



# Design Automation API for Revit アップデート

小笠原 龍司  
オートデスク株式会社

# アジェンダ

- 1 Design Automation API 概要
- 2 AppBundle の作成方法
- 3 OSS Direct アップロード
- 4 Revit クラウドモデル
- 5 ネットワークアクセス
- 6 Design Automation for Revit with Dynamo



# 免責事項

本イベントでのプレゼンテーションには、当社の見通し、将来の実績および関連する仮定、獲得可能な最大市場規模、買収、製品および製品能力、戦略に関する将来の見通しに関する記述が含まれる場合があります。これらの記述は、現在判明している要因に基づく当社の最善の判断を反映したものです。実際の出来事や実績は大きく異なる可能性があります。当社の実績が将来の見通しに関する記述と異なる原因となりうる重要なリスクおよびその他の要因については、[www.sec.gov](http://www.sec.gov) で入手可能な最新の Form 10-K および Form 10-Q を含む当社の SEC 提出書類をご参照ください。

これらのプレゼンテーションにおける将来の見通しに関する記述は、プレゼンテーション実施日当日時点でのものです。これらのプレゼンテーションが実施日当日時点以降に見直される場合、たとえその後当社が当社の Web サイトその他で利用可能にしたとしても、それらのプレゼンテーションには最新または正確な情報が含まれていない可能性があります。当社は、将来の見通しに関する記述を更新または修正する義務を一切負いません。

当社の製品およびサービスに関する計画済みまたは将来的な開発努力に関する記述は、製品、サービス、または機能が将来利用可能になることを約束または保証することを意図したのではなく、単に当社の現在の計画を反映したものであり、現在当社が把握している要素に基づくものです。これらの記述に依存して購入の意思決定を行うべきではありません。

