



# **AUTODESK Platform Services**

## **APS Online Training – Revit Automation**

小笠原 龍司

Developer **A**dvocacy & **S**upport、Autodesk Developer Network

# アジェンダ – ご紹介する内容

- はじめに
- 理由と仕組み
- Revit Automation の理解
- デベロッパキーの取得
- サーバーの作成
- 基本 UI
- プラグイン（アドイン）の作成
- Activity（アクティビティ）の定義
- WorkItem（ワークアイテム）の実行
- Automation API サポート ポリシー
- コストとサポート



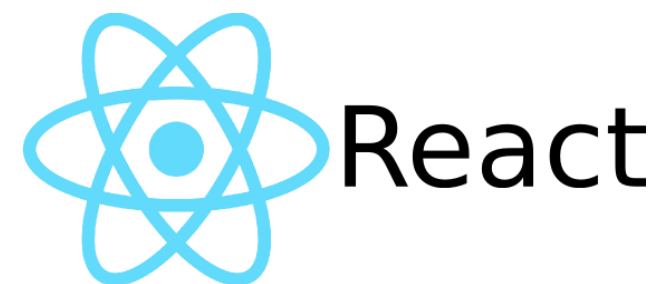
はじめに

# インターネットで各社提供 API への接続が可能な時代

- 業界標準 & オープンソース API テクノロジ



RESTful API  
GET PUT POST DELETE

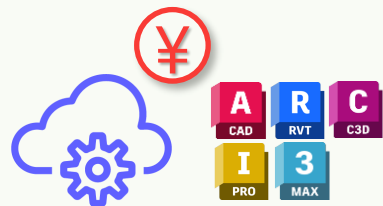


# Autodesk Platform Services – 主要 API

## Core APIs



Model Derivative



Automation



Viewer



Authentication



Webhooks

## Data APIs



AEC Data Model



Manufacturing Data Model



Data Management



Data Exchange

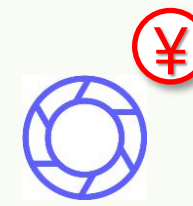
## Product APIs



Forma



Fusion



Reality Capture



ACC



BIM 360



Tandem

RESTful API  
GET PUT POST DELETE



# 公式ドキュメント（英語のみ）

<https://aps.autodesk.com/developer/documentation>

- APS ポータル (<https://aps.autodesk.com/>) に記載

**AUTODESK**  
Platform Services

Search

Sign in

Solutions ▾ Getting Started Documentation ^ Success Stories Community ▾ Support ▾ Pricing App Store ▾

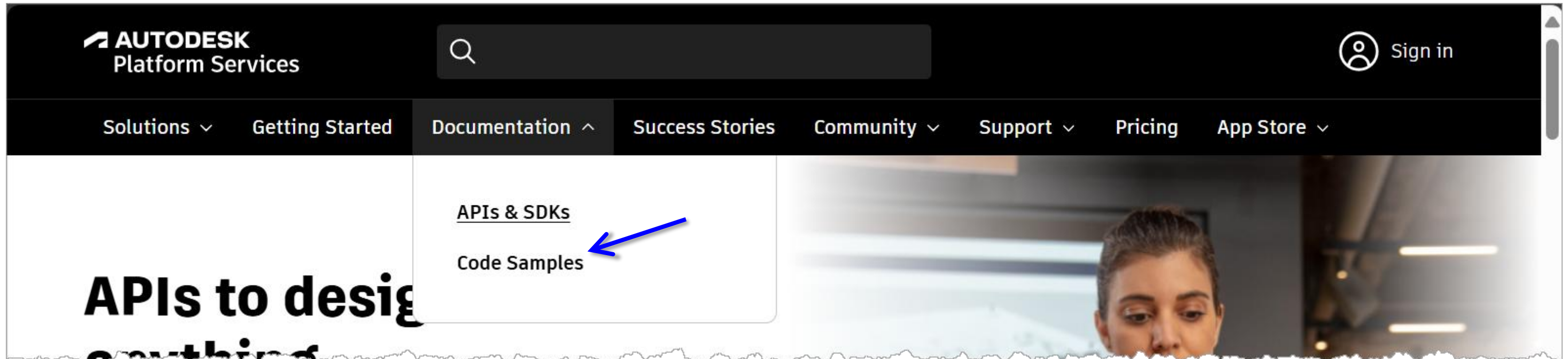
APIs & SDKs ←  
Code Samples

## APIs to design anything

Get the APIs and services you need to build better, custom business solutions. Autodesk Platform Services unlocks your Design and Make data to power new ways of working.

# APS サンプル – GitHub リポジトリ

<https://aps.autodesk.com/code-samples>



AUTODESK	Solutions	Documentation	Resources	About
Company overview	AEC Data Model	AEC Data Model	Get Help	About APS
Careers	Autodesk Construction Cloud	Authentication	API Status	Pricing
Investor relations	Autodesk Fusion	Autodesk Construction Cloud	Blog	Success Stories
Newsroom	BIM 360	BIM 360	FAQ	Certified Partners
	BuildingConnected API	Data Exchange	Code Samples	Partnerships
			ADN Member Sign-in	

# 学習リソース : Learn APS Tutorial (英語のみ)

<https://tutorials.autodesk.io/>

Get the APIs and services you need to build better, custom business solutions. Autodesk Platform Services unlocks your Design and Make data to power new ways of working.

Sign up to try Autodesk Platform Services



## APS Tuesdays

Join a free weekly webinar to get to know APS



## Learn APS Tutorial

Step by step guide on GitHub



## Developer Newsletter

Stay up to date with the developer community



# 学習リソース : Learn APS Tutorial

<https://get-started.aps.autodesk.com/>

The screenshot displays the Autodesk Platform Services (APS) website. The top navigation bar includes the Autodesk logo, a search bar, and a 'Sign in' button. Below the navigation bar, a menu lists various sections: Solutions, Getting Started, Documentation, Success Stories, Community, Support, Pricing, and App Store. The left sidebar contains a list of tutorials, with 'Automation' highlighted in blue. A blue bracket on the left side of the page groups the 'Automation' section and its sub-items: 'Create Server', 'Basic UI', 'Create Plugin', 'Define Activity', and 'Execute Workitem'. The main content area shows the 'Automation' page with a breadcrumb trail: Home > Tutorials > Automation. The page title is 'Automation' and the sub-page title is 'Introduction'. The introduction text states: 'This tutorial will guide you through creating a webapp that can upload an input file, change its width and height parameter and save the output file. The UI will let you define AppBundle & Activity (as an initial configuration) and execute Workitems, with an output result.' Below the text, a paragraph reads: 'Both input and output files are saved in OSS Buckets, you can use Simple Viewer tutorial to view them.' At the bottom, there is a screenshot of a web application interface. The interface shows a form with 'Width:' and 'Height:' input fields, both containing the value '80'. Below the form, there is a section for 'Input file'. To the right of the form, there is a console or log area showing the message 'Uploading input file...' and 'Workitem started: 48eea0ddec344d508e347844fed2dd42'.

**AUTODESK**  
Platform Services

Search

Sign in

Solutions ▾ Getting Started Documentation ▾ Success Stories Community ▾ Support ▾ Pricing App Store ▾

Getting Started  
Environment Setup  
**Tutorials**  
Simple Viewer  
Hubs Browser  
Dashboard  
**Automation**  
Create Server  
Basic UI  
Create Plugin  
Define Activity  
Execute Workitem  
ACC Administrator

Home > Tutorials > Automation

## Automation

### Introduction

This tutorial will guide you through creating a webapp that can upload an input file, change its width and height parameter and save the output file. The UI will let you define AppBundle & Activity (as an initial configuration) and execute Workitems, with an output result.

Both input and output files are saved in OSS Buckets, you can use Simple Viewer tutorial to view them.

Autodesk Forge - Design Autom...  
localhost:3000

**AUTODESK**  
Platform Services

Width:  
80

Height:  
80

Input file

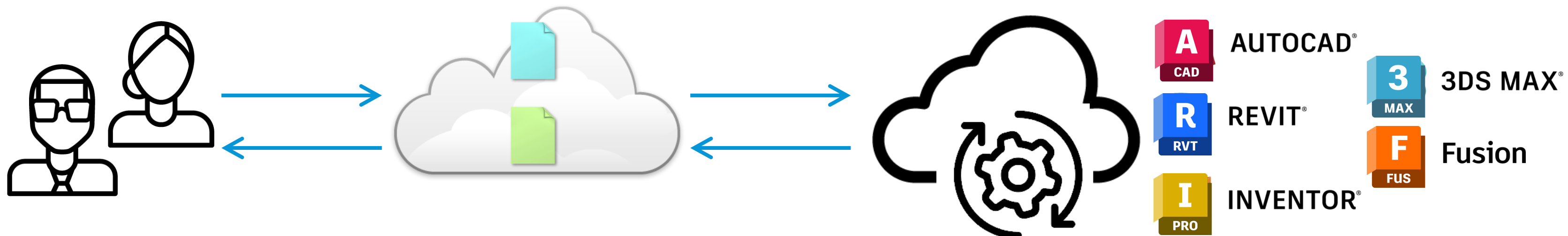
Uploading input file...  
Workitem started: 48eea0ddec344d508e347844fed2dd42

Configure

# 2025年7月1日（日本標準時） - 6月30日（米国太平洋標準時）～ Design Automation API 名称変更

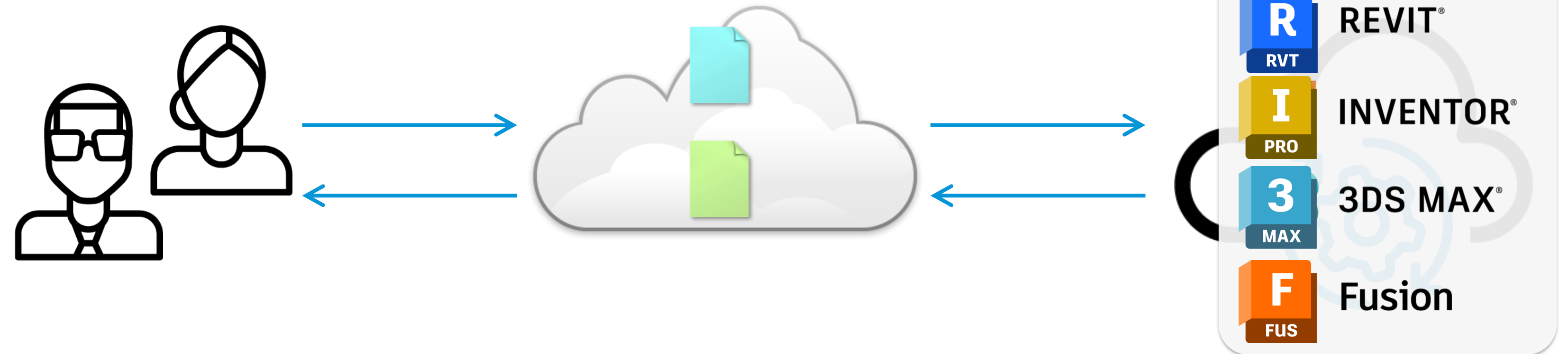
- Design Automation API → **Automation API**

旧呼称（エンジン別）	新呼称（エンジン別）
Design Automation API for AutoCAD	AutoCAD Automation API
Design Automation API for Revit	Revit Automation API
Design Automation API for Inventor	Inventor Automation API
Design Automation API for 3ds Max	3ds Max Automation API
Design Automation API for Fusion	Fusion Automation API



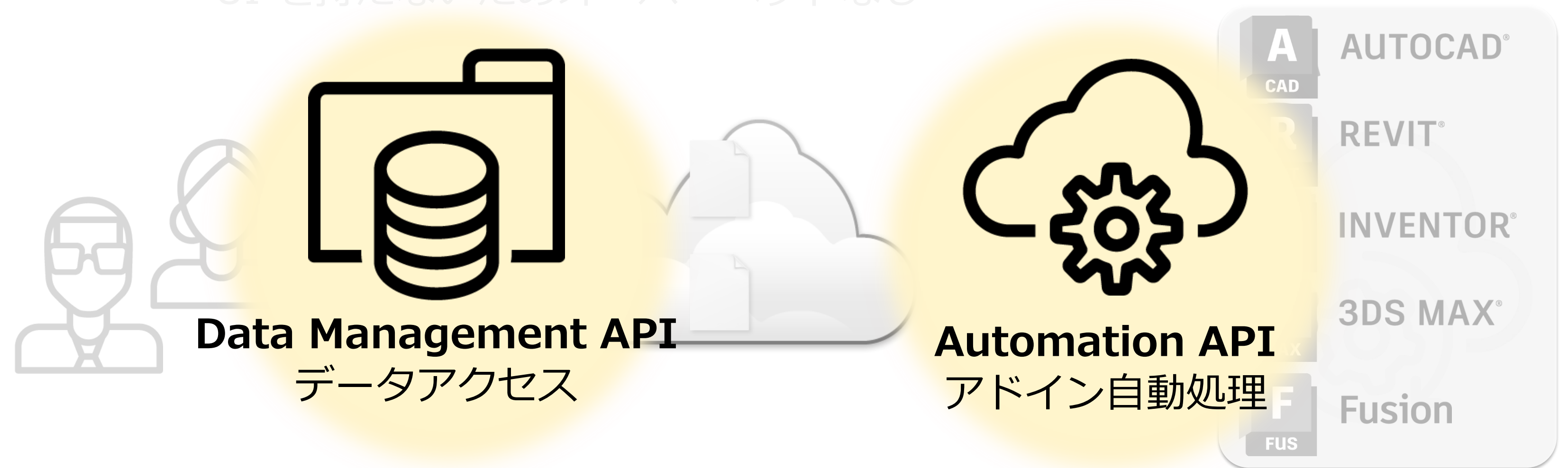
# 自動化ソリューション

- **Automation API (旧名 Design Automation API)**
- CAD エンジンで**アドイン**を実行する仕組み
  - 対面操作を意図した‘製品’ではない
  - UI を持たないためオーバーヘッドなし
  - 製品一般規約による制限事項は不適用



# 自動化ソリューション

- Automation API (旧名 Design Automation API)
- CAD エンジンでアドインを実行する仕組み
  - 製品一般規約による制限事項は不適用
  - UI を持たないためオーバーヘッドなし



# Automation API

## Web API + デスクトップ API の知識が必要

APS 開発

 **AUTODESK**  
Platform Services



製品別アドイン開発



# Forge SDK : 2016 年 ~

- エンドポイントをラップした環境毎の 'ライブラリ'
  - Node.js、.NET、PHP、Java、(Python) ...
  - Design Automation API v2 ( AutoCAD のみ) 対応
  - Design Automation API v3 対応は別 SDK で対応
- その後の暫定処置
  - Node.js、.NET のみ
  - [Ditect-to-S3 アプローチ](#)対応
  - [Authentication v2](#) 対応
- 現在、GitHub リポジトリは 'アーカイブ' 状態
  - 今後の機能面の変更は後続の APS SDK のみで対応予定

# APS SDK : 2024 年 ~

- 本 Design Automation Tutorial では未使用
- エンドポイントをラップした環境毎の 'ライブラリ'
  - SDKManager と API 毎に分割したパッケージで提供
  - .NET と Node.js (TypeScript) を用意
- .NET SDK
  - NuGet パッケージとして提供
  - <https://www.nuget.org/profiles/AutodeskPlatformServices.SDK>
- Node.js SDK (TypeScript SDK)
  - Node.js パッケージとして提供
  - <https://www.npmjs.com/~aps.sdk>



2016年 Forge 正式リリース

2019年 Design Automation v3 リリース

Forge SDK

Forge Design Automation SDK - v3用

APS SDK

2016

2017

.....

2023

2024





## Developer's Guide

### Overview

- > API Basics
- GDPR Compliance
- Webhooks
- Field Guide
- Supported Translations
- > Rate Limits

