AUTODESK UNIVERSITY

SD22363

Free your design data

Adam Nagy
Autodesk

Learning Objectives

- Know what services are available as part of Forge
- Know how to access files on OSS and A360
- Know how Data Management and Model Derivative services can work together
- Know how to get information from A360 files through the Model Derivative service

Description

Is this the end of the proprietary file format? The Forge Model Derivative API allows you to easily translate your design files to different formats and also extract data from those files for you to use almost anywhere. In this class we will introduce these APIs by walking you through a series of sample code demonstrations.

Your AU Expert

Adam Nagy joined Autodesk back in 2005 and has been providing programming support, consulting, training and evangelism to external developers ever since. He started his career in Budapest working for a Civil Engineering CAD software company, then worked for Autodesk in Prague for 3 years and now lives in South England, UK.
Currently focusing on the manufacturing product API's and Forge API's.
Adam has a degree in Software Engineering and has been working in that area since even before leaving college.

Know what services are available as part of Forge

Evolution of design access

Back in the 80's when people wanted to share data with each other they ran into a problem. Each computer was different, was using different programs, and so was storing data in different ways. Fortunately, the **World Wide Web** has been invented which standardized how data was represented and made it possible for each computer to make it readable for the users.

For a long time we had a similar issue with design files. Each design program is using its own file format making it difficult to share data with other people using other **CAD** programs.

DESIGN FILE FORMATS

You need to install the design software

First in order to view a given design file you needed to install the actual design software that crated the file. That could be an issue because you would need to buy the software.

You need to install "the" viewer

Realizing the problem with the previous approach, companies created free Viewer applications for their own files formats. This was better but it still required people to install an application and so it did not work on all computers and devices where people might want to be able to view your design from.

Just use Viewer from a web browser

Now we have a solution that will enable people to share their design ideas with people anywhere in the world – all they need is a web browser. Using our technology, you can get many different design file formats into a lightweight format that can be streamed into our Viewer component running inside a web browser. See how below.

Forge API's

The **Forge API's** are a set of **RESTful** API's covering multiple components that can work together like **LEGO®** pieces. Currently these include the following:

Authentication
Let's you authenticate your application with Forge, so that you can use the various other services

Data Management API
Let's you access and store data on **A360** & **BIM 360 Docs**, or in your application's own private bucket on **OSS** (Object Storage System)

Model Derivative API
Enables you to get data out of any of the supported design file formats. This data includes geometry, hierarchical information and properties of all the components and subcomponents inside the model

Design Automation API
While the **Model Derivative API** only provides read-only access to the original design files, the **Design Automation API** enables you to modify the supported file types. At the moment it only supports **AutoCAD** files.

If you go on the **Forge Developer** site (https://developer.autodesk.com/), then you'll find further information about these **API's** and **Step-by-step Tutorials** that show you how exactly to use these **API's** to achieve various tasks.

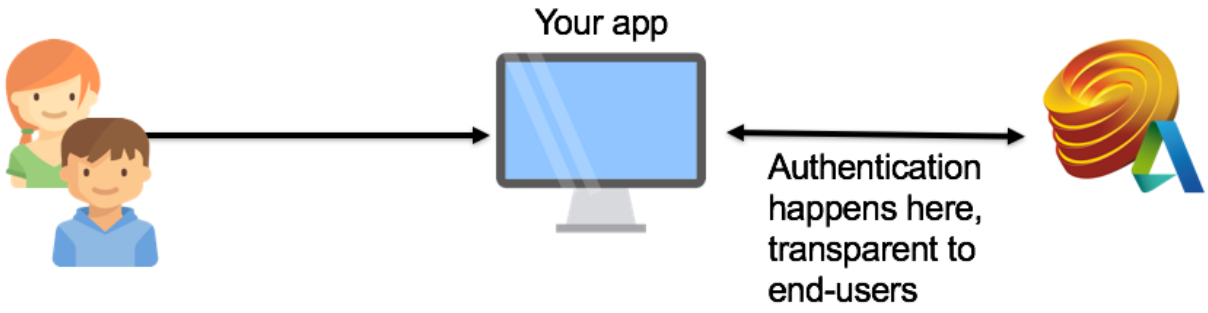
Know how to access files on OSS and A360

A360 is the data storage system available to Autodesk users, and **OSS** is the **Object Storage System** which is available to **Forge App's** for storing data on the cloud.

Authentication

Whichever service you need, first of all you'll have to authenticate yourself. That's what the **Authentication** endpoints are for.

