

AULON481

Revit on Forge: Learn How to Run Your Revit Add-ins in the Cloud

Mikako Harada, Ph.D.
Autodesk

Learning Objectives

- Understand what Design Automation for Revit is
- Get ideas about what you can do with Design Automation for Revit
- Learn the basics of Design Automation for Revit and where you can get started
- Learn related Forge technologies to store and view a Revit model

Description

A long-awaited design automation for Revit (“DA for Revit” for short) has finally arrived! Currently, DA for Revit is available as public beta. Using DA for Revit, you can create, edit and extract Revit models through the Revit API without having the Revit desktop environment. For example, you can create a Revit family automatically by passing parameters. You can check a model against certain design criteria and create a report, and if needed, automatically modify the model. It is also possible to extract model data which was not possible with Forge Model Derivative API. In this session, we will introduce you to DA for Revit to those who have experience writing Revit add-ins and are interested in learning Forge. You will learn how to convert your existing Revit add-ins to run in the cloud, and get started with leveraging Forge services, for example, integrating with viewer and BIM 360.

Disclaimer: As of this writing, Design Automation for Revit is in beta. We are continuously updating and improving the functionality and documentations online. Please refer to the latest documentation at: https://forge.autodesk.com/en/docs/design-automation/v3/developers_guide/overview/

Speakers

Mikako Harada works as an AEC manager for Developer Technical Services team at Autodesk. She provides API (Application Programming Interface) technical support for AEC products. Prior to joining Autodesk, she worked as a researcher for the Swiss Federal Institute of Technology (ETH) in Zurich. While at ETH, she worked with projects involving the development of web-based collaborative environment with the Swiss building industry and web-based visualization projects for business data archive systems. She was also a researcher at Engineering Design Research Center in Carnegie Mellon University (CMU), and Artificial Intelligence Cognitive Systems group at General Motors Technical Center. Her interest is in the areas of interactive techniques, optimization and layout synthesis. More at her [blog](#).

Prerequisites

This class assumes you are familiar with Revit add-in development and have the basic knowledge of Web services and REST API. Experience with existing Forge API, such as Document Management API and Viewer will be a plus to understand the discussion in detail. The class covers a wide range of topics within the limited time! The intention is to give a big picture and point to the learning materials so that you can jump start right after the talk.

Agenda

- Design Automation for Revit
- BIM 360 and API
- Forge Viewer

The focus of talk is about Design Automation for Revit. Majority of the time will be spent for Design Automation for Revit (“DA for Revit” for short). As we will explain shortly, DA for Revit itself does not store the data, and there is no UI (user interface). BIM 360 will be a good place to store data and Forge Viewer to view the model. We will talk about those quickly.

Design Automation for Revit: Big Picture

Design Automation for Revit is a Revit on cloud offered as a service. You can think of it as Revit running on a remote desktop without any UI. You can execute your add-in command just like desktop, but without any UI, i.e., with no user intervention. It’s a part of Autodesk Forge, a set of web services that Autodesk provide. Currently, it’s public beta. Anybody can use it without needing to have Revit installed on your local machine.

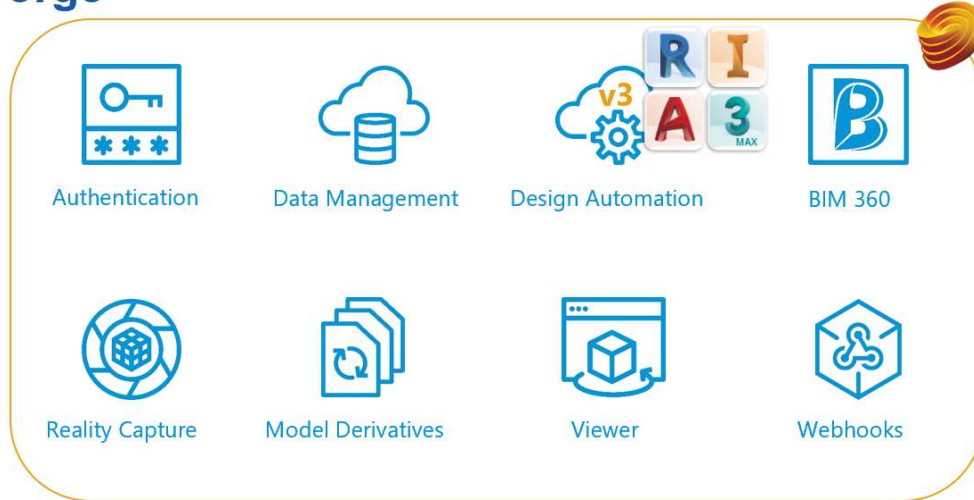
That said, for any serious development, you do need Revit installed on your desktop as after all, you will be running Revit add-ins in the cloud. If you want to learn how Design Automation works and get a feel of it by running code samples, it is possible to try out without having a desktop version of Revit.

