SD466729

Using Design Automation for Revit for Displaying RFAs in the Forge Viewer

Viraj Voditel
Techture

Learning Objectives

- Discover the different formats the Forge Viewer supports, and why the RFA format is a limitation and currently is not supported.
- Learn how to design a workflow for visualizing Revit families and their family types and accessing parameters in the Forge Viewer.
- Learn how to render hundreds of Revit families in the Forge Viewer and optimize for loading time and Forge credits usage.
- Learn how to create headless Revit plug-ins to enable a cloud-connected workflow using the Design Automation for Revit API.

Description

There are a variety of formats that the Forge viewer supports. The Model Derivative API enables translation of more than 60 different types of source file formats. Yet, there is one format ubiquitous in the architecture, engineering, and construction (AEC) industry that does not find its place in the list—RFA. There could be various ways to visualize RFA files in the viewer; the most common is: conversion to a format that's supported by Forge, though it's not the fastest or most effective. In this class, we'll look at a workflow that can help developers optimize up to 90% of their Forge credits usage. Accomplishing this requires building a headless Revit plug-in using the Revit API, preparing it for the Design Automation API, processing the outputs using the Model Derivative API, and finally rendering individual Revit families in the viewer. We'll also look at how we can give end users the ability to switch between multiple family types, as well as access the Instance and Type parameters directly in the Forge Viewer.

Speaker

Viraj is the CEO and Founder of Techture, a global BIM consulting firm having offices in UK, UAE, India and Singapore. He started out as a Student Expert for Autodesk while pursuing his Civil Engineering degree and currently is an Autodesk Expert Elite and a Certified Professional for various software. He is a BIM evangelist and frequently talks about BIM at various platforms. He has delivered technical lectures at the national and international level and is actively involved in championing the newest technologies in the AEC space. At Techture, he leads multidisciplinary teams on developing newer and more efficient workflows. He strives towards ensuring that they always keep up with the latest technologies and diversify into broader segments. Viraj has been able to amass a rich experience on BIM Implementation for various large scale projects including hospitals, hotels, airports, hydropower projects and smart cities.

What we'll Cover

Introduction

What is Forge?

Forge is a cloud-based developer platform from Autodesk. The Forge Platform offers APIs and services that help you access and use your design and engineering data via the cloud.

- BIM 360 API
- Data Management API*
- Model Derivative API*
- Design Automation API*
- Authentication API*
- Viewer API*
- Reality Capture API
- Token Flex API
- Webhooks

The ones marked (*) are the ones we'll be referring to as a part of this class.

The two which are particularly important are:

Model Derivative API

- This API can be used to prepare designs for rendering in the Viewer
- It can also be used to convert design files into other formats

Design Automation API

- Design Automation API for Revit is Revit's engine running in the cloud as a Forge service.
- It provides access to the full Revit DB API without a Revit desktop install, so that you can build cloud-native apps and services that create, extract, and modify Revit data

Different Formats supported by Forge

- Autodesk Forge viewer supports 70+ formats.
- Pertaining to the AEC industry, the popularly supported formats include .rvt, .dwg, .ifc, .dgn and more.
- However, .rfa, which is a format used for Revit families is not supported.

Different Revit formats

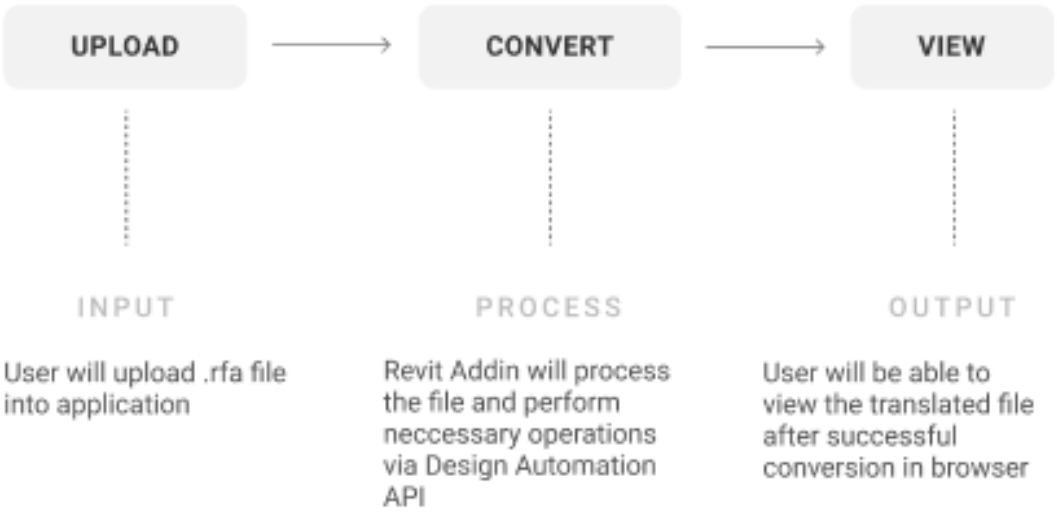
- There are 4 major formats in Revit
- When working in a project environment, the format is .rvt
- When working in a component/family environment the format is .rfa
- The other 2 formats are templates for the respective environments

Application Functionality

This section contains the description about using design automation and other forge APIs to display Revit Families (RFA) in Autodesk Forge Viewer.

