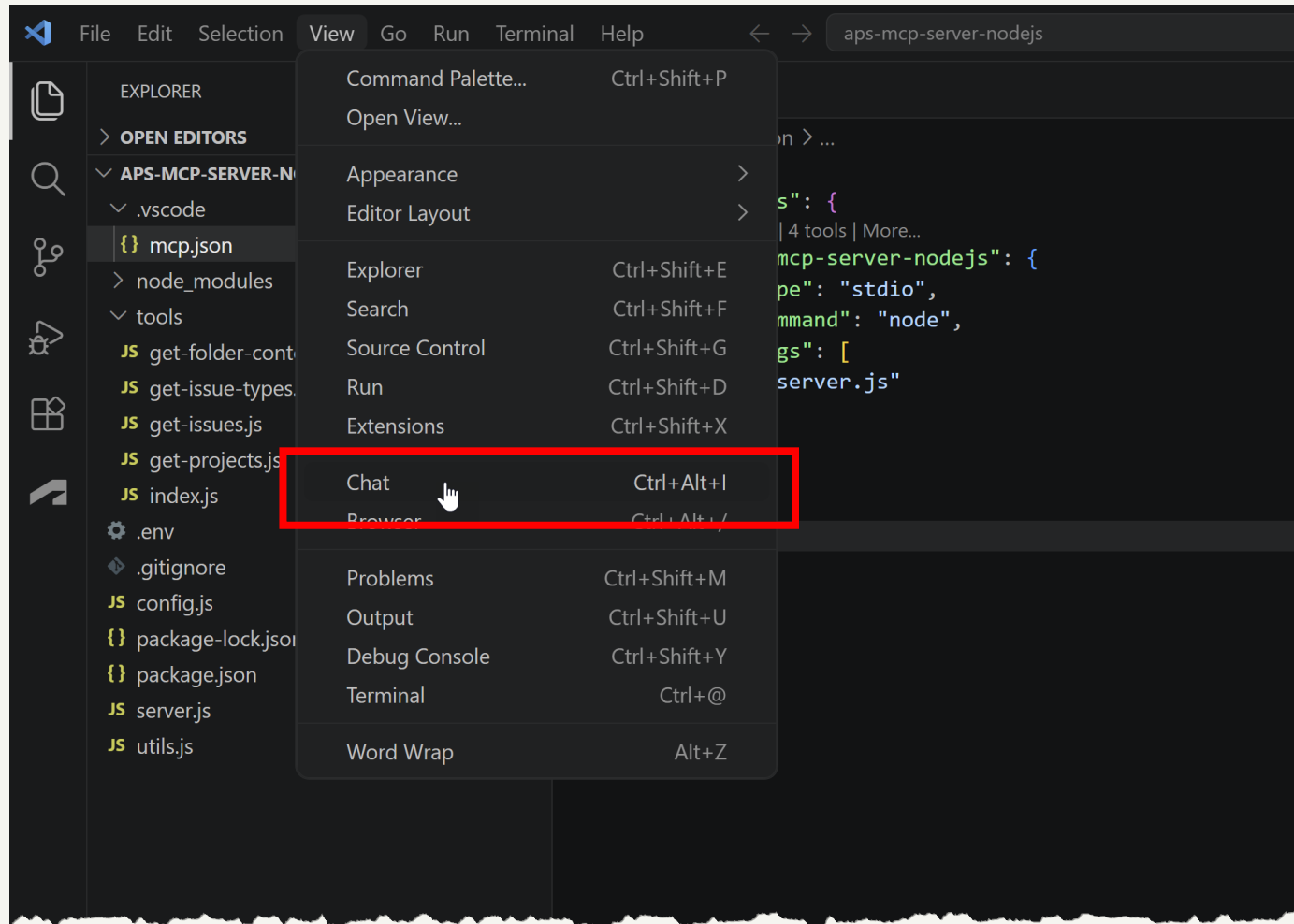
- Extension タブから **@mcp playwright** を検索してインストール



