FDC129788

Streamline Fusion 360 processes with Forge

Adam Nagy
Autodesk

Learning Objectives

- Know the capabilities of the Fusion 360 API
- Know how to get started with Forge
- See how you can use Forge services from inside Fusion 360
- See detailed examples of Forge integrations with Fusion 360

Description

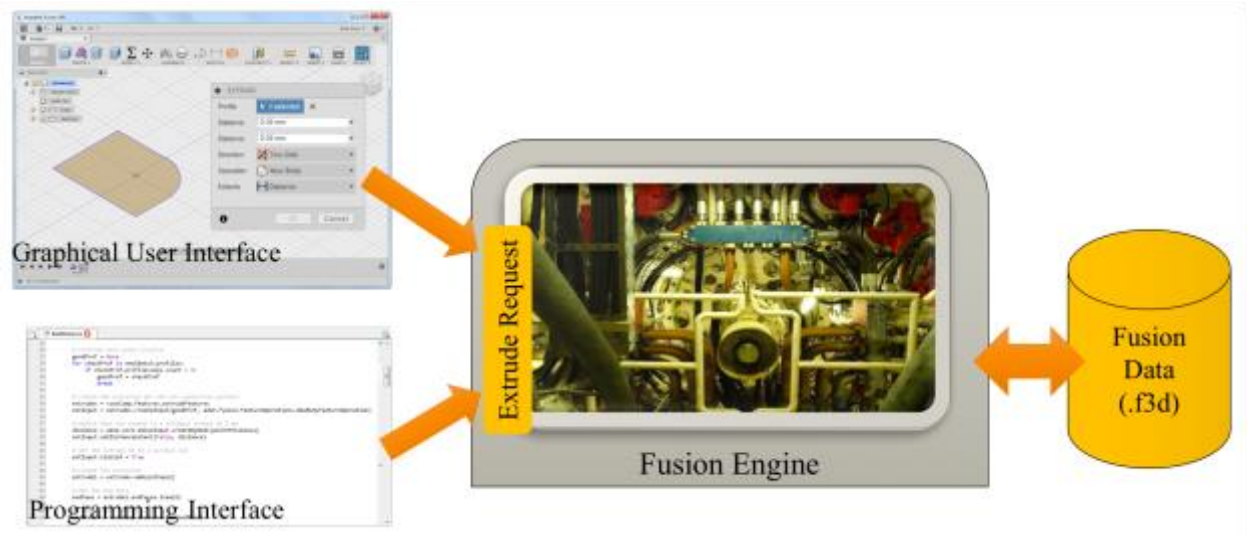
Integrating Fusion 360 with Forge can be a really powerful way to streamline your existing processes.
In this class, we will provide an introduction to both the Forge and Fusion 360 API's and will explain how you can get started with implementing workflows using both Fusion 360 and Forge.

Your Forge DevCon Expert(s)

Adam Nagy joined Autodesk back in 2005, and he has been providing programming support, consulting, training, and evangelism to external developers. He started his career in Budapest working for a civil engineering CAD software company. He then worked for Autodesk in Prague for 3 years, and he now lives in South England, United Kingdom. Twitter @AdamTheNagy