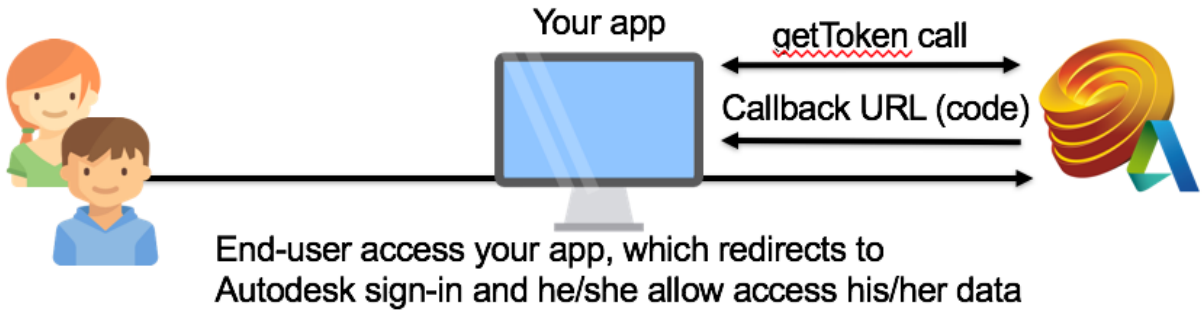
Depending on where the data you are trying to access is, you need to use either **2 legged** or **3 legged** authentication.



2 LEGGED AUTHENTICATION

2 legged authentication

In case of two legged authentication the two legs are **your app** and the **Forge service**. This is what you use if you want to store the design files on **OSS** (Object Storage System). In this case your application creates its own private buckets on our servers and uses that to store design files



3 LEGGED AUTHENTICATION

3 legged authentication

In case of three legged authentication the three legs are **your app**, the **Forge service** and **the user** whose data your app wants to access. This is what you use if you want to access a user's files stored on **A360**. In this case you need the approval of the user, who will be the third leg of this operation.

One thing you also have to be aware of is **scopes**. This is an extra layer of security which restricts the use of the access token to the type of operations you want. The current list is:

user-profile:read	View your profile info	The application will be able to read the end user's profile data.
data:read	View your data	The application will be able to read the end user's data within the Autodesk ecosystem.
data:write	Manage your data	The application will be able to create, update, and delete data on behalf of the end user within the Autodesk ecosystem.
data:create	Write data	The application will be able to create data on behalf of the end user within the Autodesk ecosystem.
data:search	Search across your data	The application will be able to search the end user's data within the Autodesk ecosystem.
bucket:create	Create new buckets	The application will be able to create an OSS bucket it will own.
bucket:read	View your buckets	The application will be able to read the metadata and list contents for OSS buckets that it has access to.
bucket:update	Update your buckets	The application will be able to set permissions and entitlements for OSS buckets that it has permission to modify.
bucket:delete	Delete your buckets	The application will be able to delete a bucket that it has permission to delete.
code:all	Author or execute your codes	The application will be able to author and execute code on behalf of the end user (e.g., scripts processed by the Design Automation API).
account:read	View your product and service accounts	For Product APIs, the application will be able to read the account data the end user has entitlements to.
account:write		

The simpler authentication is the **2 legged authentication** where you can log into the system with a single call. Here is a **cURL** example of that:

```
curl -v 'https://developer.api.autodesk.com/authentication/v1/authenticate'
  -X 'POST'
  -H 'Content-Type: application/x-www-form-urlencoded'
  -d '
```

```
  client_id=obQDn8P0GanGFQha4ngKKVWcxwyvFAGE&
  client_secret=eUruM8HRyc7BAQ1e&
  grant_type=client_credentials&
  scope=data:read
'
```

