BES500019

BEYOND DYNAMO: THE POWERFUL AUTOMATION POTENTIAL OF FORGE AND THE REVIT API

Majd Makhlouf
Founding Manager at Building Information Researchers and Developers OÜ

Learning Objectives

- Explore automation opportunities not exposed through Dynamo
- Learn about technical Revit API and Forge features that make automating the aforementioned cases possible
- Explore real case scenarios where Dynamo non-exposed Revit API functionalities came to the rescue through add-ins and macros
- Explore real case Forge apps where additional Forge functionalities have been used to achieve an even better automation

Description

We all love Dynamo. It introduces visual programming to our daily workflows and makes task automation accessible to non-developers. It's great for prototyping and testing.

However, Revit macros and extensions and Forge APIs offer many automation opportunities that cannot be technically exposed through Dynamo.

This class aims to help Revit and Dynamo users think outside the box by introducing real-case customer success scenarios in which these advanced and powerful Forge and Revit features have been implemented—proving that you can literally "Make Anything" with Forge and Revit when you explore both solutions deeply enough.

Speaker

Majd Makhlouf is a Mechanical Engineer and Design Technologist, with a Master of Science in Mechanical Engineering. He's an Autodesk Revit Certified Professional and a member of the Autodesk Developer Network. In January 2020, he founded Building Information Researchers and Developers OÜ, a software development company based in Estonia and providing services for the AEC sector worldwide. He specializes in BIM Management, Autodesk Revit and AutoCAD Add-in development, both public and custom developed, Forge web and cloud-based apps, Dynamo Zero Touch Node Packs, and mobile VR/AR applications.

Co-Speaker

Ghida Shehadeh is an Electrical Engineer with a Bachelor of Engineering in Electrical and Computer Engineering degree from the American University of Beirut, where she also worked as a teacher assistant and collaborated on several research projects as a research assistant. Her research and documentation contribution within the IoT domain were featured at the 2020 IEEE International Symposium on Antennas and Propagation and North American Radio Science Meeting, where the IEEE Paper “TU-SP.2P.7”, titled “A Digitally Tuned Flexible Reconfigurable Antenna for IoT Devices”, which she contributed to as a co-author, was discussed. Nowadays, she’s an Electrical Engineer at EMDC Group, one of the most renowned Engineering Design and Consultancy firms in the MENA region, where she benefits from her Autodesk Revit and Autodesk AutoCAD proficiency and dedicates her passion for engineering and love for advanced technology and automation to overcome daily challenges, ensure the highest quality of project deliverables, and contribute to her company’s success.

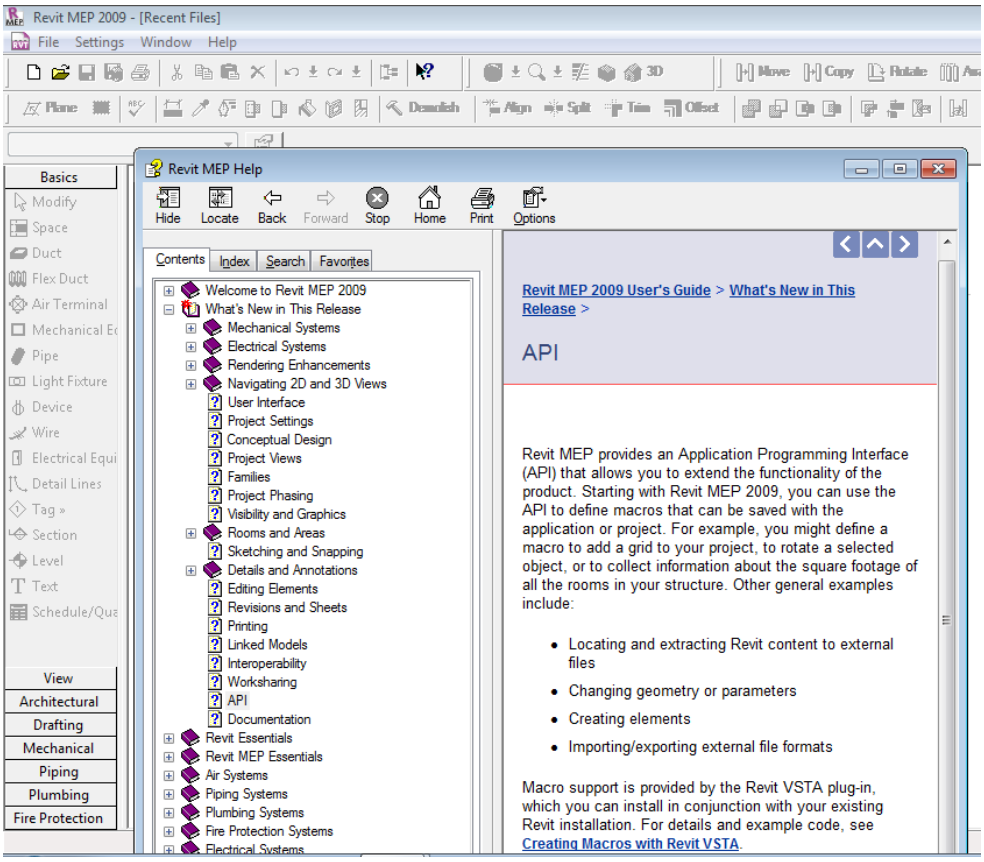
Dynamo: A beloved product

Wherever you may roam around blogs, forums, or social media accounts related to the AEC industry, everyone is talking about Dynamo, and that's for a good reason: Dynamo is a great Revit extension that is a time and life saver to many, as it helps us users reduce tremendous amounts of time. You may encounter "BIMfluencers" preaching about Dynamo's benefit, experienced technologists spreading helpful tips and sharing their experience using Dynamo and sharing some of the scripts they have built, and users showing off their newly acquired skills with social media posts that consist of short videos showcasing their Dynamo scripts in action. Due to its ease of use, great potential, and awesome community, Dynamo has become a trending topic over the internet and every Revit user is aiming to gain more Dynamo skills.
A question can be asked on that matter: why is Dynamo the trending topic? Why aren't other forms of automation as trending as Dynamo? Let's go back in time to try and explain that.

Revit Automation Throughout History

Back in 2008, a major Revit release saw the dawn of light: Revit 2009. It was a revolutionary Revit release, as it has, and besides being the first release where an annual naming and release frequency format were adopted, taken the product to another level. Can you imagine that all previous versions did not include a functionality to export models into the IFC open format? Well, they didn't. They did not include a worksharing monitor either, and the list of newly added features can go on and on…

Among many of the additions that were introduced back then was a public Revit API. Although different from what the Revit API looks like today, it was the first step for advanced users to start automating their repetitive tasks, and many were looking forward to that, as at that time AutoCAD was still the major dominating platform and the AutoCAD API had been around for years and any Revit user at the time was one that had recently migrated his/her workflow from a CAD one into a BIM one, so users were still familiar with AutoLISP and its use to achieve automation within AutoCAD.



Revit 2009 – API Release Notes

Why Dynamo?

As Revit itself is based on the Microsoft .NET Framework, it's API had to be based on that framework as well, and knowledge in a .NET environment was required to start exploring it as a means of automation. Languages required included Visual C++, VB.NET, F#, but C# stood alone as the most widely used and documented language for Revit API development.

As our industry consists of non-developers, C# proved to have a relatively steep learning curve.