Forge

Forge is a set of web services APIs (or Application Programming Interface) that Autodesk offers to external developers. Autodesk uses it internally to develop its own web services as well, such as BIM 360 and Fusion 360.

The picture below lists the types of currently available public Forge APIs:

Forge



Currently, following services are offered externally:

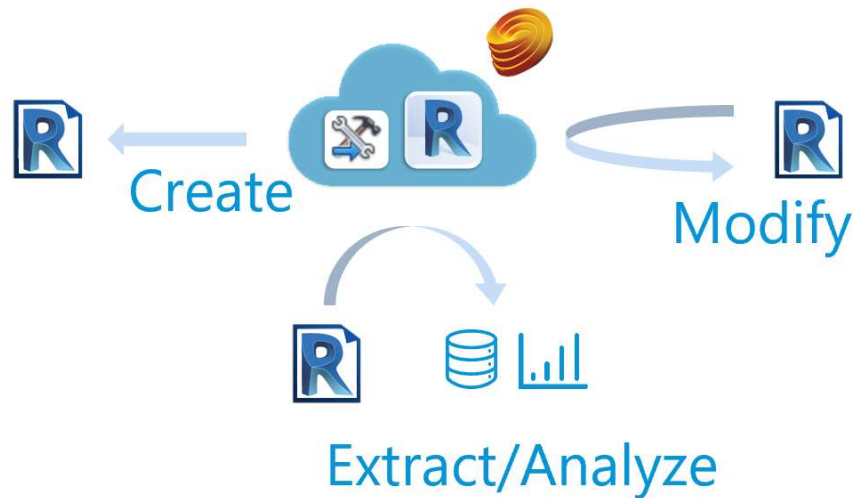
- Authentication
- Data Management
- Design Automation (v3: beta)
- BIM 360
- Reality Capture
- Model Derivatives
- Viewer
- Webhooks

We will come back to the more explanation of each later. Design Automation is a part of Forge offering. Currently the officially released version of Design Automation is version 2. It supports AutoCAD. With version 3, it supports Revit, Inventor, and 3ds Max in addition to AutoCAD.

What Can We Do?

Using Design Automation for Revit, you can create, modify, and analyze Revit models. For example, you can create a family file, modify a Revit model with company standard font, and check the “health” of a Revit model.

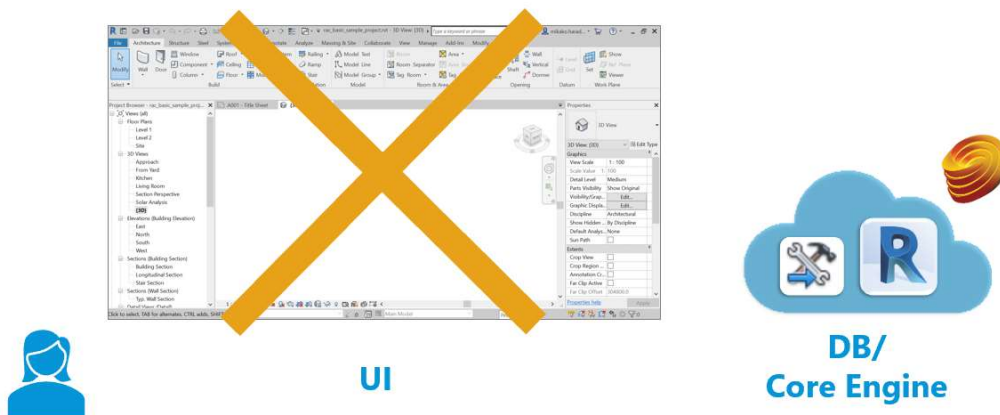
What Can We Do?



What is Not

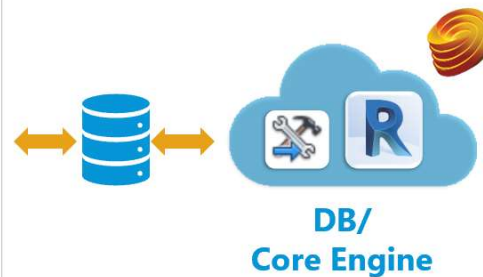
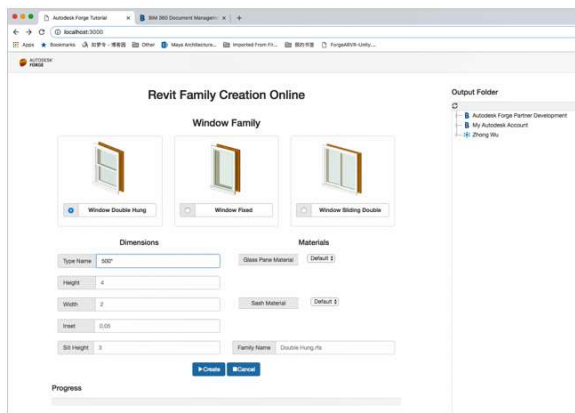
Design Automation for Revit does not include UI component. Your add-in will run in the cloud “automatically” without the user’s interaction. If you have a desktop version of Revit add-in which asks for the user input, then you will need to separate such a user interaction from the core DB (Database) module.

