SD124141

Best Practices with Forge REST APIs

Jaime Rosales D.
Autodesk, Inc

Philippe Leefsma
Autodesk, Inc.

Learning Objectives

- Start development with the Forge server-side (REST) APIs
- Create workflows using REST apps for a faster development (PAW, Postman, Insomnia)
- cURL your way through Forge with prettified JSON responses
- Learn useful resources for Forge server-side API focus

Description

This class will cover how to accelerate your development with the Forge server-side (REST) APIs. A journey through different workflows demonstrating how to access JSON data. We will also cover best practices on how to use REST client applications to work faster and get the most out of the Forge Platform. Use of Model Derivative API, Data Management API, including access to BIM 360 Docs data, Authentication and Design Automation API exploration from cURL. This class is not suitable for developers at the beginner level.

Your Forge DevCon Expert(s)

Jaime Rosales *is a Senior Developer Consultant since July 2014 for the Autodesk Developer Network and Forge Development Partner Program. He joined Autodesk in 2011 through the acquisition of Horizontal Systems: the company that developed the cloud-based collaboration systems—now known as BIM 360 Glue (the Glue). He was responsible for developing all the add-ins for BIM 360 Glue, using the API's of various AEC desktop products.*

He is currently empowering customers with the use of Autodesk Forge platform, through the world with hosted events such as Cloud Accelerators, AEC Hackathons, VR & AR Hackathons.. He spends part of his time presenting in different parts of the US, Europe and Latin America about the implementation of 3D content for web applications through the JS and NodeJS communities.

Phillipe Leefsma *has been a developer advocate on the Autodesk API's for the last 10 years. Fascinated by 3D graphics and CAD, he had the chance to play with the company's top desktop products API's using various programming languages such as C++, .Net, Python, JavaScript and a few others helping programmers solve their daily challenges on a world-wide scale.*

When Autodesk started a cloud turn a few years ago by pulling out the Forge platform – a connected set of web services aiming at democratizing 3D CAD in the browser, he realized that writing apps for the world wide web is THE most exciting job a developer can do. Interested about all kinds of optimizations, he loves to learn and experiment about new frameworks, full stack development, transpilers, bundlers and alike…

Big fan of full JS stack, never hesitate to chat with him about why the Web is the greatest playground ever!

Start development with the Forge server-side (REST) APIs

In this class, we will focus on the REST side of the Forge Platform.

A simple definition to what is REST?

An ***architectural style*** called ***REST (Representational State Transfer)*** advocates that web applications should use HTTP as it was ***originally envisioned***. Lookups should use GET requests. PUT, POST, and DELETE requests should be used for ***mutation, creation, and deletion respectively***.

Many of the different API's we have with the Forge Platform except the Viewer will use this architectural style.

OAuth

OAuth, specifically OAuth2, is the open standard used across the Forge Platform for token-based authentication and authorization.

The basic flow for using OAuth on the platform is as follows:

1. Your app makes an HTTP call to an OAuth REST endpoint and provides its credentials.
2. A token is returned to your app.
3. In making subsequent HTTP calls to various APIs on the platform, your app includes the token in a request header.

Data Management API

The Data Management API provides a unified and consistent way to access data across BIM 360 Team, Fusion Team (formerly known as A360 Team), BIM 360 Docs, A360 Personal, and the Object Storage Service.

With this API, you can accomplish a number of workflows, including accessing a Fusion model in Fusion Team and getting an ordered structure of items, IDs, and properties for generating a bill of materials in a 3rd-party process. Or, you might want to superimpose a Fusion model and a building model to use in the Viewer.

The Data Management API is composed of the following services:

Project Service	Navigate to a project from a BIM 360 Team hub, a Fusion Team hub, an A360 Personal hub or a BIM 360 Docs account. The project acts as an "anchor" point for data available through the API.
Data Service	Navigate and manage the BIM 360 Team, Fusion Team, Bim 360 Docs, or A360 Personal metadata in terms of folders, items, and versions, as well as the relationships between these entities. An item could be a file or a Fusion Team design. Each item can have multiple versions.
Schema Service	This allows your application to understand the structure and semantics of extended data types, like Fusion designs.
Object Storage Service (OSS)	This allows your application to download and upload raw files (such as PDF, XLS, DWG, or RVT) that are managed by the Data Service.

There are two key data access paradigms that make up the Data Management API.

- Accessing data from Autodesk SaaS applications using any of the Data Management services.
    - For BIM 360 Team, Fusion Team, and A360 Personal, end users need to provide 3-legged authentication for your app to access the data.
    - For BIM 360 Docs, an account administrator needs to add an integration with your app in BIM 360 Enterprise. You can access data using either 2-legged or 3-legged authentication.
- Managing and storing files from your app on the Forge platform, independent of any Autodesk SaaS application. You need to use the Object Storage service (OSS).