In the meantime, more programming environments started to rely on Visual Programming, and new Visual Programming solutions started to emerge, such as MATLAB's Simulink and IBM's Node-RED. Game engines such as the Unreal Engine and Unity hopped on the visual programming train and saw many commonly used functionalities start to be converted into class libraries and methods that could be in turn visually represented as graphical nodes that a user or a developer can place in a certain order and connect through "wires" to achieve a certain desired functionality.

Therefore, having a similar visual programming environment within Revit is nothing but a logical consequence that only needed the right innovator to be hit by the right apple on the head at the right moment. Now as much as I'm sure that no apples were harmed during the process, that time came around 2011 and that innovator was Ian Keough, also known as the father of Dynamo. The rest is history.

Why should one go beyond Dynamo?

Before going into more details, just be sure that Dynamo is a really powerful tool and that you would be missing out as a Revit user if you've never used it before. Almost all tasks can be automated using Dynamo, from submittal and export operations, to automated element creation and placement, to Q/A and model cleanup… All those repetitive tasks can be automated using Dynamo, which also has the potential of taking you to the next level through Generative Design and many other advanced R&D ventures that you may be willing to explore.

However, if you're limiting yourself to the built-in Dynamo functionalities without exploring other available APIs, you would be missing out on many opportunities that may take you and your company to an even higher level of automation. Two main topics would be tackled hereafter: advanced in-product automation through the Revit API, and cloud automation through Forge and its many APIs, as we're no longer in 2009 and a long era of Cloud Computing is here.

Part 1: Advanced Revit API workflows

As stated earlier, Dynamo consists of built-in nodes that users place and connect in a certain order to build a useful script.

These nodes are actually static methods within class libraries. These static methods are themselves built on top of the Revit API and are meant to perform a certain task that may be used repetitively within a certain script.

Therefore, when you are building a Dynamo script, you would be limited to the nodes that you have available. If any functionality is not available out of the box or hasn't been developed as a node by a third-party entity, you're rather stuck. That's one of Dynamo's limitations, and it's not Dynamo's fault: it's one of visual programming's limitations. In fact, visual programming doesn't give you enough low-level control.

Fortunately, to overcome that limitation, one great node has been made available out of the box. That node is the Python scripting node which is a "universal" node allowing users to access the Revit API within Dynamo whenever a non-exposed functionality is needed. Any advanced user would start diving deep into the realms of the Revit API at a certain tipping point of their automation journey, driven by a thirst for more "super powers". That comes to no surprise, as it's the natural way of evolution: throughout history, written languages evolved from a logographic system (ex: Egyptian Hieroglyphs, 4000 BC) into a phonetic, alphabetic system (Ex:

the Phoenician Alphabet, 1050 BC) due to the need of a better, more detailed means of documentation and communication. Even as a human being, learning starts visually for a child, then evolves into reading and writing as both are needed for him/her to evolve academically.

Therefore, and as the evolution into text-based programming is a must to break free from Visual Programming's limitations, isn't it better to do that within an IDE instead of doing it as part of a Dynamo script? And wouldn't that gradually reveal more features that would take your automation journey to another level? Let's explore the other forms of Revit automation available and their benefits.

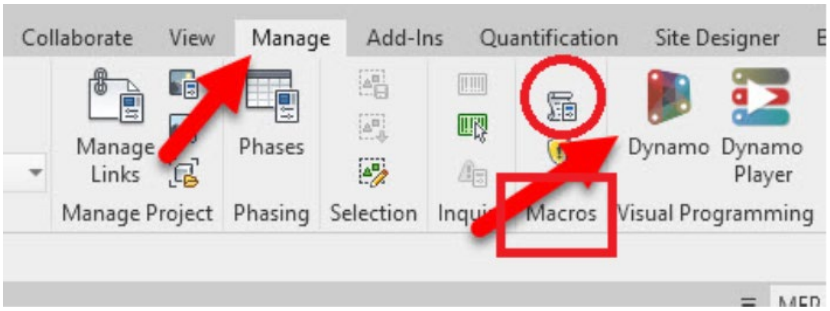
The Revit API and its forms of automation

There are several ways to implement the Revit API. I can think of four: one is Dynamo of course. Another is Forge Design Automation for Revit, which will be presented in more detail later on. For now, let's talk about two: add-ons and macros, starting with the later which is being unfairly neglected nowadays.

Revit Macros

Most are familiar with Dynamo's pushbutton under the Manage ribbon tab. What many are not familiar with is a colorless, much smaller pushbutton right next to it: it's the Macro Manager pushbutton.

A Revit Macro is a form of automation that consists of a script that can be written within an IDE that ships in with Revit (that IDE is SharpDevelop and is an Open Source alternative of the more popular Visual Studio IDE). These scripts can be C# scripts, VB.NET scripts, (Iron)Python scripts, and, for releases prior to Revit 2022, Ruby scripts.



Manage Ribbon Tab

Macros come in two forms: application macros, which are stored locally and accessible throughout different Revit instances, and document macros, which can be embedded within a Revit model and can thus be accessed by all the collaborators working on that model and which can be set to control that model at certain predefined events.

One thing that would be recommended if a user is migrating from Dynamo scripting and wants to start exploring the Revit API is for him/her to start by converting his/her Dynamo graphs into Revit macros, whether those macros are C# ones or Python ones. As a matter of fact, and once a user starts to include Python scripts into his/her Dynamo graphs, the need for graphical nodes decreases and he/she is ready to start writing that same algorithm as a full text-based script since Visual Programming becomes almost useless once a user is advanced enough to write his/her own code. Even for Python scripts, having an IDE with a functioning autocomplete feature and live debugging would make that process way easier and enjoyable.

Macro development is usually easier to deal with than Add-In development, as the IDE is easily accessible within Revit, and the macros can be easily tested from within the Macro Manager (which has an interface that's similar to that of Dynamo Player with a more nostalgic vintage appearance) without having to set the environment up for debugging.

Revit Plugins/Add-Ins/Add-Ons/Extensions

A mystery that will never be solved is the difference between plugin, add-in, add-on and extension. To many they mean the same thing. What matters for now is their common definition.

An add-in is a compiled class library that is loaded by Revit on startup and that integrates within the Revit interface and environment to perform additional functionalities and modify or monitor built-in ones.

Unlike Dynamo scripts and Revit macros, Add-ins require an external IDE to be developed (although an experienced developer can use the one that ships with Revit to develop his own external application). That IDE can be Visual Studio, or, if you're an Open Source fan, SharpDevelop.

Add-ins are the ultimate form of Revit Desktop automation. They have one disadvantage: learning to build them is a bit hard. However, starting an automation journey with Dynamo, and following it up with Python scripting and macros, should ease that process and make it an enjoyable venture.

Macros and add-ins offer all the advantages that will be enumerated in the following parts. To conclude this part, I would like to start with one of those advantages: deployment and embedding.

Deployment is a challenge for companies, and the bigger the company is, the bigger that challenge becomes. Dynamo scripts and macros are ideal when a BIM manager is prototyping or even building a final script that he/she will be using alone. Whenever a certain functionality

needs to be used by every team member, deploying Dynamo scripts becomes a nightmare, and there's always Murphy's law to take into account: something may go wrong during deployment or during operation, especially that these scripts can be edited by anyone. To ensure a safe deployment, a packaged add-in is the best way to go. An installer can be built to be deployed remotely on every workstation, which ends up reducing the time required for deployment and making sure that the end solution would be easily accessible by end users.
As all the components are compiled, there is less risk of having a new inexperienced user modify the code by mistake. Even if the BIM manager/Developer would generously like to expose his/her code for educational purposes to the rest of the team, he/she can do so without worrying about the end solution being altered in case it's modified later on, as the end solution is a compiled one and independent of the initial solution.