What is Not



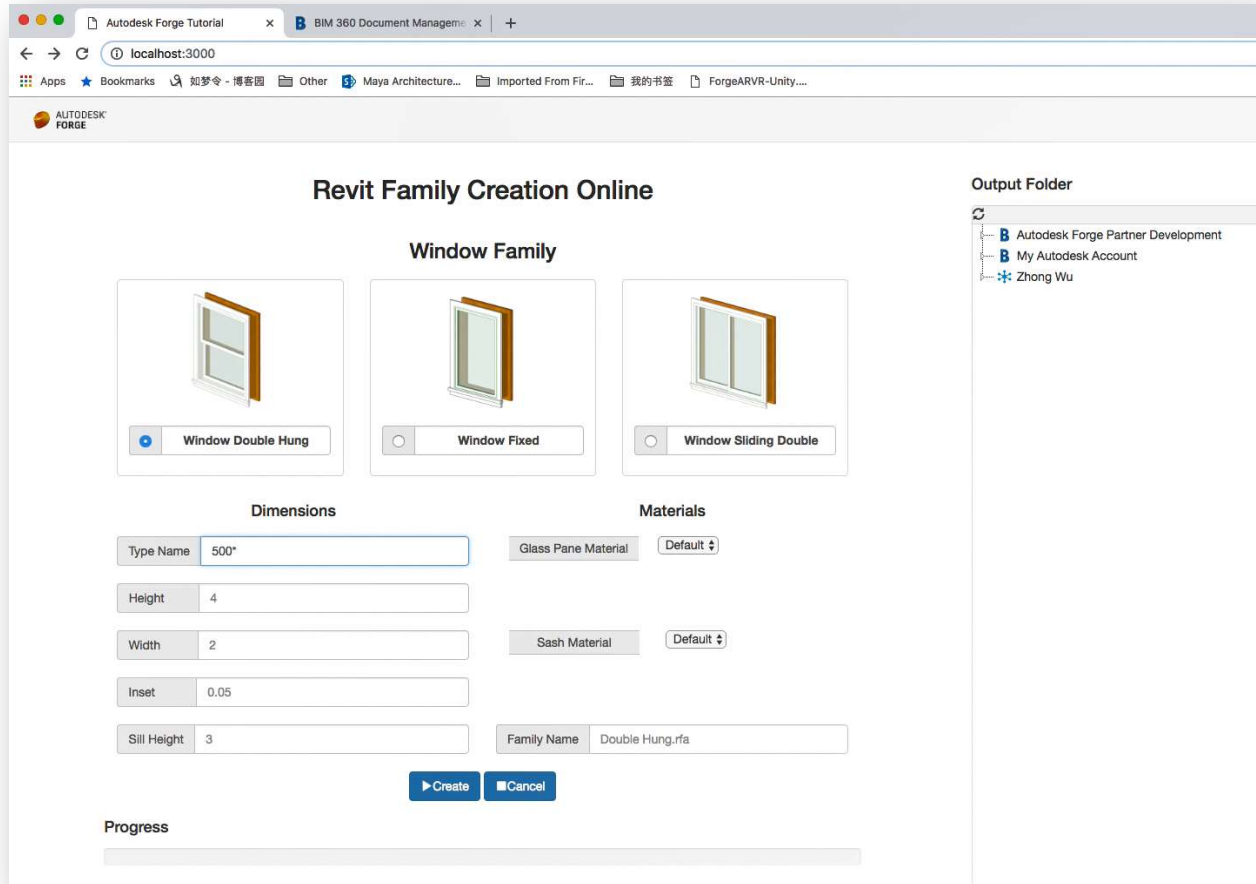
Instead, you will need to provide a separate UI before running UI-less add-in in Design Automation (DA). It could be, for example, a web page. You then pass necessary input parameters as json string or any other type of data to the Revit running as DA. DA itself does not store data. You will be providing information about the location of a Revit file in a cloud storage. DA will download the file from it and put it back if needed. The figure below depicts the flow and the relations among UI, Revit core engine, and the file storage location.

