ADAM NAGY: Good morning, everyone. I'm glad to see that so many of you survived the party yesterday. Did you enjoy it? Yes, so my name is Adam Nagy. I'm working in the Forge Partner Development Team, so we are helping our partners build on to provide a Forage platform. And so if they have any questions they come to us. And as a matter of fact, they also have the Forge booth down in the exhibition hallway. If you haven't visited it yet, you're very welcome to come by and then you can learn more about this technology.

So just a class summary, basically we will focus on one main component of this Forge platform, which is the Model Derivative API. A couple of the learning objectives-- and let's start with a sort of random looking question-- So are any of you familiar with the history of the world wide web? Or how it came about? Like a couple of you, OK.

So it started with Tim Berners-Lee. So back in the 1980s he was working at CERN, which is the European Organization for Nuclear Research, so as you can imagine it's full of scientists, they are all doing their researches. They were all using their own computers, their own applications, which is all fine. But the problem started when they had to collaborate. Of course, that was one part of the research, you know, you have to be able to share your data with others, and that's when they hit a snag. In Sir Tim's own words, In those days, which is back in the 1980s, there was different information on different computers, but you had to log on to different computers to get at it. Also, sometimes you had to learn a different program on each computer. Often it was just easier to go and ask people when they were having a coffee. Right?

So we have a similar issue in the CAD industry as well. So I could just create my own saying based on that, which is relevant to our industry. It says you are there, there are loads of different file formats that you need to use and the way started off as when you were using a specific design software, then the only way to access the data on it was to actually install that specific design software on the computer. You needed to buy it, you needed to get a license to it. And of course people didn't like it, you know, if you just wanted to have access to the data, you did not even want to modify anything, you just wanted to see the model, you had to go into all that trouble.

So the various companies came up with a solution, we started to provide viewers. So we provided a viewer for DWG files, we provided a viewer for Inventor. Our competitors did a similar thing. But even in that case, you know, you have to install something on the computer which is not a nice thing. And especially nowadays when people are using all sorts of different devices, they are using their mobile phones, they're using their tablets, and so on, using different operating systems Linux, Mac, and Windows, it's not really a good solution. So that's one of the things we were trying to solve using the Forge platform, one component of which is the viewing technology, which enables you to show the model to basically any of your users, only needing a browser. They don't have to install anything.

So a couple of the components that we currently have on top of the Forge platform, we have authentication, so obviously when you're creating your own application on top of Forge, you need to authenticate your application with us, that you know who you are, what you have access to. So there's the authentication part, then the data management APIs. These provide access to the various files that your users are storing on A360, Bin 360 Docs and the rest. Or it also enables you to upload files into your application's own private bucket on our servers.

And once you have the data, you have the file on our server, then you can use the Model Derivative API to translate them into either other file formats where you extract information out from them. Get the geometry information out from them, get the hierarchy information, properties, and so on, or even translate it into another file format. Hi, [? Sean ?] [? Miller. ?]

And the old server Design Automation API currently it only includes AutoCAD, so the way you could look at it is like a headless AutoCAD running on the cloud, which enables you to not only access the data from the [? data.bg ?] files but you can also create new [? data.bg ?] files or modify them as you wish just as if you were using AutoCAD directly.

And then the last component is the viewer that I just mentioned. So this enables you to view any models from almost any file format directly in a browser.

The way you can look at the Forge platform is like it's a set of LEGO components. So you can just put them together and build whatever you want from it. You know, it's a plane, it's a building, whatever it might be. So you can fully customize how you're using the various components. At the moment, as you can see, we don't have too many components, but even those are really powerful. But later on, we keep adding more and more functionality to them that you can use.

So first of all, I start off with a viewer. This will be the introduction to that. I'm pretty sure most of you have already seen it. If you've been on the A360 side this, is the exact same technology which is being used there as well. But you might not realize that this is actually powered by the Model Directive API, which then provides the geometric information or the properties which show up in there. So if I go to one of my models on A360--

Are you all familiar with the viewer already? Using it? No? Some? Most? I don't know. Anyways, I'm looking at my model and I mean it, obviously the geometry will be there, which is already in it, is quite powerful, so that you can do funky things like explore the whole model. It's much easier for you to understand what sort of components it consists of. You can also do sectioning, if you want to look inside the model. So here you go. I can rotate it around.