Many great third-party solutions are attempting and succeeding in solving the deployment issue of Dynamo scripts. Still, that involves integrating a third-party extension within a company's workflow, so it's back to add-ins.

Another great feature is the document embedding of macros. Embedded macros have the great advantage of making a script available to all collaborators without any deployment hassles, as the macros are embedded within the Revit model.

Runtime Performance Enhancement

Many know that add-ins (and macros) have a better runtime performance than that of Dynamo scripts (to be fair though, the Autodesk Dynamo team are doing a great job and are closing the gap with every release).

What many may not be familiar with are the reasons behind that, and the scale of that runtime performance enhancement.

One of the reasons behind that is also due to Visual Programming: when less control is available, less optimization can be achieved. As the nodes are prebuilt, there are many operations that may be common between two successive nodes that cannot be combined within Dynamo and that can be combined through a conventional text-based script. For instance, and through text-based programming, the number of iterations and of document commit operations can be reduced: basically, whenever a document modification is required, a commit operation is required to modify the document. That operation regenerates all the elements of the model each time, which ends up affecting performance. This may be happening internally within many (if not all) nodes, while the entire script may require only one commit operation. That would be one of the factors that may affect the overall performance and add to the processing delay.

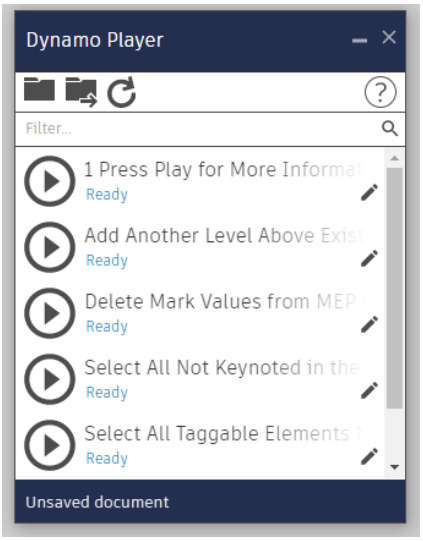
Another disadvantage Dynamo has performance-wise is that, and to avoid DLL conflicts, many libraries are dynamically loaded, which also increases the processing time.
In addition, and in order to be compatible with the .NET Framework, a Python interpreter that is built on top of the .NET Framework would be needed, in this case IronPython, or, starting with Dynamo 2.7, CPython3. IronPython can be 32 times slower that an equivalent solution built directly on top of the .NET Framework which is another reason to switch to macros and to a .NET compatible language once text-based scripting is mastered.

Here's a real-case example to illustrate the improved processing time of an extension, compared with an equivalent Dynamo script: out of the many Dynamo conversion jobs we have completed, one involved a sheet generation and viewport placement script. The final product was able to reduce the execution time down to 45 seconds compared to the Dynamo Script's execution time of 18 minutes. The original script was a very well-built one, and all it needed was to be rebuilt into an add-in and optimized. A lighter script may not be worth the trouble, but for complex ones, it can make a huge difference.

Customized User Interfaces

A user interface is always a requirement for any application, as it allows users to specify the input required by that application's back-end engine to do its magic.

Dynamo out of the box doesn't provide any way to customize the user interface of its scripts. The only quick access tool for Dynamo scripts is Dynamo Player, which lists the scripts to be executed by the user. So, the user needs to open Dynamo Player and execute the scripts manually, which in itself is a waste of time.



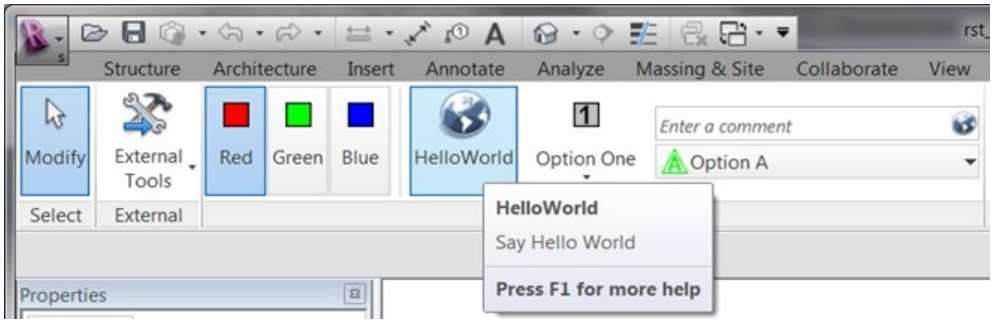
Dynamo Player

Add-ins can be built to customize the Revit ribbon, and the developed external commands can therefore be accessed through pushbuttons like any other built-in Revit command. Keyboard shortcuts can also be assigned to these pushbuttons for an even quicker access. That itself would make a huge difference once the automation tools you develop need to be accessed on a daily basis by every team member.

Again, many third-party developers have built excellent add-ons to map Dynamo scripts to the Revit ribbon and give Dynamo that same advantage, but again, you're back to using add-ins, so

why not build your own directly, knowing that such add-ins are always risky as they may be a cause of DLL conflict with some of Dynamo's libraries.

Creating new pushbuttons is not the only interface customization that one is able to make when he/she goes deeper into the dark realms of the API: a developer can also override built-in commands (technically known as command binding) or have a certain routine run prior to a certain built-in command or right after it. That is useful whenever any built-in command is required to be disabled for example so that no one uses it. A well-known example is the "Import CAD" built-in command, which can be overridden and bound to an alternative command that displays a text message (also known technically as a TaskDialog) denying the user from using that tool. More customizations can be made as well, if one wanders around libraries that are "not really part of the Revit API". One particular example is the AdWindows.dll library, which would allow you to customize the interface to a greater extent, by colorizing the controls or moving built-in pushbuttons across tabs…



Revit Ribbon

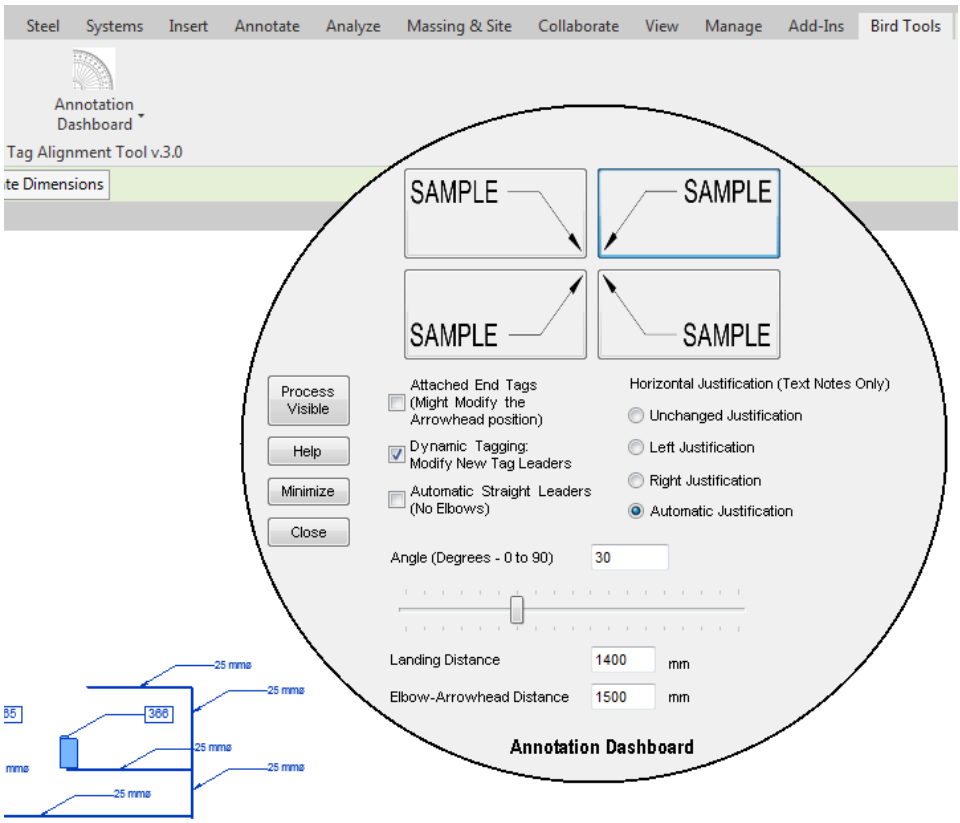
In addition, creating interfaces where users can specify the input for the algorithm is crucial and is one of the advantages of the ".NET way". For a Dynamo exclusive workflow, nodes that are specified as input nodes can have their values controlled within Dynamo Player which many times may not be enough, is not straightforward, and is a slower way for users.