To navigate and access BIM 360 Team, Fusion Team, BIM 360 Docs, A360 Personal, and OSS data, you need to be familiar with the following terminology:

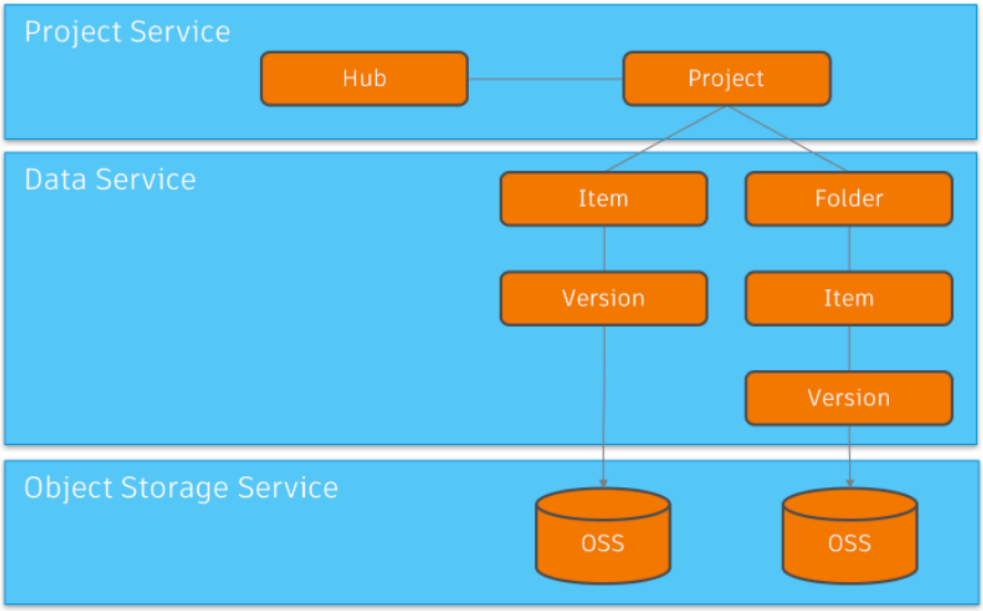
Service	Base Type	Description
Project	`hubs`	a BIM 360 Team hub, Fusion Team hub, BIM 360 Docs account, or A360 Personal hub
Project	`projects`	a BIM 360 Team, Fusion Team, BIM 360 Docs, or A360 Personal project
Data	`folders`	a logical organization of items within a project
Data	`items`	one or more versions of files, such as `dwg`, `pdf`, or Fusion designs and drawings
Data	`versions`	a specific state of an item; analogous to a specific version of a file
OSS	`buckets`	containers for objects with globally unique names
OSS	`objects`	binary data identified by a URN or key, stored in a specific bucket

Generally speaking, apps will use the Project service to navigate from a `hub` to a `project`.

One of the attributes associated with a `project` is the `rootFolder`.

Using the Data service, applications can navigate the folder hierarchy to a specific `item` and `version`.

When a `version` represents a file, the file can be downloaded using the endpoints exposed by the Object Storage Service (OSS).



Object Storage Service (OSS)

In OSS, files are stored as `objects` in `buckets`. In addition to providing your app the ability to download data from the broader Forge ecosystem, it also provides the functionality to manage your app's own buckets and objects (including creation, listing, deleting, uploading, and downloading). To avoid arbitrarily long API responses, OSS paginates when listing buckets and objects.

Each bucket also has a retention policy that determines object retention time:

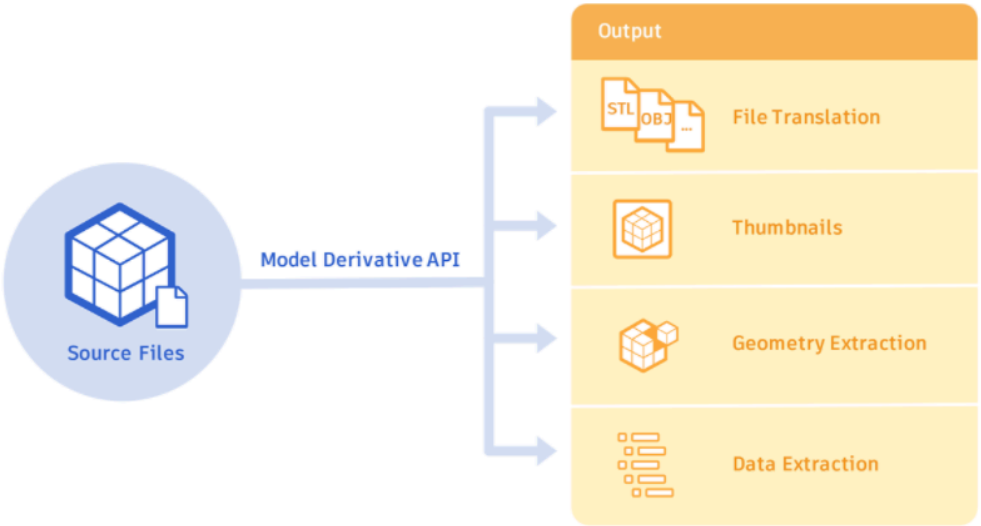
- `transient`: Cache-like storage that persists for only 24 hours, ideal for ephemeral objects
- `temporary`: Storage that persists for 30 days, good for data that is uploaded and accessed, but not needed permanently
- `persistent`: Storage that persists until it's deleted