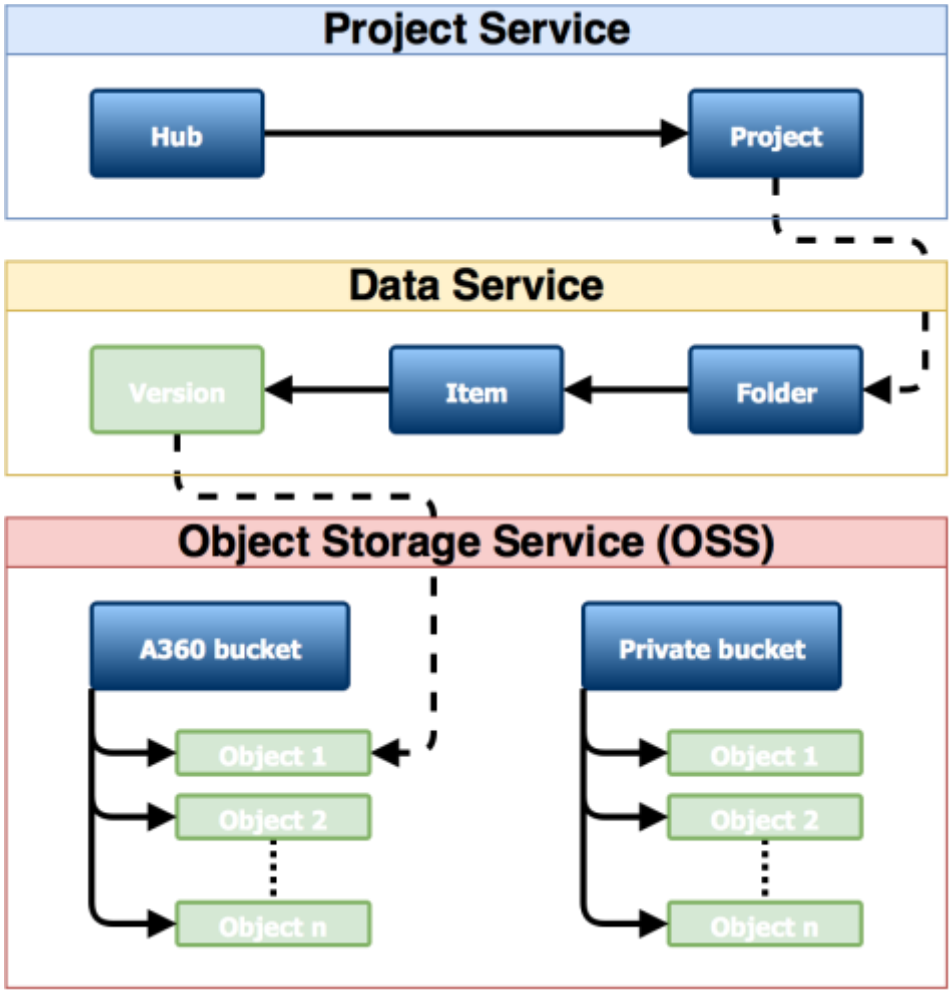
This call will reply with something like this which will include an **access token**:

```
{
  "token_type": "Bearer",
  "expires_in": 1799,
  "access_token": "Ff387cyQXw1elOT3nldiFiKBqDOs"
}
```

Now that you have your **access token**, you can start using the other **Forge API's**.

Data Management API

Once you are authenticated you can use the **Data Management API** to access a file whether it's in your app's **private bucket** on **OSS** or on **A360**. The **Project** and **Data Services** provide the means to organize the data, while the **Object Storage Service** provides the place where the files can be stored.



FIND A FILE ON OSS OR A360

Data Management API endpoints

Here is the list of the most frequently used endpoints on **https://developer.api.autodesk.com/**

/projects/

- Hubs /project/v1/hubs
- Projects /project/v1/hubs/:hub_id/projects

/data/

- Folders /data/v1/projects/:project_id/folders/:folder_id/contents
- Items /data/v1/projects/:project_id/items/:item_id
- Version /data/v1/projects/:project_id/versions/:version_id

/oss/

- Buckets /oss/v2/buckets *(POST to create, GET to list)*
- Upload /oss/v2/buckets/:bucketKey/objects/:objectName

Using these API's you can get a list of all the **hubs** of the **user**, the **projects** in those **hubs**, the **folders**, **items** and their **versions**.

With a simple call like this you can e.g. get a list of all the **hubs** the **user** has access to:

```
curl -X GET -H "Authorization: Bearer kEnG562yz5bhE9igXf2YTcZ2bu0z"
"https://developer.api.autodesk.com/project/v1/hubs"
```

The reply will be something like this:


```
{
  "data": [
    {
      "relationships": {
        "projects": {
          "links": {
            "related": {
              "href": "/project/v1/hubs/a.1256/projects"
            }
          }
        }
      },
      "attributes": {
        "name": "my hub",
        "extension": {
          "data": {},
          "version": "1.0",
          "type": "hubs:autodesk.core:Hub",
          "schema": {
            "href": "/schema/v1/versions/hubs%3Aautodesk.core%3AHub-1.0"
          }
        }
      },
      "type": "hubs",
      "id": "a.1256",
      "links": {
        "self": {
          "href": "/project/v1/hubs/a.1256"
        }
      }
```

```
    }
  ],
  "jsonapi": {
    "version": "1.0"
  },
  "links": {
    "self": {
      "href": "/project/v1/hubs"
    }
  }
}
```


Each reply will contain the **id** of the **available objects** which you can use in your next query. E.g. now that you got the above **HubId** "**a.1256**", you could use that to get a list of all the **projects** in it:

```
curl -X GET -H "Authorization: Bearer kEnG562yz5bhE9igXf2YTcZ2bu0z"
"https://developer.api.autodesk.com/project/v1/hubs/a.1256/projects"
```

.

