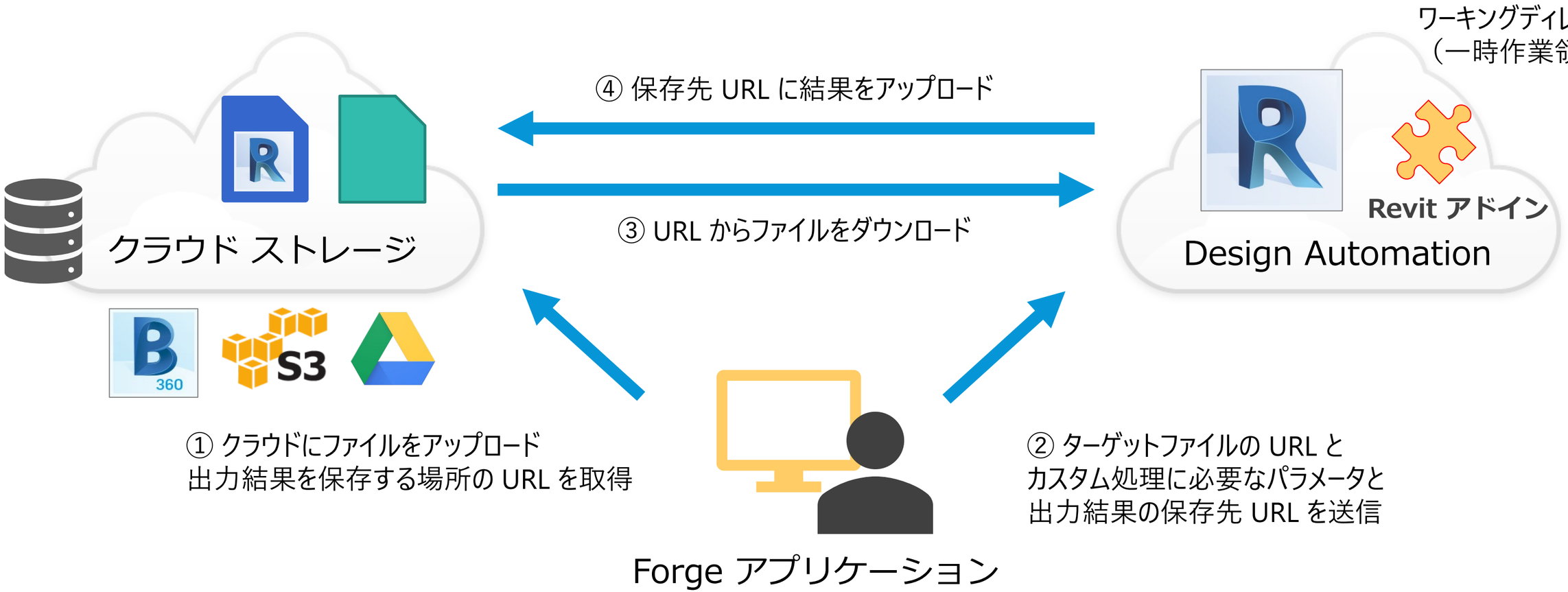
Forge Online

Revit タスクの自動化：Design Automation API for Revit の理解

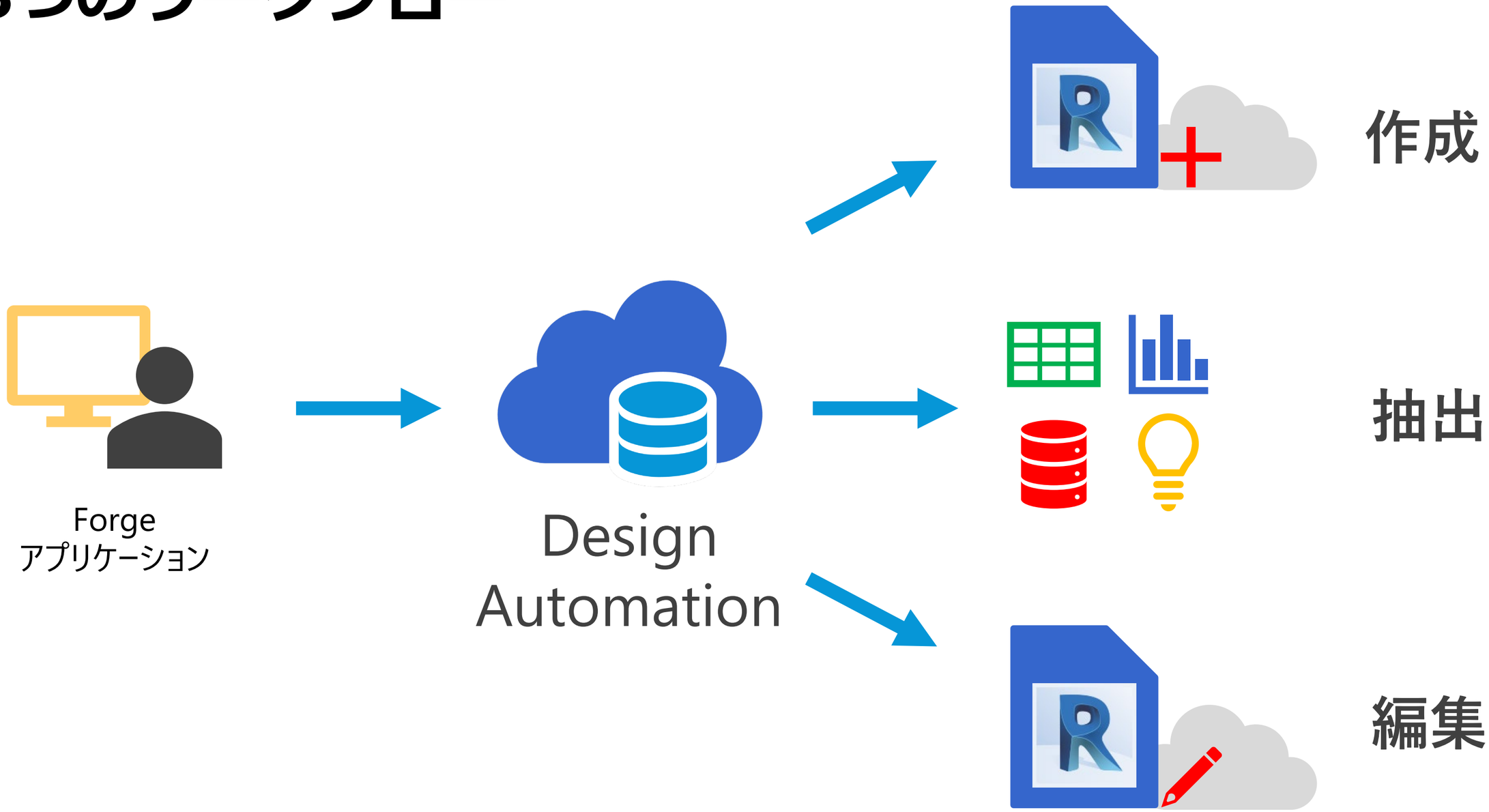
小笠原 龍司
オートデスク 株式会社

オンデマンドで Revit データをクラウド処理

- Revit がローカル環境になくても、クラウドサービスを通じて Revit アドインにフルアクセス
- Revit API を利用して開発した Revit アドインを実行
- どこからでもデータを読み取り、どこにでもデータを保存



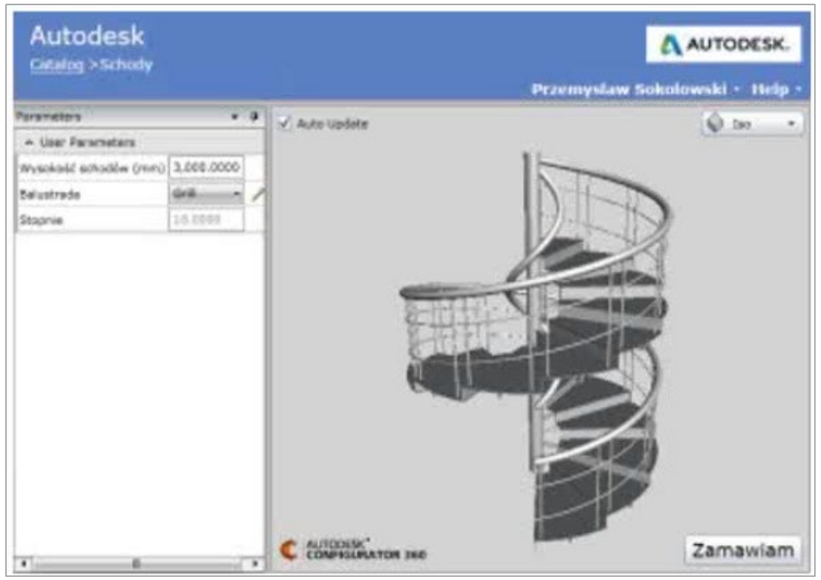
3つのワークフロー



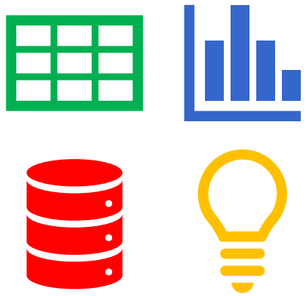
ワークフロー：作成

- ファミリ（RFA）を作成
- レイアウトアプリケーションから RVT プロジェクトを作成
- サードパーティのフォーマットから RVT/RFA に変換
- 仕様書から設計図書を自動生成