## How-to Guide

## Code Samples

## Reference Guide

- > REST API Reference
- > .NET SDK Reference
- > TypeScript SDK Reference

## Change History

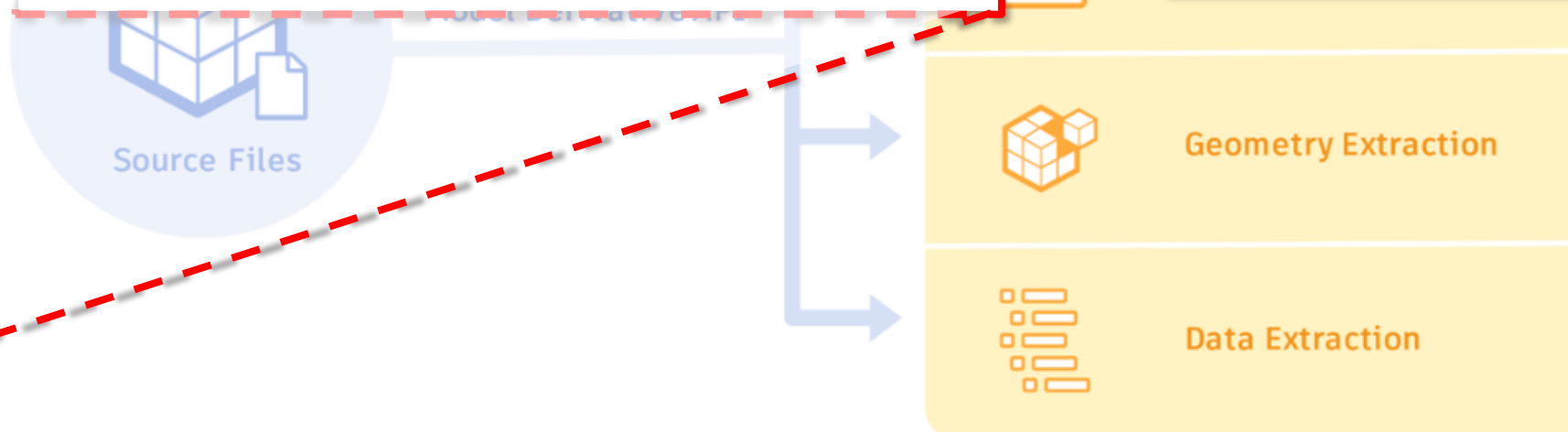
## About the Model Derivative API


- > REST API Reference
- > .NET SDK Reference
- > TypeScript SDK Reference

エンドポイント リファレンス

.NET SDK リファレンス

Node.js リファレンス





はじめに ～ 理由と仕組み

# Revit アドインの目的？

- モデリング補助機能の提供

- 反復タスクの自動化の提供

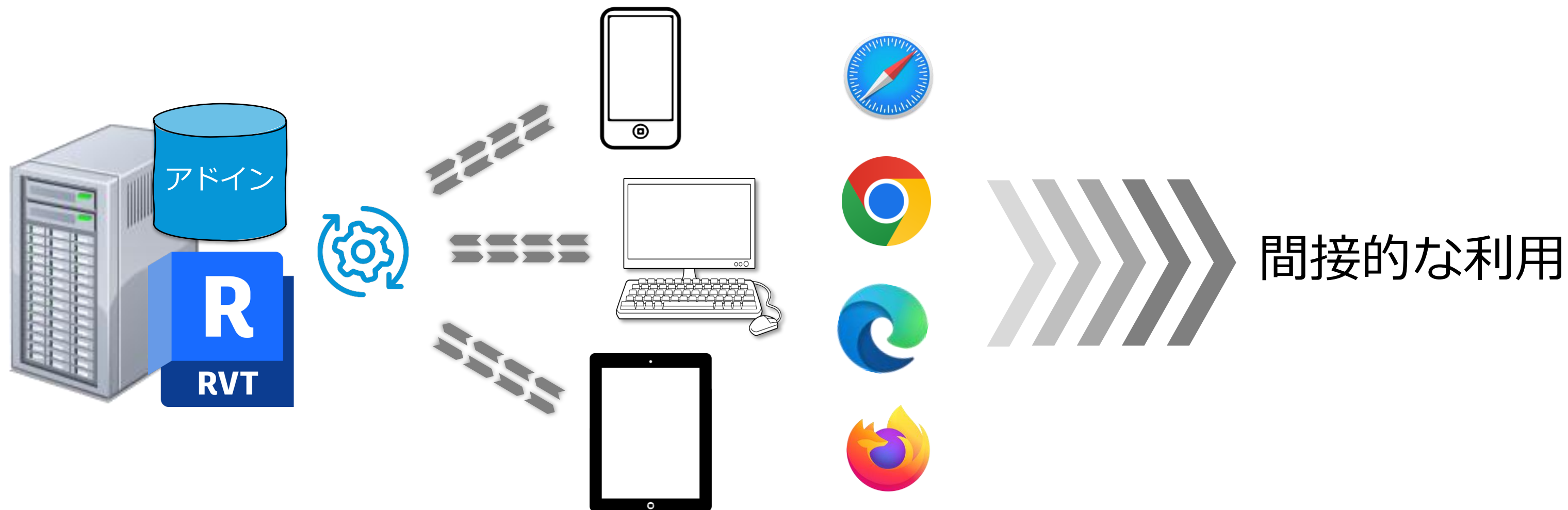
- モデル内情報を利用した外部システムの連携



生産性の向上

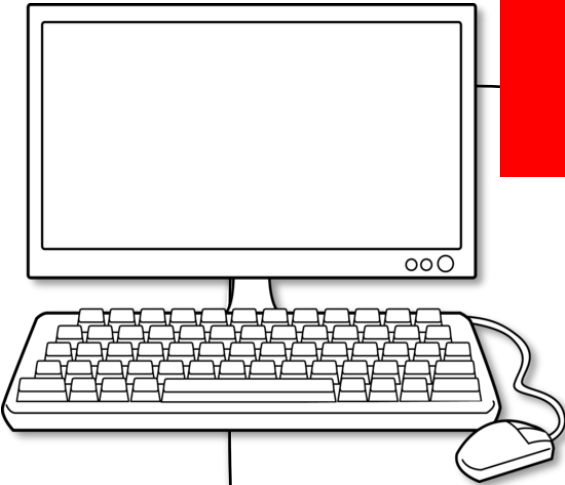
# Revit アドイン運用の多様化

- インターネット - クラウドの普及にともなう要求の変化



# オートデスク デスクトップ製品の API で出来ること

## デスクトップ製品内の自動化を実現



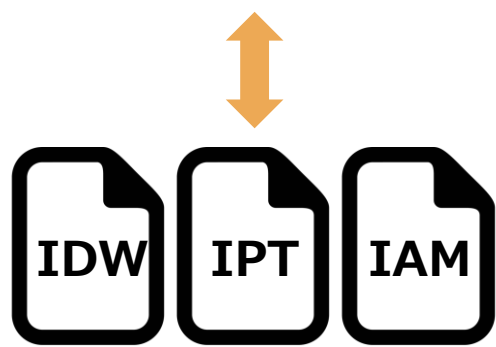
**R**  
RVT

プレゼンテーション  
ビジネス ロジック  
データ



**I**  
PRO

プレゼンテーション  
ビジネス ロジック  
データ



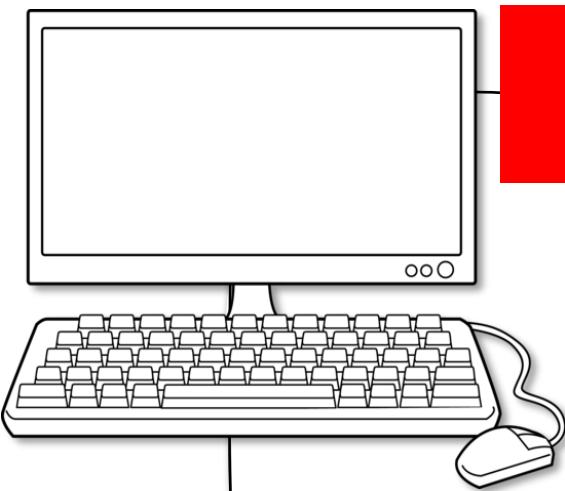
**A**  
CAD

プレゼンテーション  
ビジネス ロジック  
データ



# Windows の API で出来ること

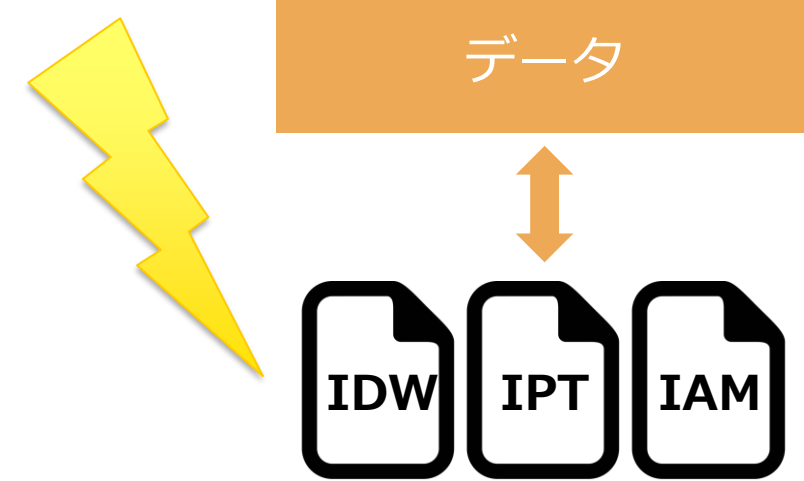
デスクトップ製品の外部からの起動を実現



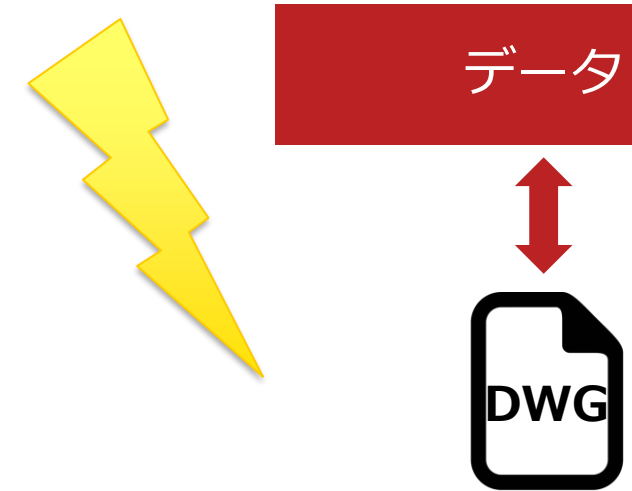
プレゼンテーション  
ビジネス ロジック  
データ



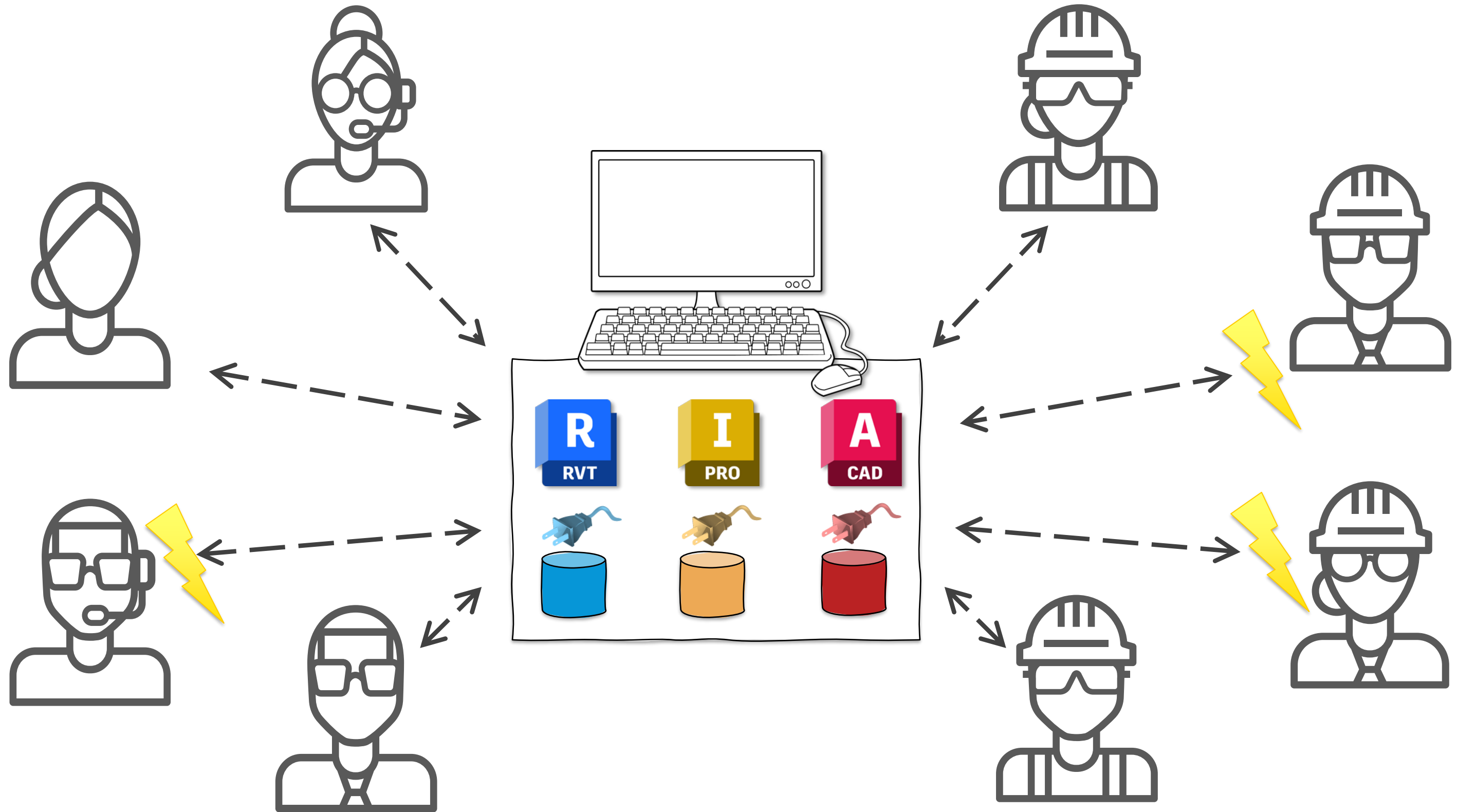
プレゼンテーション  
ビジネス ロジック  
データ



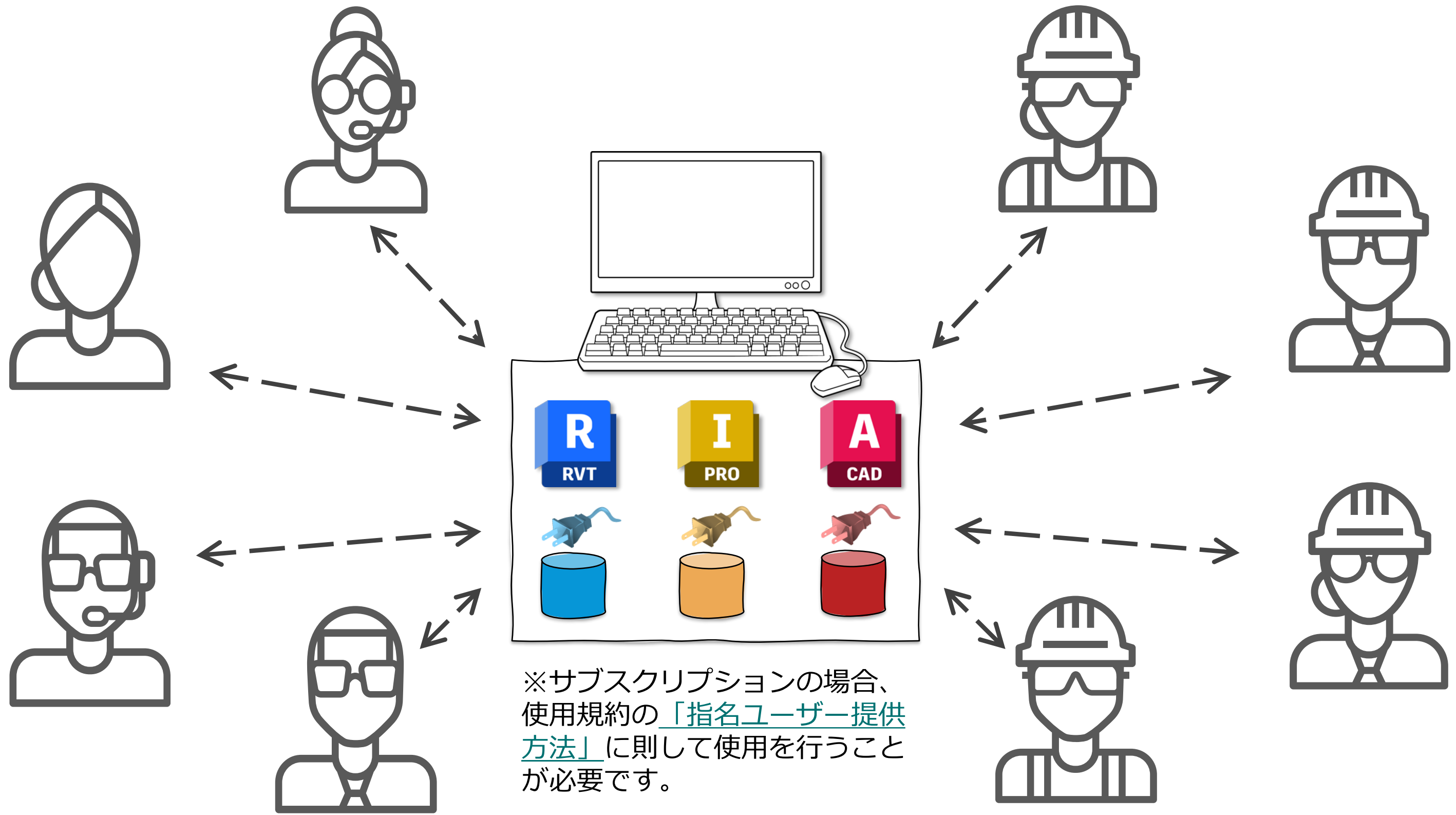
プレゼンテーション  
ビジネス ロジック  
データ



# デスクトップ製品を利用した Web 利用の可否？

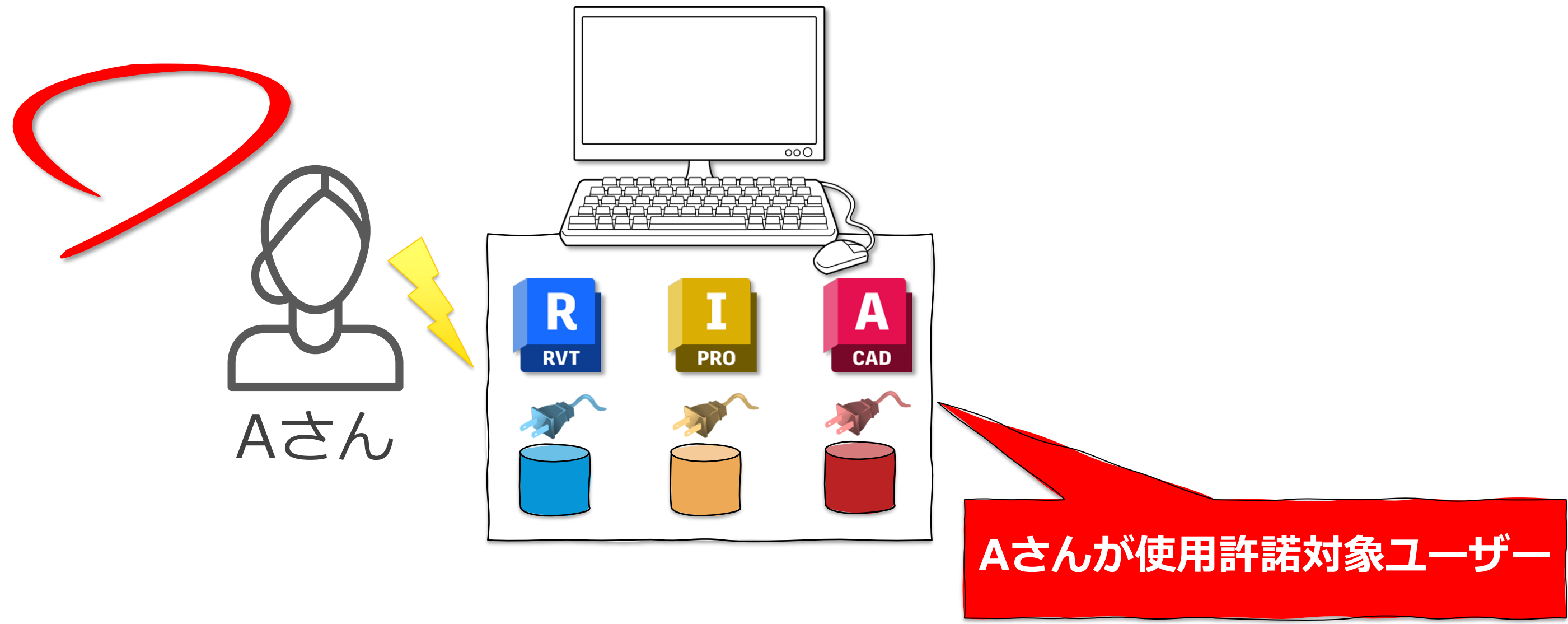


# デスクトップ製品は誰にライセンスされているか？



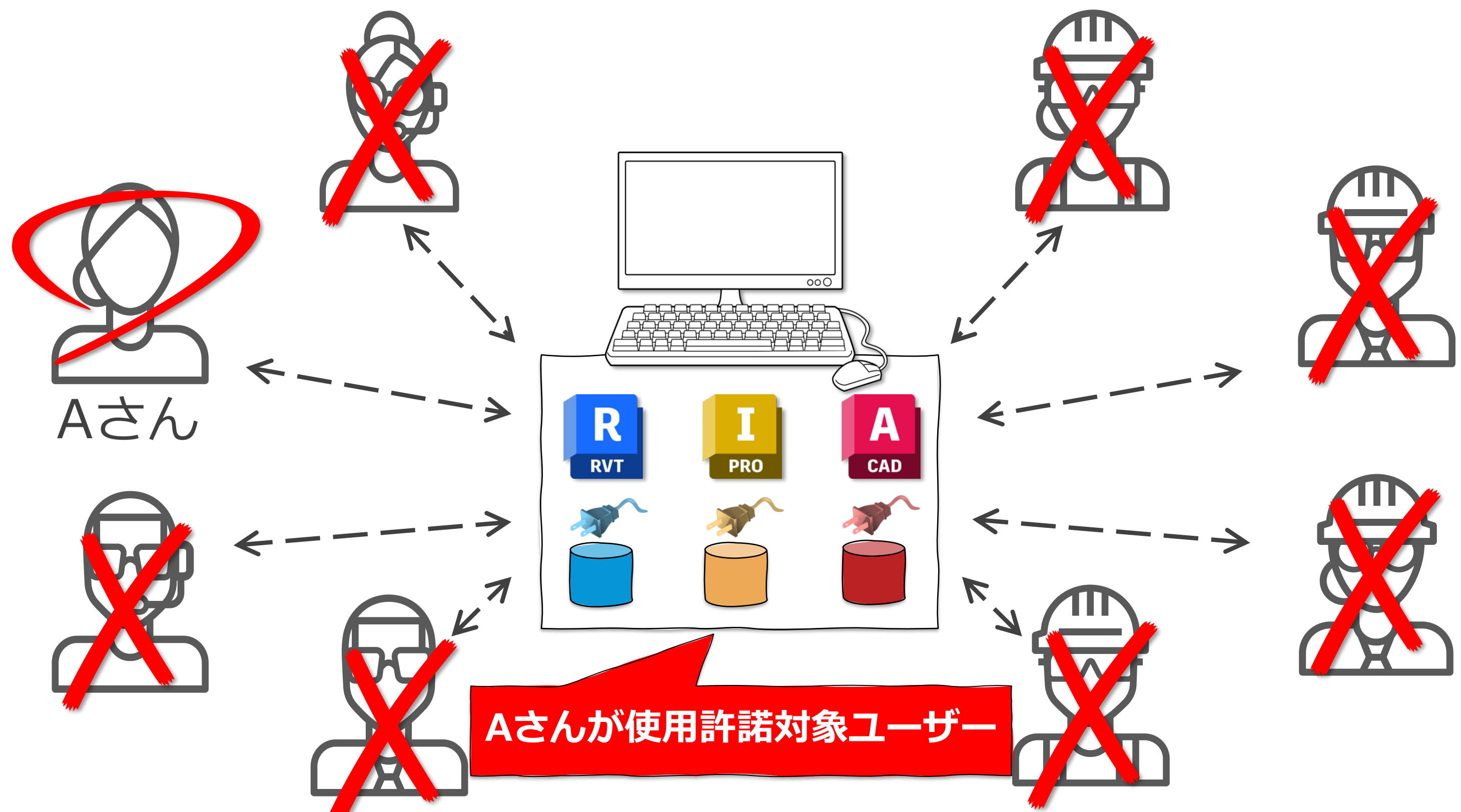
※サブスクリプションの場合、  
使用規約の「[指名ユーザー提供方法](#)」に則して使用を行うことが  
必要です。

# デスクトップ製品は誰にライセンスされているか？

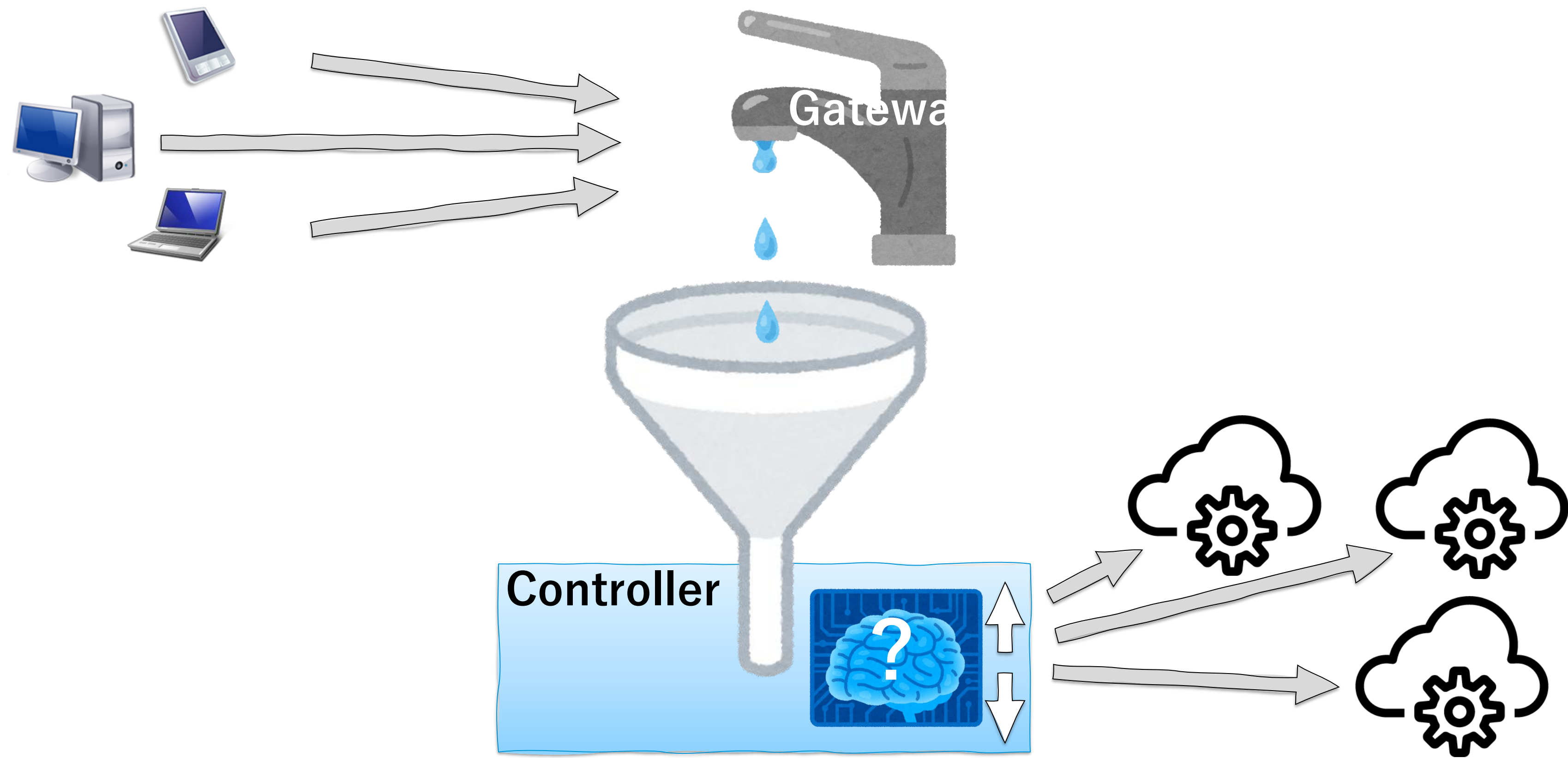


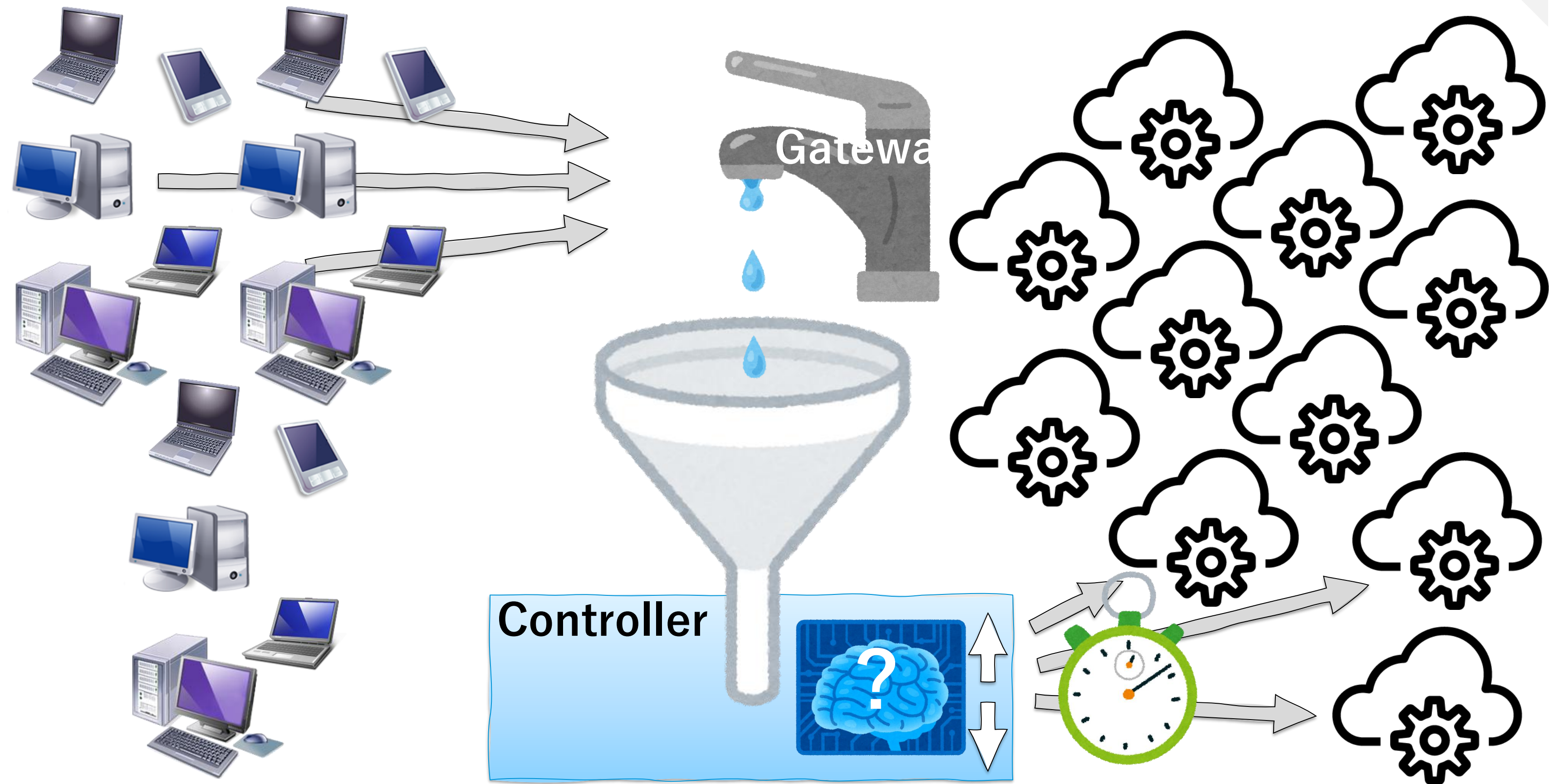
**Aさんが使用許諾対象ユーザー**

# デスクトップ製品は誰にライセンスされているか？



**Aさんが使用許諾対象ユーザー**

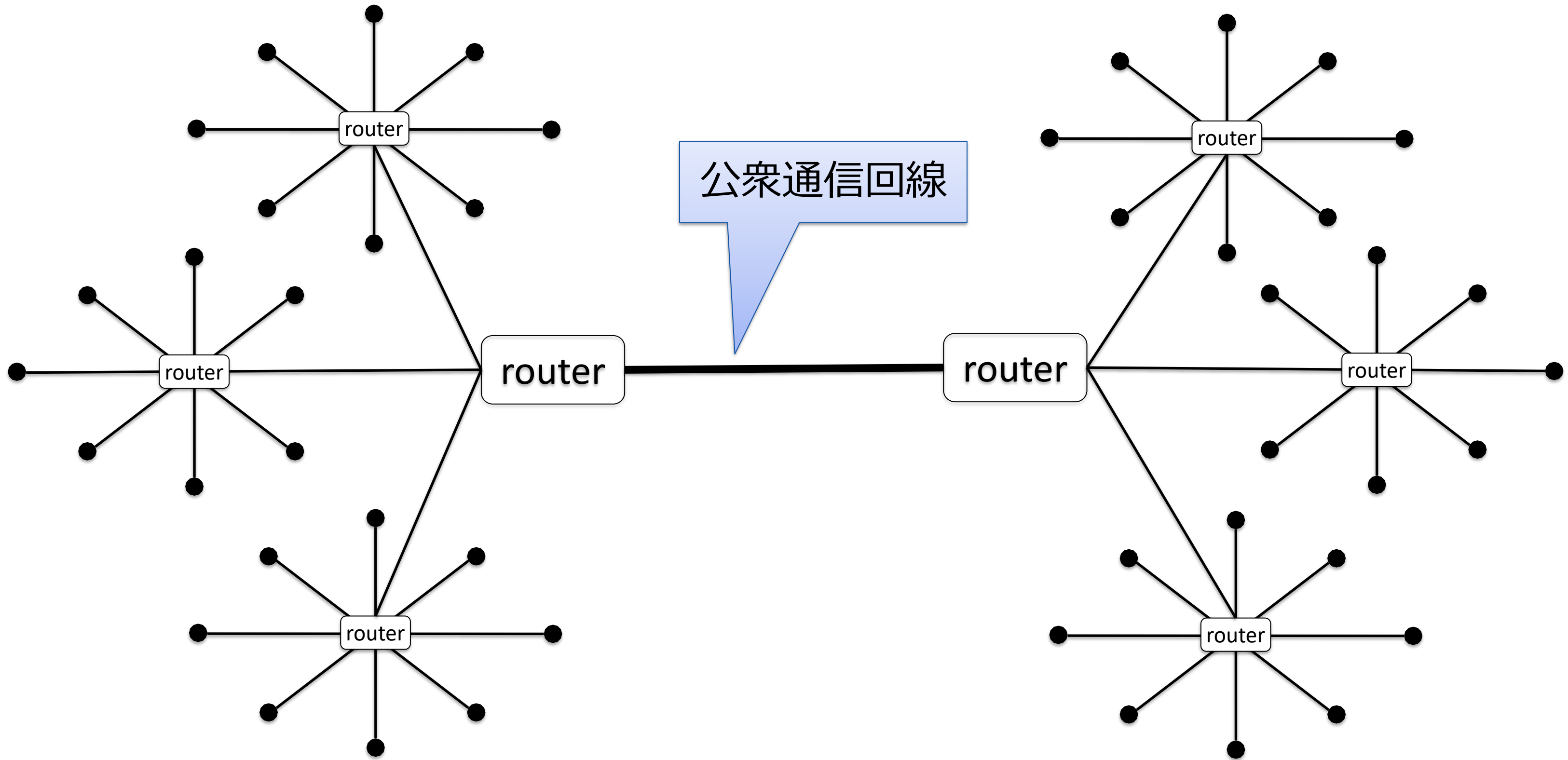




# Elastic Computing ? : Automation API の場合

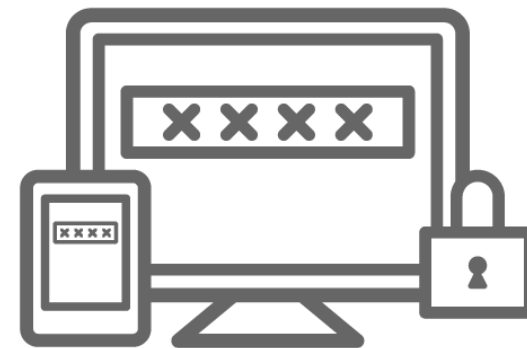
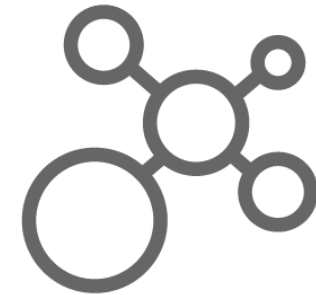