Once more, many third-party developers have developed custom Dynamo node packages to allow Dynamo users to include a Windows interface to their graphs. As great as that is, these interfaces cannot be customized to the full potential offered by the .NET Framework, for both WinForms and WPF forms.

In fact, you are able to customize your forms to many degrees: you can customize your form's layout, make its shape unusual, make it more compact when a compact form is required, and expand it whenever an algorithm is advanced enough to require a multitude of inputs.

Moreover, you can build your form to be a modeless one: that form would be opened once per session and kept open while working, allowing buttons, sliders, checkboxes… to modify elements dynamically in your model. If you're looking for an example to inspire you build your own app, you can check out Bird Tools' Tag Alignment Tool. It's an Autodesk App Store add-on that can control tag angles. It features a modeless Annotation Dashboard that users can open

once per session and keep opened wither on the side or on a secondary screen while working. As soon as a group of tags or text notes are selected, their leader angle can be dynamically controlled and specified using the slider that can be seen in the following image. For an even better visualization, you may check out the demo videos that show this feature in action here: https://www.birdtools-developers.com/tagalign.html



Bird Tools' Tag Alignment Tool: Annotation Dashboard

An additional UI option one can go with is a dockable panel. Examples of built-in dockable panels include the project browser, system browser, and property panel. A custom panel can be built and docked anywhere next to these built-in panels.

Revit Events

To take your automation to an even higher level, you can monitor many of the built-in Revit events through the Revit API and execute an instance of automation as soon as these events are triggered.

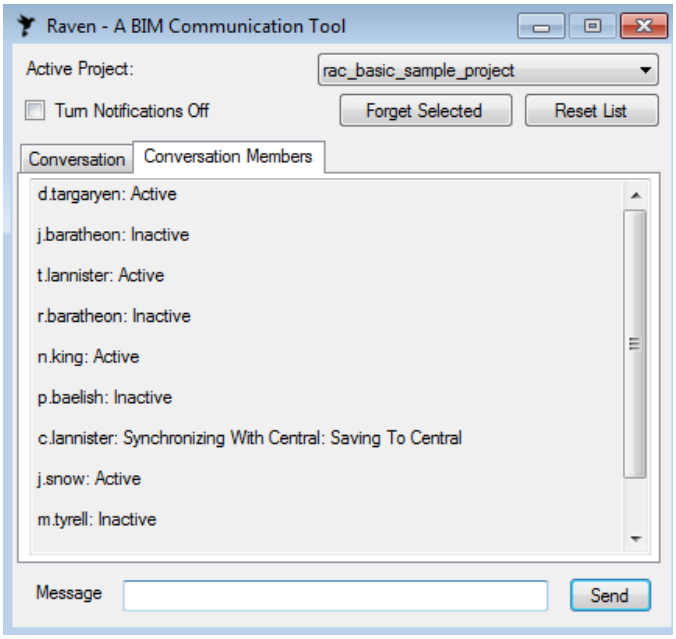
The best way to define Revit events and simplify the concept is to enumerate many of these events.

Events that are accessible through the Revit API include but are not limited to element placement, element modification, element deletion, document opening, document closing, document saving, document synchronizing with central, Revit application instance opening, Revit shutting down… Whenever one of these actions happens, an already defined monitor can be triggered to perform a defined action. Even more, it's also possible to perform custom actions when the model is completely idle through idling events.

Some of the potential applications of Revit events include auto-saving/synchronizing with central when a model is idle. Another great use of both Revit macros and events would be an embedded Revit document macro that sets the active workset automatically as soon as a model is opened.

Let's illustrate that with another real-case example that we worked on at Bird Tools. Back when BIM 360 Design was first introduced, the old Revit communicator was dropped and since the worksharing monitor doesn't support BIM 360 models, a worksharing monitor was needed for BIM 360.

After being requested by many of the Autodesk forum, we decided to build our own and make it public. It's called "Raven" and what it does is monitor worksharing events and report these events back to a central cloud hosted server, which in turn sends a notification to an instant messenger that the entire team uses, whether it's the instant messenger that comes with Raven, or whether it's any of the popular team communication platforms out there such as Zoom, Microsoft Teams, Slack, Webex Teams… This is a great example illustrating Revit events that can be found here for your own inspiration: https://www.birdtools-developers.com/raven.html



Raven, reporting worksharing notifications for a BIM Collaborate Pro Revit model

Getting Started with the Revit API

The best way to get started with the Revit API is to start converting your Dynamo scripts into Revit macros. While you do that, you can attend to many technical difficulties on the spot and if you're stuck, you can always look for a solution on the spot through a simple search over the internet (although that would be less fun).

To ease that journey, here's a list of online resources and tools that will come in handy:

- Revit API Docs – It's the official Revit API documentation converted into an online database: https://www.revitapidocs.com
- Jeremy Tammik's "The Building Coder", which features many code examples https://thebuildingcoder.typepad.com
- The Revit Developer Center, where one can download the official Revit SDK, which is full of examples and useful documentation: https://www.autodesk.com/developer-network/platform-technologies/revit
- Revit Lookup: This is a must have tool for developers and users alike, allowing users to explore elements within a Revit model and their hidden API properties and methods. Again, this is a must have! https://lookupbuilds.com
- Boost Your BIM, by Harry Mattison, one of the oldest Revit users and developers ever: https://boostyourbim.wordpress.com/
- The official Revit API forum, where many of the issues one would face have already been discussed and potentially solved: https://forums.autodesk.com/t5/revit-api-forum/bd-p/160

Part 2: Forge

This is no longer 2009. People nowadays are using cloud storage and cloud computing to collaborate in every domain, and the AEC sector is no exception.

Cloud services have been offered by Autodesk for a while now, from BIM Collaborate Pro (formerly known as BIM 360 Design), to Autodesk Docs, to the Autodesk Construction Cloud, to digital twin platforms such as Autodesk Tandem.

Ever wondered how these platforms are built? Ever wanted to automate any of your cloud workflows? One word answers both questions: Forge.

Forge is Autodesk's answer to cloud automation. It's a cloud developer platform that aims to automate anything cloud related.

Even Autodesk's own cloud services are actually built on top of Forge, and you can even build your own custom system on top of Forge.

To be able to automate all cloud services, and since there are several different cloud services offered by Autodesk, several APIs had to be conceived for each service, and they are all gathered in their masses under the Forge umbrella.