注意：すべてのオートデスクのコンテンツは所有権で保護されています。許可なくコピー、投稿、配布しないでください。

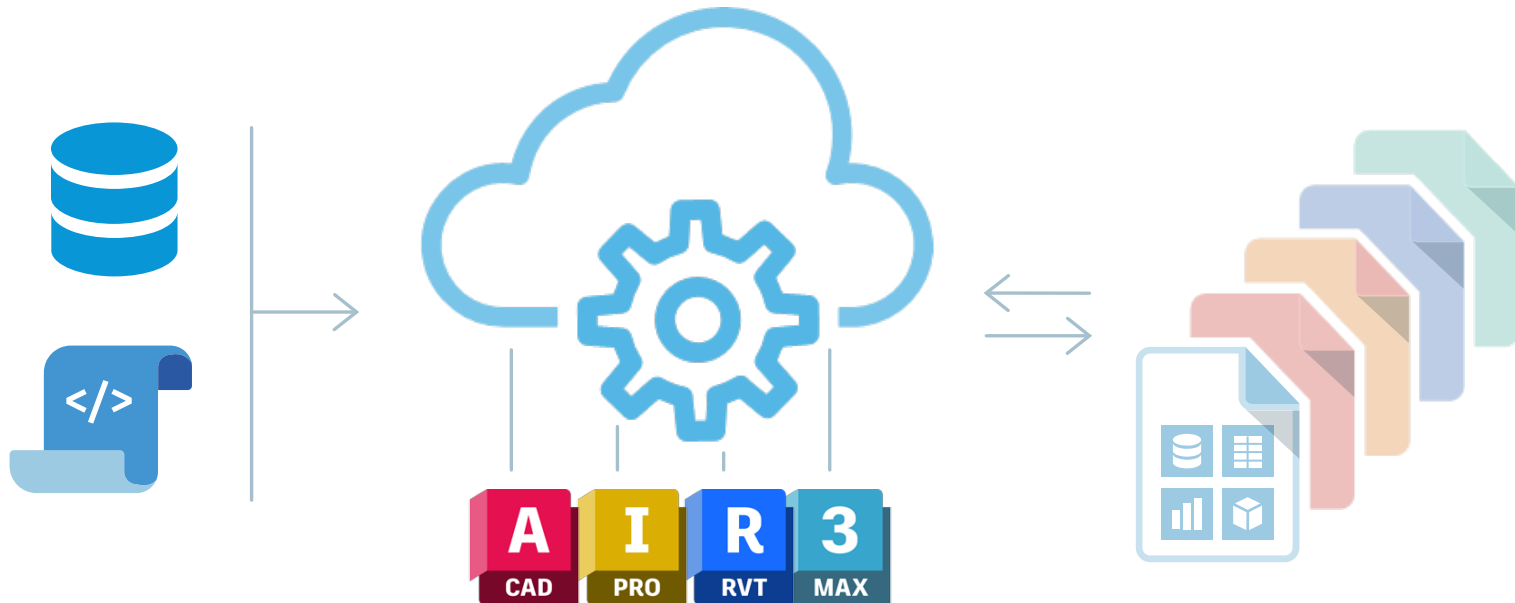


# Design Automation API 概要

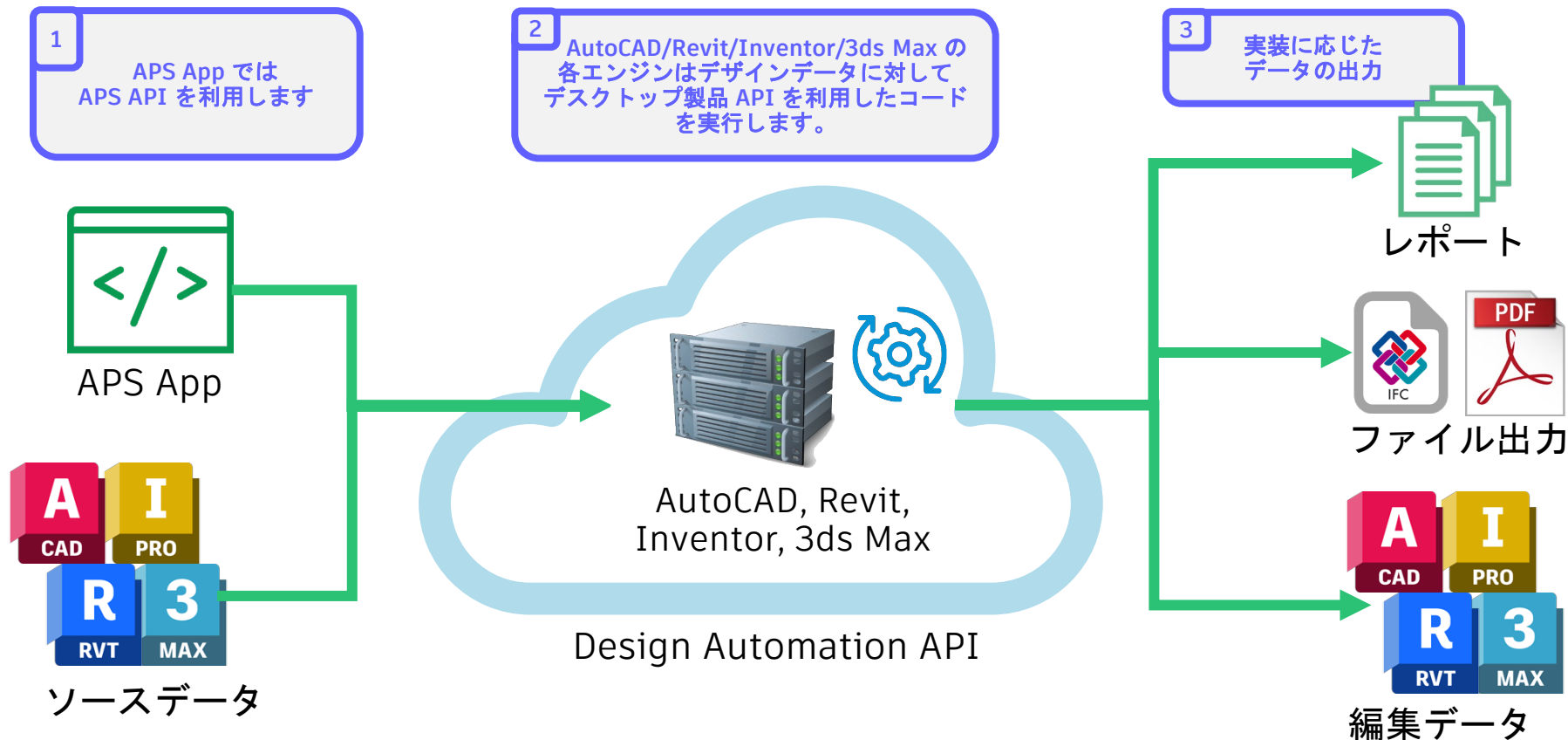


# Design Automation

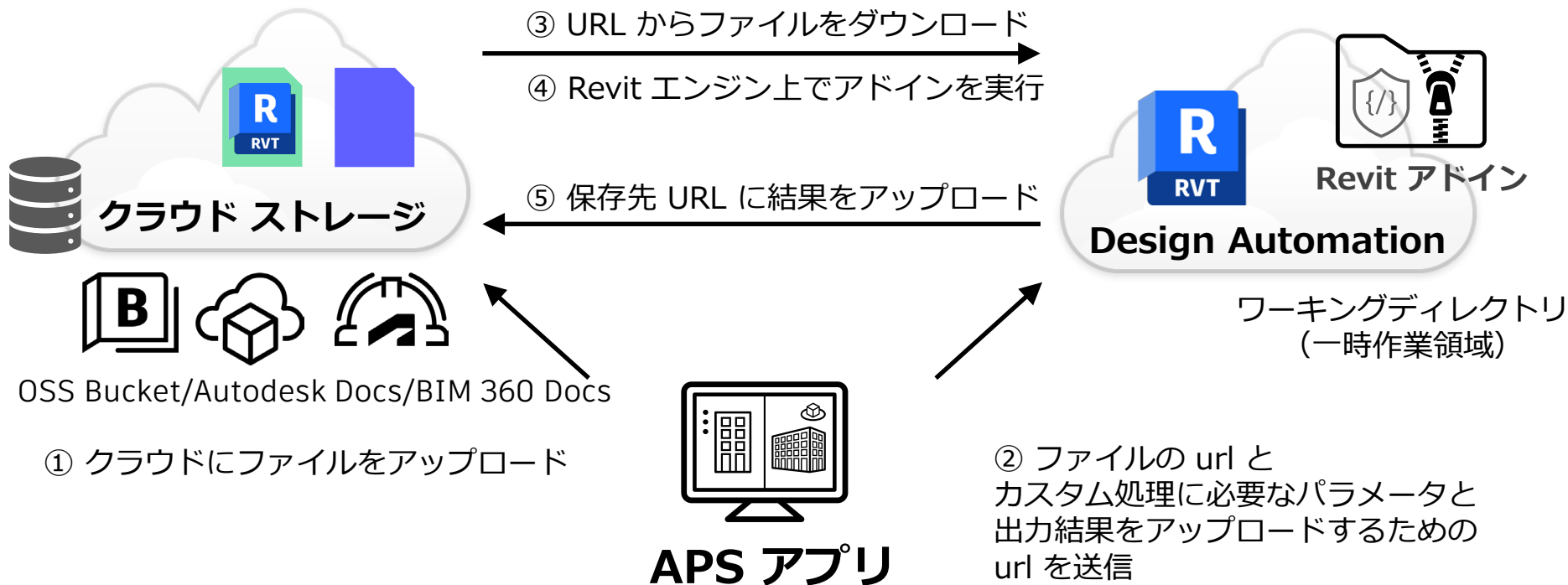
- データの作成・編集（繰り返し行う作業）の自動化 + コンフィギュレータ
- 各エンジンのネイティブデータ・ファイルフォーマットで動作



# データの作成と編集の自動化の仕組み



# Design Automation for Revit のワークフロー



# Design 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 Forge 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



# 3つのキーワード

## AppBundle



Revit アドインのパッケージ

Id : CustomApp

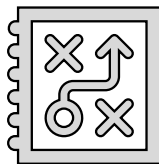
Engine : 2023

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。

# 関数のアナロジー

```
static void Activity(int first, int second, out int sum, out int product)
{
    sum = first + second;
    Console.WriteLine($"The sum of {first} and {second} is {sum}");
    product = first * second;
    Console.WriteLine($"The product of {first} and {second} is {product}");
}

static void Work()
{
    // Workitem 1
    int sum, product;
    Activity(first: 13, second: 17, sum: out sum, product: out product);

    // Workitem 2
    int another_sum, another_product;
    Activity(first: 23, second: 31, sum: out another_sum, product: out another_product);
}
```

# AppBundle と Activity の登録手順

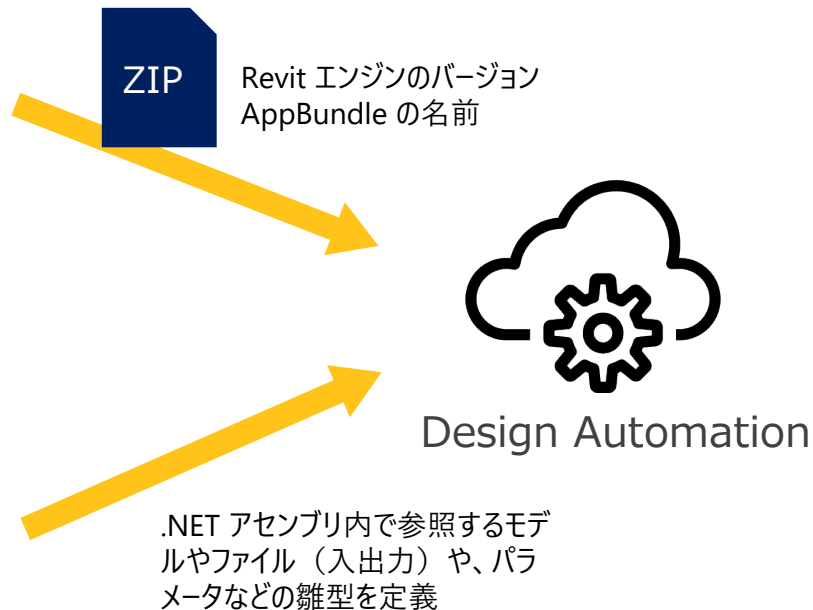
## AppBundle を登録

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



## Activity の定義例

**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

**"id": "UpdateRVTParamActivity", Activity ID**

**"commandLine": [ "\$(engine.path)\\\\\\revitcoreconsole.exe /i \"\$(args[inputRvtFile].path)\" /al \$(appbundles[UpdateRVTParam].path) ]**,

**"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": {**

**"zip": false,**

**"ondemand": false,**

**"verb": "put",**

**"description": "output file",**

**"required": true,**

**"localName": "OutputFile.rvt"**

**}**

**}, Engine**

**"appbundles": [ "MyFirstForgeAppNickname.UpdateRVTParam+dev" ], AppBundle ID**

**"description": "."**

入力 Revit ファイル

入力 JSON ファイル

出力 Revit ファイル

# Alias と Version

- AppBundle と Activity は、それぞれバージョンを追加していくことができます。
- 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できます。