# インターネット通信網の利用



# デスクトップ製品のアドイン開発者の方へ

- デスクトップ開発と Web 開発の違いをご理解ください
  - Web アプリの特性 – セキュリティ視点
  - 通信経路
  - 呼び出し数制限
  - 非同期処理
  - 仮想環境
  - オンプレミスとの違い



# Rate Limit 厳守のお願い

- 1分間あたりのエンドポイント呼び出し数制限
  - 429 レスポンス ステータス受信時に適切に対応する必要あり
  - 例) Automation API WorkItem 実行時のポーリング





# Revit Automation の理解

# クラウド上でデスクトップ製品アドイン実行環境を提供

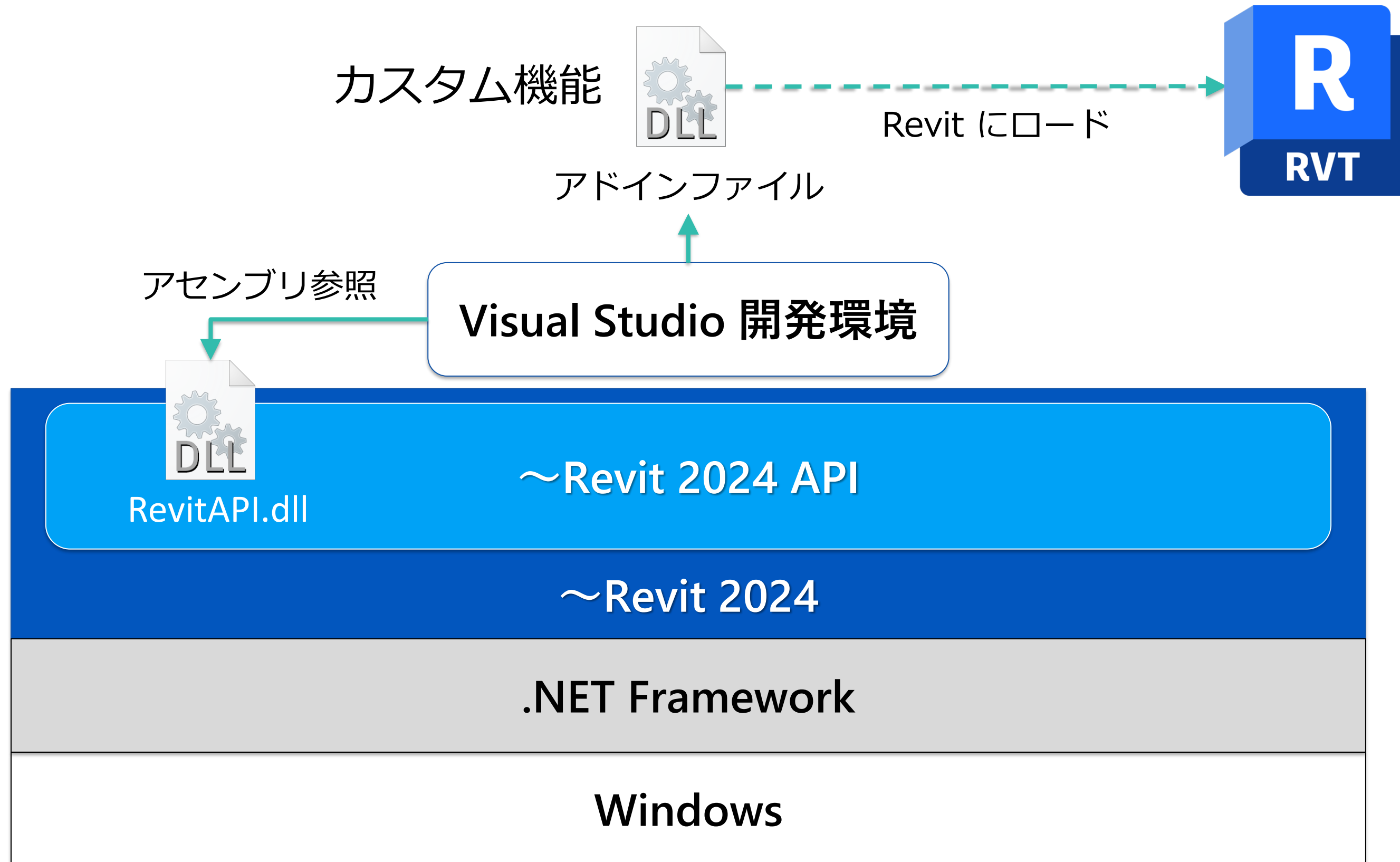
- クラウド上の Revit エンジンリモートで起動
- 起動した Revit エンジンにアドインをロードして実行
- 成果物（デザインファイルや関連ファイル）を作成
- 成果物をダウンロードして利用

## ➤ Automation API

# Automation API の誤った認識

- 製品を対話/対面操作するシンククライアント環境ではない
  - リモートデスクトップのような仕組みはありません
- Web ブラウザ上で CAD を実現する SDK ではない
  - フロント UI となる Web ページは HTML で実装が必要です
  - APS Viewer は Automation API には含まれません
- 処理時の利用ファイルはローカル PC との間で直接入出力不可
  - パブリック クラウド ストレージの利用が必須です
  - Automation API 仮想環境へダウンロード/からアップロードします

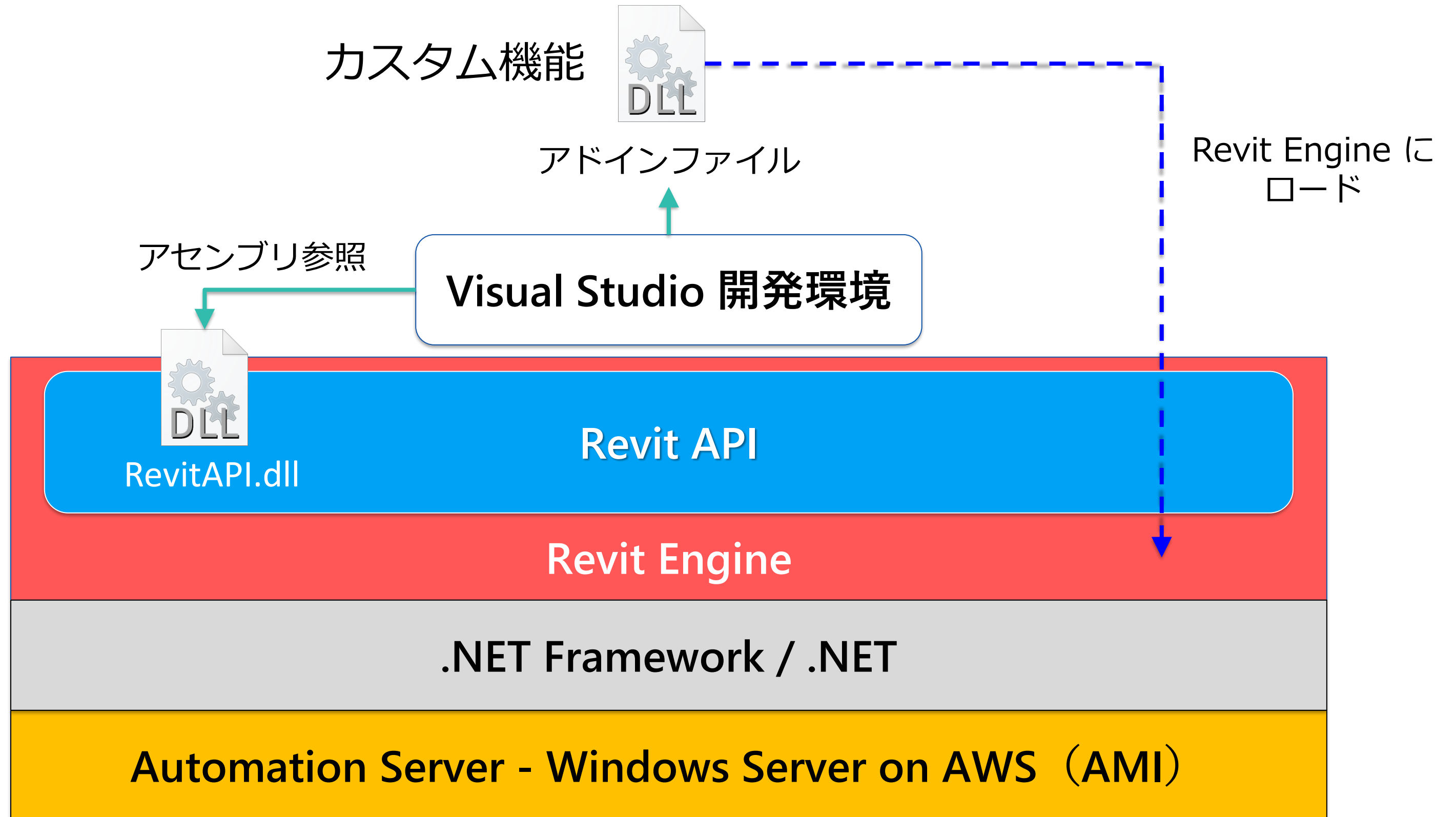
# Revit アドインの仕組み (Revit 2024 まで)



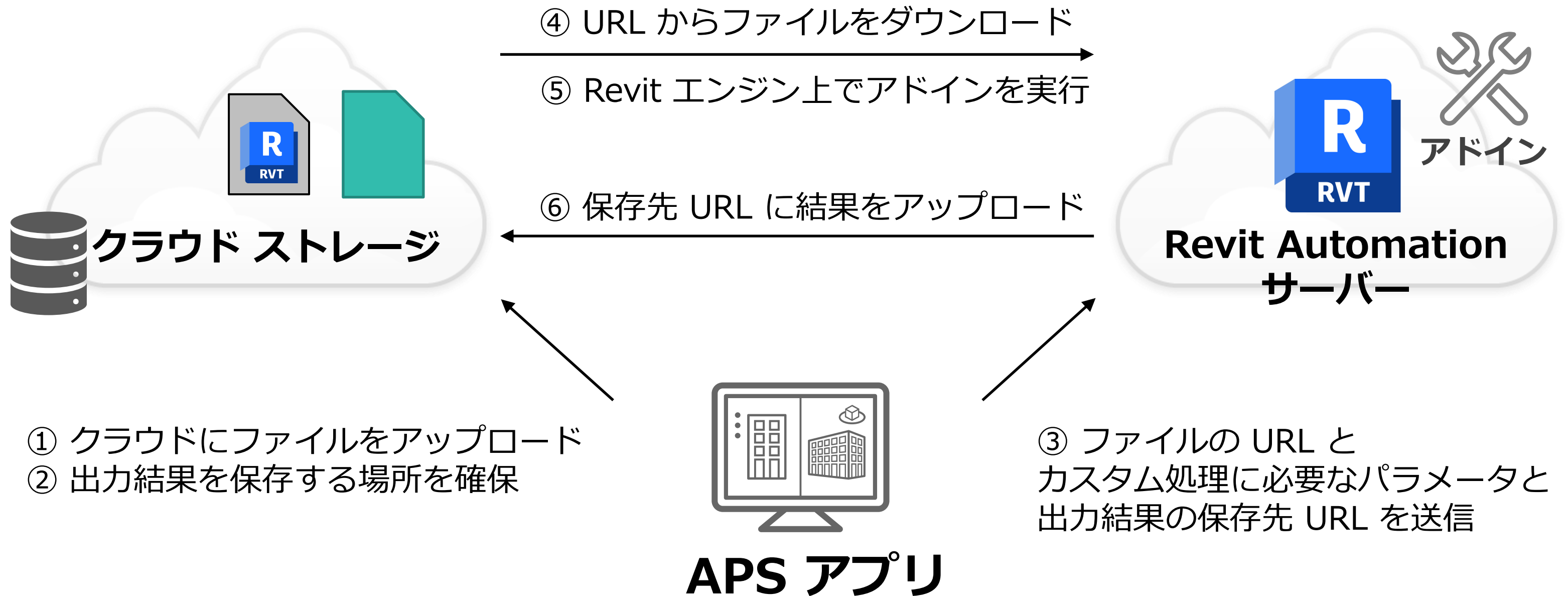
# Revit アドインの仕組み (Revit 2025 以降)



# Revit Automation の仕組み



# Revit Automation 利用のワークフロー



# Automation API を正しく理解しましょう

## 対話的な表示/編集機能は ありません

- Revit Web 版 (Revit UI 画面) のようなものではありません

## ビューア機能は ありません

- 必要に応じて APS Viewer の利用を検討出来ます

## エンドユーザ向けのサービスでは ありません

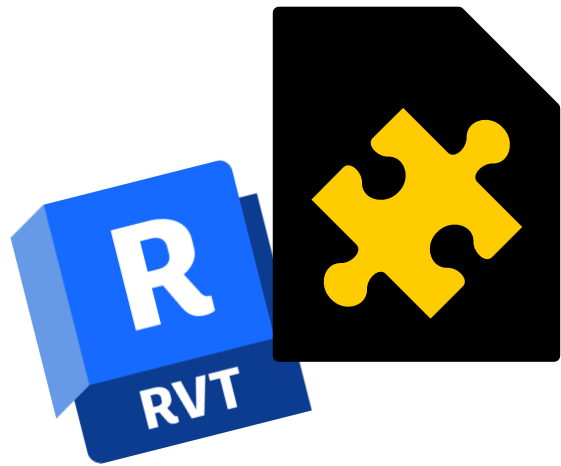
- 開発者向けのサービスです

## サーバー モジュールでは ありません

- オンプレミス(プライベート)サーバー版はありません

# Automation API の用語

## AppBundle



Revit アドインのパッケージ

Id : CustomApp

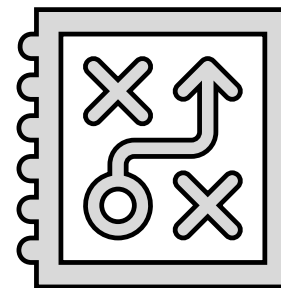
Engine : 2024

Description: Test custom app

Package: Storage URL

Revit の .NET API で作成したアセンブリや関連ファイルを ZIP 圧縮してアップロードし、AppBundle として登録する。

## Activity



実行されるアクションの定義

Id: UpdateParamActivity

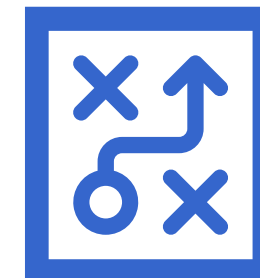
Input Parameter A : RVT, TXT

Output Parameter B: RVT

AppBundle: CustomApp

カスタム処理の雛型を定義する。  
.NET アセンブリ内でどんなデータを入力して、どんなデータを出力するか定義する。

## WorkItem



指定のアクションを呼び出すジョブ

Id: 返却される文字列

Activity : UpdateParamActivity

Input Parameter A : File URL

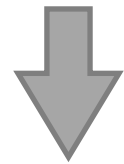
Output Parameter B: Storage URL

REST API でリクエストするジョブ。  
対象のモデルやテキストデータ、出力先の URL と、実行する Activity を指定する。

# AppBundle と Activity の登録

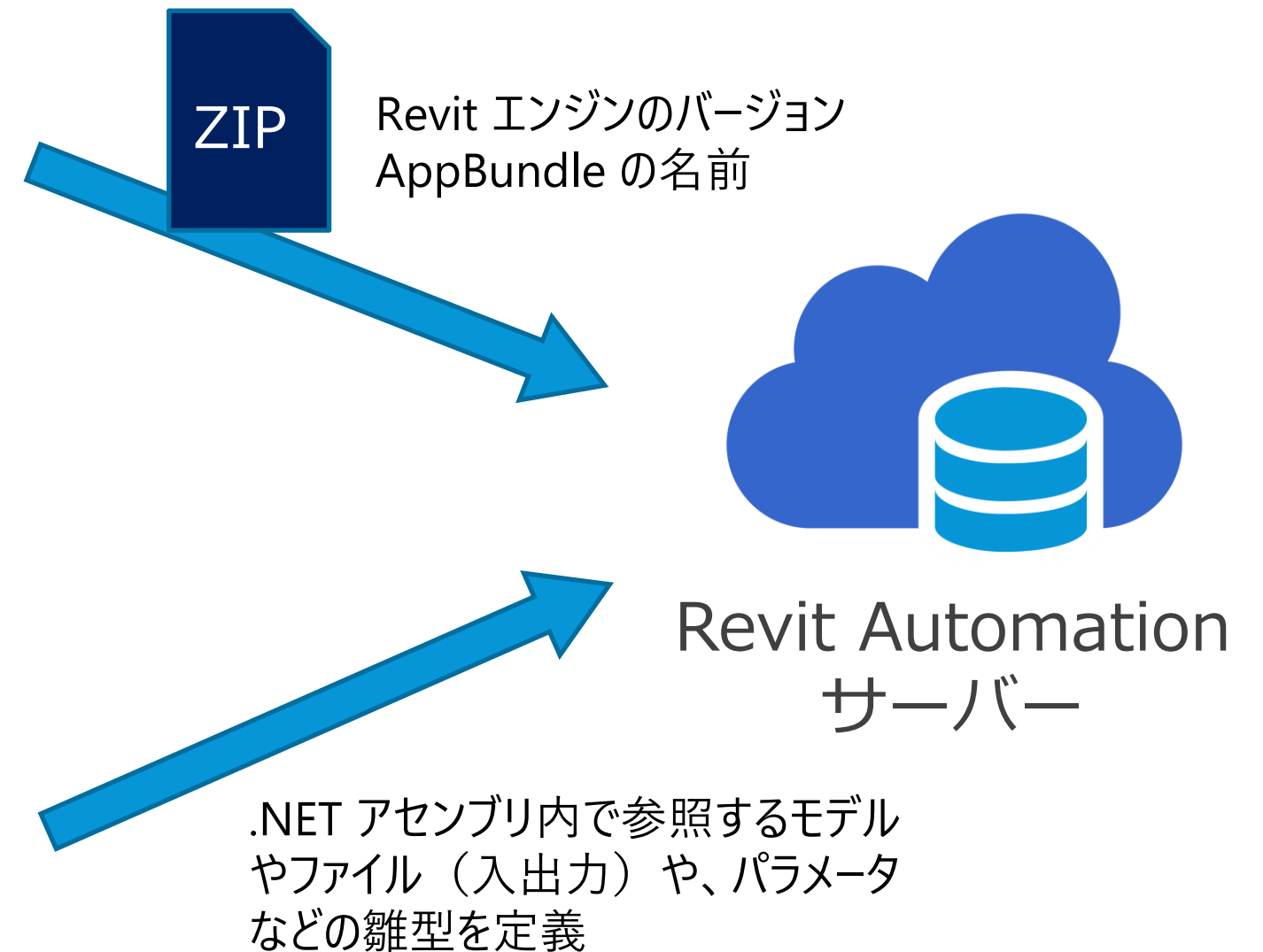
## AppBundle を登録

1. 新しい AppBundle を作成
2. バンドルパッケージをアップロード
3. エイリアスとバージョンを設定



## Activity を登録

1. 新しい Activity を作成
2. エイリアスとバージョンを設定
3. 必要があればロケールを設定



# Alias と Version の設定

- AppBundle と Activity は、それぞれにバージョンを設定します。
- 特定のバージョンに任意のラベルで名前をつけてエイリアスを作成します。

	Version	Alias
AppBundle	1	
	2	Production
	3	
	4	Test
	5	Development

	Version	Alias
Activity	1	
	2	
	3	Production
	4	Beta
	5	Test
	6	Development

- 形式: YourNickname.SomeAppBundleId+SomeAliasName
- 例: MyFirstForgeAppNickname.DeleteWallApp+test

例えば、Activity のテストを行う際に、Test エイリアスを作成してテストを実行する。ただし、AppBundle は、稼働中の Production のエイリアスを呼び出す、といった開発・テスト用の使い方ができます。

# Automation API で扱う ID

- AppBundle と Activity で利用

- Client ID に関連付けられる

- 例)

"nqpwqsDLFGkSO6LgA2mvaSXY5AeH5VSJ.UpdateRVTParam+dev"



- App 名

- AppBundle と Activity 登録時に ID に指定


- Alias 名

- AppBundle と Activity バージョン識別に利用

- 例) **prod**、**dev**

- WorkItem は作成時に個別 ID を割り当て

# ID と Nickname (ニックネーム)

- Client ID に関連付けられた AppBundle と Activity の ID
  - 例)  
**"nqpwqsDLFGkS06LgA2mvaSXY5AeH5VSJ.UpdateRVTPParam+dev"**
- Nickname のマッピング登録が可能 (任意)
  - 例)  
"nqpwqsDLFGkS06LgA2mvaSXY5AeH5VSJ" を "DAS\_Japan\_Revit"  
  
**"DAS\_Japan\_Revit.UpdateDWGParam+dev"**
  - 注意 : Nickname 登録後 Client ID 名指定は使用不可に
  - Nickname は 20 文字以内で、a-z, A-Z, 0-9, \_ から作成

# WorkItem を作成する

- Activity の定義に基づいて、実際のファイルのストレージ URL やパラメータを設定して、WorkItem を作成します。
- Automation サーバーは、**一時作業領域**を作成し、ファイルを取得、Revit エンジンを実行して、AppBundle を展開してカスタム処理を実行します。



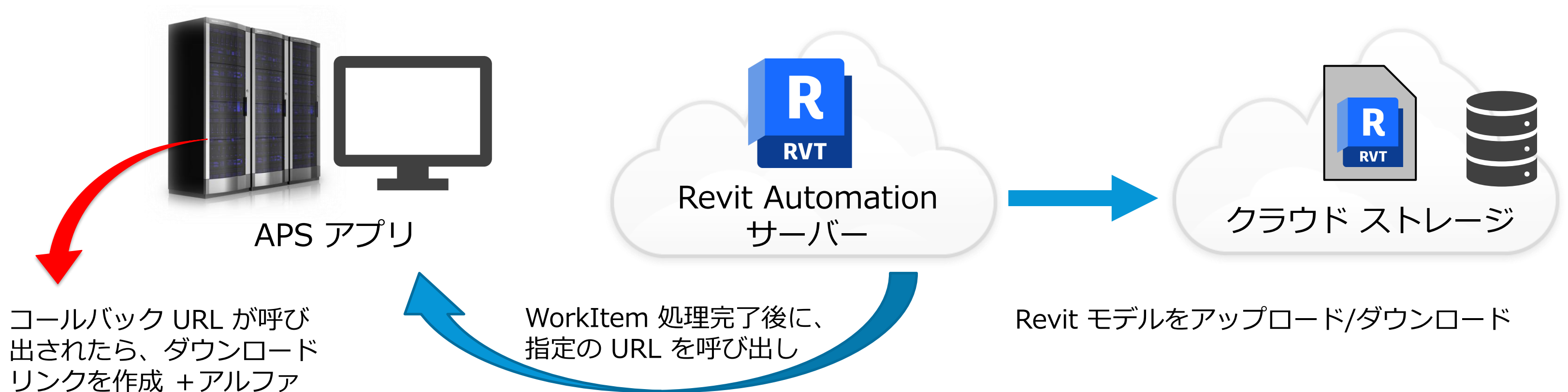
## WorkItem を作成

- 実行する Activity を指定
- モデル・ファイルの URL を設定
- パラメータを設定

## Revit モデルや関連ファイルのアップロード/ダウンロード

# WorkItem の完了通知

- WorkItem を POST する際に、“**onComplete**” という引数にコールバック URL を設定することができ、WorkItem の完了時に、自動的に呼び出してくれます。
- コールバック URL には、**クエリパラメータ**を組み合わせたことができます。



# Automation API とは？


- **起動したコアエンジンにアドインをロード/実行**
  - アドイン+自動ロード定義を ZIP 圧縮したもの = AppBundle
  - AppBundle は WorkItem 実行前に登録が必要
- **成果物としてデザインファイルや関連ファイルの作成してダウンロードして利用**
  - Activity で入出力ファイル/パラメータを宣言
  - Activity は WorkItem 実行前に登録が必要
  - WorkItem (ファイル/パラメータ渡しを実行)
- **クラウド上の CAD コアエンジンをリモートで起動**
  - WorkItem (実際の処理実行)

# Automation API で利用可能なコアエンジン バージョン

- デスクトップ製品と同期したコアエンジン バージョン
  - AppBundle はエンジン バージョンに合わせた作成が必須
  - コアエンジン ID の形式はコアエンジン毎に異なる
  - 最新 + 過去 3 バージョンがサポート対象

<b>“Autodesk.Revit.2023”</b>	⇒	Revit 2023	}	.NET Framework 4.8
<b>“Autodesk.Revit.2024”</b>	⇒	Revit 2024		
<b>“Autodesk.Revit.2025”</b>	⇒	Revit 2025	}	.NET 8 (.NET Core)
<b>“Autodesk.Revit.2026”</b>	⇒	Revit 2026		

- AppBundle と Activity の登録時に指定



# デベロッパキーの取得

# APS を始めるには?

- まずは <https://aps.autodesk.com/> へ

1



2



3

Autodesk ID で  
サインイン

APS アプリ作成

開発

- 目的はデベロッパ キーの取得
  - **Client ID** (別名 : Consumer Key)
  - **Client Secret** (別名 : Consumer Secret)

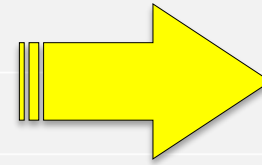
# My applications からのアプリ登録

## どのAPIを使うか指定

### API Access

These are the APIs your app is allowed to access. It can be changed anytime.