Know the capabilities of the Fusion 360 API

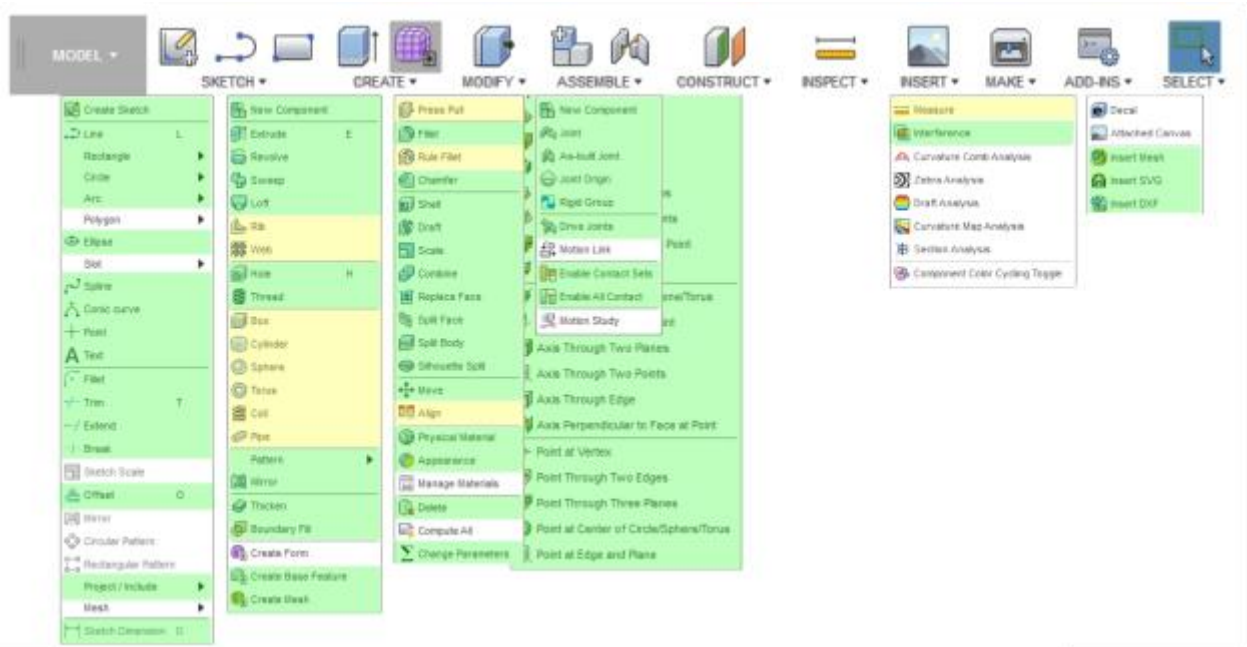
Most things inside **Fusion 360** can be automated. Here you can see how a **UI** function provides its functionality and how it could be automated with a bit of programming:

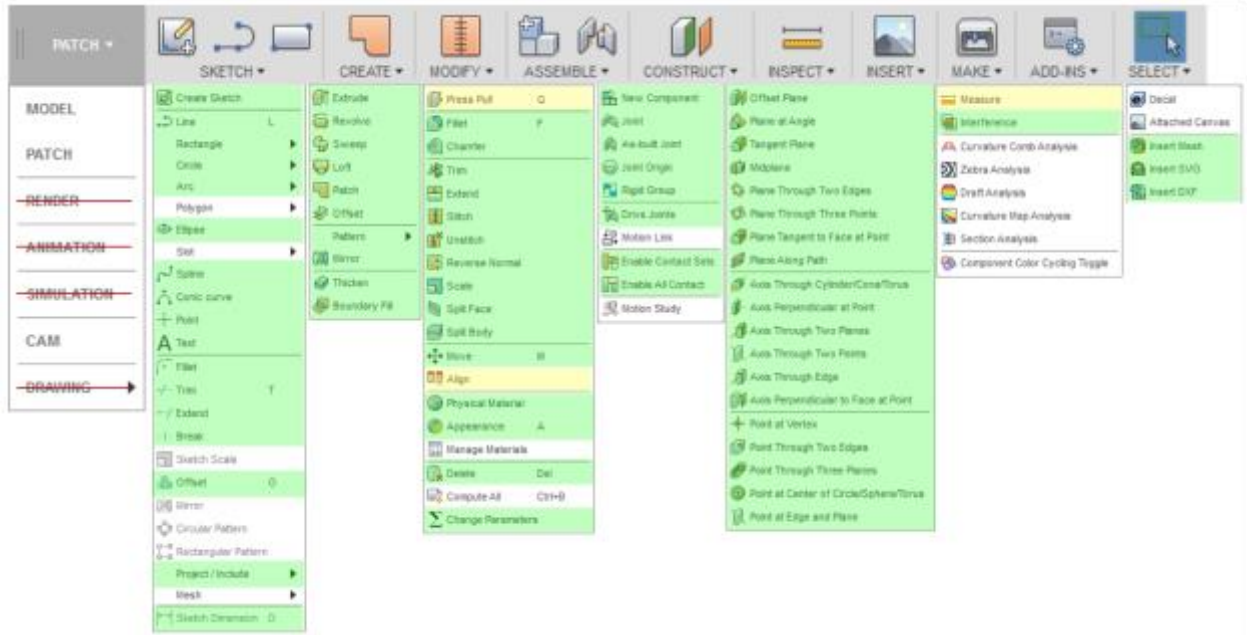


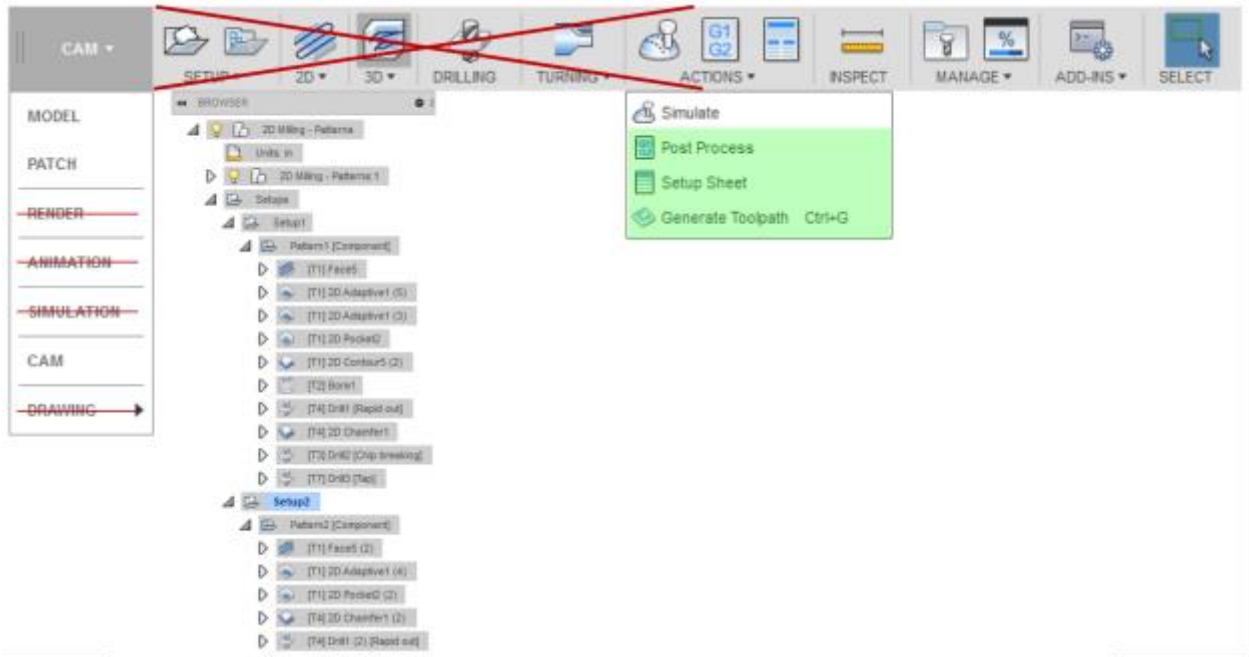
Whether it's an **internal command** or one coming from your **add-in**, they need to provider a **list of parameters** that they need. Based on that **Fusion 360** will pop up a dialog where the user can specify those parameters.

If the parameters are acceptable then **Fusion 360** will manipulate the model based on those parameters – and your **add-in** can do the same. Once it got the parameters, it can call **API** functions to manipulate the **Fusion 360** model.