想定イメージ



ワークフロー：抽出



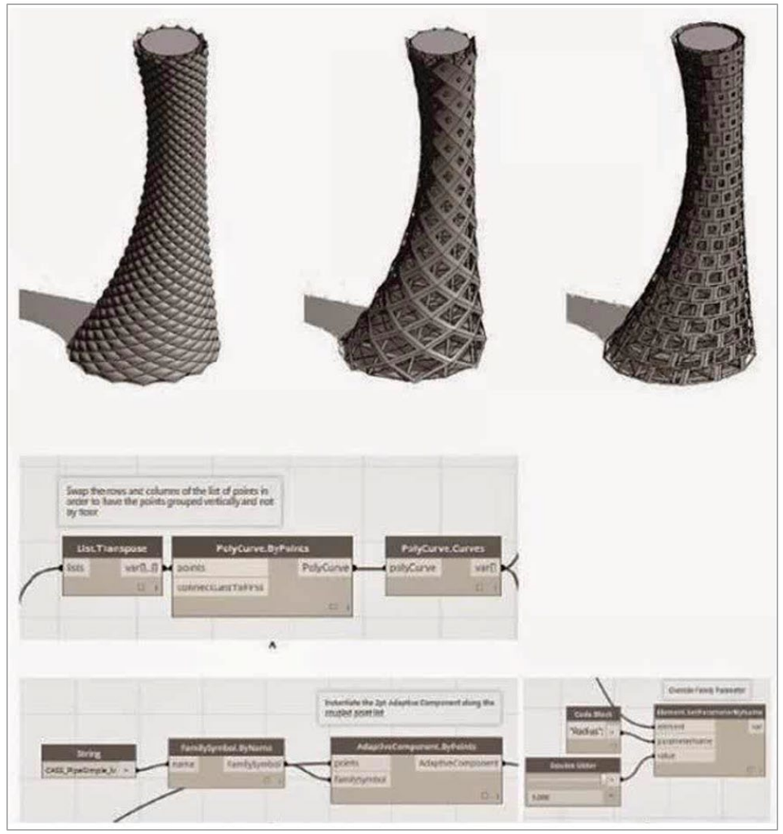
- カスタムの抽出処理（CSV、外部DB、未サポートフォーマットなど）
- 抽出の自動実行
- モデルの品質保証/品質管理（QA/QC）チェック
- Revit を所有していないコンサルタントとモデルの情報を共有

想定イメージ

ワークフロー：編集

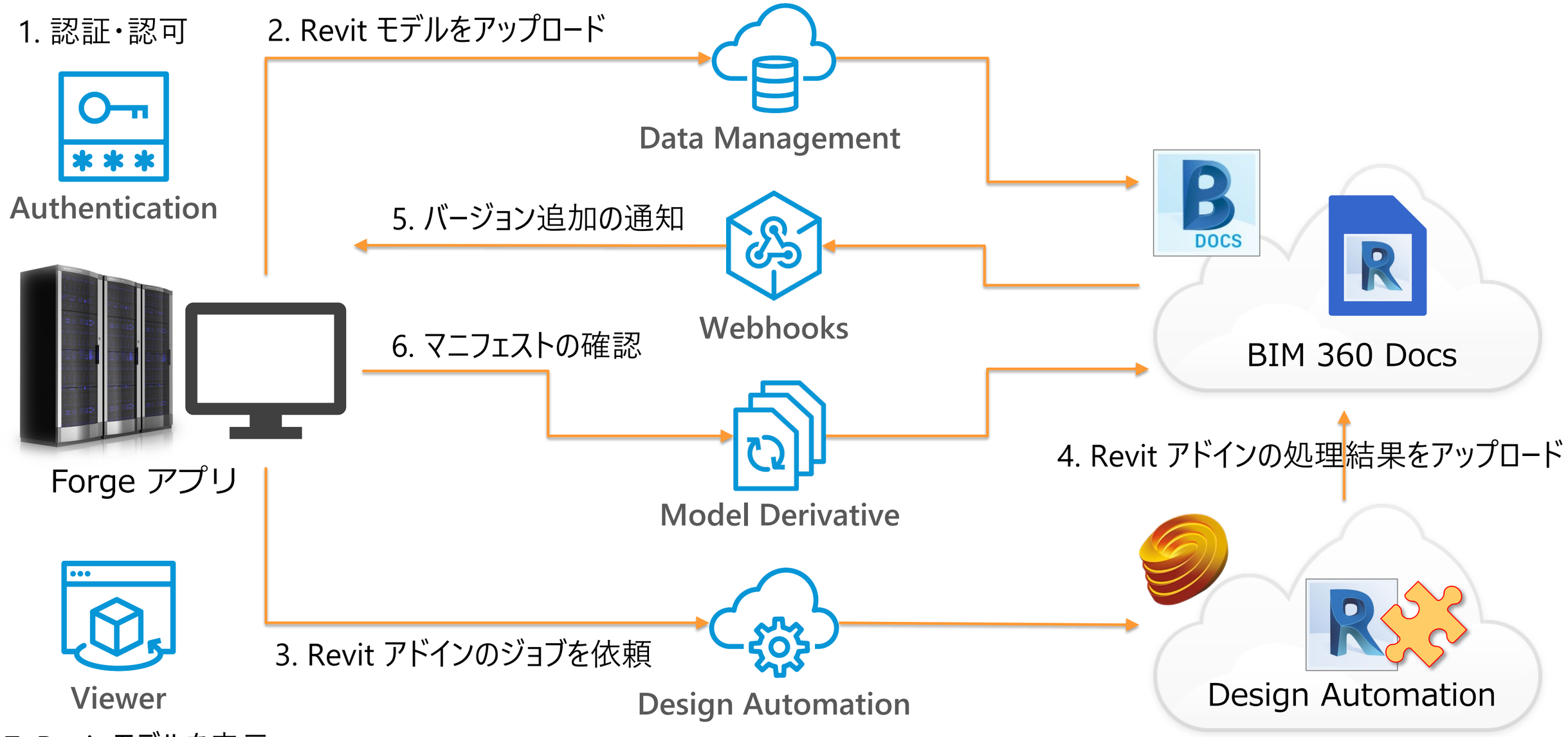
- モデリングや設計図書の誤りを修正
- Upgrade/Update のバッチ実行
- 古いデータを自動的に最新版に更新
- それ以外の編集処理

- 将来サポート予定
  - Revit Cloud Worksharing モデル対応
  - BIM 360 中央モデルに変更内容を同期
  - Dynamo スクリプトの実行



想定イメージ

Forge API との連携例



Forge Online

Revit タスクの自動化：Revit アドインの DA4R パッケージへの変換

小笠原 龍司
オートデスク 株式会社

Revit アドインのバンドルパッケージの準備手順

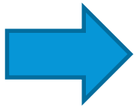
Revit アドインの変換

① .NET C# で実装したアドインプロジェクトを用意

② ローカルの Revit で動作確認

③ DesignAutomationBridge.dll を参照に追加
※ RevitAPIUI.dll は参照を削除

④ IExternalApplication または、IExternalCommand を IExternalDBApplication に変更



バンドルパッケージの作成

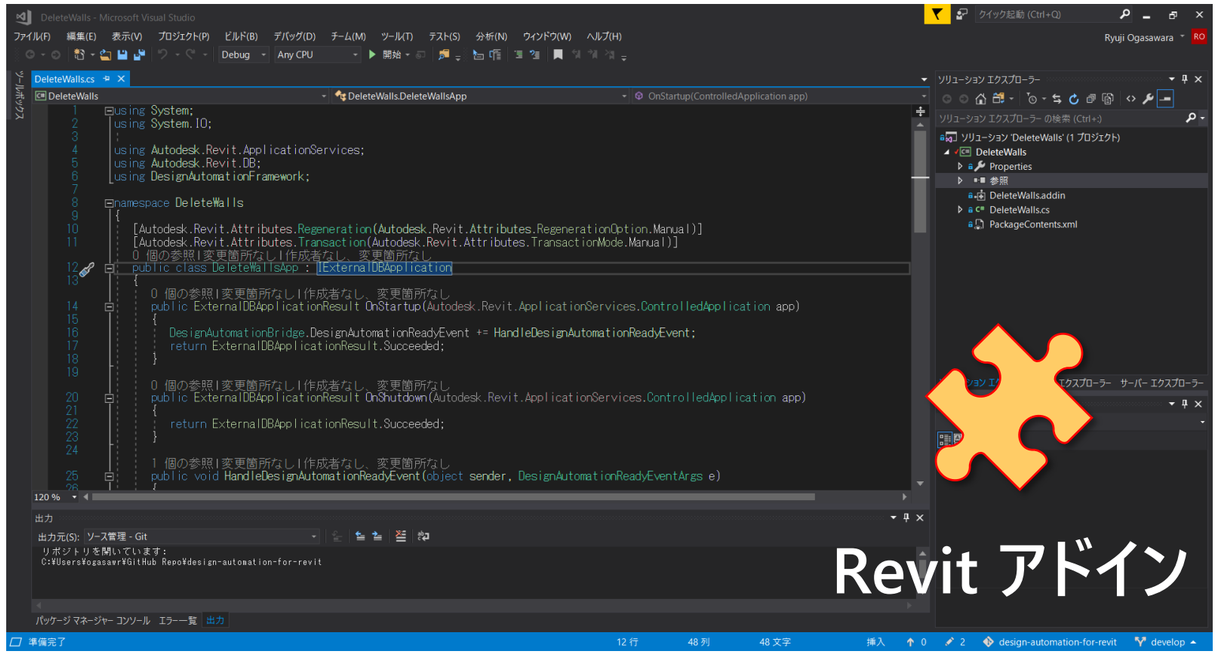
.bundle フォルダ構成のパッケージを ZIP ファイルに圧縮

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

https://adndevblog.typepad.com/technology_perspective/2019/02/understanding-steps-to-use-design-automation-api-for-revit.html

Revit アドインの用意

- Revit API
  - C#
- .NET Framework
  - Revit 2021 -> .NET Framework 4.8
  - Revit 2020 -> .NET Framework 4.7
  - Revit 2019 -> .NET Framework 4.7
  - Revit 2018 -> .NET Framework 4.6
- Microsoft Visual Studio
  - 2017 or 2019
- 参照ライブラリ
  - RevitAPI.dll
  - DesignAutomationBridge.dll