The following sections aim to enumerate these APIs, explain what each API is and what it can help you automate, along with a couple of tips for the reader to start diving into the realms of Forge automation, and a couple of real-case Forge automation success stories, both public, and privately developed by us at Bird Tools, to help demonstrate the powerful potential of Forge.

Forge BIM 360 API

The first API that we'll be talking about is the BIM 360 API. The BIM 360 API aims to automate anything BIM 360 related.

For instance, BIM 360 operations such as company creation, project creation, permission assignment for each user, activating services, cost management, issue management, issue creation, PDF export straight from BIM 360, Model Coordination and managing the clash results obtained within the BIM 360 model coordination feature, RFI management… can all be automated through the BIM 360 API.

Forge Data Management API

Simply put, the Data Management API is meant to give the developer control over all cloud storage and file transfer operations related to BIM 360 and Autodesk Docs.

For instance, if you ever wanted to automate or schedule file backup operations, then the Data Management API can allow you to automate and schedule model downloads.

Operations such as file uploads, attachments, publishing cloud models, deleting files and restoring deleted files, can all be automated through the Data Management API.

Forge Viewer

The Forge viewer, in its API aspect, consists of a 2D-3D JavaScript rendering library (based on Three.js). Through the Forge viewer, you can make your Autodesk compatible model, whether it's an AutoCAD 2D drawing, a Revit model, a Revit family, and Inventor part or assembly… compatible with any web browser, where anyone can access that model, navigate it and access its element properties.

Being compatible with any web browser also implies that your model is compatible with any platform, whether a Desktop one or a mobile one. You can thus create your own company's Common Data Environment (CDE) and share your data across locations, whether with office workers, or with site workers that can access all that on site though their own tablet devices.

With IoT integration within BIM being projected to be a trend in the upcoming years, Forge Viewer is already equipped with an IoT toolkit that can remotely integrate with any number of sensors/devices and report any captured data in real-time straight into the viewer, where that data would be displayed as sprites, heat maps, timelines… Such a feature is ideal in case you are not satisfied with any of the digital twin solutions commonly available, as it can be integrated within any bespoke Digital Twin platform you may conceive.

Forge Model Derivative API

Integrating a model within Forge Viewer requires another API to convert that model into a compatible file format. That file format is the SVF file format, and that API is the Forge Model Derivative API.

In addition to making your Revit model, AutoCAD drawing, Inventor part… compatible with Forge Viewer, the model derivative API can be a quick an easy way to convert and extract geometry information into other universal formats, such as OBJ or SAT.

Geometry extraction is not the only purpose of the Model Derivative API. In fact, any type of data can be extracted through this API. Examples include the extraction of element properties, such as information about rooms, spaces…

Webhooks

As stated above, monitoring events through the Revit API proved to be highly useful. Cloud events can also be monitored, and Forge includes a series of webhook events that can be setup to automatically trigger an action or send notifications to another listening server, or to another incoming webhook that users can setup within apps such as Slack, Zoom or Microsoft Teams.

Cloud events that can be monitored through these webhooks are file creation/modification/deletion, folder creation/modification/deletion, model modification, synchronization, model publish operations, model version modification… You can, for example, receive a notification inside Microsoft Teams as soon as a model is published.

Forge Reality Capture API

This API is an interesting API as it's one that uses machine learning to perform many of its functionalities.

In fact, this API aims to handle and convert raster images into geometry elements supported by common CAD platforms, such as meshes, or other Reality Capture Data formats, such as point clouds.

It is ideal to convert satellite images or aerial images into elements such as 3D topography for example.

Forge Design Automation API

Last but definitely not least, this API is the one that is the most disruptive in my opinion.

The Forge Design Automation API is in itself a service that can run a software instance remotely over a cloud remote server.

Think of it as a remote machine that can run AutoCAD, Revit, Inventor, 3DS Max over models without you having to cache them locally or download them, and that has very high resources: a powerful processor, 32 GB of memory as of October 2021, and almost infinite storage…

The potential of this API is just enormous. Not just does it eliminate the time wasted to download and synchronize models back, and not just does it eliminate the time needed to open and close and display a user interface, it also does everything in less time as its server is way more powerful than anything desktop based, and its interface is not limited to the Revit/AutoCAD…interface, nor to any platform: the interface can be anything your heart desires, including a web page that can be accessed from any device. If you're dedicated enough, you can combine it with Forge Viewer to build your own web version of Revit. More examples about this API will be presented in the following sections.

Getting Started with Forge

Now that almost every Forge API has been presented in detail, it is only natural to start exploring these APIs and building on top of them. Here are some of the resources and tutorials to be followed, and some of the tips based on my own experience with Forge.

The first step one needs to take when starting to build his/her own Forge solution is determining which functionalities he/she needs and in turn determine which APIs are needed to create that

functionality. For example, if one needs to batch assign user permissions within BIM 360, then he/she needs to refer to the BIM 360 API. The previous section should be able to give you a clear idea about each API and should help you with that step.

Once you've determined which API is required for your First Forge solution, it would be time to start learning how to build that solution based on the chosen API. As per Paolo Coelho's masterpiece, "The Alchemist", "there's only one way to learn. It's through action", which in our case means that the best way to learn how to develop on top of Forge is to actually start with the project and learn as you move along.

For that, Autodesk has provided some awesome step-by-step tutorials that you can literally follow to develop your own solution. Each of the functionalities that can be automated through the Forge API in question has its own step-by-step tutorial that you can follow to build your own solution. You can find these step-by-step tutorials here:
https://forge.autodesk.com/developer/documentation
https://learnforge.autodesk.io

One great place to check as well is the Forge Community Blog. Here's a link:
https://forge.autodesk.com/blog

If you have questions, stackoverflow.com is a great place to ask them. Make sure to tag your inquiries with "autodesk-forge": https://stackoverflow.com/questions/ask?tags=[autodesk-forge]

Here are some tips to ease that for you: when you go through the step-by-step tutorials, something that you will encounter a lot is cURL. cURL is a multi-platform command line tool allowing you to test all the web calls and data transfer operations that all of the Forge APIs involve. If you're a windows user you can find a windows version of cURL that you can integrate within your system and use as per the step-by-step tutorials.

However, it is important that you keep an open mind about that as all of the cURL syntaxes you may encounter are actually universal REST requests that can be performed using any programming language, whether JavaScript, C#, Python, or any other language. The most recommended one would definitely be JavaScript as it's the most universal. Therefore, make sure to try and write the equivalent of those cURL statements in your own language. To get you started with that, here's the equivalent of the 2-legged token cURL approach featured in the step-by-step tutorial, in C#, JavaScript, and Python. Make sure to replace your client id and client secret with your own of course.