Model Derivative API

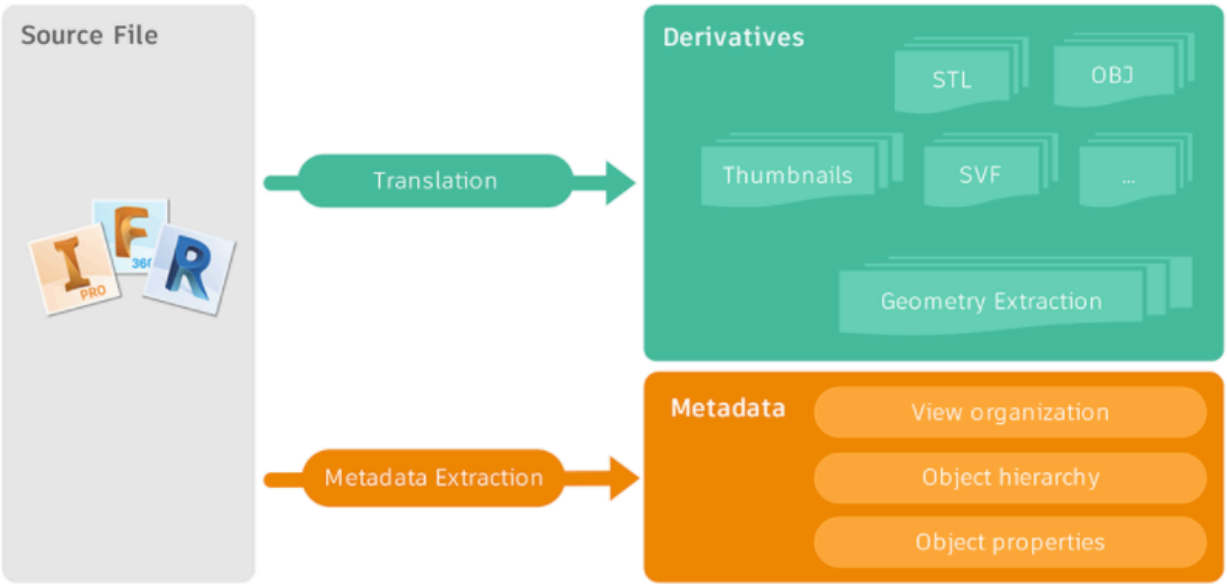
The Model Derivative API enables users to represent and share their designs in different formats, as well as to extract valuable metadata. (Its translation functionality was previously bundled as part of the "View and Data API".)

The API offers the following features:

- Quickly translate the design into different formats, such as STL and OBJ.
- Translate designs into SVF format for extracting data and for rendering files in the Viewer.
- Extract object hierarchy trees, properties, and geometries of selected parts of a design.
- Create different-sized thumbnails from design files.



The Model Derivative API translates source files into output files (derivatives) of different formats. Information about these files, such as output file URNs and translated job statuses, is conveniently stored in manifests, which reference data about the derivatives. You can download derivatives and retrieve metadata, which allows you to identify individual objects within a model, their geometric representations, and associated properties.



Design Automation API

Formerly known as the “AutoCAD I/O API”, the Design Automation API provides the ability to run scripts on your design files, leveraging the scale of the Forge Platform to automate repetitive tasks.

At present, the API provides the ability to run scripts on AutoCAD DWG files, with plans in the works to expand to file types generated by other design software.

Imagine you have thousands of DWG files stored in the cloud, and you need them all converted to PDF files. Ordinarily, you would have to download all the files, run a script on them in the AutoCAD desktop software, and then potentially upload them all back to the cloud. Your efficiency would be bottlenecked by the processing power of your computer and your network bandwidth, and you would have to instrument logging and retry logic in your code to ensure that the entire job completed.

With the Design Automation API, you can offload all that processing to the Forge Platform, which can process those scripts at a much greater scale and efficiency.