I can push it down. I can look inside the components. I can also select various components in it and then interrogate them. But apart from the geometry, I also have access, as I mentioned, to the hierarchy of the model. So I can see what sort of assembly, sub-assemblies, and various components, and then solids it contains. And I can also select the components and in that case, it will be highlighted in the user interface, so I know exactly what I'm looking at.

And that's not all. You also have access to the various properties of the model. If you select something in the model then you will have information which was extracted from the original model that you were using, but maybe coming from SolidWorks, an inventor, or whatever that may be. So these are all available through the Model Derivative API as well. Previously they were sort of bundled together with a viewer, but now they are all separate.

So if you want to implement some sort of workflow, which will not even require to show the model to the user, you just want to programmatically access data from the origin or model files and do something with them. Maybe do an analysis on them or you want to do the rendering yourself, then you can do that.

And now the data management API. So that's what you use for accessing the data or uploading data to our servers. Here you can see how the data is structured. So this is the logical order, how the data is organized for example on A360. So if you go into a A360, you will see the various hubs you had access to. Then in each hub, the projects, then the folders, items and versions. And each versions are actually storing the actual file on the object story service that you have programmatic access to as well.

So you can either access these objects or you can create your own private bucket on our server and then upload files there and then use them later on for data extraction, for example. So as I mentioned it here, all of the items which are in green can be used directly by them model derivative service.

And depending on what data you are accessing, you have to use a different kind of authentication. There are types of authentication. One is the three legged, because in that case the three legs are your application, the AutoCAD server and the user, who's data you are trying to access. Obviously the user will need to approve your application to access their data so that you can work with them.

The other part is much simpler, there's just a single call to our server, is the two legged authentication in which case you are using your application's own private bucket. So obviously you don't need the approval from anyone else, it's your own private bucket and only you have access to it.

So here's the quick overview of the two legged application. This is really simple. Once you registered yourself on developer.auto.com you can create an application, and straight away you will get a client ID and client secret for your application. Using these two things you can authenticate your application with our server. You just send a single request and then Autodesk server will say OK, got it. We are sent back an authentication token and using that authentication token now, you have access to all the functionality that it provide on top of Forge.

This is just some overview on what it means. And in case of three legged authentication, it's a bit more complex because in that case, your venue-- the user is using your application, you will request access from the Autodesk server. The Autodesk server will redirect the user to a web page where they can log in. And if the user says OK, I want to give access to this application, then your web server's endpoint, will be called and then as a response, you can make a final call to get the access token from our server. So it's a three step process instead of just a single one.

We also have something called the authentication scope. This is just an extra layer of security on our servers. Basically, this enables you to when you're requesting the token and you want to interact with our services, then you can specify what sort of access you want. Maybe you just want the read-only access, which means that if somehow someone intercepts the access token which your application is using, it's not really a problem because they might be able to just read a specific data, at least they won't be able to modify it. Or they won't be able to find out what other files you're storing. So it's just an extra layer of security.

And when you're using any of the endpoints which are provided by our web services, you can go on to develop.autodesk.com, you will have information about all the various endpoints. So all these areas are what we call endpoints. And when you get the information about them, it will be specified what the exact authentication scopes [? dysfunction ?] [? that you will ?] require.

So when you are accessing or when you're requesting an access token from our server, you have to make sure that you say, in this case this is the most basic access, you just say I just want to read data so we use the data-read scope.

And then once you have your data on our server then you can take advantage of the other services, including the model derivative API. And here's just a quick overview of the function it provides. It provides file translation, so taking a source file, whether it's a Revit file or whatever it might be, you will be able to translate it into other file formats, including OBJ, STL, or whatever is supported for the specific file format. For example, in case of Revit, we just added the functionality to translate to IFC format as well.

And then you can also get sub-mails, extraction geometry out of any of the models. Sorry, you have a question?

AUDIENCE: On the extraction geometry, does that imply that you get adjacent version of the geometry? What do you actually get?