Let's imagine on a screen there is an upload button and convert button, and the user would be able to upload RFA/s for his/her choice via clicking on the upload button and then would be able to view the uploaded RFAs on the viewer on the web browser.

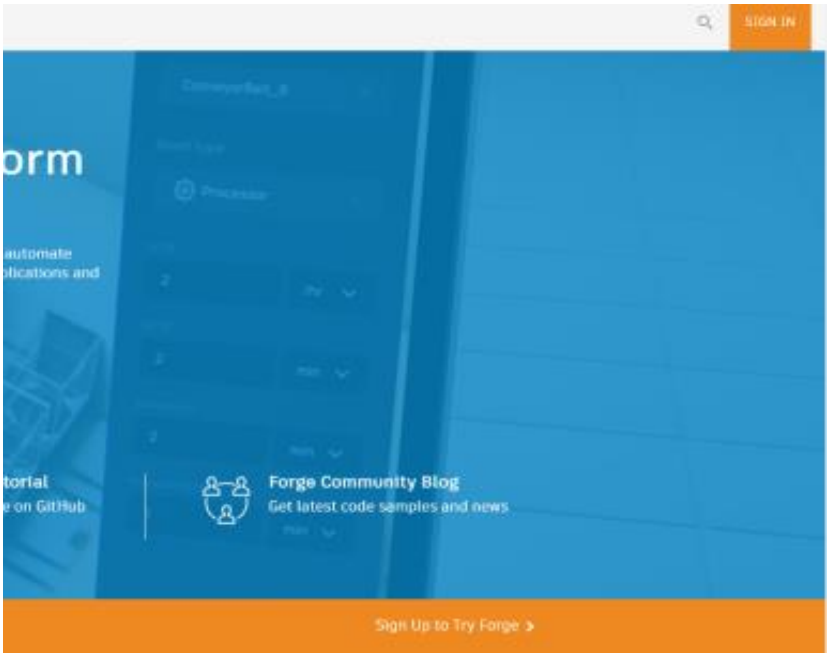
On the surface the functionality looks very straightforward right? But a lot is happening here, let's have a look at how things are handled under the hood



First Steps and Prerequisites

Forge is a cloud-based developer platform from Autodesk, The Forge Platform offers APIs and services that help you access and use your design and engineering data via the cloud.