Curl:

```
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
  '
```

C#:
(make sure to add references to the System.Net, System.Net.HTTP, and System.Net.Http.Headers libraries and namespaces)

```
using (var httpClient = new HttpClient())
          {
                using (var request = new HttpRequestMessage(new
HttpMethod("POST"),
"https://developer.api.autodesk.com/authentication/v1/authenticate"))
                {
                    var contentList = new List<string>();
                    contentList.Add("client_id=" +
“obQDn8P0GanGFQha4ngKKVWcxwyvFAGE”);
                    contentList.Add("client_secret=" + “eUruM8HRyc7BAQ1e”);
                    contentList.Add("grant_type=client_credentials");
                    contentList.Add("scope=code:all data:write data:read
bucket:create bucket:delete");
                    request.Content = new StringContent(string.Join("&",
contentList));
                    request.Content.Headers.ContentType = new
MediaTypeHeaderValue("application/x-www-form-urlencoded");

                    ServicePointManager.SecurityProtocol =
SecurityProtocolType.Tls12;
                    HttpResponseMessage response = await
httpClient.SendAsync(request);
                    string responses = await
response.Content.ReadAsStringAsync();

                }

            }
```

JavaScript:

```
var xhttp = new XMLHttpRequest();


xhttp.onreadystatechange = function() {
      //do your additional work here
    };


xhttp.open("POST",
"https://developer.api.autodesk.com/authentication/v1/authenticate", true);
    xhttp.setRequestHeader("Content-type", "application/x-www-form-
urlencoded");
    xhttp.send("client_id= obQDn8P0GanGFQha4ngKKVWcxwyvFAGE&client_secret=
eUruM8HRyc7BAQ1e&grant_type=client_credentials&scope=code:all data:write
data:read bucket:create bucket:delete");
```

Python:

```
import requests

headers = {
    'Content-Type': 'application/x-www-form-urlencoded',
}

data = {
  'client_id': 'obQDn8P0GanGFQha4ngKKVWcxwyvFAGE',
  ' client_secret': 'eUruM8HRyc7BAQ1e',
  ' grant_type': 'client_credentials',
  ' scope': 'data:read'
}

response =
requests.post('https://developer.api.autodesk.com/authentication/v1/authentic
ate', headers=headers, data=data)
```

So keep in mind that you can perform these requests using any programming language you prefer. You will encounter the cURL statements for all of the step-by-step tutorials of all Forge APIs so hopefully the sample conversions above of the very first tutorial will be of help. You can proceed the same way for the rest of the tutorials and convert the cURL statements into your preferred programming language in a similar manner.

On Forge Design Automation for Revit: DA4R apps consist of two components: a back-end application and a front-end interface. The front-end interface can be anything you want: a web page, a mobile application, a desktop application… the back-end application is essentially a Revit add-in stripped away from any reference to the “RevitAPIUI” library, as the UI part is handled by the front-end user interface that is developed separately. Setting up the server will require REST requests similar to the ones above. These can be found in the DA4R step-by-step tutorials. As of the date when this document was written, Dynamo scripts can’t be executed within DA4R and won’t be for a period of time, which is an additional reason to dive deeper into the Revit API and master Revit plugin development, as the transition from a Revit plugin into a DA4R application is rather easy when you are a master of the Revit API.

Real-Case Forge Success Stories

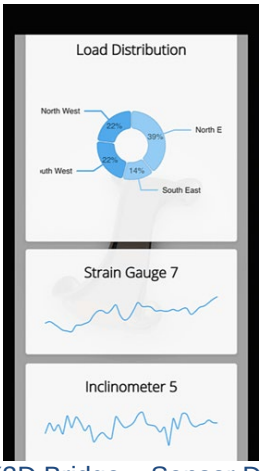
Wherever you may wander over the internet these days, you may here the expression “Forge is the future” which is totally inaccurate: Forge is here. Many R&D projects have already emerged that are Forge dependent, and, according to my own daily experience, many firms have already started reaching out to integrate bespoke internal Forge solutions, and many of these projects have been completed for a while now. These projects include automating BIM 360 operations such as project creation and permission assignment, automatic cloud data backup applications, customized Common Data Environments (CDEs), remote cloud model processing applications that perform remote automated operations over cloud models without locally opening and caching the models in question…

Throughout this section will be showcased four real-case projects that have been built on top of Forge and its APIs, three of which I have worked on personally.

MX3D Bridge – Amsterdam

This is an R&D project that you are most likely familiar with as it has been all over the news. Probably the first robotically 3D printed structural entity ever created, this bridge is located in Amsterdam and is a smart structure in a sense that it is monitored by all types of sensors. What you may not be familiar with is that a customized digital twin platform for this bridge was developed by the Autodesk research team as part of Project Dasher, and that this platform itself is built on top of Forge. All the data that these sensors pick up is processed by that platform, and can be accessed live by anyone. You can check it out here: https://www.smartbridgeamsterdam.com/

MX3D Bridge Inauguration



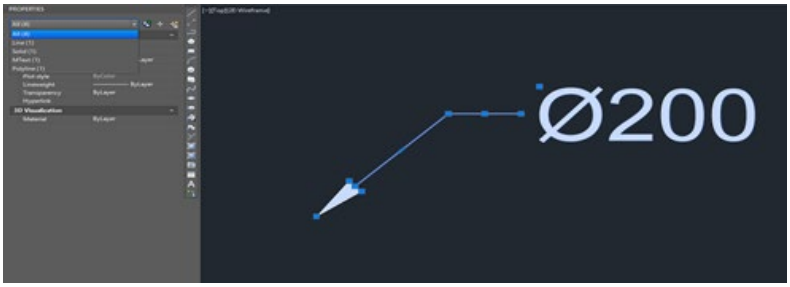
MX3D Bridge – Sensor Data

From AU 2020: EMDC Group's Forge Integration Success Story

For those of you who have missed last year's AU class, the whole session was dedicated to showcase the following two projects that we have collaborated on with EMDC Group and to discuss the machine learning algorithms and Forge technologies involved with these projects.

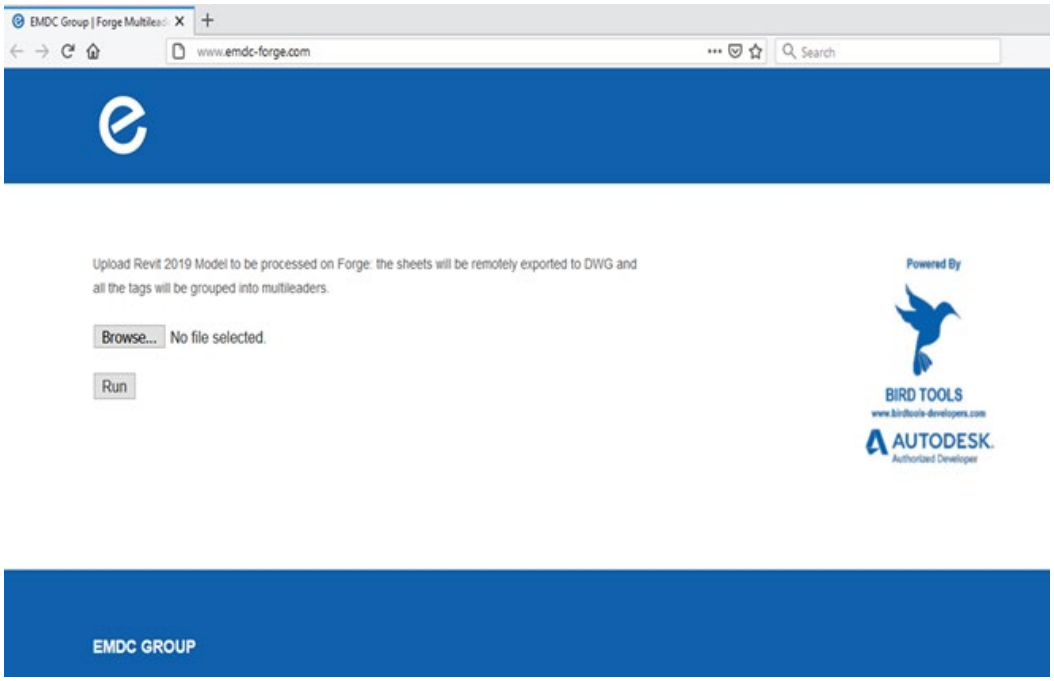
It's hard to find a firm that embraces automation as much as EMDC Group and one that is open to sharing its internal automation workflow, as many firms are a bit reserved regarding to that and try to keep everything confidential, and that early adoption of automation and emerging technologies and sharing spirit is one of the secrets of EMDC Group's success as a leading design and consultancy firm in the MENA region, in addition to them being hard-workers and to their vast experience in the field.