AppBundle	Version	Alias
	1	
	2	Production
	3	
	4	Test
	5	Development

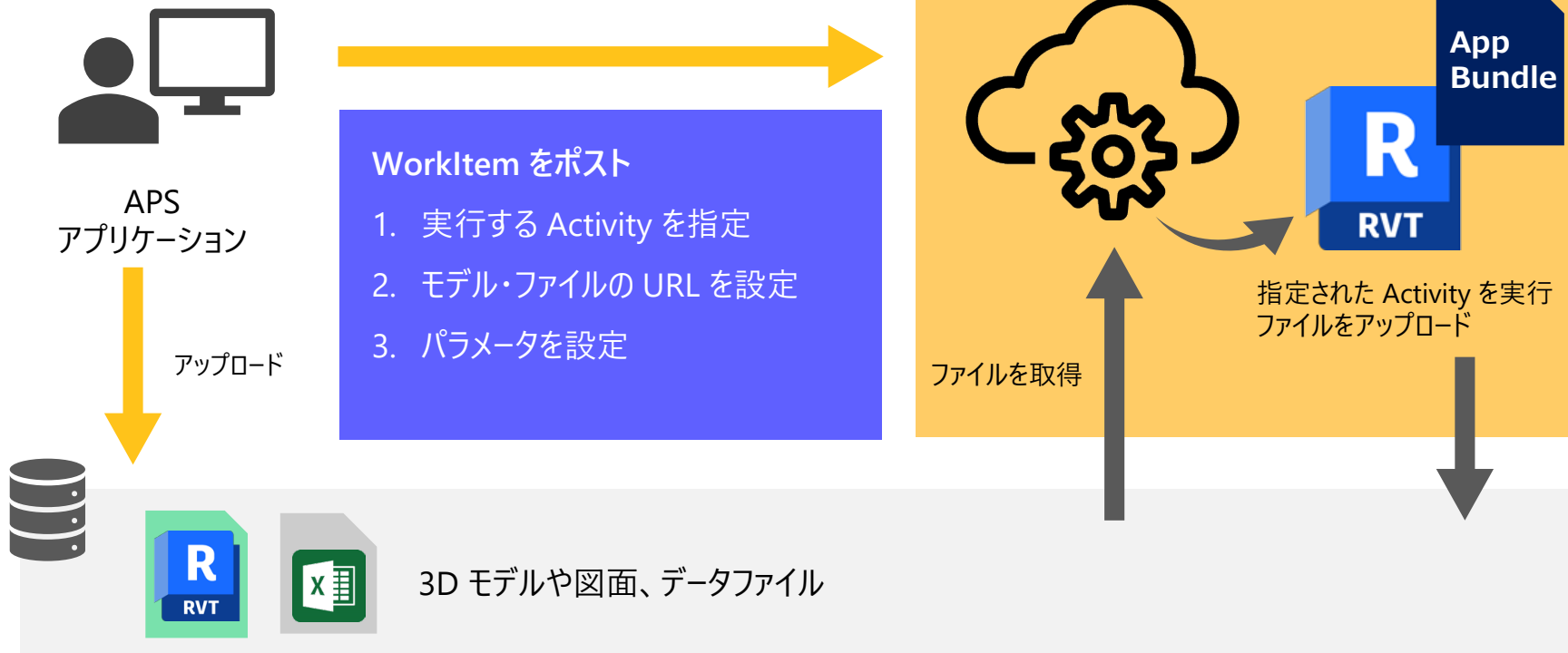
形式: YourNickname.SomeAppBundleId+SomeAliasName

例: MyFirstForgeAppNickname.DeleteWallApp+test

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

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。

# WorkItem を登録する



# WorkItem の例

```
1  {
2    .."activityId":.. "{{dasNickName}}.{{activityName}}+{{activityAlias}}",
3    .."arguments": {
4      ...."inputFile": {
5        ..... "url": .. "{{base_domain}}/oss/v2/buckets/{{bucketKey}}/objects/input.rvt",
6        ..... "headers": {
7          ..... → "Authorization": "Bearer {{access_token}}"
8        ..... }
9      .... },
10     .... "inputJson": {
11       ..... → "url": .. "data:application/json,{\"Width\": \"2\", \"Height\": \"4\"}"
12     .... },
13     .... "outputFile": {
14       ..... "verb": .. "put",
15       ..... "url": .. "{{base_domain}}/oss/v2/buckets/{{bucketKey}}/objects/output.rvt",
16       ..... "headers": {
17         ..... → "Authorization": "Bearer {{access_token}}"
18       ..... }
19     .... }
20   .. }
21 }
```

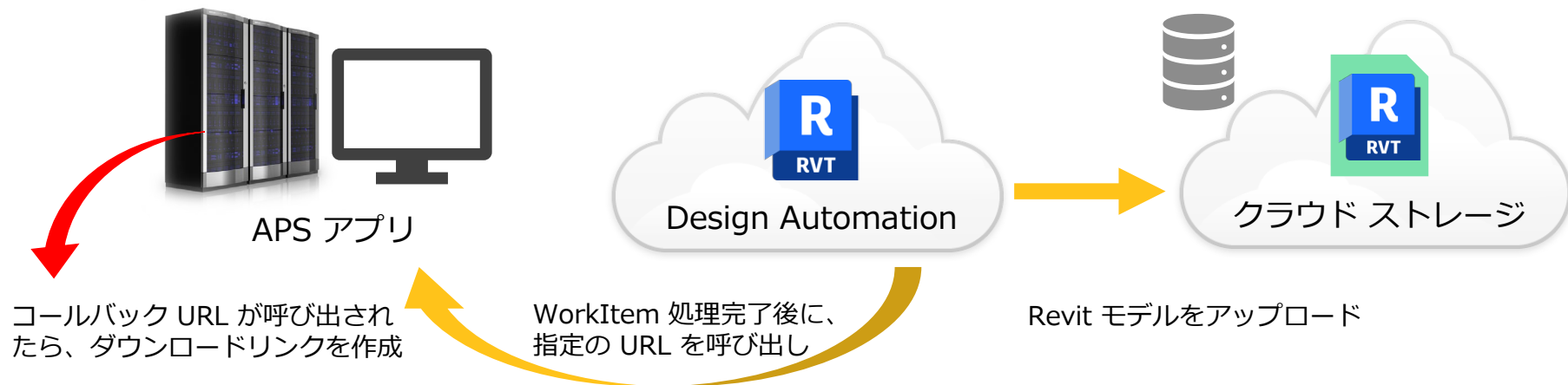
入力 Revit ファイル

入力 JSON データ

出力 Revit ファイル

# WorkItem の完了通知

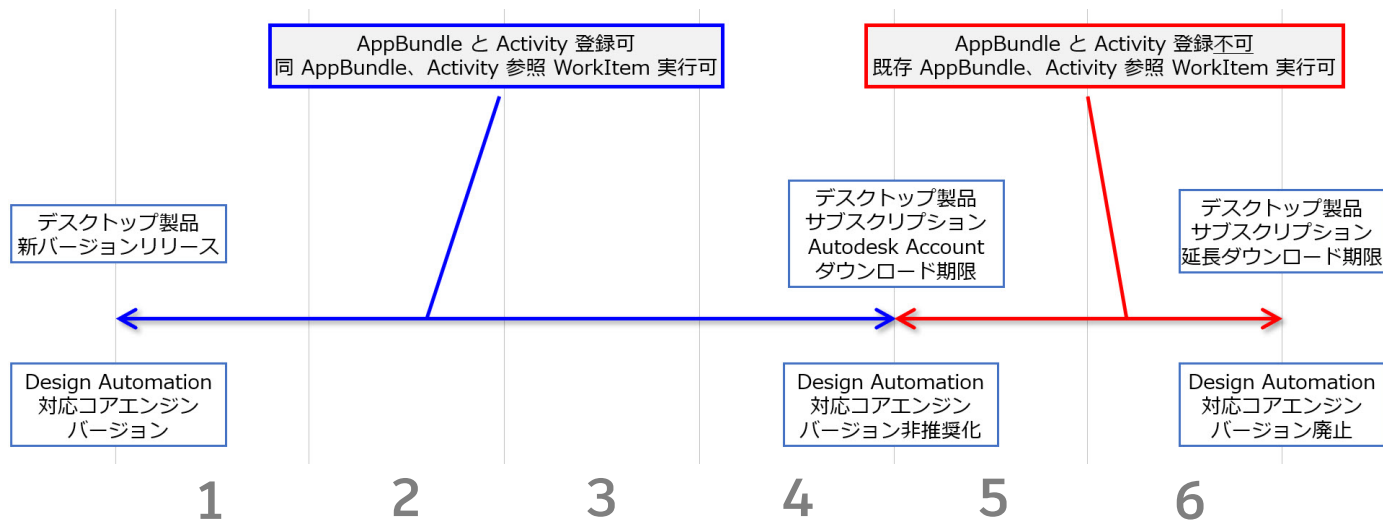
- WorkItem を POST する際に、“onComplete” という引数にコールバック URL を設定可能。
  - WorkItem の完了時に、**指定の URL** を自動的に呼び出してくれます。
  - コールバック URL には、**クエリパラメータ**を組み合わせたことができます。
- 進捗状況を30秒毎に “onProgress” コールバック URL で受け取ることもできます。
- Webhooks API でファイルアップロード・変換処理を検知





