AS320859

Dynamo Automating Coordination with Annotations, Tags and Note Block Keynotes

Alanna Watts
Parkhill Smith and Cooper, Inc

Brandon Hartley
Parkhill Smith and Cooper, Inc

Learning Objectives

- Learn how to create Windows Task Schedules to run Dynamo Sandbox that opens in multiple projects and runs various Dynamo scripts
- Learn how to develop generic annotation comments and Dynamo script to enable faster coordination between the project manager and designers
- Learn how to integrate keynote note blocks in a floor plan and automatically hide keynotes on incorrect views
- Learn how to prepare a Dynamo script to assist in the tagging and noting of documents

Description

Learn how to use Dynamo to automate tedious tasks involving project coordination, note block keynotes, and plan tagging. This class will provide three different scripts, which will be shared with attendees. The first script will employ Dynamo Sandbox, the packages from Dynamo Automation, and Windows Task Scheduler. This script lets Revit users create generic annotation note comments that will then automatically export these views and comments to Microsoft Excel. This script is designed to cut back on Bluebeam sessions and red marks, letting a project manager, non-tech-savvy technical experts, and varying discipline engineers communicate with one another through Revit and Excel. The second script automatically hides generic annotation note block keynotes on sheets where certain keynotes do not belong, letting the keynote note blocks be placed directly in the view instead of on the sheet. The third script enables more automatic placement of tags and notes for mechanical ductwork plans.

Speaker(s)

Alanna Watts, PE is a Mechanical HVAC Design Engineer at Parkhill, Smith and Cooper, Inc. Her passions lie in streamlining tedious processes and saving time. This desire is likely driven by her prior experience in Supply Chain Management at Bell Helicopter. Alanna has been involved with BIM Development at PSC since first joining the team in 2013 and is Autodesk Revit Mechanical Certified. More recently, she spends a portion of her free time trolling twitter for like-minded Dynamo Enthusiast. She likely already follows you.

Brandon Hartley, RA, AIA is an Architect in the Healthcare Sector at Parkhill, Smith and Cooper, Inc. He is a BIM Coordinator and a Revit Architecture Certified Professional who constantly works to find new and better ways to collaborate with all disciplines within the A/E firm. Brandon is an AU veteran and has had the privilege of attending several conferences over the past 10 years. While Brandon generally prefers to reside behind the scenes, this year he hopes to bring AU attendees some of the most interesting nuggets he has learned over the past decade.

Color Guide
Dynamo Scripts
Packages
Nodes
Revit Category / Item / Option

AUTODESK. UNIVERSITY

Table of Contents

Part 1 - Dynamo Automation – Utilizing Windows Task Scheduler and Dynamo Sandbox

Task 1 – Master Script in Dynamo Sandbox

Create a "Master" Script in **Dynamo Sandbox** to open a **Revit Local Copy** and **Run** a Dynamo Script(s). This "Master" Script will allow further automation through **Dynamo Sandbox**. The user will later be able to use **Windows Task Scheduler** to **Run** various **Dynamo** scripts at user defined intervals.

> *Note: Andreas Dieckmann created and detailed this Master Sandbox Script on GitHub. For further information on this Dynamo Sandbox Master Script, refer to the references at the end of this document.*

Process 1

In addition to the ***MasterScript.dyn*** **Dynamo Sandbox** script, it is also recommended that you move any local copies of the **Revit Central Model** to be opened to a separate folder. The local copy needs to avoid having a user name that involves an @ symbol, as this symbol will cause the ***MasterScript.dyn*** to fail. This separate folder will be referenced in the ***MasterScript.dyn***.

Utilize Revit 2020 and **Dynamo Sandbox** and install the following **packages** which will ensure the script will **Run** successfully:

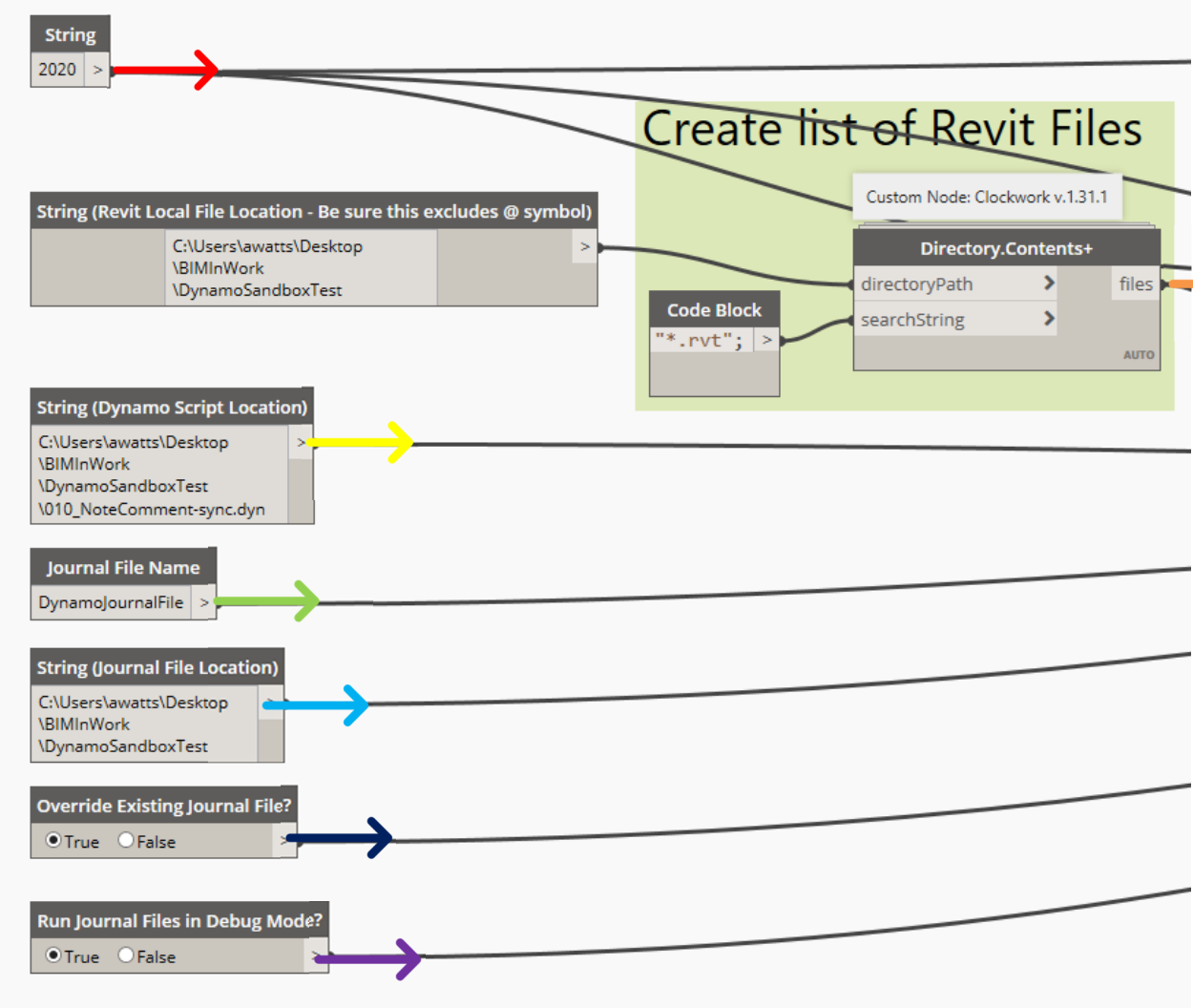
1. **Clockwork for Dynamo 1.x** 1.33.1
2. **Clockwork for Dynamo 1.x** 2.1.2
3. **DynamoAutomation** 1.31.1

Upon a successful **Run** of the **MasterScript.dyn** script, the script will open the **Revit Local Files** placed in the folder and execute the specified Dynamo Revit native .dyn "Slave" file.

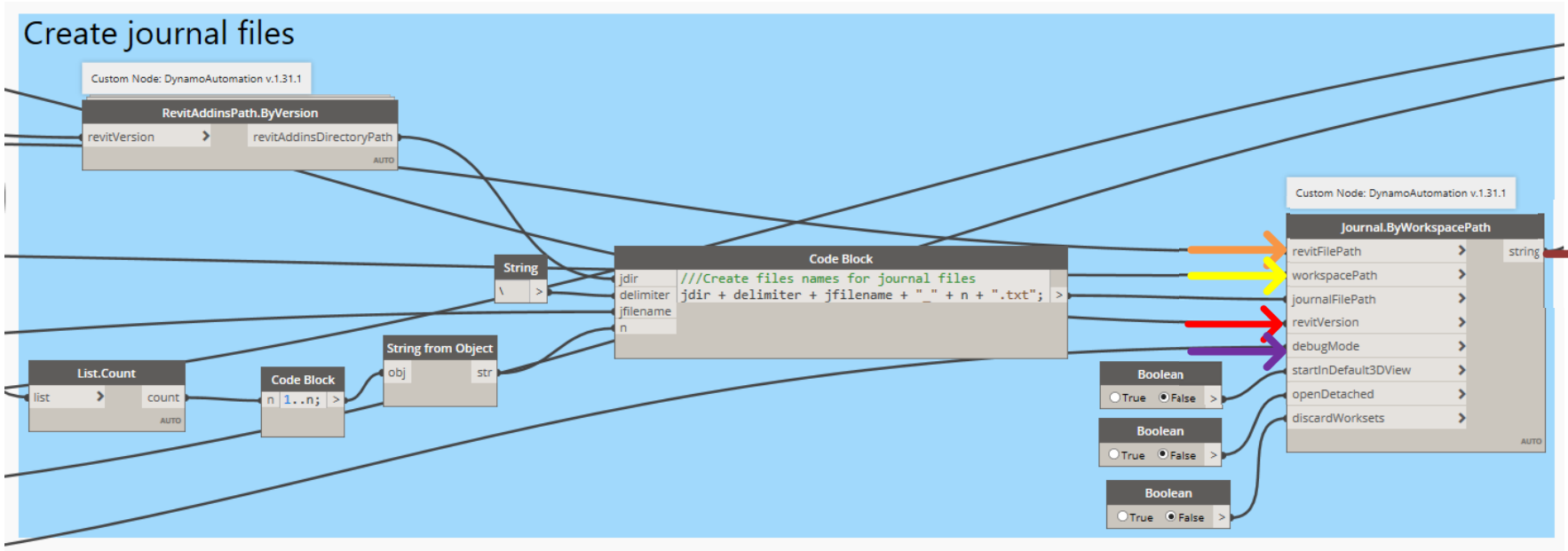
Dynamo Sandbox (In-Depth)

Master Script – Open Revit Local Copies and Run Dynamo Files

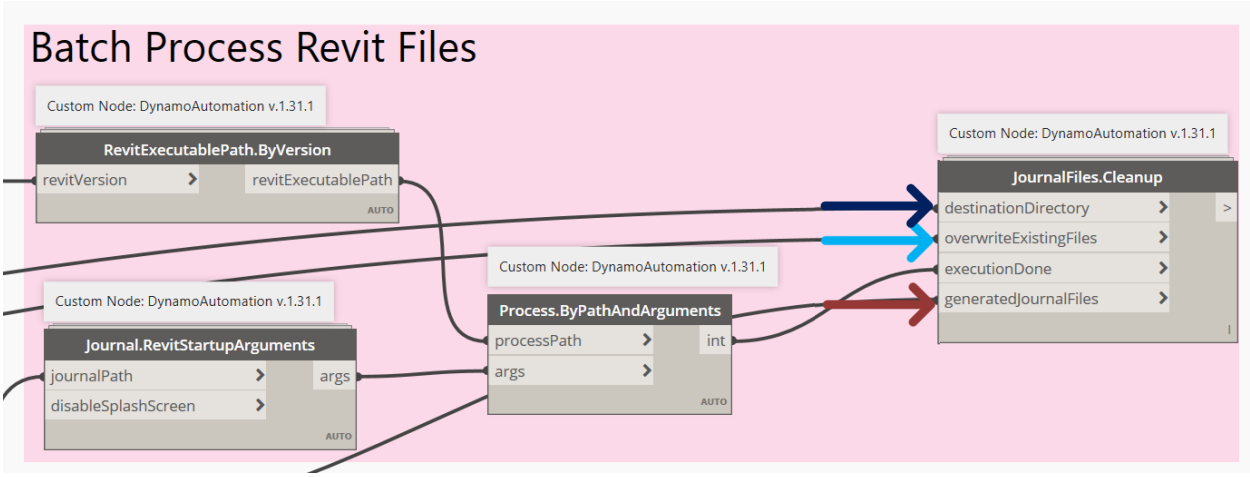
Begin by selecting the Revit version year, the folder location of the **Local Revit** file copies to be opened, the slave script Revit Dynamo file location, the **Journal File** name, the **Journal File** location, whether to override the journal file, and if the **Journal File** should be run in Debug mode. Then use **Clockwork** package node **Directory.Contents+** to obtain a list of all .rvt files in the folder location.



Then use **DynamoAutomation** package node **RevitAddinsPath.ByVersion** to obtain the file location for the Revit Addins for the selected Revit version. Combine this Addin Version path with additional parameters to obtain the .txt file for the Journal File Path. Then use **DynamoAutomation** package node **Journal.ByWorkspacePath** to obtain the **Journal File** Workspace Path.



Use the **DynamoAutomation** package node **RevitExecutablePath.ByVersion** to obtain the .exe executable Revit file location for the specified version number. Use the **DynamoAutomation** package node **Journal.RevitStartupArguments** to obtain the language and the **Journal File** .txt Path.



Based on the user defined inputs the **MasterScript.dyn** will open a **Revit Local Copy** file and **Run** the "Slave" **Dynamo** script.

> *Note: If you would like for this "Slave" .dyn file to synchronize the changes that it made back into Revit, this needs to be accomplished in the "Slave" script itself.*

Task 2 – Setting Up Windows Task Scheduler

Create a **Task Schedule** to execute the **MasterScript.dyn** in **Dynamo Sandbox** at various times and automatically run these without additional user input.

This will be accomplished by using the **MasterScript.dyn** to open a **Revit Local Copy** and **Run** a Dynamo Script(s). A .vbs file will need to be created and edited to call the specific file location for the **MasterScript.dyn**. Then a Windows Task Schedule task will be created to execute the .vbs file at the desired intervals.

> *Note: Windows Task Scheduler will open the DynamoSandbox.exe. Currently there is no way to request that it open the specific **MasterScript.dyn** file. While DynamoCLI.exe will open a specific file, it will not actually execute the script effectively as it does not see all of the library file paths native to Dynamo Sandbox. It was determined that a .vbs file would allow the user to open Dynamo Sandbox and Run the specific **MasterScript.dyn** file. Ensure that these tasks are scheduled for after hours so that you do not interfere with your typical workflow.*

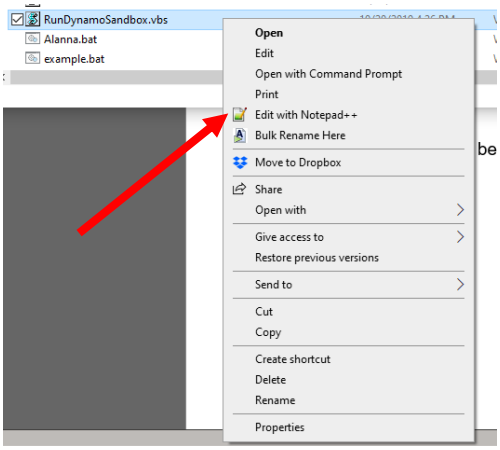
Process 2

In addition to the **MasterScript.dyn** Dynamo script, it is recommended that you download and transfer the following item and place it in a folder location:

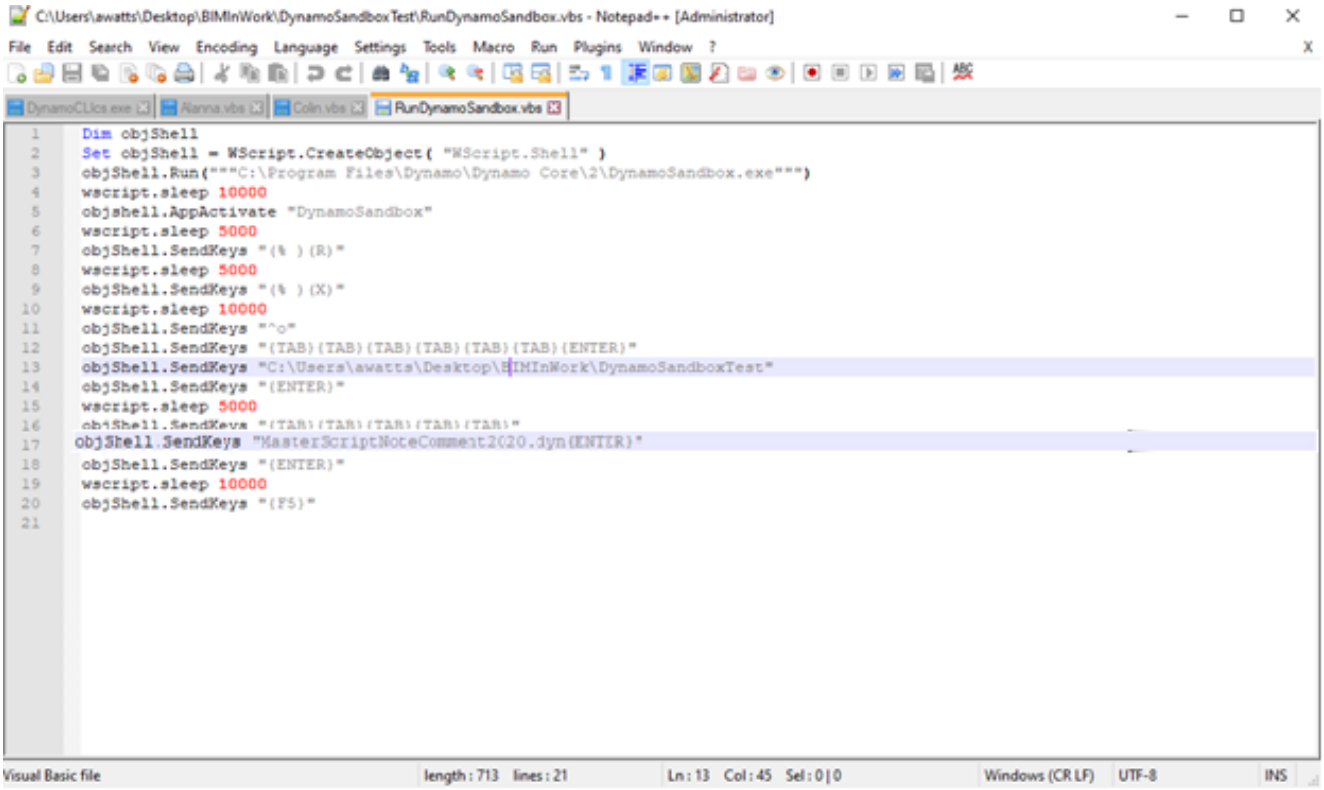
1. ***RunDynamoSandbox.vbs***

.vbs Detail

Once the .vbs file has been downloaded. Right click on the file and select Edit with Notepad++ or some other text editor.



Be sure to edit the file location of the “Master” Script and the “Master” Script name itself.

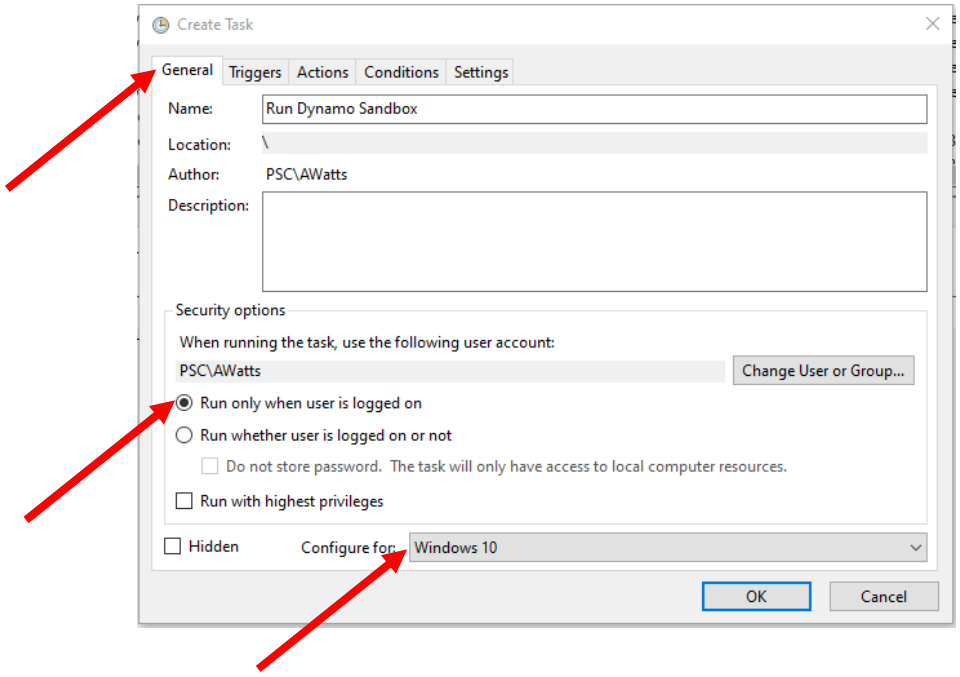


Below is the example of the text in the provided ***RunDynamoSandbox.vbs*** file.

