FORGE DevCon 2017

FDC124079

Integrating Forge Data Management API with Other Storage Providers

Augusto Goncalves
Autodesk, Forge Partner Development

Learning Objectives

- Understand how files are stored with Forge
- Learn steps to download, upload, and interact with files via the Data Management API Learn about integration with other web-services providers
- Review storage providers' common practices (Box, Google Drive, Dropbox)
- Learn how to transfer files according to provider rules

Description

Data is spread over different providers, so you often need to around to take advantage of each web service. Using the Forge Data Management API you can access the storage structure, download and upload files from A360 Team, BIM 360 Docs and Fusion Team. And this allows integration with other providers using apps and programming. This class will cover the basics to create your own storage provider integration and show samples with Box, Google Drive and Dropbox.

Your Forge DevCon Expert(s)

Augusto Goncalves has been an API evangelist at Autodesk, Inc., since 2008. He works with all sorts of technologies, from classic desktop to modern mobile and web platforms, including .NET for AutoCAD software and Revit software, and JavaScript application programming interface for NodeJS.

AUTODESK UNIVERSITY

Introduction

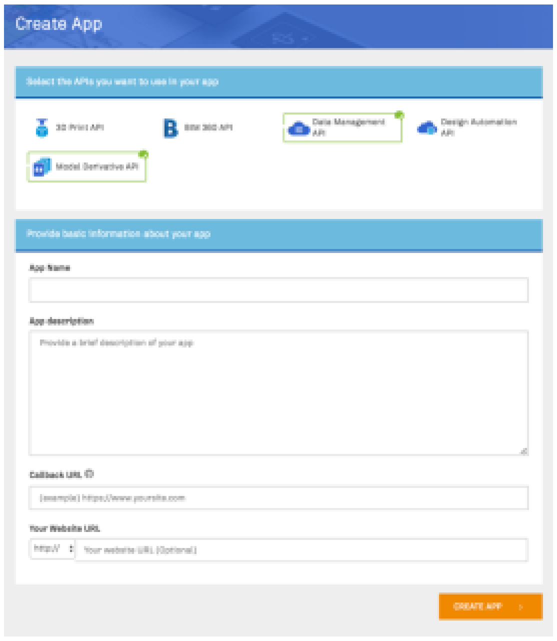
This class will present how Autodesk stores files under user hubs and projects. This idea will be used to access file and respective storage locations, which is then used to move files from Autodesk to other places, and vice-versa. Being a heavy-duty task, the last section will show the benefits of implementing this using server-less approach.

Forge Data Management Structure

A company will not reset all files and backups, it doesn't make sense to lose or start from zero every time a new storage appears, so it's important to be able to migrate data, or move it around when needed. This class will focus on end-user files stored on Autodesk, including Fusion Team, BIM 360 Team (formerly A360) and BIM 360 Docs, and how to move it to and from other providers.

<u>First things first: create a Forge Account</u>

First, you'll need to create an account at the Developer Portal (http://developer.autodesk.com) and create an app. At this page, shown at the image on below, make sure to select the APIs your app will use. The *Callback URL* is used for 3-legged OAuth (when the data is stored under the end-user account), but is not used for 2-legged OAuth (when the data is stored on the application bucket). We'll come back to the differences between these 2 models later.


Create App

Select the APIs you want to use in your app

3D Print API
BIM 360 API
Data Management API [✓]
Design Automation API
Model Derivative API [✓]