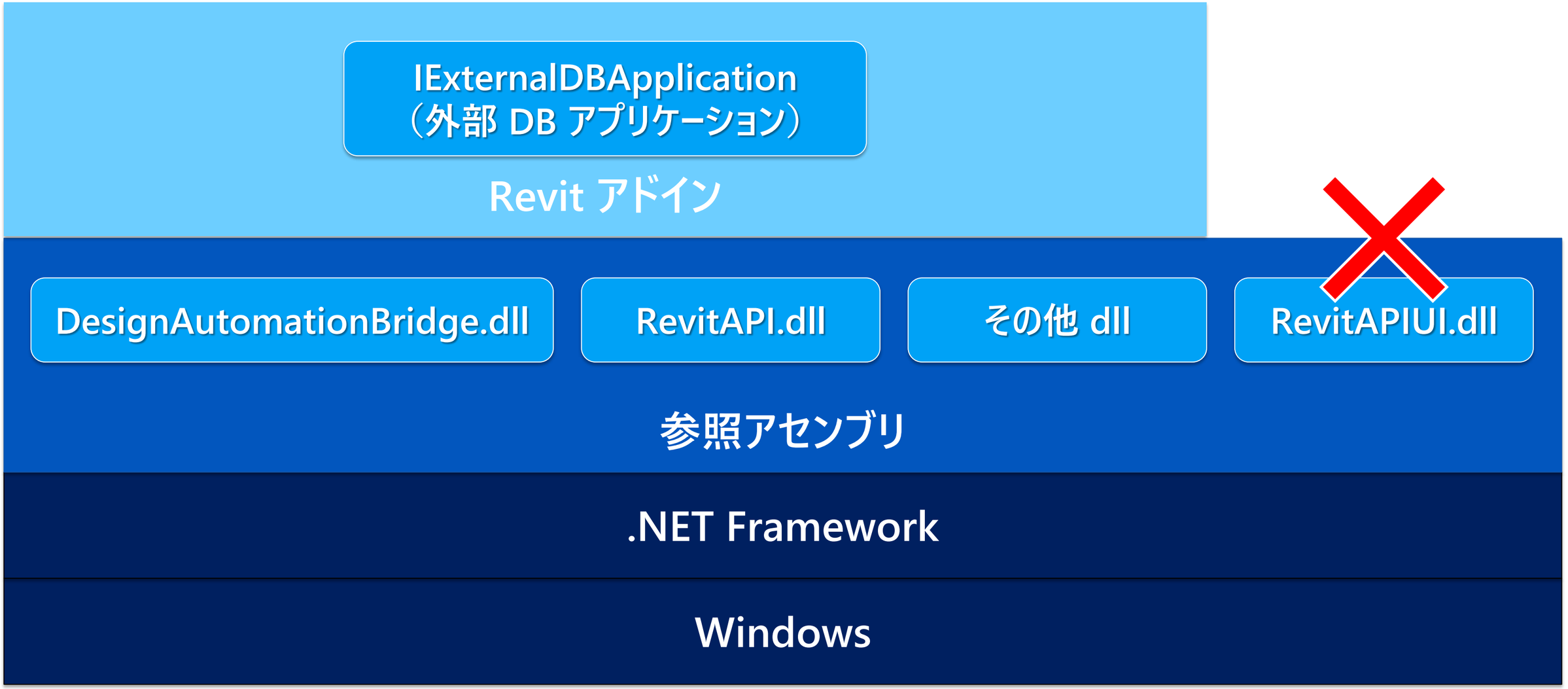


UI に関連する参照の削除

- Design Automation では、ユーザーインタラクションはサポートされておりません。
- Autodesk.Revit.UI 名前空間のクラスやメソッド、プロパティにアクセスしないようにコードを修正してください。
- WPF や Windows.Forms 等の UI ベースのライブラリも参照から外してください。
- 最低限必要な名前空間は以下の通りです。

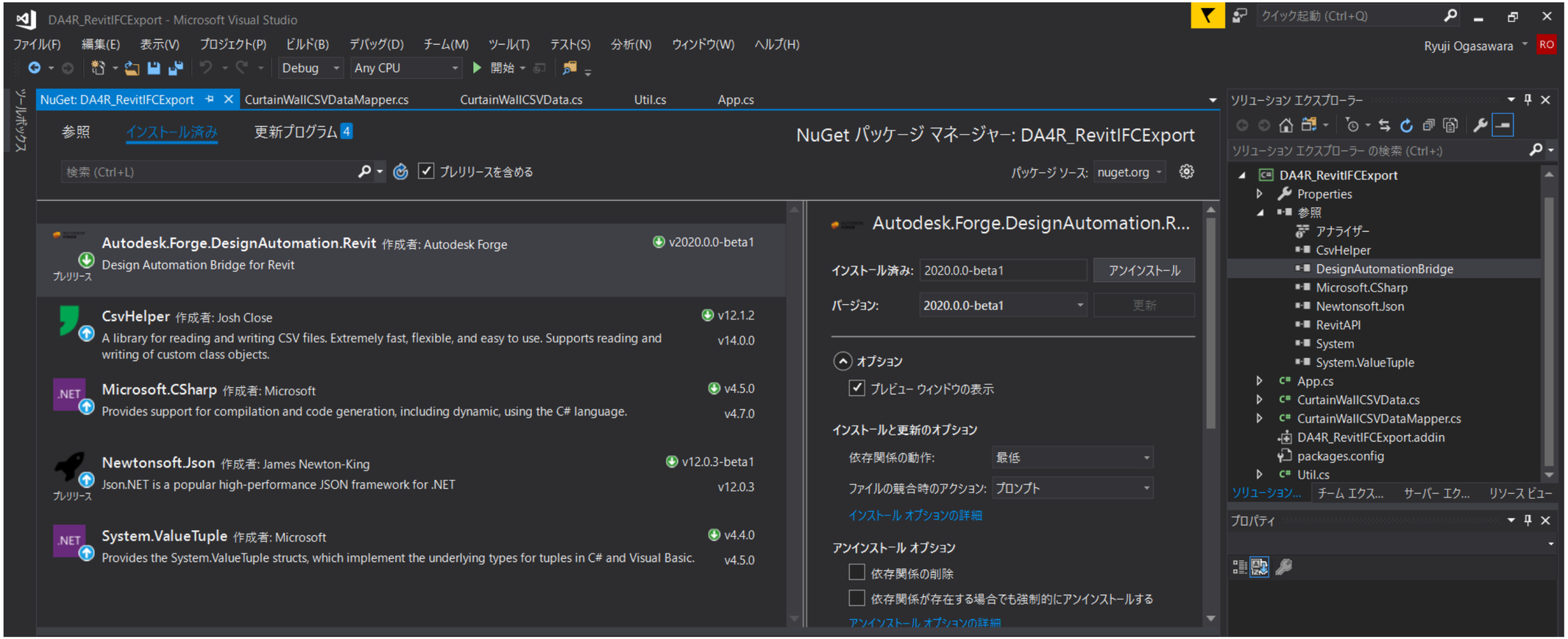
```
using Autodesk.Revit.ApplicationServices;
using Autodesk.Revit.DB;
using DesignAutomationFramework;
```

Revit アドインの変換に必要なアセンブリ参照



DesignAutomationBridge.dll のインストール

- NuGet パッケージ マネージャーで下記のパッケージをインストール
  - Autodesk.Forge.DesignAutomation.Revit



IExternalDBApplication の実装

```
using System;
using System.IO;

using Autodesk.Revit.ApplicationServices;
using Autodesk.Revit.DB;
using DesignAutomationFramework;

namespace DeleteWalls
{
    [Autodesk.Revit.Attributes.Regeneration(Autodesk.Revit.Attributes.RegenerationOption.Manual)]
    [Autodesk.Revit.Attributes.Transaction(Autodesk.Revit.Attributes.TransactionMode.Manual)]
    public class DeleteWallsApp : IExternalDBApplication
    {
        public ExternalDBApplicationResult OnStartup(Autodesk.Revit.ApplicationServices.ControlledApplication app)
        {
            DesignAutomationBridge.DesignAutomationReadyEvent += HandleDesignAutomationReadyEvent;
            return ExternalDBApplicationResult.Succeeded;
        }

        public ExternalDBApplicationResult OnShutdown(Autodesk.Revit.ApplicationServices.ControlledApplication app)
        {
            return ExternalDBApplicationResult.Succeeded;
        }

        public void HandleDesignAutomationReadyEvent(object sender, DesignAutomationReadyEventArgs e)
        {
            e.Succeeded = true;
            DeleteAllWalls(e.DesignAutomationData);
        }
```

イベントハンドラの登録

カスタム処理を呼び出し

Forge アプリからのパラメータを Revit アドインで受け取る方法

- JSON ファイルとしてパラメータを取得できます。

```
33
        1 個の参照
34      public void ExportIFC(DesignAutomationData data)
35      {
36          Document doc = data.RevitDoc;
37          Application app = data.RevitApp;
38
39          InputParams inputParameters = JsonConvert.DeserializeObject<InputParams>(File.ReadAllText("params.json"));
40
41          using (Transaction tx = new Transaction(doc))
42          {
43              tx.Start("Transaction Export IFC and CSV");
44
45              Console.WriteLine("カーテンウォールタイプ UniqueID: " + inputParameters.CurtainWallUniqueId);
46
47              // カーテンウォールのタイプを取得
48              Element curtainWallType = doc.GetElement(inputParameters.CurtainWallUniqueId);
49
50              // 壁要素を取得
51              FilteredElementCollector collector1 = new FilteredElementCollector(doc)
52                  .WhereElementIsNotElementType()
53                  .OfCategory(BuiltInCategory.OST_Walls)
54                  .OfClass(typeof(Wall));
55
56              // カーテンウォールのインスタンスを取得
57              List<Element> curtainWallInstances = new List<Element>();
```