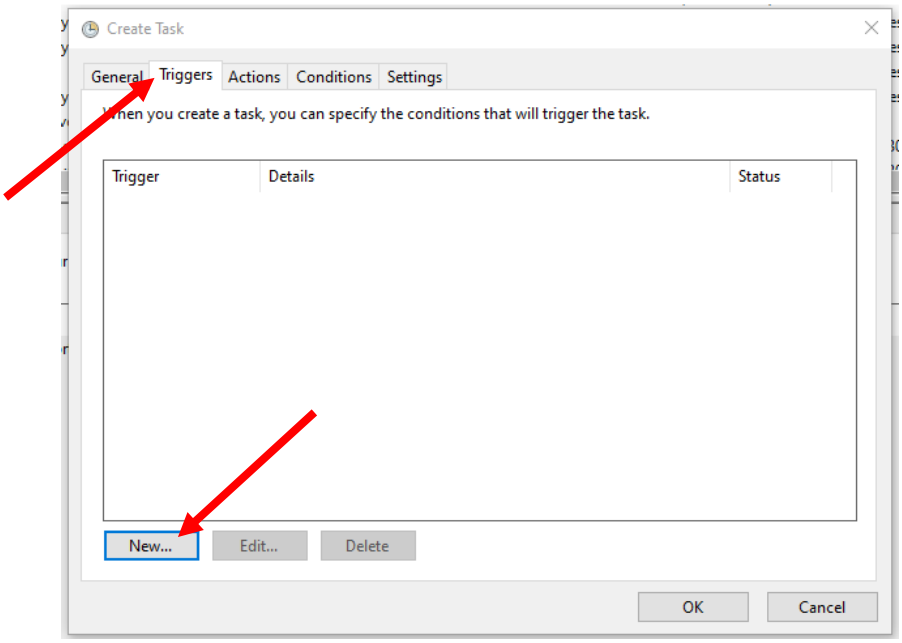
```
Dim objShell
Set objShell = WScript.CreateObject( "WScript.Shell" )
objShell.Run("""C:\Program Files\Dynamo\Dynamo Core\2\DynamoSandbox.exe""")
wscript.sleep 10000
objshell.AppActivate "DynamoSandbox"
wscript.sleep 5000
objShell.SendKeys "(% )(R)"
wscript.sleep 5000
objShell.SendKeys "(% )(X)"
wscript.sleep 10000
objShell.SendKeys "^o"
objShell.SendKeys "{TAB}{TAB}{TAB}{TAB}{TAB}{TAB}{ENTER}"
objShell.SendKeys "C:\Users\awatts\Desktop\BIMInWork\DynamoSandboxTest"
objShell.SendKeys "{ENTER}"
wscript.sleep 5000
objShell.SendKeys "{TAB}{TAB}{TAB}{TAB}{TAB}"
objShell.SendKeys "MasterScriptNoteComment2020.dyn{ENTER}"
objShell.SendKeys "{ENTER}"
wscript.sleep 10000
objShell.SendKeys "{F5}"
```

Windows Task Scheduler Detail

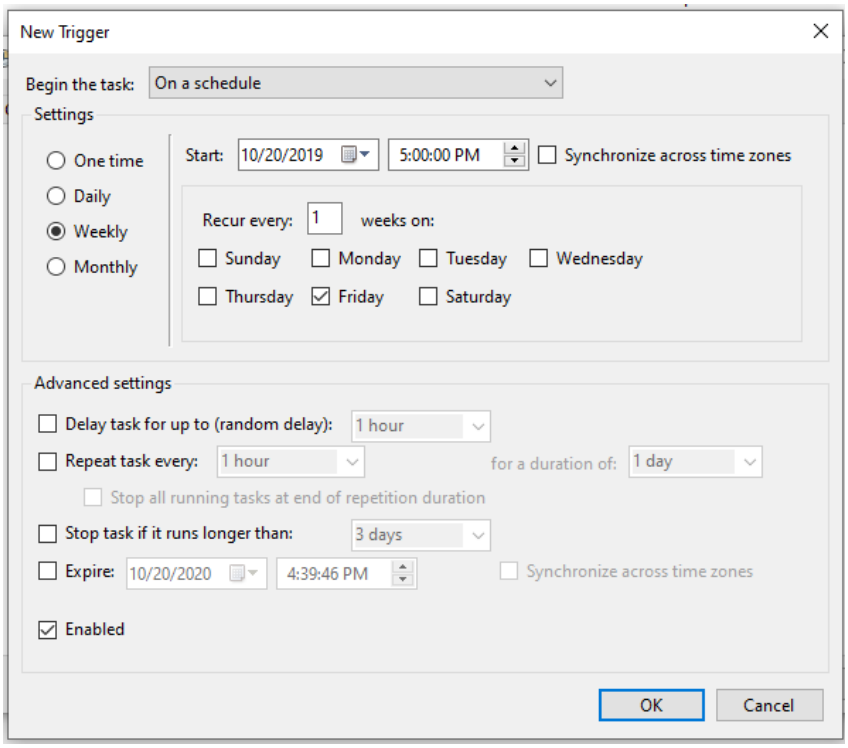
Open **Task Scheduler** and click **Create Task**, name the task in the **General Tab** and change the **Configure for:** “Windows 10” in this case and the **Security options** to “Run whether user is logged on or not”.



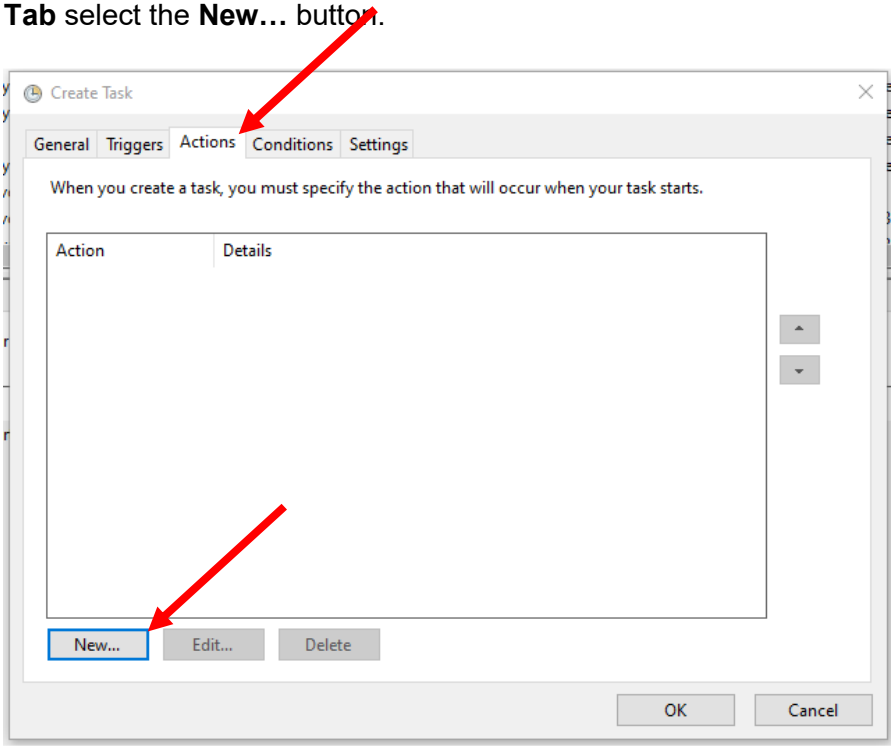
In the **Triggers Tab** select the **New…** button.



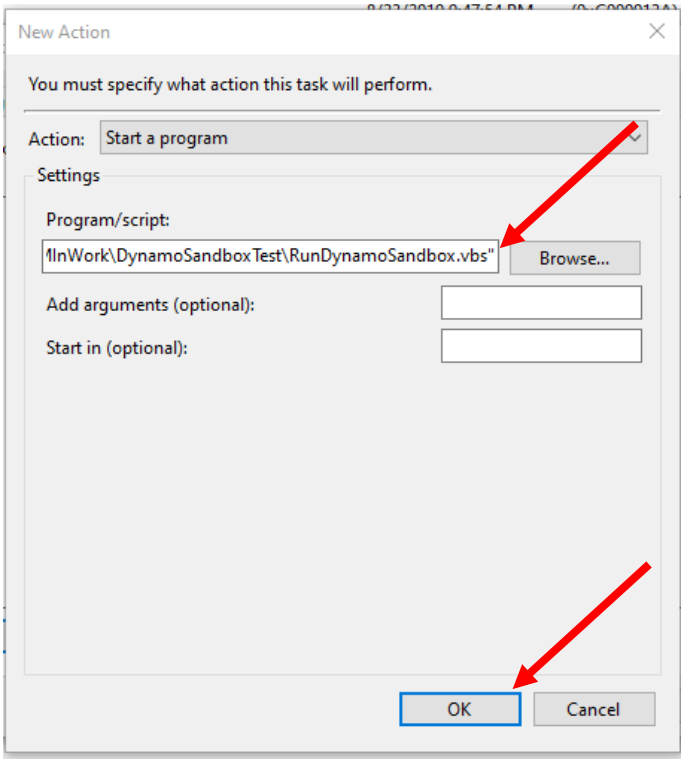
Input the Trigger time and/or schedule that you wish to perform the task.



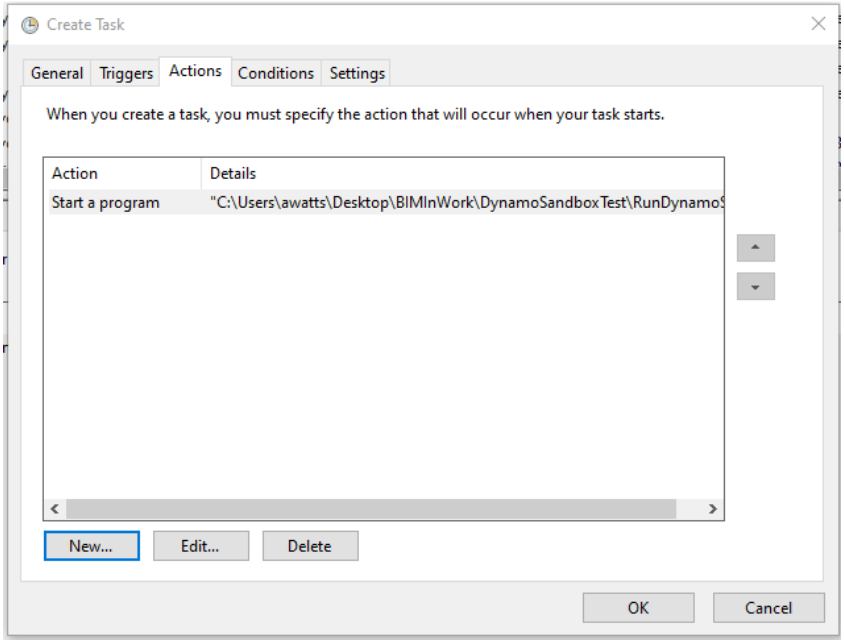
In the **Action Tab** select the **New…** button.



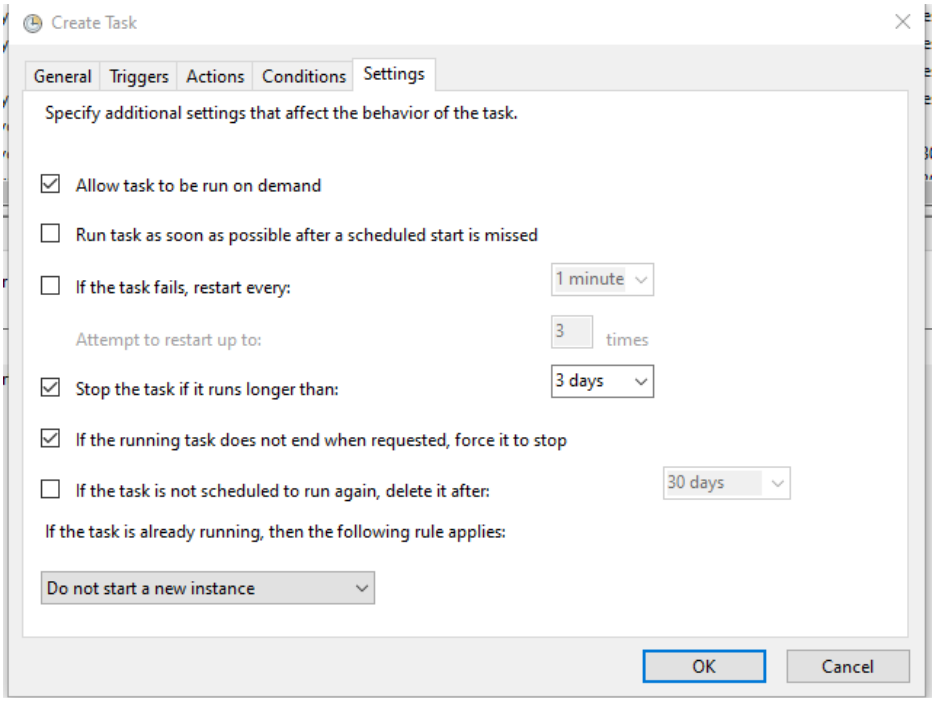
Input, in quotations, the file path location for the .vbs file location and select **OK**.



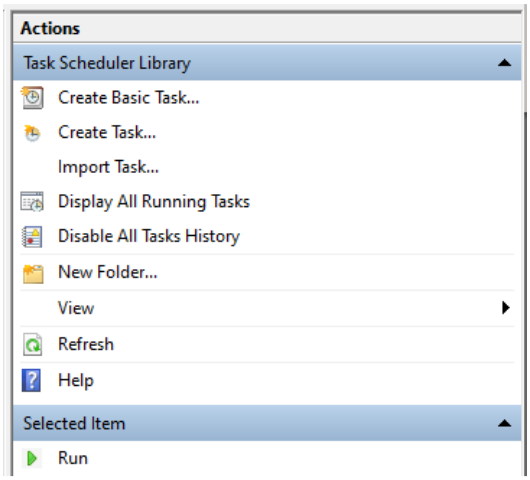
The **Action Tab** will now reflect program you wish to start and file location that was selected.



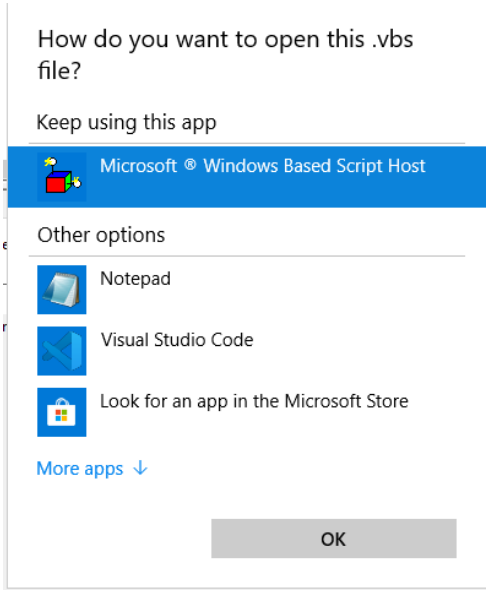
Edit any **Conditions** or **Settings** that you wish to change. Then select **OK** and the **Task** will now be created.



Click **Run** to verify that the **Task Schedule** will function.



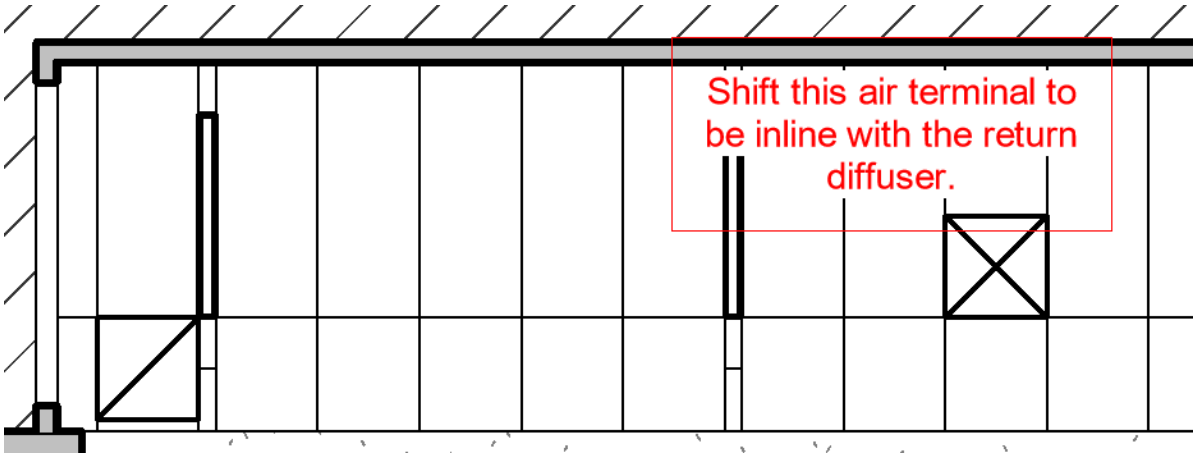
You maybe be prompted to specify which program will be used to run the .vbs files. If so, select the **Microsoft @ Windows Based Script Host** and select **OK**.



Task 3 – Note Comment Family and Schedule

Bluebeam Review Sessions are often used to assist in the coordination between disciplines and project managers. Often, these items can be identified well before a Bluebeam Session is created and put in to use. The **Note Comment** family and **Schedule** is intended to reduce the amount of coordination and assist in a more effective and seamless communication between linked models and project managers that may never open the Revit model themselves.

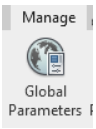
A simple **NoteComment.rfa** family was generated to help track questions for other disciplines and/or Project Managers. This **Note Comment** will allow the Revit users to add text comments to views within Revit for future coordination.



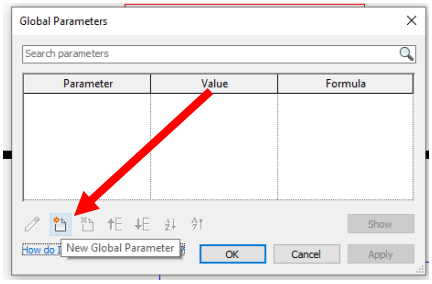
These comments are attached to a **Global Parameter** that will allow them to be hidden on all **Views** for interim deadlines.

> *Note: Global parameters are a great way to control the visibility of various items within a Revit project. Controlling the visibility of Editors Notes is also a common practice that is easy to implement into your project template.*

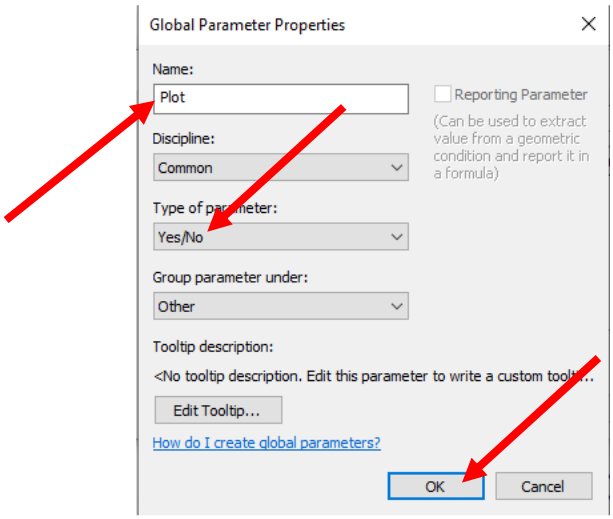
A **Global Parameter** is created the same way that family parameters are created. Select the **Manage** tab, then select **Global Parameters**.



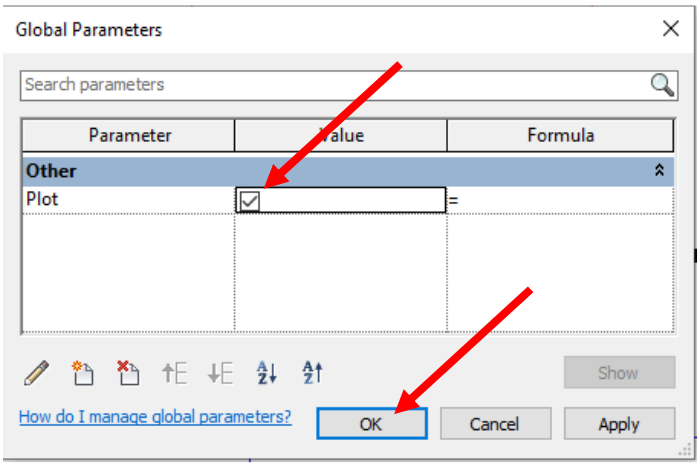
Click the **New Global Parameter** button.



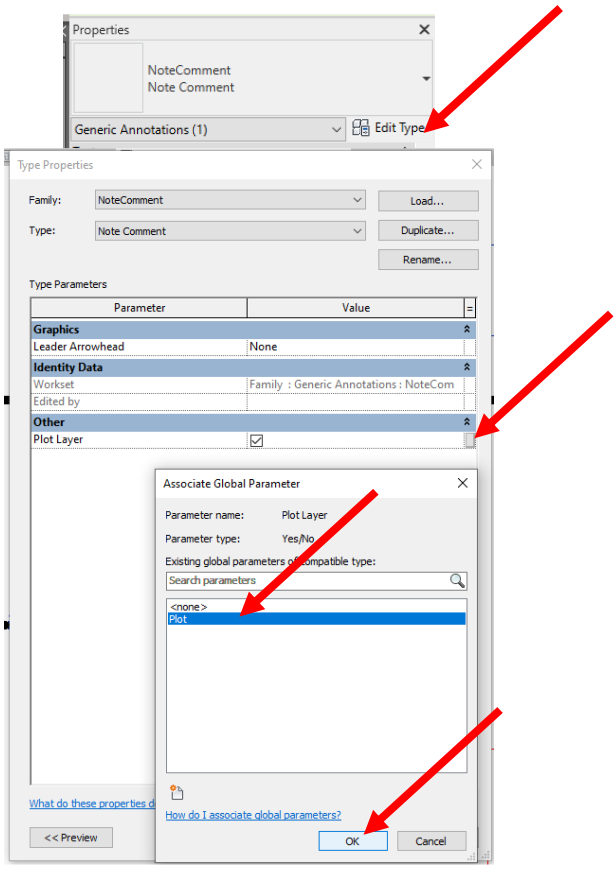
Define the name to be **Plot**, and parameter type to be **Yes/No** then select **OK**.



The **Global Parameter** has then been created ensure the check mark is checked and click **OK**.