# コアエンジンのライフサイクル ポリシー

1. コアエンジンは、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。

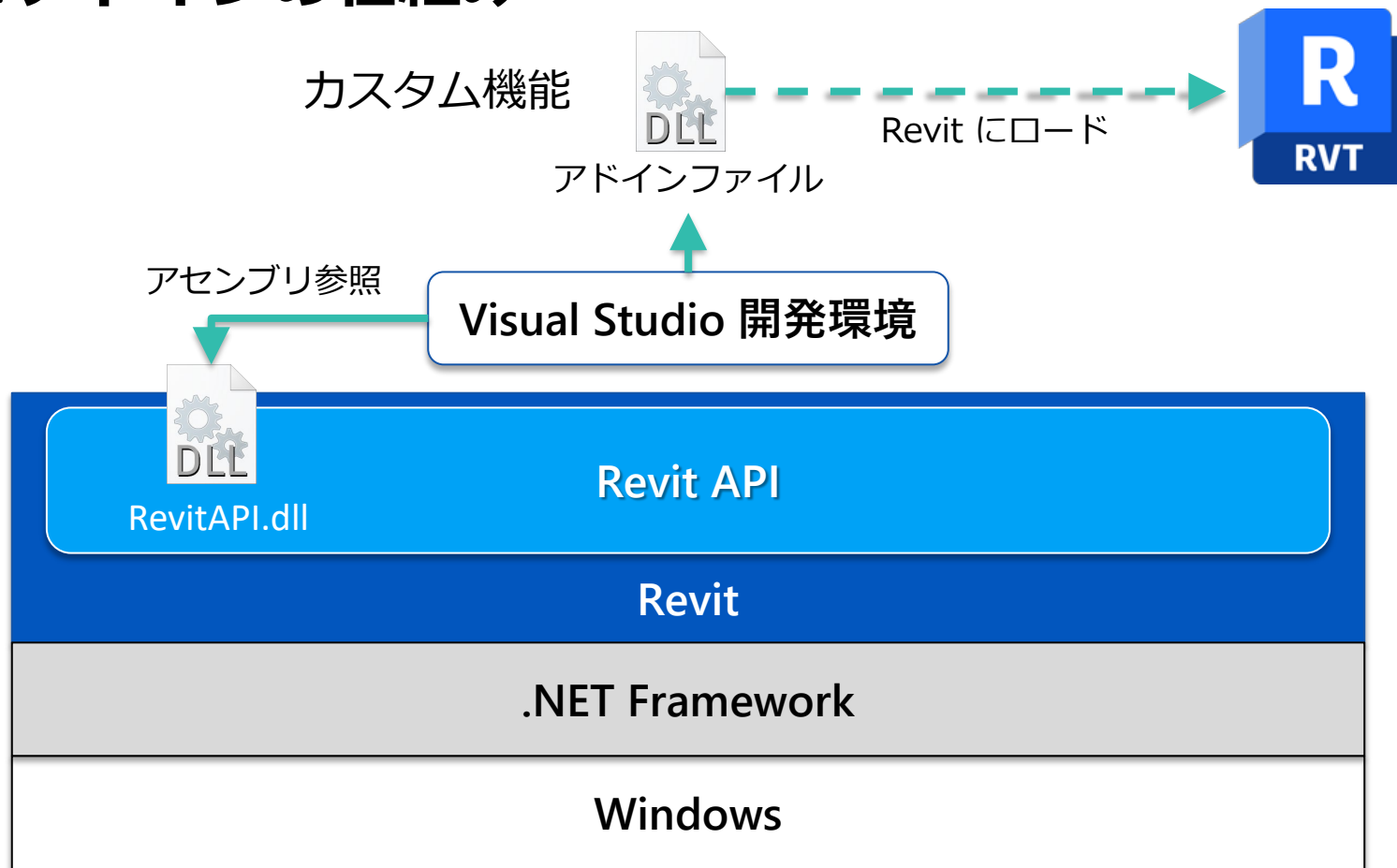




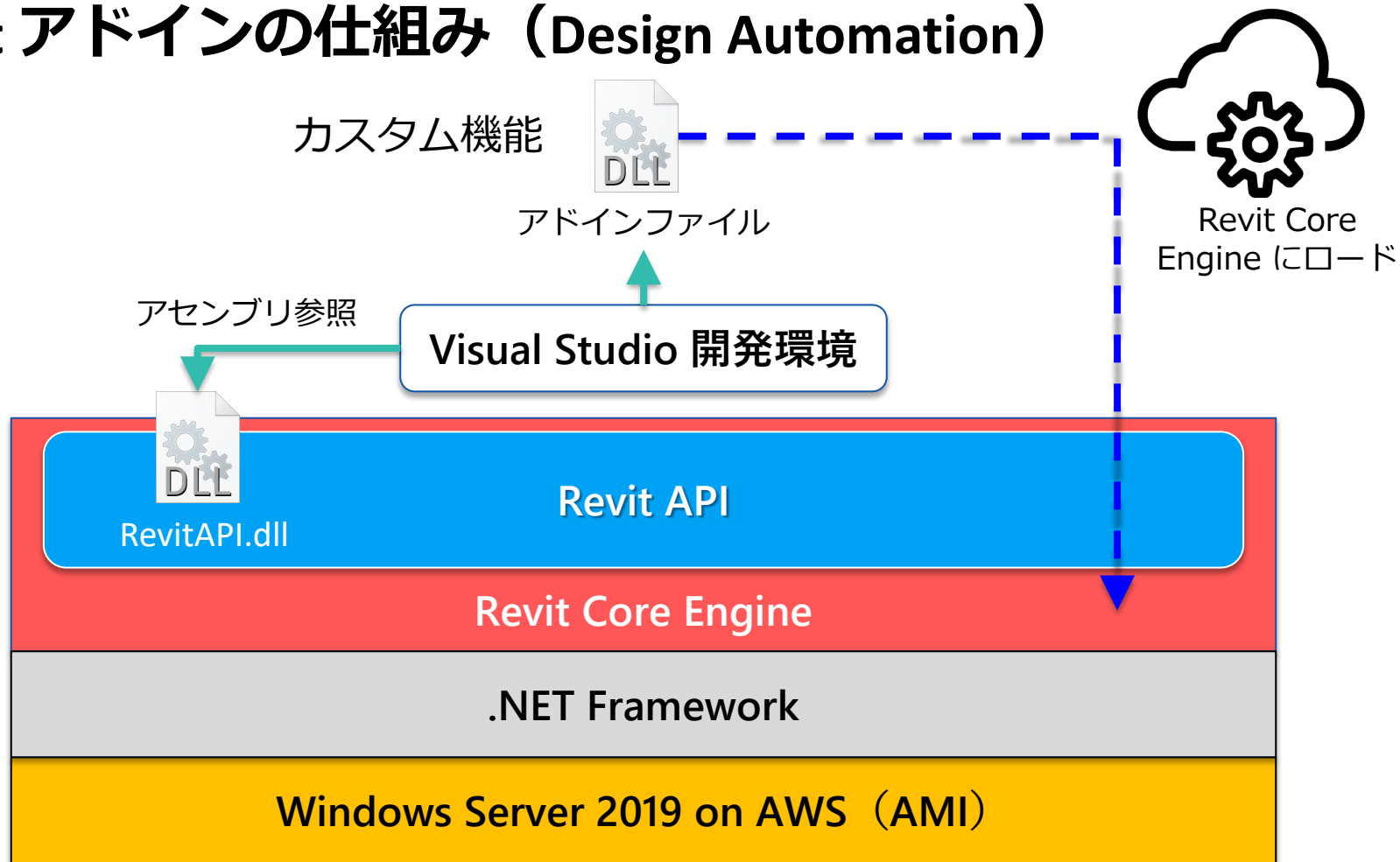
# AppBundle の作成方法



# Revit アドインの仕組み



# Revit アドインの仕組み (Design Automation)



# Revit プラグインの作成に必要なアセンブリ参照

ExternalDBApplication  
(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 Framework

