Controlling MEP Fabrication Parts in Revit by Using the API and Dynamo

Travis Voss
Mechanical Inc.

Luke Vetroczky
Mechanical Inc.

Learning Objectives

- Use templates with good database management to create deliverables.
- Read AND write data to fabrication parts within Revit using dynamo and the API
- Extract points from Revit using out of the box tools. No add-ins
- Create accurate BOMs with the Fabrication Reports tool.

Description

The goal of this class is to show users how they can model more accurately using fabrication parts in Revit by using the power of Dynamo, the Revit API, and some good Fabrication CAD database management practices. For time sake, this demo will focus on the issues Mechanical Inc incurred when modeling out MEP Fabrication hangers. A series of problems will be outlined including producing hanger drawings, extracting point data out of Revit for field install, and producing accurate material BOMs using fabrication reports. For each problem we will discuss how that problem came about, the issues it was causing, what our solution for it is, and how we developed that solution natively in Revit and Fabrication CAD without the use of paid add-ins. While the workflows being demonstrated will focus mainly on MEP hangers, the workflows demonstrated can be applied to all categories of MEP Fabrication elements and used to overcome current limitations with fabrication parts that come out of the box.

Speaker(s)

Travis Voss came into the construction industry almost four years ago from the software development world and is currently serving as the Leader of Innovative Technology at Mechanical, Inc. Travis uses his experience in research, development, and the application of emerging technologies toward helping Mechanical, Inc. lead the construction industry transformation.

tvoss@mechinc.com

Luke Vetroczky is a project engineer for Mechanical Inc. While pursuing his bachelors in mechanical engineering he broke into the construction world working with multiple engineering design firms. After finishing graduate he started at Mechanical Inc. As a project engineer there he works as a detailer, an EIT, and was responsible for training Mechanicals VDC (virtual design in construction) department in Revit. He is also a representative of Mechanical in a group of peer companies VDC think-tank, where VDC experts from a group of non-competing contractors around the country meet and share workflows and strategies about BIM and technology implementation in the construction world.

lvetroczky@mechinc.com
Controlling MEP Fabrication Parts in Revit by Using the API and Dynamo

**TABLE OF CONTENTS:**

| Controlling MEP Fabrication Parts in Revit by Using the API and Dynamo | 1 |

**Learning Objectives**

| Learning Objectives | 1 |

**Description**

| Description | 1 |

**Speaker(s)**

| Speaker(s) | 1 |

---

**TABLE OF CONTENTS:**

| Introduction | 2 |

- Why Revit? Why Fabrication? | 3 |
- Overview of Class Contents | 3 |

**Issue 1 – Creating Deliverables using Templates and Database Management**

| Problem | 5 |

- A Two-Sided Solution | 5 |
- FAB Database Approaches | 5 |
- Revit Approaches | 6 |

**Issue 2 – Reading/Writing Data to ITM Parts**

| Solution | 7 |

- What Data is there? | 7 |
- Setting Rod Extension | 8 |
- Renumbering | 12 |

**Fabrication Bolognese – A Dynamo Package**

| Solution | 13 |

- A Dynamo Package | 13 |

**Issue 3 – Exporting Points for Layout**

| Solution | 14 |

- API and Dynamo for Hangers | 14 |
- Adaptive Point Family | 16 |

**Issue 4 – Creating Accurate BOMs w/ Fabrication Reports**

| Solution | 17 |

- Careful Use of Custom Data | 17 |
- A Look Back at the Renumber Tool | 18 |

**Conclusion – A Custom Tailored Suit**

| Solution | 18 |
Introduction

We all know by this point that creating a high LOD (level of detail) BIM model is becoming a regular necessity in the world of the fabricating contractor. The clear lines between conceptual design, construction documents, and fabrication are more and more blurred with each job. One-way fabricating contractors are adapting to this trend is by transitioning into Revit for the production of fabrication models. In Revit 2016, fabrication parts were introduced into the software, giving contractors an out-of-the-box method of using Revit to produce fabrication level ITM parts within Revit models natively. As expected with its first release, there were some pitfalls to using this workflow. But through each new version implementing fabrication parts in Revit has improved, and the benefits are starting to outweigh the drawbacks. However, there are still drawbacks that need to be addressed. In this instructional demo, we will look workflows that can be used to overcome these drawbacks from the viewpoint of a mechanical contractor.