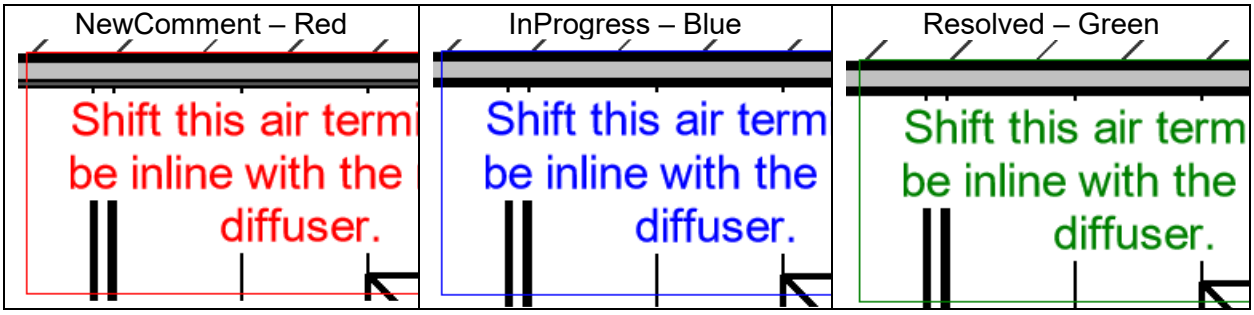
Then tie the **Note Comment** family type to the created **Global Parameter** by selecting a instance of the **Note Comment** family then in the properties panel selecting **Edit Type**. Click the blank box on the right side of the **Plot Layer** parameter and change this to be associated with the newly created Plot **Global Parameter** and select **OK**.



The provided **Note Comment** family contains the following parameters which will be used throughout this process.

Text	
CommentBy	Alanna
DisciplineToNotify	Mechanical
Note Text	Shift this air terminal to be i...
ResponseExcel	
ResponseRevit	
Identity Data	
Visibility	
NewComment	☐
InProgress	☐
Resolved	☑
Other	
DynamoDetailNumber	
DynamoSheetNumber	
ElementID	0.000000

So that the review process can be made clearer, a **NewComment**, **InProgress**, and **Resolved** parameter can be used to change the color of the **Note Comment** family.



The user will need to fill out the **Note Text**, **CommentBy**, and **DisciplineToNotify** parameters. The **Note Comments** are available for review in a Revit Schedule "**NOTE COMMENTS"**. These comments are automatically added to the schedule. But the Dynamo Script **SlaveNoteCommentImportExport.dyn** will add the **DynamoSheetNumber**, **DynamoDetailNumber**, and **ElementID** parameters to the **Note Comment** automatically after a successful run. These modified parameters will drive the sorting and grouping of the "**NOTE COMMENTS"** schedule.

<NOTE COMMENTS>

A	B	C	D	E	F	G	H	I	J
	In Progress	Resolved	Issue	Detail No.	Notify	Comment By	Revit Response	Excel Response	ElementID
A-112A									
☑	☐	☐	A-112A	A1					7089072
A-116A									
☑	☐	☐	A-116A	A1					7089078

These comments will then be exported to **Excel** by **Dynamo**. This file will allow different disciplines to answer and/or reply to comments, related to their discipline, and then have this information reimported back into **Revit**.

Process 3

In addition to the **SlaveNoteCommentImportExport.dyn** Dynamo script, it is recommended that you download and transfer the following item and place the family and schedule it in the project and the excel file in a folder location:

1. ***NoteComment.rfa***
2. **NOTE COMMENTS.rvt**
3. ***NoteCommentOrginal.xlsx***

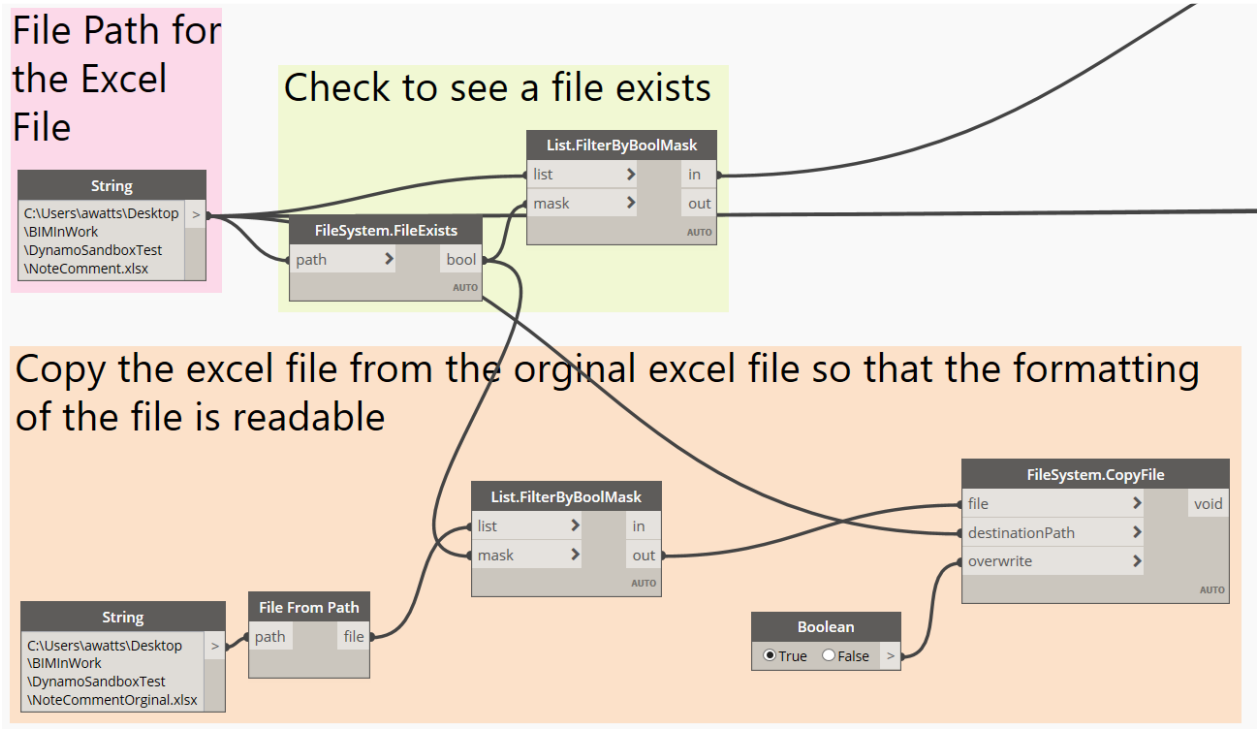
Utilize Revit 2020 and Dynamo and install the following **packages** which will ensure the script will **Run** successfully:

1.	**archi-lab.net**	2020.22.5
2.	**bimorphNodes**	3.0.3
3.	**Clockwork for Dynamo 1.x**	1.31.1
4.	**Clockwork for Dynamo 2.x**	2.1.2
5.	**DynamoAutomation**	1.31.1
6.	**MEPover**	2019.9.3
7.	**spring nodes**	203.1.0

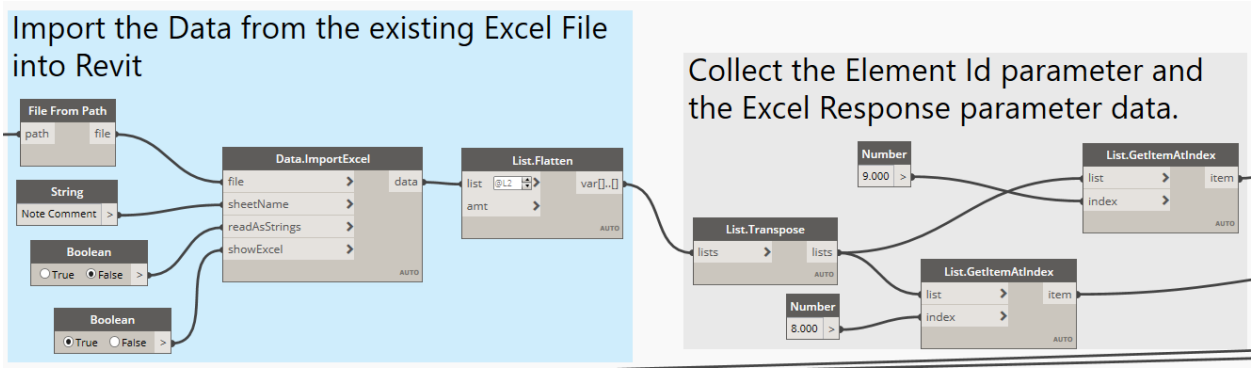
Upon a successful **Run** of the **SlaveNoteCommentImportExport.dyn** script, the script will import the **ResponseExcel** parameter information back into Revit and then export the Revit "**NOTE COMMENTS"** schedule back to the Excel file location. The script will then **Synchronize** the model and then close **Excel** and close Revit.

Slave Script – Note Comment Excel Import and Export to a Revit Schedule

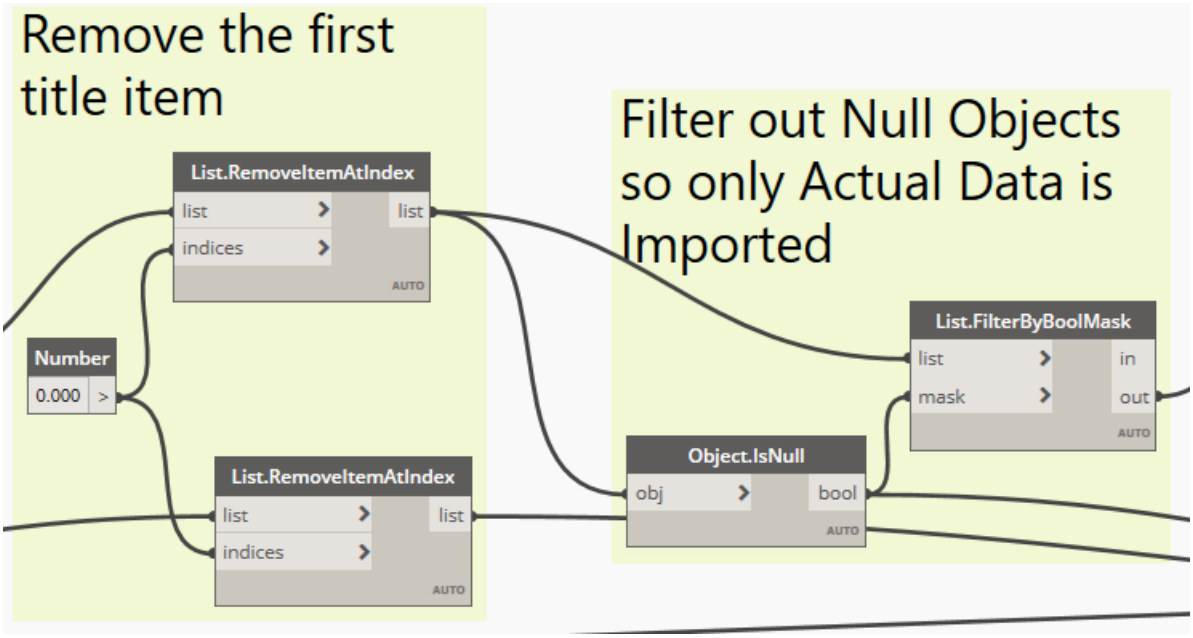
Begin by defining the file path location of the **Excel** export Note Comment file, ***NoteComment.xlsx***, then check to see if this file already exists. If it was previously created, later nodes will extract the **ResponseExcel** parameter information. If it was not previously created, the file will be copied from a standard location so that the formatting of the file is consistent.



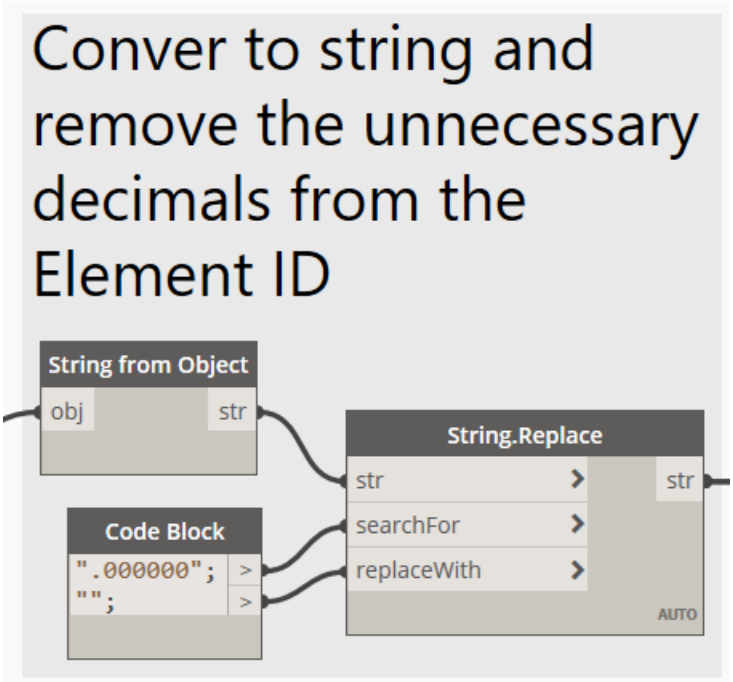
Import the **Excel** data into **Dynamo**, and then transpose the list and use the **List.GetItemAtIndex** node to pull the **ElementID** and **ResponseExcel** data.



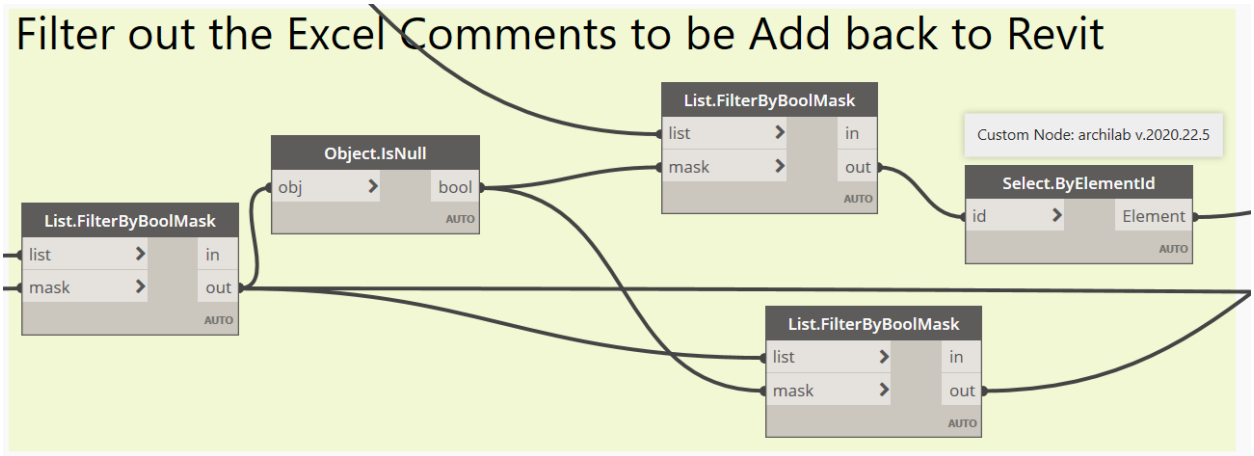
Remove the first item of the index to remove the Title item. Then filter out any null objects so only actual data is imported back into Revit.



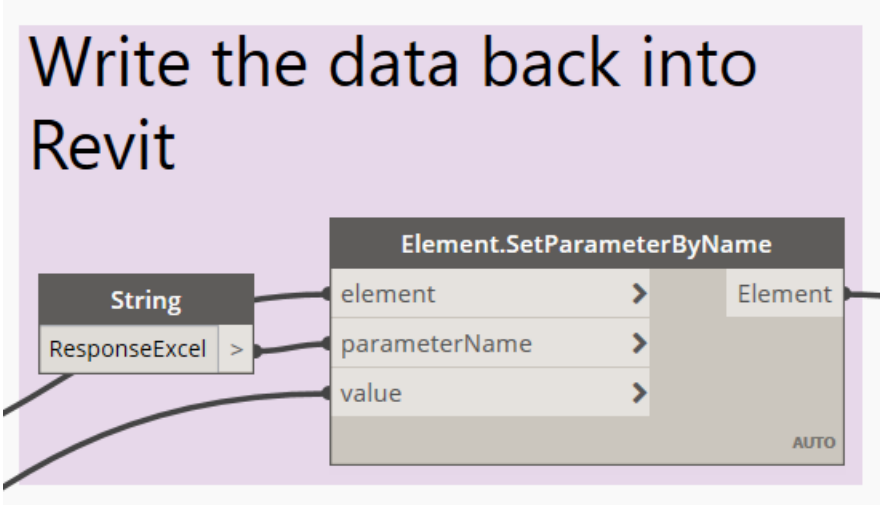
Convert the **ElementID** parameter to a string and remove the additional unnecessary zeros after the decimal points.



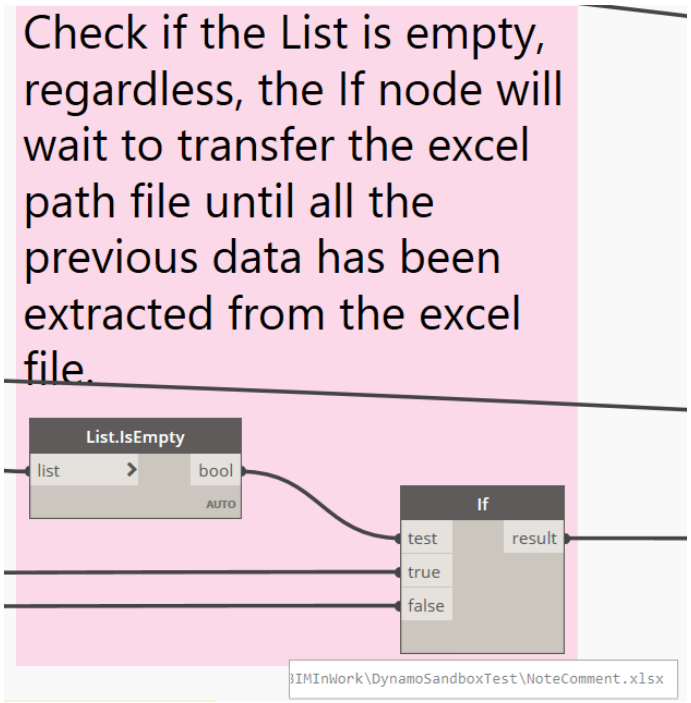
Filter out the **Excel Comments** to be added back to Revit. Then use the **archilab** package node **Select.ByElementId** to select the **NoteComment** instance within Revit.



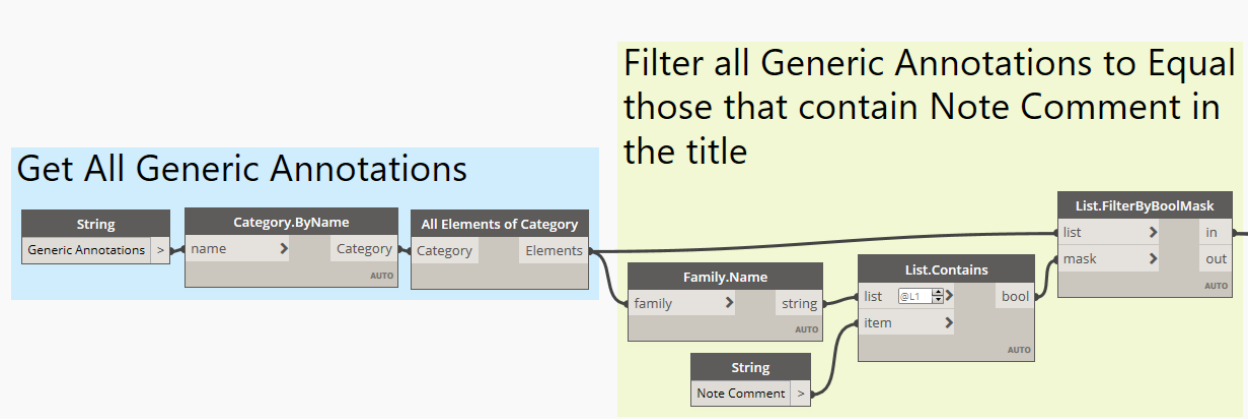
Write that data back into Revit.



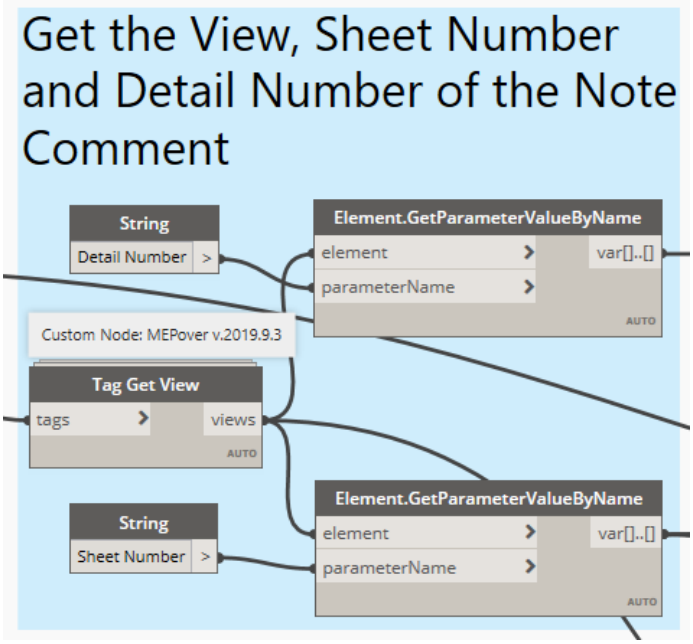
Check if the **Excel** data has been imported into Revit. Regardless, the **If** node will cause the script to wait for this process to complete prior to pulling the **Excel File Path** through. Both **True** and **False** parameters contain the same **Excel File Path** location.



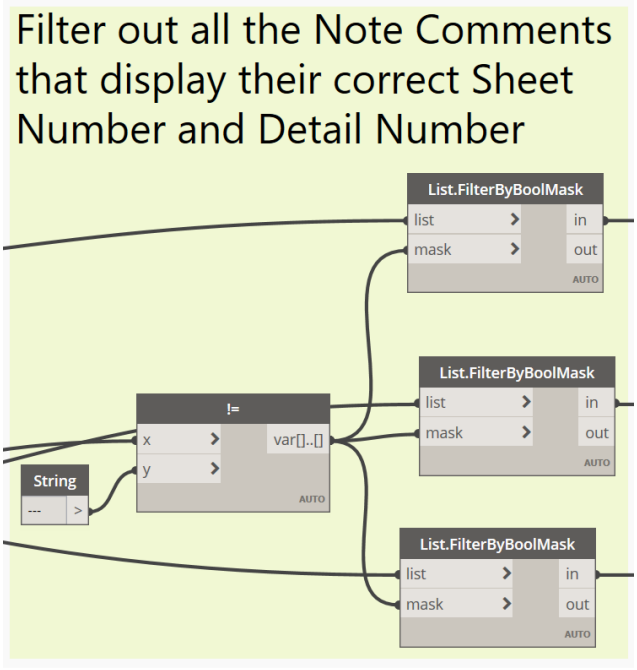
For the second part of the script, select all the **Note Comments**.