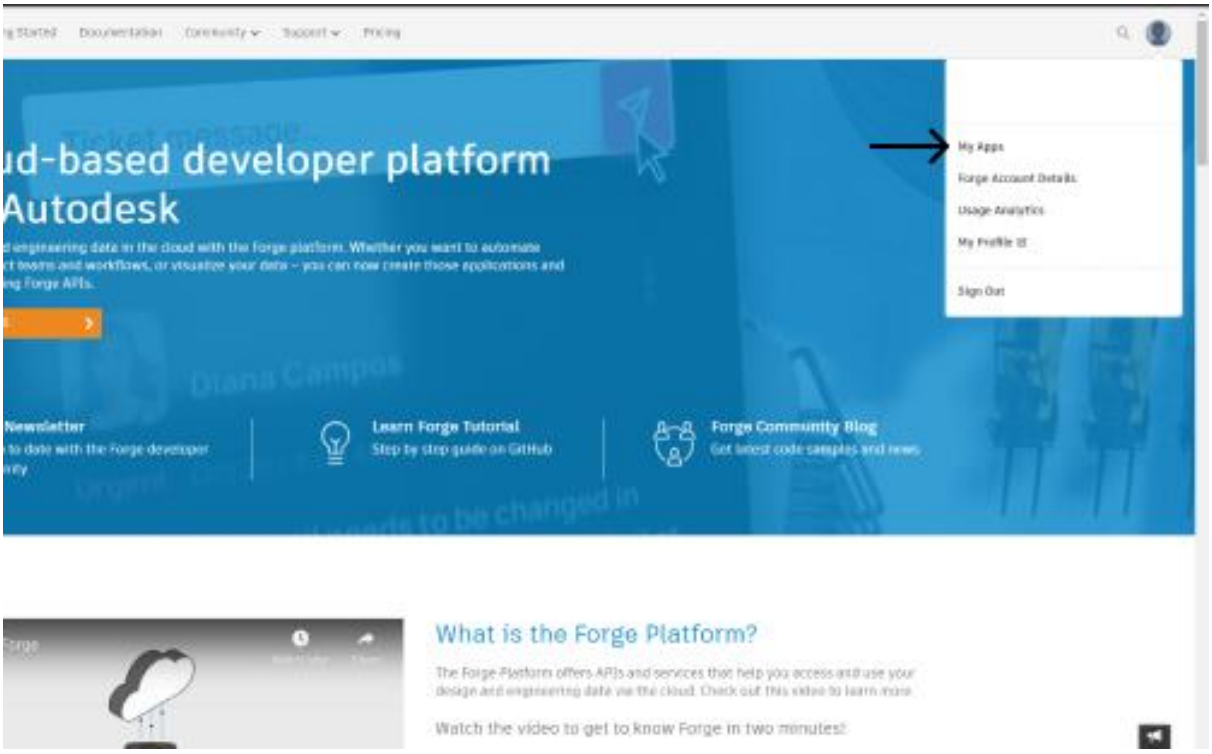
Sign up for a Forge account



Every user will need a Forge Account for using Forge APIs. creating an account is very simple, just go to https://forge.autodesk.com and create an account and follow the steps needed to register. You can also refer official Autodesk document on https://forge.autodesk.com/developer/getting-started

Create an app in your Forge account

After you sign in into your Forge account, go to “My Apps” and click on the “Create app” button, fill out the information given on the page and hit the “Create app” button. This will create a new app in your Forge account



Click on the newly created app, and copy client_id and client_secret under the App information section. These will be your Forge app credentials which will be used to communicate between your app and Forge. Note: Do not share your client_secret with anyone.

Forge APIs required for displaying RFAs in Forge Viewer

Following are the 5 APIs which are used in an end to end process for displaying RFA files in Forge Viewer

1. Data Management API
2. Model Derivative API
3. Design Automation API
4. Authentication API
5. Viewer API

Documentation provided by Autodesk for forge can be found at this link:
https://forge.autodesk.com/developer/documentation

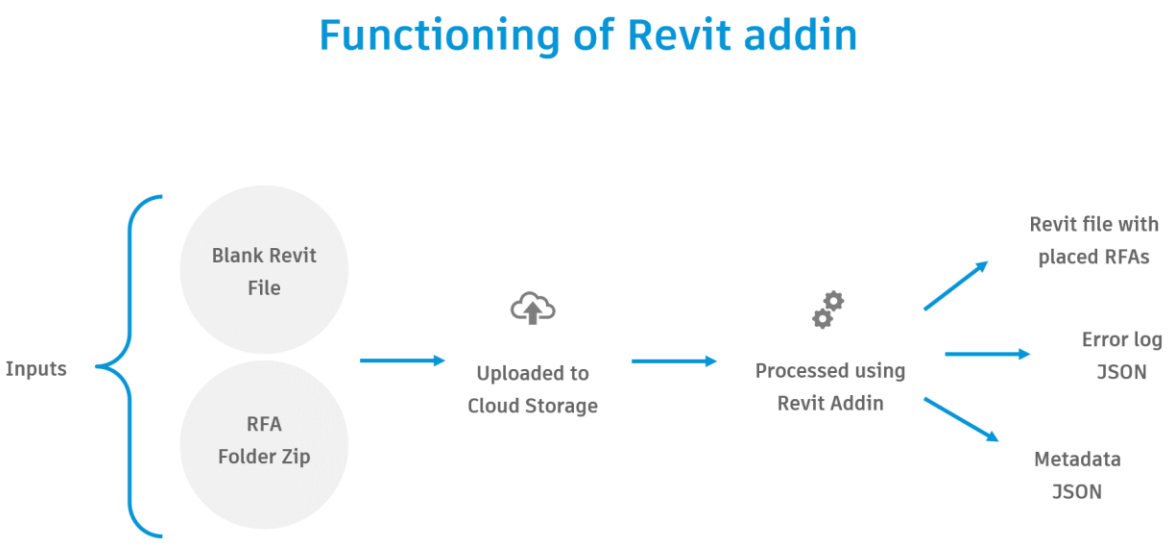
Problem Statement: How to convert RFA?

Usually, Model Derivative APIs are used for converting files which are then later shown in Viewer on the web. Model Derivative API supports 60 extensions, such .rvt, .ifc, .dwf, etc. but .rfa is not one of them. This means .rfa files cannot be directly converted by using Model Derivative APIs. This class will look at a workflow which will effectively suggest that we would require to put the .rfa files into one revit file which can be converted using Model Derivative APIs and shown in viewer.