- AEC Data Model API
- Application Management API
- Autodesk Construction Cloud API
- BIM 360 API
- BuildingConnected API
- Data Exchange API
- Data Management API
- Design Automation API
- Flow Graph Engine API
- Manufacturing Data Model API
- Model Derivative API
- Parameters API
- Premium Reporting API
- Reality Capture API
- Tandem Data API
- Token Flex Usage Data API
- Webhooks API



# My applications からのアプリ登録

## 課金対象のチーム

### Team Assignment

Assign an [Autodesk team](#) to pay for any charges incurred by this application

Toshiaki Isezaki - 2013

Assign

アカウントで複数のチームを編成している場合には、プライマリ管理者、または、セカンダリ管理者に適切なチームを確認してください。通常はアカウントに1つのチームしか関連付けられていないので、表示されるチームを割り当て（Assign）ます。



← Back to Applications

# designAutomationSample

App settings

Token usage

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

[Redacted Client ID]

Client Secret

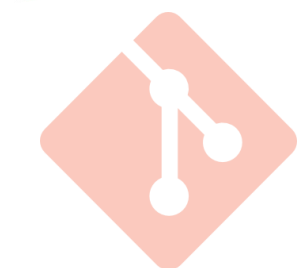
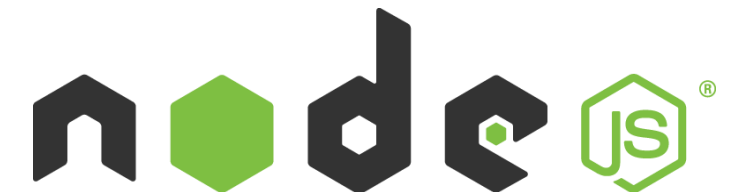
.....

**Regenerate**

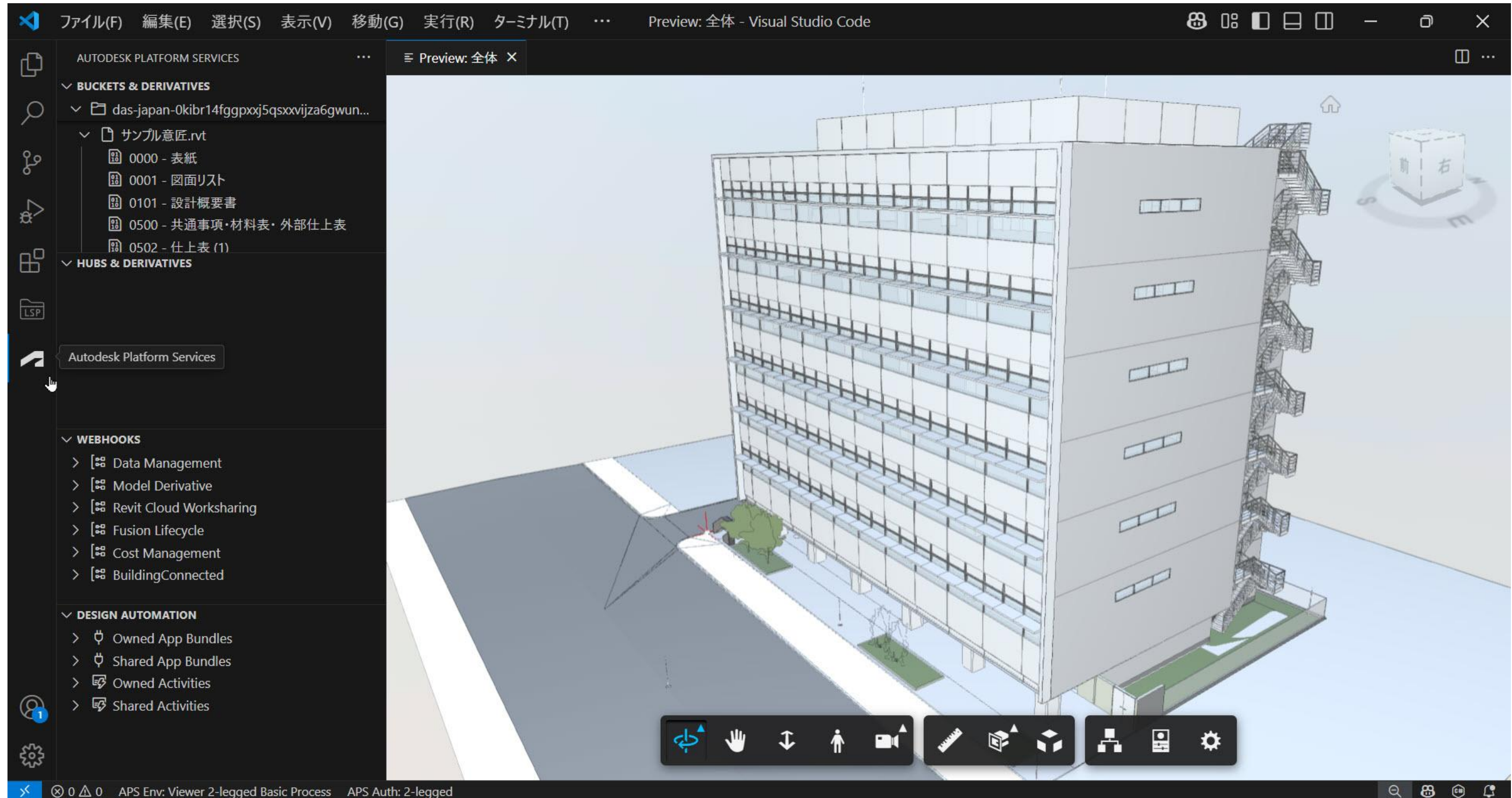
# 今回使用する開発環境

## これに限定するものではありません

- Web ブラウザ : **Google Chrome**
- HTML/JavaScript エディタ : **VS Code**
- Web サーバー実装 : **Node.js**
- Revit アドイン C# エディタ : **Visual Studio 2019/2022 Professional**
- テスト ツール : **Postman**
- リポジトリ ユーティリティ : **git for Windows**



# VS Code - APS エクステンション



# RESTful API のテストツール



- Postman
  - <https://www.getpostman.com/>
  - コードを書かずに RESTful API のテストが可能

POSTMAN


PRODUCT SOLUTIONS PLANS & PRICING DOCS API NETWORK SIGN IN

## Postman Makes API Development Simple.

Developers use Postman to build modern software for the API-first world.

Download the App

Want to improve your Postman skills? [Take the next step!](#)



# サーバーの作成

# サーバーの作成 – Create Server

**AUTODESK**  
Platform Services

Search

Sign in

Solutions ▾ Getting Started Documentation ▾ Success Stories Community ▾ Support ▾ Pricing App Store ▾

Getting Started  
Environment Setup  
**Tutorials**  
Simple Viewer  
Hubs Browser  
Dashboard  
**Design Automation**  
**Create Server**  
Basic UI  
Create Plugin  
Define Activity  
Execute Workitem

Home > Tutorials > Design Automation > Create Server

## Create Server

Your **Client ID** & **Secret** should be protected and kept confidential as all your files will be bound to your account. For a web application, keep it on your server. This section demonstrate how to prepare create a local development server.

Please review Environment Setup section for required software.

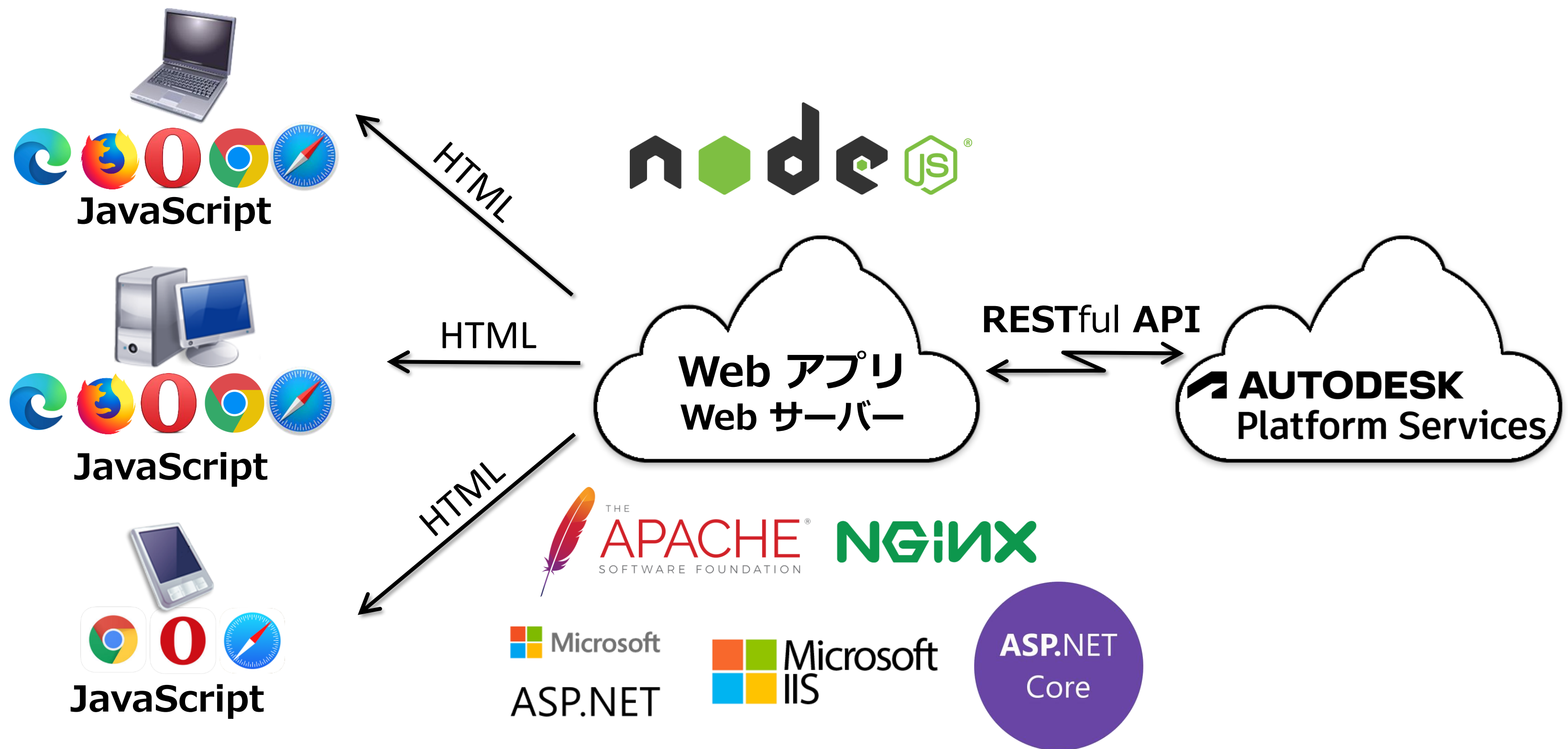
### Setup Project

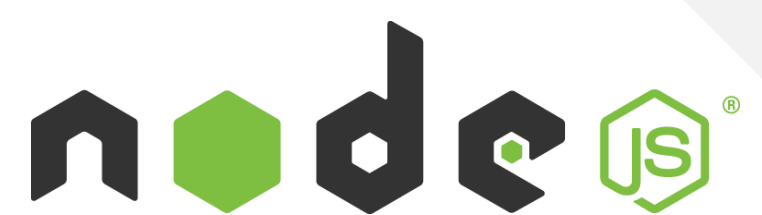
**Node.js & VSCode** .NET & VSCode .NET & VS2022

Setup Project  
Application Config  
Server Setup

Create new folder for your application and set it in the command line and

# 一般的な Web アプリの構成





# Node.js とは

- オープンソース
- JavaScript をサーバー上で実行するための環境
- Google V8 JavaScript Engine が使用
- Node Package Manager で拡張可能
- 今回使用しているパッケージ（ミドルウェア）の一部：
  - **autodesk.forge.designautomation**
  - cookie-session
  - express
  - express-formidable
  - **forge-apis**
  - socket.io

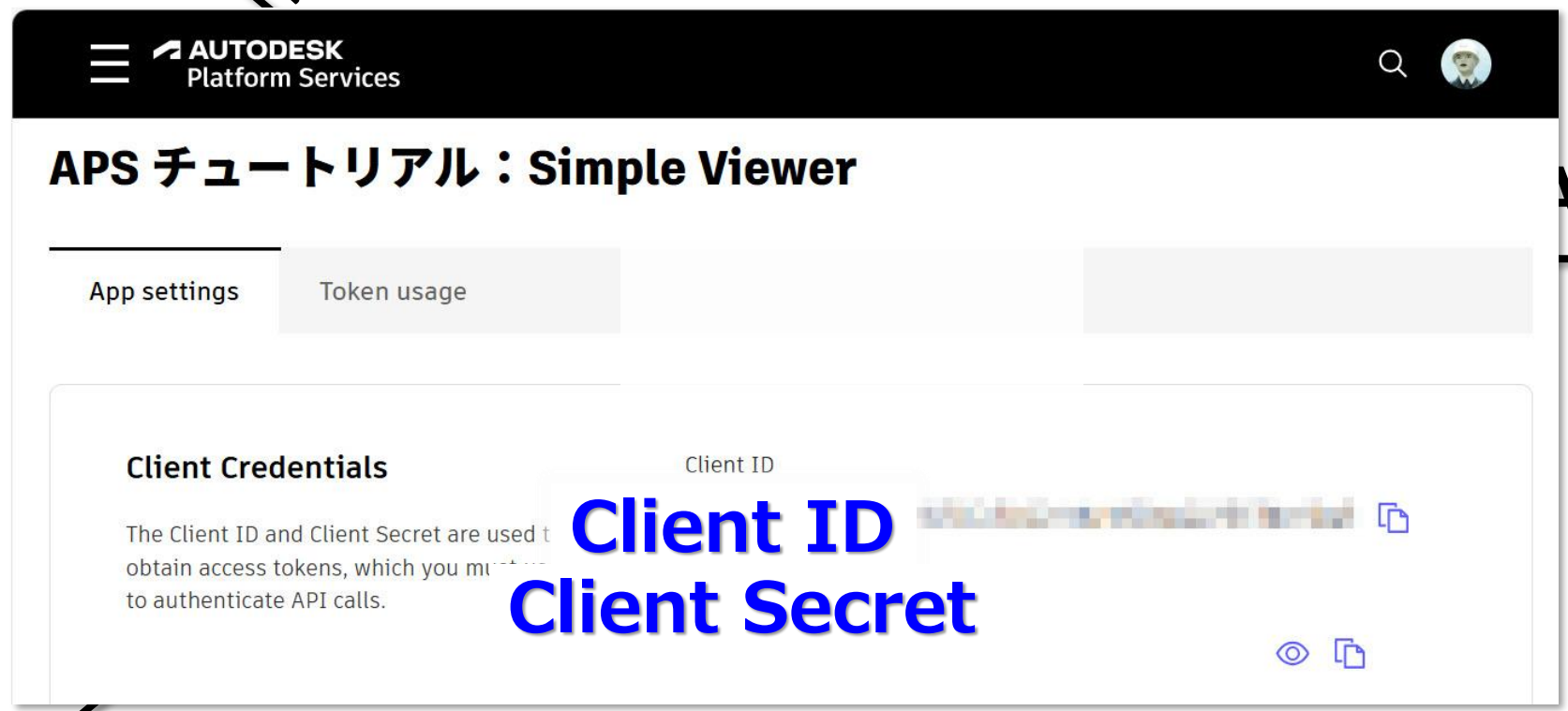
**Forge Design Automation SDK - v3用**

**Forge SDK**

# デベロッパ キーとアクセス トークン

## APS ポータル (aps.autodesk.com)

HTML



API



# アクセス トークン (Access Token) とは

- **クラウド リソース** へのアクセス権限をチェックする仕組み
  - **有効期限**が設定される
- 生成に必要なもの
  - **デベロッパキー**
    - **Client ID** と **Client Secret** のペア
    - APS ポータルでアプリ作成時に取得可能
  - **スコープ (Scope)**
    - リソースへのアクセス権限を指定
    - 使用するリソースによって適切に使い分けが必要
    - 複数の Scope 文字列を半角スペースで結合して指定

# Automation API Tutorial の 認証フロー

## Data Management

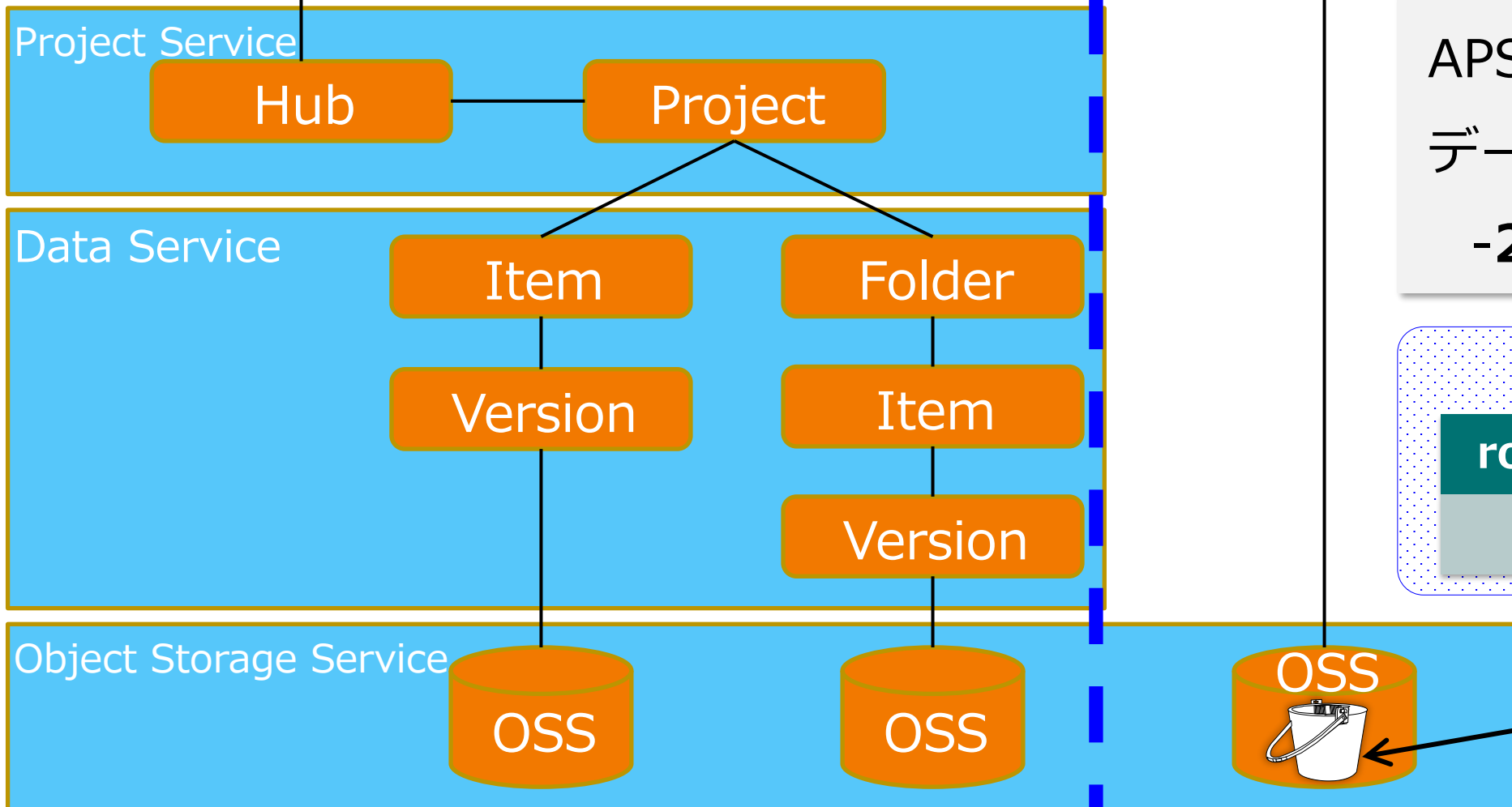
## OSS

APS アプリ

APS アプリ

3-legged 認証フロー

2-legged 認証フロー



APS アプリが Fusion Team、ACC のユーザーデータ領域に直接アクセスして運用  
**- 3 -Legged OAuth -**

APS アプリが Bucket でデータを直接管理・運用  
**- 2-Legged OAuth -**

**サーバー実装**

```

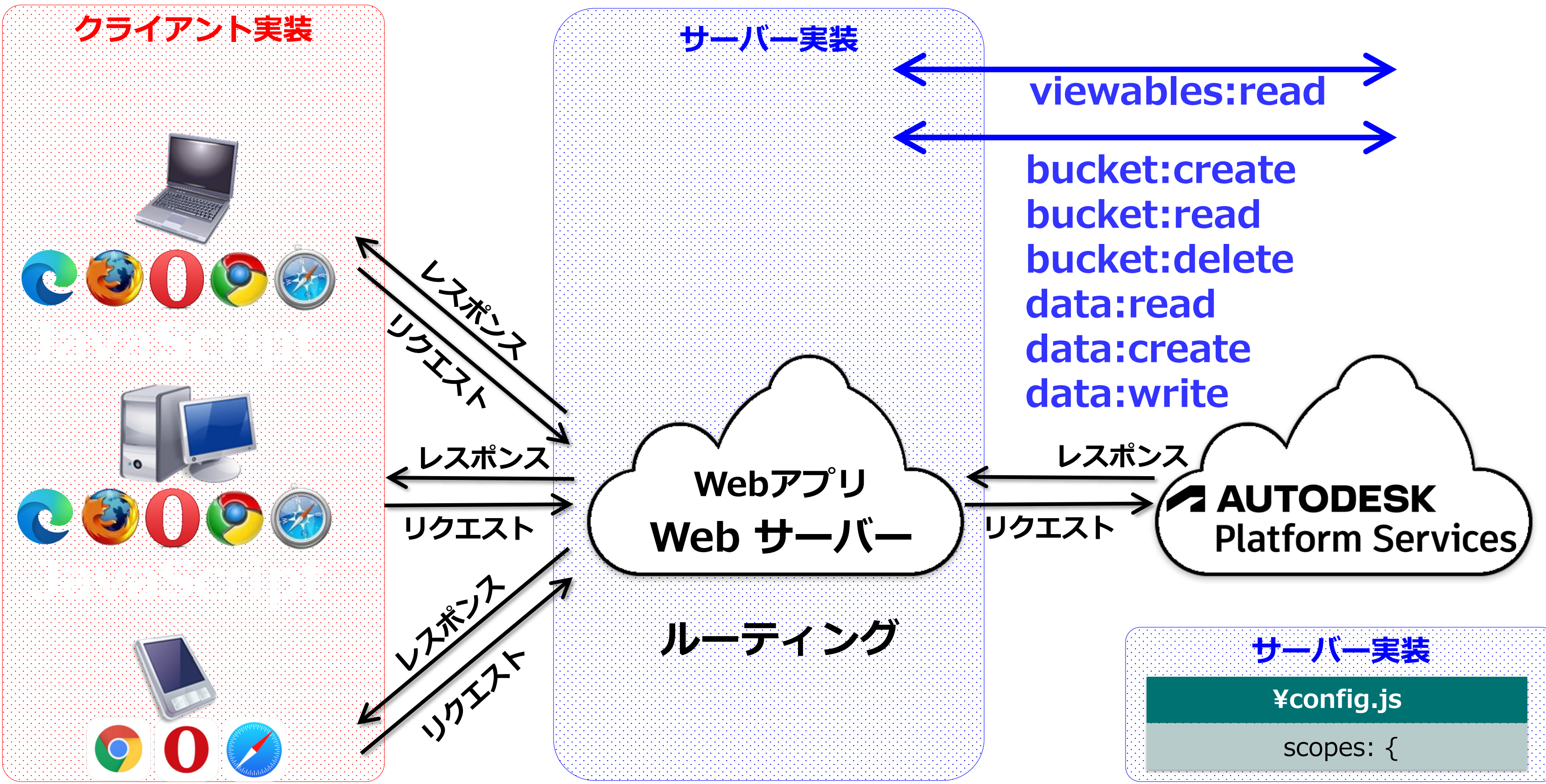
routes/common/oauth.js
getClient()
  
```

Bucket

# Automation API Tutorial が使用する Scope

Scope 文字列	意味
user-profile:read	プロフィール (Autodesk ID) の表示
user:read	プロフィール (Autodesk ID) の読み取り
user:write	プロフィール (Autodesk ID) の書き込み
<b>viewables:read</b>	<b>変換後のデザインデータ (SVF) の読み取り (表示)</b>
data:read	ストレージ データの読み取り
data:write	ストレージ データの書き込み (編集)
data:create	ストレージ データの作成
data:search	ストレージ データの検索
<b>bucket:create</b>	<b>新しい Bucket の作成</b>
<b>bucket:read</b>	<b>Bucket の読み取り</b>
<b>bucket:delete</b>	<b>Bucket の削除</b>
bucket:update	Bucket の更新
<b>code:all</b>	<b>コードの生成または実行 (Automation API)</b>
account:read	アプリやサービス アカウントの読み取り
account:write	アプリやサービス アカウントの書き込み