Provide basic information about your app

App Name

App description

Provide a brief description of your app

Callback URL

(example) https://www.yoursite.com

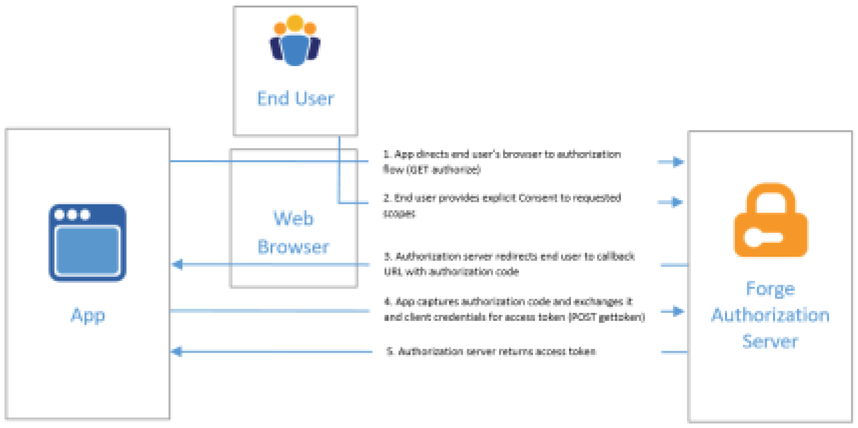
Your Website URL

http:// Your website URL (Optional)

CREATE APP WEB PAGE

<u>Access scenario</u>

As the files belong to the user, in most transferring cases, your application will need to implement the 3-legged OAuth workflow. This gives access to user files via his/her login and password on Autodesk. The image below describes the OAuth 3-legged flow, which is an industry standard.

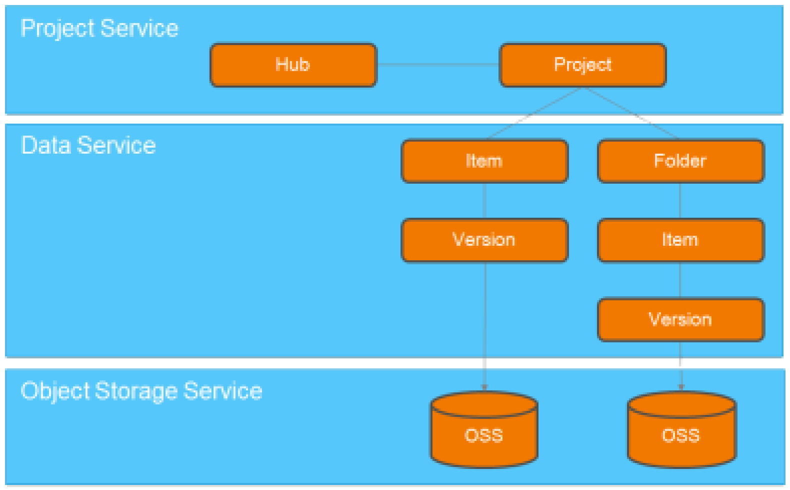

End User
Web Browser
App
Forge Authorization Server
1. App directs end user's browser to authorization flow (GET authorize)
2. End user provides explicit Consent to requested scopes
3. Authorization server redirects end user to callback URL with authorization code
4. App captures authorization code and exchanges it and client credentials for access token (POST gettoken)
5. Authorization server returns access token

OAUTH 3-LEGGED WORKFLOW

<u>Data Structure</u>

Ultimately everything is stored under OSS (**Object Storage Service**), but would without an organization would be really hard to locate a specific file or resource. That's why the Data Management API introduces the Hubs, Projects, Folders, Items and Versions.

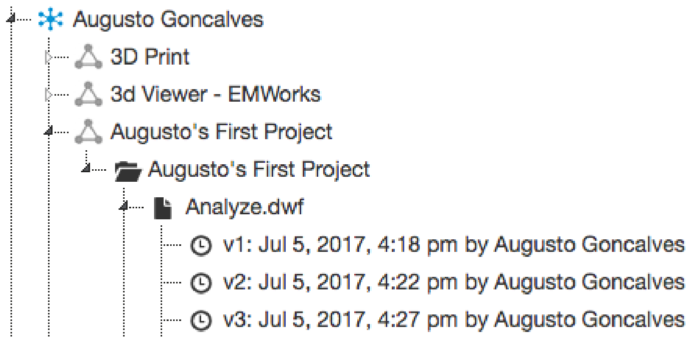
Each end-user account has, by default, at least his own hub named with the same end-user name. Inside this hub, the end-user can create projects. To access this, the Data Management API have the **Project Service** end-points for hubs and projects. As end-users can create new hubs and share team (e.g. for team work), an account can have multiple hubs. Next, inside each project, an end-user can create items (i.e. a file) with multiple versions. Every item will have, at least, the first version. The project has a root folder, by default, and this can have as many subfolders as needed, in order to organize it better. This is accessible by **Data Service** end-points. The image below shows a schematic version of this data structure.