Current functionalities for DWG files include

- creating new DWG files
- querying for information in existing DWG files
- purging drawings and saving them to other DWF file formats
- plotting DWG files to DWF and PDF
- translating text from one language to another

Familiarize yourself with these concepts that are used throughout the documentation:

Term	Definition
Activity	an action that can be executed within the AutoCAD core engine For example, this could be the plotting of a DWG to a PDF file or the updating of the CAD standards currently applied to a drawing.
AppPackage	a module referenced by an Activity in order to perform specific functions For example, this might be a custom AutoLISP routine that extracts Xdata attached to objects in a drawing or a script file that plots a DWG to a PDF file.
module file	an AppPackage entity
WorkItem	a job that is submitted to and executed by the AutoCAD core engine A WorkItem is used to execute an Activity; the relationship between an Activity and WorkItem can be thought of as a “function definition” and “function call”, respectively. *Note: Once a WorkItem is created, it cannot be modified.*
Engine	the actual processing engine that runs the WorkItem job and processes the Activity Two engines are available: AutoCAD 2015 and AutoCAD 2016.

BIM 360 API

The BIM 360 API allows developers to develop apps that integrate with the Autodesk BIM 360 platform to extend its capabilities in the construction ecosystem. Integration use cases include, but not limited to, connecting BIM 360 with Enterprise Resource Planning (ERP) systems for project automation, connecting BIM 360 data with project management solutions, integrating BIM 360 data with cost estimation and financial systems, and many others. Integrations built with the BIM 360 API help designers, engineers, and contractors reduce manual data entry, improve data quality and consistency, and automate construction workflows.

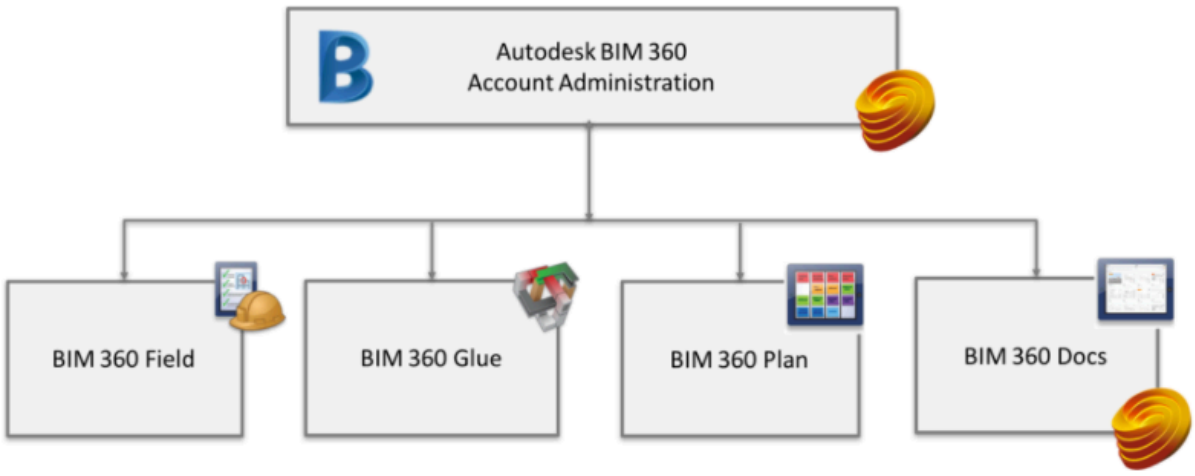
At present, the BIM 360 API offers account-level functionalities that enable developers to work with BIM 360 account administration workflows.

Account-Level Functionality

The account-level functionalities of BIM 360 API allow customers to connect their BIM 360 accounts with external applications in their construction ecosystem. They provide secure access to your BIM 360 accounts, serving as the single point of integration for projects, partner companies, user management, and other account-level information. They enable better-connected workflows across the project lifecycle, shared data across the project team, and construction insight across your project portfolio.

Features

- Account Access Control: secure access to BIM 360 enterprise accounts/tenants
- Project Service: ability to create a unified project across all BIM 360 services, including Docs, Glue, Field, and Plan (as illustrated below)
- Project Profile: shared project ID and profile
- Partner Companies Directory: add, edit, and synchronize company data with external systems
- Master Member Directory: add and synchronize users with external systems
- Business Unit: add and edit business units with external systems



Create workflows using REST apps for a faster development (PAW, Postman, Insomnia)

Since we know that many of the API's in the Forge Platform are REST based, I will share with you which tools I have been using to make my development more robust and less time consuming so results are reach faster. As I think it's not just for me and anybody who is using REST API would benefit, I'm going to give a quick summary here.