# スコープによるトークンを使い分け



# コマンド プロンプト/VS Code 上の操作手順

1. プロジェクト フォルダの作成と Node.js の初期化
2. Node.js パッケージをインストール、**package.json** ファイルを確認
3. VS Codeでプロジェクトフォルダを開き **routes**、**wwwroot** フォルダを作成
4. Node.js 構成 を追加、**launch.json** ファイルに Client ID/Secret を指定
5. ルートフォルダに **start.js** ファイルの作成と実装
6. ルートフォルダに **server.js** ファイルの作成と実装
7. ルートフォルダに **socket.io.js** ファイルの作成と実装
8. ルートフォルダに **config.js** ファイルの作成とサーバー実装記述の追加
9. routes/common/**oauth.js** ファイルの作成と実装

Getting Started

Environment Setup

**Tutorials**

Simple Viewer

Hubs Browser

Dashboard

**Automation**

**Create Server**

Basic UI

Create Plugin

Define Activity

Execute Workitem

ACC Administrator

ACC Issues (beta)

Learn More

# Create Server

Your `Client ID` & `Secret` should be protected and kept confidential as all your files will be bound to your account. For a web application, keep it on your server. This section demonstrate how to prepare create a local development server.

Please review Environment Setup section for required software.

## Setup Project

**Node.js & VSCode**   .NET & VSCode   .NET & VS2022

Create a new folder for your project, navigate to it in the command line, and initialize a new Node.js project:

```
mkdir designAutomationSample
cd designAutomationSample
npm init -y
```

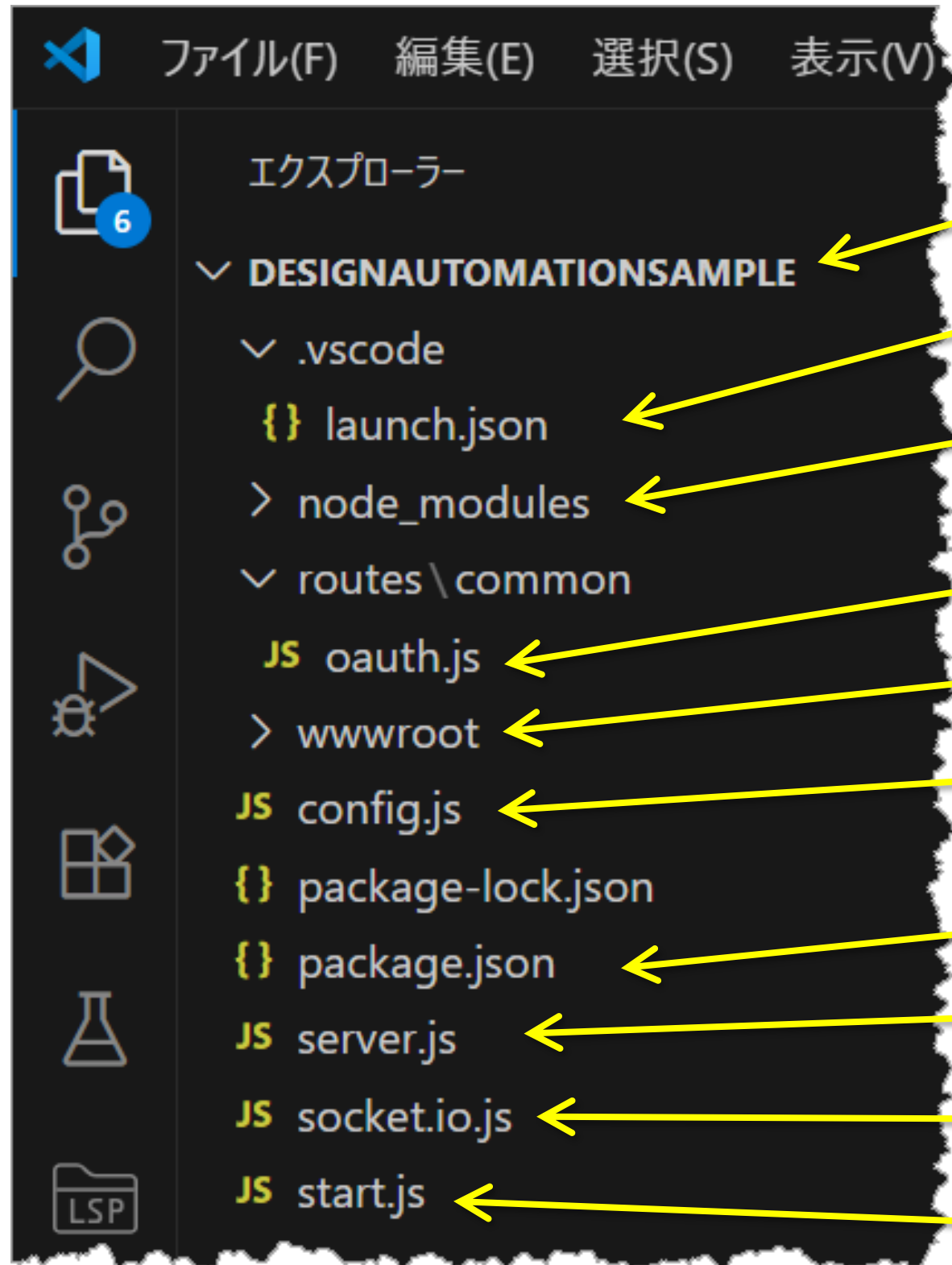
Next, install all the Node.js dependencies we're going to use. In this case it will be the `Express.js` framework, an `Express.js` `middleware` for handling `multipart/form-data` requests, and finally the `APS SDK`:

```
npm install --save express express-formidable forge-apis
```

- Setup Project
- Application Config
- Server Setup



# ここまでのプロジェクト構成



## サーバー実装

- プロジェクト フォルダ
- VS Code デバッグで使用
- 依存関係からインストールされた Node.js パッケージ
- 2-legged 認証実装
- クライアント構成 (未実装)
- スコープ等認証情報情報
- Node.js パッケージ依存関係
- セッション クッキー等情報
- ソケット通信接続/切断実装
- Node.js サーバーの起動構成



# 基本 UI

# Basic UI - 基本 UI

The screenshot shows the Autodesk Platform Services documentation page for 'Basic UI'. The page has a dark header with the Autodesk logo and 'Platform Services' text. A search bar is located in the center of the header, and a 'Sign in' button is on the right. Below the header is a navigation bar with links for 'Solutions', 'Getting Started', 'Documentation', 'Success Stories', 'Community', 'Support', 'Pricing', and 'App Store'. The main content area is divided into a left sidebar and a main content area. The sidebar contains a 'Getting Started' section with links for 'Environment Setup', 'Tutorials', 'Simple Viewer', 'Hubs Browser', and 'Dashboard'. The 'Tutorials' section is expanded, showing 'Design Automation' as the current category, with 'Basic UI' selected. Below 'Design Automation' are links for 'Create Server', 'Basic UI', 'Create Plugin', 'Define Activity', and 'Execute Workitem'. The main content area shows a breadcrumb trail: 'Home > Tutorials > Design Automation > Basic UI'. The title 'Basic UI' is prominently displayed. The text explains that the interface is based on vanilla HTML5 & JavaScript and mentions differences in Websocket implementation using socket.io (Node.js) or SignalR (.NET 6). It instructs the user to create an index.html file in the wwwroot folder. Below the text are three tabs: 'Node.js & VSCode', '.NET & VSCode', and '.NET & VS2022'. The 'Node.js & VSCode' tab is active. A code editor shows the file path 'wwwroot/index.html' and the beginning of the HTML document: '<!DOCTYPE html>'. The page is styled with a dark theme and a torn paper effect at the bottom.

**AUTODESK**  
Platform Services

Search

Sign in

Solutions ▾ Getting Started Documentation ▾ Success Stories Community ▾ Support ▾ Pricing App Store ▾

Getting Started  
Environment Setup  
**Tutorials**  
Simple Viewer  
Hubs Browser  
Dashboard  
**Design Automation**  
Create Server  
**Basic UI**  
Create Plugin  
Define Activity  
Execute Workitem

Home > Tutorials > Design Automation > Basic UI

ApsDesignAutomation.js

## Basic UI

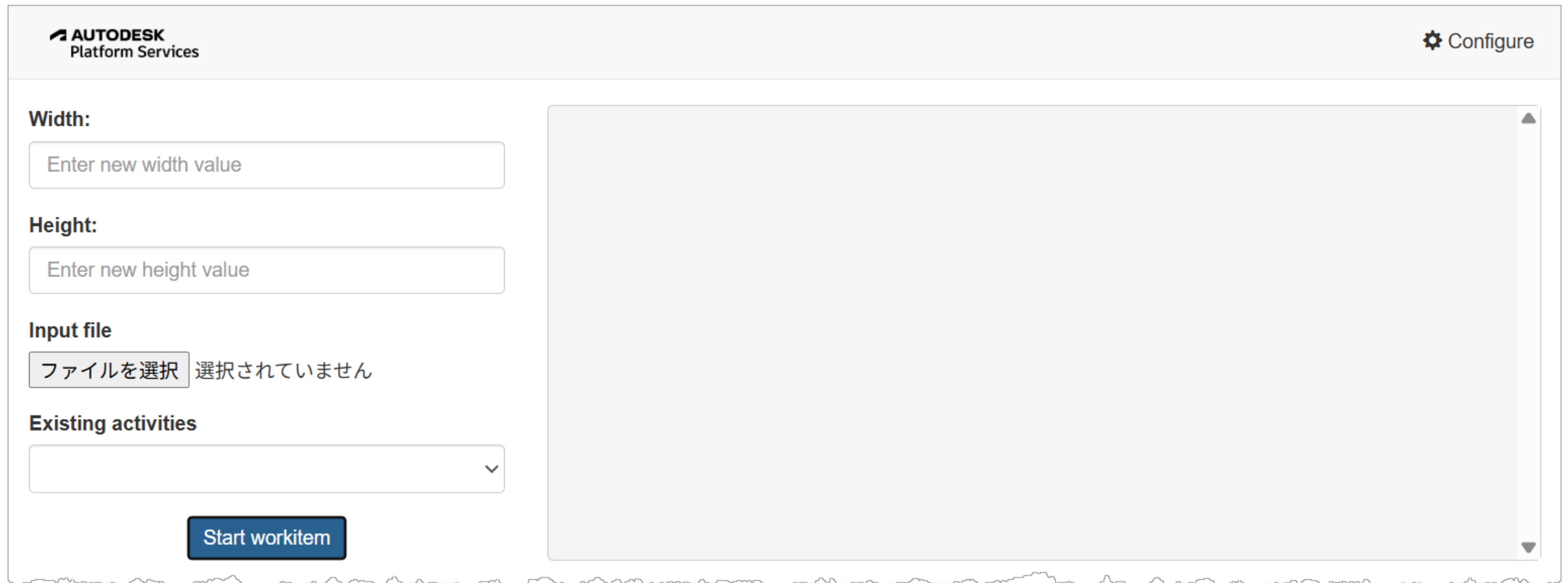
The interface is based on vanilla HTML5 & JavaScript. It essentially the same for any server-side, but there are a few differences: the Websocket implementation uses socket.io (Node.js) or SignalR (.NET 6). Let's start with the UI (HTML and JavaScript) files. Under the wwwroot folder, create an index.html file with the following content:

Node.js & VSCode .NET & VSCode .NET & VS2022

```
wwwroot/index.html  
<!DOCTYPE html>
```

# VS Code 上の操作手順

1. **wwwroot** フォルダに **index.html** ファイルを作成して実装
2. **js** サブフォルダを作成、**ApsDesignAutomation.js** ファイルを作成して実装
3. デバッグ実行 >> [localhost:8080](http://localhost:8080)



The screenshot shows the Autodesk Platform Services configuration interface. At the top left is the Autodesk logo and 'Platform Services'. At the top right is a 'Configure' button with a gear icon. The main area contains several input fields and a large empty workspace. The 'Width:' field has a placeholder 'Enter new width value'. The 'Height:' field has a placeholder 'Enter new height value'. The 'Input file' section has a button 'ファイルを選択' and the text '選択されていません'. The 'Existing activities' section has a dropdown menu. At the bottom center is a blue 'Start workitem' button.

On this page

# Basic UI

The interface is based on vanilla HTML5 & JavaScript. It essentially the same for any server-side, but there are a few differences: the Websocket implementation uses socket.io (Node.js) or SignalR (.NET 6). Let's start with the UI (HTML and JavaScript) files. Under the wwwroot folder, create an index.html file with the following content:

- Node.js & VSCode
- .NET & VSCode
- .NET & VS2022

```
wwwroot/index.html
<!DOCTYPE html>
<html>
  <head>
    <title>APS - Design Automation</title>
    <meta charset="utf-8" />
    <link rel="shortcut icon" href="https://cdn.autodesk.io/favicon.ico" />
    <!-- Common packages: jQuery, Bootstrap -->
    <script src="//cdnjs.cloudflare.com/ajax/libs/jquery/3.3.1/jquery.min.js"></script>
    <script src="//cdnjs.cloudflare.com/ajax/libs/twitter-bootstrap/3.4.1/js/bootstrap.min.js"></script>
    <link rel="stylesheet" href="https://cdn.autodesk.io/css/bootstrap.min.css">
  </head>
  <body>
    <div class="container">
      <div class="row">
        <div class="col">
          <h1>Design Automation</h1>
        </div>
      </div>
    </div>
  </body>
</html>
```

designAutomationSample

EXPLORER

- DESIGNAUTOMATIONSAM...
  - .vscode
    - launch.json
  - node\_modules
  - routes\common
    - oauth.js
  - wwwroot
    - config.js
    - package-lock.json
    - package.json
    - server.js
    - socket.io.js
    - start.js

routes > common > JS oauth.js > ...

```

15  */
16  async function getClient(scopes) {
17    scopes = scopes || config.scopes.internal;
18    const key = scopes.join("+");
19    if (cache[key]) return cache[key];
20
21    try {
22      const { client_id, client_secret } = config.credentials;
23      let client = new AuthClientTwoLegged(
24        client_id,
25        client_secret,
26        scopes || config.scopes.internal,
27        true
28      );
29      let credentials = await client.authenticate();
30      cache[key] = client;
31      console.log(`OAuth2 client created for ${key}`);
32      return client;
33    } catch (ex) {
34      return null;
35    }
36  }
37
38  module.exports = {
39    getClient,
40  };

```

PROBLEMS OUTPUT C#

Filter

```

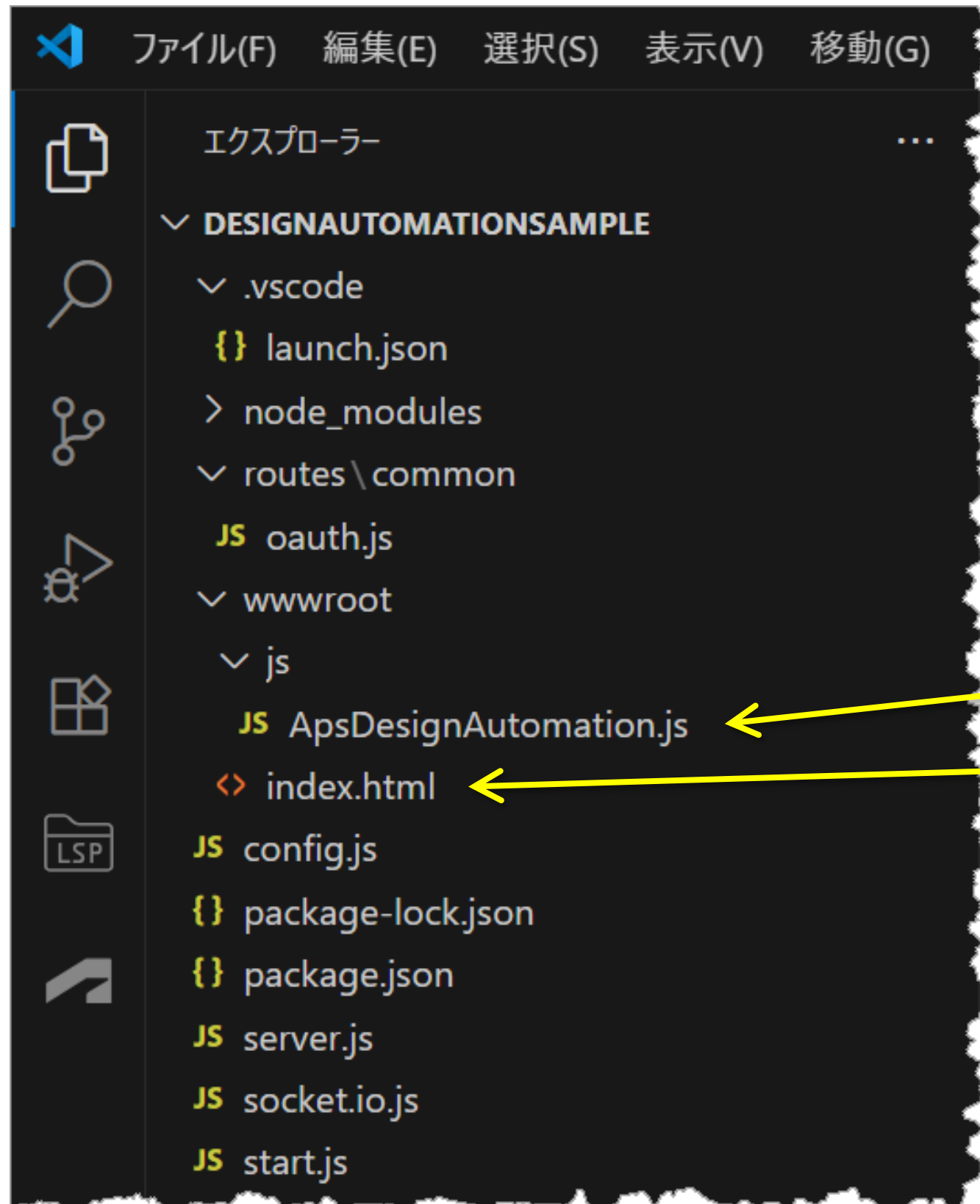
2025-10-10 15:18:12.549 [info] Installing C# dependencies...
2025-10-10 15:18:12.549 [info]
2025-10-10 15:18:12.549 [info] Platform: win32, x86_64
2025-10-10 15:18:12.549 [info]
2025-10-10 15:18:12.549 [info] Downloading package 'Language server for Roslyn Copilot integration'
2025-10-10 15:18:13.185 [info] (47 KB)

```

OUTLINE TIMELINE

No Solution Ln 40, Col 3 Spaces: 4 UTF-8 CRLF {} JavaScript


# ここまでのプロジェクト構成



## クライアント実装

Web ページ スクリプト実装

Web ページ UI 実装



# プラグインの作成

# プラグインの作成 – Create Plugin

**AUTODESK**  
Platform Services

Search

Sign in

Solutions ▾ Getting Started Documentation ▾ Success Stories Community ▾ Support ▾ Pricing App Store ▾

Getting Started  
Environment Setup  
**Tutorials**

Home > Tutorials > Design Automation > Create Plugin

## Create Plugin

Prerequisites  
Additional prerequisites  
Choose the engine

### Tutorials

Simple Viewer  
Hubs Browser  
Dashboard

### Automation

Create Server  
Basic UI  
**Create Plugin**

## Choose the engine

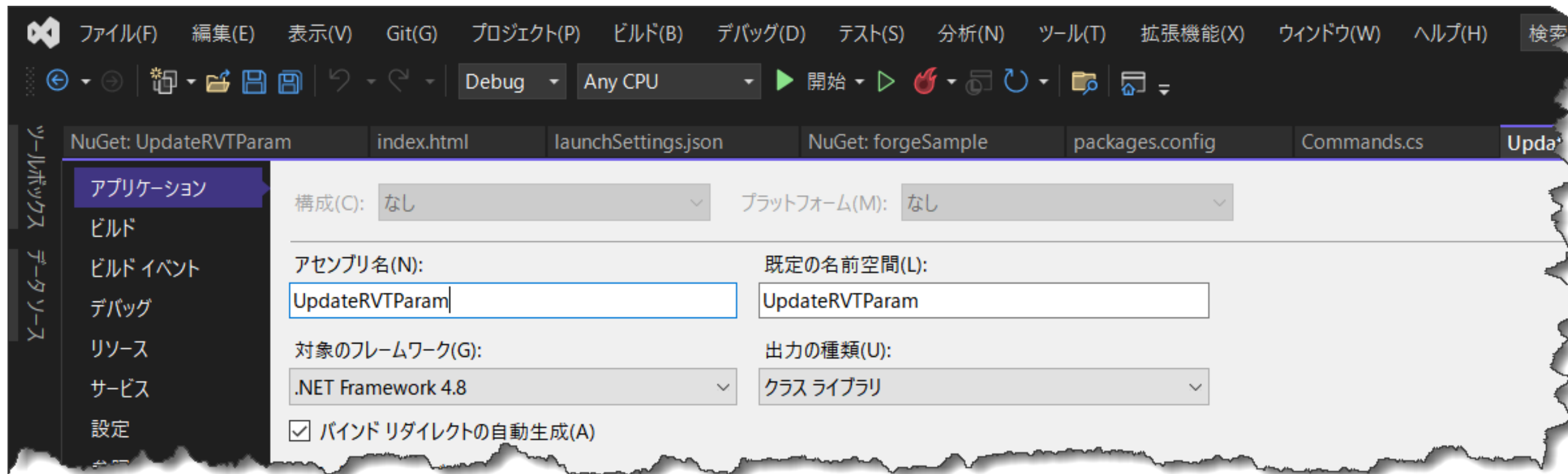
AutoCAD Plugin Inventor Plugin **Revit Plugin** 3ds Max Plugin

This step will help you create a basic Revit plugin for Design Automation. For more information, please visit [My First Revit Plugin](#) tutorial.

You may [download the Bundle ZIP](#) into the `bundles/` (Node.js) or `/designAutomationSample/wwwroot/bundles` (.NET 6) folder and skip to **Upload Plugin Bundle** section.