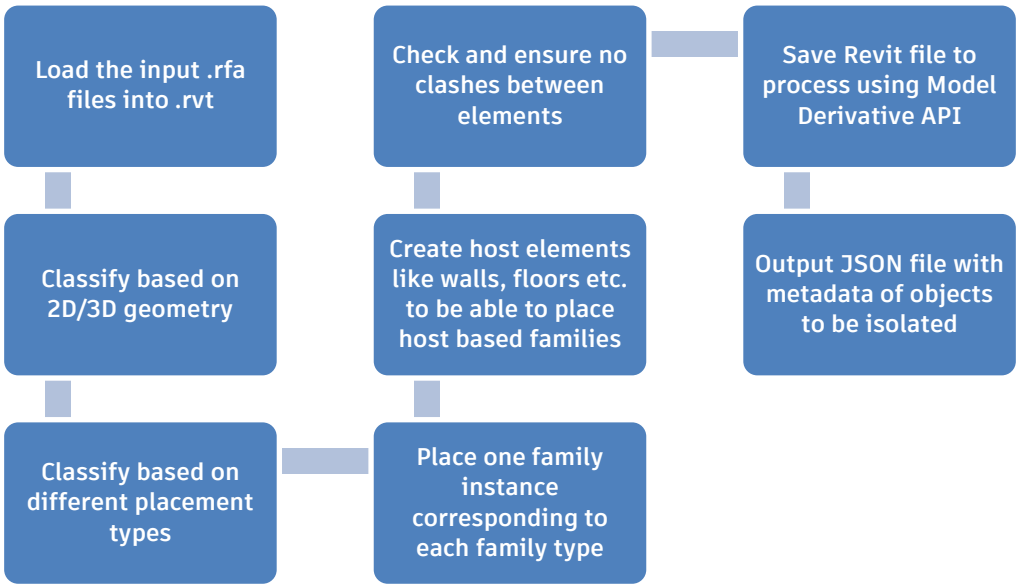
Majorly there are 3 key elements, in-order to achieve what we set out to do.

- Revit addin which would perform the action we want i.e place .rfa files into revit file.
- Execute the addin in Design Automation and process it
- Convert and display the processed file in Forge Viewer.

Functioning of Revit Addin



Revit Addin Logic



Let's take look at how the addin works internally. The addin we created basically requires 2 inputs:

1) Input Revit File - This file will be treated as an input revit file in which we place instances of that respective family type of that RFA

2) Input RFA Folder - We add the input RFA's of the user in this folder, zip it then upload it for processing in the addin.

Both of these files should be uploaded on a secure cloud storage. Once we start the Design Automation Process, it runs this Plugin, the Plugin will load Input Revit file in addin.

There is a limit on Autodesk Design Automation API for only 200 objects, this being a limiting factor, we have designed our addin to do is to place these instances on a level in the revit file to handle a much larger input of RFA files.

In the beginning the input revit file should be blank for the first iteration of 200 RFA's.

We create the first level if the input file has no levels created in it. We then iterate over each of the RFA and load each of the RFA in our Input Revit file.

In each iteration, we check the placement type of rfa and depending on that, we have used different Revit API functions to place instances of that RFA.

But majorly, we have distinguished them into 2 types-

1) Host Based - In this case, we need to create Host for that specific rfa.For e.g. if we need to place Void family, we need to create the wall first and then place that Void Family Instance on it.

2)Non Host Based - In this case, we can directly place the instance of that specific family type of that family.For e.g., if we need to place an instance of chair, we can directly place it without hosting it on any other object.

We are placing instances of all of the types of each RFA. During placement, we are storing a json file which will store the structure which will be like this:

Family name -> Type name -> ElementID of that type instance
and store it in the project information and also on our cloud storage in a JSON file.
In case of any errors we also log the errors in a separate file which is also on cloud storage.

If for instance, we have to devise a solution in which we have 400 RFA's.
So in this case, we will first work on 200 RFA's as it is the current limit.
We call it the 1st iteration and all of the process would be similar as above.

In the 2nd iteration, plugin expects an input file which is an output from the first iteration.

We will do all of the normal procedures as above for the first iteration except that, in this iteration, we are creating another level above the first level which has all of the previous instances placed.
We will place all of the instances for this 200 RFA's on this level.

As it has all of it's instances placed for 1st iteration, we will read the information of all JSON files from the one of the project parameters as we have already stored in 1st

iteration and likewise all the data will be added to JSON files after processing of the rfa files.

Getting the addin ready for Design Automation

Create nickname for app

You will need to setup a nickname for your forge app that will be used in Design automation, below snippet shows the request with which u can create a nickname for your forge app

```
PATCH /da/us-east/v3/forgeapps/me HTTP/1.1
Host: developer.api.autodesk.com
Authorization: Bearer eyJhbGciOiJIUzI1NiIsImtpZCI6Imp3dF9zeW1...
Content-Type: application/json

{
        "nickname": "<YOUR NICKNAME>"
}
```