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。

```
33
        1 個の参照
34      public void ExportIFC(DesignAutomationData data)
35      {
36          Document doc = data.RevitDoc;
37          Application app = data.RevitApp;
38
39          InputParams inputParameters = JsonConvert.DeserializeObject<InputParams>(File.ReadAllText("params.json"));
40
41          using (Transaction tx = new Transaction(doc))
42          {
43              tx.Start("Transaction Export IFC and CSV");
44
45              Console.WriteLine("カーテンウォールタイプ UniqueID: " + inputParameters.CurtainWallUniqueId);
46
47              // カーテンウォールのタイプを取得
48              Element curtainWallType = doc.GetElement(inputParameters.CurtainWallUniqueId);
49
50              // 壁要素を取得
51              FilteredElementCollector collector1 = new FilteredElementCollector(doc)
52                  .WhereElementIsNotElementType()
53                  .OfCategory(BuiltInCategory.OST_Walls)
54                  .OfClass(typeof(Wall));
55
56              // カーテンウォールのインスタンスを取得
57              List<Element> curtainWallInstances = new List<Element>();
```

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。

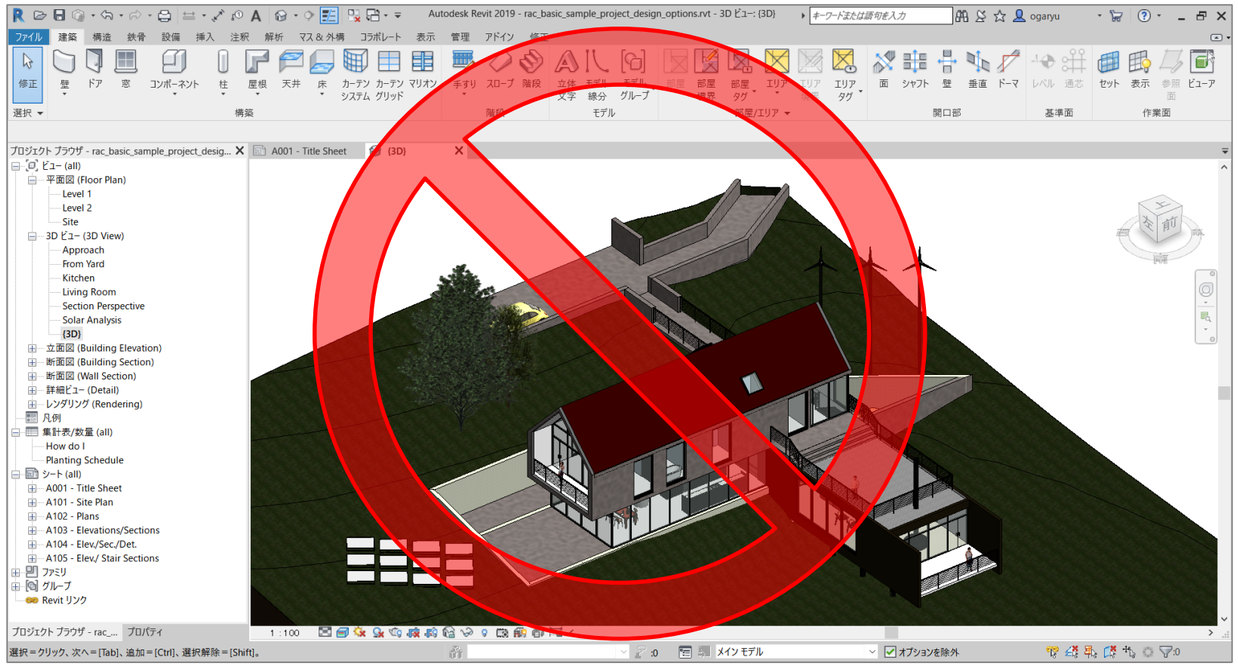
```
public void HandleDesignAutomationReadyEvent(object sender, DesignAutomationReadyEventArgs e)
{
        // Hook up the CustomFailureHandling failure processor.
        Application.RegisterFailuresProcessor(new CustomFailureHandlingProcessor());

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

        e.Succeeded = true;
}
```

Revit 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、アセンブリ参照はできません。
- [x] REST API で直接 Revit のデータに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] アドインからネットワークにアクセスすることはできません。
- [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。



RevitAPIUI.dll

アドインマニフェストの修正

- <AddIn Type=“Command”>を<AddIn Type=“DBApplication”>に変更

```
1  <?xml version="1.0" encoding="utf-8"?>
2  <RevitAddIns>
3    <AddIn Type="DBApplication">
4      <Name>SketchIt</Name>
5      <Assembly>.¥SketchIt.dll</Assembly>
6      <AddInId>B69A15D3-48AA-4792-8502-CB48154EE0F9</AddInId>
7      <FullClassName>SketchIt.SketchItApp</FullClassName>
8      <Description>"SketchIt"</Description>
9      <VendorId>Autodesk</VendorId>
10     <VendorDescription>
11     </VendorDescription>
12   </AddIn>
13 </RevitAddIns>
```

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 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
```