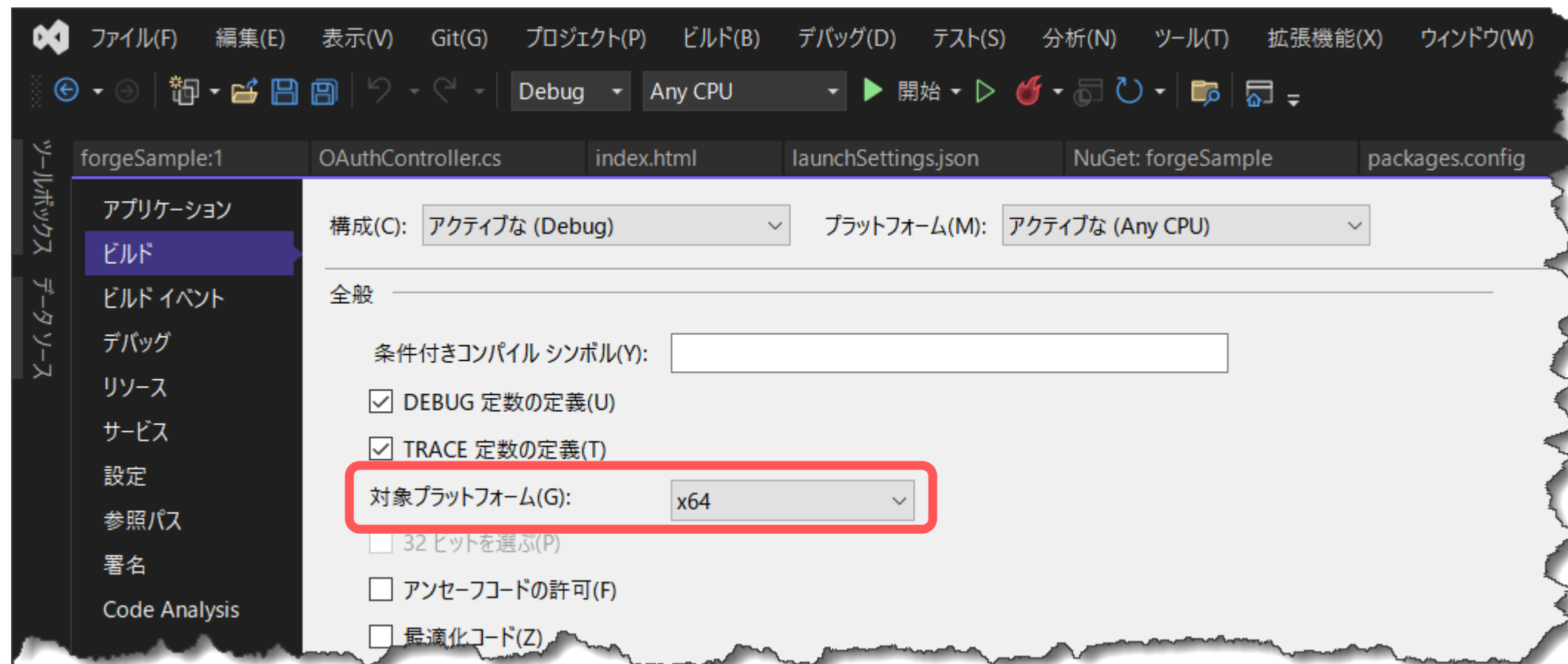
# UpdateRVTParam アドインの作成

- Visual Studio 2019/2022 Professional を使用可
  - Revit アドイン開発で主流 (Revit .NET API)
    - [使用する Microsoft Visual Studio のエディション\(.NET\)](#)
  - クラスライブラリ プロジェクト
    - エンジンバージョンによって .NET/.NET Framework 差に注意
    - Revit .NET Wizard でプロジェクト作成も利用可



# NuGet パッケージ追加時の警告

- Autodesk.Forge.DesignAutomation.Revit を参照に追加した際に警告が表示される場合があります。
  - UpdateRVTParam プロジェクトのプロパティを開きます。
  - ビルドタブで対象プラットフォームを x64 に変更してください。



# プロジェクトの設定を確認する

- packages.config をダブルクリックで開く

```
<?xml version="1.0" encoding="utf-8"?>
<packages>
  <package id="Autodesk.Forge.DesignAutomation.Revit" version="2023.0.1" targetFramework="net48" />
  <package id="Microsoft.CSharp" version="4.5.0" targetFramework="net48" />
  <package id="Newtonsoft.Json" version="13.0.2-beta1" targetFramework="net48" />
</packages>
```

- 使用する Revit のバージョンに対応する .NET Framework と Autodesk.Forge.DesignAutomation.Revit パッケージをインストール
  - .NET Framework
    - Revit 2023, 2024 -> .NET Framework 4.8
    - Revit 2025, 2026 -> .NET 8

# Revit プラグインの作成に必要なアセンブリ参照

IExternalDBApplication  
(外部 DB アプリケーション)

Revit アドイン

DesignAutomationBridge.dll

RevitAPI.dll

その他 dll

RevitAPIUI.dll

参照アセンブリ

.NET / .NET Framework

Windows Server

# IExternalDBApplication の実装

```
1 using Autodesk.Revit.ApplicationServices;
2 using Autodesk.Revit.Attributes;
3 using Autodesk.Revit.DB;
4 using DesignAutomationFramework;
5 using Newtonsoft.Json;
6 using System.Collections.Generic;
7 using System.IO;
8
9 namespace Autodesk.Forge.Sample.DesignAutomation.Revit
10 {
11     [Transaction(TransactionMode.Manual)]
12     [Regeneration(RegenerationOption.Manual)]
13     0 個の参照
14     public class Commands : IExternalDBApplication 実装するインターフェース
15     {
16         //Path of the project(i.e)project where your Window family files are present
17         string OUTPUT_FILE = "OutputFile.rvt";
18
19         0 個の参照
20         public ExternalDBApplicationResult OnStartup(ControlledApplication application) イベントハンドラの登録
21         {
22             DesignAutomationBridge.DesignAutomationReadyEvent += HandleDesignAutomationReadyEvent;
23             return ExternalDBApplicationResult.Succeeded;
24         }
25
26         1 個の参照
27         private void HandleDesignAutomationReadyEvent(object sender, DesignAutomationReadyEventArgs e)
28         {
29             LogTrace("Design Automation Ready event triggered...");
30             e.Succeeded = true;
31             EditWindowParametersMethod(e.DesignAutomationData.RevitDoc); ← カスタム処理を呼び出し
32         }
33     }
34 }
```

# APS アプリからパラメータを渡す

**パラメータの入力** → WorkItem リクエスト時に JSON 形式で送信

**Revit ファイル選択**

**アクティビティ選択**

**WorkItem 開始ボタン**

**WorkItem ログ出力**

# パラメータを Revit アドインで受け取る

```
31 private void EditWindowParametersMethod(Document doc)
32 {
33     InputParams inputParameters = JsonConvert.DeserializeObject<InputParams>(File.ReadAllText("params.json"));
34 }
35 using (Transaction trans = new Transaction(doc))
36 {
37     trans.Start("Update window parameters");
38
39     FilteredElementCollector WindowCollector = new FilteredElementCollector(doc)
40         .OfCategory(BuiltInCategory.OST_Windows).WhereElementIsNotElementType();
41     IList<ElementId> windowIds = WindowCollector.ToElementIds() as IList<ElementId>;
42
43     foreach (ElementId windowId in windowIds)
44     {
45         Element Window = doc.GetElement(windowId);
46         FamilyInstance FamInst = Window as FamilyInstance;
47         FamilySymbol FamSym = FamInst.Symbol;
48         SetElementParameter(FamSym, BuiltInParameter.WINDOW_HEIGHT, inputParameters.Height);
49         SetElementParameter(FamSym, BuiltInParameter.WINDOW_WIDTH, inputParameters.Width);
50     }
51
52     trans.Commit();
53 }
54
55 ModelPath ProjectModelPath = ModelPathUtils.ConvertUserVisiblePathToModelPath(OUTPUT_FILE);
56 SaveAsOptions SAO = new SaveAsOptions();
57 SAO.OverwriteExistingFile = true;
58
59 LogTrace("Saving file...");
60 doc.SaveAs(ProjectModelPath, SAO);
61 }
```

.json ファイルでパラメータを取得可能

# 任意のタイミングでログにテキスト出力

- Revit アドインのプログラム内で、`Console.WriteLine()` メソッドを呼び出せば、任意のタイミングで、`report.txt` にカスタムのログを出力することができます。

```
private static void LogTrace(string format, params object[] args) { System.Console.WriteLine(format, args); }
```

# Revit アドインのエラーハンドリング

- Revit アドインで発生するエラーは、トランザクションのコミット時、あるいはロールバック時に呼び出されます。
- DesignAutomationBridge は、デフォルトのエラーハンドラを搭載しており、すべての警告を抑止し、エラーの場合は、それを解決しようと試みます。解決したらコミットします。
- デフォルトの解決方法が「要素の削除」となる場合は、エラーハンドラは要素を削除せずに、ロールバックします。
- 開発者は、カスタムのエラーハンドラを実装して、デフォルトのエラーハンドラを上書きすることができます。

```
public void HandleDesignAutomationReadyEvent(object sender, DesignAutomationReadyEventArgs e)
{
    // Hook up the CustomFailureHandling failure processor.
    Application.RegisterFailuresProcessor(new CustomFailureHandlingProcessor());

    // Run the application logic.
    SketchItFunc(e.DesignAutomationData);

    e.Succeeded = true;
}
```

# Revit ローカライズ版のエンジンを指定可能

- 起動する Revit のエンジンを、英語以外の言語も指定することができます。
  - /I JPN を指定。
- 日本語版で起動すれば、すべての Revit の出力も日本語版で実行されます。
- 日本語の名称を含んだビルトインの要素やタイプをサポートする場合に利用します。
- ただし、Revit を起動する OS は Windows Server 英語版になります。



```
{  
  "commandLine": [  
    "$(engine.path)¥¥¥¥revitcoreconsole.exe /i $(args[rvtFile].path) /al $(apps[DeleteWallsApp].path) /I DEU"  
  ],  
  ...  
}
```

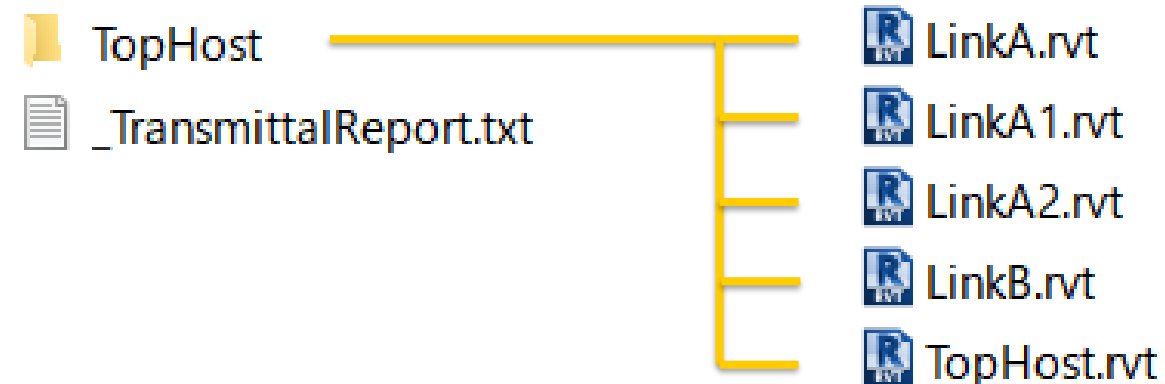
# 追加のフォントサポート

- Revit を起動する OS は **Windows Server 英語版**になります。
- Revit Automation API を使用してモデルを処理する場合、**Worker インスタンスで使用できないフォントはすべて置換されます。**  
エクスポートなどの操作では、フォントが置換されると、結果の外観が異なる場合があります。
- Revit Automation API では、リストに掲載されているフォントがWorker インスタンスに追加でプレインストールされるため、多くの場合、フォントの置換が不要になり、結果の外観が向上します。

# eTransmit ファイルの送信

- eTransmit for Revit で出力した ZIP ファイルをサポート。

```
"arguments": {  
  "rvtFile": {  
    "url": "https://s3-us-west-2.amazonaws.com/revitio-dev/test-data/TopHost.zip"  
  },  
  "countItParams": {  
    "url": "data:application/json,{'walls': true, 'floors': true, 'doors': true, 'windows': true}"  
  },  
  "result": {  
    "verb": "put",  
    "url": "https://myWebsite/signed/url/to/result"  
  }  
}
```



# Revit リンクモデルの送信

```
"rvtFile": {  
  "url": "https://s3-us-west-2.amazonaws.com/revitio-dev/test-data/TopHost.rvt",  
  "references": [  
    {  
      "url": "https://s3-us-west-2.amazonaws.com/revitio-dev/test-data/LinkA.rvt",  
      "references": [  
        {  
          "url": "https://s3-us-west-2.amazonaws.com/revitio-dev/test-data/LinkA1.rvt"  
        },  
        {  
          "url": "https://s3-us-west-2.amazonaws.com/revitio-dev/test-data/LinkA2.rvt"  
        }  
      ]  
    },  
    {  
      "url": "https://s3-us-west-2.amazonaws.com/revitio-dev/test-data/LinkB.rvt"  
    }  
  ]  
},  
}
```

```
TopHost.rvt  
|-- LinkA.rvt  
| |-- LinkA1.rvt  
| |-- LinkA2.rvt  
|  
|-- LinkB.rvt
```

※ サブフォルダのリンクモデルもサポート

# ZIP ファイルの送信

- アップロード速度を向上させるために、Revit モデルを ZIP 圧縮して送信することができます。

```
"arguments": {  
  "rvtFile": {  
    "url": "https://path/to/zip/file.zip",  
    "pathInZip": "RevitFile.rvt"  
  },  
}
```

の Revit モデルのパスを指定しま

# クラウドモデル・ワークシェアリングモデルのサポート

- Revit クラウドモデルを開く

```
var cloudModelPath = ModelPathUtils.ConvertCloudGUIDsToCloudPath(  
    inputParams.Region,  
    inputParams.ProjectGuid,  
    inputParams.ModelGuid);  
Document doc = rvtApp.OpenDocumentFile(cloudModelPath, new OpenOptions());
```

- Revit クラウドモデルを保存する

```
if (doc.IsWorkshared) // work-shared/C4R model  
{  
    SynchronizeWithCentralOptions swc = new SynchronizeWithCentralOptions();  
    swc.SetRelinquishOptions(new RelinquishOptions(true));  
    doc.SynchronizeWithCentral(new TransactWithCentralOptions(), swc);  
}  
else  
{  
    // Single user cloud model  
    doc.SaveCloudModel();  
}
```

# クラウドモデル・ワークシェアリングモデルのサポート

- Revit クラウドモデルを作成する

```
Document newDoc = data.RevitApp.NewProjectDocument(UnitSystem.Imperial);
this.AddWalls(newDoc);

var cloudModelLocation = CloudModelLocation.Parse("CloudModelLocation.json");

newDoc.EnableWorksharing("Shared Levels and Grids", "Workset1");

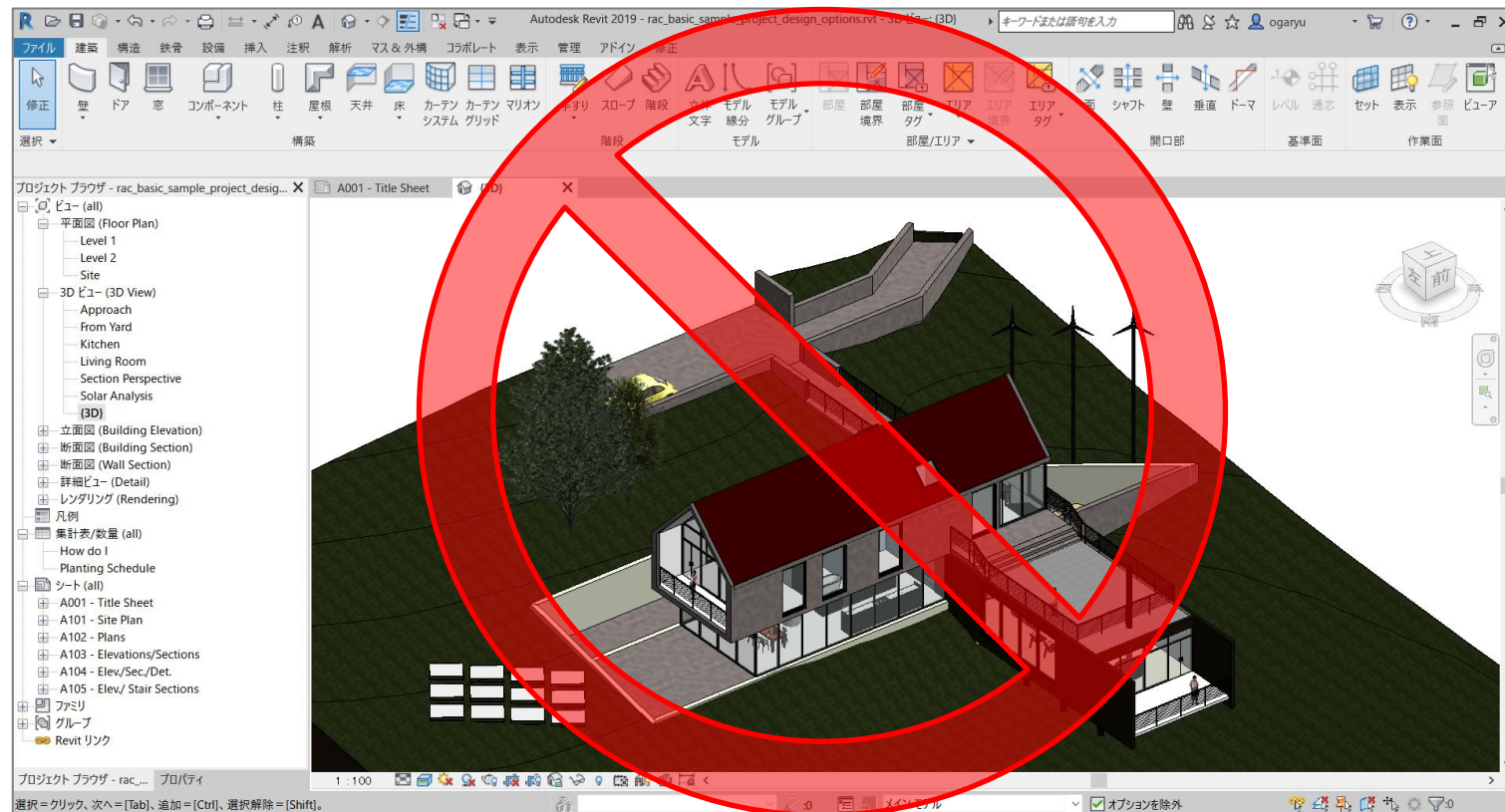
newDoc.SaveAsCloudModel(
    cloudModelLocation.AccountId, cloudModelLocation.ProjectId, cloudModelLocation.FolderId,
    "newRCWModel.rvt");
```

- クラウドモデルを利用する場合に WorkItem で必須

```
{
  inputRVT: {
    url: "XXXXXXXXXX"
  },
  onComplete: {
    verb: "post",
    url: designAutomation.webhook_url
  },
  adsk3LeggedToken: access_token
}
```

# Revit アドイン開発時の制約事項

- [x] Revit UI 名前空間へのアクセス、UI 画面へのアクセスはできません。
- [x] ユーザーとのインタラクションが発生する処理はサポートされておりません。
- [x] ActiveView と ActiveDocument プロパティにはアクセスできません。
- [x] 複雑なセッション管理は想定されていません。（バッチ処理を想定）
- [x] Navisworks への書き出し、Desktop Connector は未サポートです。



RevitAPIUI.dll

# バンドルパッケージの作成

- AppBundle に登録するファイルは、アセンブリファイルとアドインマニフェストファイルを指定のフォルダ構成で格納して **UTF-8 形式で ZIP 圧縮したパッケージ**です。

DeleteWallsApp.zip

|-- DeleteWalls.bundle

| |-- PackageContents.xml

| |-- Contents

| | |-- DeleteWalls.dll

| | |-- DeleteWalls.addin

```
<?xml version="1.0" encoding="utf-8"?>
<RevitAddIns>
  <AddIn Type="DBApplication">
    <Name>DeleteWalls</Name>
    <Assembly>. #DeleteWalls.dll</Assembly>
    <AddInId>d7fe1983-8f10-4983-98e2-c3cc332fc978</AddInId>
    <FullClassName>DeleteWalls.DeleteWallsApp</FullClassName>
    <Description>"Walls Deleter"</Description>
    <VendorId>Autodesk</VendorId>
    <VendorDescription>
    </VendorDescription>
  </AddIn>
</RevitAddIns>
```

```
<?xml version="1.0" encoding="utf-8" ?>
<ApplicationPackage>
  <Components Description="Delete Walls">
    <RuntimeRequirements OS="Win64"
      Platform="Revit"
      SeriesMin="R2024"
      SeriesMax="R2024" />
    <ComponentEntry AppName="DeleteWalls"
      Version="1.0.0"
      ModuleName=". /Contents/DeleteWalls.addin"
      AppDescription="Deletes walls"
      LoadOnCommandInvocation="False"
      LoadOnRevitStartup="True" />
  </Components>
</ApplicationPackage>
```

# AppBundle の登録

## API エンドポイント : POST appbundles

Method and URI **POST** <https://developer.api.autodesk.com/da/us-east/v3/appbundles>

Authentication Context app only

Required OAuth Scopes code:all

Data Format JSON

### Request

```
curl -X POST ¥
  'https://developer.api.autodesk.com/da/us-east/v3/appbundles' ¥
  -H 'Authorization: Bearer <YOUR_ACCESS_TOKEN>' ¥
  -H 'Content-Type: application/json' ¥
  -d '{
    "id": "DeleteWallsApp",
    "engine": "Autodesk.Revit+2024",
    "description": "Delete Walls AppBundle based on Revit 2024"
  }'
```

### Response Body

```
{
  "uploadParameters": {
    "endpointURL": "https://dasprod-store.s3.amazonaws.com",
    "formData": {
      "key": "apps/Revit/DeleteWallsApp/1",
      "content-type": "application/octet-stream",
      "policy": "eyJleHBpcmF0aW9uIjoimjAxOC... (truncated)",
      "success_action_status": "200",
      "success_action_redirect": "",
      "x-amz-signature": "6c68268e23ecb8452... (truncated)",
      "x-amz-credential": "ASIAQ2W... (truncated)",
      "x-amz-algorithm": "AWS4-HMAC-SHA256",
      "x-amz-date": "20180810... (truncated)",
      "x-amz-server-side-encryption": "AES256",
      "x-amz-security-token": "FQoGZXIvYXdzEPj//////////wEaDHavu... (truncated)"
    }
  },
  "engine": "Autodesk.Revit+2018",
  "description": "Delete Walls AppBundle based on Revit 2018",
  "version": 1,
  "id": "YOUR_NICKNAME.DeleteWallsApp"
}
```

# バンドルパッケージのアップロード

endpointURL に formData にファイルを入れて送信

```
curl -X POST ¥  
  'https://dasprod-store.s3.amazonaws.com' ¥  
-H 'Cache-Control: no-cache' ¥  
-F 'key=apps/Revit/DeleteWallsApp/1' ¥  
-F 'content-type=application/octet-stream' ¥  
-F 'policy=eyJleHBpcmF0aW9uIjoimjAxOC... (truncated)' ¥  
-F 'success_action_status="200"' ¥  
-F 'success_action_redirect=' ¥  
-F 'x-amz-signature=6c68268e23ecb8452... (truncated)' ¥  
-F 'x-amz-credential=ASIAQ2W... (truncated)' ¥  
-F 'x-amz-algorithm=AWS4-HMAC-SHA256' ¥  
-F 'x-amz-date=20180810... (truncated)' ¥  
-F 'x-amz-server-side-encryption=AES256' ¥  
-F 'x-amz-security-token=FQoGZXIvYXdzEPj//////////wEaDHavu... (truncated)' ¥  
-F 'file=@path/to/your/app/zip' ←
```