Create AppBundle for Design Automation

Below snippet will show you how to can create an app bundle in Design Automation, in this request you will send the information related to your app bundle

```
curl --location --request POST 'https://dasprod-store.s3.amazonaws.com' \
--header 'Cache-Control: no-cache' \
--form 'key=apps/<appnick>/<app bundlekey>/1' \
--form 'content-type=application/octet-stream' \
--
form 'policy=eyJleHBpcmF0aW9uIjoiMjAyMC0wOS0y…XduSGNHWmpWbklTekl3MUJDZz09In1d
fQ==' \
--form 'success_action_status=200' \
--form 'success_action_redirect=""' \
--form 'x-amz-
signature=7edabb42a05a447c872a948b230feda6de296b4eff4a9bd70f6c8148ab391f36' \
--form 'x-amz-credential=ASIATGVJZKM3OOQALMD5/20200923/us-east-
1/s3/aws4_request/' \
--form 'x-amz-algorithm=AWS4-HMAC-SHA256' \
--form 'x-amz-date=20200923T103841Z' \
--form 'x-amz-server-side-encryption=AES256' \
--form 'x-amz-security-
token=IQoJb3JpZ2luX2VjEIn/////////wEaCXVz…ZjVnISzIw1BCg==' \
--form 'file=@<FILEPATH>'
```

Now you need to setup the alias for the app bundle

```
curl --location --request POST 'https://developer.api.autodesk.com/da/us-
east/v3/appbundles/<app bundle name>/aliases' \
--
header 'Authorization: Bearer eyJhbGciOiJIUzI1NiIsImtpZC…au7NgbTuM9RFi9kIGxpY
wd1KX7Pxpg2sRH8LTZv-0' \
--header 'Content-Type: application/json' \
--data-raw '{
      "version": 1,
      "id": "test"
}'
```

Create activity for the app bundle

```
curl --location --request POST 'https://developer.api.autodesk.com/da/us-
east/v3/activities' \
--
header 'Authorization: Bearer eyJhbGciOiJIUzI1NiIsImtpZCI6Imp3dF9zeW1tZXRyaWN
fa2V5In0…wMjc2OTczN30.hRau7NgbTuM9RFi9kIGxpYwd1KX7Pxpg2sRH8LTZv-0' \
--header 'Content-Type: application/json' \
--data-raw '{
    "id": "RevitAnalyticsActivity",
    "commandLine": [ "$(engine.path)\\\\revitcoreconsole.exe /i $(args[inputF
ile].path) /al $(appbundles[RevitFileAnalytics].path)" ],
    "parameters": {
      "inputFile": {
        "zip": false,
        "ondemand": false,
        "verb": "get",
        "description": "Creates revit file and loads rfa into it",
        "required": true,
        "localName": "$(inputFile)"
      },
      "inputZip":{
        "zip": true,
        "ondemand": false,
        "verb": "get",
        "description": "Creates revit file and loads rfa into it",
        "required": true,
        "localName": "$(inputZip)"
      },
      "outputFile": {
        "zip": false,
        "ondemand": false,
        "verb": "put",
        "description": "Output Revit model",
        "required": true,
        "localName": "outputFile.rvt"
```

```
      }
    },
    "engine": "Autodesk.Revit+2021",
    "appbundles": [ "<appnick>.<app bundle name>+test" ],
    "description": "write description"
}'
```

Now set up alias for activity

```
curl --location --request POST 'https://developer.api.autodesk.com/da/us-
east/v3/activities/RevitAnalyticsActivity/aliases' \
--
header 'Authorization: Bearer eyJhbGciOiJIUzI1NiIs…u7NgbTuM9RFi9kIGxpYwd1KX7P
xpg2sRH8LTZv-0' \
--header 'Content-Type: application/json' \
--data-raw '{
      "version": 1,
      "id": "test"
}'
```

Create work item

Set up drop box or similar cloud storage where you can provide access to design automation with API.

```
curl --location --request POST 'https://developer.api.autodesk.com/da/us-
east/v3/workitems' \
--
header 'Authorization: Bearer eyJhbGciO…au7NgbTuM9RFi9kIGxpYwd1KX7Pxpg2sRH8LT
Zv-0' \
--header 'Content-Type: application/json' \
--data-raw '{
    "activityId": "<appnick>.<app bundle name>+test",
    "arguments": {
        "inputFile": {
            "verb": "get",
            "url": "https://content.dropboxapi.com/2/files/download",
            "headers": {
                "Authorization": "Bearer Pa0NeErce7AAAAAAAAAA2AOczAx9QihaGl48
RtCHQdGlxOEOc8X8y53ntPqqcRzO",
                "Dropbox-API-Arg": "{ \"path\":\"/foldername/Test.rvt\" }"
            }
        },
        "inputZip": {
            "verb": "get",
            "url": "https://content.dropboxapi.com/2/files/download",
```

```
            "headers": {
                "Authorization": "Bearer Pa0NeErce7AAAAAAAAAA2AOczAx9QihaGl48
RtCHQdGlxOEOc8X8y53ntPqqcRzO",
                "Dropbox-API-
Arg": "{ \"path\":\"/foldername/RFA_FOLDER.zip\" }"
            }
        },
        "outputFile": {
            "verb": "post",
            "url": "https://content.dropboxapi.com/2/files/upload",
            "headers": {
                "Authorization": "Bearer Pa0NeErce7AAAAAAAAAA2AOczAx9QihaGl48
RtCHQdGlxOEOc8X8y53ntPqqcRzO",
                "Content-Type": "application/octet-stream",
                "Dropbox-API-
Arg": "{\"path\":\"/foldername/outputFile.rvt\", \"mode\":\"add\"}"
            }
        },
        "inputJson": {
            "verb": "post",
            "url": "https://content.dropboxapi.com/2/files/upload",
            "headers": {
                "Authorization": "Bearer Pa0NeErce7AAAAAAAAAA2AOczAx9QihaGl48
RtCHQdGlxOEOc8X8y53ntPqqcRzO",
                "Content-Type": "application/octet-stream",
                "Dropbox-API-
Arg": "{\"path\":\"/foldername/params.json\", \"mode\":\"add\"}"
            }
        },
        "inputError": {
            "verb": "post",
            "url": "https://content.dropboxapi.com/2/files/upload",
            "headers": {
                "Authorization": "Bearer Pa0NeErce7AAAAAAAAAA2AOczAx9QihaGl48
RtCHQdGlxOEOc8X8y53ntPqqcRzO",
                "Content-Type": "application/octet-stream",
                "Dropbox-API-
Arg": "{\"path\":\"/foldername/Error.json\", \"mode\":\"add\"}"
            }
        }
    }
}'
```