```xml
<?xml version="1.0" encoding="utf-8" ?>
<ApplicationPackage>
  <Components Description="SketchIt">
    <RuntimeRequirements OS="Win64"
                         Platform="Revit"
                         SeriesMin="R2018"
                         SeriesMax="R2018" />
    <ComponentEntry AppName="SketchIt"
                    Version="1.0.0"
                    ModuleName="./Contents/SketchIt.addin"
                    AppDescription="Sketches some walls and a floor."
                    LoadOnCommandInvocation="False"
                    LoadOnRevitStartup="True" />
  </Components>
</ApplicationPackage>
```

Forge Online

Revit タスクの自動化：サンプルアプリによるワークフローの解説

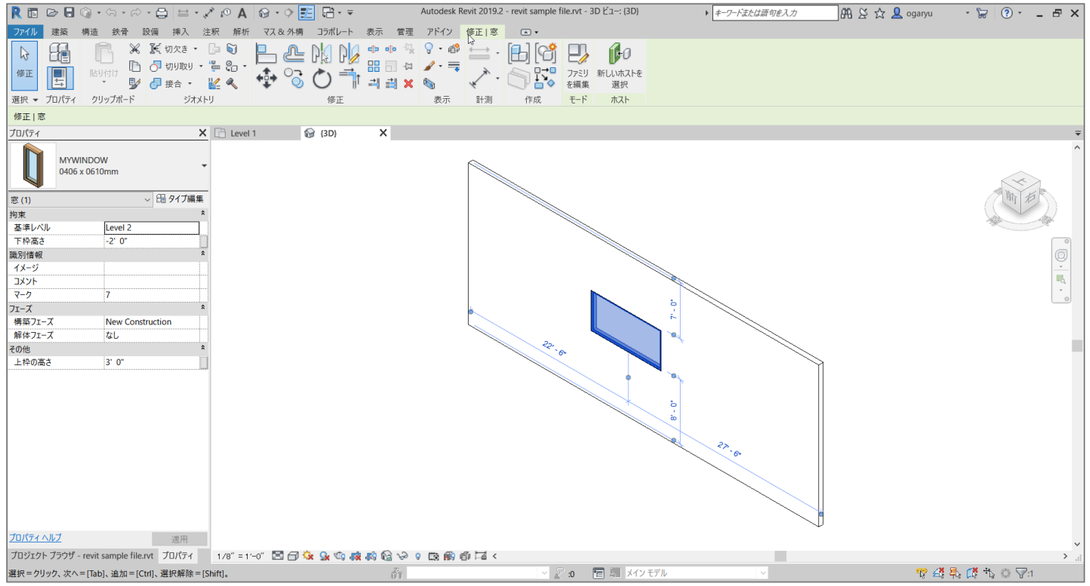
小笠原 龍司
オートデスク 株式会社

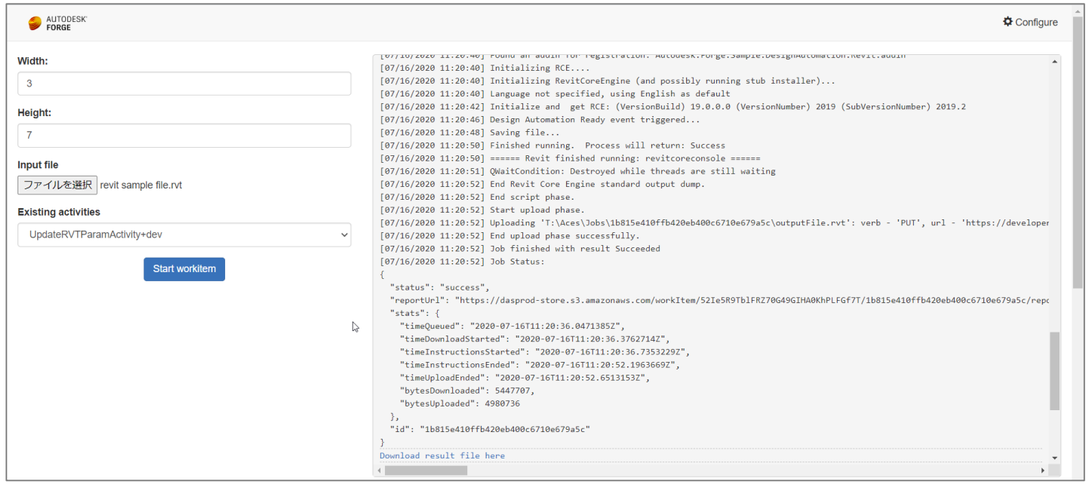
Learn Autodesk Forge

- Forge API を学べるチュートリアル
  Learn Autodesk Forge（英語）

- **Modify your models セクション**
  （Design Automation API）
  - .NET Core / Node.js を利用して Forge アプリを開発する手順と、 AutoCAD、Inventor、Revit、3ds Max の各種サンプル

- **Revit サンプル**
  - Revit プロジェクトの窓ファミリタイプの幅と高さを変更して、OSS の Bucket に保存します。





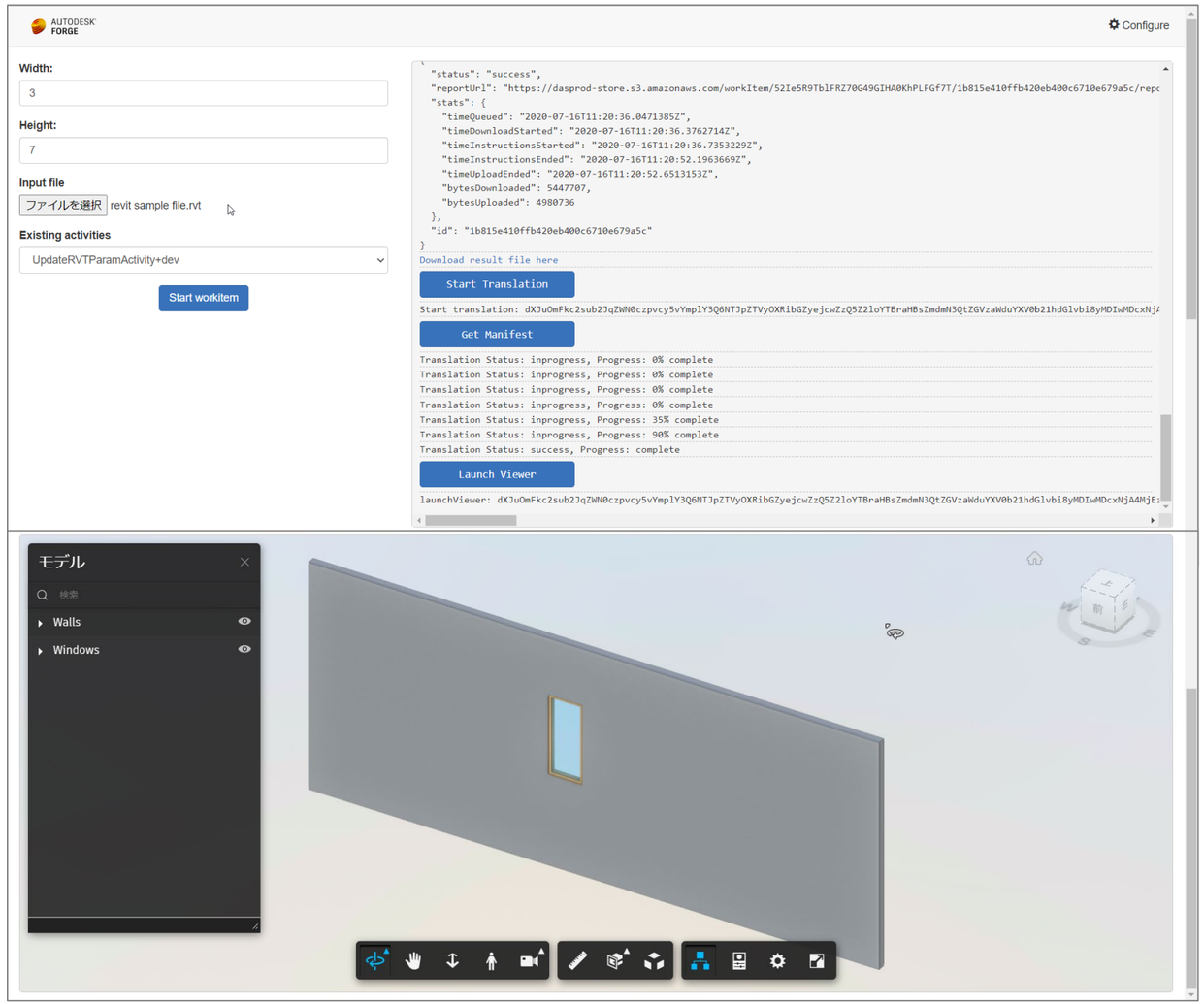
Forge SDK の活用

- Forge の REST API を、各開発言語の環境で利用しやすいようにラップしたパッケージ
- Learn Autodesk Forge のサンプルでも使用
- https://github.com/Autodesk-Forge

Node.js	forge-api-nodejs-client	NPM パッケージ
.NET	forge-api-dotnet-client	NuGet パッケージ
Java	forge-api-java-client	Apache Maven ツール
PHP	forge-php-client	Composer ツール

Revit サンプルの拡張

1. Revit アドインのバンドル Pkg の作成
2. AppBundle と Activity の登録
3. Revit ファイルを OSS Bucket にアップロード
4. クライアント画面で寸法のパラメータを入力
5. WorkItem の登録
6. 処理結果のファイルをダウンロード
7. **Model Derivative API で SVF 変換**
8. **Forge Viewer で表示**



- Design Automation API for Revit - Learn Autodesk Forge チュートリアル .NET Core Sample のセットアップ
- Design Automation API for Revit - Learn Autodesk Forge チュートリアル .NET Core Sample で動作確認
- Design Automation API for Revit - Learn Autodesk Forge チュートリアル .NET Core Sample の処理結果を Viewer で表示

Revit アドインの実装

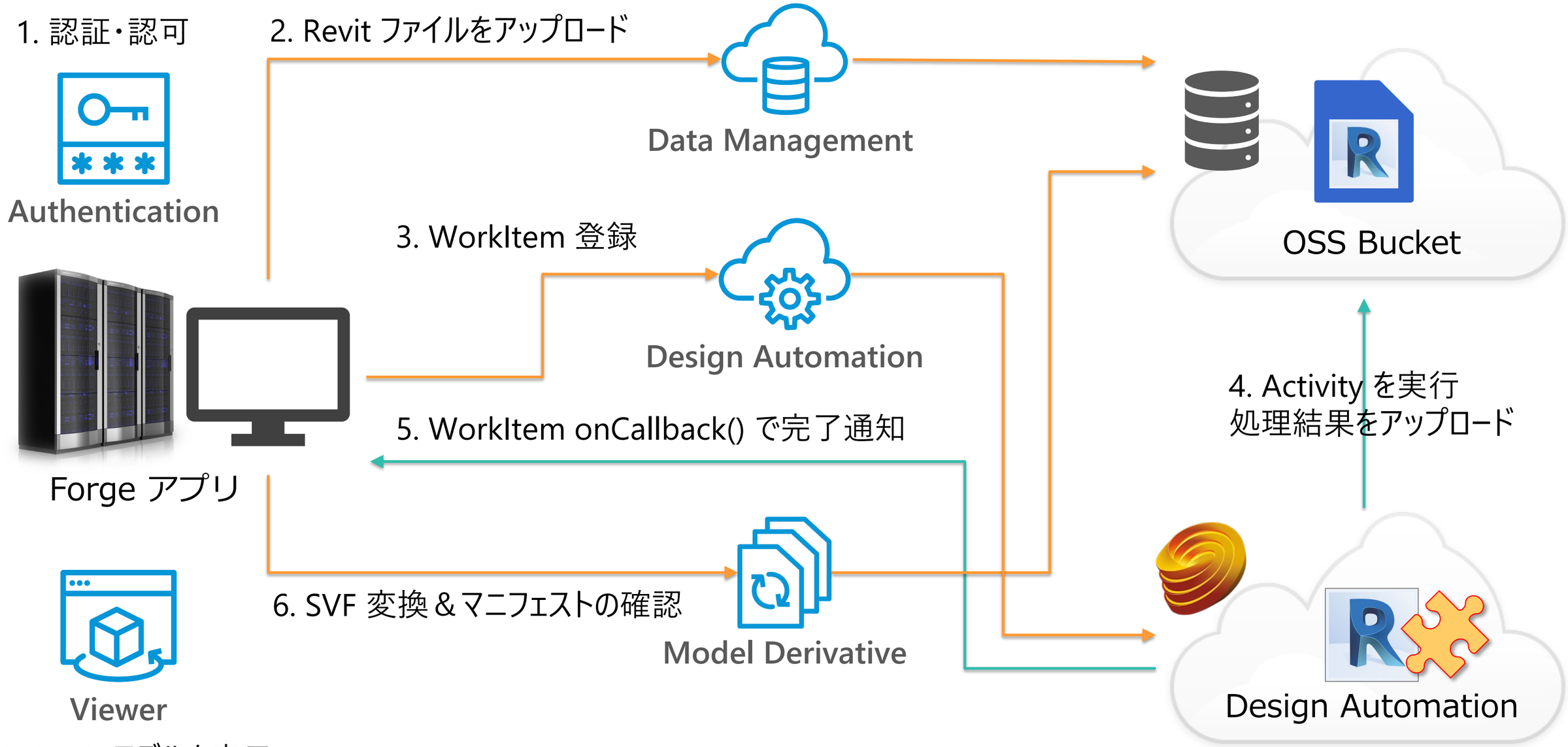
```
30
        1 個の参照 | 変更箇所なし | 作成者なし、変更箇所なし
31      private void EditWindowParametersMethod(Document doc)
32      {
33          InputParams inputParameters = JsonConvert.DeserializeObject<InputParams>(File.ReadAllText("params.json"));
34
35          //Modifying the window parameters
36          //Open transaction
37          using (Transaction trans = new Transaction(doc))
38          {
39              trans.Start("Update window parameters");
40
41              //Filter for windows
42              FilteredElementCollector WindowCollector = new FilteredElementCollector(doc)
43                                                          .OfCategory(BuiltInCategory.OST_Windows)
44                                                          .WhereElementIsNotElementType();
45
46              IList<ElementId> windowIds = WindowCollector.ToElementIds() as IList<ElementId>;
47
48              foreach (ElementId windowId in windowIds)
49              {
50                  Element Window = doc.GetElement(windowId);
51                  FamilyInstance FamInst = Window as FamilyInstance;
52                  FamilySymbol FamSym = FamInst.Symbol;
53                  SetElementParameter(FamSym, BuiltInParameter.WINDOW_HEIGHT, inputParameters.Height);
54                  SetElementParameter(FamSym, BuiltInParameter.WINDOW_WIDTH, inputParameters.Width);
55              }
56
57              //To save all the changes commit the transaction
58              trans.Commit();
59          }
```

プロジェクトの保存

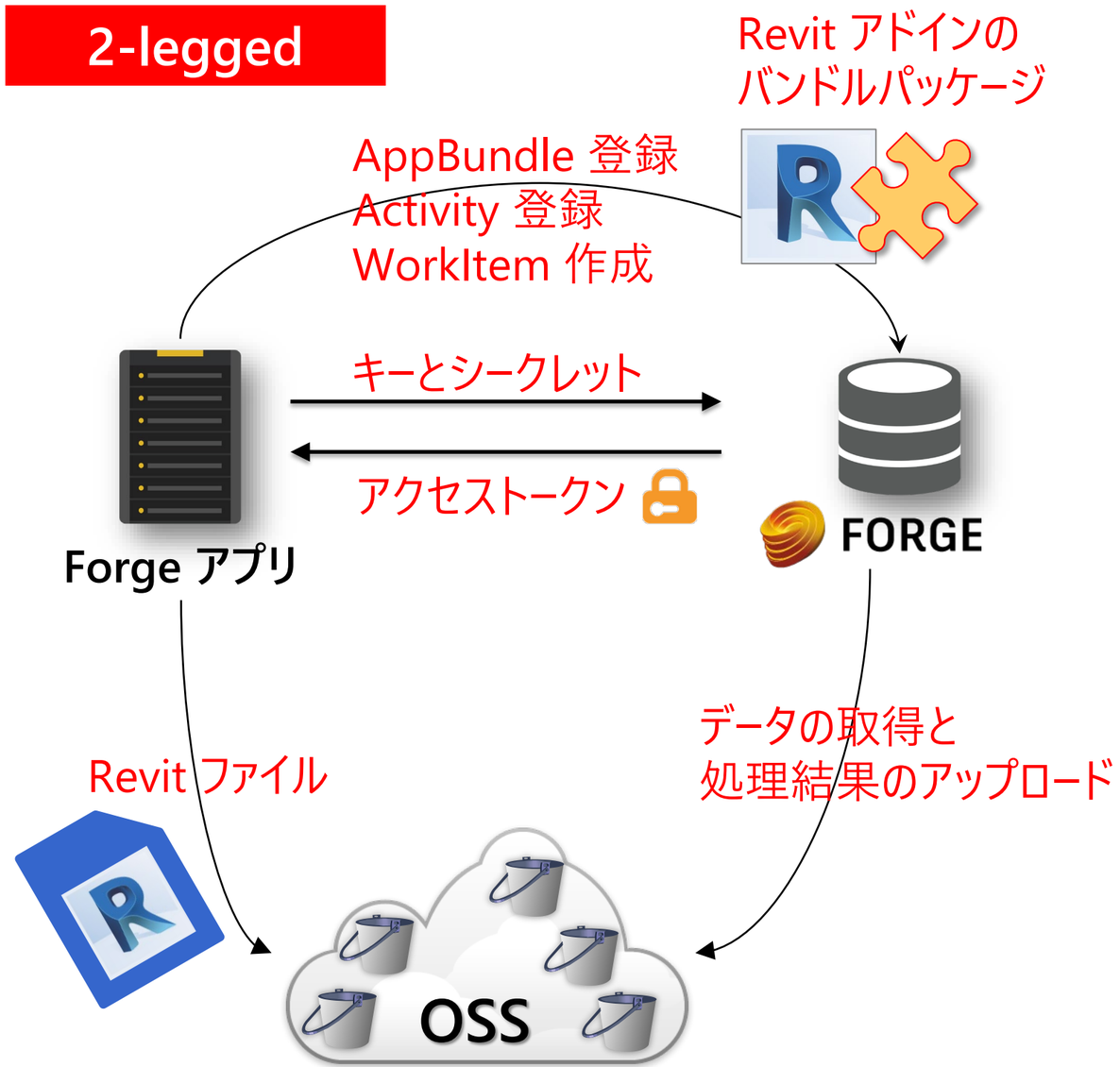
- Document.SaveAs() メソッドでワーキングディレクトリ上に保存されます。

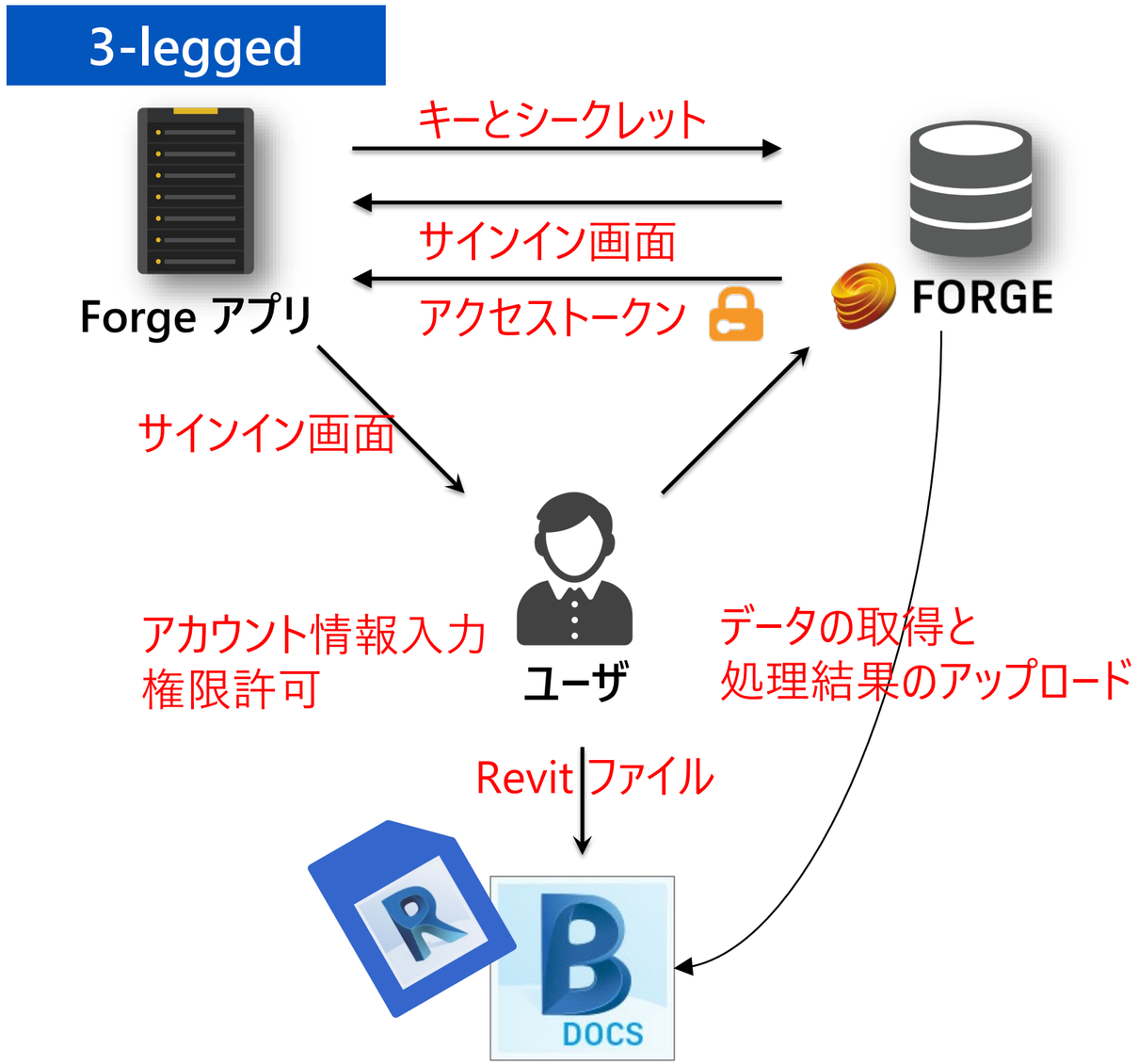
```
60
61              //Path of the project(i.e)project where your Window family files are present
62              string OUTPUT_FILE = "OutputFile.rvt";
63
64              //Save the updated file by overwriting the existing file
65              ModelPath ProjectModelPath = ModelPathUtils.ConvertUserVisiblePathToModelPath(OUTPUT_FILE);
66              SaveAsOptions SAO = new SaveAsOptions();
67              SAO.OverwriteExistingFile = true;
68
69              //Save the project file with updated window's parameters
70              LogTrace("Saving file...");
71              doc.SaveAs(ProjectModelPath, SAO);
72          }
73
```

Revit サンプル拡張版のワークフロー



アクセストークンの取得





Design Automation API に必要なスコープ

- Design Automation API の場合、コードの生成と実行の権限に相当する、code:all を指定します。

user-profile:read	プロファイル（Autodesk ID）の表示
user:read	プロファイル（Autodesk ID）の読み取り
user:write	プロファイル（Autodesk ID）の書き込み
viewables:read	変換後のデザインデータ（SVF）の読み取り（表示）
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
bucket:create	新しい Bucket の作成
bucket:read	Bucket の読み取り
bucket:update	Bucket の更新
bucket:delete	Bucket の削除
code:all	コードの生成または実行（Design Automation API）
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

https://adndevblog.typepad.com/technology_perspective/2018/11/about-access-token.html

Data Management API に必要なスコープ

- "bucket:read bucket:update bucket:create bucket:delete data:write data:create data:read"

Scope

user-profile:read	プロファイル（Autodesk ID）の表示
user:read	プロファイル（Autodesk ID）の読み取り
user:write	プロファイル（Autodesk ID）の書き込み
viewables:read	変換後のデザインデータ（SVF）の読み取り（表示）
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
bucket:create	新しい Bucket の作成
bucket:read	Bucket の読み取り
bucket:update	Bucket の更新
bucket:delete	Bucket の削除
code:all	コードの生成または実行（Design Automation API）
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

https://adndevblog.typepad.com/technology_perspective/2018/11/about-access-token.html

AppBundle, Activity, WorkItem の相互関係

AppBundle

Revit アドインのバイナリパッケージ

Id : UpdateRVTParam

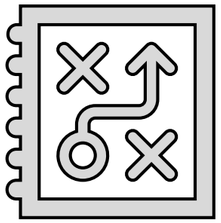
Engine : Autodesk.Revit+2021

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: UpdateRVTParamActivity

Input Parameter : RVT, JSON

Output Parameter: RVT

AppBundle: UpdateRVTParam

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 : UpdateRVTParamActivity

Input Parameter : File URL, JSON

Output Parameter: 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。

https://adndevblog.typepad.com/technology_perspective/2019/02/understanding-steps-to-use-design-automation-api-for-revit.html

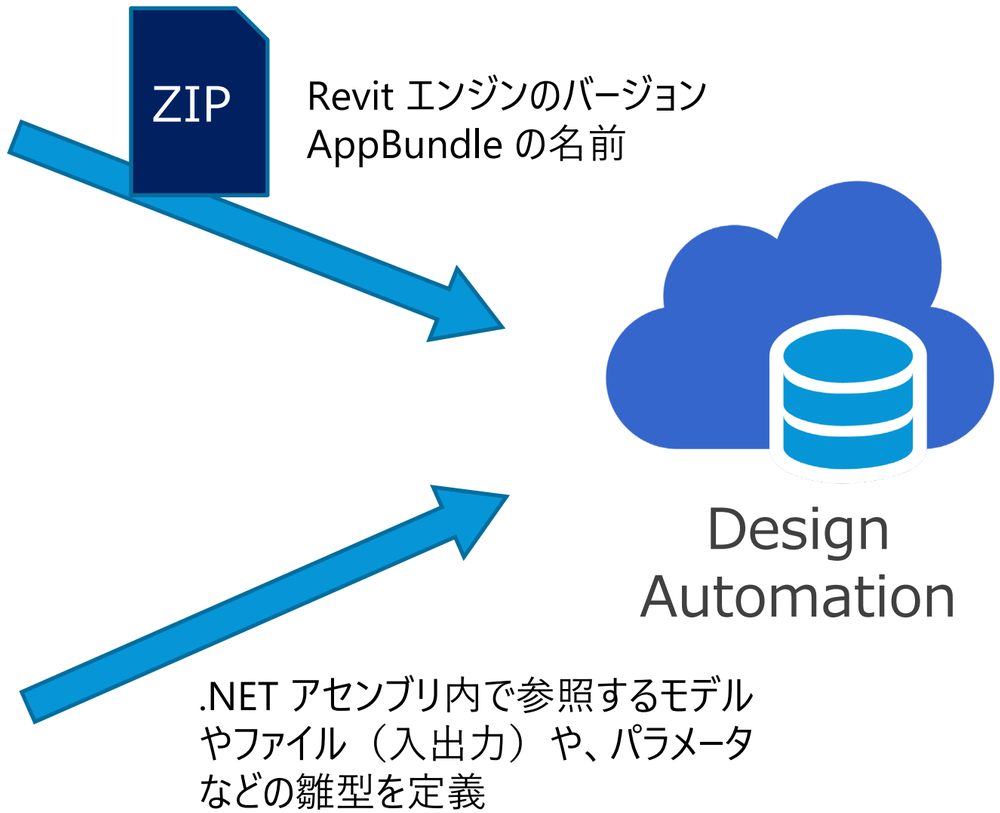
AppBundle と Activity の登録

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



https://adndevblog.typepad.com/technology_perspective/2019/02/understanding-steps-to-use-design-automation-api-for-revit.html

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.UpdateRVTParam+dev

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

•形式: YourNickname.SomeActivityId+SomeAliasName
•例: MyFirstForgeAppNickname.UpdateRVTParamActivity+dev

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だす、といった開発・テスト用の使い方ができます。

https://forge.autodesk.com/en/docs/design-automation/v3/developers_guide/aliases-and-ids/
https://adndevblog.typepad.com/technology_perspective/2019/02/understanding-steps-to-use-design-automation-api-for-revit.html

Activity の定義