# AppBundle Alias/Version の作成

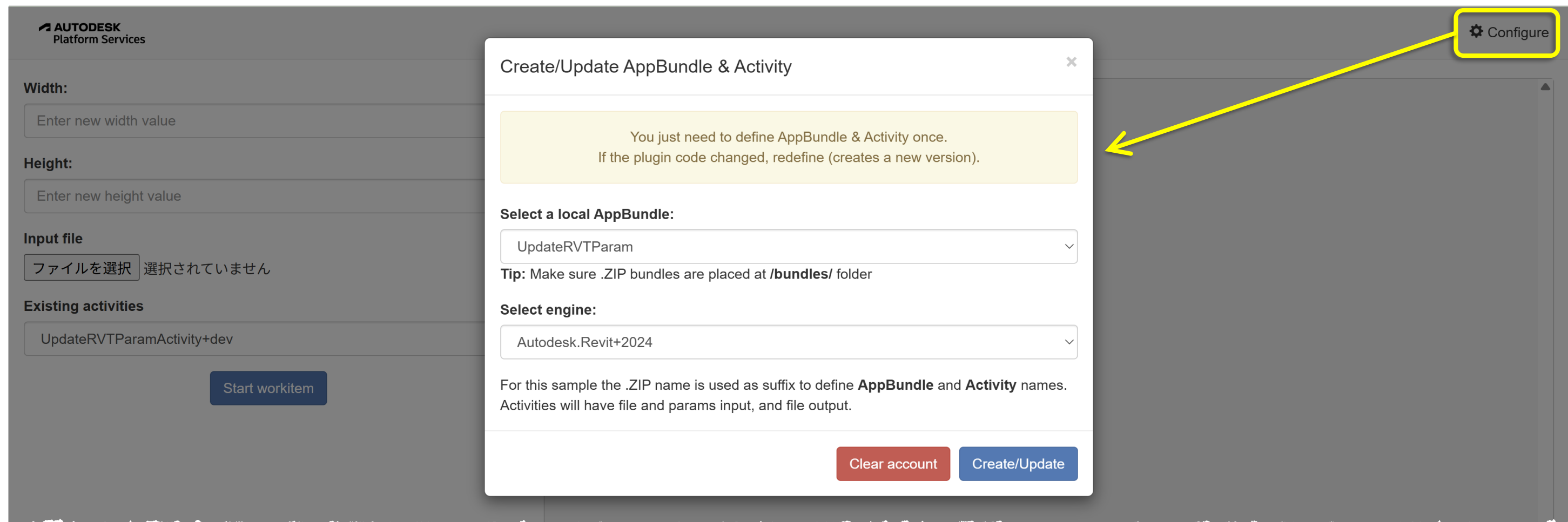
API エンドポイント : POST appbundles/:id/aliases

Method and URI	<b>POST</b> <a href="https://developer.api.autodesk.com/da/us-east/v3/appbundles/:id/aliases">https://developer.api.autodesk.com/da/us-east/v3/appbundles/:id/aliases</a>
Authentication Context	app only
Required OAuth Scopes	code:all
Data Format	JSON

```
curl -X POST ¥  
  'https://developer.api.autodesk.com/da/us-east/v3/appbundles/DeleteWallsApp/aliases' ¥  
-H 'Content-Type: application/json' ¥  
-H 'Authorization: Bearer <YOUR_ACCESS_TOKEN>' ¥  
-d '{  
  "version": 1,  
  "id": "test"  
}'
```

# VS Code 上の操作手順

1. ルートフォルダに **bundles** フォルダを作成
2. UpdateRVTPParam アドイン パッケージを作成 または ダウンロード
3. **routes** フォルダ下に **DesignAutomation.js** ファイルを作成して実装
4. デバッグ実行 >> [localhost:8080](http://localhost:8080)



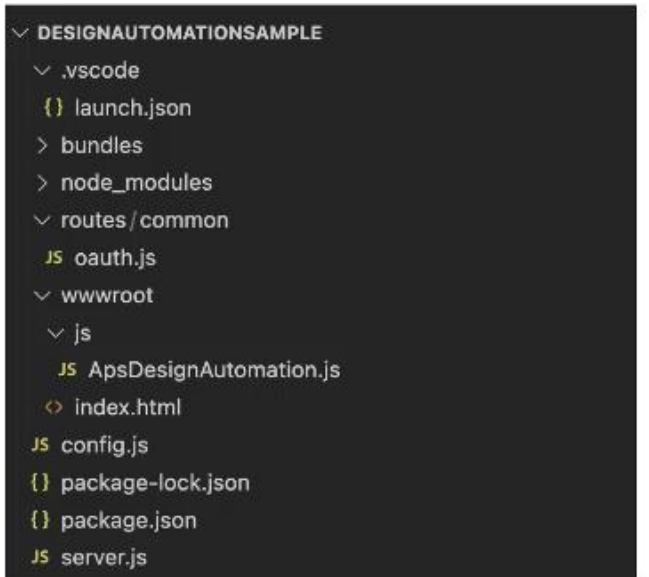
On this page

# Create Plugin

Design Automation uses `.bundle` just like the Autodesk App Store, meaning you need to create a PackageContents.xml and a ZIP with the DLL (and other required files). For detailed information on how to create them, please visit Autodesk App Store Developer Center.

At this section we will create a basic plugin that update width and height parameter and save the resulting file. Also the supporting files (PackageContents.xml) and the folder structure to place them. Finally create a .ZIP file ready to upload to Design Automation.

In the root folder, create a `bundles` folder.



```
wwwroot > js > JS ApsDesignAutomation.js > startConnection
181
182 var connection;
183 var connectionId;
184
185 function startConnection(onReady) {
186     if (connection && connection.connected) {
187         if (onReady) onReady();
188         return;
189     }
190     connection = io();
191     connection.on("connect", function () {
192         connectionId = connection.id;
193         if (onReady) onReady();
194     });
195
196     connection.on("downloadResult", function (url) {
197         writeLog('<a href="' + url + '>Download result fi
198     });
199
200     connection.on("downloadReport", function (url) {
201         writeLog('<a href="' + url + '>Download report fi
202     });
203
204     connection.on("onComplete", function (message) {
205         if (typeof message === "object") message = JSON.st
206         writeLog(message);
207     });
208 }
```

PROBLEMS OUTPUT **DEBUG CONSOLE** TERMINAL PORTS

Filter (e.g. text, \exclude, \escape)

```
C:\nvm4w\nodejs\node.exe --experimental-network-inspection .\star
t.js
Server listening on port 8080 start.js:5
a client is connected socket.io.js:9
Process exited with code 1
```

# 処理内容：窓インスタンスのタイプ パラメータの変更

- 高さ
- 幅

Autodesk Revit 2024.3 - revit\_sample\_file.rvt - 3D ビュー: (3D)

修正 | 窓

修正 | 窓

プロパティ

MYWINDOW  
0406 x 0610mm

窓 (1) ▼ タイプ編集

拘束

基準レベル Level 2

下枠高さ -2' 0"

識別情報

イメージ

コメント

マーク 7

フェーズ

構築フェーズ New Construction

解体フェーズ なし

IFC パラメータ

定義済み IFC タイプ

書き出し IFC クラス

IFC に書き出し タイプ別

IfcGUID 3vfAdAf\_rD39TxWnGR7I3Y

その他

上枠の高さ 3' 0"

タイプ プロパティ

ファミリー(F): MYWINDOW   ロード(L)...

タイプ(T): 0406 x 0610mm   複製(D)...

名前変更(R)...

タイプ パラメータ(M)

パラメータ	値
<b>構成</b>	
開口処理	ホスト別
構成タイプ	
<b>マテリアルと仕上げ</b>	
Frame Exterior Material	Sash
Frame Interior Material	Sash
Glass Pane Material	Glass
Sash	Sash
<b>寸法</b>	
高さ	5' 0"
Default Sill Height	3' 0 3/128"
幅	10' 0"
Window Inset	0' 0 191/256"
全幅	
全高	
<b>解析用プロパティ</b>	
可視光線透過率	0.900000

並べ替え: [アイコン] [アイコン] [アイコン]

<< プレビュー(P)   OK   キャンセル   適用

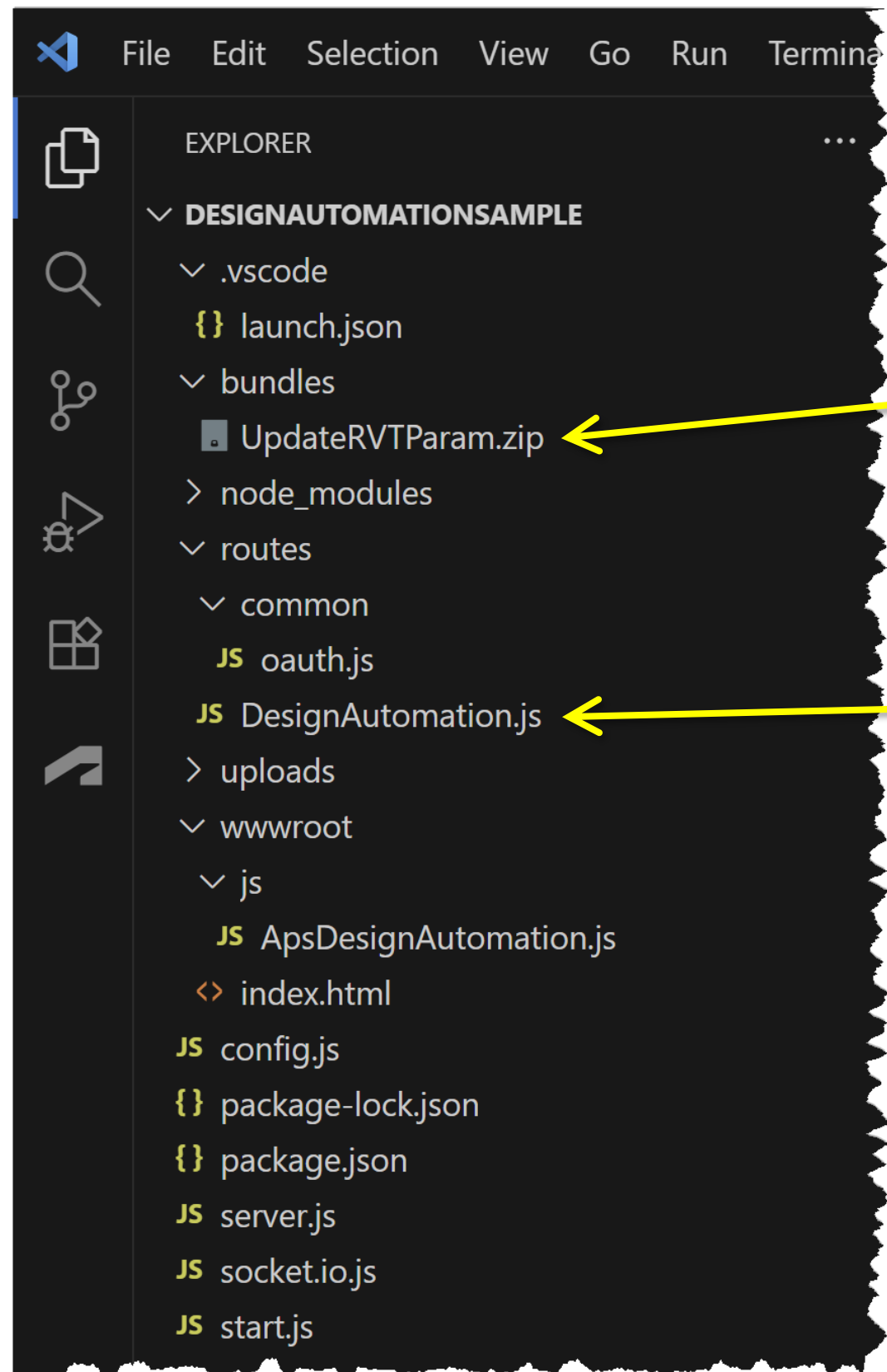
プロジェクト ブラウザ - revit\_sample\_file.rvt   プロパティ

1/8" = 1'-0"

準備中

メイン モデル

# ここまでのプロジェクト構成



## サーバー実装

パッケージ バンドル

ユーティリティ クラス実装  
AppBundle、Activity 登録



# Activity の定義

# Activity の定義 - Define Activity

The screenshot shows the Autodesk Platform Services website. The top navigation bar includes the Autodesk logo, a search bar, and a 'Sign in' button. Below the navigation bar, there are several menu items: Solutions, Getting Started, Documentation, Success Stories, Community, Support, Pricing, and App Store. The left sidebar contains a list of links: Getting Started, Environment Setup, Tutorials (highlighted), Simple Viewer, Hubs Browser, Dashboard, Design Automation (highlighted), Create Server, Basic UI, Create Plugin, Define Activity (highlighted), and Execute Workitem. The main content area shows the breadcrumb path: Home > Tutorials > Design Automation > Define Activity. The title 'Define Activity' is prominently displayed. Below the title, there is a paragraph explaining that an activity is a specification of an action that can be executed using a specified engine, detailing input and output files and the AppBundle and entry-point. Below this, it states that in the tutorial sample, the activity has 2 inputs (file & JSON data) and 1 output (file). There are three tabs for the tutorial: Node.js & VSCode (selected), .NET & VSCode, and .NET & VS2022. A list of topics is shown below the tabs, with 'Activity' selected. The bottom of the page shows the start of a paragraph: 'Now we will write endpoints for creating new activity and getting the existing'.

**AUTODESK**  
Platform Services

Search

Sign in

Solutions ▾ Getting Started Documentation ▾ Success Stories Community ▾ Support ▾ Pricing App Store ▾

Getting Started  
Environment Setup  
**Tutorials**  
Simple Viewer  
Hubs Browser  
Dashboard  
**Design Automation**  
Create Server  
Basic UI  
Create Plugin  
**Define Activity**  
Execute Workitem

Home > Tutorials > Design Automation > Define Activity

## Define Activity

Activity is the specification of an action that can be executed using a specified engine. It specifies the number of input and output files, and the AppBundle and entry-point to use.

In this tutorial sample, the activity has 2 inputs (file & JSON data) and 1 output (file).

Node.js & VSCode .NET & VSCode .NET & VS2022

- Activity

Now we will write endpoints for creating new activity and getting the existing

# Activity – アクティビティ

1. WorkItem が処理する入出力ファイル/パラメータを宣言
2. WorkItem が実行するコマンド名を指定
  - Activity の種類
    - **パブリック共有 Activity**
      - オートデスクが提供する Activity
    - **カスタム Activity (独自の Activity)**
      - アドインによるカスタム コマンド
      - 標準コマンド (アドイン不要 = AppBundle 不要)
      - スクリプト

# API エンドポイント

## POST activities

---

Method and URI **POST** <https://developer.api.autodesk.com/da/us-east/v3/activities>

---

Authentication Context app only

---

Required OAuth Scopes `code:all`

---

Data Format JSON

---

## POST activities/:id/aliases

---

Method and URI **POST** <https://developer.api.autodesk.com/da/us-east/v3/activities/:id/aliases>

---

Authentication Context app only

---

Required OAuth Scopes `code:all`

---

Data Format JSON

---

# Activity の定義の例

```
1 {
2   "id": "UpdateRVTPParamActivity", Activity ID
3   "commandLine": [ "$(engine.path)\\\\\\revitcoreconsole.exe /i \\\"$(args[inputRvtFile].path)\\\" /al $(appbundles[UpdateRVTPParam].path)" ],
4   "parameters": {
5     "inputFile": {
6       "zip": false,
7       "ondemand": false,
8       "verb": "get",
9       "description": "input file",
10      "required": true
11    },
12    "inputJson": {
13      "zip": false,
14      "ondemand": false,
15      "verb": "get",
16      "description": "input json",
17      "required": false,
18      "localName": "params.json"
19    },
20    "outputFile": {
21      "zip": false,
22      "ondemand": false,
23      "verb": "put",
24      "description": "output file",
25      "required": true,
26      "localName": "OutputFile.rvt"
27    }
28  },
29  "engine": "Autodesk.Revit+2021", Engine
30  "appbundles": [ "MyFirstForgeAppNickname.UpdateRVTPParam+dev" ], AppBundle ID
31  "description": "."
32 }
```

入力 Revit ファイル

入力 JSON ファイル

出力 Revit ファイル

**verb: get** Revit アドインに渡すファイル

**verb: put** Revit アドインから出力するファイル

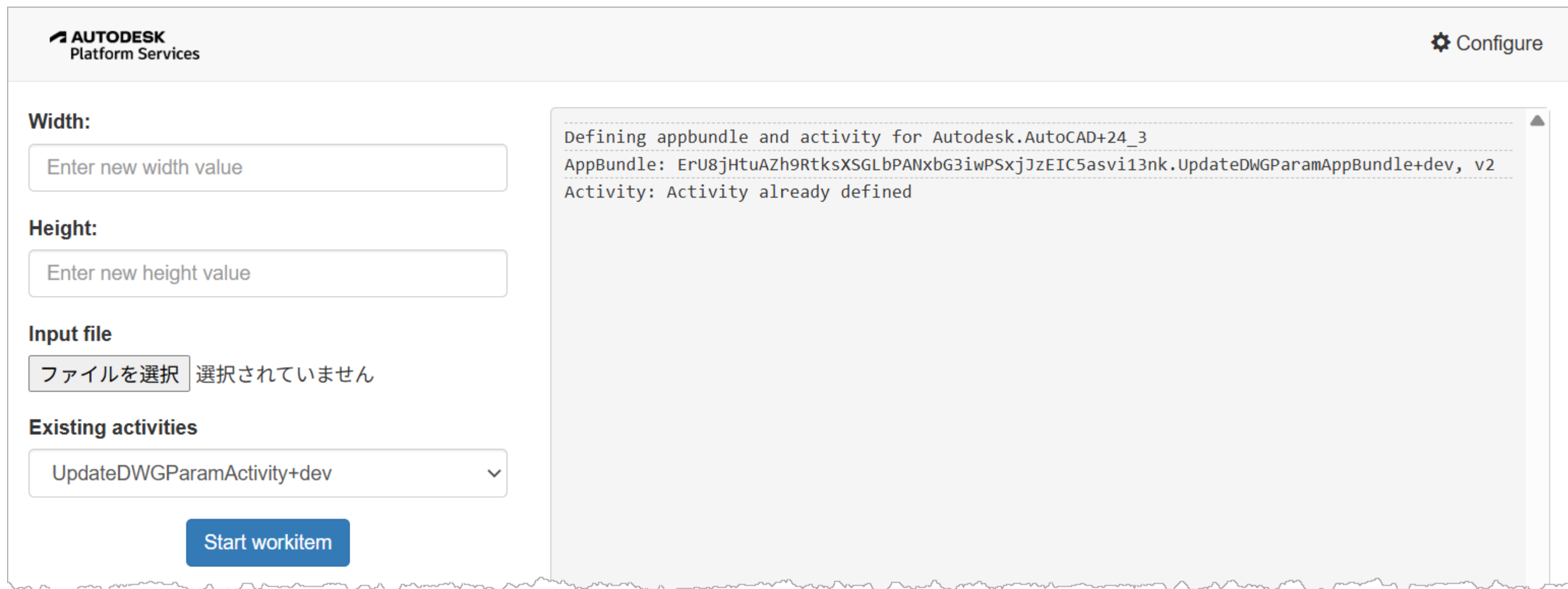
**localName** ワーキングディレクトリ上でのファイル名

**required** 必須かそうでないかを指定

**ondemand** WorkItem の処理中に任意のタイミングで外部データファイルを取得する方法

# VS Code 上の操作手順

1. DesignAutomation.js ファイルに Activity 定義の登録を実装
2. デバッグ実行 >> [localhost:8080](http://localhost:8080)



# Define Activity

Activity is the specification of an action that can be executed using a specified engine. It specifies the number of input and output files, and the AppBundle and entry-point to use.

In this tutorial sample, the activity has 2 inputs (file & JSON data) and 1 output (file).

- Node.js & VSCode
- .NET & VSCode
- .NET & VS2022

- Activity

Now we will write endpoints for creating new activity and getting the existing activities, copy the following code into

DesignAutomation.js file before the last line module.exports = router;

```
routes/DesignAutomation.js

/// <summary>
/// CreateActivity a new Activity
/// </summary>
router.post(
  "/aps/designautomation/activities",
  async (/*CreateActivity*/ req, res) => {
    const activitySpec = req.body;
```

The VS Code editor displays the file `server.js` with the following JavaScript code:

```
3 const cookieSession = require("cookie-session");
4 const config = require("./config");
5 if (!config.credentials.client_id || !config.credentials.client_secret)
6   return console.error(
7     "Missing APS_CLIENT_ID or APS_CLIENT_SECRET env variables."
8   );
9
10 let app = express();
11 app.use(express.static(path.join(__dirname, './wwwroot')));
12 app.use(
13   cookieSession({
14     name: "aps_session",
15     keys: ["aps_secure_key"],
16     maxAge: 60 * 60 * 1000, // 1 hour, same as the 2 legged lifespan token
17   })
18 );
19 app.use(
20   express.json({
21     limit: "50mb",
22   })
23 );
24
25 app.use("/api", require("./routes/DesignAutomation"));
26 app.set("port", process.env.PORT || 8080);
27
28 module.exports = app;
```

The terminal window shows the following output:

```
C:\nvm4w\nodejs\node.exe --experimental-network-inspection .\start.js
Server listening on port 8080
a client is connected
OAuth2 client created for bucket:create+bucket:read+bucket:delete+data:read+dat ...h.js:31
a:create+data:write+code:all
a client disconnected
a client is connected
Process exited with code 1
```

# AppBundle と Activity の実装

Create/Update AppBundle & Activity ×

You just need to define AppBundle & Activity once.  
If the plugin code changed, redefine (creates a new version).

**Select a local AppBundle:**

Tip: Make sure .ZIP bundles are placed at /bundles/ folder

**Select engine:**

For this sample the .ZIP name is used as suffix to define **AppBundle** and **Activity** names.  
Activities will have file and params input, and file output.

Clear account
Create/Update

## クライアント実装

**wwwroot¥index.html**

Id="createAppBundleActivity"

**wwwroot¥js¥ApsDesignAutomation.js**

createAppBundleActivity()

**ApsDesignAutomation.js**

createActivity()

**ApsDesignAutomation.js**

createAppBundle()

**ApsDesignAutomation.js**

"api/aps/designautomation/activities"

**ApsDesignAutomation.js**

"api/aps/designautomation/appbundles"

**¥routes¥DesignAutomation.js**

router.post("/aps/designautomation/appbundles",

**¥routes¥DesignAutomation.js**

router.post("/aps/designautomation/appbundles",

## サーバー実装

# クライアントサイド (ApsDesignAutomation.js)

```
function createAppBundleActivity() {
  startConnection(function () {
    writeLog("Defining appbundle and activity for " + $("#engines").val());
    $("#defineActivityModal").modal("toggle");
    createAppBundle(function () {
      createActivity(function () {
        prepareLists();
      });
    });
  });
}
```

```
function createAppBundle(cb) {
  jQuery.ajax({
    url: "api/aps/designautomation/appbundles",
    method: "POST",
    contentType: "application/json",
    data: JSON.stringify({
      zipFileName: $("#localBundles").val(),
      engine: $("#engines").val(),
    }),
    success: function (res) {
      writeLog("AppBundle: " + res.appBundle + ", v" + res.version);
      if (cb) cb();
    },
    error: function (xhr, ajaxOptions, errorThrown) {
      writeLog(
        " -> " +
        (xhr.responseJSON && xhr.responseJSON.diagnostic
          ? xhr.responseJSON.diagnostic
          : errorThrown)
      );
    }
  });
}
```

```
function createActivity(cb) {
  jQuery.ajax({
    url: "api/aps/designautomation/activities",
    method: "POST",
    contentType: "application/json",
    data: JSON.stringify({
      zipFileName: $("#localBundles").val(),
      engine: $("#engines").val(),
    }),
    success: function (res) {
      writeLog("Activity: " + res.activity);
      if (cb) cb();
    },
    error: function (xhr, ajaxOptions, errorThrown) {
      writeLog(
        " -> " +
        (xhr.responseJSON && xhr.responseJSON.diagnostic
```

サーバーの URL にPOSTリクエスト送信

# サーバーサイド (DesignAutomation.js)

```
router.post(
  "/aps/designautomation/appbundles",
  async (/*CreateAppBundle*/ req, res) => {
    const appBundleSpecs = req.body;

    // basic input validation
    const zipFileName = appBundleSpecs.zipFileName;
    const engineName = appBundleSpecs.engine;
```

クライアントからの POSTリクエスト  
で呼び出されるメソッド

```
router.post(
  "/aps/designautomation/activities",
  async (/*CreateActivity*/ req, res) => {
    const activitySpecs = req.body;

    // basic input validation
    const zipFileName = activitySpecs.zipFileName;
    const engineName = activitySpecs.engine;
```



# WorkItem の実行

# WorkItem の実行 - Execute Workitem

The screenshot shows the Autodesk Platform Services website. The top navigation bar includes the Autodesk logo, a search bar, and a 'Sign in' button. Below the navigation bar, there are several menu items: Solutions, Getting Started, Documentation, Success Stories, Community, Support, Pricing, and App Store. The left sidebar contains a list of links: Getting Started, Environment Setup, Tutorials (highlighted), Simple Viewer, Hubs Browser, Dashboard, Design Automation (highlighted), Create Server, Basic UI, Create Plugin, Define Activity, and Execute Workitem (highlighted). The main content area displays the 'Execute Workitem' tutorial page, which includes a breadcrumb trail (Home > Tutorials > Design Automation > Execute Workitem), a title 'Execute Workitem', and a description: 'A job that executes a specified Activity, using specified input files and generating appropriate output files.' Below the description, there is a paragraph explaining the relationship between an Activity and WorkItem, and another paragraph describing the tutorial sample. A right sidebar contains links for 'Run & Debug', 'Using the sample', and 'Troubleshooting'.

**AUTODESK**  
Platform Services

Search

Sign in

Solutions ▾ Getting Started Documentation ▾ Success Stories Community ▾ Support ▾ Pricing App Store ▾

Getting Started  
Environment Setup  
**Tutorials**  
Simple Viewer  
Hubs Browser  
Dashboard  
**Design Automation**  
Create Server  
Basic UI  
Create Plugin  
Define Activity  
**Execute Workitem**

Home > Tutorials > Design Automation > Execute Workitem

## Execute Workitem

A job that executes a specified Activity, using specified input files and generating appropriate output files.