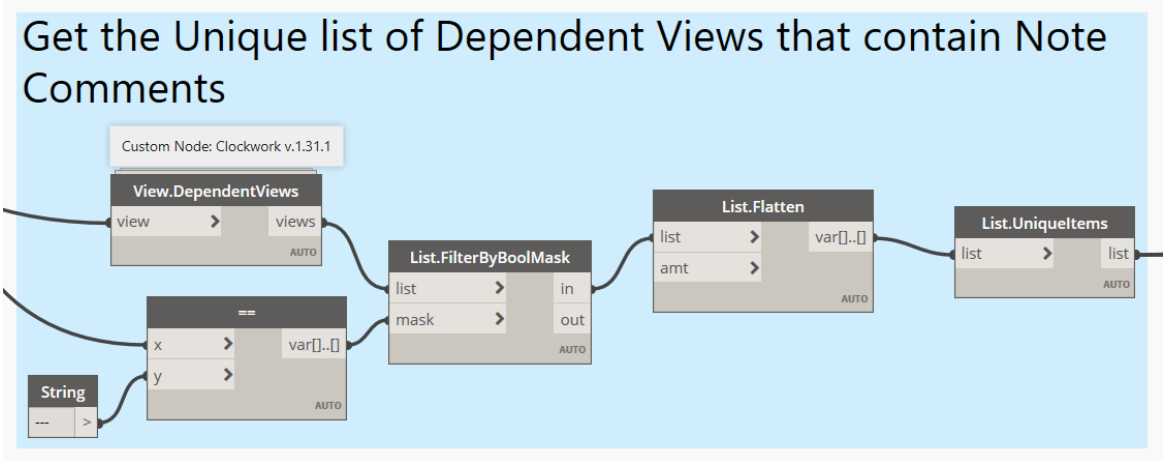
Use the **MEPover** package node **Tag Get View** to select all the **Views** where the **Note Comments** are located. Then get the **Sheet Number** and **Detail Number** of the **View** on which these **Note Comments** are placed..



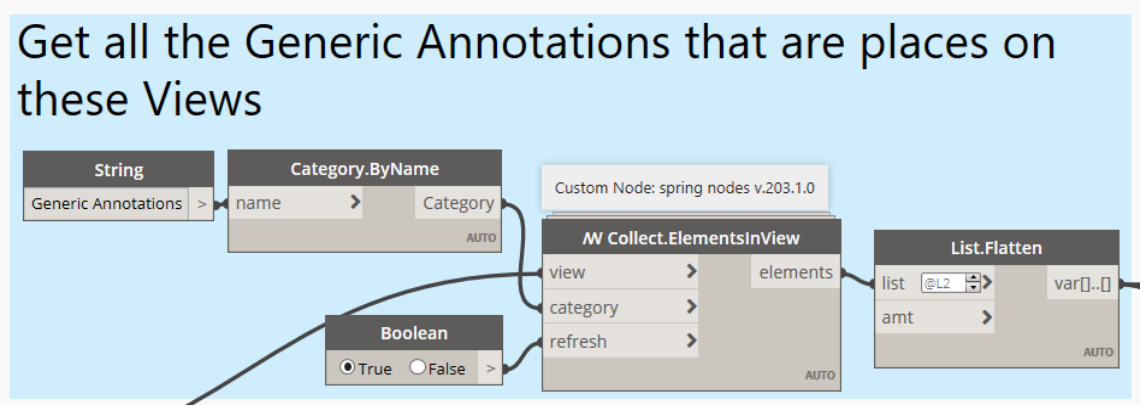
If the **Note Comments** are placed on a Floor Plan Views that contains other dependencies, these **Note Comment** locations will not appear. Filter out the **Note Comments** that display the correct **Sheet Number** and **Detail Number**.



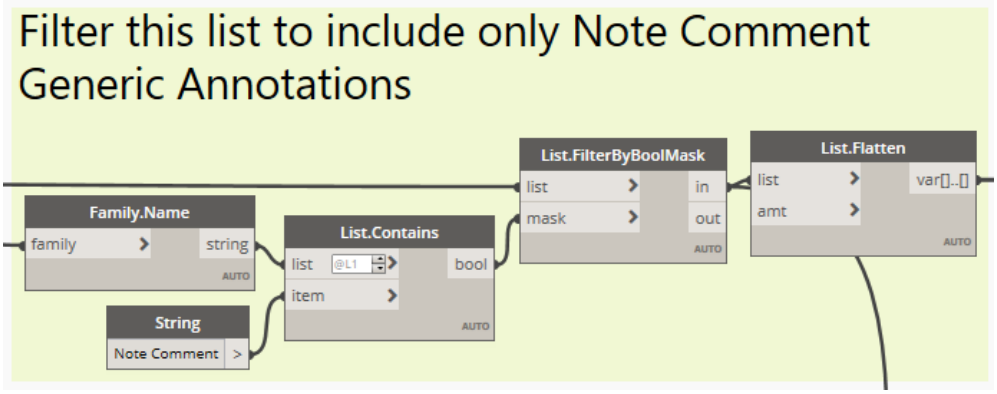
Use the **Clockwork** package node **View.DependentViews** to select all the dependent views associated with a specific view. Then get the unique items of these views.



Get a list of all the **Generic Annotations** that are located on these unique dependent views by using the **springs node** package node **MW Collect.ElementsInView**.

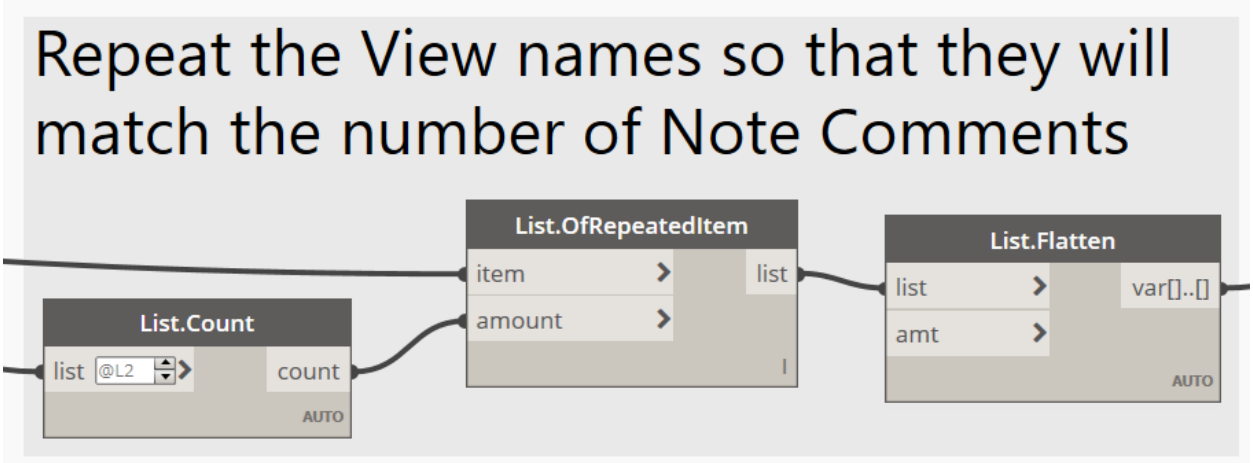


Then filter this list to include only the **Note Comments**.

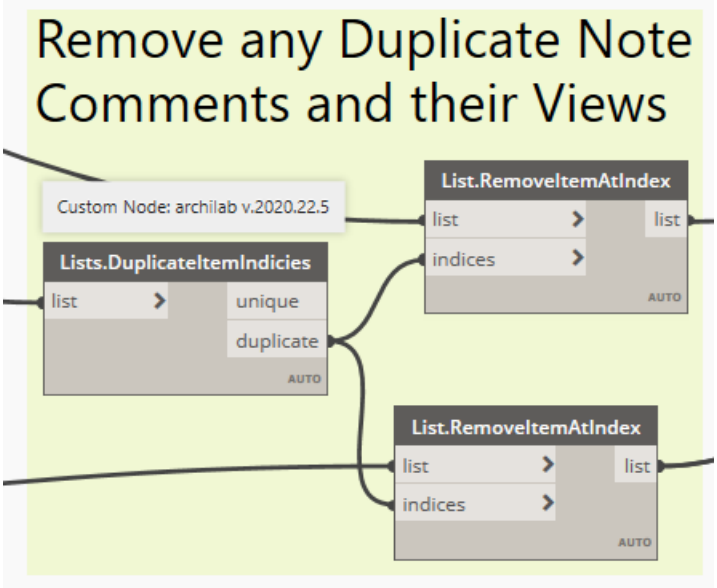


Note: Though the Note Comments were all previously selected, only selected dependent views were listed. By handling the filters twice, the computing power of the script will be reduced and the script will run faster.

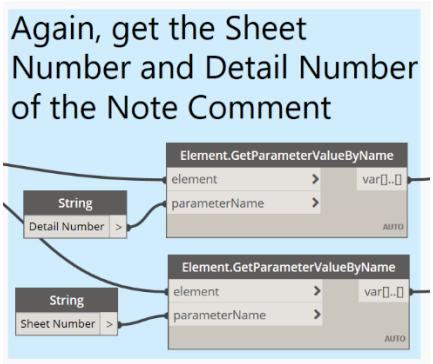
Get the **View** names repeated so that the **View** names will match the number of **Note Comments** that exist.



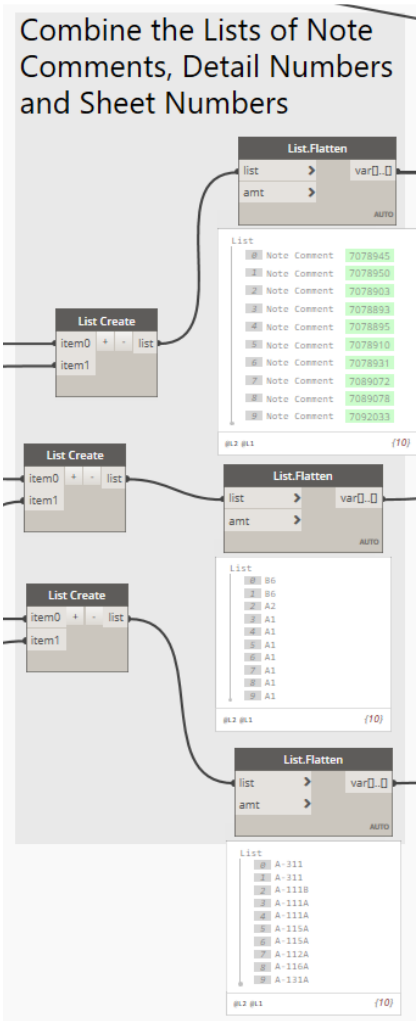
Remove any duplicate **Note Comments** and their **Views**.



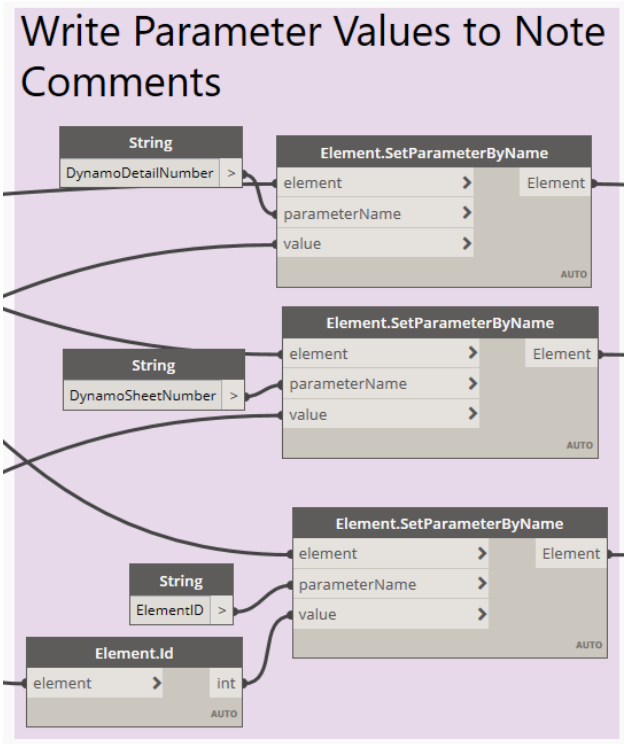
Get the **Sheet Number** and **Detail Number** of these **Note Comments** that are on **Dependent Floor Plan Views**.



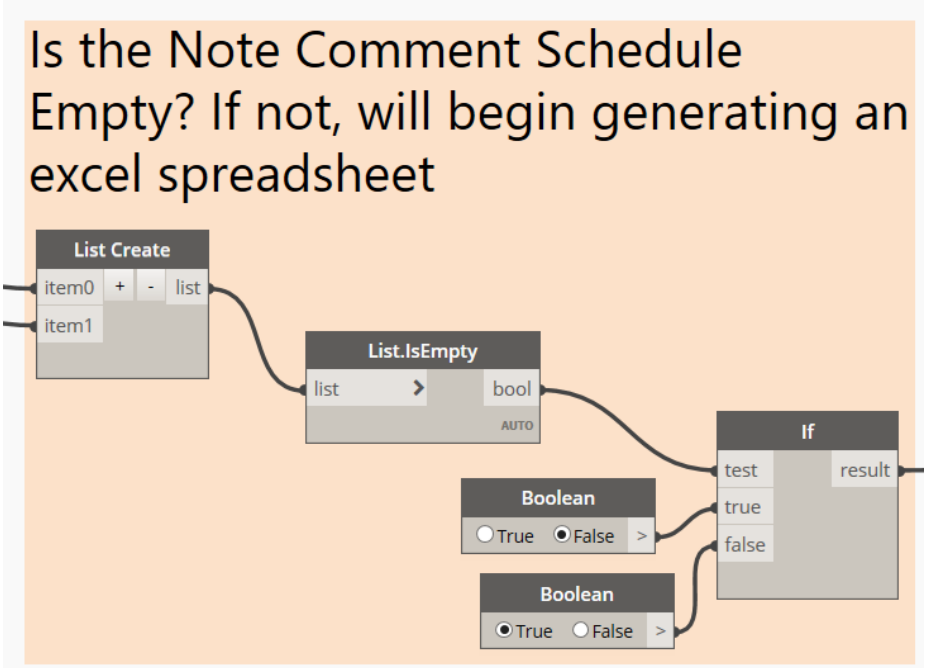
Add these **Views**, **Dependent Floor Plan Views, Note Comments**, **Sheet Number**, and **Detail Number** lists together.



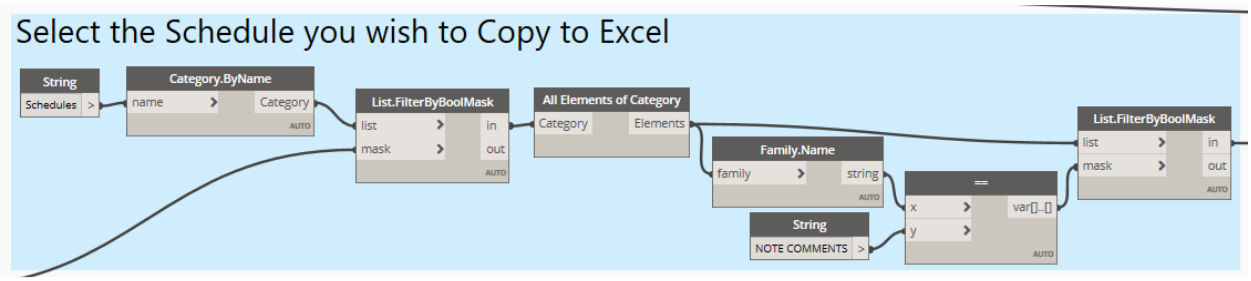
Write the **DynamoDetailNumber**, **DynamoSheetNumber**, and **ElementID** parameters to the **Note Comments**.



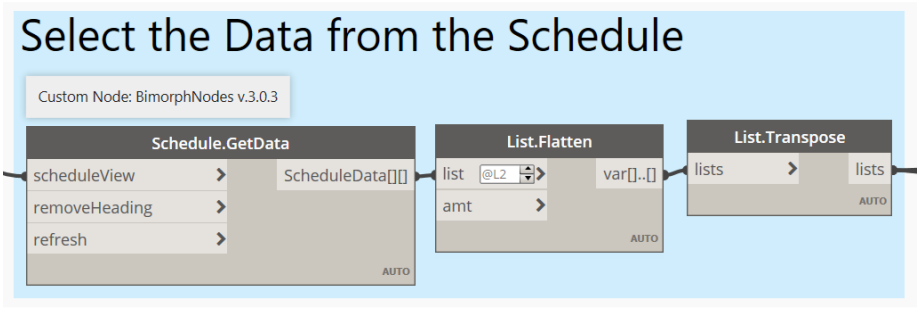
Check to see if there are no **Note Comments**.



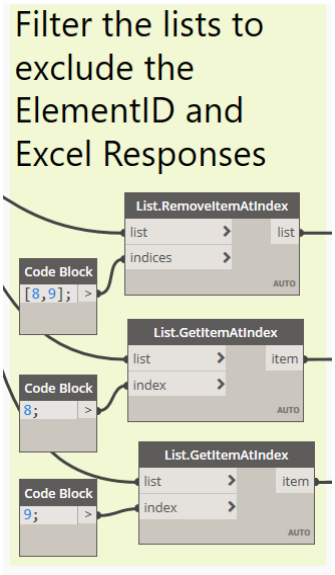
If there are no **Note Comments**, then the process will end here. If there are **Note Comments**, then these nodes will pull the "**NOTE COMMENTS"** schedule.



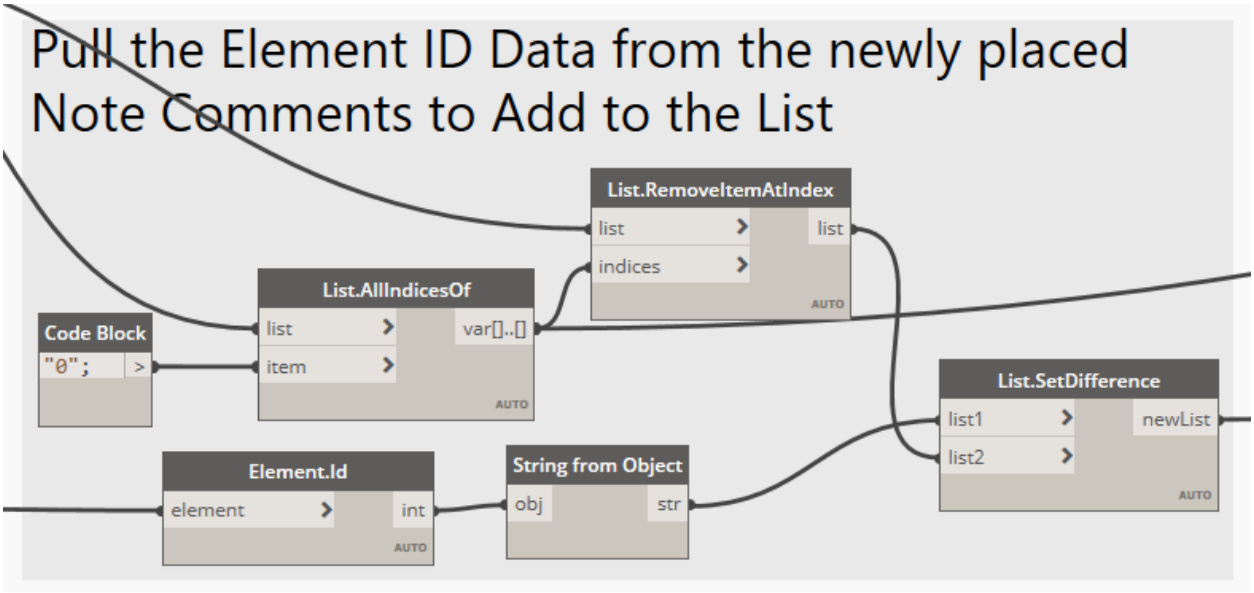
Use the **BimorphNodes** package node **Schedule.GetData** to pull the data value from the Schedule in the correct order and with the same filters.



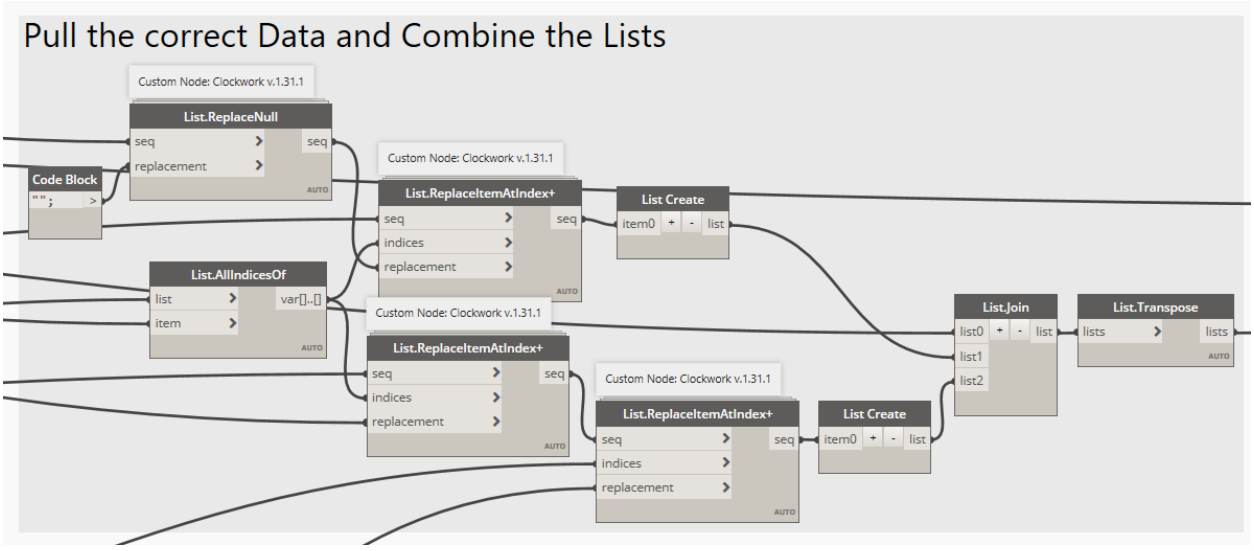
Filter this Schedule Data to exclude the **ElementID** and **ResponseExcel** parameters and pull the lists of these parameters.