```
{
    "id": "UpdateRVTParamActivity",
    "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": "Autodesk.Revit+2021",
    "appbundles": [ "MyFirstForgeAppNickname.UpdateRVTParam+dev" ],
    "description": "."
}
```

Activity ID

入力 Revit ファイル

入力 JSON ファイル

出力 Revit ファイル

Engine

AppBundle ID

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

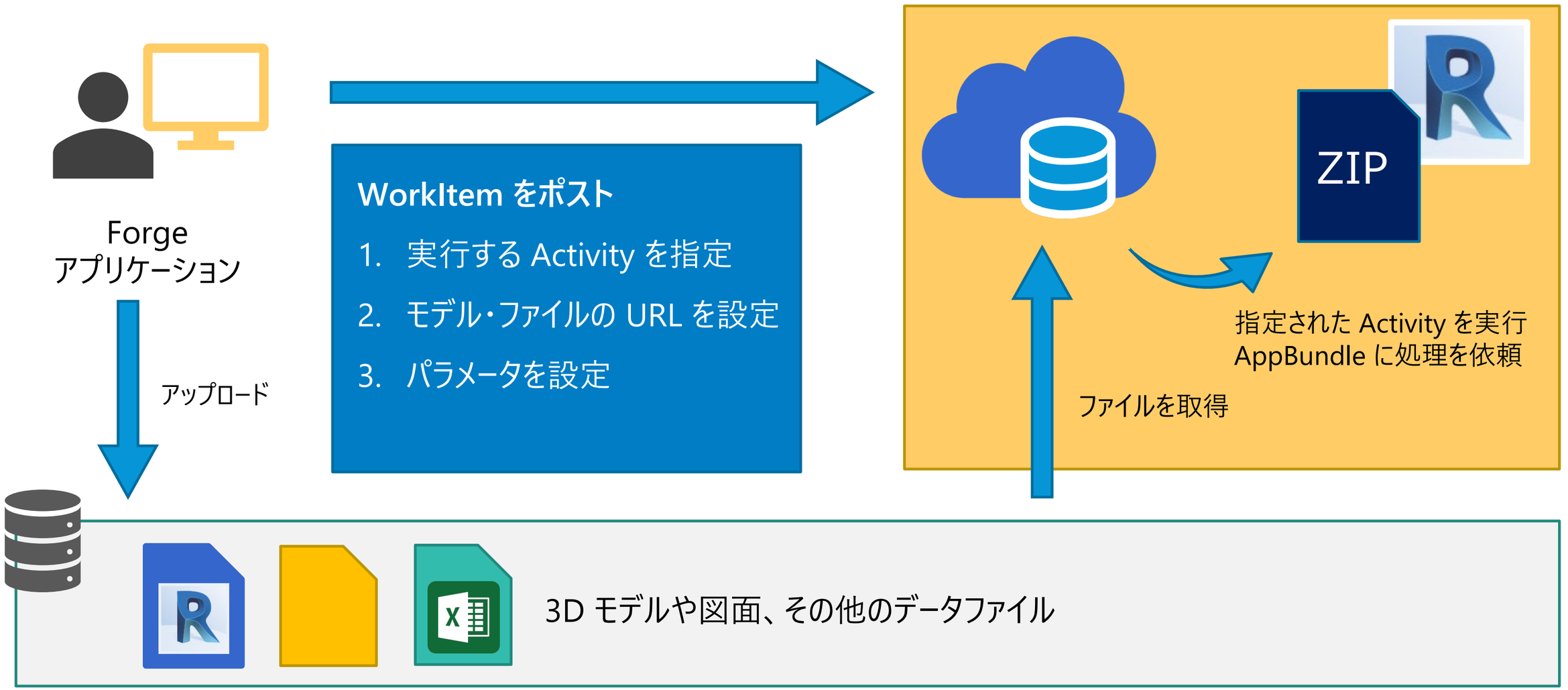
ondemand WorkItem の処理中に任意のタイミングで外部データファイルを取得する方法

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。
  - /l 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。

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

WorkItem を登録する



https://adndevblog.typepad.com/technology_perspective/2019/02/understanding-steps-to-use-design-automation-api-for-revit.html

Forge アプリから Revit アドインにデータを渡す方法

Activity