Check work Item status

Now that we have created workitem, we need to check the status of the same, and for that Autodesk provides another API, given below

```
curl --location --request GET 'https://developer.api.autodesk.com/da/us-
east/v3/workitems/<WORKITEM ID>\
```

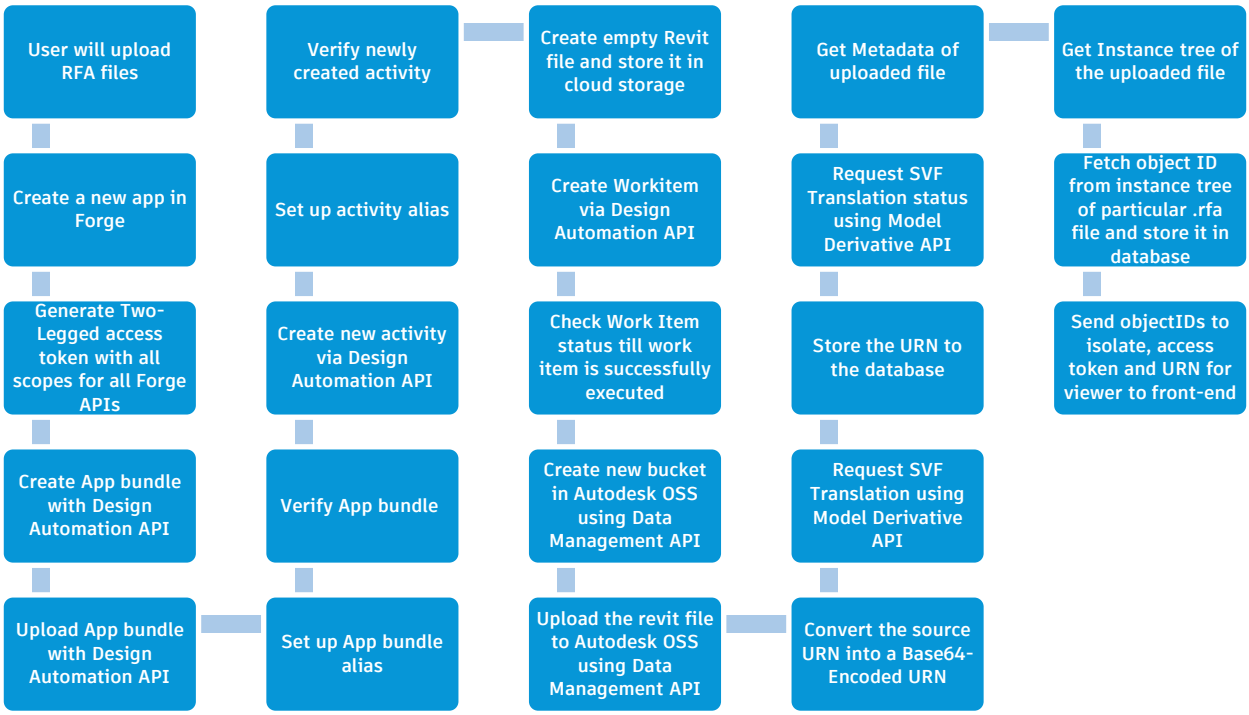
```
--
header 'Authorization: Bearer eyJhbGciOiJIUzI1NiIsImtpZCI6Imp3dF9zeW1tZXRyaWN
fa2V5In0.eyJzY29wZSI6Wy…bTuM9RFi9kIGxpYwd1KX7Pxpg2sRH8LTZv-0' \
--header 'Content-Type: application/json'
```

After completing above steps , your design automation execution cycle is complete

You can refer Design automation guide provided by Autodesk here
https://forge.autodesk.com/en/docs/design-automation/v3/tutorials/revit/

Overall Process for Conversion

There are many minor steps involved in the conversion process, let's have an overview of the process end to end.