Design Automation for Revit



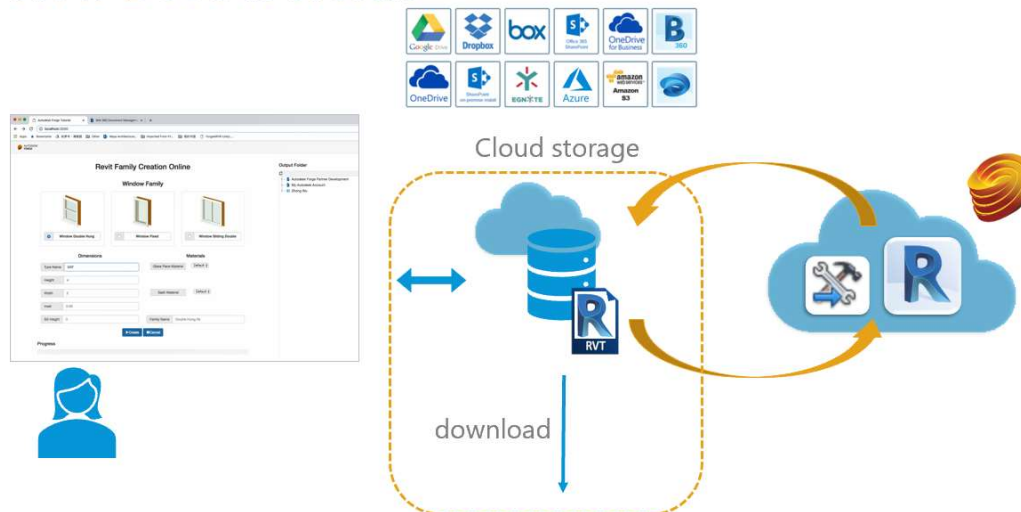
**Your own
frontend/backend**

Example: Family Creation



Let's take a look at an example of creating a window's family and see how it works. The above picture shows a sample image of the UI. In this example, the user is presented with a web page. The user chooses a style of a window and input parameters, such as type name, height, width and inset and sill height. The user also specifies the location to save a family file. Once the user clicks a "Create" button, the process of windows family creation starts in the background and result is saved to the storage.

How Does it Work?



Behind the scenes, DA is triggered after the user input all the necessary data. Once an add-in is executed in DA, DA automatically uploads to the storage location. The location of the storage can be anywhere. It could be Google Docs, Box, Egnyte, BIM 360 and any other cloud locations as long as it supports API to download and upload. If needed, the user can download to the local machines from the cloud storage; e.g., to open in a local Revit. DA itself will not download directly to a local machine.

Design Automation for Revit: How To

There are three phases when using DA for Revit:

1. Write a Revit add-ins/app
2. Upload the app to DA
3. Run Revit on DA

Phase 1: Write a Revit App

The first phase is to write a Revit add-in or an app. You can use Revit desktop for this. It's exactly the same environment as desktop add-in development. Differences are:

- No UI. Revit DB only. i.e., no reference to RevitAPIUI.dll
- Use failure handler
- No network access from your app
- Output to a file
- EntryPoint:
 - IExternalDBApplication

- HandleDesignAutomationReadyEvent()
- Bridge dll (nuget)
- The same app .bundle package

You can find Bridge dll on [nuget](#).

Below is an example of entry point for DA for Revit. Notice that it's using IExternalDBApplication and HandleDesignAutomationReadyEvent(). When DA for Revit is ready to execute your app, your event handler will be triggered. Once you get hold of rvtApp, the rest is the same as Revit API that we know of.

Entry Point

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

    public void HandleDesignAutomationReadyEvent(object sender, DesignAutomationReadyEventArgs e)
    {
        Autodesk.Revit.ApplicationServices.Application rvtApp = e.DesignAutomationData.RevitApp;
        bool stairsCreated = LayoutUtils.GenerateStairs(rvtApp, LayoutUtils.ReadSpreadsheetData());
        if (stairsCreated)
            e.Succeeded = true;
    }
    // more code ...
}
```

For more information about app bundle, please refer to the page about how to write app store app:

<https://www.autodesk.com/developer-network/app-store/revit>

You can find a table of tutorial videos toward the end of the page. Test your app using desktop version of Revit as much as possible before moving to the next phase.

Phase 2: Upload the App to DA

Once we have an app in a bundle format and are ready to submit to the DA, the next phase is to .zip up your app .bundle and upload it to DA. Then, you define your DA action or “activity” which includes the information about:

- App Bundle
- Input
- Output

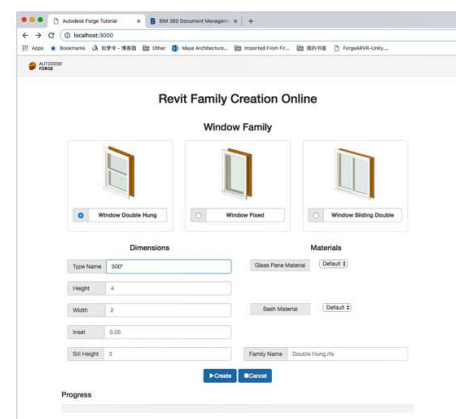
using:

<https://forge.autodesk.com/en/docs/design-automation/v3/reference/http/activities-POST/>

Here is a sample excerpt of Post activities' body looks like in case of the Window family creation sample. Notice it defines two input parameters and one output as well as app bundles and Revit version to use.