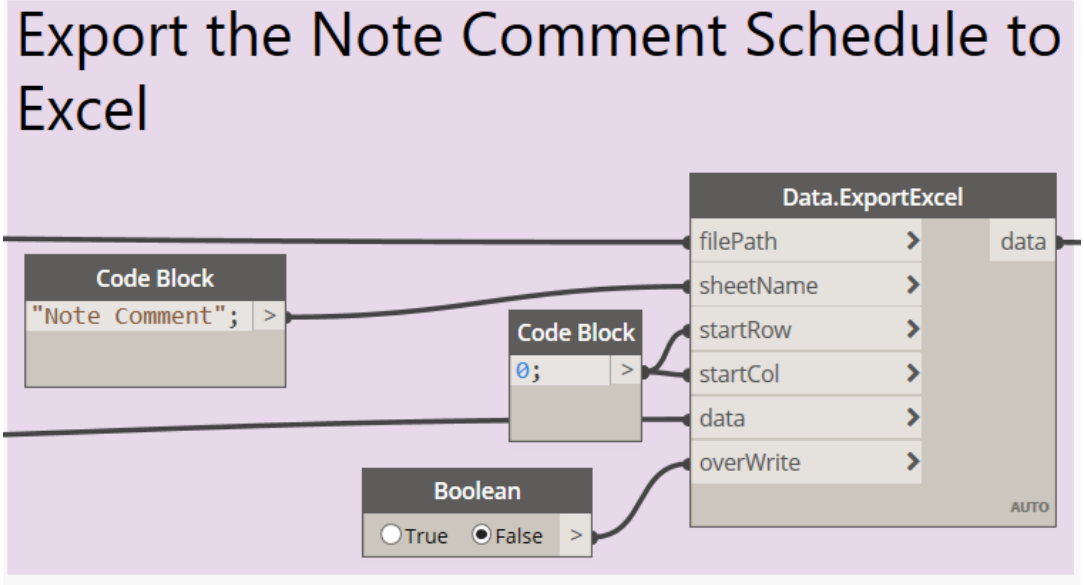
Pull the **Element ID** data from the newly placed **Note Comments** and add these new element ID's to the existing list.



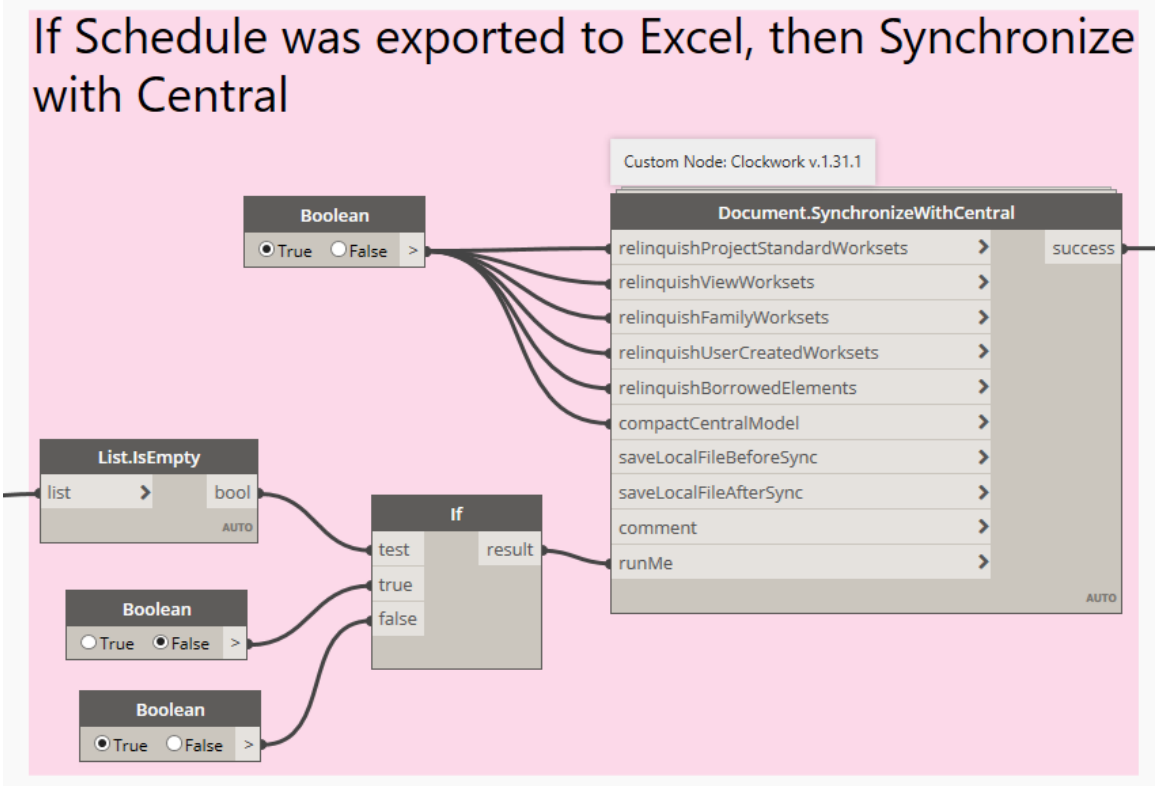
Use the **Clockwork** package nodes **List.ReplaceNull** and **List.ReplaceItemAtIndex+** to pull, compile and combine the lists so that the **ElementID** and **ResponseExcel** contain the most up to date information.



Now export this updated **Schedule** back to **Excel**.



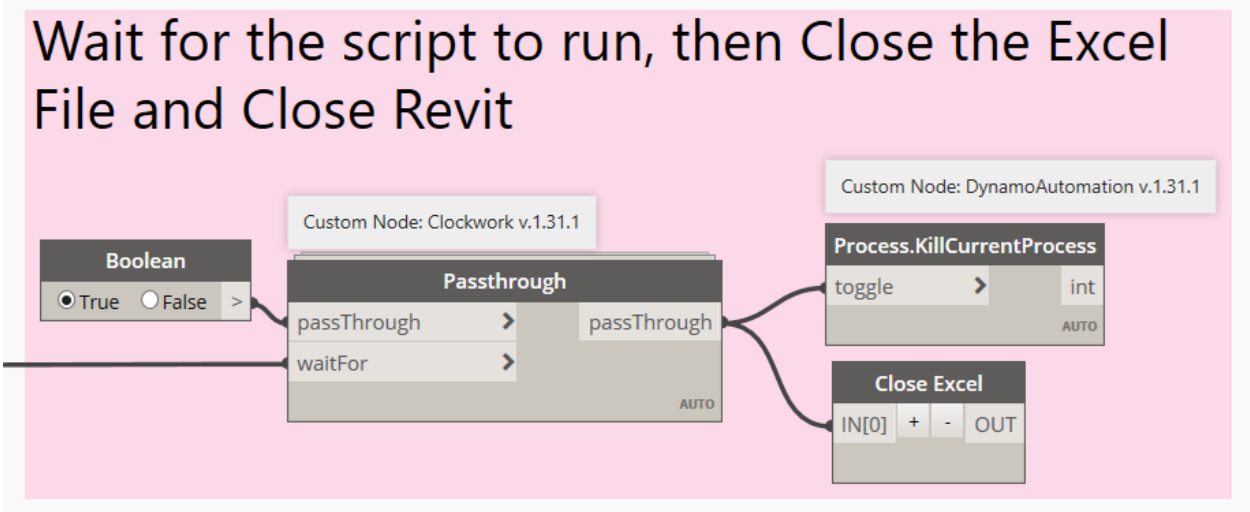
Use the **Clockwork** package node **Document.SynchronizeWithCentral** to synchronize with the central model if there was an updated exported "**NOTE COMMENTS**" schedule to **Excel**.



Use the Clockwork package node Passthrough to wait for the synchronization to be completed. Use the Python code written below to close the Excel file that was just updated.

> *Note: This Python code is curtesy of Raja refer to the references for further information.*

Then use DynamoAutomation package node Process.KillCurrentProcess to close the Revit File.



The Close Excel node is a **Python** node that was edited to the following (To edit this **Python** code double click the **Python** node).

Python Code to Close Excel

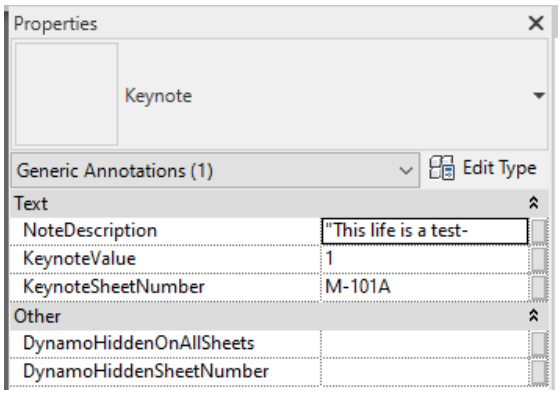
```
# Enable Python support and load DesignScript library
import clr
clr.AddReference('ProtoGeometry')
from Autodesk.DesignScript.Geometry import *

# The inputs to this node will be stored as a list in the IN variables.
dataEnteringNode = IN
# Place your code below this line

from System.IO import Directory
import re
import time
import sys
sys.path.           ('C:\Program Files (x86)\IronPython 2.7\Lib')
import subprocess
import os
```

```
import errno
import csv
import clr
clr.AddReferenceByName('Microsoft.Office.Interop.Excel, Version=11.0.0.0, Culture=neutral,
PublicKeyToken=71e9bce111e9429c')
from Microsoft.Office.Interop import Excel
from Microsoft.Office.Interop.Excel import ApplicationClass
from System.Runtime.InteropServices import Marshal

true_false = IN[0]

def excel_close(_bool):
    if _bool:
        xlApp = Marshal.GetActiveObject("Excel.Application")
        xlApp.Visible = True
        xlApp.DisplayAlerts = False
        xlApp.Workbooks.Close()
        xlApp.Quit()
        Check =  "Success"
    else:
        Check = "Set the boolean to True"

    return Check

OUT = excel_close(true_false)


# Assign your output to the OUT variable.
OUT = 0
```

Part 2 - MPE Keynote Note Block Automation with Dynamo

Task 1 – Hide Keynotes in Unwanted Views

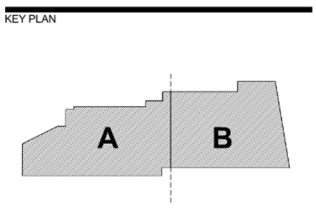
MPE teams often make use of the **Keynote Note Block** method to create **Keynotes** for their Construction Documents. Parameters are added to **Keynote Note Blocks** to designate the **Sheet Number**, **Keynote Value**, and **Description** as pictured below.



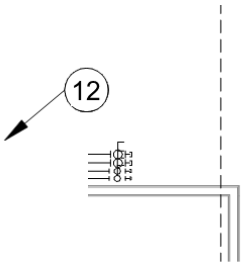
> *Note: Michael Massey detailed this process in his 2016 class. For more information on these Note Block Keynotes, refer to the references at the end of this document.*

While this method has its upsides, it also requires a different level of QA/QC before the documents can be released.

It is often much more efficient to create the **Keynotes** within dependent **Floor Plan Views** instead of on a **Sheet**. However, this method lends itself to more potential inaccuracies. For projects that contain multiple areas, **Keynotes** that are placed within a **Dependent Floor Plan** will sometimes show up in “Area B” when they were only written for “Area A”.



The **Keynotes** that straddle the line between multiple areas will sometime be visible in multiple **Views** which can result in the **Keynote Note Block** referring to inaccurate data or **Keynotes** referencing nothing visible on the **Sheet**.



Dynamo was determined to be one of the most efficient ways to eliminate **Keynotes** that are just outside of the **View** area of an active **View** window and thus ensure that the **Keynotes** on any one **Sheet** were valid and present.

A simple **Dynamo** script was created to QA/QC the **Keynotes** within the project which can be run at any time within the project.

Process 1

In addition to the ***KeynoteHide.dyn*** **Dynamo** script, it is also recommended that you download and transfer the following items to your project:

1. ***Keynote*.rfa**
2. ***Keynote Check* Schedule**

Utilize Revit 2020 and install the following **packages** which will also to ensure the script will **Run** successfully:

4. **Achirlab.net** 2020.22.1
5. **Rhythm** 2019.9.12
6. **spring nodes** 203.1.0

Upon a successful “Run” of the **KeynoteHide.dyn** script, the **Keynotes** that are potentially visible across multiple areas will be hidden on the views that are not user defined, and the **Keynote Check Schedule** will indicate which **Keynotes** have been hidden. The **Keynote Check Schedule** will look something like the example shown below.

<Keynote Check>

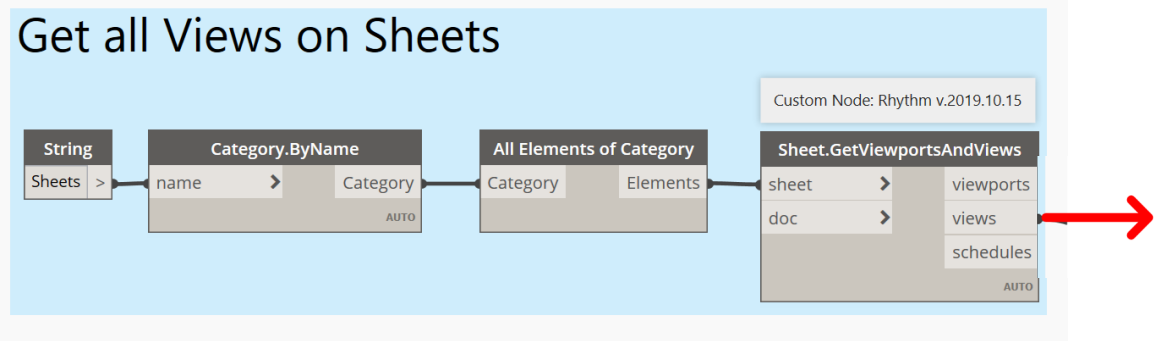
A	B	C	D	E
KeynoteSheetNumber	KeynoteValue	NoteDescription	DynamoHiddenOnAllSheets	DynamoHiddenSheetNumber
M-101A	1	"This life is a test-		
M-101A	2	it is only a test.		
M-101A	3	If it had been an actual life,		M-101B/M-101C
M-101A	4	you would have received further instructions		M-101C
M-101A	5	on where to go and what to do.		M-101B/M-101C
M-101A	6	Remember, this life is only a test."		M-101B/M-101C
M-101A	7	-Jack Kornfield		M-101B/M-101C
M-101B	1	DYNAMO SCRIPT BY:		M-101A
M-101B	2	Alanna Watts		M-101A
M-101C	1	Presentation by:		
M-101C	2	Alanna Watts		
M-101C	3	and Brandon Hartley		
M-101C	4	from Parkhill, Smith and Cooper, Inc		M-101A/M-101B
M-101C	5	THIS NOTE WAS RANDOM PREVIOUSLY HIDDEN ON THIS SHEET M-101C. IT WILL NOT BE VISIBLE ON ANY VIEW.	Yes	M-101B

Dynamo (In-Depth)

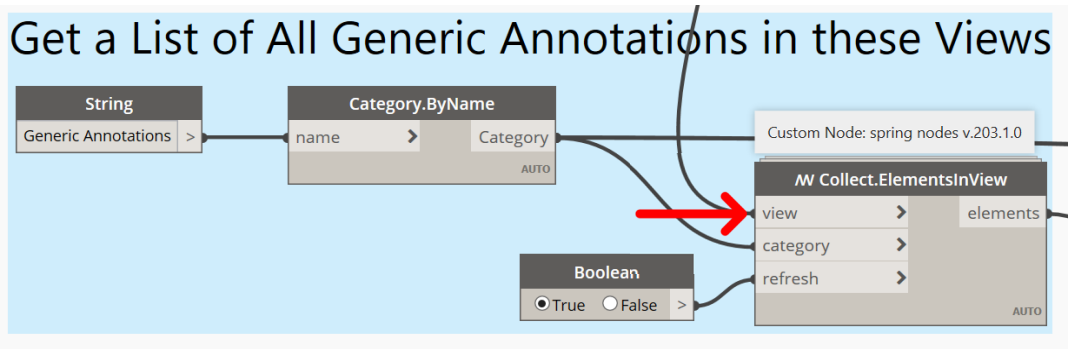
Hide Keynotes in Unwanted Views

Begin by selecting all the **Sheets** in the model. Use the **Rhythm** package node **Sheet.GetViewportAndViews** to obtain all of the **Views** on the selected **Sheets**.

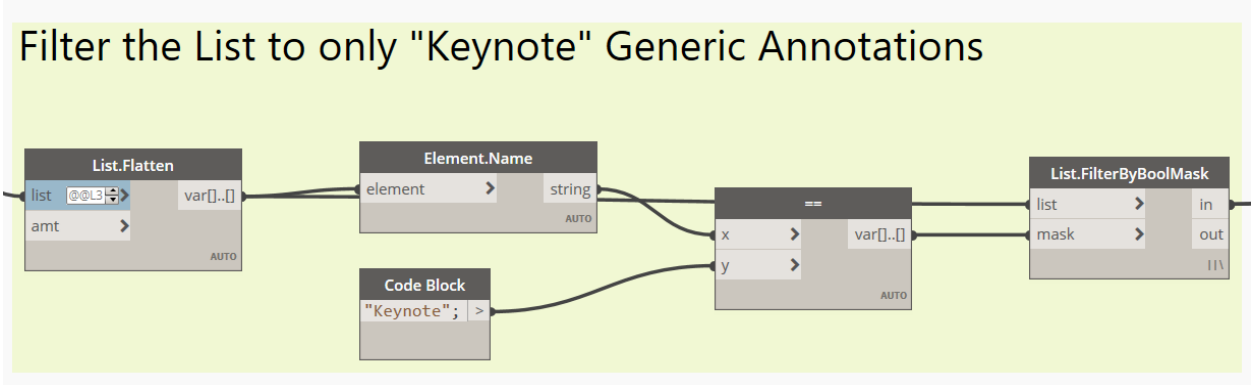
> *Note: The **Sheet** category is called with a string instead of the typical category list because users will sometimes experience different lists which may cause discrepancies in the selected value. Calling the **Sheet** category by a string avoids any changes in the category list which could ultimately affect the accuracy of the script.*



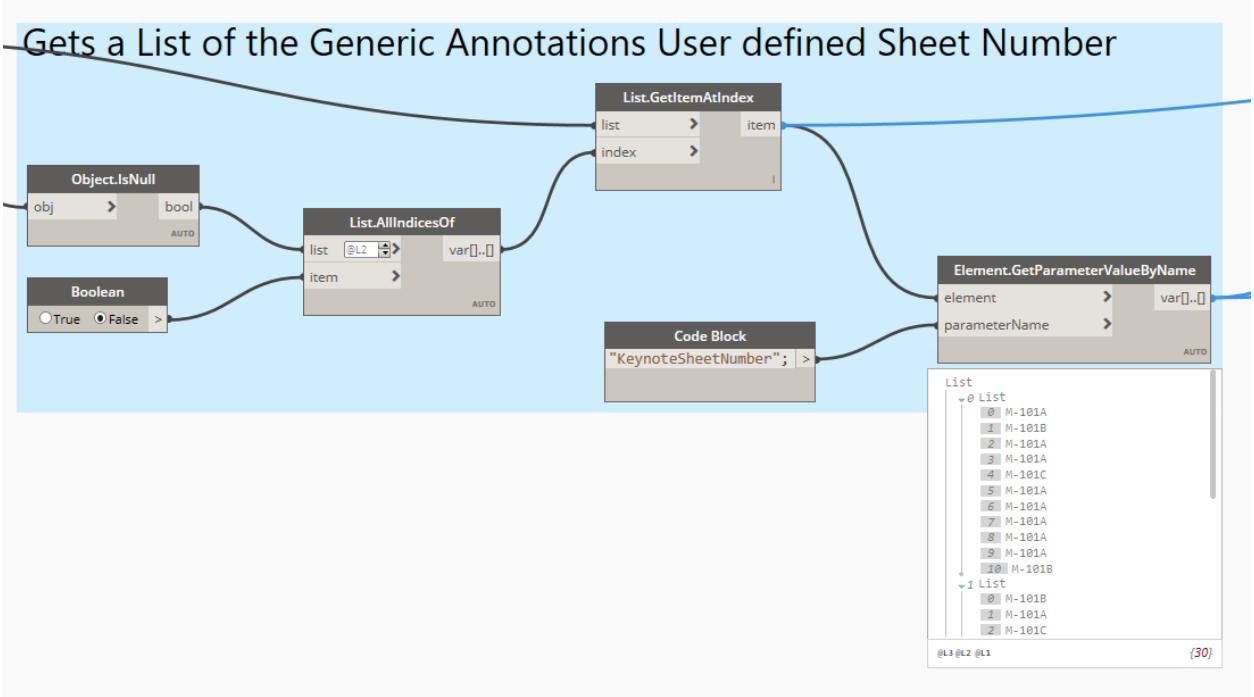
Use the **spring nodes** package node **Collect.ElementsInView** to get a list of all **Generic Annotations** that are placed in the **Views** that were identified.



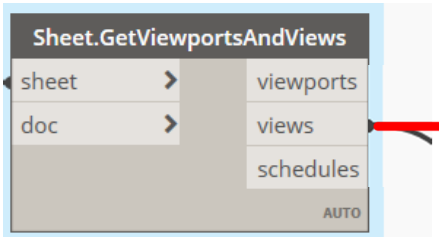
Flatten the **Generic Annotations** list and filter by using **List.FilterByBoolMask** to only those **Generic Annotations** that are named **Keynote**.