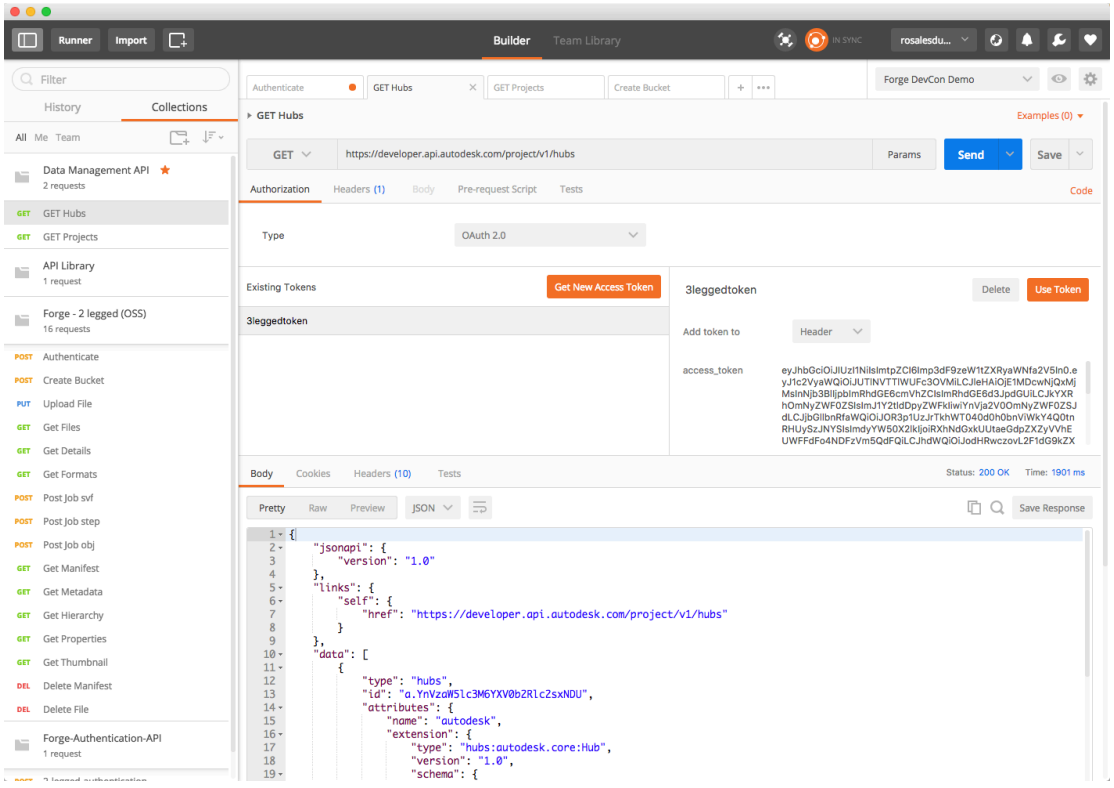
The ones that I'm currently using actively are:

Postman: https://www.getpostman.com/

Paw: https://luckymarmot.com/paw

Insomnia: https://insomnia.rest/

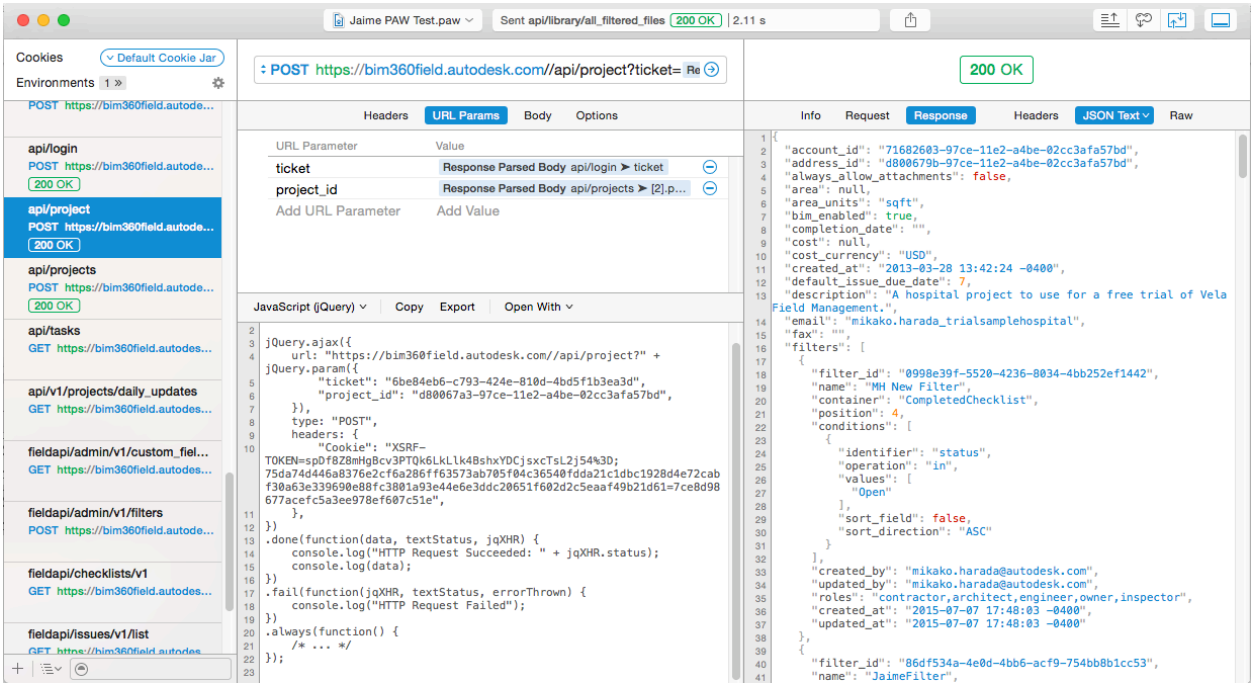
Postman:
Is a HTTP client tool for testing web services calls, it has a very intuitive UI and anybody who has a basic understanding of REST API can instantly make use of it to put together necessary REST request parameters and test them. It also keeps track of what you have already typed, and allows making a collection to reuse. You can download from chrome web store. You can find a list of features including recording and documentation from the Postman site. There is also a good review on programmable web site, which summarizes the feature nicely. Below shows a sample image from Postman UI.