```
{
  "id": "CreateWindowFamilyActivity",
  "commandLine": [ "${engine.path}\\\\revitcoreconsole.exe /i ${args[templateFile].path} /al $
    (appbundles[CreateWindowFamilyApp].path)" ],
  "parameters": {
    "templateFile": {
      "zip": false,
      "ondemand": false,
      "verb": "get",
      "description": "Input Revit Window Family Template",
      "required": true
    },
    "windowParams": {
      "zip": false,
      "ondemand": false,
      "verb": "get",
      "description": "Window Family parameters",
      "required": false,
      "localName": "WindowParams.json"
    },
    "resultFamily": {
      "zip": false,
      "ondemand": false,
      "verb": "put",
      "description": "new created Window Family",
      "required": true,
      "localName": "WindowFamily.rfa"
    }
  },
  "engine": "Autodesk.Revit+2019",
  "appbundles": [ "${dasNickName}.CreateWindowFamilyApp+dev" ],
  "description": "Create Window Family Activity."
}
```

POST activities ex. Window Creation



Phase 3: Run DA with Your App

You are ready to run your app in DA. To do so, submit a job or “workitems” with the following information:

- The activity ID to use
- Input data or location of the data
- Output location

Return status will be:

- Pending | Complete | Error

You can use:

- Polling or Callback to check the status
- Log file to check the detail of result.

Here is an example of workitems for Window Family creation.

Post workitems

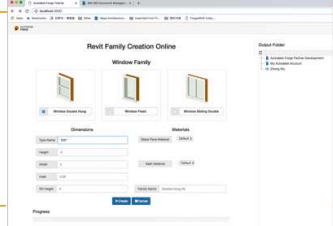
e.g. Window Creation

```
{
  "activityId": "{{dasNickName}}.CreateWindowFamilyActivity+dev",
  "arguments": {
    "templateFile": {
      "url": "https://developer.api.autodesk.com/oss/v2/buckets/revitiosamplebyzhong/objects/windowFamily.rft",
      "Headers": {
        "Authorization": "Bearer {{dataApiToken}}"
      }
    },
    "windowParams": {
      "url": "data:application/json,{ 'Types': [{ 'TypeName': 'My Type', 'WindowHeight': 6, 'WindowWidth': 3, 'WindowInset': 0.05, 'WindowSillHeight': 3 }, { 'TypeName': 'My New Type', 'WindowHeight': 8, 'WindowWidth': 6, 'WindowInset': 0.1, 'WindowSillHeight': 6 }], 'GlassPaneMaterial': 'Glass', 'SashMaterial': 'Maple', 'WindowStyle': 'DoubleHungWindow' }",
      "Headers": {
        "Authorization": "Bearer {{dataApiToken}}"
      }
    },
    "resultFamily": {
      "verb": "put",
      "url": "{{createWindowFamilyResultUrl}}",
      "Headers": {
        "Authorization": "Bearer {{dataApiToken}}"
      }
    }
  }
}
```

Input 1

Input 2

output



Notice workitems' arguments associate with the activities parameter definitions:

Post workitems

e.g. Window Creation

```
{
  "activityId": "{{dasNickName}}.CreateWindowFamilyActivity+dev",
  "arguments": {
    "templateFile": {
      "url": "https://developer.api.autodesk.com/oss/v2/buckets/revitiosamplebyzhong/objects/windowFamily.rft",
      "Headers": {
        "Authorization": "Bearer {{dataApiToken}}"
      }
    },
    "windowParams": {
      "url": "data:application/json,{ 'Types': [{ 'TypeName': 'My Type', 'WindowHeight': 6, 'WindowWidth': 3, 'WindowInset': 0.05, 'WindowSillHeight': 3 }, { 'TypeName': 'My New Type', 'WindowHeight': 8, 'WindowWidth': 6, 'WindowInset': 0.1, 'WindowSillHeight': 6 }], 'GlassPaneMaterial': 'Glass', 'SashMaterial': 'Maple', 'WindowStyle': 'DoubleHungWindow' }",
      "Headers": {
        "Authorization": "Bearer {{dataApiToken}}"
      }
    },
    "resultFamily": {
      "verb": "put",
      "url": "{{createWindowFamilyResultUrl}}",
      "Headers": {
        "Authorization": "Bearer {{dataApiToken}}"
      }
    }
  }
}
```

Input 1

Input 2

output

activities