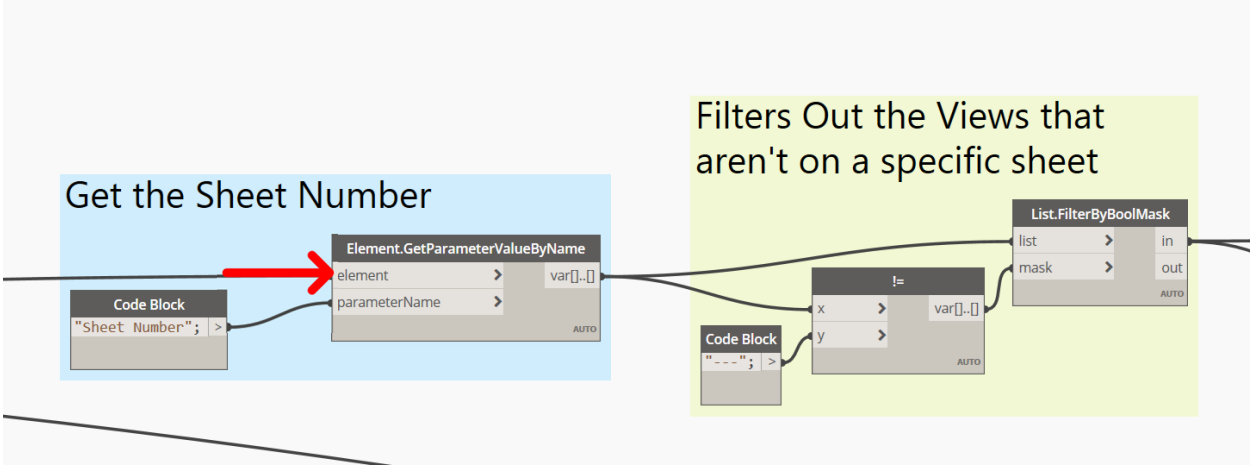


Get a list of the **Keynote** user defined **Sheet Numbers** based on the **KeynoteSheetNumber** parameter.

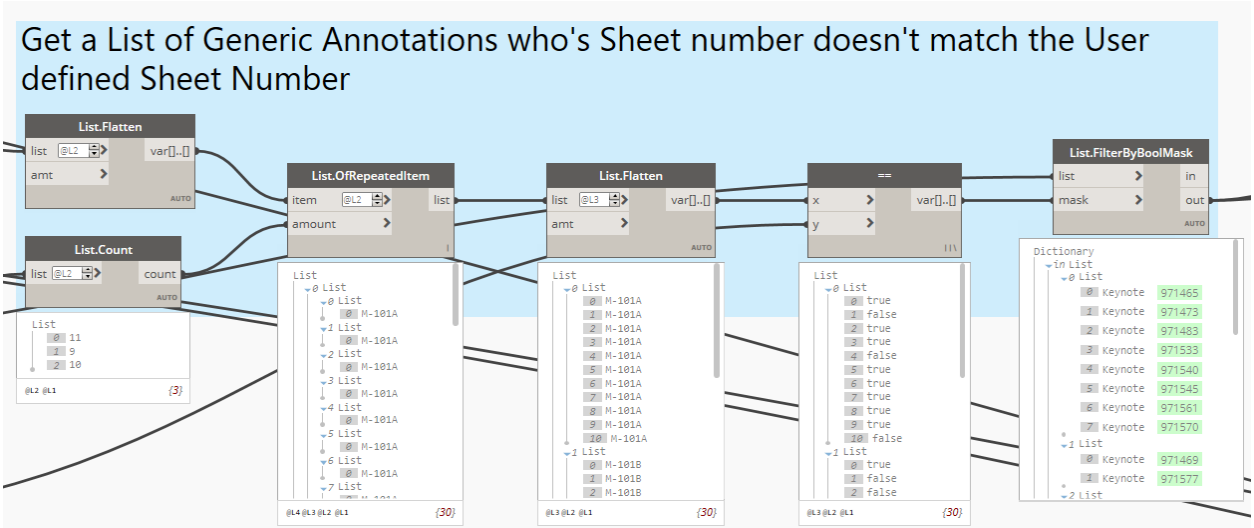


Get the **Sheet Number** of the **Views** and filter out any views that are not placed on a specific **Sheet**.

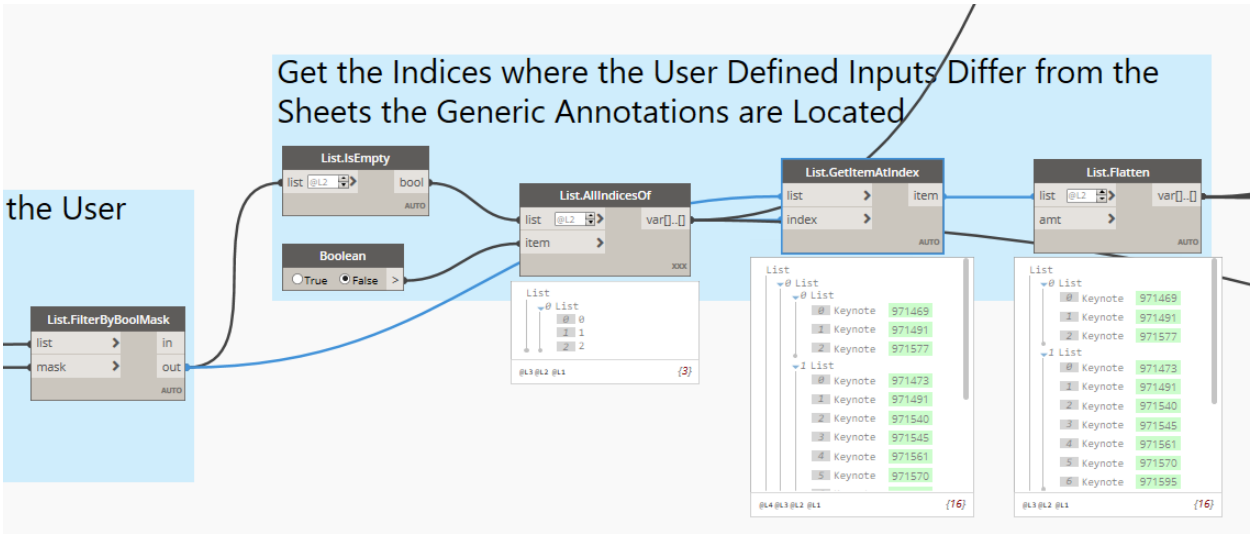




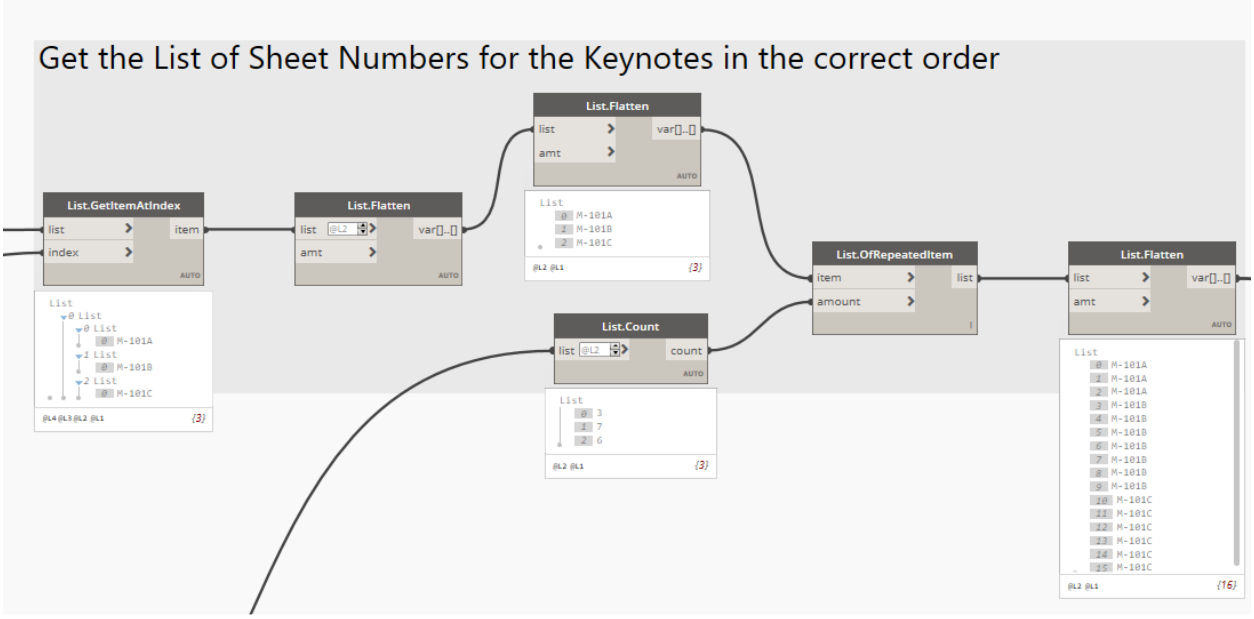
Get a list of **Generic Annotations** and user defined **Sheet Numbers** based on the **KeynoteSheetNumber** parameter and see if the **KeynoteSheetNumber** matches the **Sheet Number** on which the **Keynotes** are placed. Use this information to filter any **Keynotes** placed on inaccurate **Sheets**.



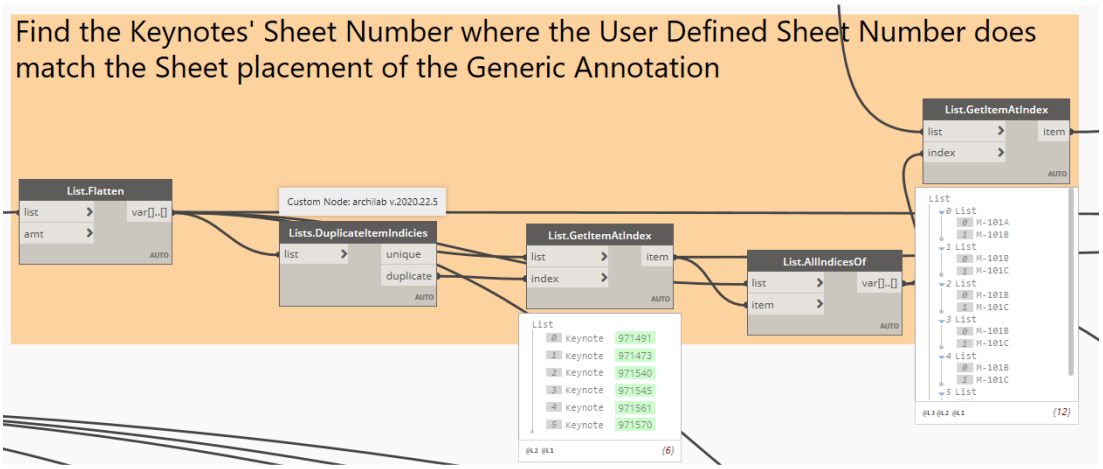
Get the indices where these **Keynote** discrepancies occur.



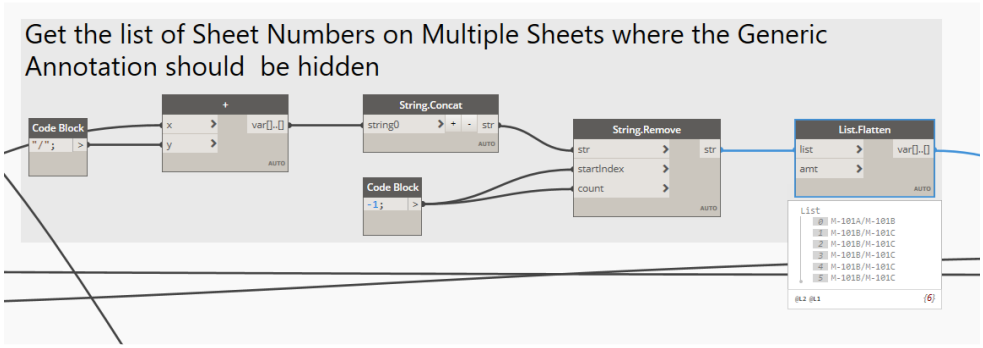
Get the list of the **Keynotes' Sheet Numbers** in the correct order.



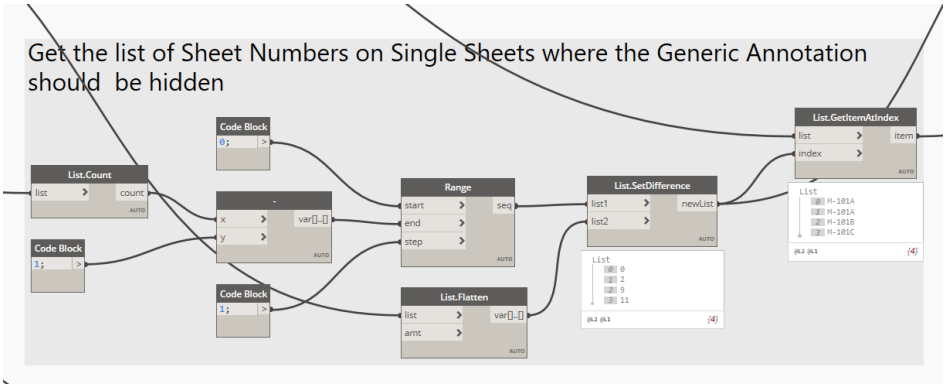
Use the **archilab** package node **Lists.DuplucateItemIndices** to determine the location of the indices where duplicates occur. These indices generate the **Keynotes'** list of **Sheet Numbers** for the **Keynotes** that are on the wrong **Sheet**. This list needs to be created in order to write these **Sheet Numbers** to the **DynamoHiddenSheetNumber** parameter.



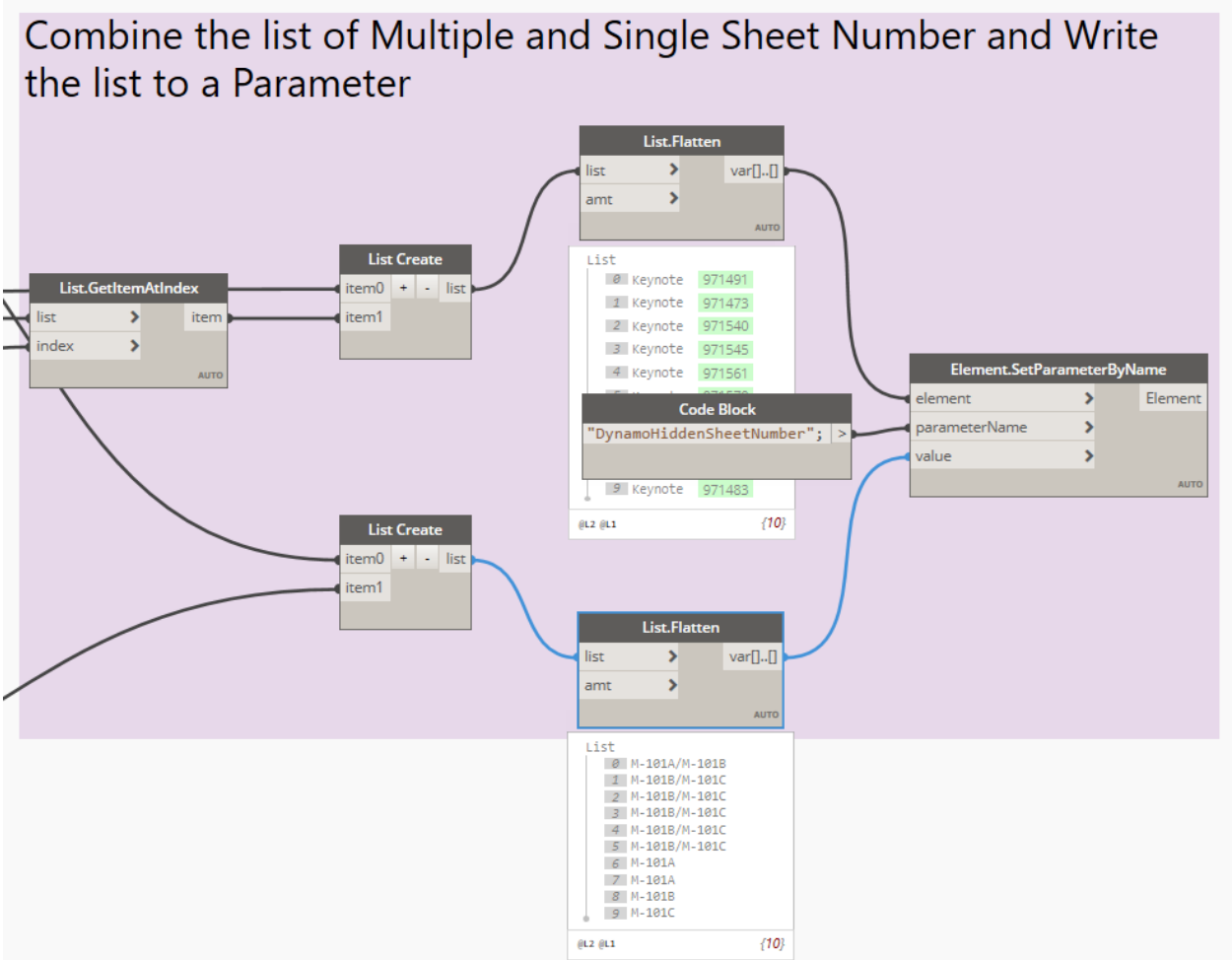
Combine these **Sheets Numbers** to one list so that the incorrect **Sheet Numbers** can be placed on one parameter.



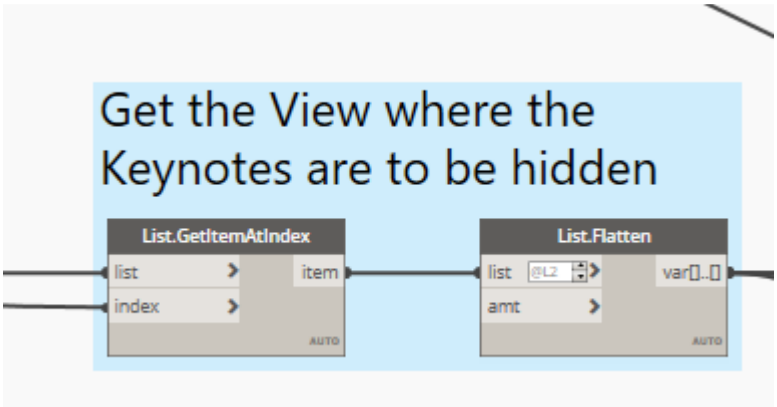
Get the list of **Keynotes** that need to be hidden on a **Sheet**.



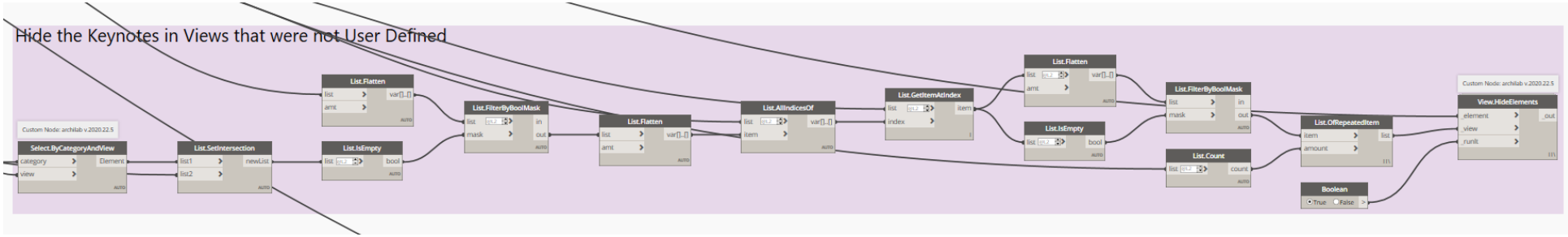
Combine these two lists and write the **Sheet Numbers** to the **Keynotes** for the **DynamoHiddenSheetNumber** parameter.



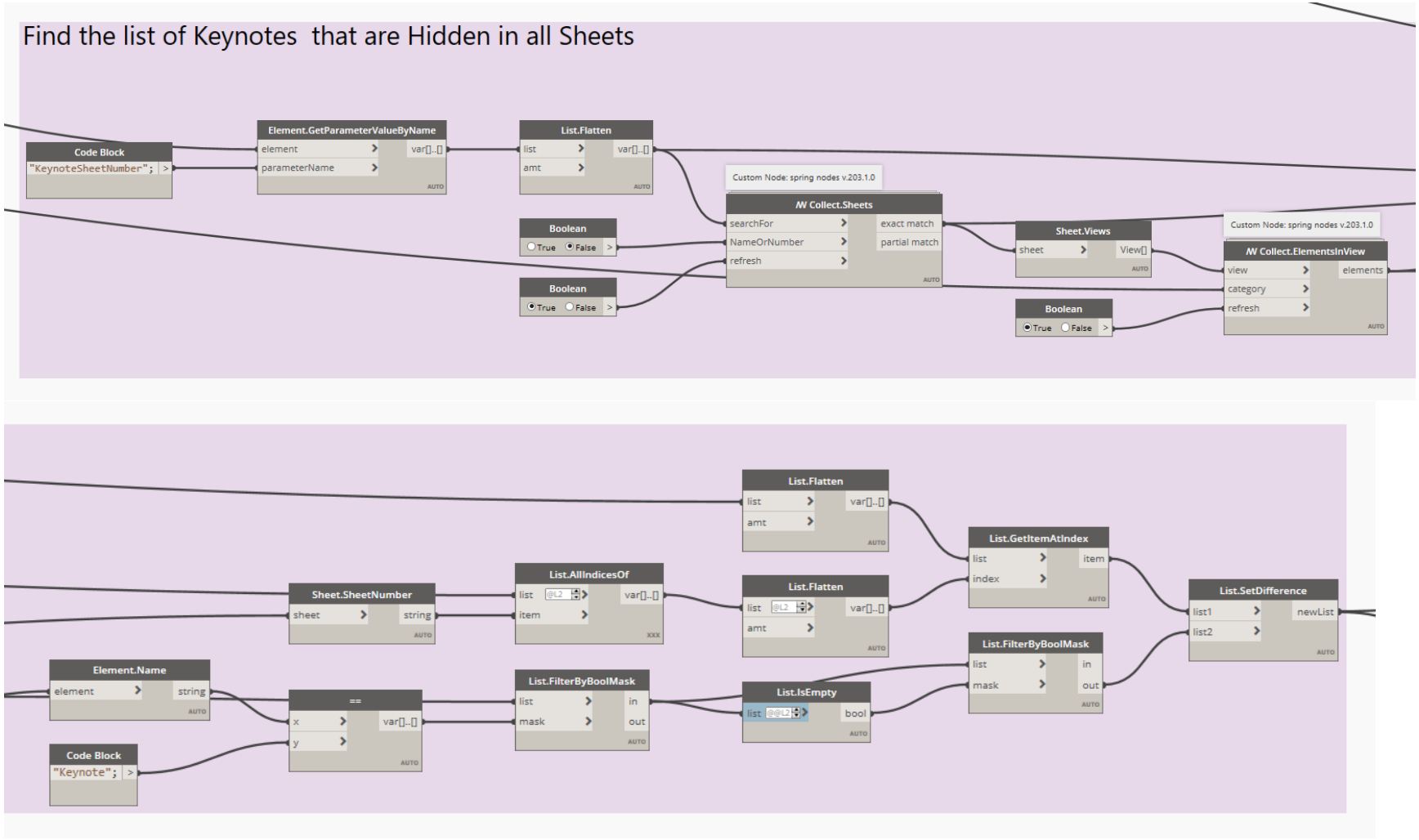
Get the **Views** where the **Keynotes** are to be hidden.