A sample image from Postman UI. You can simply specify URL, method and parameters to make a REST call. A response can be formatted for easy viewing.

PAW:

Paw is a full-featured designed Mac app that makes interaction with REST services much easier. Whether you are an API maker or consumer, Paw helps you build HTTP requests, inspect the server's response and even generate client code. Using PAW, you can build HTTP Requests that will be ready to copy/paste into your code. They support most of the popular code languages; you can even build your own custom code generator via Extensions. It allows you to Access data from previous responses, so you can send back whatever the server returned. Compute hashes, authentication tokens, signatures, randomize data for testing. Paw comes with a large set of Dynamic Values, and you can write your own in JavaScript via Extensions.



PAW request with JSON Text Result and JS code generator.

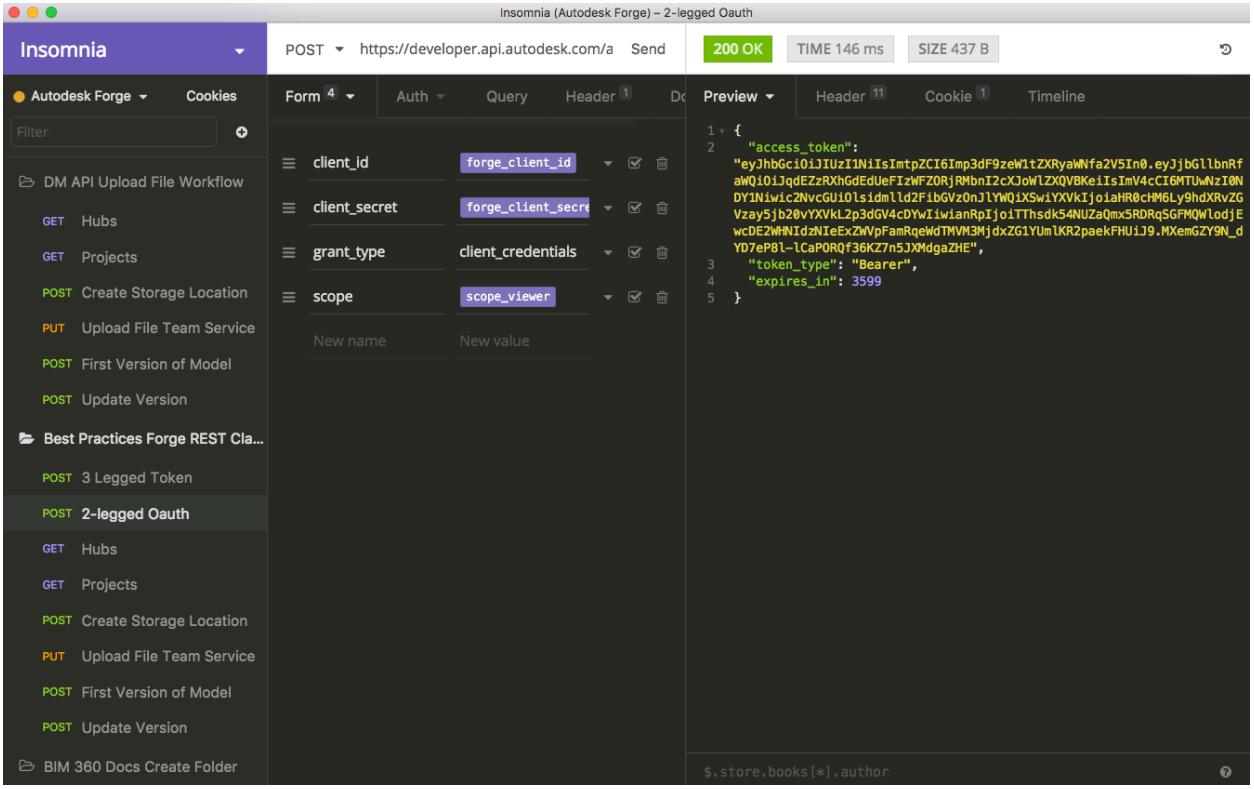
Insomnia:

Insomnia is collaborative, free, open source, and cross-platform – making it the perfect companion for both individuals and teams It was created in 2016 and just recently joined the open source community with more than 30,000 active monthly users. Why I like this app, well it is very simple to use, it gets you started fast and you can create any workflow to use your API's in a matter of minutes. Some of the functionalities that make this app outstanding are the following:

Create HTTP requests: Specify URL, payload, headers, and authorization all in one place. Then just hit send.