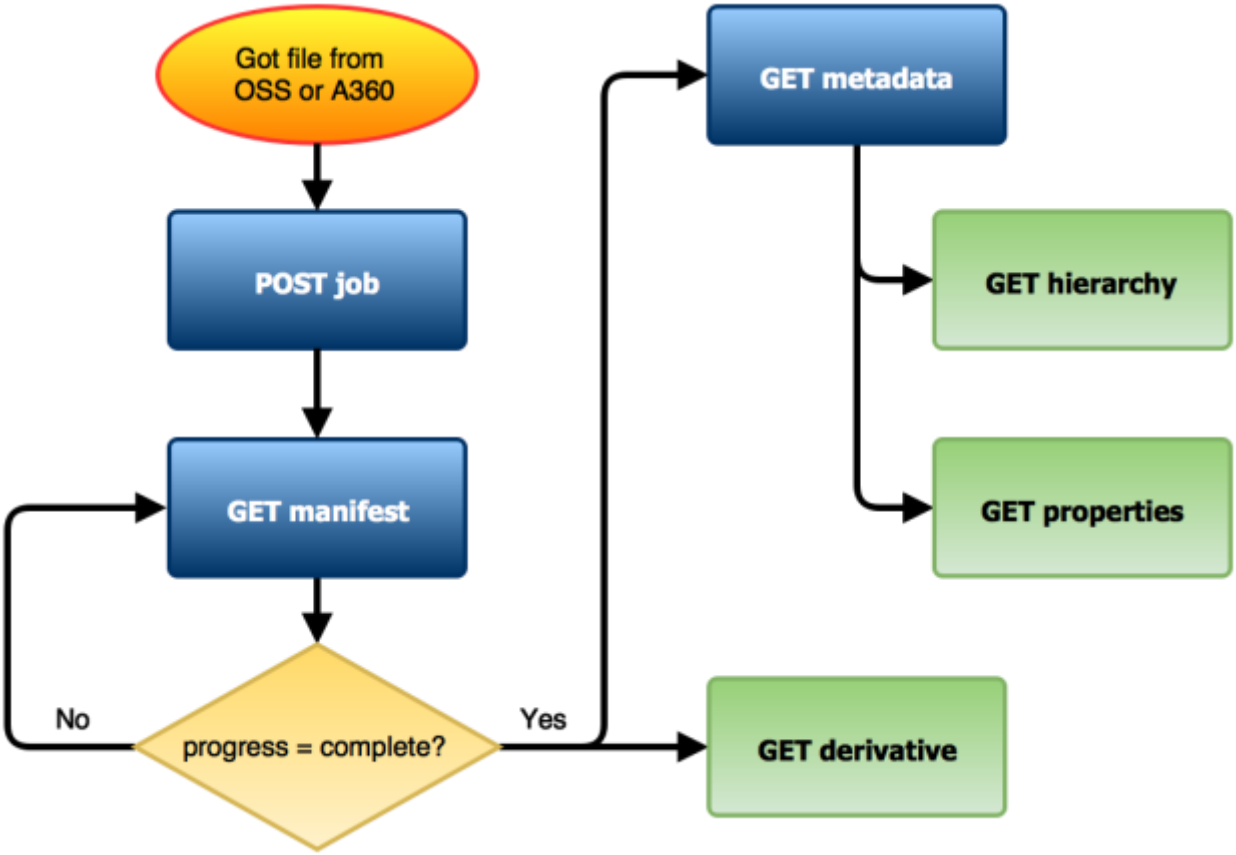
Know how Data Management and Model Derivative services can work together

Workflow

In order to use the **Model Derivative API**, you first need a file stored on either **OSS** in your own bucket or on **A360**. For that you can use the **Data Management API**. Once you have the **urn** of the file, you can **POST** a job that would translate the file to the format you need. Most of the time it would be the **SVF** format which will make it viewable inside the **Autodesk Viewer** component.