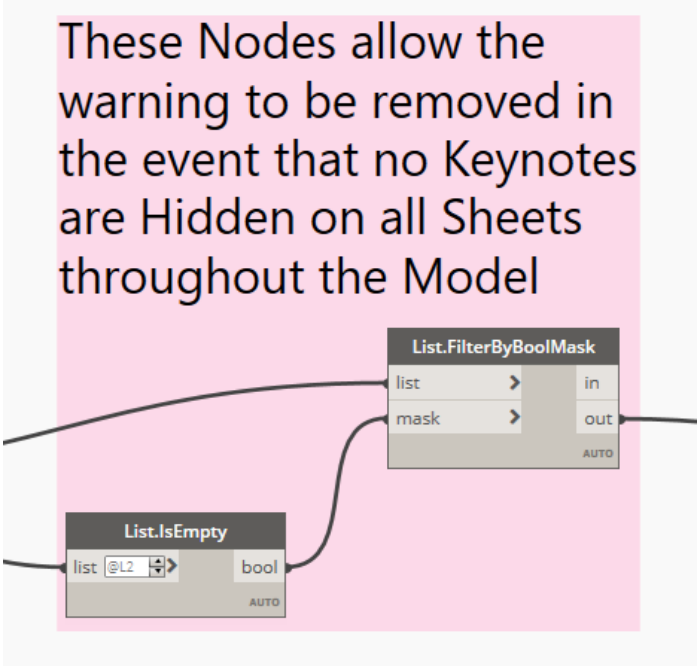
Use the archilab package node Select.ByCategoryAndView to determine the List.SetIntersection where the **Keynotes** should not be placed. Filter and flatten these lists to obtain the **Views** where the **Keynotes** need to be hidden. Use the archilab package node View.HideElements to hide the **Keynotes** in appropriate **Views**.



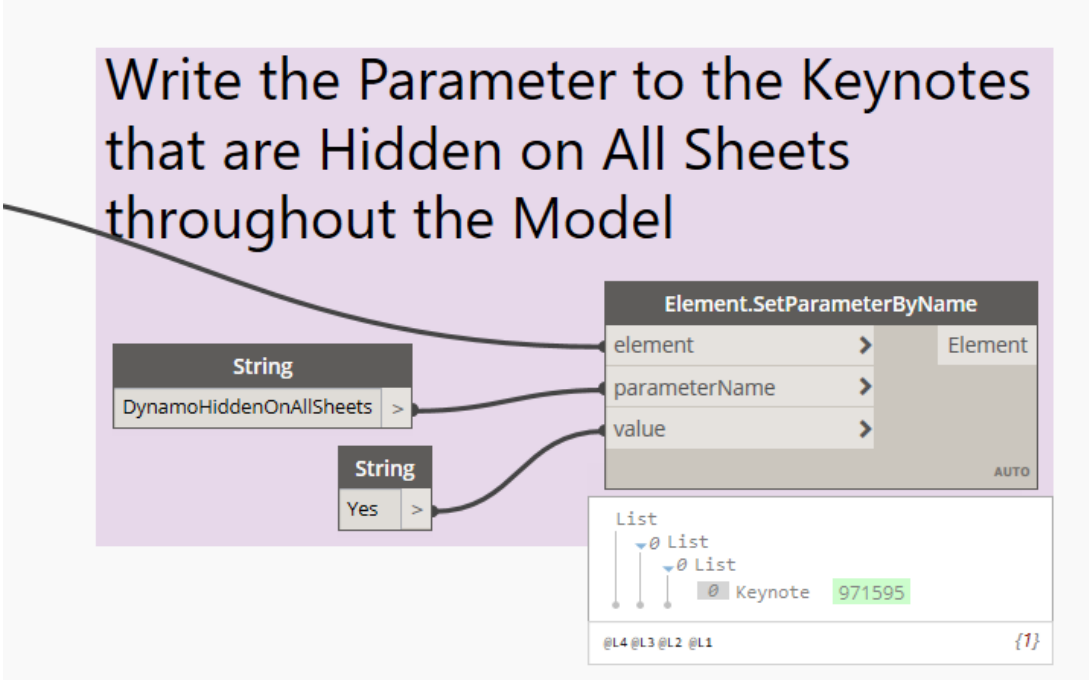
Use the **spring nodes** package to node ***MW Collect.Sheets*** to determine the list of **Sheets**. Then get the list of **Sheet Views** and again use the **spring nodes** package node ***MW Collect.ElementsInView*** to determine a list of **Keynotes** that are hidden on all **Sheets**.



If no **Keynotes** are hidden in all **Views**, the script will generate a warning message. In order to remove this warning message, a filter has been added.

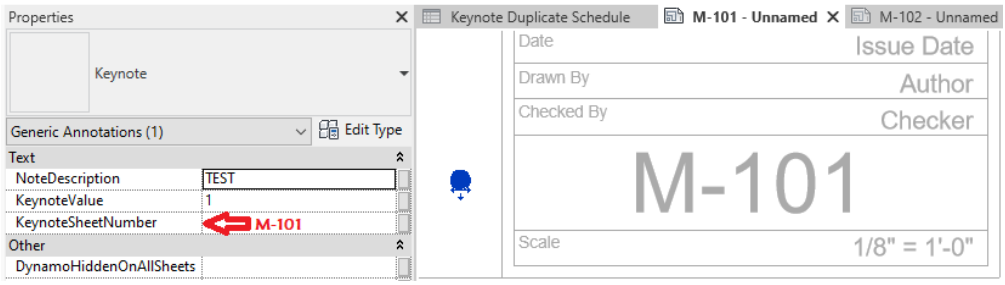


The **DynamoHiddenOnAllSheets** parameter is written to indicate which **Keynotes** need further review.

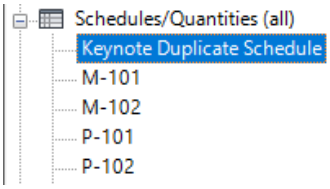


Task 2 – Number Keynotes Based on Sheet Number, Create Note Block Schedules for Each Sheet and Place these Schedules on the Sheets

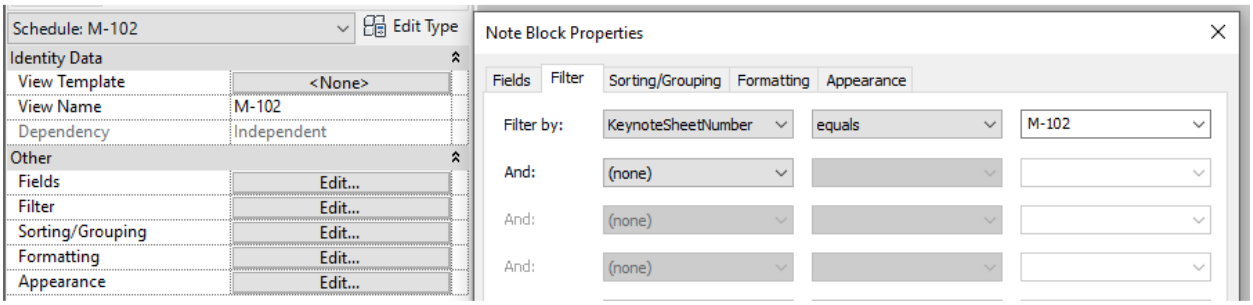
Use the ***KeynoteLabelAndPlaceOnSheets.dyn*** Dynamo script to change the **KeynoteSheetNumber** parameter to equal the **Sheet Number** on which it is placed.



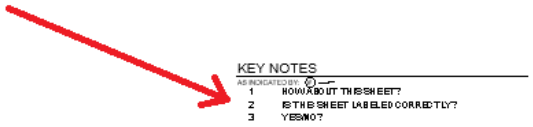
This script will duplicate the ***Keynote Duplicate Schedule*****,** that was provided with the dataset files, for each different **Sheet Number** on which a **Keynote** is placed.



It will then set a filter on the newly created **Schedules** to "*Filter by:*" **KeynoteSheetNumber** which will match the **Sheet Number** on which the **Schedule** is to be placed.



Finally, this script will place these **Schedules** on their respective **Sheets** that match the **Sheet Number** to which the **Schedule** belongs.



Note: This script is not to be used on projects with dependent views and multiple areas unless the Keynote have been placed on the Sheets themselves.

Process 2

In addition to the ***KeynoteLabelAndPlaceOnSheets.dyn*** Dynamo script, it is recommended that you download and transfer the following items to your project:

2. ***Keynote*.rfa**
3. ***Keynote Duplicate Schedule***

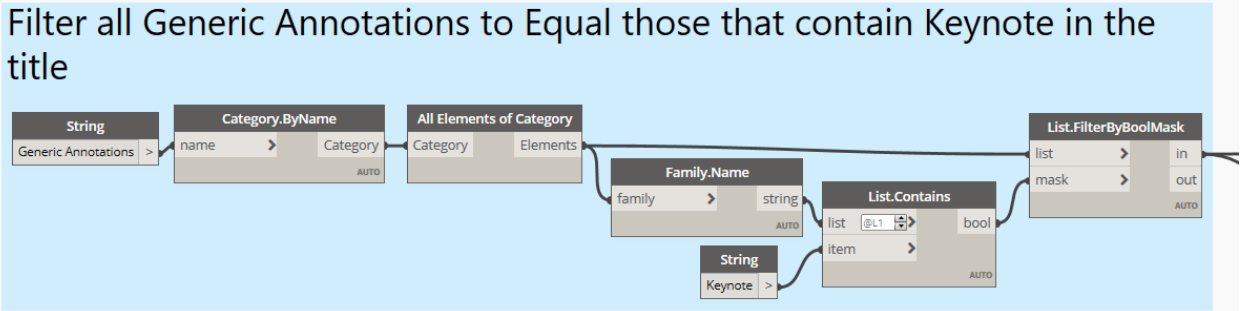
Utilize Revit 2020 and install the following **packages** which will also to ensure the script will **Run** successfully:

1. **Clockwork for Dynamo 2.z** 2.1.2
2. **Genius Loci** 2019.9.26
3. **MEPover** 2019.9.3
4. **Rhythm** 2019.9.12

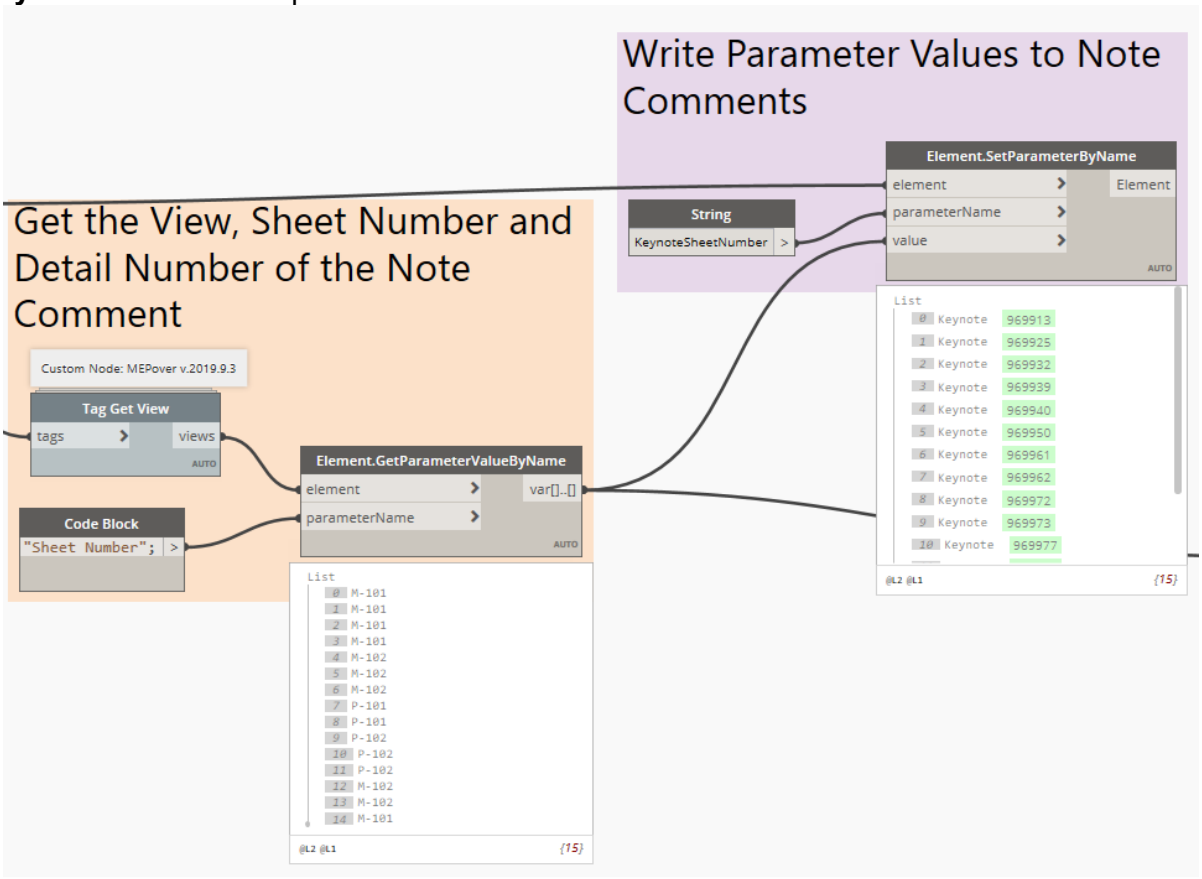
Dynamo (In-Depth)

Number Keynotes Based on Sheet Number, Create Note Block Schedules for Each Sheet and Place these Schedules on the Sheets

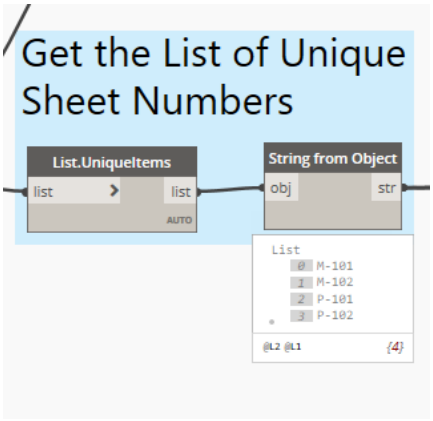
Begin by Selecting all the **Generic Annotations** in the model. **Filter** the **Generic Annotations** to only those that are **Keynotes**.



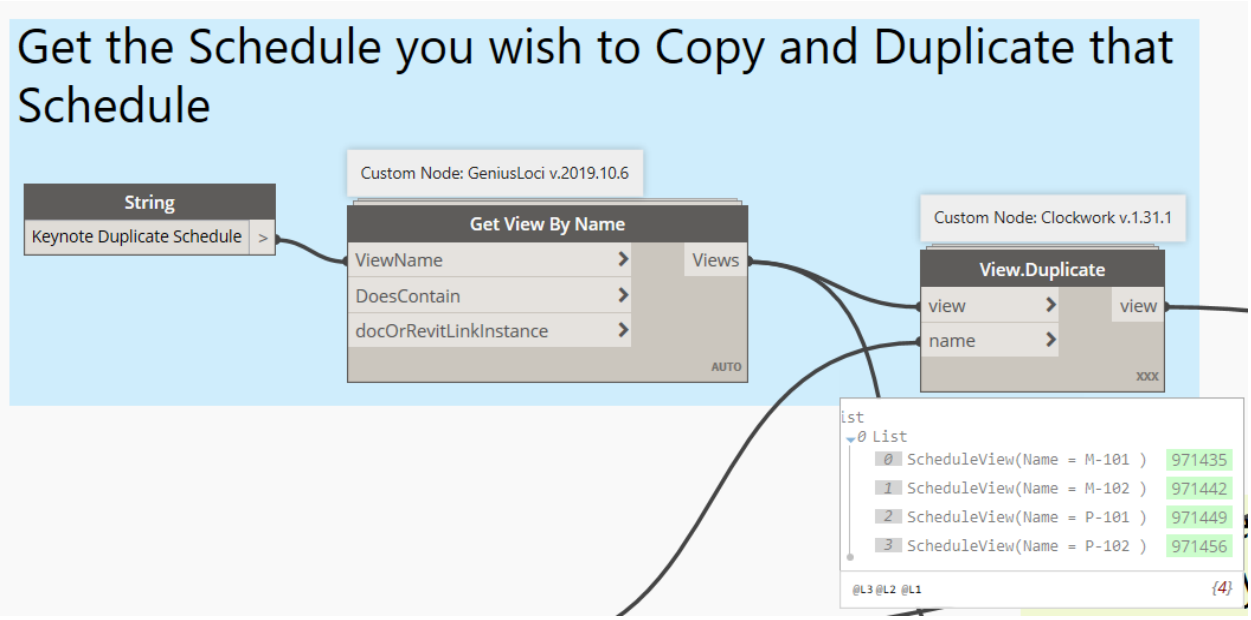
Use the **MEPover** package node **Tag Get View** to locate a **Sheet Number** on which the **Keynote** is placed. Write this information back to the **Keynote Note Block** through the **KeynoteSheetNumber** parameter.



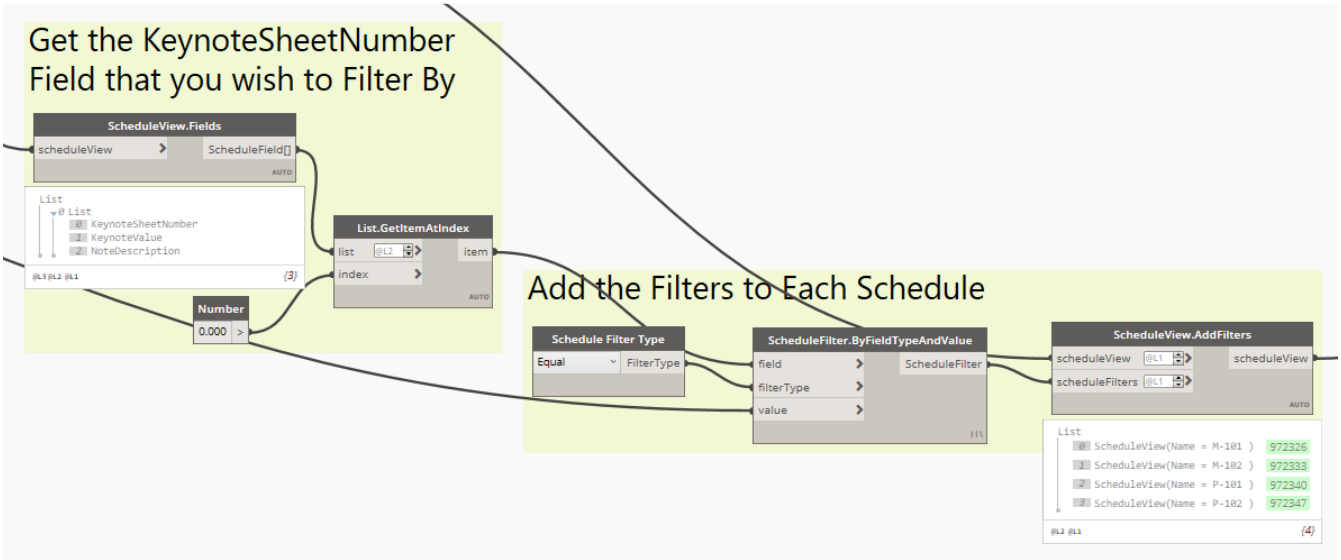
Get a list of all the unique **Sheet Numbers** that the **Keynotes** are placed.



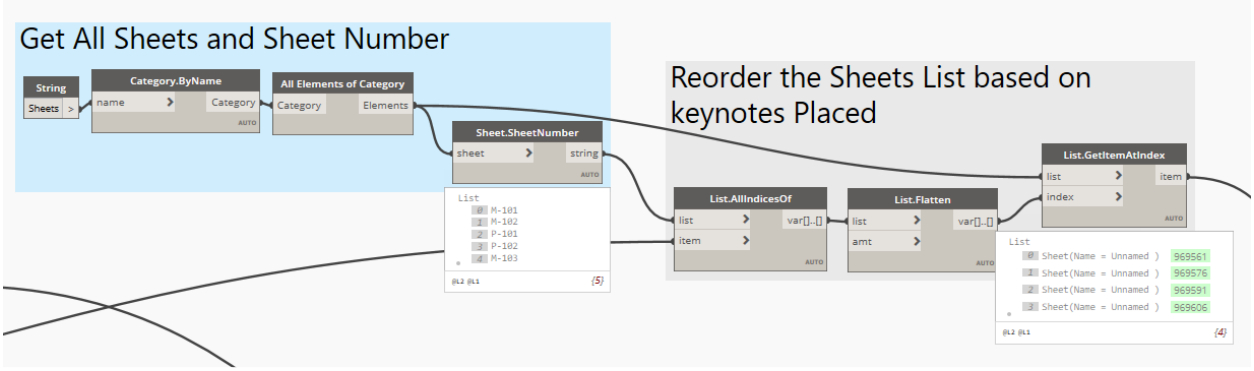
Use the **GeniusLoci** package node **Get View By Name** to get the **Note Block Schedule** that you wish to duplicate. Use the **Clockwork** package node **View.Duplicate** to duplicate the **View** equal to the number of times a unique **Sheet Number** exist.