One of the challenges that EMDC Group faced a couple of years ago was automating the issuing of their deliverables for a huge project consisting of tens of thousands of sheets. The project they modeled consisted of a series of Revit models which is only natural these days, while the deliverables were required to be submitted as AutoCAD drawings. A simple export may seem to be the answer but specific DWG standards were required so a custom exporter was required. Among these standards, having the tags submitted as multi-leaders and according to a certain multileader style was a major requirement. Revit out of the box exports these as exploded entities and the manual work involved with regrouping the already placed Revit tags was a pure waste of time.



Exported Revit Tag – Entities not grouped into multileaders

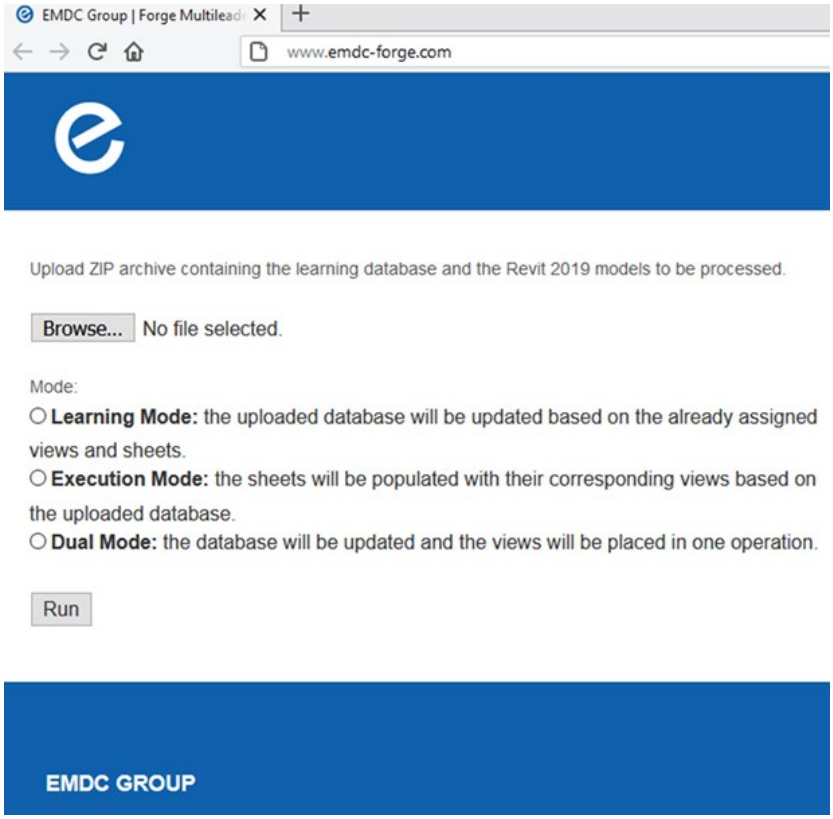
The solution developed used machine learning, particularly clustering, to regroup the tags as soon as the drawings were exported to avoid any manual labor, and due to the high number of Revit models involved and to the fact that this involved interoperability between Revit and AutoCAD which is ideal for Forge Design Automation as it reduces the time needed to open and close the drawings, the final solution ended up being an interoperability Forge Design Automation application for both Revit and AutoCAD, while its interface was a web page that any team member could access from anywhere to submit the required drawings, which in turn are sent automatically to the client.



Forge Multileader Processor – Web Interface

Another application where machine learning has been combined with Forge to automate project setup is the Forge View Placement App, which was built on top of the Forge Data Management API to retrieve the models that were used as a learning set, that was in turn fed into a supervised learning Forge DA4R application that was used to find patterns and correlations between view parameters and the parameters of the hosting sheets. The goal was to use fuzzy logic within Forge DA4R to place views on sheets automatically and without any human interaction, based on the statistical relation found by the first supervised machine learning application. DA4R was definitely the way to go for this project as thousands of sheets needed to be processed and hundreds of models needed to be automatically processed, which implied a great amount of time that DA4R could save by reducing operations such as downloading, opening, and closing models.

For more information about these two apps and more details about the technologies used to build them, you may check out this highly insightful AU 2020 class: https://www.autodesk.com/autodesk-university/class/Combining-Forge-and-Machine-Learning-Automate-Time-Consuming-Tasks-2020



Forge Sheet Generator – Web Interface

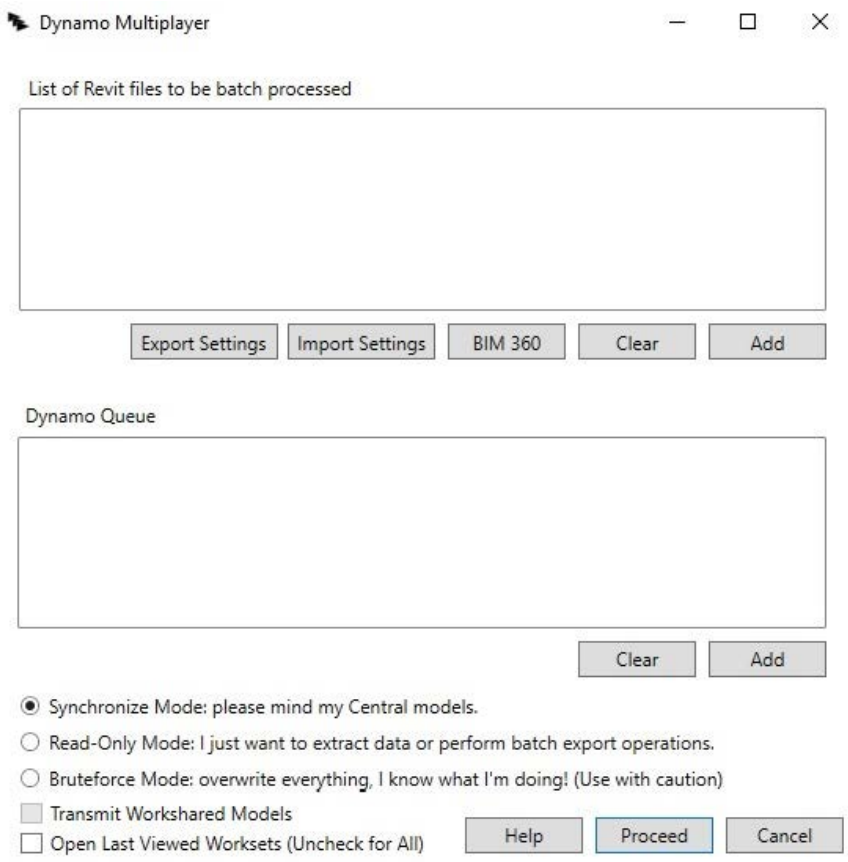
Bonus Example: Dynamo Multiplayer

This example is a bonus example that wasn't featured in the original video presentation as the app wasn't released yet back when the session was recorded.

Batch processing is one of the most highly demanded features for any software, and for Revit, building a universal batch processor is a challenge as Revit's commands involve interface interactions, unlike a product such as AutoCAD where all commands are script based.

In order to achieve a universal Revit batch processing tool, what we built was a tool that can batch open and save Revit files, and which can run a sequence of Dynamo scripts to process files individually. That tool was called Dynamo Multiplayer.