To check on the progress of the translation and get access to previous translations as well, you need to check the **manifest** file of the source file.

Once the translation is ready, the **manifest** file will contain the **urn** (file identifier) of the **derivative** file, and with that you can download the translated file.



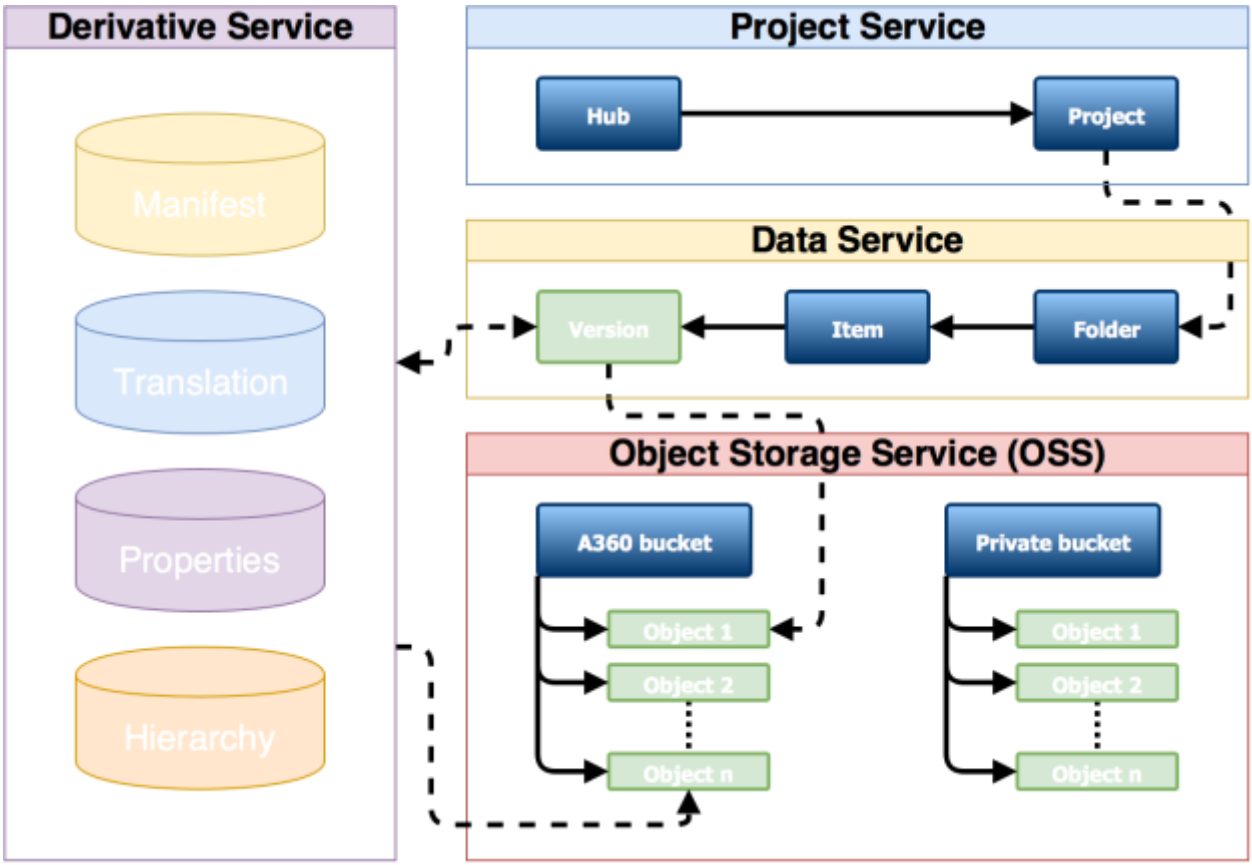
MODEL DERIVATIVE API WORKFLOW

Model Derivative API endpoints

Here is the list of endpoint which are available for programmers:

https://developer.api.autodesk.com/modelderivative/v2/designdata/ …		
GET	formats	- get table of supported translations
POST	job	- start a translation
GET	:urn/ thumbnail	- get thumbnail
GET	:urn/ manifest	- get manifest (info about translations)
DELETE	:urn/ manifest	- delete manifest
GET	:urn/ manifest/ :derivativeurn	- download a derivative (translation)
GET	:urn/ metadata	- get view GUID's
GET	:urn/ metadata/ :guid	- hierarchy
GET	:urn/ metadata/ :guid/ properties	- component properties

Translate the file



MODEL DERIVATIVE AND DATA MANAGEMENT SERVICES

The green objects in the above image show which items' **urn** can be passed to the **Model Derivative** service – could be a specific **Item Version** from **A360** or the actual **Object** from the **OSS**.