Windows

# 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 }
```

# サーバーからのデータを Revit アドインで受け取る方法

```
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 ファイルでデータを取得可能

# ログの出力とデバッグオプション

- WorkItem の処理のログは、report.txt というファイルで取得できます。
- 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 にテキスト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); }
```

- WorkItem 実行時に使用した作業フォルダの内容を ZIP 圧縮して、ダウンロードできます。
  - POST workitems エンドポイントのリクエストボディ（JSON ペイロード）に指定

```
"adskDebug": {  
  "uploadJobFolder": true  
},
```

- GET workitems/:id エンドポイントでレスポンスを取得すると、“debugInfoUrl” から ZIP ファイルのダウンロード URL が得られます。（有効期間は 1 時間）

```
{  
  "status": "failedInstructions",  
  "reportUrl": "https://dasprod-store.s3.amazonaws.com/workItem/<<長いので省略>>",  
  "debugInfoUrl": "https://dasprod-store.s3.amazonaws.com/workItem/<<長いので省略>>",  
}
```



# 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;
}
```

# 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manifest ファイルを指定のフォルダ構成で格納して ZIP 圧縮したパッケージファイルです。

DeleteWallsApp.zip

```
|-- DeleteWalls.bundle
|  |-- PackageContents.xml
|  |-- Contents
|    |-- DeleteWalls.dll
|    |-- DeleteWalls.addin
```

```
1  <?xml version="1.0" encoding="utf-8" ?>
2  <ApplicationPackage>
3    <Components Description="SketchIt">
4      <RuntimeRequirements OS="Win64"
5        Platform="Revit"
6        SeriesMin="R2018"
7        SeriesMax="R2018" />
8      <ComponentEntry AppName="SketchIt"
9        Version="1.0.0"
10       ModuleName="./Contents/SketchIt.addin"
11       AppDescription="Sketches some walls and a floor."
12       LoadOnCommandInvocation="False"
13       LoadOnRevitStartup="True" />
14    </Components>
15  </ApplicationPackage>
```

# 注意事項

- 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。
- 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いません。
- ActiveView と ActiveDocument プロパティにはアクセスできません。
- 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）
- Revit を起動する OS は **Windows Server 2019 英語版**になります。
  - 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。
  - テキストの行間の位置がずれる既知の問題があります。
  - 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。
  - 利用可能なフォントのリストが公開済み。

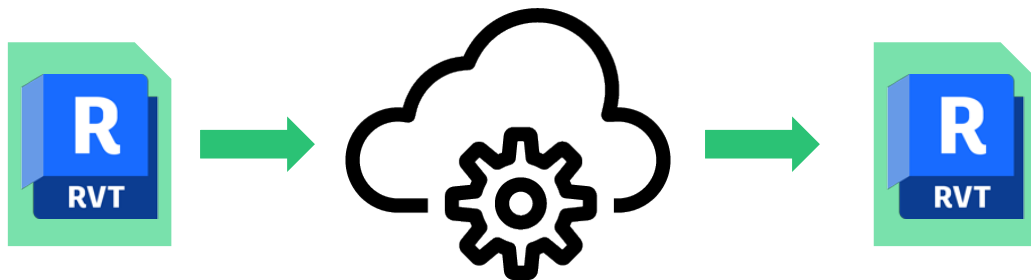


# OSS Direct アップロード



# OSS Direct アップロードのサポート

- WorkItem の入出力ファイルに OSS Bucket、Autodesk Docs、BIM 360 Docs を指定している場合、オブジェクトURNを指定して簡単にアップロードできるようになりました。



AWS S3 Bucket と Object に直接アクセス  
大容量ファイルのアップロード/ダウンロード速度が向上



# 変更点

- WorkItem リクエスト時に入力・出力ファイルの URL を指定する際、署名付き S3 URL を生成する必要がなくなりました。
- objectId と アクセストークン のみで OK !

```
{
  "activityId": "{{dasNickName}}.{{activityName}}+{{activityAlias}}",
  "arguments": {
    "inputRvtFile": {
      "verb": "get",
      "url": "urn:adsk.objects:os.object:test_bucket/Sample_Revit_Original.rvt",
      "headers": {
        "Authorization": "Bearer {{access_token}}"
      }
    },
    "resultRvtFile": {
      "verb": "put",
      "url": "urn:adsk.objects:os.object:test_bucket/Sample_Revit_Updated.rvt",
      "headers": {
        "Authorization": "Bearer {{access_token}}"
      }
    }
  }
}
```

# Object Storage Service (OSS) Bucket の場合

- 入力ファイル

- Bucket に保存されている Object の一覧を取得します。
  - bucketKey をキーに GET Objects エンドポイントを呼び出します。

```
{  
  "bucketKey": "test_bucket",  
  "objectKey": "サンプル意匠.rvt",  
  "objectId": "urn:adsk.objects:os.object:test_bucket/%E3%82%B5%E3%83%B3%E3%83%97%E3%83%AB%E6%84%8F%E5%8C%A0.rvt",  
  "sha1": "6be5ee13c56179c6a1bb8342a2a8d2a671910c22",  
  "size": 66318336,  
  "location": "https://developer.api.autodesk.com/oss/v2/buckets/test_bucket/objects/%E3%82%B5%E3%83%B3%E3%83%97%E3%83%AB%E6%84%8F%E5%8C%A0.rvt"  
}
```

- 出力ファイル

- “urn:adsk.objects:os.object:<BucketKey>/<ObjectKey>” の形式で直接パスを作成します。

# Autodesk Docs / BIM 360 Docs の場合

- 入力ファイル

- ファイルのバージョンは GET Version エンドポイントで取得し、objectId は response.data.relationships.storage.data.id で確認します。

```
"storage": {  
  "data": {  
    "type": "objects",  
    "id": "urn:adsk.objects:os.object:wip.dm.prod/ecc3802e-d16a-46c8-afdc-aa6e482a93c2.rvt"  
  },  
  ~ 省略 ~  
}
```

- 出力ファイル

- POST projects/:project\_id/storage エンドポイントを使用して、ファイルをアップロードするためのストレージを作成します。この際、objectId は response.data.id で取得することが出来ます。