The relationship between an Activity and WorkItem can be thought of as a “function definition” and “function call”, respectively. The Activity specifies the AppBundle(s) to use, which in turn specify the Engine to use. The Workitem is then called to execute those.

In this tutorial sample, the workitem specifies the input file URL, the input JSON data with the new parameter values, and the destination URL for the output file. This sample will upload the input file to a OSS bucket before starting the workitem.

Run & Debug  
Using the sample  
Troubleshooting

# WorkItem – ワークアイテム

- 実際にクラウド上で処理されるタスクの単位
- 定義済 Activity を参照する WorkItem を作成
- Activity の応じた入出力パラメータ値の指定
  - 入力パラメータ
  - 出力ファイル
- WorkItem の進捗状態の監視と処理終了の検出

# WorkItem 実行時に起こること

- AppBundle (アドイン) は実行時に作成の一時作業領域に展開
- Automation API は WorkItem で指定された値を利用
  - 指定のクラウド ストレージからファイルをダウンロード (任意)
  - **指定の値を持つ JSON ファイルを作成**

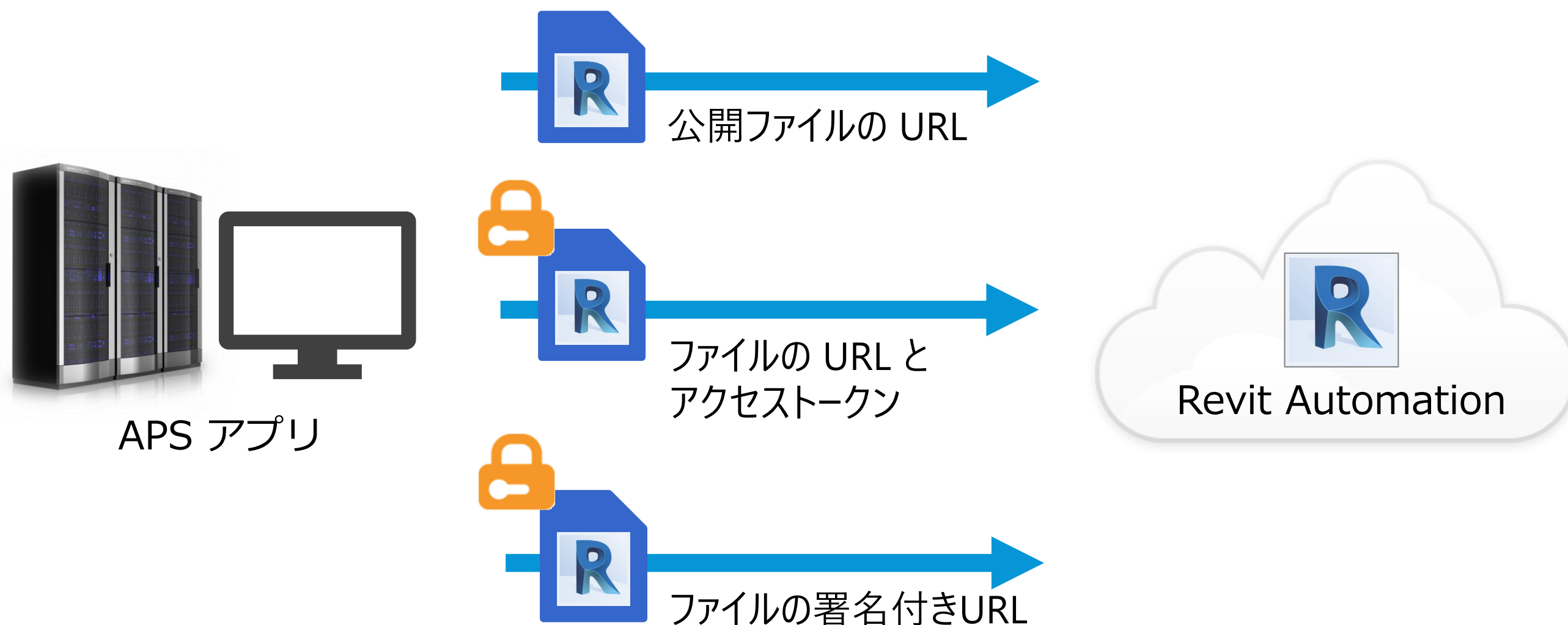
```
{  
  'width': 999,  
  'height': 555  
}
```



- 成果ファイルを指定のクラウド ストレージへ保存 (アップロード)
    - RVT や IFC など
- WorkItem 終了後に一時作業領域は自動削除 (含む使用ファイル)
  - キャッシュして後日の WorkItem で使用することは不可

# クラウドストレージ上のファイル URL の取り扱い

- クラウドストレージ上のファイルを Automation サーバーに送信する方法
  - ファイルの URL のみ (インターネット上の公開ファイルなど)
  - **ファイルの URL とアクセストークン** → オートデスク クラウドストレージ/OSS
  - **署名付き URL** (URLの有効期限、One Time URL、読み書きアクセス権) → OSS



# 署名付き URL の取得

- Data Management API には、OSS にファイルの保存場所を事前に確保して、署名付き URL を発行することもできます。
- この URL に対して、ファイルをアップロードする必要があるため、書き込みアクセス権のある署名付きURLを取得する必要があります。
  - 有効期限の設定
  - 最初に1度だけ使用された後に期限切れになる One Time URL の設定
  - アクセス権の設定
    - access=read
    - access=write
    - access=readwrite

# API エンドポイント : POST workitems

Method and URI

**POST** <https://developer.api.autodesk.com/da/us-east/v3/workitems>

Authentication Context

user context optional

Required OAuth Scopes

code:all

Data Format

JSON

```
curl -X POST ¥
' https://developer.api.autodesk.com/da/us-east/v3/workitems' ¥
-H 'Content-Type: application/json' ¥
-H 'Authorization: Bearer <YOUR_ACCESS_TOKEN>' ¥
-d '{
  "activityId": "<YOUR_APP_NICKNAME>.DeleteWallsActivity+test",
  "arguments": {
    "rvtFile": {
      "url": "urn:adsk.objects:os.object:<YOUR_BUCKET_KEY>/<OBJECT_KEY_4_INPUT_FILE>",
      "verb": "get",
      "headers": {
        "Authorization": "Bearer <YOUR_ACCESS_TOKEN>"
      }
    },
    "result": {
      "url": "urn:adsk.objects:os.object:<YOUR_BUCKET_KEY>/<RESULT_FILE_OBJECT_KEY>",
      "verb": "put",
      "headers": {
        "Authorization": "Bearer <YOUR_ACCESS_TOKEN>"
      }
    }
  }
}
```

# JSON データの作成

```
"parameters": {  
  "inputFile": {  
    "zip": false,  
    "ondemand": false,  
    "verb": "get",  
    "description": "input file",  
    "required": true,  
  },
```

```
    "inputJson": {  
      "zip": false,  
      "ondemand": false,  
      "verb": "get",  
      "description": "input json",  
      "required": false,  
      "localName": "params.json"  
    },
```

```
    "outputFile": {
```

Activity

```
"arguments": {  
  "inputFile": {  
    "url": "https://myWebsite/revit sample file.rvt"  
  },  
  "inputJson": {  
    "url": "data:application/json, {  
      'width': 3,  
      'height': 7  
    }"  
  },  
  "outputFile": {  
    "verb": "put",  
    "url": "https://myWebsite/signed/url/to/revit sample file.rvt"  
  }  
}
```

WorkItem

JSONデータをリクエストのパラメータに埋め込むと、Revit アドインから JSON ファイルとして取得可能。

# WorkItem 処理終了判定と処理レポートについて

- [POST workitems](#) エンドポイント呼び出し
  - 非推奨：150 rpm（150 回/1 分間）が上限
  - レスポンス ボディの **reportUrl**：WorkItem 毎に作成されるレポートログ
- OnComplete コールバックの実装
  - 推奨：要サーバー実装、ただし、クライアント⇔サーバー間トラフィックが低減
  - 開発時のクライアント実装では [ngrok](#) ツール等の利用が必要
- Automation API Tutorial の場合
  - クライアント⇔サーバー実装はソケット通信
  - サーバー実装が [POST workitems](#) 呼び出し（&サーバーからプッシュ）

## サーバー実装

```
¥socket.io.js & ¥routes¥DesignAutomation.js>autodesk.forge.designautomation パッケージ>bundle.js
```

```
.getWorkitemStatus()
```

# GET workitem / OnComplete

## GET workitems/:id

```
curl -X GET ¥
' https://developer.api.autodesk.com/da/us-east/v3/workitems/YOUR_WORKITEM_ID' ¥
-H 'Content-Type: application/json' ¥
-H 'Authorization: Bearer <YOUR_ACCESS_TOKEN>'
```

## OnComplete

```
curl -X POST ¥
https://developer.api.autodesk.com/da/us-east/v3/workitems ¥
...
-d '{
  "activityId": ""QDsYgPEnrk7aPsaP6VvmEXPBmwYAZcMx.timeSleep7x10s+prod",
  "arguments": {
    ...
    "onProgress": {
      "verb": "post",
      "url": "https://onprogress.free.beeceptor.com"
    },
    "onComplete": {
      "verb": "post",
      "url": "https://oncomplete.free.beeceptor.com"
    }
  }
}'
```

```
{
  "status": "success",
  "reportUrl": "https://dasprod-store.s3.amazonaws.com/workItem/<YOUR_APP_NICKNAME>/<YOUR_WORKITEM_ID>/report.txt?... (truncated)",
  "stats": {
    "timeQueued": "2018-04-13T03:15:15.9772282Z",
    "timeDownloadStarted": "2018-04-13T03:15:17.2960823Z",
    "timeInstructionsStarted": "2018-04-13T03:15:20.2803318Z",
    "timeInstructionsEnded": "2018-04-13T03:15:41.6075799Z",
    "timeUploadEnded": "2018-04-13T03:15:42.0450494Z"
  },
  "id": "<YOUR_WORKITEM_ID>"
}
```

WorkItem処理のレポート

- WorkItem を POST する際に、"onComplete" という引数にコールバック URL を設定することができます。
- この引数を事前に設定しておけば、WorkItem の完了時に、**指定の URL** を自動的に呼び出してくれます。
- コールバック URL には、**クエリパラメータ** を組み合わせることができます。
- 進捗状況を30秒毎に "onProgress" コールバック URL で受け取ることもできます。

# VS Code 上の操作手順

1. DesignAutomation.js ファイルに WorkItem 実行処理の実装
2. デバッグ実行 >> [localhost:8080](http://localhost:8080)

The screenshot displays the Autodesk Platform Services interface. On the left, there is a configuration panel with the following fields:

- Width:** 2
- Height:** 4
- Input file:** revit\_sample\_file.rvt (selected via a file picker)
- Existing activities:** UpdateRVTPParamActivity+dev

A "Start workitem" button is located at the bottom of the configuration panel.

On the right, a log window shows the execution details of the WorkItem. The log includes the following entries:

```
[10/13/2025 13:27:40] 6:T:\Aces\Jobs\f5e666d73a1c44ee84819fe02091a863\userdata
[10/13/2025 13:27:40] Selected Revit\RCE install Path: (from app.config)
[10/13/2025 13:27:40] Resolving location of Revit/RevitCoreEngine installation...
[10/13/2025 13:27:40] Loading RCE ....
[10/13/2025 13:27:45] Running user application....
[10/13/2025 13:27:45] Found an addIn for registration: Autodesk.Forge.Sample.DesignAutomation.Revit.addin
[10/13/2025 13:27:45] Language not specified, using English-United States(ENU) as default.
[10/13/2025 13:27:50] Get RCE: (VersionBuild) 24.3.0.13 (VersionNumber) 2024 (SubVersionNumber) 2024.3
[10/13/2025 13:28:20] Design Automation Ready event triggered...
[10/13/2025 13:28:21] Saving file...
[10/13/2025 13:28:29] Finished running. Process will return: Success
[10/13/2025 13:28:29] ===== Revit finished running: revitcoreconsole =====
[10/13/2025 13:28:31] End Revit Core Engine standard output dump.
[10/13/2025 13:28:31] End script phase.
[10/13/2025 13:28:32] Start upload phase.
[10/13/2025 13:28:32] Uploading 'T:\Aces\Jobs\f5e666d73a1c44ee84819fe02091a863\outputFile.rvt': verb - 'Put', url - 'urn:adsk.objects:0:
[10/13/2025 13:28:32] End upload phase successfully.
[10/13/2025 13:28:33] Job finished with result Succeeded
[10/13/2025 13:28:33] Job Status:
{
  "status": "success",
  "reportUrl": "https://dasprod-store.s3.amazonaws.com/workItem/Ty4l4YJnsDne7CS3l0zgwB1zUoTdw9tFAFXrCeYEHqnWGAAz/f5e666d73a1c44ee84819fe
  "activityId": "Ty4l4YJnsDne7CS3l0zgwB1zUoTdw9tFAFXrCeYEHqnWGAAz.UpdateRVTPParamActivity+dev",
  "stats": {
    "timeQueued": "2025-10-13T13:27:38.2002064Z",
    "timeDownloadStarted": "2025-10-13T13:27:38.3037219Z",
    "timeInstructionsStarted": "2025-10-13T13:27:39.5887852Z",
    "timeInstructionsEnded": "2025-10-13T13:28:32.1675359Z",
    "timeUploadEnded": "2025-10-13T13:28:32.9993353Z",
```

Execute Workitem x APS Tutorials | Aut x autodesk-platform x +

get-started.aps.autodesk.com/tutorials/design-automat... ☆

**AUTODESK**  
Platform Services

Sign in

Tutorials > Automation > Execute Workitem

On this page

# Execute Workitem

A job that executes a specified Activity, using specified input files and generating appropriate output files.

The relationship between an Activity and WorkItem can be thought of as a “function definition” and “function call”, respectively. The Activity specifies the AppBundle(s) to use, which in turn specify the Engine to use. The Workitem is then called to execute those.

In this tutorial sample, the workitem specifies the input file URL, the input JSON data with the new parameter values, and the destination URL for the output file. This sample will upload the input file to a OSS bucket before starting the workitem.

**⚠ CAUTION**

Please note that executing an activity in the Design Automation service has a cost associated with it, see the [Pricing](#) page for more details. We recommend that you use a non-expired trial subscription when following this tutorial.

Node.js & VSCode .NET & VSCode .NET & VS2022

designAutomationSample

EXPLORER

- DESIGNAUTOMATIONSAM...
- .vscode
- launch.json
- bundles
- node\_modules
- routes
  - common
  - JS oauth.js
  - JS DesignAutomation.js
- wwwroot
- js
  - JS ApsDesignAutoma...
  - index.html
- JS config.js
- package-lock.json
- package.json
- JS server.js
- JS socket.io.js
- JS start.js

```

routes > JS DesignAutomation.js > router.get("/aps/designautomation/activities") callbac
489   async (/*GetDefinedActivities*/ req, res) => {
492     let activities = null;
493     try {
494       activities = await api.getActivities();
495     } catch (ex) {
496       console.error(ex);
497       return res.status(500).json({
498         diagnostic: "Failed to get activity list",
499       });
500     }
501     let definedActivities = [];
502     for (let i = 0; i < activities.data.length; i++) {
503       let activity = activities.data[i];
504       if (
505         activity.startsWith(Utils.NickName) &&
506         activity.indexOf("$LATEST") === -1
507       )
508         definedActivities.push(activity.replace(Utils.
509       )
510     }
511     res.status(200).json(definedActivities);
512   }
513 );
514
515 module.exports = router;

```

PROBLEMS OUTPUT **DEBUG CONSOLE** TERMINAL PORTS

Filter (e.g. text, !exclude, \escape)

```

Server listening on port 8080 start.js:5
a client is connected socket.io.js:9
OAuth2 client created for bucket:create+bucket:read+bucket:d...js:31
elete+data:read+data:create+data:write+code:all
a client disconnected socket.io.js:14
a client is connected socket.io.js:9
a client disconnected socket.io.js:14
a client is connected socket.io.js:9

```

Launch Program (designAutomationSample) No Solution UTF-8 CRLF {} JavaScript

# WorkItem 作成の実装

**AUTODESK**  
Platform Services

Width:

Height:

Input file  
 revit\_sample\_file.rvt

Existing activities

## クライアント実装

wwwroot¥index.html

Id="startWorkitem"

wwwroot¥js¥ApsDesignAutomation.js

startWorkitem()

wwwroot¥js¥ApsDesignAutomation.js

"api/aps/designautomation/workitems"

¥routes¥DesignAutomation.js

```
router.post('/aps/designautomation/workitems', multer({
```

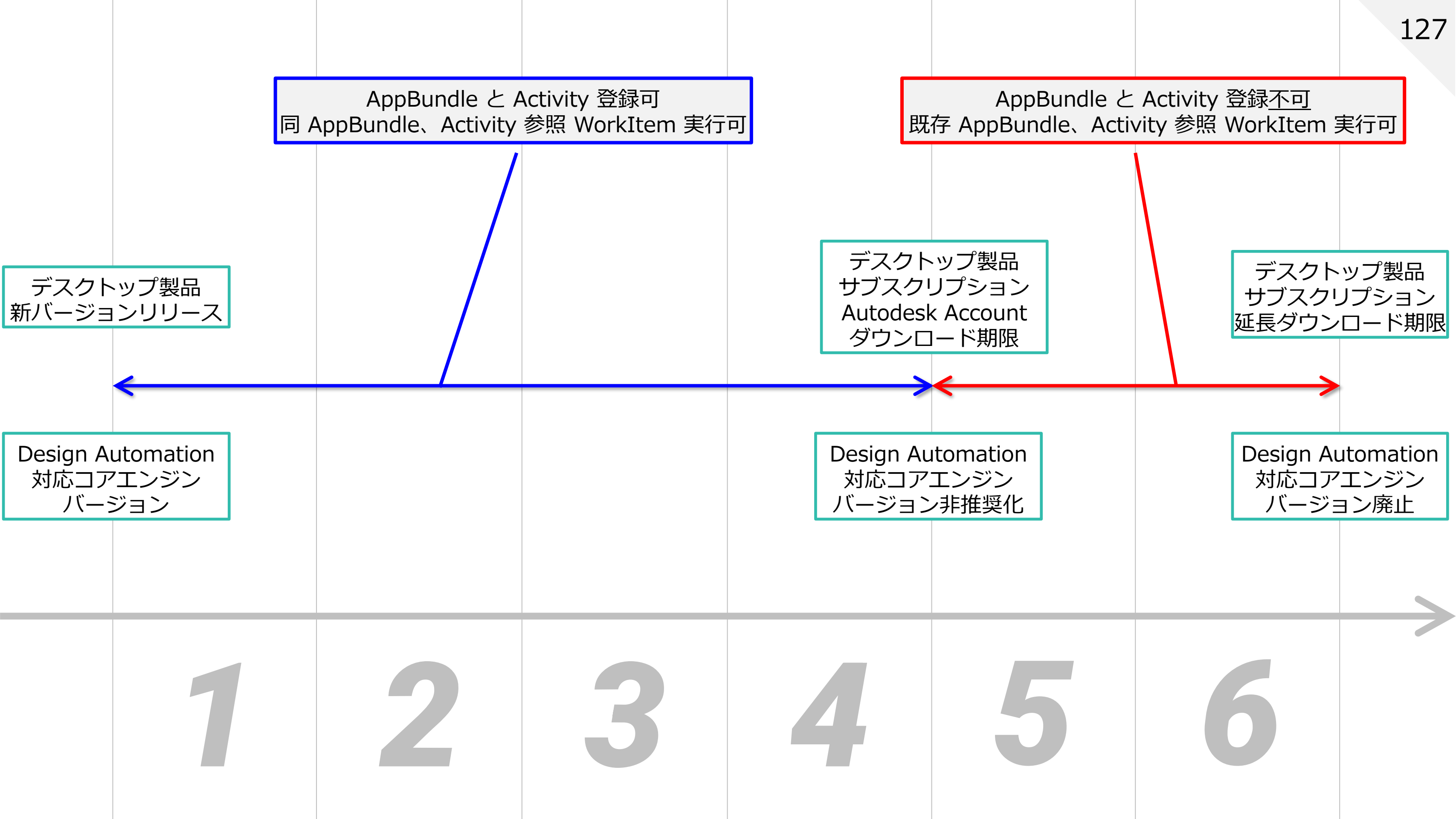
## サーバー実装



# Automation API サポート ポリシー

# Automation API サポートポリシー

1. コアエンジンは、対応するデスクトップ製品の最初のリリースから4年間サポートされます。
2. 4年後にコアエンジンバージョンは非推奨となりますが、さらに2年間は利用可能で、その後削除されます。
3. 非推奨のエンジンバージョンでは、新しい AppBundle や Activity の登録は出来なくなりますが、登録済の AppBundle/Activity を参照する WorkItem は機能し続けます。
4. 削除されたコアエンジンバージョンを参照する WorkItem は実行出来ません。



AppBundle と Activity 登録可  
同 AppBundle、Activity 参照 WorkItem 実行可

AppBundle と Activity 登録不可  
既存 AppBundle、Activity 参照 WorkItem 実行可

デスクトップ製品  
新バージョンリリース

デスクトップ製品  
サブスクリプション  
Autodesk Account  
ダウンロード期限

デスクトップ製品  
サブスクリプション  
延長ダウンロード期限

Design Automation  
対応コアエンジン  
バージョン

Design Automation  
対応コアエンジン  
バージョン非推奨化

Design Automation  
対応コアエンジン  
バージョン廃止

1

2

3


4

5

6

# ご注意：進化し続ける APS・開発作業に終わりはなし クラウドサービス

- APS の機能向上によって動作/仕様が変わる可能性
  - 例) [Design Automation API への OAuth スコープ code:all 適用](#)
  - APS Viewer は特定バージョン指定も可能
- クラウド開発にはメンテナンスが必須
- デスクトップ製品のアドイン開発とは異なります！
- 開発ベンダーとのサブスクリプション？契約が必要！！



# コストとサポート

# APS 利用に対する '課金' とは? (10月15日時点)

<https://aps.autodesk.com/pricing>

- 特定の APS API 利用に **Autodesk Flex** による従量課金

プレミアムAPIとサービス	コスト	情報
Fusion Automation API	3.0	Flex トークン / 処理時間
Automation API (その他)	2.0	Flex トークン / 処理時間
Flow Graph Engine API	1.0	Flex トークン / 処理時間
Model Derivative API	0.5 0.1	Flex トークン / <u>コンプレックスジョブ</u> Flex トークン / <u>シンプルジョブ</u>
Reality Capture API	1.0	50枚の写真毎のFlexトークン

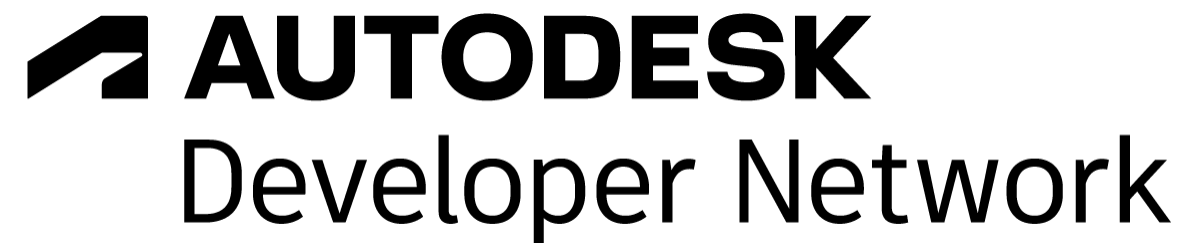
- **Autodesk Flex とは?**
  - <https://www.autodesk.com/jp/buying/flex>



# APS ビジネスモデルの進化（**今後について**）

- <https://aps.autodesk.com/blog/aps-business-model-evolution>
- **新しい 2 階層ビジネスモデル**
  - 12月から導入予定
  - 無料枠と有料枠
    - 前払い
    - 後払い
- 新ビジネス モデルによる有料 API の価格設定の変更
- **11月のオンラインウェビナーの中でご案内予定**

# Autodesk Developer Network (ADN) とは



- **年間契約の開発サポートプログラム（有償）です**
  - 新製品/バージョン、API、テクノロジーの啓蒙と流布
  - 無償ブログ公開、ウェビナー等イベント開催、その他
  - ADN 加入メンバーへの API 開発サポートの提供
  - API の問題点、不具合、要望の米国本社へのフィードバック
  - Autodesk App Store アプリの公開審査とサポート

# ADN 特典

1. ADN Extranet（会員専用サイト）アクセスを提供
2. オートデスク API についてのお問合せ窓口提供
3. 開発用途の製品ライセンスを提供
4. マーケティング用ロゴの提供
5. ユーザー数に応じた Autodesk Flex の提供
6. 開発者向け有償イベントへの優待価格提供

# 参考リソース

- サンプルコード : GitHub
  - <https://github.com/Developer-Autodesk>
  - <https://github.com/Autodesk-APS>
- サポート : StackOverflow (英語)
  - <https://APS.autodesk.com/en/support/get-help>
- ブログ :
  - <https://APS.autodesk.com/blog> (英語)
  - <https://blog.autodesk.io/about-us/>
  - 日本語ブログは現在移行作業中
    - <https://blog.autodesk.io/category/technology-perspective-from-japan/>



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.

© 2025 Autodesk. All rights reserved.