```
{
  "parameters": {
    "templateFile": {
      "zip": false,
      "ondemand": false,
      "verb": "get",
      "description": "Input Revit Window Family Template",
      "required": true
    },
    "windowParams": {
      "zip": false,
      "ondemand": false,
      "verb": "get",
      "description": "Window Family parameters",
      "required": false,
      "localName": "WindowParams.json"
    },
    "resultFamily": {
      "zip": false,
      "ondemand": false,
      "verb": "put",
      "description": "new created Window Family",
      "required": true,
      "localName": "WindowFamily.rfa"
    }
  }
}
```

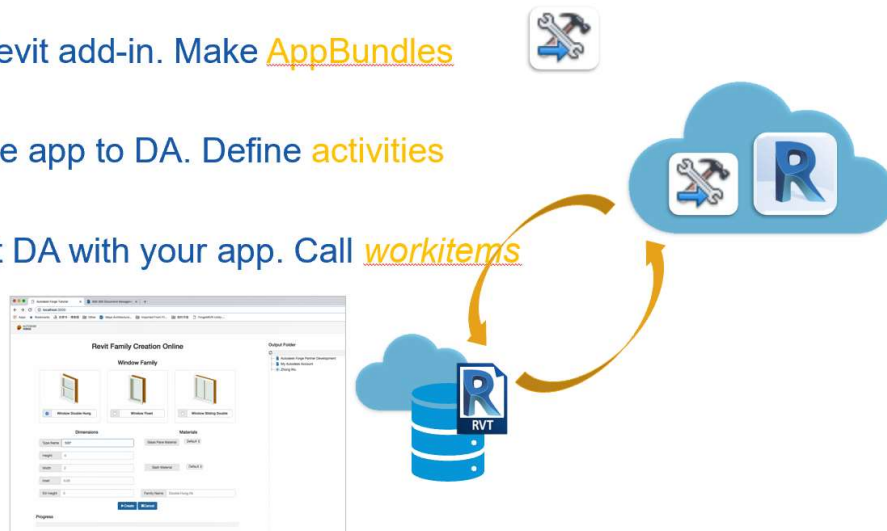


DA How To - Recap

Here is a recap of three main phases:

DA How To - Recap

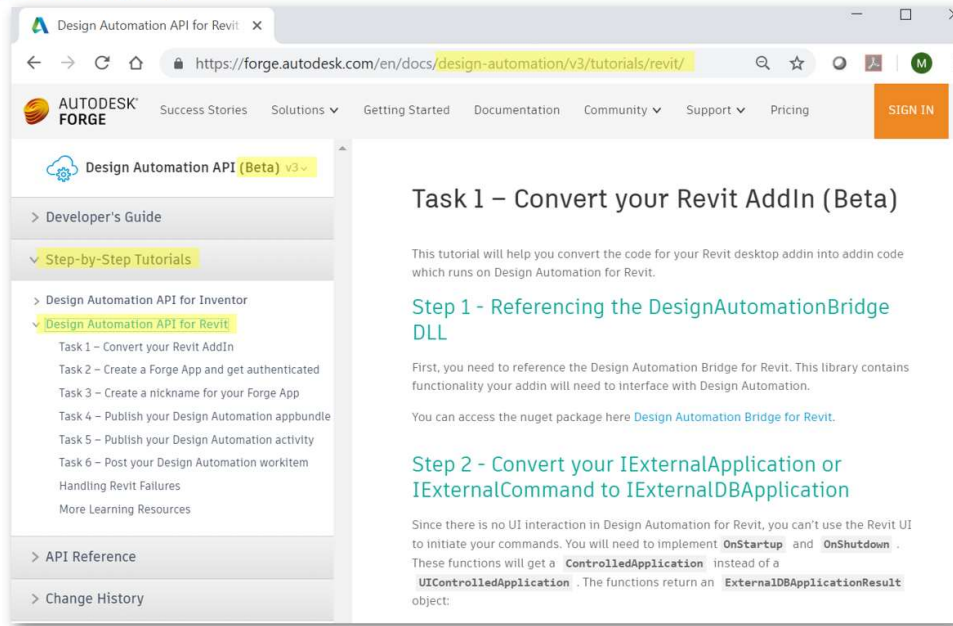
1. Write a Revit add-in. Make AppBundles
2. Upload the app to DA. Define activities
3. Run Revit DA with your app. Call workitems



Learn More

To learn more, please take a look at documentation and tutorials. In particular, [Step by Step Tutorials for Design Automation for Revit](#) is an excellent source to understand each step. [Modify Model](#) section of [LearnForge](#) is an excellent source of learning from example.

Learn More



Official Documentation

<https://forge.autodesk.com/en/docs/design-automation/v3/>

Tutorials

<https://forge.autodesk.com/en/docs/design-automation/v3/tutorials/revit/>

<http://learnforge.autodesk.io/#/tutorials/modifymodels>

Code samples

<https://github.com/autodesk-forge> (window family, upgrader, sketch it, local debug tool.)

Recordings

DevDay Online: <https://www.youtube.com/watch?v=G-pP37GQycQ>

AU presentation

<https://www.autodesk.com/autodesk-university/class/Revit-Data-Forge-How-Can-Design-Automation-Revit-API-Help-Me-2018>

BIM 360

In the sample we have shown, we use BIM 360 Docs. BIM 360 can be a natural choice because of the following reasons:

- DA itself does not store data
- Collaboration for Revit (C4R) and Revit Cloud Worksharing
- Additional features/tools which are construction specific

BIM 360 Products Family Overview

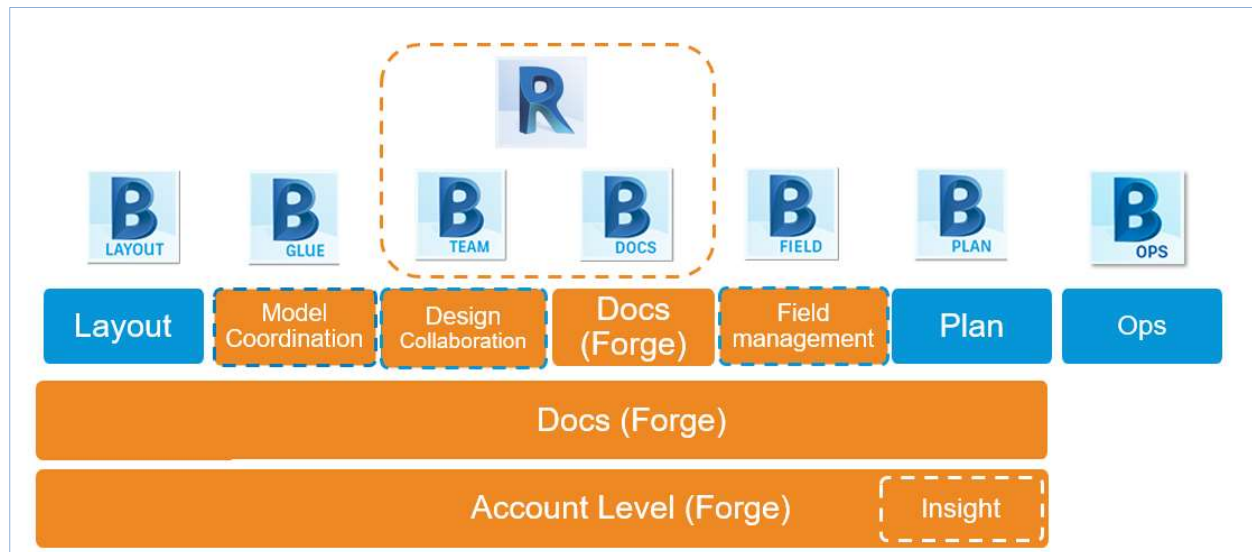
Before we talk about API, let's first look at where we are with BIM 360 family products. We have new names like BIM 360 Design, BIM 360 Build, Design Collaboration and Model Coordination. It can be confusing. Currently, we have the following products under BIM 360 branding:

- BIM 360 Glue
- BIM 360 Field
- BIM 360 Layout
- BIM 360 Plan
- BIM 360 Docs
- BIM 360 Team
- BIM 360 Ops
- (BIM 360 Account Admin)
- BIM 360 Design

Note that there is also **BIM 360 Build** if you look at the [BIM 360 home page](#). BIM 360 Build is named for marketing/sale purpose; it's a bundle of BIM 360 Field and BIM 360 Docs with Field Management module. Those two are known as Field classic and Next Generation (or Next Gen for short) Field. For our audience (i.e., developer community), it's important to understand that those two are separate products. Next Gen or Field Management module is built on top of Docs.