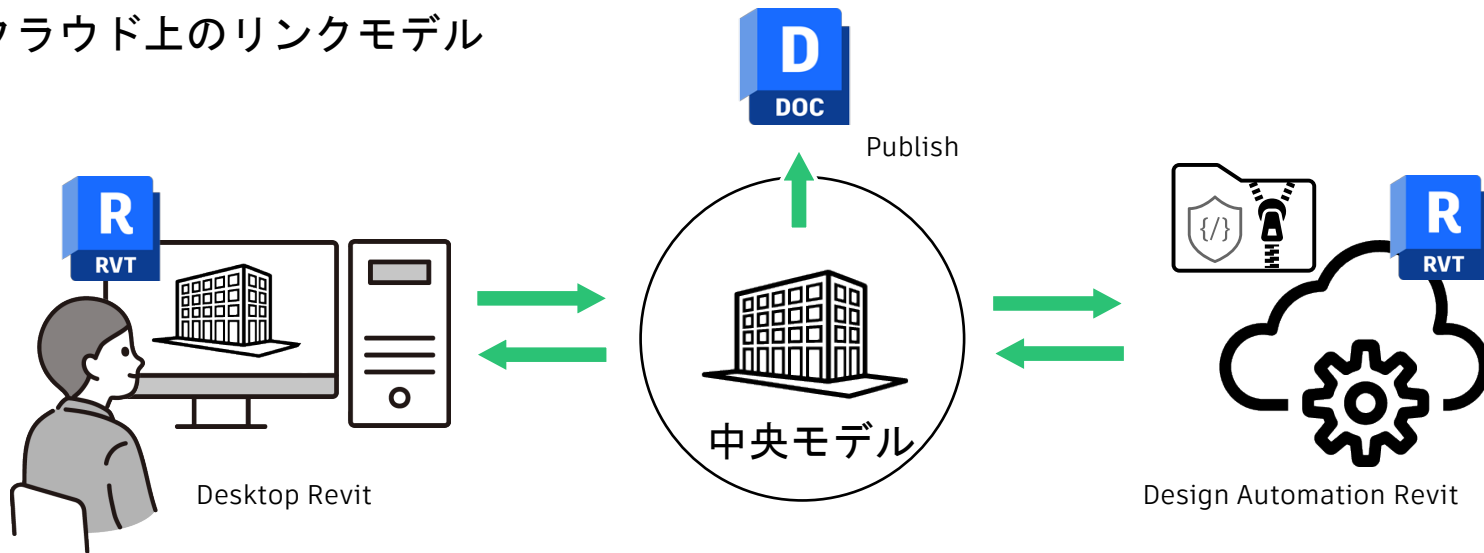


# Revit クラウドモデル



# Revit クラウドモデルのサポート

- ワークシェアされていないクラウドモデルと、ワークシェアされているクラウドモデルのどちらもサポート。
- Revit 2022, 2023, 2024 対象
- ファイルを開く、保存、編集、チェックアウト、中央モデルと同期
- クラウド上のリンクモデル



# Revit クラウドモデルの利用方法

- WorkItem リクエスト時に、scope=code:all を指定して取得した 3-legged アクセストークンを adsk3LeggedToken プロパティで渡します。
  - これにより Design Automation Revit でクラウドモデルへのアクセスが有効になります。

```
{  
  inputXls: {  
    url: "XXXXXXXXXXXX"  
  },  
  onComplete: {  
    verb: "post",  
    url: designAutomation.webhook_url  
  },  
  adsk3LeggedToken: access_token  
}
```

# Revit クラウドモデルを作成する

- AppBundle (Revit アドイン) から直接、新規にワークシェアリングモデルを作成することができます。
- モデルのセットアップ作業を自動化したい場合に非常に便利です。

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

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

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

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

# Revit クラウドモデルを開く

- AppBundle のアドインからクラウドモデルを直接開くことで、モデルへのアクセス処理が大幅に簡素化されます。
- WorkItem リクエスト時の入力モデルの URL 設定は不要に。
- Region、ProjectGuid、ModelGuid の情報を渡します。

```
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/同期する

- 出力する Revit モデルをストレージの URL を指定してアップロードする必要はありません。
- 以下のコードを使用して、編集内容を直接クラウドモデルに保存することができます。

```
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();
}
```

# 中央モデルと同期後のパブリッシュ

- 中央モデルに変更内容を同期した後、モデルをパブリッシュする必要がある場合
  - onComplete() コールバックから Data Management API の Publish Model Command を使ってパブリッシュすることができます。
  - "commands:autodesk.bim360:C4RModelPublish"

**POST** [https://developer.api.autodesk.com/data/v1/projects/:project\\_id/commands](https://developer.api.autodesk.com/data/v1/projects/:project_id/commands)

# パブリッシュの完了通知

- Data Management API
  - GetPublishModelJob コマンドをポーリング
- Webhooks API
  - Revit Cloud Worksharing Events
    - model.sync : 中央モデルと同期後に通知。
    - model.publish : モデルがパブリッシュ開始・パブリッシュ中のキューにあるかどうか通知。
  - ACC/BIM 360 にモデルをアップロードした際、自動的に Model Derivative サービスが実行されて SVF/SVF2 変換が実行されます。この処理の完了通知を受け取れるようになりました。
    - extraction.finished : 変換処理終了後に通知
    - extraction.updated : 変換処理過程を通知
    - SVF 変換後、SVF2 変換後の 2 回呼び出されます。



[Webhook now supports translation progress of BIM360 or ACC files](#)

[Forge Online - WebHooks : イベント通知と活用](#)



# Workshared Revit Cloud Model – Revit API



*Revit 2022~ enhancement*

Revit API Class and Method	Description
Document. <b>SaveAsCloudModel</b> ( Guid    accountId, Guid    projectId, String  folderId, String  modelName)	Autodesk Docs にモデルを保存します。 <ul style="list-style-type: none"><li>• ワークシェアされていないモデル</li><li>• ワークシェアリングモデル</li></ul>
<b>RevitLinkType.Create</b> ( Document    document, ModelPath    path, RevitLinkOptions  options)	Revit リンクのタイプを作成し、リンクドキュメントをロードします。 ModelPathでクラウドのパスをサポート済み。
<b>RevitLinkType.LoadFrom</b> ( ModelPath    path, WorksetConfiguration  config)	ローカルまたはクラウドの Revit リンクをロード・再ロードします。

# Workshared Revit Cloud Model – Revit API



Revit API Class and Method	Description
Document. CanEnableCloudWorksharing() EnableCloudWorksharing() GetCloudModelPath() SaveCloudModel()	クラウドワークシェアリングが有効か判定 クラウドワークシェアリングを有効化 クラウドモデルパスを取得 クラウドモデルを保存
Document. GetCloudFolderId() <sup>*1</sup>	クラウドモデルの APS DM folder id を取得
ModelPathUtils. ConvertCloudGUIDsToCloudPath() <sup>*2</sup>	クラウドのプロジェクトとモデルの GUID から、クラウドモデルパスに変換

\*1) 2022 enhancement    \*2) 2021 enhancement

# Revit Model in Docs Data Management API

e.g., GET contents/items/versions  
Response

```
{
  "type": "versions",
  "id": "urn:adsk.wipprod:fs.file:vf.b909RzMKR4mhc3O7UBY_8g?versi",
  "attributes": {
    "name": "rac_basic_sample_project.rvt",
    "displayName": "rac basic sample project.rvt",
    ...