ADAM NAGY: No, you get an OBJ file.

AUDIENCE: OK.

ADAM NAGY: And well, I will talk more about the OBJ file in a couple of seconds. And then you have also the data extraction that means the information about the hierarchy of the model, the various components, sub-components in it, and also all the properties of the various components in it. So you have access to all this information through the service. And we are supporting 60 plus file formats. You find the least of that on the developer.autodesk.com site and you click on the Model Derivative Service, so you can have a look at it to find out what is supported currently.

We keep adding functionality, so it's never up to date really.

So again it just highlights that we are the supporter of 60 plus file formats. The OPJ file is actually possible for all the supported formats. It might be possible directly, so you just uploaded the model and say I want OBJ and it will be supported exactly. But in some cases you have to first get NS via file translation and then you will have access to OBJ files as well.

And the really cool thing about the OBJ export is that since you have access to the hierarchy you can just pick and choose the components that you want to extract the geometry for. And only those will be contained in the OBJ file. And their places to be kept as well.

And other supported formats STEP, IGES, STL, and as I just mentioned, we have new formats coming as well. So this is just what we had when I created the slide a while back. And of course, let us know what you need. So if there's a specific workflow you need from Revit to something else, which we're currently not supporting, do come back to us and let us know that you need that.

And how the workflow goes. So first of all we need obviously a file. The file needs to be on our server. So as I mentioned, it could be either A360, Bin 360 Docs, or the rest. Or it could be in your applications private bucket, stored on the Object Storage service.

Once you have the file, now you can start using the model derivative service, you can post a job. That means that you are starting a translation. When you're posting the job you can specify which file format you want to get and also all the options you can provide for that translation. Once that's done, you have to get the manifest for the file. This manifest will contain information about all the translations which were requested so far for this file. So you will be able to find out what translation was requested and how far the translation process got. So in case you might had a bad file, it might have failed, or it might be at 50%, so you have to wait a bit or it might have been completed, and you were able to find it out from the manifest.

At the moment, you have to keep polling our server to get an up to date manifest. But we are working on providing web hooks later on, so you will not have to do that. Once you created your own web server we will just call an endpoint on your web server and then you will be notified that the translation finished and then you can continue with your own process.

Let's say now the process completed. Now you can download the file that we generated, but it is an OBJ file or STL or whatever it may be, and then you just download it, call one of the endpoints on our web server, or if you want to get the hierarchy or properties for the model, in that case, first you have to get the metadata for the file. The metadata will contain information about the various views which are available for the model.

So, for example, in case of a Revit model, you might have multiple views, and then they are all showing different things, so you have to pick the view that you're interested in. You will get back a GUI ID, a GUI from there, and use that in order to get the hierarchy for the specific view or get back all the properties for the components interview.

So these are the endpoints we try using. As you can see, it's not too many, but it's really powerful, and all the options will be provided in the post file so-- or when you're posting a job. So these are the endpoints that they are supporting.

First of all, the formats. So using these endpoints, you will have access to the table of the order-supported translations from which file to which other file they are supporting translation. And then using the posting to the job endpoint will are able to start the translation. Then you will be able to get back a thumbnail for the model. Then get back the manifest, which I mentioned contains the information about all the translations which were requested. Then you could even delete it if for some reason you want to. Or you can download the generated derivative file, which could be like an OBJ file or IFC file, and then you can get the metadata, which provides a list of all the views which are available for the model. And then you can get the hierarchy information for the specific view and get all the properties of the components for the specific view.

Fortunately when you're going to develop.autodesk.com site and you click on the Derivative service, you will find a few step by step tutorials. We will take you through all the necessary steps so you can just follow them just one by one and be a be able to implement whatever is listed there.

So again, this is just showing how the data is being structured on our servers and how you can access them.

So when you're posting a job, one thing you have to watch out for is that you might want to work with a complex model like an inventor assembly which consists of multiple files. You have a main assembly, a top assembly, you'll have sub-assemblies, and you have it you have additional part files. So if you want to handle these, then you have to zip them up to enter into a file and then send that to our server and then you can start the translation on that file. If that's the case, you're working with the complex model then when posting a job request then you just have to set these additional two properties which is compressed URN2. URN is basically the name that we are using for identification of a file, so we call it URN. And that set at the root file name, whatever the root file name is for the model.