HUBS & PROJECTS ABSTRACTION ON TOP OF OSS

The actual file is not stored under this structure, but on a storage location on **Object Storage Service** (OSS). The item version has a storage attribute that defines this location, used to upload or download files.

As we are used to a folder tree structure, the image below shows a Hub, then the respective Projects. A project will have 1 or more *Top Folders*, which for BIM 360 Team will (most likely) have the same name as the project, while for BIM 360 Docs it will different. Each folder (including Top) will have items with 1 or more versions. The image shows an item (file) with multiple versions.


Augusto Goncalves
3D Print
3d Viewer - EMWorks
Augusto's First Project
Augusto's First Project
Analyze.dwf
v1: Jul 5, 2017, 4:18 pm by Augusto Goncalves
v2: Jul 5, 2017, 4:22 pm by Augusto Goncalves
v3: Jul 5, 2017, 4:27 pm by Augusto Goncalves

FOLDER TREE STRUCTURE

Download & Upload of files

For this text, a **Storage** means somewhere else where files are located. It can be an online storage, like Box, Dropbox, Egnyte, Google Drive, OneDrive, along many other. And can also be an “offline” storage that resides in an intranet, like Vault or a simple local HD. Regardless, to transfer between 2 locations we need to define a download source and an upload destination.

Transferring from Autodesk to Storage

First the user needs to select a file or folder to transfer. As shown before, drill down from Hubs, Projects, Folders, Items and, finally, Versions, which contains the storage location. If transferring an entire folder, the app will need to get all items and respective versions (e.g. last version of each item).

Now to define the destination the application should allow the user to select a folder. Depending on the **Storage** it may need adjustments, like creating a file “place holder” before actually writing the file. For instance, Google Drive requires to first create a file location, then PUT the actual bytes there (i.e. upload the file). Dropbox, for instance, receive the file with the name as a header and create the location for it on the fly.

Transferring from Storage to Autodesk

Similar to the previous, the user needs to select a file or folder. Some storages will have just the file without versions, like Google Drive, others have versioning of files, like Box. So, the UX needs to allow that selection. To upload a file (or a folder with files) to Autodesk an application first needs a UI that allow to select the destination. On the destination, create a storage for the file, upload the file to this destination, then create a first version (or a new version, if that's the case).

Storage providers

Just like for Autodesk Forge, in order to use storage APIs you need to create a developer account and a user account for testing. There are many storage providers, the list below includes 5 of then, for your reference, with links to the respective Developer portals. This class presentation will demo a sample with these storages.