    "versionNumber": 4,
    "mimeType": "application/vnd.autodesk.r360",
    "storageSize": 111297725,
    "fileType": "rvt",
    "extension": {
      "type": "versions:autodesk.bim360:C4RModel",
      "version": "1.1.0",
      "schema": {
        "href":
          "https://developer.api.autodesk.com/schema/v1/versions/versions:autodesk.bim360:C4RModel-1.1.0"
      }
    },
    "data": {
      "modelVersion": 3,
      "isCompositeDesign": true,
      "mimeType": "application/vnd.autodesk.r360",
      "compositeParentFile": "rac_basic_sample_project.rvt",
      "projectGuid": "733f13b0-502d-4a1e-6624-b7f22f920290",
      "originalItemUrn":
        "urn:adsk.wipprod:dm.lineage:b909RzMKR4mhc3O7UBY_8g",
      "modelType": "multiuser",
      "latestEpisodeGuid": "fda2d360-5466-4ev3-b051-0dacbaa509f5",
      "modelGuid": "d1e3bc70-30d0-4e42-86a5-9d00da685Taf",
      "processState": "PROCESSING_COMPLETE",
      ...
    }
  }
}
```

File type

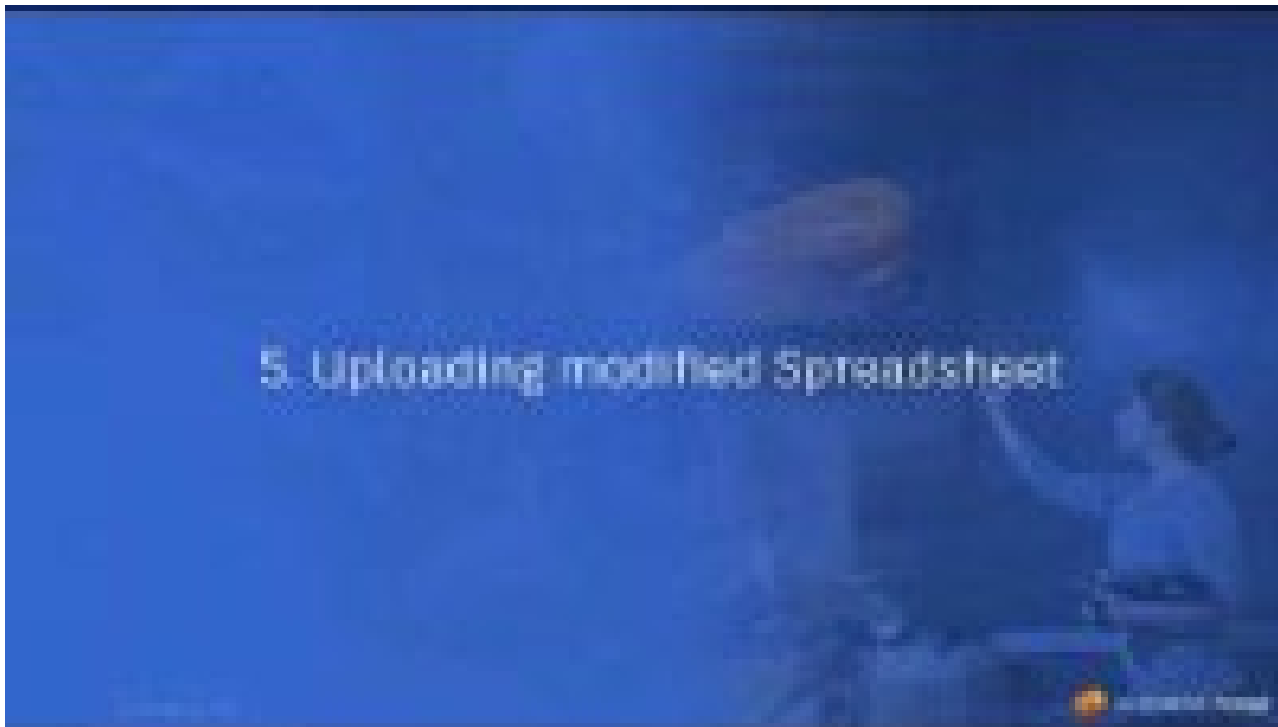
Workshared model

Composed with multi-files

Parent or host file

Workshared. If "singleuser", non-workshared

# サンプル



- [design.automation-nodejs-revit.rcw.parameters.excel](#)
- [Design Automation API Supports Revit Cloud Model](#)
- [New sample to support access Revit Cloud Model from Revit Design Automation Engine](#)



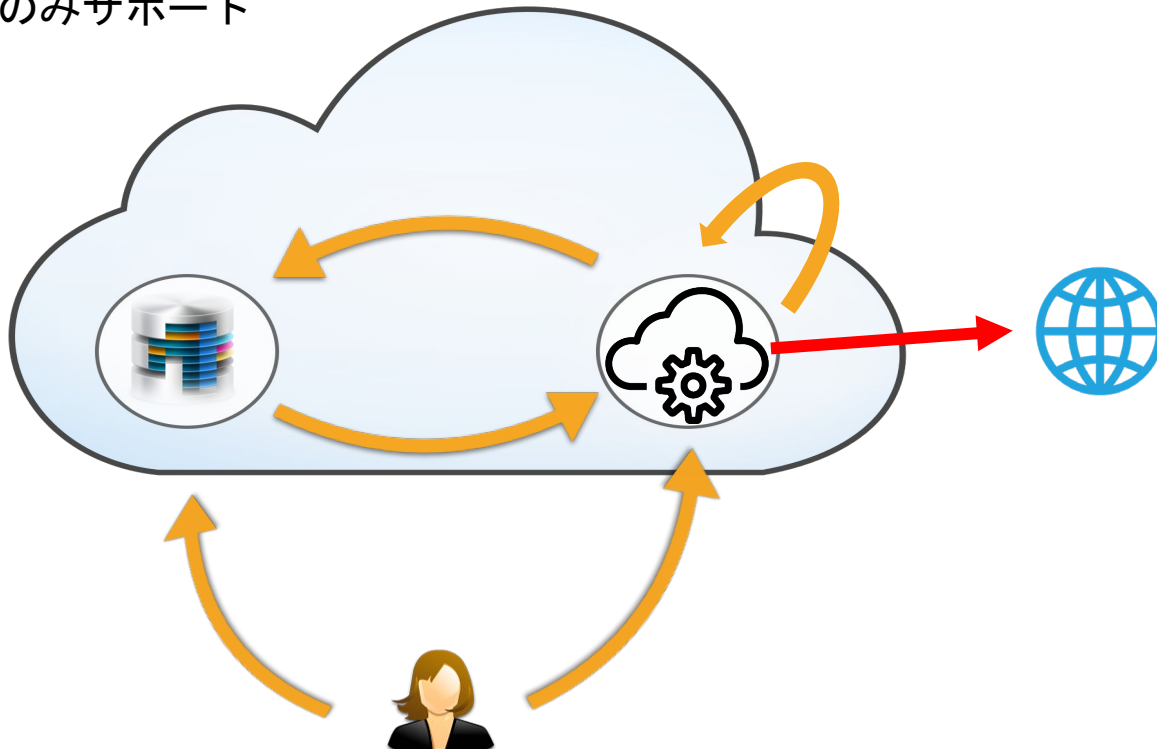
# ネットワークアクセス



# 外部ネットワークアクセスのサポート(New)

- https のリクエストのみサポート

- 気象データ
- GIS 情報
- カタログ情報など





# Design Automation for Revit **with Dynamo**



# Design Automation for Revit with Dynamo

Keep it secret, keep it safe

Please remember to keep all feedback, project, and product information completely confidential.

- Welcome to the Design Automation with Dynamo Beta project Last month 



Design Automation  
for Revit

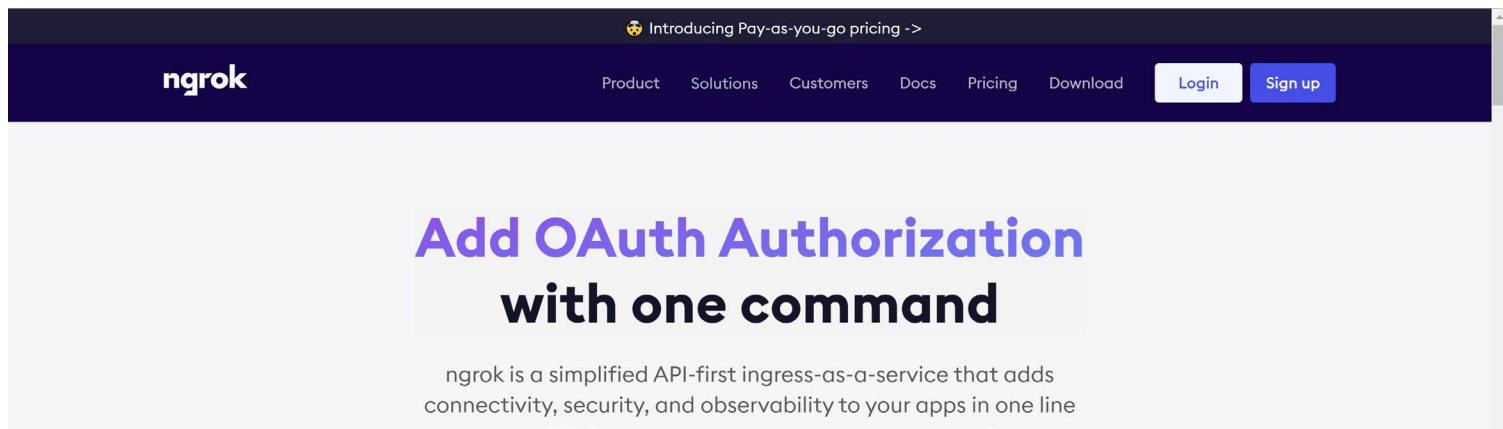


Dynamo

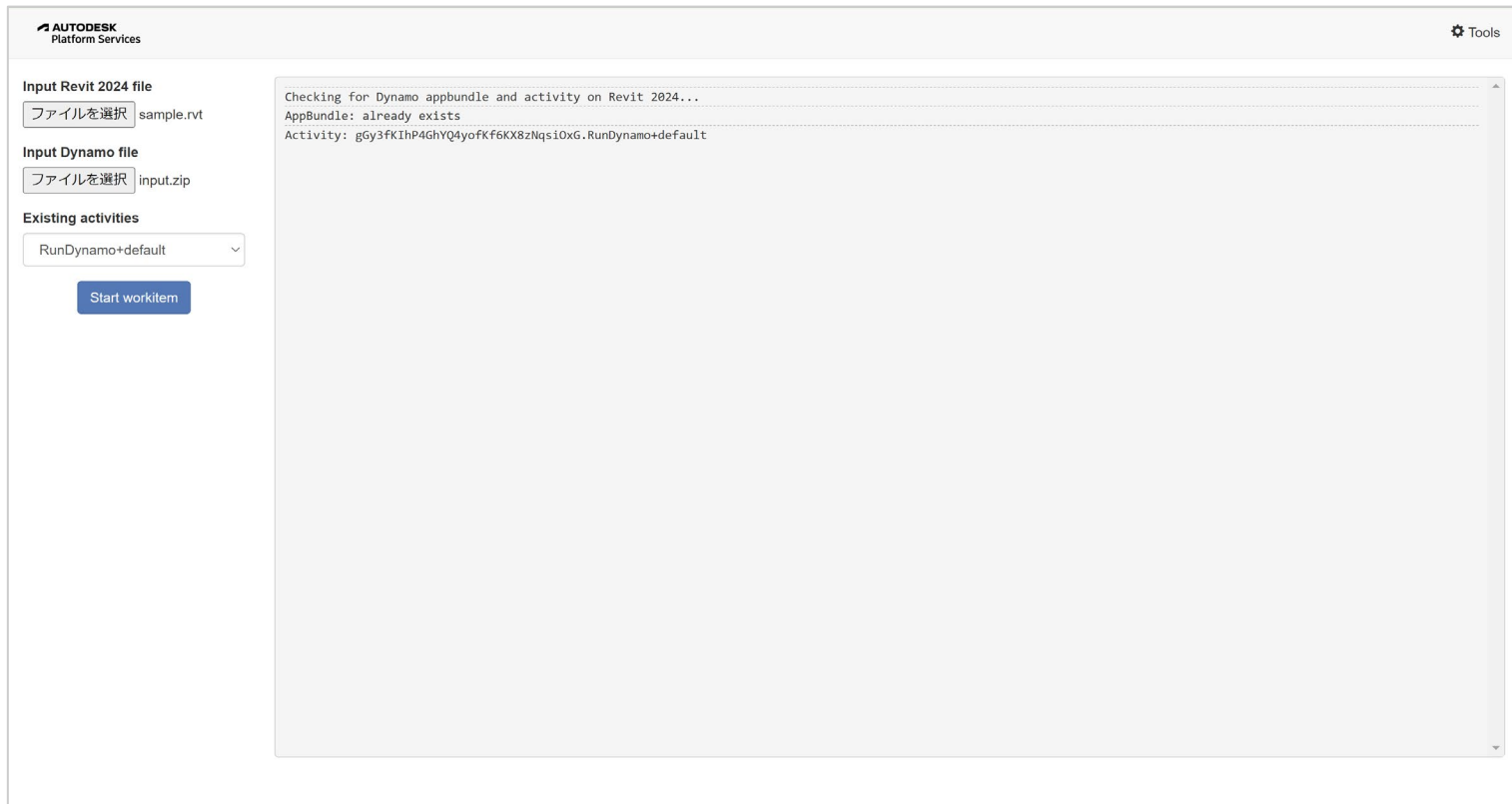


# Nodejs サンプル + Postman Collection

- AppBundle と Dynamo グラフが同梱
- APS App (Client Id, Client Secret, Callback URL)
- ngrok ツール
  - ローカルサーバのポート（ローカルホスト）に外部ネットワークから直接アクセスすることを可能にするトンネリングツール



# APS Tutorial ベース



# サンプル概要

- AppBundle と Activity を自動的に作成
- WorkItem の入力パラメータ
  - Revit サンプルモデル
  - ZIP 圧縮したファイル (Input.zip)
- Input.zip フォルダ
  - input.json
    - 実行する Dynamo グラフの名前
    - 入出力フォルダ
    - Dynamo のパッケージ (カスタムノード)
    - 編集するノード
  - python-3.9.12-embed-amd64.zip

- Dynamo グラフのファイル
  - GetCategoryParameters.dyn
  - GetProjectCounts.dyn
- Dynamo グラフのファイル
  - Mark Doors.dyn

# Dynamo グラフの内容

- GetProjectCounts.dyn
  - プロジェクトのファミリインスタンスの要素数を、カテゴリごと、ファミリごと、タイプ毎に集計して CSV を出力します。
- GetCategoryParameters.dyn
  - 特定のカテゴリのファミリインスタンスを全て取得して、そのインスタンスのパラメータを CSV で出力します。
- Mark Doors.dyn
  - モデル内の全てのドアインスタンスを取得し、どの部屋に属するかを調べます。属する部屋番号とドアのマークを組み合わせ、ドアのパラメータに書き込みます。

# Activity パラメータ定義と input.json