```
"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": {
```

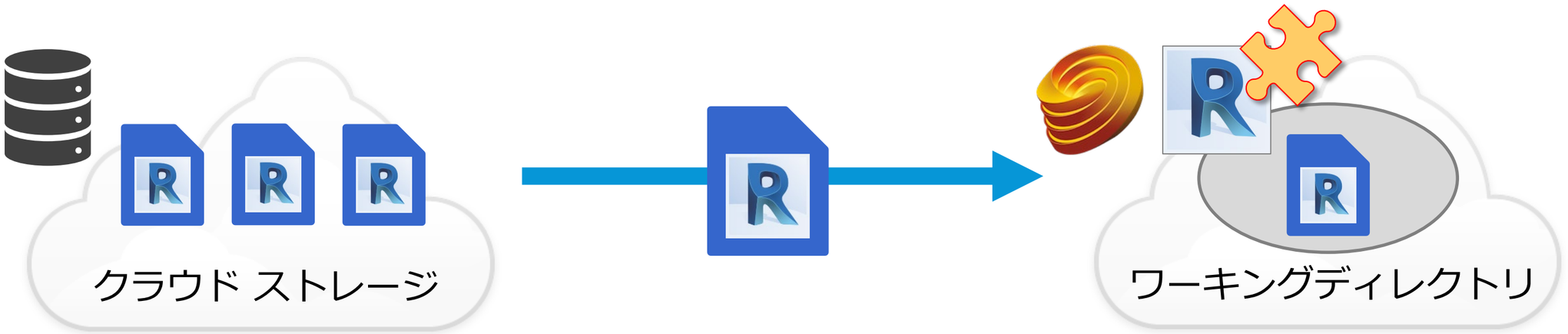
WorkItem