Mechanical Inc is a self-performing mechanical contractor in the midwestern United States. With 200,000sf of fabrication space, Mechanical self performs HVAC, plumbing, sheet metal, process piping, waste water, and aquatic work as well as performing facility management, building automation, and other services.

Why Revit? Why Fabrication?

So why even make the switch? Fabrication CAD, CAMduct, and ESTmep have served fabricators well for years. Well, so did ammonia-based blueprints in the 1800s. The discussion over which is better, CAD or Revit, is subjective at best and could be its own topic for a class of this nature. But there is undoubtable value in keeping a BIM model under one platform thorough the life of a project. So much so, it is not uncommon to see the software as a requirement in the terms of construction contracts. If fabricating contractors are not choosing to switch to Revit, it is being mandated by their contracts.

Fabrication products strength lies within its database and its configurations. Information can be packed into these databases and end up on the plasma table and being cut into sheet metal fittings. The fabrication parts provide a link to the physical fabrication shops at a level of detail that is hard to match with pure Revit families alone. The database of parts can also be directly tied into estimating software (ESTmep) allowing contractors to make more accurate estimates by building the costs of a job into the model.

Overview of Class Contents

The goal of this class is not to convince us that detailing in Revit is worth doing, or to go over the pros and cons of doing so. Neither is the goal to teach users how to do things like create new fabrication services in FabricationCAD or workflows for modeling fabrication parts within Revit. The goal of this class is to help those contractors who have already made the jump to Revit by showing them some of the more complicated pitfalls we at Mechanical Inc. have had using fabrication parts and how they were overcome. In the interest of making the class as accessible to as many types of contractors as possible, the issues will focus on those pertaining to hangers. Not every MEP contractor practices the same trade but many, if not all of them, use
hangers of some kind. We will look at four problems Mechanical Inc. has had to overcome using a real-world healthcare project as an example.

It should be noted that these examples reflect the setup of Mechanical Inc’s fabrication database. Due to the large amount of customization and configurations possible in these databases, the exact methods may not be reflective of the readers actual setup. However, the general methodology should still be applicable. For example, Mechanical Inc’s parameter A may be the equivalent of the readers parameter B. If adapted by the reader, the workflows demonstrated will need to be adapted for these differences. The example issues will be as follows.

1. Creating hanger deliverables for field personnel using good templates and some fabrication database standardization.
2. Using Dynamo and the Revit API to read and write fabrication database data to hangers to ensure correct modeling and attachment to structure.
3. Extract point coordinates of hanger rods, sleeves, and other elements to a CSV file for use point layout machines using some Dynamo and reading of fabrication part data.
4. Use fabrication reports to create accurate BOMs for hanger fabrication and installation.
Issue 1 – Creating Deliverables using Templates and Database Management

Problem

One of the first problems we faced with hangers is how to quickly develop deliverables across jobs that have different types of hangers on different services and be able to adapt those deliverables to changing field conditions. An easy way was needed to identify hangers that were modified after being sent out to the field due to field changes. In CAD, the workflow was simply to overwrite the color, and fabrication parts in Revit offer limited access to modifiable parameters. While graphic overwrites are possible in Revit, project parameters offer a more intricate solution.

This problem stemmed from one of the largest issues users will face when adapting fabrication parts, the lack of controllable parameters. Many parts such as fittings, pipe, and duct patterns have access to the “Edit Part” button, which allows for users to modify dimension and connector options. Hangers however, are the most limited category of fabrication parts offering almost no configurable options outside of the product entry, and even that is normally locked when properly hosted to a fabrication element.

A Two-Sided Solution

Our solution involved setting up basic templates to control overall visibility, and a few project parameters used along side filters to track modified hangers. This was backed up by a heavily standardized database to keep filters robust.