Another thing which worth mentioning is **BIM 360 Team** and **BIM 360 Design**. Both are related to the change with Collaboration for Revit (C4R). Autodesk no longer offers standalone BIM 360 Team. However, BIM 360 Team comes with C4R subscription, which is now branded as BIM 360 Design. The Revit feature that we know of as C4R, now called Revit Cloud Worksharing, is integrated with BIM 360 Docs since the last April with Revit 2019. If you use Revit 2018.3, you can use Both BIM 360 Team and Docs. Prior to Revit 2018.3, you can continue using BIM 360 Team.

Account Admin is not a product name as it is. Rather, it a feature that controls an account and project level access across BIM 360 product except Ops and Team. It has a good coverage of API's that you can take advantage of as a Forge developer. The figure below depicts the structure of BIM 360 family products as of this writing:



BIM 360 Product Family

BIM 360 Products API Today

In terms of API, following lists the status of API:

- BIM 360 Docs (+ anything built on top) – Forge API
- BIM 360 Team – Forge API
- BIM 360 Admin – Forge API
- BIM 360 Glue/Layout – not Forge API except Admin functionality
- BIM 360 Field – not Forge API except Admin functionality
- BIM 360 Plan – no API
- BIM 360 Ops – no API

Any APIs which will be built on top of Docs, including Issues, RFI and Checklists, will be Forge when API is exposed.

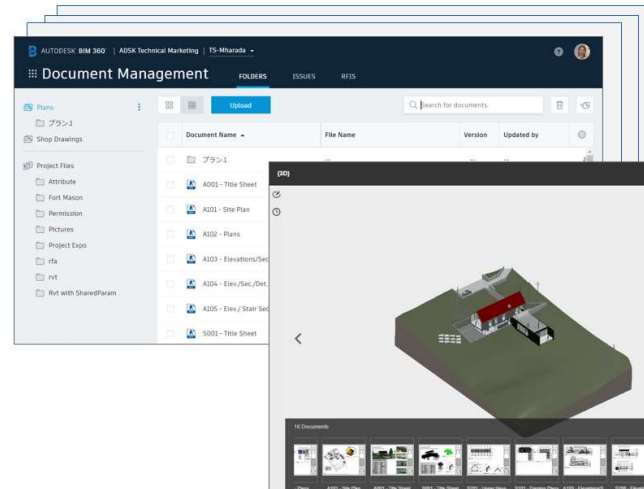
Note that Glue and Field have API, but they are not a part of Forge.

BIM 360 and API

API's for BIM 360 Docs is a collection of Forge API's. There is no single set of API specifically for BIM 360 Docs. Forge is a collection of web services that Autodesk is using internally to build their own product. BIM 360 Docs is a good example of such a product. The following is the list of Forge API that you can use with Docs:

BIM 360 Docs – API Overview

- Authentication (OAuth)
- Account Admin API
- Data Management API
- Model Derivative API
- Viewer
- Issues API
- RFIs API
- Checklists API



Authentication (OAuth) is used to login to the product. BIM 360 API (Account and project administration) are BIM 360 specific. Its API is for Account and Project management feature. Data Management is all about storing data: from hubs to projects to folders to each contents level. Once you get to an individual file, the next level is about model data. Model derivative API allows you to translate and extract meta data. You can use a viewer to view a model. Issues/RFI and Checklists are BIM 360 field/project management features.

Learn More – Forge and BIM 360 API

For more information, please take a look at the following information:

- Documentations
 - <https://forge.autodesk.com/developer/documentation>
 - <https://fieldofviewblog.wordpress.com/2017/01/17/bim-360-and-forge-overview/>