MCP サーバーの起動方法



Chat ウィンドウの表示 (Ctrl+Alt+I)



File Edit Selection View Go Run Terminal Help

aps-mcp-server-nodejs

EXPLORER

OPEN EDITORS

APS-MCP-SERVER-NODEJS

.vscode

mcp.json

node_modules

tools

get-folder-c

get-issue-ty

get-issues.js

get-projects

index.js

.env

.gitignore

config.js

package-lock

package.json

server.js

utils.js

Agent Customizations

Local

Agents 3

Skills 10

Instructions

Prompts 1

Hooks 1

MCP Servers 2

Plugins

MCP Servers

An open standard that lets AI use external tools and services. MCP servers provide tools for file operations, databases, APIs, and more.
[Learn more about MCP servers](#)

Type to search...

Browse Marketplace +

Workspace 1

aps-mcp-server-nodejs **Running**

User 1

Playwright
Automate web browsers using accessibility trees for testing and data extraction. **Stopped**

PS C:\Users\ogasawr\GitHubRepository\aps-mcp-server-nodejs>

Describe what to build

+ | | Auto |

Local | Default Approvals

Ln 12, Col 1 Spaces: 2 UTF-8 CRLF {} JSON with Comments

プロンプトレシピ集



ディスカバリ系

私がアクセスできるプロジェクトは何ですか？

私の Forma アカウントのサマ리를教えてください。



ファイル探索

Forma プロジェクト [プロジェクト名] にある Revit ファイルを全てリストアップしてください。

フォルダー X の中にあるドキュメントで先週更新されたものを探してください。

クライアント連携時のトラブルシューティング



サーバーに接続できない

コマンド／引数のパスを再確認。VS Code は相対パス可だが Claude Desktop は絶対パス必須



AI がツールを呼んでくれない

description が曖昧／類似ツールと競合。description を明確にする・名前を一意にする



応答が極端に遅い

Get Projects のように複数 API を直列で叩くツールは並列化検討。N+1 を Promise.all で改善



認証エラーが頻発

アクセストークンのキャッシュ／expires_in 確認。サーバー再起動で .env 再読み込み



ログが何も出ない

stdio トランスポートでは console.log が干渉する → console.error を使う（stderr へ）

次のステップ

ツールの拡張

- Get Issue / Get Issue Type：指摘事項・指摘事項タイプの取得
- Create Issue：自然言語から指摘事項を作成（書き込み権限が必要）
- Get Sheets / Get Views：図面・ビュー情報の取得

セキュリティ強化

- 秘密鍵をシークレットマネージャ（Azure Key Vault 等）に移行
- スコープを最小権限に絞る（data:read だけで十分なら write は要らない）
- ログ・監査ログを整備（誰の AI が何をしたか追跡可能に）

デプロイ

- チーム共有：社内ネットワークに常駐させ複数人で共有（Streamable HTTP transport も検討）

本日のまとめ

- MCP (Model Context Protocol) の概念と全体像を理解しました。
- Secure Service Account でユーザー操作なしの認証フローを構築しました。
- Node.js + @modelcontextprotocol/sdk で MCP サーバーを実装しました。
- Forma/Fusion プロジェクトの情報とファイルを公開する 2 つのツールを実装しました。
 - – Get Projects / Folder Contents
- MCP Inspector で動作確認の流れを身につけました。
- VS Code で AI から利用しました。



これで「AI から Forma/Fusion にアクセス」できる土台が完成

これから先は組織のニーズに応じてツールを増やしていくフェーズ。



Autodesk and the Autodesk logo are registered trademarks or trademarks of Autodesk, Inc., and/or its subsidiaries and/or affiliates in the USA and/or other countries. All other brand names, product names, or trademarks belong to their respective holders. Autodesk reserves the right to alter product and services offerings, and specifications and pricing at any time without notice, and is not responsible for typographical or graphical errors that may appear in this document. © 2026 Autodesk. All rights reserved.