FAB Database Approaches

The first step in getting deliverables ready is not a hanger specific step, and takes place within the fabrication database setup. Standardize. We start every new fabrication service based off of a generic service template for each common system type. What this enables us to do is build in key values that will be common across common service types. In the snip below you can see how a common format is used for every single service. Job number and job name will change from job to job, but the final section of the service name remains constant. For example, every heating hot water supply service on all jobs will end with “-HHWS” in the service name. Having this consistency gives something for our Revit templates to grab onto without needing to adjust the templates from job to job.
Revit Approaches

Now, with standardized services, we can build templates around the keys built into the database. Filter rules can grab onto these keys as well and be used for coloring services and so on. Be sure to distinguish between the parameters “Fabrication Service” and Fabrication Service Name”. The former parameter includes the entire database group with the service name appended to the end the later parameter is just the service name from the database. For filtering, “Fabrication Service” is often the better parameter.

Another way we deal with the lack of accessible data with fabrication parts is by creating project parameters. If you have never used **project parameters** before you allow you to add parameters to elements at the category level. Since MEP Fabrication Hangers is its own category, we were able to use a project parameter to add a shared parameter to all hangers in the job. This parameter, named “MI_Altered Hanger”, is a yes/no parameter that works as a toggle for detailers to mark specific hangers that have been altered by the coordination team after field points have been shot and therefore require dimensions. Adding filters to our hanger view templates changes their colors making them easily identifiable (see figure 1). The lead detailer can coordinate and mark the necessary hangers, and then hand the view of to a junior detailer to be dimensioned. Because it is also a shared parameter, these hangers can be tracked on separate schedules and BOMs if need be.

Another example of using project parameters is our “MI_Exclude from Navis” parameter. Unlike the altered hanger parameter this one is applied to most of the mechanical equipment categories and is another yes/no toggle. This one is used in
conjunction with a filter on our Navisworks exporting template to hide elements in the exporting view so they will not be brought into a coordinating Navisworks model.

**Issue 2 – Reading/Writing Data to ITM Parts**

**Problem**

Adding shared parameters is one way to overcome the data shortage in fabrication hangers, but it didn't work for all our issues. The most immediate problem we noticed with fabrication hangers was how they attach to structure. MEP Fabrication Hangers using CID patterns with rods will automatically extend those rods up to the nearest Revit structural element. The issue was that this is just now how hangers are installed in the field. Depending on the type of anchor being used those hanger rods should extend slightly into the structure or may stop slightly before the structure. The rod extension option that is accessible in fabrication CAD could not be accessed in Revit. For our shop that prefabricates hangers, this was a problem. BOMs and hanger cut reports were inaccurate and required calculated fields that would need to be adjusted from job to job.

**Solution**

This is where the cheap tricks like project parameter and templates stopped cutting it and we needed to break out dynamo and the Revit API. As it turns out the ability to adjust the hanger rod extension is built into the API, it has just not been developed into Revit out of the box yet. There is lots of data hidden that can be used.

**What Data is there?**

Before we could start developing a solution, we needed to know what fabrication parts data was accessible from the loaded configuration. One of the best, and free, add-ins for doing this is called Revit Lookup by Jeremy Tammik. (He hosts the .msi on google drive so I wont host it in this PDF, but a basic google search should yield results). The second thing needed to start diving into data is the Revit API. The best way to navigate the API for new users is www.revitapidocs.com. It is a searchable guide to the Revit API and lets you see what methods are at your disposal for mining fabrication parts for data. A good place to start is the Fabrication Parts Class. Investigating this class one available property is “ItemCustomId”. The Revit lookup tool can be used to see this properties value on an element within Revit. Selectomg an element and using the add-in to snoop current selection will use the api and attempt to pull all the properties.

![Figure 4: The ItemCustomId property for fabrication parts as shown on the Revit api docs website.](image-url)
and methods possible for that element. For example, snooping a hanger and checking for the ItemCustomId field we can see there is a value. Database managers will recognize this value as the CID pattern number used by ITM parts. This CID number, which is one of the core building blocks of ITM parts is not normally available within Revit by itself, but the data is still contained within the part and can be accessed via the API, and used with Dynamo.