Developer pages

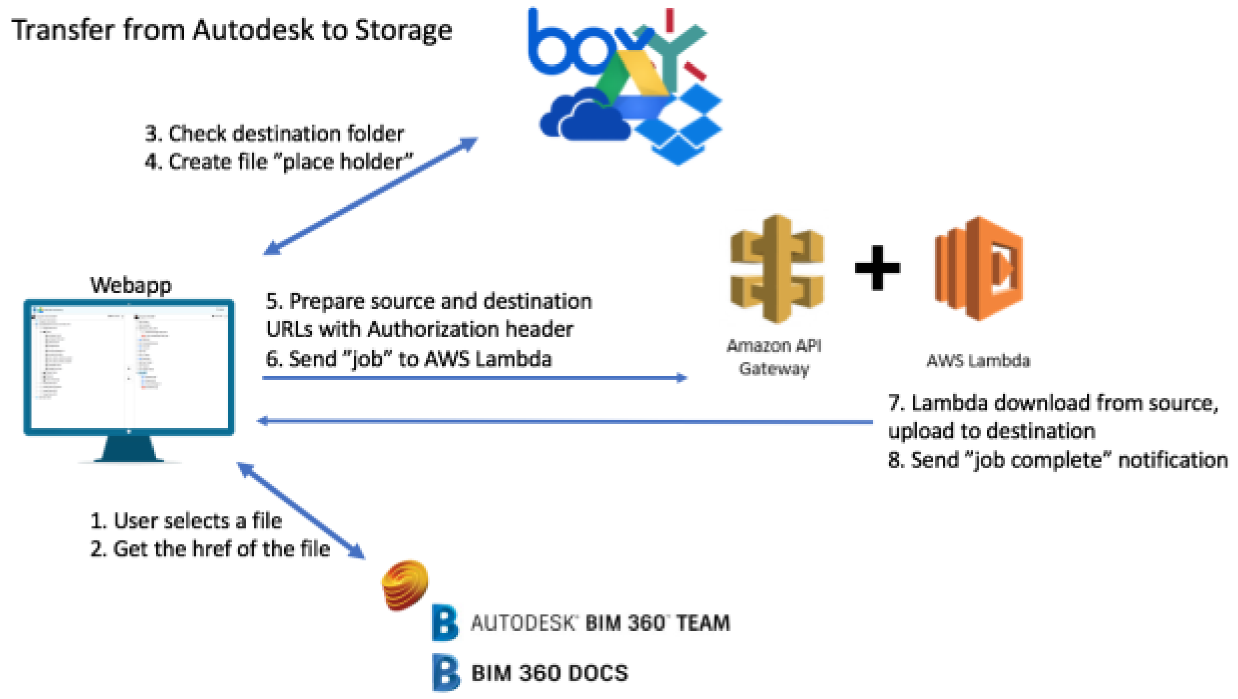
• Box	https://developer.box.com/
• Dropbox	https://www.dropbox.com/developers
• Egnyte	https://developers.egnyte.com/
• Google Drive	https://console.developers.google.com/
• OneDrive	https://dev.onedrive.com/app-registration.htm

Transferring files

In case both source files and destination location are online, an application will need to download from source and upload to destination. A server implementing this code will have a big bandwith traffic during the transfer, but will not use that power the rest of the time as the UI is just showing list of folders and files from Autodesk and other storage.

You may decide to implement the entire code into a single server, but a server-less approach may reduce cost. There are different provides, like Microsoft Azure and Google Cloud. For this sample, I'm using AWS Lambda, which will only allocate the server when needed (i.e. when transferring the files), turning off the rest of the time (meaning no cost when not in use). There are other benefits, like scalability, so review the respective AWS documentation for details. An AWS API Gateway is required to invoke this server-less code on Lambda.

The image below shows a step-by-step to transfer from **Autodesk** to a **Storage** using AWS Lambda. After using OAuth to authorize on both sides (Autodesk and Storage), let the user select the file on Autodesk, selected the folder at destination storage, create the source and destination requests headers and delegate the actual transfer task to Lambda. This will callback when finish.


Transfer from Autodesk to Storage
3. Check destination folder
4. Create file "place holder"
Webapp
5. Prepare source and destination URLs with Authorization header
6. Send "job" to AWS Lambda
Amazon API Gateway
AWS Lambda
7. Lambda download from source, upload to destination
8. Send "job complete" notification
1. User selects a file
2. Get the href of the file
AUTODESK BIM 360 TEAM
BIM 360 DOCS
TRANSFER FROM AUTODESK TO STORAGE

As mentioned, the request is a simple collection of URL and Headers. As on Autodesk the file is keep on a storage, the source will be something like the following (in NodeJS JavaScript):