So if it's the top assembly it's called main.iam then that's what you specify there. If it's a simple model, you don't have to care about it, if you are just uploading part file where the user uploaded a part file to A360, and you want to work with that. In that case, you just set compressed URN to force.

And one important thing is that you shouldn't change the extension of the files, because this is how the derivative API will find out what which translator to use in order to translate the file into something else.

So once you have access to the file, let's say I'm accessing a specific version from A360, then I can post the translation and then the active service will provide the various information, the manifest for each version, then the translations which are available for that and then you can download that from the derivative service or get the properties or get the hierarchy information from it.

So one nice thing is actually that when you're using Data Management API to get to a specific version on A360, then the information which would be provided for that version will automatically contain a direct link to the manifest file as well for that specific version. So you can just follow that in order to get the information about the requested translations, but what is already available.

So for example, when you're uploading files in the user interface of A360 to A360 then it will automatically kick off a translation to SVF format and this SVF format is what we are using in order to show the model in any browser. So it will automatically be there and you will have access to it programmatically. One really nice tool that I like is Postman. Any of you familiar with Postman?

AUDIENCE: No.

ADAM NAGY: Yeah, so it's really nice because it's free, for a start. So you can just go on Chrome and this will be like a chrome extension. I already started it here. And using this, you can specify the various cores that you're interested. In so basically without doing any kind of programming, you can play with the four JPIs and find out how they are working, what sort of information they are passing back. So whenever you're calling any of the endpoints, we tend to expect the information from you in adjacent body, and we will provide information in adjacent body, so that's how the information would be structured.

So let's go to here. So for example, how do you authenticate yourself? I'm just showing here the simple version, which is the two legged authentication. So if you go on the web site developer develop.autodesk.com I went on having a look how to authenticate my application with the Autodesk servers. I have a step by step tutorial for creating an app, getting two legged and three legged token, and you can see what you have to do. So basically you just have to post the request to this endpoint. This is the URL. This is the only header you have to provide and this is the data that the requests will need.

And then following that information you can build up your request inside Postman and you will be able to save these and then keep reusing them and it's a really flexible user interface as well. So you could even take advantage of environment variables. I can go into more details if you talk to me after the presentation. So here I set up, this is the endpoint I need to call. You can see that this needs to be a post request. Actually even if I did not set the header, I think it would be actually automatically set because of this. So I just added the information and that's it, everything is set up. Now when I'm sending a request, and this is what you will get back. Autodesk service says OK, here is your access token, your authenticated, now you can start using the various APIs.

So, for example, if I wanted to create my own private bucket on the object stories service of Autodesk or Forge, then I can do that. Again, you can have a look at how that should be set up, so you just go on Data Management API, you can have a look at the API reference and you can see how to create a bucket, basically just a post request. This is the endpoint that you need to call and it provides most of the information you need to provide and also it will show basically a sample for that sort of data view you need to specify. And following all that information, you can just build up your request again and then save it. And As you can see, I've created so many requests already that I can just select them and then run them whenever I need them.

So in this case, I'm trying to create a bucket, which I already created previously, so I get back the response bucket already exists. Or I can upload files. I can get back the files which are already uploaded to the service. As you can see, it's all stored so I just click and choose whatever request I want to test. I can provide the data for it, I can quickly just adjust the data if I want to do something different, click send and then you can see how the API is working.

So for example, this is how you can start the translation. All you have to do is you call this endpoint, you send a post HG3P request, and then you specify a body, something like this. As you can see, this is adjacent data, so it's organized in this fashion. This is all the information you need to provide since in this particular case, I'm working with a complex model which consists of multiple files so all the information is in a single zip file, therefore I say compress UN2 and I specify the root file name. If you're familiar with Inventor, I'm pretty sure you've seen this model before. This is as iam, this is one of the sample files that are available as part of the inventory installation. So I could just post the job.