**Setting Rod Extension**

*SetRodStructureExtension Method*

FabricationRodInfo Class | See Also
--- | ---
Set the length of the rod's top extension into structure. The rod must be attached to structure.

Syntax

C#

```csharp
public bool SetRodStructureExtension(
    int rodIndex,
    double extension
)
```

One of the other things available in the API is a method in the Fabrication Rod Info class called “SetRodStructureExtension”. It can set the length of a fabrication hanger rod’s top extension into the structure. Exactly what we needed to solve our prefabrication problem. Implementing this method however requires some python development and the use of dynamo to execute it.
As a reminder, this class is not a dynamo training class. So if you are new to dynamo I would recommend going to https://dynamobim.org, dynamo’s home website. It is a great place to learn how to use dynamo to speed up essentially everything you do in Revit. We have found in our experience that if a task is mundane and repetitive, it can almost always be made obsolete with a good dynamo script. This class does not cover the basics of dynamo and it is assumed the reader knows how to build a normal graph. With that being said, let’s look at the solution we developed for our hanger problem.

At the time of developing these solutions, there were only three dynamo packages available that were developed for fabrication parts. “DynaFabrication2017”, “DynaFabrication2018”, and “Fabrication API”. Each of these are available in the packages browser for download and are an arguable a necessity if you will be using dynamo with fabrication parts. However none of these three contain a node that accesses the method for extending rods into the structure. A new dynamo script with custom nodes was needed.

Below are snapshots of the graph we developed that applies rod extensions into structure. The captions from each snip describe what each section does.

Figure 8: The overall graph

Figure 7: The IsSameAs node from the Fabrication API dynamo package. An incredibly useful node for grouping identical fabrication parts into sublists.
Figure 9: The input formating for the graph. We are using the Data-Shapes package for its custom input forms. This is optional, but it provides an easier to use GUI than the dynamo players input method. Using the dynamo player inputs can work just as well.

Figure 10: The first step takes the inputed list of hangers and checks to see if they are actually attached to structure. Those not attached are removed from the list.
The next step takes in input extension depth in inches, converts it to decimal feet, and sends it along with the part list to the custom node that executes the extension method from the API. At this point the hanger rods are extended into structure, but we’re not done yet. The input depth can also be negative, if the rod needs to pull away from the structure instead of insert into it.

At the end of the script are two checks. The first check simply returns the value of the extension and is used to ensure the rods are actually extending. The second check returns the number of hangers that were excluded.
from the tool because they were not attached to a structural element to a dialogue box that displays when the script is complete.

The most popular method of inputting for scripts used by our detailers is simply input by selection. Detailers can select one or more areas of hangers using the data-shapes package nodes (again this package is optional, there are many, many ways to get inputs). Elements that are not hangers are filtered out in the input process, so that only MEP Fabrication Hanger elements are processed. Other methods of input are possible, such as getting all hangers in a view, a scope box, or a selection set.

With this tool we are able to accurately model our hangers and produce cut sheets for our fabrication shops with correct rod lengths for cutting.

### Renumbering

We can also access custom data fields and both read and write to these fields. A method we have implied in another tool built to renumber hangers sequentially.

In our case, we use a custom data text field on all of our hanger buttons to give each hanger an identifying two letter abbreviation. A clevis hanger is designated with a CH for example. (Many databases may use the alias field for this purpose. We use the alias field for unrelated purposes in reports. But the values from the alias field can be pulled just like the custom data field in this example if that is what you use in your database) Just like the example above, a method from the API can be used to read custom data. There is a node in the Fabrication API dynamo package that is called “Get Custom Data” but it appears to be incomplete as it only returns dimension data. As a result another custom node was created to read the custom data fields.

Figure 15 shows a part of our renumbering tool. A list of hanger elements is inputed into a node that reads the custom data and adds it to the front of the number sequence been generated by the list of hangers. Each hanger then has an identifying tag added to its number in the form of the custom datafield mentioned above. Fabrication hangers are not families, so there is no “Type Mark” parameter to build a proper label with. Generating your own numbers lets you apply the number to parameters like “Mark”
for tagging, or “Item Number” which can be carried across MAJ exports of fabrication parts.