View entire transaction: Get all the details on every response. View status code, body, headers, cookies, and more!

Organize everything: Create workspaces or folders, drag-and-drop requests, and easily import and export your data.



A sample image from Insomnia REST UI. You can simply paste your cURL commands in the address bar to create the method and parameters to make a REST call. A response can be formatted for easy viewing

cURL your way through Forge with prettified JSON responses

For quite some time I've been playing with cURL and the Forge Platform, using Terminal, I found a quicker and simpler way for myself to use different workflows when using the REST API from the different Forge Web Services. From authentication, creation of projects, upload of files to OSS, Team Services folder and BIM 360 Folders, request model translation, access to Hubs and add new files to specific projects that I'm part of. Everything from the Terminal using cURL. But I know what you can be thinking "*cURL from terminal, is so unorganized, so hard to read, so easy to mess up.*" I could not agree more with some of these thoughts. When your response has more than 1 line of JSON data returned, I agree it can be messy and hard to read.

```
Jaimes-MacBook-Pro:modelderivative-nodejs-tutorial jaimerosales-adsk$ cur
l --request POST   --url https://developer.api.autodesk.com/authenticatio
n/v1/authenticate   --header 'content-type: application/x-www-form-urlenc
oded'   --cookie PF=6Vt8Lwv9sjCasZYNKUMtYK   --data 'client_id=NGzuS2kNHV
ON4wHtnubZF8CKgDu2K2Ma&client_secret=WArVzdjoWvxdttCP&grant_type=client_c
redentials&scope=data%3Aread%20data%3Awrite%20data%3Acreate%20bucket%3Are
ad%20bucket%3Acreate'
{"access_token":"eyJhbGciOiJIUzI1NiIsImtpZCI6Imp3dF9zeW1tZXRyaWNfa2V5In0.
eyJjbGllbnRfaWQiOiJOR3p1UzJrTkhWT040d0h0bnViWkY4Q0tnRHUySzJNYSIsImV4cCI6M
TUwNzMyMjk5Niwic2NvcGUiOlsiZGF0YTpyZWFkIiwiZGF0YTp3cml0ZSIsImRhdGE6Y3JlYX
RlIiwiYnVja2V0OnJlYWQiLCJidWNrZXQ6Y3JlYXRlIl0sImF1ZCI6Imh0dHBzOi8vYXV0b2R
lc2suY29tL2F1ZC9qd3RleHA2MCIsImp0aSI6InZaTlVCYkR0aHdRRFpFR0JiS205UjluZnUz
U2RDbktqRkszZU1Mdkh1YnNPOHJsR1RpUTFLSld6NHNjckJLY0gifQ.3jK6anpINm2ZJaimes
JaimesJaimes-MacBook-Pro:modelderivative-nodejs-tutorial jaimerosales-ads
Jaimes-MacBook-Pro:modelderivative-nodejs-tutorial jaimerosales-adsk$
```

I know I mentioned about REST Apps such as Insomnia, Paw or Postman, since the fear of messing up one character or one extra space in your cURL can give you problems and at the

same time, you get a better organized JSON result. But bear with me, since another of my great practices to interact with REST API's I found is using something called JQ.

What is JQ?

JQ is a lightweight and flexible command-line JSON processor. A JQ program is a "filter": it takes an input, and produces an output. There are a lot of built-in filters for extracting a particular field of an object, or converting a number to a string, or various other standard tasks. It lets you visualize the JSON response in an organized way and it becomes easier to read when using terminal. How to use it? It requires a basic installation, JQ can be download for different OS platforms from https://stedolan.github.io/jq/

After installation has been performed you should restart your Terminal and you would be ready for testing. You can check your version of JQ by simply typing "jq --version" in your terminal, which will assure you JQ was successfully installed.