- User will upload any number of RFA files
- Create a new app in Autodesk Website
- Generate Two Legged access token with all scopes for ALL forge APIs
- Create App bundle with Design Automation API
- Upload App bundle with Design Automation API
- Set up App bundle alias
- Verify App bundle
- Create new activity via Design Automation API
- Set up activity alias
- Verify newly created activity
- Create empty Revit file and store it in cloud storage
- Create Workitem via Design Automation API
- Check Work Item status till work item is successfully executed
- Create new bucket in Autodesk OSS using Data Management API
- Upload the revit file to Autodesk OSS using Data Management API
- Convert the source URN into a Base64-Encoded URN
- Request SVF Translation using Model Derivative API
- Store the URN to the database
- Request SVF Translation status using Model Derivative API
- Get Metadata of uploaded file
- Get Instance tree of the uploaded file
- Fetch object ID from instance tree of particular rfa file and store it in database
- Send objectIDs to isolate, access token and URN for viewer to front-end
- Using Forge Autodesk APIs

A step by step look at the process

Let's do the actual process of uploading , converting and viewing the file and see how simple it is actually. The only prerequisite is that you must have created an Autodesk Forge account and should have your client id and client secret available with you

STEP 1: Get your authentication token

Let's take a look at how to get a simple 2 legged oauth token from Forge APIs

```
POST /authentication/v1/authenticate HTTP/1.1

Host: developer.api.autodesk.com

Content-Type: application/x-www-form-urlencoded

client_id=<CLIENTID>&client_secret=<CLIENTSECRET>&grant_type=client_credentials&scope=data:read
```

A valid response body with status code of 200. You can take a look at official documentation here: https://forge.autodesk.com/en/docs/oauth/v2/overview/

```
{

  "access_token": "eyJhbGciOiJIUzI1NiIsI...mtpZCI6Im 1tZXRyaWNfa2V5InO.eyJzY29wZg....UTJO-LD2WSC",

  "token_type": "Bearer",

  "expires_in": 3599

}
```

STEP 2: Upload file to Autodesk Storage for conversion

Since it would not be possible for us to do the first half of the process that was related to Design Automation APIs, lets see how we can convert a file with Model Derivative APIs

There are 3 major steps in converting any supported file.
1) Upload a file
2) Process the file via Model Derivative API

3) View it in Forge Viewer

Make sure you have already created a bucket in Autodesk OSS before uploading any file. If not you can do so by below code snippet or for detailed understanding visit https://forge.autodesk.com/en/docs/data/v2/reference/http/buckets-POST/

```
POST /oss/v2/buckets HTTP/1.1

Host: developer.api.autodesk.com

Authorization: Bearer eyJhbGciOiJI-LITD2WSCo....

Content-Type: application/json
```

```
{

  "bucketKey": "yourbucketname",
  "policyKey": "persistent"

}
```

Now that you have created a bucket in Autodesk OSS, lets upload a file in to it for conversion

```
PUT /oss/v2/buckets/mybucket/objects/FILENAME HTTP/1.1

Host: developer.api.autodesk.com

Authorization: Bearer eyJhbGciOiJIUzI1NiIs...
Content-Type: application/octet-stream

Cookie: PF=kQMtLhxRMMFS5ZLUXw9K9Q

"<file contents here>"
```

You would be able to upload a file into Autodesk OSS via above API. A response body would look similar to the response below with a status code of 200. Refer https://forge.autodesk.com/en/docs/data/v2/reference/http/buckets-:bucketKey-objects-:objectName-PUT/ for more details.

```
{
  "bucketKey": "mybucket",
  "objectId": "urn:adsk.objects:os.object:mybucket/ABC.rvt",
  "objectKey": "ABC.rvt",
  "sha1": "1ad934ea67a012bd557f3c24b7ba929cf7f95aaa",
  "size": 28807168,
  "contentType": "application/octet-stream",
  "location": "https://developer.api.autodesk.com/oss/v2/buckets/mybucket/objects/ABC.rvt
}
```

STEP 3: Convert the uploaded file

You can convert any supported extensions by Forge with the Model Derivative APIs. Let's take a look at the API below:

```
POST /modelderivative/v2/designdata/job HTTP/1.1
Host: developer.api.autodesk.com
Authorization: Bearer eyJhbGciOiJ...
Content-Type: application/json
```

```
{
 "input": {
```

```
  "urn":
"dXJuOmFkc2sub2JqZWN0czpvcy5vYmplY3Q6dGVzdGFuZGRldjIyMDkyMDIwLOhWQUNfQUhVLENoa
WxsZXIlMjAmJTIwQ29vbGluZyUyMFRvd2VyMzEwNy5ydnQ"

 },

 "output": {

  "formats": [

   {

    "type": "svf",

    "views": [

     "2d",

     "3d"

    ]

   }

  ]

 }

}
```