Figure 15: Part of a renumbering script that pulls custom data from the database to be used in creating marks for hangers.

**Fabrication Bolognese – A Dynamo Package**

Now at this point we have mentioned a few custom nodes, but have only shown how to use them and not so much how to make them. Well it is possible to dig into them yourselves. All the custom nodes we have developed in our efforts have been published in the fourth package for use with fabrication parts, Fabrication Bolognese.

It is called Fabrication Bolognese for a few reasons. First, dynamo package names tend to be creative, and second, most of my code looks like meat sauce. That is the one joke made in this handout so enjoy it.

It doesn’t contain a massive library of nodes like some packages, but the most useful are outlined below.

- **FB_Read/Write Custom Data** – These two nodes can be used to read and write from custom data fields of fabrication parts. The index of the custom data field must be known. Also, these two nodes can currently only read and write custom data fields set up for text. A simple change of the API method used can alter them for integer or double values if need be.

- **FB_FAB Ancillary Usage** – This node pulls all the ancillaries from the fabrication part. This can be very useful when dealing with hangers that use ancillaries, or ancillary kits, to govern hanger rods. Hanger rod counts, size, fasteners, and fixings, can all be retrieved with this node and brought into Revit.
• FB_Fab Part CID/NOT CID Filter – These nodes can take a list of any kind of elements and filter them down to only fabrication parts. The CID filter is an optional step that will further filter down the list to fabrication items that only match, or only do not match, the input CID number depending on which node is used.

• FB_Set Hanger Struc Extension – Can be used with an input list of fabrication hangers to set hanger rod extensions into (or away from) their host structures.

These nodes as well as a few others can be found in the package on the package browser in dynamo.

**Issue 3 – Exporting Points for Layout**

**Problem**
Now that we have some tools at our disposal, we can look at one of the more complicated issues fabricators face in Revit. Point layout. How can we take point coordinates of hangers and other items like sleeves and equipment slabs and export them to point layout machines for the field to layout?

**Solution**
Many third-party add-ins use a method of creating a common point family of some kind that is nested within other families and then read by the add-in. The coordinates for the nested point family are recorded and exported to whatever software the add-in is made for. Unfortunately, hangers are not families, and therefore cannot have a point family nested in them to be read by an add-in or dynamo tool. Fortunately, there is another API method that allows us to get the point coordinates of hanger rods.

**API and Dynamo for Hangers**
The API method GetRodEndPosition in the FabricationRodInfo Class can be used to get the coordinates of hanger rods just as if there was a point family on them. The node “Part(s) Rod Length” from the DynaFabrication 2018 package can be used to acquire the coordinates of any input hanger rod. The hard part is formatting that list to match the list of hangers, and keeping the right coordinates with the right hangers. Trapeze hangers with multiple rods make this especially tricky, when one hanger goes to multiple rods, you get lists of differing lengths. Below are more snips of a dynamo graph we use to extract points for robotic layout. The captions again explain what is being done to take points and assign them to hangers.
Figure 18: The first step takes in a list of elements and removes anything that's not a fabrication hanger. (the offshoot to the top of the snip is just removing riser clamp hangers as they have no rods) Then, the end locations for all the rods on all are retrieved. The list of hangers, and the list of rod locations are passed on to the next stage.

Figure 17: The next step takes the list of elements and extracts item number, service abbreviation and other identifying properties from the hanger part itself, and also gets all the ancillaries. The ancillaries here are the most important part. One ancillary of hangers is the support rod ancillary, that is the one we want. From it we can extract the rod diameters and the rod quantities.
Once the list of hangers and the list of points are equal in length, and the proper hangers are repeated in the hanger list for hangers with multiple rods, all that is left to do is format the properties for export. We use a CSV format because it is usable by most point layout machines regardless of brand. We export two additional fields with our CSV. The hanger service, and the hanger element ID. These let us troubleshoot problems in the field easier. For example, if the field comes back with an issue on hanger number 123, we can pull up our original CSV export and get the element ID for point 123 and zoom directly to the hanger in question and start troubleshooting with the model. This is just one example, as there are many fields that can be exported alongside hanger points.