Once you got the **urn** of the file you want to translate e.g. for viewing, you can pass it to the **Model Derivative API**'s **job** endpoint:

```
curl -X 'POST' -H 'Content-Type: application/json; charset=utf-8' -H
'Authorization: Bearer Ptnrvrt SRp VWvUi 3407 Qhgvqd UVKL' -v
'https://developer.api.autodesk.com/modelderivative/v2/designdata/job' -d
'{
  "input": {
    "urn":
"dXJuOmFkc2sub2JqZWN0czpvcy5vYmpl Y3Q6b W9kZWkkZXJpdmF0
aXZl L0E1Lnppc A",
    "compressedUrn": true,
    "rootFilename": "A5.iam"
  },
  "output": {
    "formats": [
      {
        "type": "svf",
        "views": [
          "2d",
          "3d"
        ]
      }
    ]
  }
}'
```

In case of the above example we are translating an **assembly** consisting of multiple files. When checking such an item on **A360** it will have the **attributes** >> **extension** >> **type** of **"versions:autodesk.a360:CompositeDesign"**. That means that all the relevant files are combined in a **zip** file. When that is the case you have to pass **"compressedUrn": true** in the body of the **job** request and specify the root file name, e.g. **"rootFilename": "A5.iam"**

Check translation progress

Depending on the complexity of the model and file size the translation can go on from seconds to minutes. At the moment you have to keep requesting an up-to-date **manifest** in order to get an update on the **translation progress**. Later on we might provide so called **webhooks**, that would enable your app to subscribe to an event that would fire when the translation finished.

Using the same **urn** that you used for starting the translation you can use to request the manifest:

```
curl -X 'GET' -H 'Authorization: Bearer Ptnrvrt SRpWwUi3407QhgvqdUVKL'
-v
'https://developer.api.autodesk.com/modelderivative/v2/designdata/dXJuOmFkc2sub2JqZWN0czpvcy5vYmpl Y3Q6bW9kZWxkZXJpdmF0aXZl L0E1LnppcA/manifest'
```

This will produce a response body like this:

```
{
  "type": "manifest",
  "hasThumbnail": "false",
  "status": "pending",
  "progress": "0% complete",
  "region": "US",
  "urn":
"dXJuOmFkc2sub2JqZWN0czpvcy5vYmpl Y3Q6bW9kZWxkZXJpdmF0aXZ l L0E1Lnppc A",
  "derivatives": [
  ]
}
```

Know how to get information from A360 files through the Model Derivative service

Find the model view you want

In order to get information about the **model herarchy** or **component** in the model, you need to choose a **model view GUID**. You can get the list of those through the **metadata** endpoint:

```
curl -X' GET' -H' Authorization: Bearer Ptnrvrt SRpWwUi 3407QhgvqdUVKL'
-v
'https://developer.api.autodesk.com/modelderivative/v2/designdata/dXJuO
mFkc2sub2JqZWN0czpvcy5vYmpl Y3Q6bW9kZWxkZXJpdmF0aXZl L0E1L
nppcA/metadata'
```

If the model has only a single view (like e.g. **Inventor** and **Fusion** models) then you'd see something like this:

```
{
  "data": {
    "type": "metadata",
    "metadata": [
      {
        "name": "Scene",
        "guid": "4f981e94-8241-4eaf-b08b-cd337c6b8b1f"
      }
    ]
  }
}
```

Get model hierarchy

Using the **guid** of the **Model View** you can ask for the hierarchy of the model:

```
curl -X' GET' -H' Authorization: Bearer Ptnrvrt SRpWwUi 3407QhgvqdUVKL'
-v
'https://developer.api.autodesk.com/modelderivative/v2/designdata/dXJuO
mFkc2sub2JqZWN0czpvcy5vYmpl Y3Q6bW9kZWxkZXJpdmF0aXZl L0E1L
nppcA/metadata/4f981e94-8241-4eaf-b08b-cd337c6b8b1f'
```