And the derivative [INAUDIBLE] say, OK, I got this, I'll be working on it. And I can start polling the manifest for the file that I uploaded. So I just send the get request, this is the URL that I need to call using HTTP GET, and then I get back the same information, so you can see a list of all the derivatives. Basically, this is the list of all the translations that were requested so far, whether by you or by other services that had access to the data.

And as you can see the translation already finished. So I already have the SVF file so I would be able to use this file directly in a browser and then make it available for the user to interact with the model in the viewer. I can also get my metadata in case of the Inventor Assembly, I have a single view so that's quite simple and now that I have the GUI'd of the view that I'm interested in, I can get back the hierarchy information and the properties for them, although first of all, let's give the hierarchy a try.

So all I have to specify is the URN in this URL Endpoint. All I have to specify is URN of the origin of file that I'm dealing with and the model GUI. This is the GUI that you've just seen in my previous request that came back. And it says, OK, I got the request. I will try to extract this data now. So if I keep calling this endpoint, then sooner or later it should say, OK, so here's your hierarchy, I figured it out now. So now I have a list of all this. So it's pretty simple in case of this assembly because it's just I have a top assembly and then I think I just have three parts in the whole assembly. So I have the blade main, blade top, and the scissors spring and you have all the information about that.

Then I can also get bigger properties for each of the subcomponents and I just call a different endpoint. Again, I specified URN of the Source file that I'm working with and then the model grid, which is the identifier of the view I'm interested in. And here you are. It's a list of all the components and all the properties which could be extracted from the original model.

I could also get back the thumbnail. And very well, this is the scissors assembly. Any questions so far? Yes?

AUDIENCE: So could you also get the geometry of just one piece in there?

ADAM NAGY: I'm going to show it, yes. So you can do that. Where am I? Here. Yes, so concerning postman, I have two articles on our DEV blog that you can have a look at it because I was really impressed with postman, so I thought it would be useful for others as well. So you can have a look at those. And so here is a sample that I created, this is based on my colleague's sample. He created it for showcasing Data Management API and then on top of that I added the functionality coming from the Model Derivative API, so the hierarchy information, the properties, and all the translations which are supported for the files. So let's have a look at that then.

Did I already open this? Yes. There you go. So you can find this sample at derivatives.autodesk.io. We have quite a few samples, most of them will finish with autodesk.io, so this is how you are going to find them. So let's just log out so that you can see the whole process from the beginning. So this is how the sample starts. This has been written in Nodejs If you never done any web server implementation then I think the easiest way to get started is with Nodejs It's really simple. You just install it, you download our sample you just set client ID and Client Secret of your application, you put it inside the sample and you can run it straight away. It will be running locally on your machine, you can keep playing with it. And see how it's working.

So in the sample, first of all I need to do the authentication because it's a three legged authentication. It should ask the user to log in but since I already logged in and it stored all the information about that; therefore, it's quite easy, it doesn't pop up. But if I quickly delete all the browsing data, then it with ask me for my credentials. So just to show how it works, I'll do that quickly. So I say log in. Now the Autodesk server is showing its own, this is not something I implemented, this is what is coming from the Autodesk server, so it's asking the user to log in. And since I already use the service it wasn't asking for the permission, what sort of permission I'm providing for this application. If this was the really first time that this application tried to access my data, then you would also see a dialogue where the Autodesk server web page is the least thing or the function it is I'm trying to get on top of my data and I can say, OK, I allow the application to access my data and work with it. So now it already happened.

Now I have a list of all the hubs I have access to and this is actually coming from BIN 360 Docs, so this is half-way working now. So if you have the-- Any of you using Bin 360 Docs? No? Oh, anyway. So in that case, you don't see how exciting this is, anyway. So I'm just going to another project. so you can see all the hubs I have access to, all the projects inside those, all the files and models I have and each version. You know, I can have loads of versions of the same model. So in this case, I'm clicking on the latest version of this model.

So the thing is, since the translation was already done, this information is coming from A360. Therefore, the SVF file, which you need in order to show the model in the viewer has been already generated. So actually you haven't really seen that it kicked out the translation. So I'm trying to provide feedbacks in here. So when you're clicking on any of the versions, you see translation completed. If it was really great because I just said I wanted translation, but it was already there, actually and then I'm asking for the manifest file and it says, OK, it's all done. And then I'm requesting the hierarchy of this model.