```
- "parameters": {  
-   .... "rvtFile": {  
-       .... "zip": false,  
-       .... "ondemand": false,  
-       .... "verb": "get",  
-       .... "description": "Input Revit model",  
-       .... "required": true  
-   },  
-   .... "input": {  
-       .... "zip": true,  
-       .... "ondemand": false,  
-       .... "verb": "get",  
-       .... "description": "Input Dynamo graph(s)",  
-       .... "required": true,  
-       .... "localName": "input.zip"  
-   },  
-   .... "result": {  
-       .... "zip": true,  
-       .... "ondemand": false,  
-       .... "verb": "put",  
-       .... "description": "graph result",  
-       .... "required": true,  
-       .... "localName": "result"  
-   }  
- },
```

## input.json の例

```
"GraphName": "GetProjectCounts.dyn",  
"InputFolder": "input.zip",  
"InputFileNames": [],  
"ResultFolder": "result",  
"Packages": ["packages"],  
"NodeInput": [{  
    "Id": "2ece77cbe7be4845b3d4d94abea6410f",  
    "Name": "Value",  
    "Value": "{3D}"  
},  
],
```

# ベータ版評価のお願い

- Design Automation API for Revit で Dynamo を実行出来るよう取り組んでいます。
- Dynamoグラフ スクリプトを実行して Revit ファイルを開き、自動処理して結果を得ることが出来るようになります。
- ベータ プログラムにご参加いただき、Design Automation API for Revit で Dynamo グラフ スクリプトを評価、フィードバックをいただけると大変助かります。
- Revit 2024 コア エンジン バージョンのみサポート。
- この評価プログラムでは、Revit 用 Dynamo 2.18 の特別バージョンを含む AppBundle (zip ファイル) が必要になります。

# 参加方法

- ベータ ポータルのフィードバック プロジェクト（英語） にアクセス（要 Autodesk ID でサインイン）します。
- サンプル コードと Postman Collection をダウンロードして動作をご確認ください。
  - Nodejs のプロジェクトからサンプルの Appbundle を実行可能です。
  - 詳細は README.md に記載。
- フォーラムにて、実現したいワークフローをお知らせください。

# Design Automation for Revit (New)

- OSS ダイレクト アップロード / ダウンロード
- Revit Cloud Models (Revit 2022+)
- オープンネットワーク
- キュータイムの改善
- Design Automation for Revit with Dynamo



# Revit - Next Major Release



- **.NET Core 6/7/8** (TBD)
  - プレビューリリースは 11月を予定 (TBD)
- Revit マクロマネージャの変更
  - .NET Core (上記と同じ)
  - Microsoft Visual Studio Code
  - C# のみサポート
  - 既存のマクロとは非互換
  - **ドキュメントレベルのマクロがなくなり、アプリケーションレベルのみサポート**
  - 移行ガイドを参照ください。





# Design & Make Platform