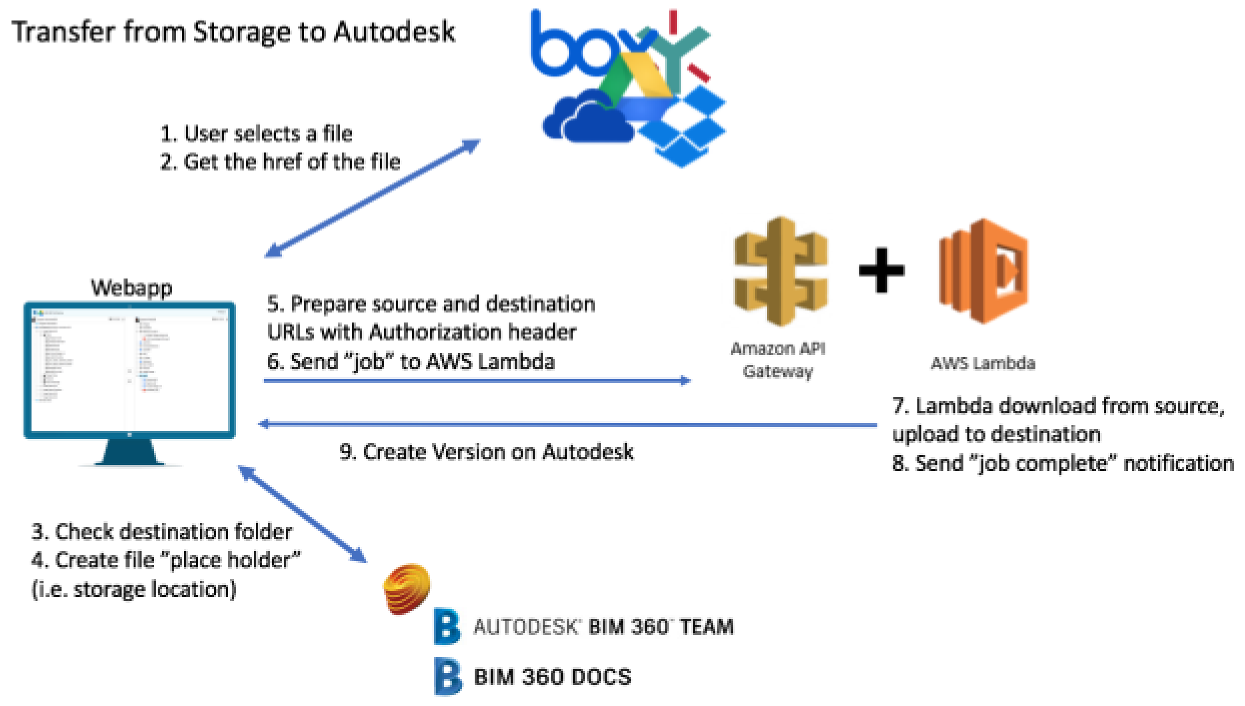
```
var source = {
  url: version.relationships.storage.meta.link.href,
  method: "GET",
  headers: {
    'Authorization': 'Bearer ' + autodeskBearerToken
  },
  encoding: null
};
```

And the destination for Google Drive (as an example) would be like:

```
var destination = {
  url: 'https://www.googleapis.com/upload/drive/v2/files/' + googleFileId + '?uploadType=media',
  method: 'PUT',
  headers: {
    'Content-Type': newFileMimeType,
    'Authorization': 'Bearer ' + googleBearerToken
  }
};
```

This sample was implemented in NodeJS on server and client, and also used on AWS Lambda due its short cold start (i.e. the time to launch the application for the first time). This is important on server-less that will perform many small operations, helping reduce the overall execution time.

The inverse route, from **Storage** to **Autodesk**, is quite similar, except the step 9: after upload a file to Autodesk the application also needs to create a version for it.


Transfer from Storage to Autodesk
1. User selects a file
2. Get the href of the file
Webapp
5. Prepare source and destination URLs with Authorization header
6. Send "job" to AWS Lambda
Amazon API Gateway
AWS Lambda
7. Lambda download from source, upload to destination
8. Send "job complete" notification
9. Create Version on Autodesk
3. Check destination folder
4. Create file "place holder" (i.e. storage location)
AUTODESK BIM 360 TEAM
BIM 360 DOCS

TRANSFER FROM STORAGE TO AUTODESK

Samples

The main sample was written in NodeJS and available on Github. See also the live instances: Box, Dropbox, Egnyte, Google Drive, OneDrive.

Conclusion

Migrating data is an important to onboard on systems. Autodesk customers will have data spread on different storages and, now, may consider moving to a centralized location on BIM 360. And that's why this class is important.

Transferring a file is, actually, a download and upload operation. This requires a server in the middle that will handle the actual transfer. As this is a heavy-duty task, server less approach can help reduce data-center costs. This is something to keep in mind.

Finally, thank you for attending this session! I hope you found the class enjoyable and valuable.