So as you can see, I have access to the complete hierarchy, whatever I'm interested in. If I click on anything here, then I get also back the properties that was extracted from the original model . And also actually, I synchronized the selection between this tree view, which shows the hierarchy and the viewers. If I select something here it will also be selected here.

So for example, if I wanted to, you know, I have this model from wherever it is. It could be, for example, fusion, I'm working on something. I already created this model but this broke, so I just bring up this web site, I have access to my data, I select this, I want to print this again, so I'm only interested in this specific subcomponent of the model. So I select it, I say I want an OBJ file out of it, I click download, the translation is starting, it's taking off, hopefully it won't take much time. And there you go, the OBJ file is already there. I downloaded it, so let's have a look at it. Show in Finder. Which one was it? I think the thing might be a bit off here. That's like the wrong component. Might be better to go back again.

So I selected the component, I said I want an OBJ download-- Yes, that's better, I think. Showing finder, this is the component I just selected. So for example, if I had some software for 3D printing, I could say OK Maker bot open it up, move the platform, make flat. Boom. Want to have a look at it and it's ready to print. So without having to go into fusion or having to go on A360, or anywhere else, you can just implement it yourself. You could do it in the background. You can easily integrate it into your own workflows and do whatever you want with the geometry.

So again, the basic example is available on derivatives.autodesk.io. You can also find the source codes for all our samples. It would be either autodesk-forge or developer-autodesk.

So as I said, the easiest way to get started with the web service development is, I think, to use Node.js, and for that we also have packages that you can take advantage of. So it's really easy to install packages in Node.js. You just specify the name of the component that you want to use And then you say NPM in-store Forge authentication and the package will be installed as part of your project and you can start calling the API functionalities.

So in order to get started you just go on developer.autodesk.com-- actually I might as well show it. Have I shown it already? I keep showing these things in the booth all day long, so I am completely confused by what I've shown today. But you can just go to the developer. autodesk.com, you can log into your Autodesk ID, and then you will have access to all the applications that you've created, or you can just create a new application.

It's all free and we're providing free access for a year. So you can keep playing around with the API without paying anything. So you just select the components that you're interested in, what you want to enable for your application, specify an application name, some description, if you want to be able to access data of your users, what they are storing on A360, Bin 360 Docs, and the rest, then you also need to provide a callback URL. This would be the exact URL that your web service will be implementing because this is the URL the Autodesk force server will be calling when the user authorized your application for access to that data.

That's all the things you have to do. Website URL, you don't even have to specify as you can see, it's optional. Click create, the application will be there. And also, you have your application automatically straightaway, you will get the client id, the client secret, and this is the only two pieces of information you need to use and add to the sample project or the project that you're creating in order to start using our services.

AUDIENCE: Excuse me. Sexy photo instructor time [INAUDIBLE].

[LAUGHTER]

ADAM NAGY: Anyway, so you can also find the documentation and all the tutorials on developer.autodesk.com. You will be able to navigate there. We are providing many samples on GitHub, so as I mentioned it's either Github.com/developerautodesk or autodesk-forge. Got migrating all the samples through Autodesk Forge, so this will be the official list of samples, and you can ask questions on Stack Overflow, you just need to use specific tag that you're interested in, but it's Autodesk model, derivative Autodesk data management, Autodesk core authentication, whatever it may be.

You can also find tutorials on the YouTube, so you just look for the Autodesk YouTube channel and then you will have access to the Autodesk Forge YouTube video list, one of which will be the thing that I just talked about today. So intro to Model Derivative API that we did as part of an online hackathon which was going on in October, November.

And again, if you have any more questions then do visit our booth and exhibition hall and then you can learn more about this technology there. So please do provide feedback so that we can improve things. And that's it. Do any of you have any questions?

AUDIENCE: You were able to the performance of the model of different geometries in terms of hierarchy. Are you able to do that in layers from like [? AutoCAD? ?]

ADAM NAGY: With layers? You can't specify the layers, no. Anything else?

Well, in that case thanks, for coming, and I hope you found this useful. OK, thank you.

[APPLAUSE]