- Tutorial
 - <https://learnforge.autodesk.io/#/tutorials/viewhubmodels>
- Webcast Recordings
 - <https://fieldofviewblog.wordpress.com/2017/08/19/materials-from-bim-360-online-hackathon-webinars/>
- Code Samples
 - <https://github.com/autodesk-forge>
- Questions?
 - stackoverflow.com
 - forge.help@autodesk.com
 - <https://fieldofviewblog.wordpress.com/2016/10/27/where-to-get-help-about-forge/>

Forge Viewer



When it comes to Forge Viewer, we have plenty of samples and documentation is quite good. You might have seen them already. If you are new to Forge and Viewer, my suggestion will be to go through demo samples. I have put together demo sites through my blog together with learning materials and links to the github source code. Please take a look:

List of demo sites

<https://fieldofviewblog.wordpress.com/2018/06/16/forge-demo-sites/>

Documentations

https://forge.autodesk.com/en/docs/viewer/v6/developers_guide/overview/

Tutorial

<https://forge.autodesk.com/en/docs/viewer/v6/tutorials/basic-viewer/>

learnforge: <https://learnforge.autodesk.io/#/tutorials/viewmodels>

<https://fieldofviewblog.wordpress.com/forge/> Labs 1-5 minimalist

Code samples

<https://github.com/autodesk-forge/> (various. We put the link next to the demo sample whenever applicable)

Forge Resource - General

Finally, We add a few links to generic information.

- Web: <https://forge.autodesk.com>
- Learning Resources
 - LearnForge Tutorials: <https://forge.autodesk.com/LearnForge>
 - GitHub: <https://forge.autodesk.com/GitHub>
 - AR/VR Toolkit: <https://forge.autodesk.com/ARVRToolkit>
 - Stack Overflow: <https://forge.autodesk.com/Stack>
 - Accelerator: <https://forge.autodesk.com/accelerator>
- Community
 - Twitter: @AutodeskForge
 - Facebook @AdskForge
 - YouTube: <https://forge.autodesk.com/YouTube>
 - Blog: <https://forge.autodesk.com/blog>

If you have questions, feel free to contact Forge team through:

- <http://stackoverflow.com>
- forge.help@autodesk.com
- <https://fieldofviewblog.wordpress.com/2016/10/27/where-to-get-help-about-forge/>

Conclusion

Design Automation for Revit is a Revit running on the cloud as a service. Using DA for Revit, you can create, edit and extract a Revit model using the Revit API without having the Revit desktop environment. As an example, you can create a Revit family automatically by passing parameters. You can check a model against certain design criteria and create a report, and if

needed, automatically modify the model. It is also possible to extract model data which was not possible with Forge Model Derivative API. In this session, we have introduced you to DA for Revit. You learned how to convert your existing Revit add-ins to run in the cloud, and how and where to get started with leveraging Forge services integrating with viewer and BIM 360.

I hope this gives you a clear, good starting point with Design Automation for Revit. Thank you and good luck!