So far these **UI** functions can be automated in the various **workspaces**:







There are also some **API-only** features available, like the following:

- **Attributes**

This enables developers to add **custom data** to any of the objects inside the **Fusion 360** model. They can also search for these inside the whole model, so it could also be used as a sort of marking mechanism

- **Client Graphics**

This lets developers create their own geometry inside the model view. One use of it could be to provide hints to the product user: e.g. marking components in the view or adding geometry to show additional information.

- **Listening to Events**

You could e.g. listen to command related events (command start/end) in order to react to the changes from your program

- **Progress dialog**

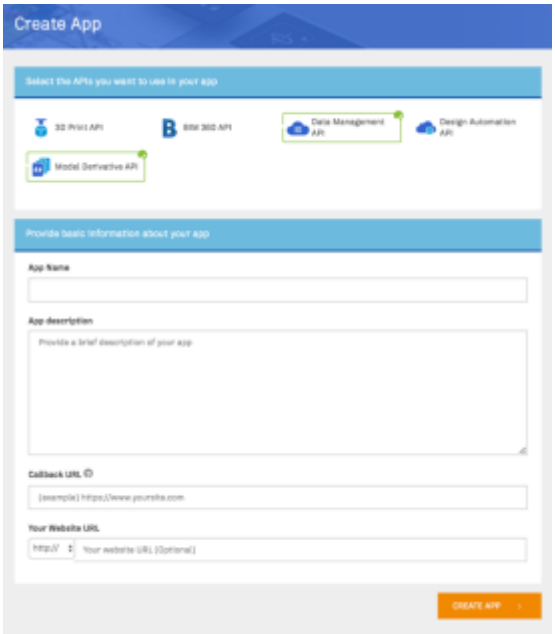
If an operation your program is doing might take a long time then you can show a progress dialog that helps the user see how much more time the operation might need to finish

Know how to get started with Forge

Forge is a set of web services that you can use from any application. Since it can be accessed through HTTP protocol therefore almost all programming languages and environments support it.

The main parts currently available to 3rd parties are:
- Authenticaton
- Data Management API
- Model Derivative API
- Viewer
- Design Automation API

Before you can use any of the services you need to register a **Forge** app on https://developer.autodesk.com



The **Callback URL** will only be needed in case of 3-legged authentication. Once you created the app it will get a **Client ID** and **Client Secret** value that is needed for authentication.

Authentication

This is something that everyone will need in order to get access to the **Forge** services.
It provides two types of authentication: 2-legged and 3-legged

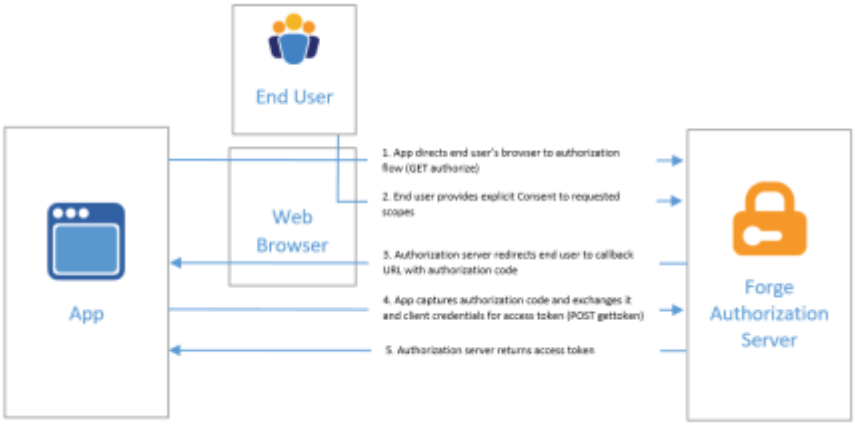
2-legged authentication

This authentication enables your application to access its own application specific data and so can be done without any user interaction.
This is quite straight forward as you just have to send an HTTP request to the https://developer.api.autodesk.com/authentication/v1/authenticate endpoint with

information about the **Client ID** and **Client Secret** of you app and the service will return an **access token** which can be used to access the other **Forge** services..