This would provide information similar to this:


```
{
  "data": {
    "type": "objects",
    "objects": [
      {
        "objectid": 1,
        "name": "A5",
        "objects": [
          {
            "objectid": 2,
            "name": "Model",
            "objects": [
              {
                "objectid": 3,
                "name": "Bottom",
                "objects": [
                  {
                    "objectid": 4,
                    "name": "Box"
                  }
                ]
              },
              {
                "objectid": 5,
                "name": "Pillar",
                "objects": [
                  {
                    "objectid": 6,
                    "name": "Cylinder"
                  }
                ]
              },
              {
                "objectid": 7,
                "name": "Top",
                "objects": [
                  {
                    "objectid": 8,
```

```
            "name": "Box"
          }
        ]
      }
    ]
  }
 ]
}
```

Currently you'll get the hierarchy of the full model. In case of a huge and complex assembly it could be quite large. Therefore we might add a way to filter down the parts of the hierarchy you are interested in.

The **objectid** values provided here correspond to the **dbId** values that you would get through the **Viewer's JavaScript API** in **view.getProperties()** function. That makes it quite easy to synchronize data between the **Viewer** and the **Model Derivative** service.

Get component data

Getting all the component properties for the selected **Model View** is just as simple. You just need to call the **properties** endpoint:

```
curl -X' GET' -H' Authorization: Bearer Ptnrvrt SRp WwUi 3407 QhgvqdUVKL'
-v
'https://developer.api.autodesk.com/modelderivative/v2/designdata/dXJuO
mFkc2sub2JqZWN0czpvcy5vYmpl Y3Q6bW9kZWkkZXJpdmF0aXZl L0E1L
nppcA/metadata/4f981e94-8241-4eaf-b08b-cd337c6b8b1f/properties'
```

This will provide something like this:

```
{
 "data": {
  "type": "properties",
  "collection": [
   {
    "objectid": 1,
    "name": "A5",
    "properties": {
     "Name": "A5",
     "__document__": {
```

```
        "is_doc_property": 1,
        "schema_name": "atf",
        "schema_version": "5.0.0.2431"
      }
    }
  },
  {
    "objectid": 2,
    "name": "Model",
    "properties": {
      "Component Name": "Model",
      "Name": "Model",
      "Design Tracking Properties": {
        "Design State": "WorkInProgress",
        "Designer": "ADSK",
        "File Subtype": "Assembly"
      },
      "File Properties": {
        "Author": "ADSK",
        "Creation Date": "2012-Jul-09 20:18:20",
        "Original System": "Autodesk Inventor 2017",
        "Part Number": "Model"
      },
      "Mass Properties": {
        "Area": 19772.676501,
        "Volume": 83673.946256
      }
    }
  },
  {
    "objectid": 3,
    "name": "Bottom",
    "properties": {
      "Component Name": "A5-P1",
      "Name": "Bottom",
      "Design Tracking Properties": {
        "Design State": "WorkInProgress",
        "Designer": "ADSK",
        "File Subtype": "Modeling"
      },
```

```
      "File Properties": {
        "Author": "ADSK",
        "Creation Date": "2012-Jul-09 20:18:35",
        "Original System": "Autodesk Inventor 2017",
        "Part Number": "Bottom"
      },
      "Mass Properties": {
        "Area": 7000,
        "Volume": 25000
      }
    }
  },
  {
    "objectid": 4,
    "name": "Box",
    "properties": {
      "Appearance": "191, 191, 191",
      "Name": "Box"
    }
  },
  {
    "objectid": 5,
    "name": "Pillar",
    "properties": {
      "Component Name": "Pillar",
      "Name": "Pillar",
      "Design Tracking Properties": {
        "Design State": "WorkInProgress",
        "Designer": "ADSK",
        "File Subtype": "Modeling"
      },
      "File Properties": {
        "Author": "ADSK",
        "Creation Date": "2012-Jul-09 20:18:35",
        "Original System": "Autodesk Inventor 2017",
        "Part Number": "Pillar"
      },
      "Mass Properties": {
        "Area": 7000,
        "Volume": 25000
```