The script is somewhat complex, so the full .dyn file is included as additional course materials for review. Be aware that this tool was built specifically to work with our (Mechanical Inc’s) database and families. There is a near 0% chance that applying it directly another database will allow it to work correctly. The file is only provided as an example and not as an actual tool for use.

Figure 19: The list cycler is the critical part of this tool. It is what cycles all the properties pulled from the hangers, like item number and service, and repeats them in the input list once for each rod of the hanger. This is what allows the list of hanger properties and the list of point coordinates to match in length, allowing for a meaningful list to be created.
Adaptive Point Family

There are other items in a project that may need points to be exported as well. These are typically non-fab part families, and as such can use the nested point family method. We use a small, generic model, sphere family called “POINT”. It is a shared family that gets nested into whatever family we need to pull coordinates out of. The example below is a vertical pipe sleeve family. These points have description parameters that correlate to the family they are a part of. For example the sleeve family has a POINT within it with the description that reflects the sleeve size and type using linked parameters within the family. Our point extraction tool then searches the selected elements for this POINT family. Using the out of the box dynamo node Element.GetLocation the coordinates for these points are extracted and then appended along with the description and mark values to the list of hanger points.

Issue 4 – Creating Accurate BOMs w/ Fabrication Reports

Problem
After extracting all the points and labeling hangers with everything we needed we ran into a few more minor snags before finishing our work with hangers. Our fabrication reports for hangers provided a field for the bottom of the pipe that the hanger was hosted to. This was due to the way our field personnel installed hangers. They would be installed at pipe elevation, and then the insulating sub-contractor would be responsible for lowering the hanger to its final height as insulation was applied. Unfortunately, the field in the reports used the Z-axis value from AutoCAD to generate this value, this value obviously does not exist in Revit, so a new solution was needed. Pulling the elevation of the hanger itself was not suitable, as it includes the insulation thickness when determining placement.

Solution
Some automated math was built into the renumbering tool mentioned earlier so that when hangers were populated and renumbered, the tool would write three values to custom data fields in the fabrication database that are used to list the proper elevation.

Careful Use of Custom Data
We determined we would need three custom data values on hangers to make our fabrication report work. Insulation thickness, bottom of pipe in the hanger, and the top of the hanger. Custom data is a convenient way to add data to parts that can be accessed in Revit quickly, but any good database manager would be wary of overusing this. Adding too many custom data fields freely could become very hard to organize and keep track of, as there is no way to categorize them or limit what ITM files they are
applied too. It is best to use this method sparingly, but it does have its uses when other options fail. In this instance there was no way we were ever going to be able to bring in the Z-coordinate axis from the CAD WCS into Revit, so we defaulted to the custom data method.

A Look Back at the Renumber Tool
Going back to the renumber tool there is another part of the script running tangent to the renumbering sequences. The flowing snip shows the section of the tool that is responsible for writing the data back to the fabrication parts. The blue inputs coming down from the top is the list of hangers being analyzed. The orange input from above is the list of pipes the hangers are hosted to. The host pipes are checked for insulation, and then based on the result the insulation thickness is added to the offset of the hanger to get the elevation of the physical pipe within. That value is then written to the custom data field that is reported in the fabrication report BOM. This data will also stay with the hangers if they are exported as an MAJ file.

Conclusion – A Custom Tailored Suit
Hopefully this has shown you that many of the limitations that may be keeping you away from using fabrication parts in Revit are not that bad, and they can be overcome with a little effort and some dynamo skills. There are also many paid add-ins that can help overcome these same issues or avoid using fabrication parts altogether while maintaining the same LOD. However, we like to take the custom-tailored suit approach. By developing all of our solutions ourselves, we have total control over how they operate and can make changes to them on the fly. Building custom tools to meet your needs can be incredibly powerful, especially if you have a heavily customized database that has had years of work put into it. You are not beholden to a developer that is trying to meet the needs of many diverse clients with one or two tools. By building it yourself you guarantee that it will work exactly the way you need it to. Like a suit, one of the rack might work fine, and in many cases that is till the right way to go. but sometimes, when the event is really important, a custom fitted solution can’t be beat, and we urge you to try it.