3-legged authertication

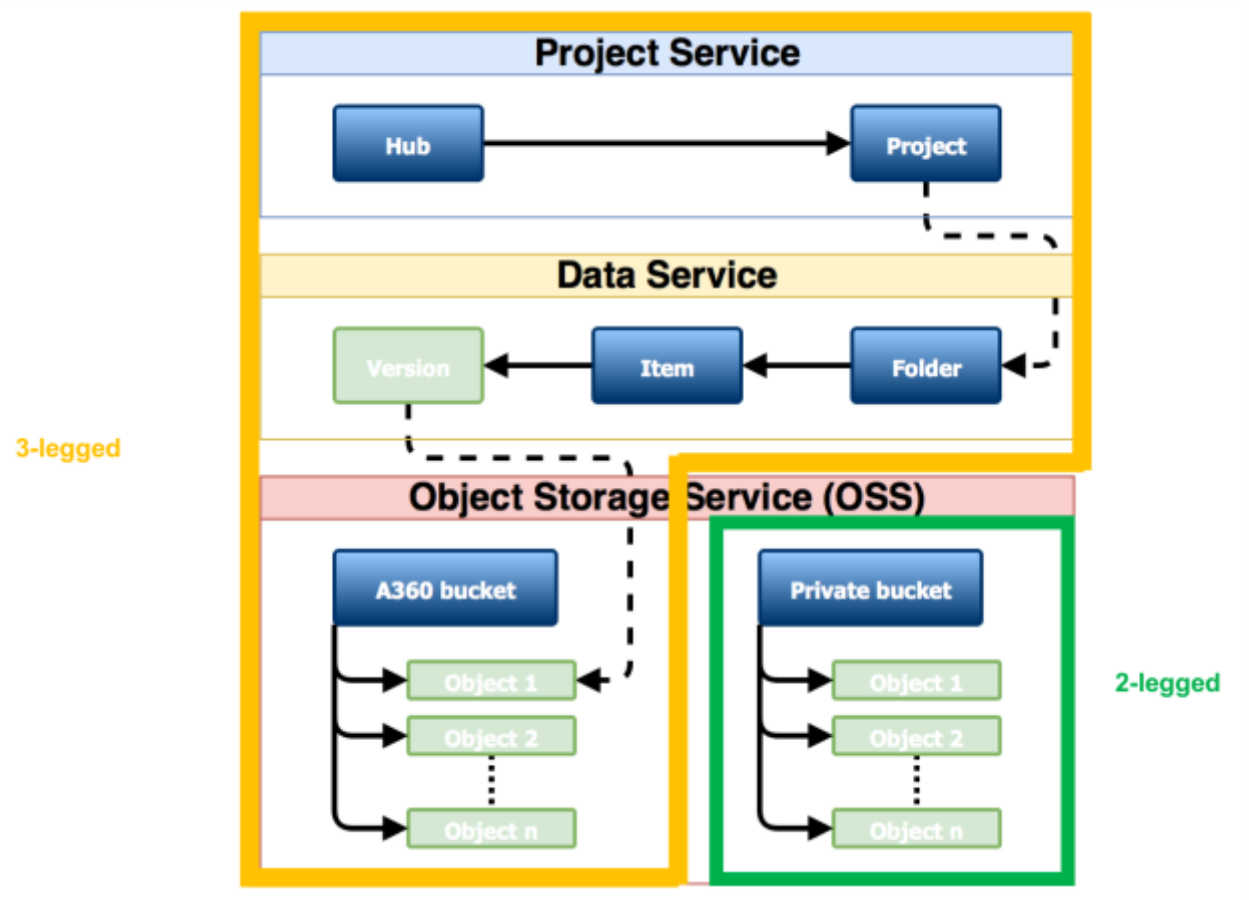
This authentication enables your application to access user data and requires the user to approve the application's access to their data



In this case your app needs to redirect the user to the Autodesk authentication website and once the user approved your app's access to their data, the Autodesk server will call your app's **Callback URL** (provided when registering your app) to send a **code** that your app can exchange for an **access token**.