Conversion status of the processing file

Converting a supported file is divided into 2 parts. Firstly you make a request to convert the file. This starts the operation of converting the file and then with another API you check the status of the conversion if it was successful and complete. Let's take a look at the second API below which checks the status of the job.

```
GET /modelderivative/v2/designdata/<URN>/manifest HTTP/1.1
```

```
Host: developer.api.autodesk.com

Authorization: Bearer eyJhbGciOiJIUzI1Ni...
Content-Type: application/json
```

Once the file has been converted, copy the urn and let's take a look at how to connect it and start the viewer on the next page.

STEP 4: View the converted file in Forge Viewer

Setup HTML File

Preparing for viewer is very simple, create a folder in your desired directory, and create index.html file inside it and copy below code into it

```
<head>

	<meta name="viewport" content="width=device-width, minimum-scale=1.0,
initial-scale=1, user-scalable=no" />

		<meta charset="utf-8">
			<link rel="stylesheet"
href="https://developer.api.autodesk.com/modelderivative/v2/viewers/7.*/style
.min.css" type="text/css">

	<script
src="https://developer.api.autodesk.com/modelderivative/v2/viewers/7.*/viewer
3D.min.js"></script>

	<style>

		body {

		margin: 0;

		}

		#forgeViewer {
```

```
                    width: 100%;
                    height: 100%;
                    margin: 0;
                    background-color: #F0F8FF;
                    }

        </style>

</head>
<body>

 <div id="forgeViewer"></div>

</body>
```

Setup Javascript file and initialize viewer

Create a main.js file in the same directory where u have created index.html file and copy paste below code into it

```
var viewer;

var options = {

  env: 'AutodeskProduction',

  api: 'derivativeV2',  // for models uploaded to EMEA change this option to 'derivativeV2_EU'

  getAccessToken: function(onTokenReady) {

    var token = 'YOUR_ACCESS_TOKEN';

    var timeInSeconds = 3600; // Use value provided by Forge Authentication (OAuth) API

    onTokenReady(token, timeInSeconds);

  }

};
```

```
Autodesk.Viewing.Initializer(options, function() {

  var htmlDiv = document.getElementById('forgeViewer');

  viewer = new Autodesk.Viewing.GuiViewer3D(htmlDiv);

  var startedCode = viewer.start();

  if (startedCode > 0) {

     console.error('Failed to create a Viewer: WebGL not supported.');

     return;

  }

  console.log('Initialization complete, loading a model next...');

});
```

Please paste the access token that you get from your forge app in place of 'YOUR_ACCESS_TOKEN

Connect the Document with Forge Viewer

Let's complete the final step and view the file in viewer, paste the below code into your main.js file

```
var documentId = 'urn:<YOUR URN HERE>';

Autodesk.Viewing.Document.load(documentId, onDocumentLoadSuccess,
onDocumentLoadFailure);

function onDocumentLoadSuccess(viewerDocument) {

  var defaultModel = viewerDocument.getRoot().getDefaultGeometry();

  viewer.loadDocumentNode(viewerDocument, defaultModel);

}
```

```
function onDocumentLoadFailure() {

  console.error('Failed fetching Forge manifest');

}
```

Paste the urn that you have saved from the file conversion, don't worry if you haven't saved or copied it, just hit that api again with all the same parameters and you will get the URN.

After pasting it, open the file and you will see your converted file in the browser.

STEP 5: Isolate specific objects in Forge viewer

Inorder to isolate the specific family you will need to get the meta data from the revit file.

You can refer https://forge.autodesk.com/en/docs/model-derivative/v2/tutorials/xtract-metadata/ on how to extract meta data from file.

You will need to isolate the specific rfa object, you wish to view, and inorder to isolate you need to perform 2 key actions.

1) Get Dbid/ Object id of the object you wish to isolate (this can be obtained from meta data)
2) Call isolate function on the dbid/object id

Below snippet shows how you can isolate object in forge viewer.

```
viewer.addEventListener(Autodesk.Viewing.OBJECT_TREE_CREATED_EVENT, ev => {
        let dbids = [<YOUR DBIDS>]
        viewer.isolate(dbids)
        viewer.fitToView(dbids[0], 1, true);
      })

}
```

We need to isolate a particular object because we are placing all the rfa objects uploaded by user into a revit file, which can be further differentiated by isolating them.

Conclusion

Hopefully you have got a good idea of how to implement RFAs in Forge viewer.

If you wish to learn more, visit https://forge.autodesk.com/en/docs/viewer/v7/developers_guide/viewer_basics/ inorder to learn more details of the viewer.