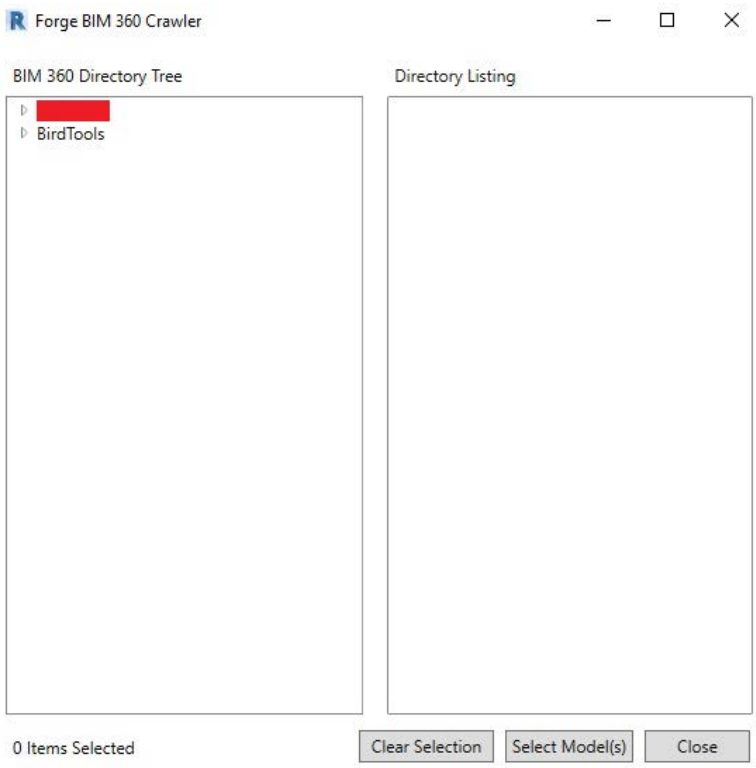
Building the tool is not too much of a challenge for offline models as the path can be accessed through a regular file browsing dialog.



Dynamo Multiplayer - Interface

However, most people are nowadays using BIM 360, and unlike other types of Revit files, these are remotely hosted and can't be accessed through the legacy file browsing method.

The way to access these files is to use Forge Data Management API to retrieve the BIM 360 hubs and project directories and model cloud paths, along with the BIM 360 API to retrieve BIM 360 projects which the user has access to and a high enough permission level to edit, which is exactly what has been built within this app: a customized file and directory browser that can list BIM 360 hubs, models and directories.



Dynamo Multiplayer – Forge BIM 360 Directory and File Crawler

This is a great example of an integrated Forge workflow within a Revit extension, and of the possibility of combining all of the three technologies that are discussed throughout this document (Dynamo, Revit API, and Forge) into one solution, and that showcases the power of Forge to at least retrieve BIM 360 models and list BIM 360 directories, and that demonstrates how an advanced knowledge in both the Dynamo API and the Revit API is capable of pushing Revit extensions to the next level and to even extend the built-in capabilities of Dynamo.

The app is a free public Revit add-in called Dynamo Multiplayer. If you want to check out its videos and give it a test to get a feel of the new possibilities that Forge offers when integrated within a Revit API workflow, you can find it here: https://www.birdtools-developers.com/dmu.html

Conclusion

To conclude all of the above, this document's main aim is to invite everyone to go as deep as possible into the realms of automation and to try to inspire you with new ideas by the real case examples presented above: as much as you may know, there's always more for you to learn, and automation opportunities are endless.

If you haven't explored Dynamo yet and are still new to automation, you have to. Dynamo is just great and will turn you into an internal superhero at your company and give you superpowers. Hop on that train today.

If you already have Dynamo superpowers, there are "megapower" hidden within the Revit API and "gigapowers" hidden within Forge that you are yet to explore. As you can see, the AEC world has started to integrate Forge solutions so if you're postponing that, you're going to be left behind.

If you prefer to focus on the technical or managerial aspects of your jobs in case that is your passion rather than automation, and although I recommend that you add automation to your skills as well because it's not just useful, it's also entertaining and may become your new hobby, you may find someone to do it for you. It may be a team member, or a third-party Autodesk Authorized Developer or Forge Systems Integrator. Create the idea, start the project, and find the entity that suits you best and have them do it for you.

Whatever you have to do, just make sure to hop on the automation train today, because, as per Tom Preston-Werner (founder of GitHub), "**you're either the one that creates the automation or you're getting automated**".

References

- https://www.shutterstock.com/image-vector/evolution-human-primitive-present-stone-age-1129231658
- https://chucknorris.io
- https://nodered.org/
- https://gettingsimple.com/ian-keough
- https://i.pinimg.com/474x/d4/55/02/d4550223009d9f452cb474ae6517269e.jpg
- https://www.gannett-cdn.com/-mm-/993aafba25524310832cb4f280d0c82221b7cbdf/c=0-0-1996-1127/local/-/media/2015/09/02/Phoenix/Phoenix/635768148971498776-henry-ford.jpg
- https://w9spd3lpn271hexb03czvhiy-wpengine.netdna-ssl.com/wp-content/uploads/2020/01/Screenshot-2020-01-14-at-3.42.12-PM.png
- https://www.azquotes.com/quote/621568?ref=think-outside-the-box
- https://www.revitapidocs.com/

- https://upload.wikimedia.org/wikipedia/commons/thumb/2/25/Egypt_Hieroglyphe4.jpg/260px-Egypt_Hieroglyphe4.jpg
- https://bubble.io/blog/visual-programming/
- https://encrypted-tbn0.gstatic.com/images?q=tbn:ANd9GcRnEs3VtgkCtc-EyJXY1RQHGW-XJAmmuTZtzA&usqp=CAU
- https://help.autodesk.com/cloudhelp/2014/ENU/Revit/images/GUID-3E56B42A-41D4-48C0-B7B4-E4FC79B613B2.jpg
- https://dynamobim.org/wp-content/uploads/forum-assets/martin-staceymatterlab-co/03/26/revit02.png
- https://www.darrenjyoung.com/wp-content/uploads/2020/02/Revit_Dynamo.jpg
- https://www.alliedmarketresearch.com/bim-in-construction-market-A10290
- https://www.maxpixel.net/static/photo/1x/Party-Event-party-Night-party-Events-Celebration-3005668.jpg
- https://www.techture.global/wp-content/uploads/2018/03/Autodesk-Forge-big.jpg
- https://static.thenounproject.com/png/1581158-200.png
- https://labs.blogs.com/.a/6a00d8341caed853ef0240a4483ba2200c-pi
- https://developer-dev.static.autodesk.com/coverpage_images/bim1.png
- https://developer.doc.autodesk.com/bPlouYTd/475/_images/dms_overview.png
- https://developer.doc.autodesk.com/bPlouYTd/475/_images/MD-overview-diagram.png
- https://developer.doc.autodesk.com/bPlouYTd/412/_images/overview1.jpg
- https://developer-dev.static.autodesk.com/coverpage_images/wh_1.png
- https://developer.doc.autodesk.com/bPlouYTd/475/_images/reality_capture_high_level.png
- https://www.azquotes.com/picture-quotes/quote-i-must-create-a-system-or-be-enslaved-by-another-mans-i-will-not-reason-and-compare-william-blake-2-88-41.jpg
- https://through-the-interface.typepad.com/.a/6a00d83452464869e20282e111e0e2200b-pi
- https://www.smartbridgeamsterdam.com/wp-content/uploads/2021/07/MX3D_Bridge_Opening_YourMajestyTheQueenMaxima_ByAdriaandeGroot-scaled.jpg