Data Management API

The **Data Management API** will provide access to any data stored on the **Forge** servers – whether it's in your app's private bucket on **OSS** or user data stored in one of the **A360** type servers (**BIM 360 Docs**, **Fusion Team**, etc.)



Depending on which types of data you want to access (data in **private bucket** or user data in **A360** type storages) you need to use **2-legged** or **3-legged** authentication to get the appropriate **access token**.

In case of **A360** type servers the data is organized by the **Project Service** and the **Data Service**, and the actual raw file data is stored on **OSS**.

Here is a list of the main **endpoints** provided by **Data Management API**:

/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

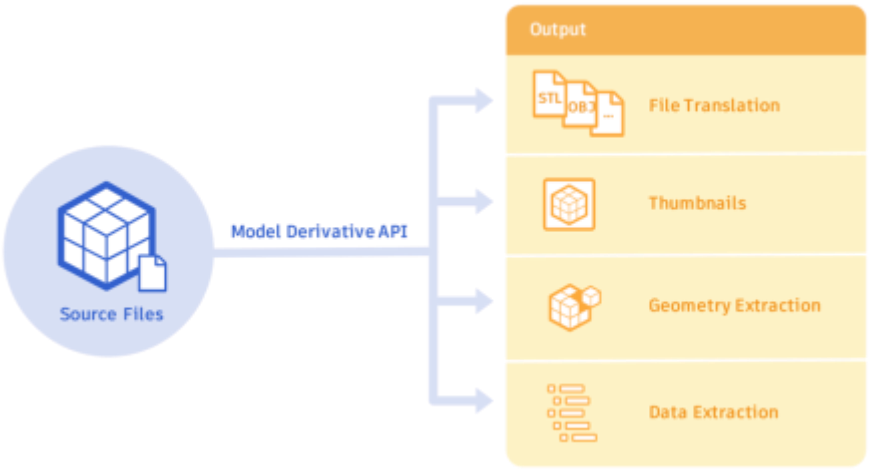
/oss/

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

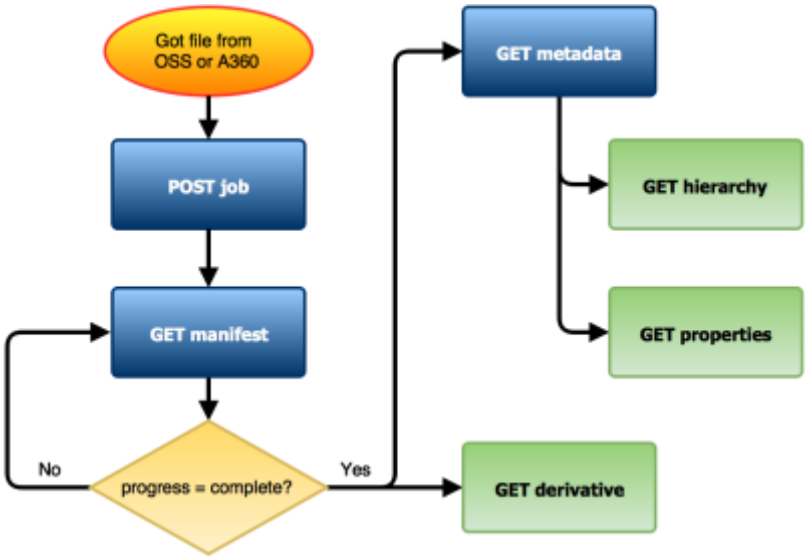
Model Derivative API

This **API** lets you do multiple things:

- get meta data from any of the models
- provide thumbnails
- get geometry information out of the model for any component(s) in the model as **OBJ**
- translate the model to other formats depending on the input file type: e.g. only **Revit** file can be translated to **IFC**



The translation process is best described by the following diagram:



First of all the model file needs to be on a **Forge** server. It could be uploaded by the users or could be done programmatically using the **Data Management API**. Then you can submit a **job**, which basically means requesting some sort of translation. You have to keep checking the **manifest** of the model to see when the translation finished. **Manifest** provides data about all the translations requested for a given file, including the progress of all the those translations.