```
"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"
   }
 }
```

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。

Design Automation 実行時のデータの取り扱い

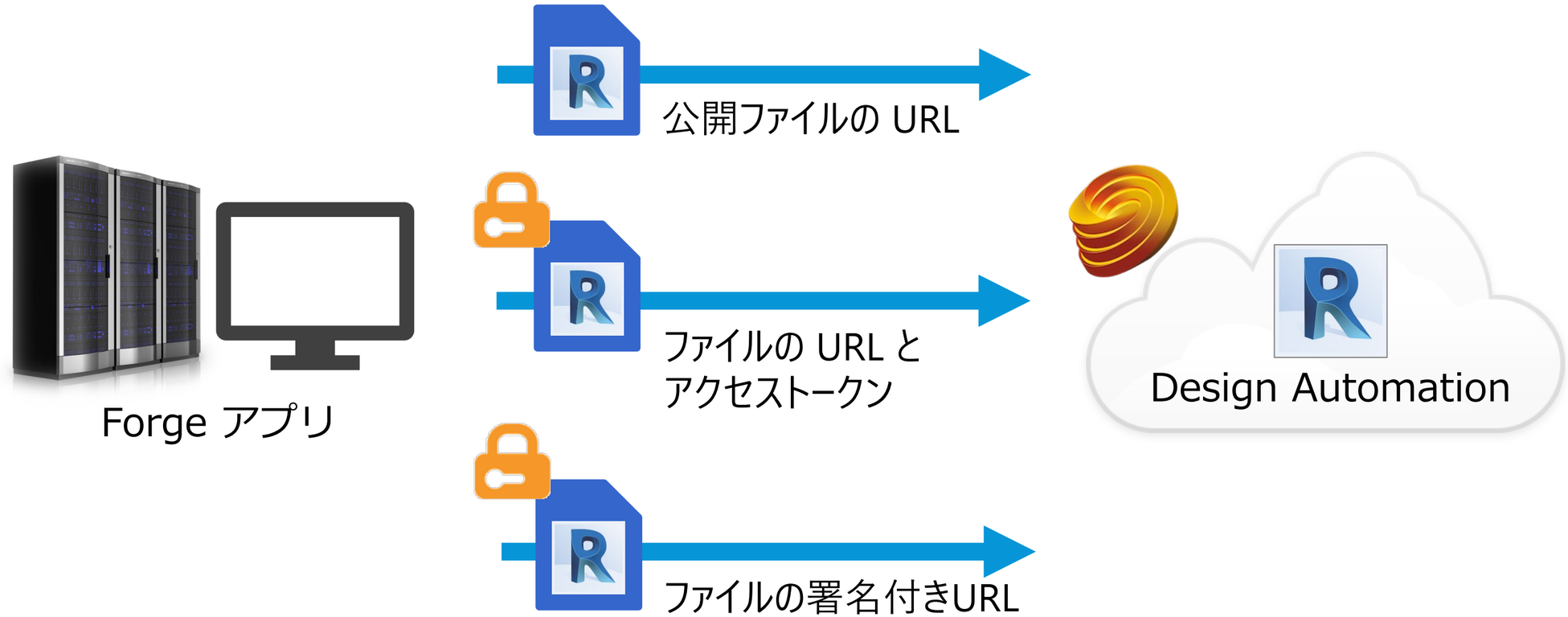
- Design Automation は、クライアント/Forge アプリが指定した任意のクラウドストレージのデータにアクセスして、Forge の一時的なデータ領域（ワーキングディレクトリ）にダウンロードします。



- Forge の一時的なデータ領域のデータは、Design Automation の実行後に破棄されます。

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 Design Automation 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
  - 署名付き URL （URLの有効期限、One Time URL、読み書きアクセス権 など）



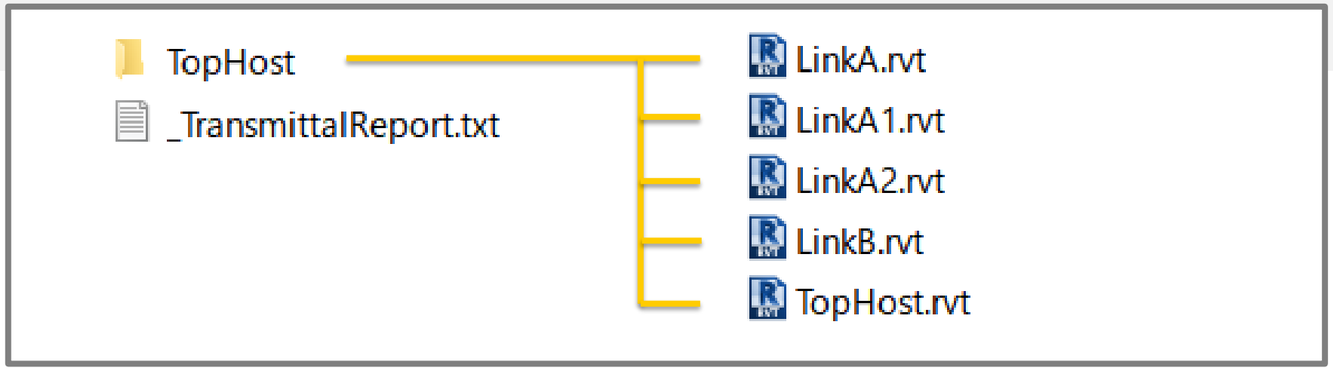
署名付き URL の取得

- Design Automation API では、処理結果の保存場所を事前に確保して、その URL を取得します。
- 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

サポートされているファイル: eTransmit ファイル

- 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"
  }
}
```



サポートされているファイル: Revit リンクモデル

```
"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

サポートされているファイル: ZIP ファイル

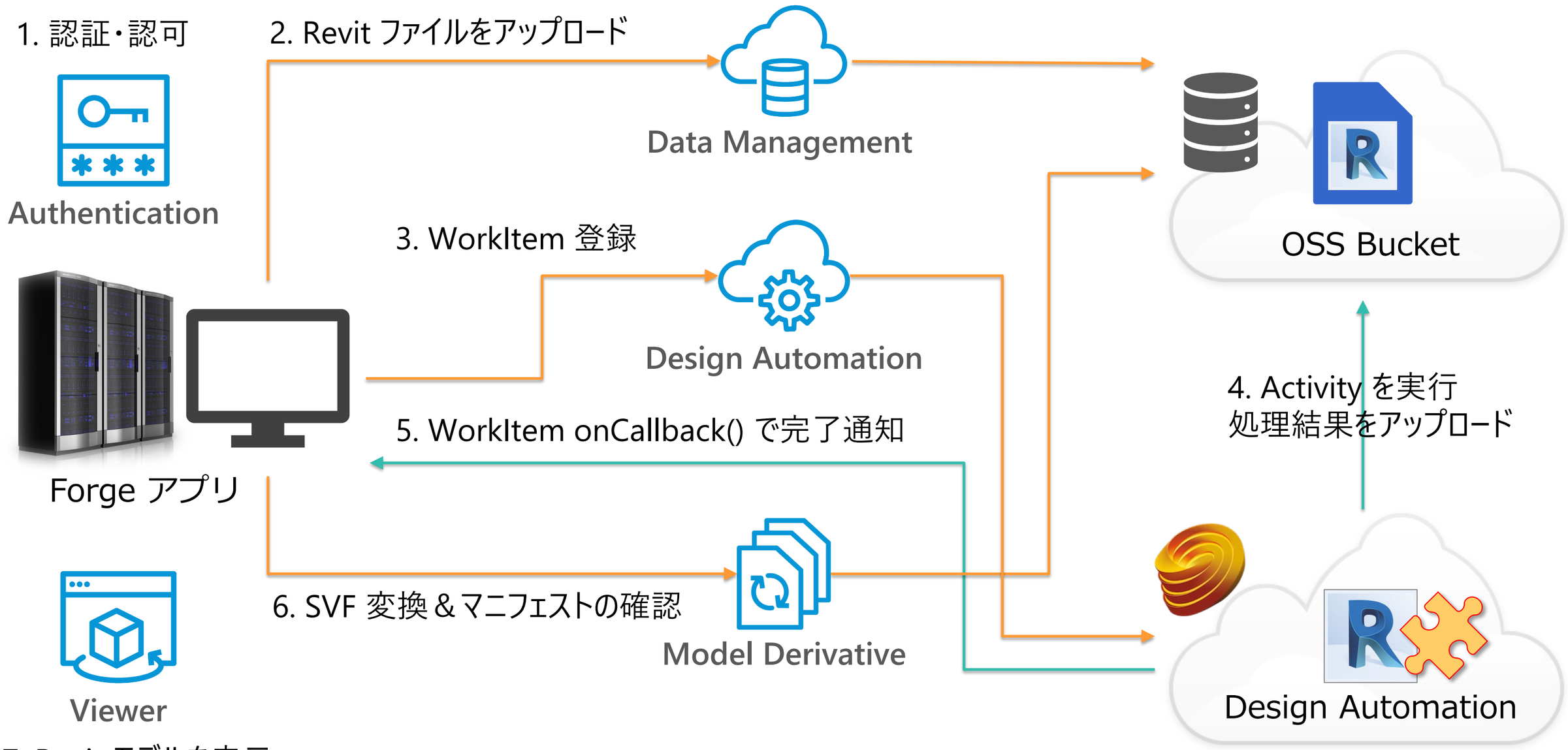
- アップロード速度を向上させるために、Revit モデルを ZIP 圧縮して送信することができます。
- “pathInZip” オプションで、メインの Revit モデルのパスを指定します。

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

WorkItem のコールバック通知処理

- 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。

Revit サンプル拡張版のワークフロー



AUTODESK®

Make anything.