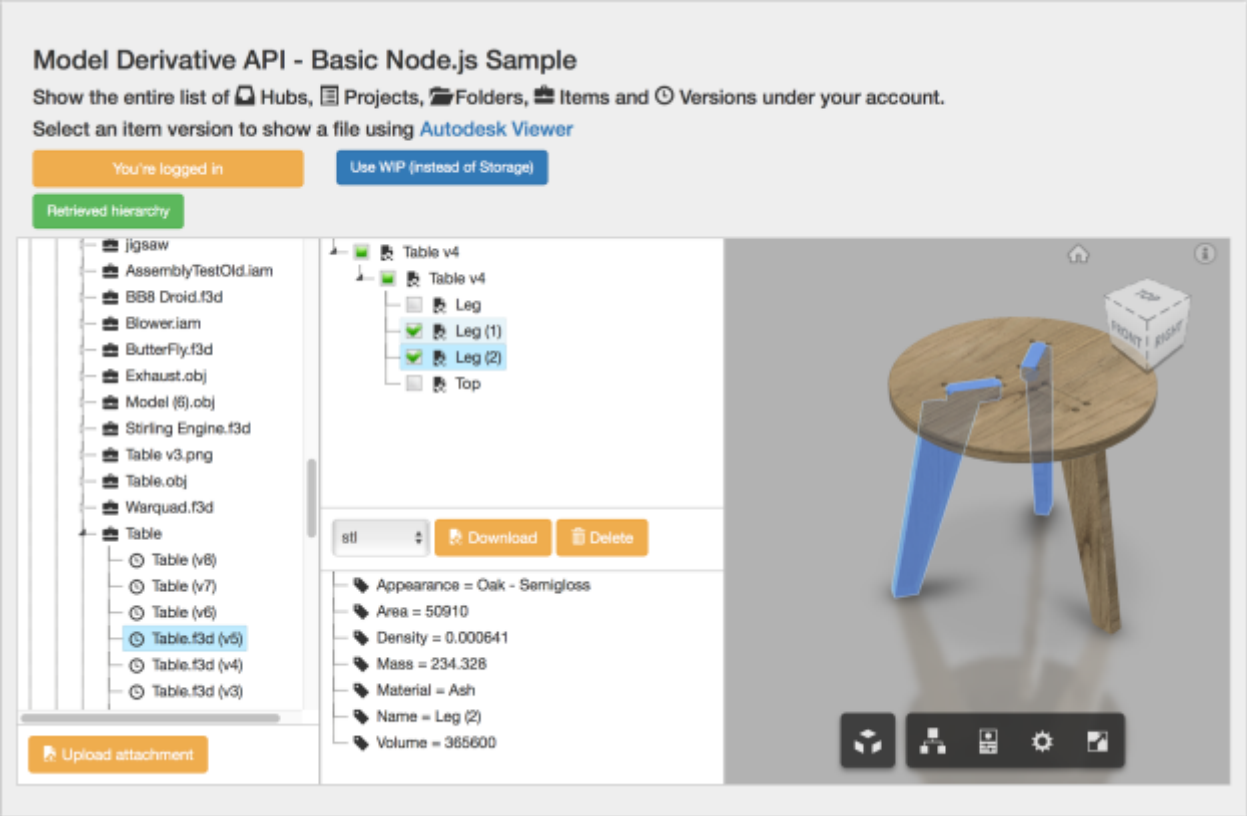
```
      }
    }
  },
  {
    "objectid": 6,
    "name": "Sylinder",
    "properties": {
      "Appearance": "191, 191, 191",
      "Name": "Sylinder"
    }
  },
  {
    "objectid": 7,
    "name": "Top",
    "properties": {
      "Component Name": "Top",
      "Name": "Top",
      "Design Tracking Properties": {
        "Design State": "WorkInProgress",
        "Designer": "ADSK",
        "File Subtype": "Modeling"
      },
      "File Properties": {
        "Author": "ADSK",
        "Creation Date": "2012-Jul-09 20:19:38",
        "Original System": "Autodesk Inventor 2017",
        "Part Number": "Top"
      },
      "Mass Properties": {
        "Area": 5772.676501,
        "Volume": 33673.946256
      }
    }
  },
  {
    "objectid": 8,
    "name": "Box",
    "properties": {
      "Appearance": "191, 191, 191",
      "Name": "Box"
```

```
      }
    }
  ]
 }
}
```

Just like in case of the hierarchy information, here as well you'll get back information for the whole model. Later on there might be a way to filter which component you are interested in and only retrieve the properties of those.

Live sample

There are multiple live samples available which show how these services can be used. One of them is https://forgemodelderivative.herokuapp.com



LIVE SAMPLE

Resources

Live sample's code
https://github.com/Developer-Autodesk/model.derivative-nodejs-sample

Forge Developer site
https://developer.autodesk.com

Sample apps
https://github.com/Developer-Autodesk/ & https://github.com/Autodesk-Forge/

Node.js packages are also available through npm package manager
https://www.npmjs.com

Forge blog
http://adndevblog.typepad.com/cloud_and_mobile/