The **manifest** also provides the **urn** of the **translation** (e.g. the **IFC** file), so once the translation finished you can use that to download the file.
However, if you want to get more information out of the model then first you need to check the available **views** of the model using the **metadata** endpoint.

Once you have the **GUID** of the **view** you are interested in then you can also request information about the component **hierarchy** inside the model and get all the **meta data** that was extracted from the model for each component and subcomponent.

See how you can use Forge services from inside Fusion 360

Since the **Forge** web services are all **RESTful** and the communication is based on **HTTP** requests, therefore you can access them even from inside a **Fusion 360 add-in**.

Depending on the programming language you are using for your **Fusion 360 add-in** (could be **Python**, **JavaScript** or **C++**) there are different libraries available to help you with this task.

In case of **Python** you could simply use the "**http.client**" library to send **HTTP** requests to **Forge** and other web services and the "**json**" library to **read** and **write** the data you send and receive from the external **web services**.

In case of our sample we'll be storing additional data in an online **mongo database** so we can use the "**pymongo**" library to communicate with our storage server.

Our sample is quite simple. It creates a **command** that lets the user select any **body** inside the model and provide extra data for it. That data will be stored in our **mongo** database on **mLab**:

```
uri = 'mongodb://<user name>:<user
password>@ds143734.mlab.com:43734/fusion-data'


class MyExecuteHandler(adsk.core.CommandEventHandler):
    def __init__(self):
        super().__init__()
    def notify(self, args):
        # Code to react to the event.
        print("MyExecuteHandler")
        eventArgs = adsk.core.CommandEventArgs.cast(args)
        inputs = eventArgs.command.commandInputs
        selInput = adsk.core.SelectionCommandInput.cast(
            inputs.itemById(commandId + '_selection'))
        strInput = adsk.core.StringValueCommandInput.cast(
            inputs.itemById(commandId + '_string'))
```

```
body = adsk.fusion.BRepBody.cast(selInput.selection(0).entity)

# If the body is not in the root component
fullPath = ''
if type(body.assemblyContext) is adsk.fusion.Occurrence:
    fullPath = body.assemblyContext.fullPathName + "+"

fullPath += body.name

dataFile = app.activeDocument.dataFile

# Get document URN, version number and project name
itemId = dataFile.id
version = dataFile.versionNumber
projectId = dataFile.parentProject.name

# Store the data in the Mongo DB
client = pymongo.MongoClient(uri)

db = client.get_default_database()
coll = db['mycollection']
for document in coll.find():
```

```
    print(document) # iterate the cursor


coll.insert({
    'itemId': itemId,
    'version': version,
    'projectId': projectId,
    'fullPath': fullPath,
    'stringData': strInput.value
})

client.close()
```

See detailed examples of Forge integrations with Fusion 360

Many **Fusion 360** related **Forge** integrations can be achieved without touching **Fusion 360**, since all its files are stored on **A360** type storage services like **Fusion Team**. Those storage services can all be access using the **Forge Data Management API**.

Designer to customer

It can be time consuming to prepare a finished model to be made available to consumers on your website so they customers can see them. The **Forge Viewer** makes this process much simpler and combined with the **Data Management API**, your models could show up on your website simply by placing them in the right folder on **Fusion Team**.

Enrich Fusion 360 data

Using the **Fusion 360 API**, the above process can be improved by also **enriching the data** that will be **available to your customers**. The additional data could also be used in your **internal processes** to track certain things in the model.

Model update notification

With the introduction of the **Webhooks API**, now your **web service** can also be notified about changes on the various **Forge** storage services including **Fusion Team**. So, in case of e.g. a new version of the design appearing you could programmatically examine the new version, compare it to previous versions or drive **PLM** or other process statuses.

3d printing from the web

In order to get a **Fusion 360** file in a **3d printable** format like **STL** or **OBJ**, you can simply use the **Forge Model Derivative API**. You do not need to have **Fusion 360** or any other software installed on your system.