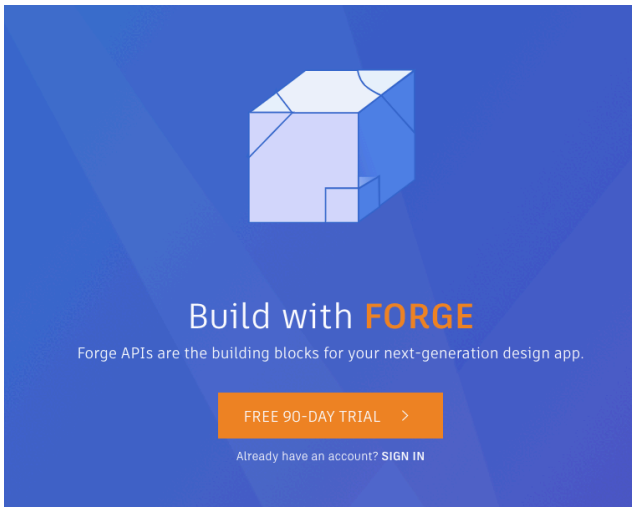
Let's look at how the previous cURL actions to obtain a 2-legged access token and create a bucket looks after using JQ from Terminal.

```
Jaimes-MacBook-Pro:modelderivative-nodejs-tutorial jaimerosales-adsk$ curl --request POST
   --url https://developer.api.autodesk.com/authentication/v1/authenticate   --header 'co
ntent-type: application/x-www-form-urlencoded'   --cookie PF=6Vt8Lwv9sjCasZYNKUMtYK   --d
ata 'client_id=NGzuS2kNHVON4wHtnubZF8CKgDu2K2Ma&client_secret=WArVzdjoWvxdttCP&grant_type
=client_credentials&scope=data%3Aread%20data%3Awrite%20data%3Acreate%20bucket%3Aread%20bu
cket%3Acreate' | jq '.'
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
  0     0    0     0    0     0      0      0 --:--:-- --:--:-- --:--:-- 100   693  100
 507  100   186   2813   1032 --:--:-- --:--:-- --:--:--  2816
{
  "access_token": "eyJhbGciOiJIUzI1NiIsImtpZCI6Imp3dF9zeW1tZXRyaWNfa2V5In0.eyJjbGllbnRfaW
QiOiJOR3p1UzJrTkhWT040d0h0bnViWkY4Q0tnRHUySzJNYSIsImV4cCI6MTUwNzMyMzAyNSwic2NvcGUiOlsiZGF
0YTpyZWFkIiwiZGF0YTp3cml0ZSIsImRhdGE6Y3JlYXRlIiwiYnVja2V0OnJlYWQiLCJidWNrZXQ6Y3JlYXRlIl0s
ImF1ZCI6Imh0dHBzOi8vYXV0b2Rlc2suY29tL2F1ZC9qd3RleHA2MCIsImp0aSI6Ik1YUm5ZeEpIczBIdDdhdXNQM
0J0b2R0a3p6dGZPRDhjRFdabG1tM01PaXRUOG9sQTg1MnlZWjRlaXJrSUl5am4ifQ.WG55UX0K3HWMw1u-50koEaR
WlOTnoiCBHQ2_zEg-WfE",
  "token_type": "Bearer",
  "expires_in": 3599
}
Jaimes-MacBook-Pro:modelderivative-nodejs-tutorial jaimerosales-adsk$
```

As we can see JQ structures and color codes the result JSON from our REST call using cURL.

Learn useful resources for Forge server-side API focus

First place to start will be our developer documents and Blog that can be found here:
https://forge.autodesk.com/

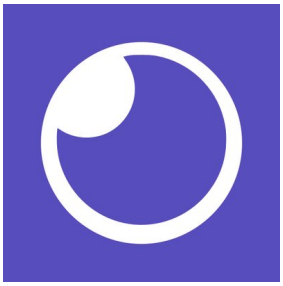


Different Tools we use when using REST API's

Postman
https://www.getpostman.com/

PAW
https://paw.cloud/

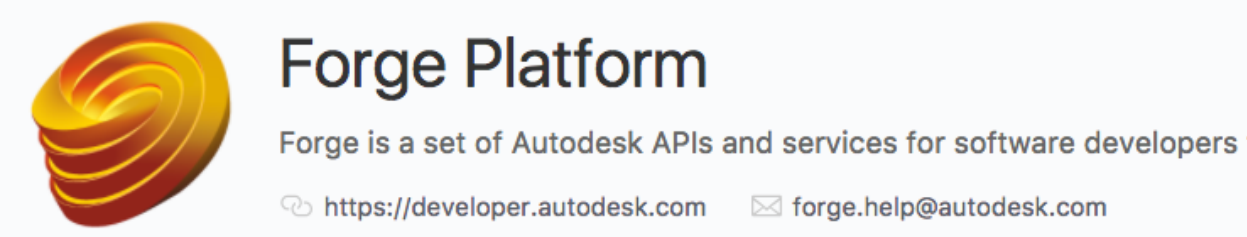
Insomnia
https://insomnia.rest/



JQ
https://stedolan.github.io/jq/

Samples and Support

Autodesk Forge Github Repo
https://github.com/Autodesk-Forge



Stack Overflow Autodesk Forge
https://stackoverflow.com/questions/tagged/autodesk-forge