Get the parameter you wish to ilter the newly created schedules by **KeynoteSheetNumber** and add that **Filter** and the appropriate filter value to each **Note Block schedule**.

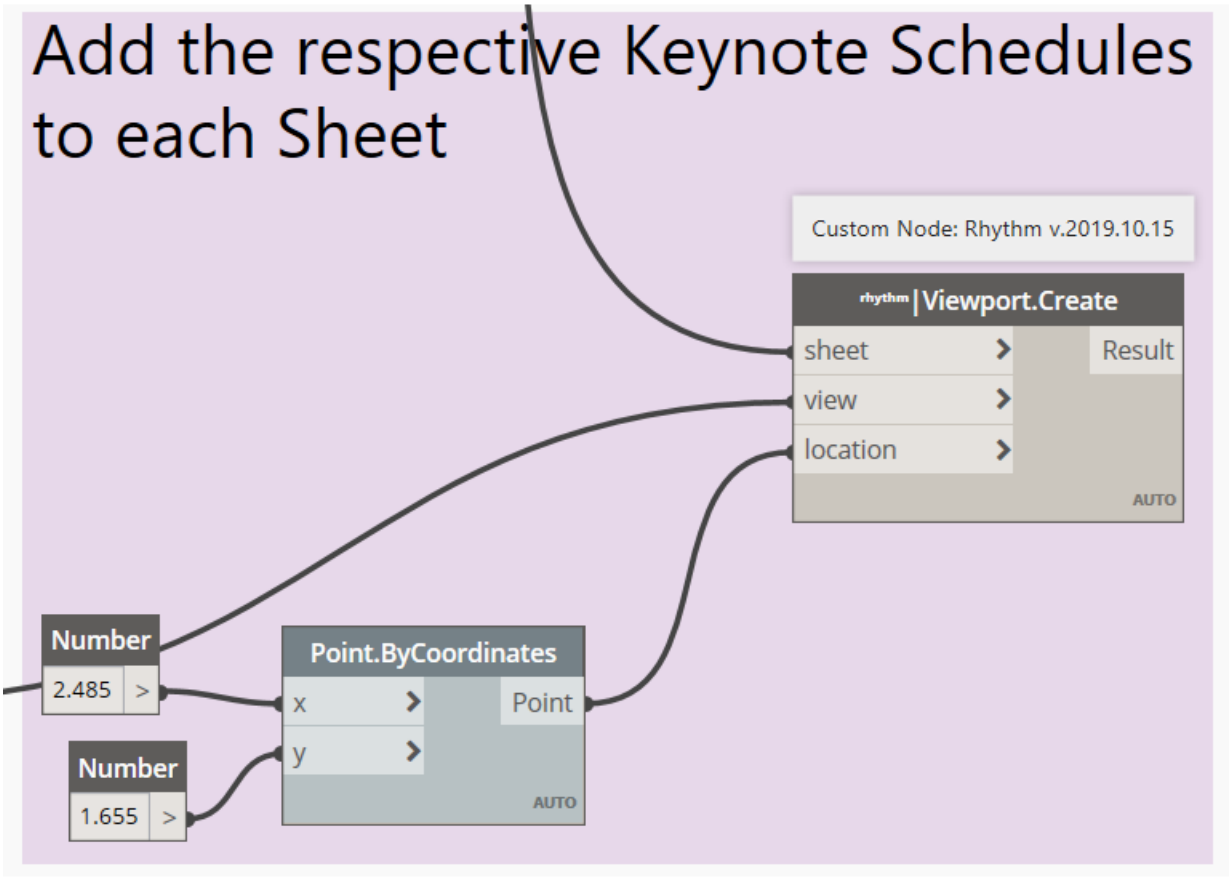


Get a list of **Sheet Numbers** in the project and filter this list to select only the **Sheets** that contain the **Keynotes**.



Use the **Rhythm** package node rhythm |**Viewport.Create** to add the duplicated **Keynote Note Block** schedules to their respective **Sheets**.

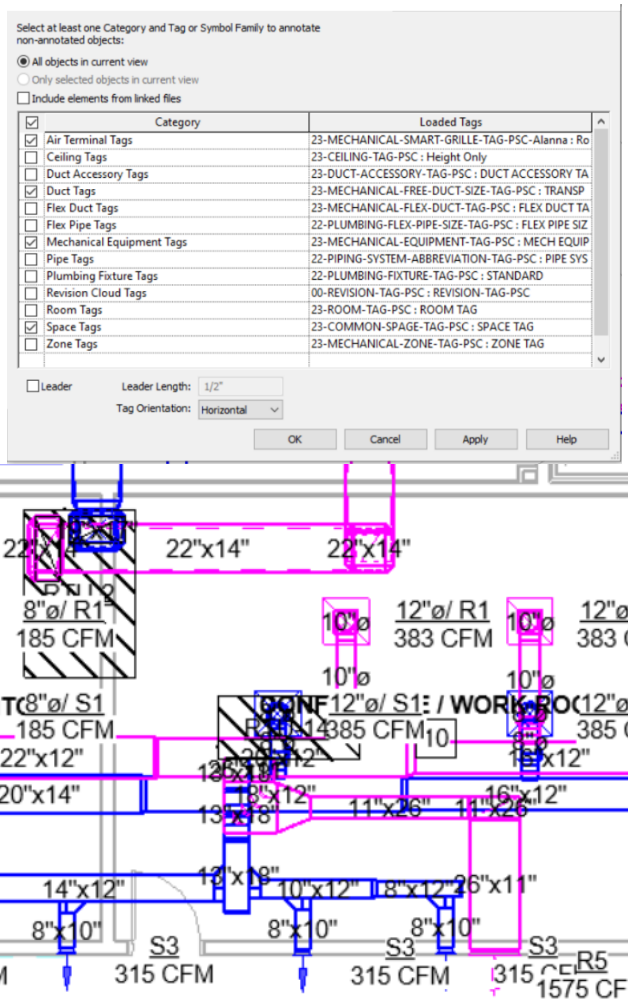
Note: You will want to change this location based on the Titleblocks that you use.



Part 3 - Mechanical Ductwork Plans – Tagging Automation with Dynamo

Task – Tag Mechanical Equipment In Selected View in Blank Locations

It can be time consuming to individually tag plans and move all the text to locations that are readable. If an individual uses the **Tag All** option within Revit, the tags inevitably end up on top of other visible elements in the plan. The need to reduce the time spent at the end of the project and help ensure that the details are correct and polished is a task that is achievable using Dynamo.



Note: A script for tagging an overall ductwork plan has been included in your course materials. Only the mechanical equipment tagging portion has been included in this handout as it gives a reasonable picture of how the overall task is able to be accomplished with Dynamo.

For more information on automating air terminal CFM's and sizes, reference Jason Boehning's class "Dynamo Design Script for MEP" provided during AU 2017. Refer to the references at the end of this document for more information.

Process

In addition to the **TagScriptMechanicalEquipmentOnly.dyn** Dynamo script, it is recommended that you download and transfer the following items to your project:

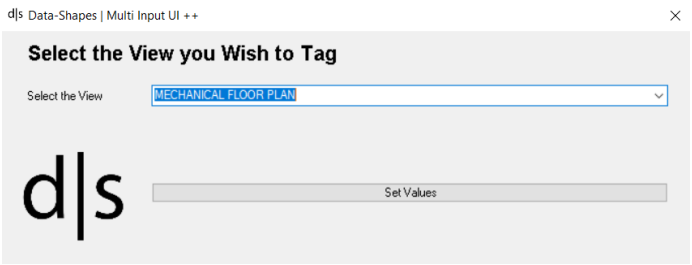
1. ***Mechanical-Equipment-Tag.rfa***

The user needs to ensure that the clearances for any equipment are not included in the families, unless these clearances will remain visible on the printed plans. Currently, this script will read the mechanical equipment length and width based on the family's overall length and width which also includes any potential clearance zones created within the families.

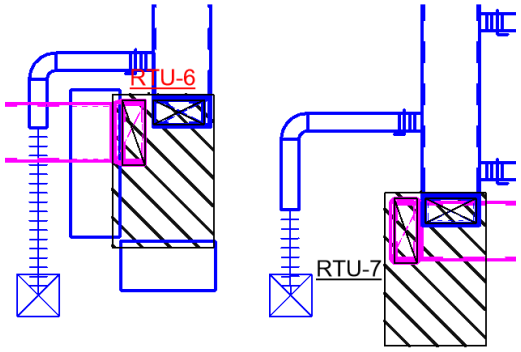
Utilize Revit 2020 and install the following **packages** which will also to ensure the script will **Run** successfully:

7. **Achirlab.net** 2020.22.1
8. **Data-Shapes** 2019.2.34
9. **spring nodes** 203.1.0

The Dynamo script, as provided, utilizes **Data-Shapes** to select the **View** the user wishes to tag.



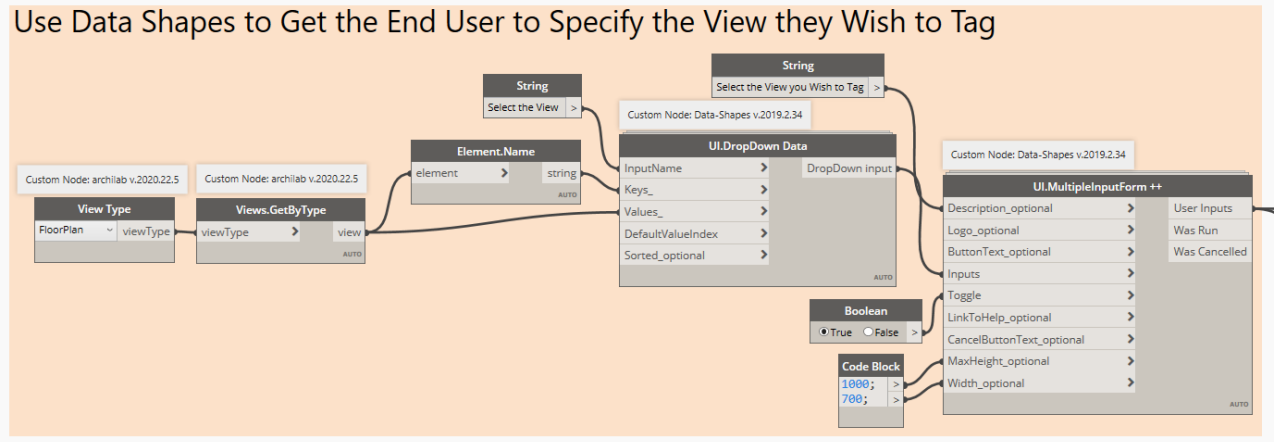
Upon a successful **Run** of the **TagScriptMechanicalEquipmentOnly.dyn** script, all mechanical equipment will be tagged on either the North, East, South and West positions depending on which position is free of any visible elements. Should all locations contain clashes with visible plan elements, the script will highlight these tags in red in order to draw the user's attention to the tags which will need to be manually adjusted. Additional locations for tagging could be easily incorporated to this script, but with each additional location option, the run time will increase as the additional bounding boxes will detect additional clashes between every existing element located on the view and those additional location options.



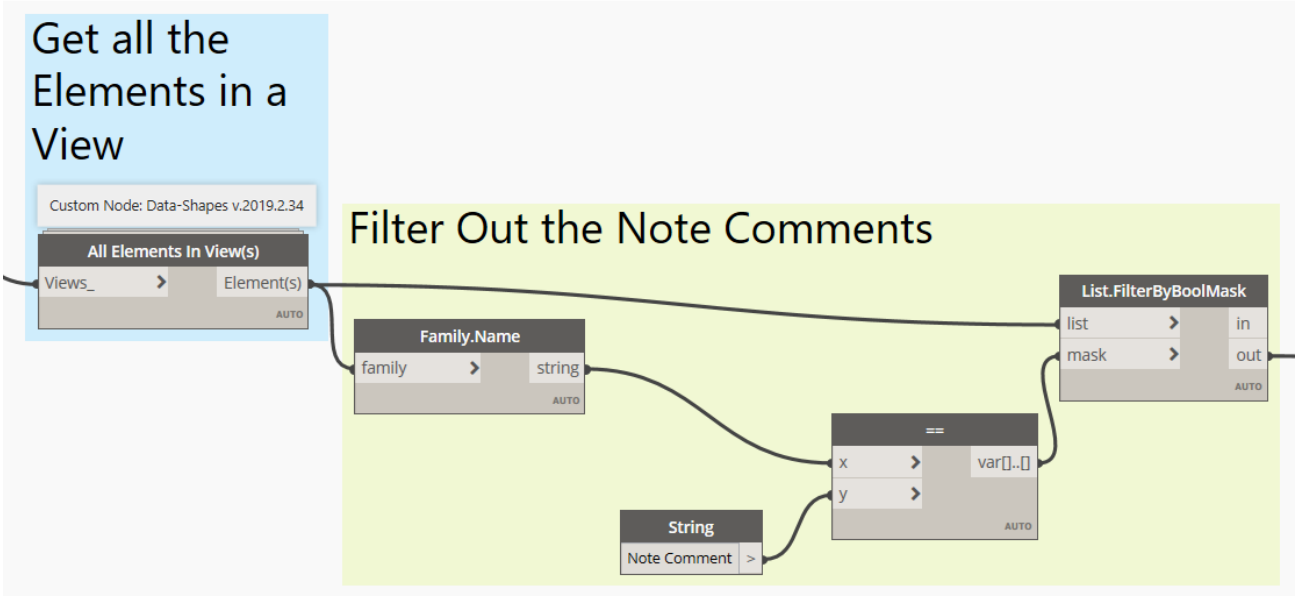
Dynamo (In-Depth)

Tag Mechanical Equipment in Selected View In Blank Locations

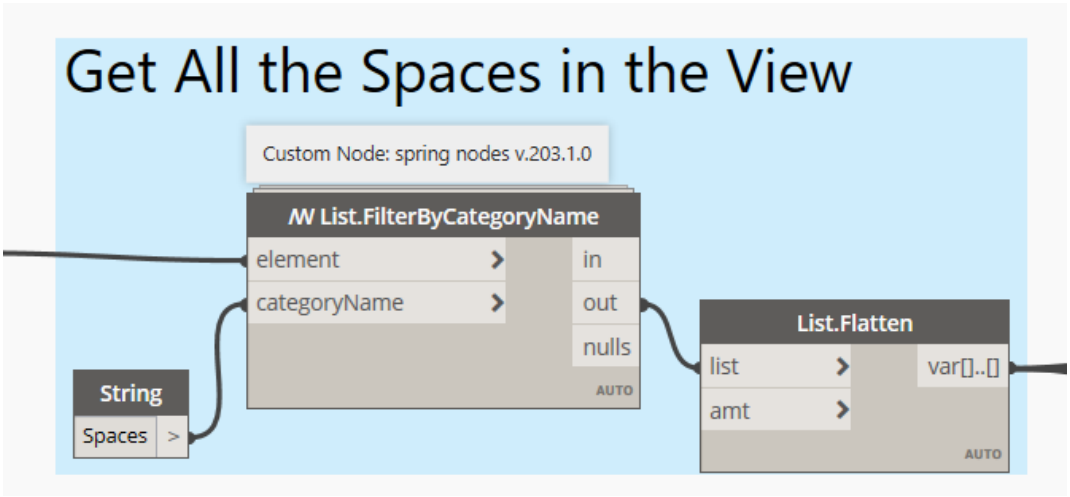
Begin by using the **achirlab** package nodes **View Type** and **Views.GetByType** to select all the **Floor Plan Views** in the project. This information will be provided to the **Data-Shapes** nodes allowing the user to select which **View** they want to tag. Use the **Data-Shapes** package node **UI.DropDown Data** and **UI.MultipleInputForm ++** to obtain the **View** that will be tagged.



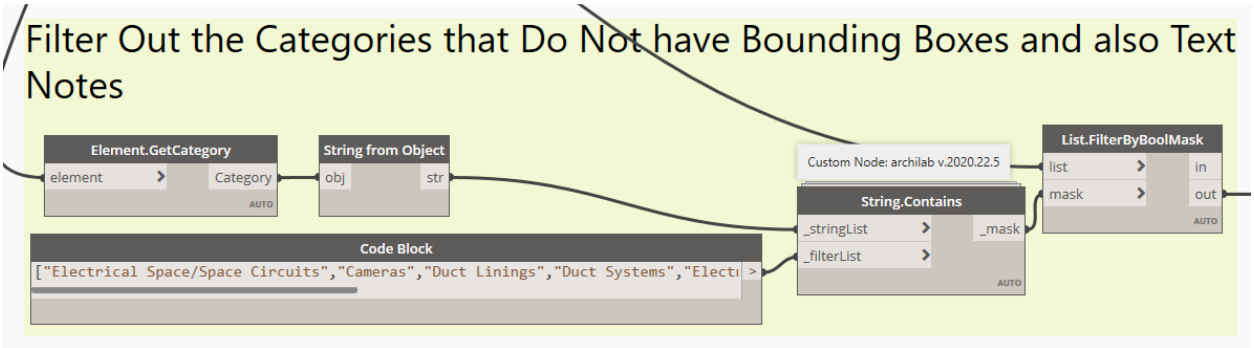
Use the **Data-Shapes** package node **All Elements in View(s)** to obtain all the elements within the **View**. If you are utilizing the part 1 DynamoAutomation with Windows Task scheduler and **Note Comment** process of this class, Filter out the **Note Comments** using **List.FilterByBoolMask**.



Use the **spring nodes** package node **MW List.FilterByCategoryName** node to filter all the **Spaces** in the **View** out of the elements list, as these **Spaces** will appear to be solid to the future **Element.BoundingBox** node that will be used in this script.

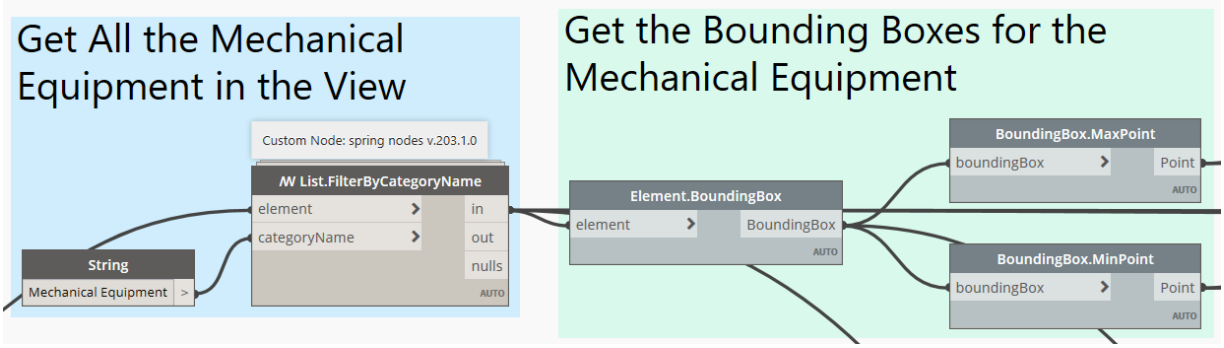


Filter out all the **Categories** that do not have Bounding Boxes and any **Text Notes**. Ideally, no **Text Notes** will have been placed on the selected floor plan prior to running this Dynamo script.

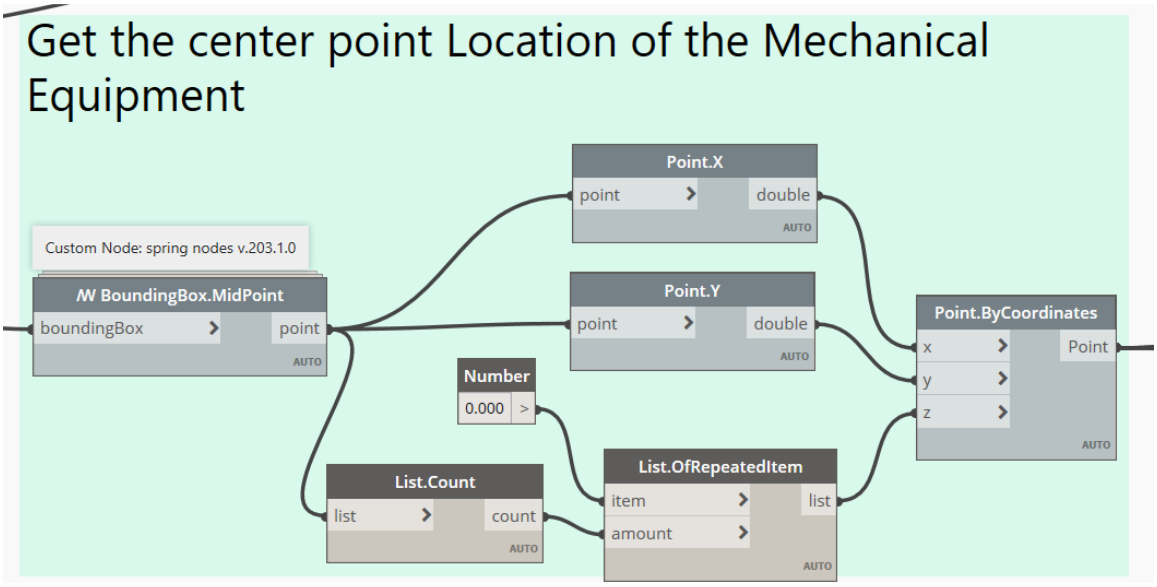


Note: Your project may have additional Categories that do not contain Bounding Boxes. To remove warning messages, add these additional Categories to the Filter list Code Block.

Use the **spring nodes** package node **List.FilterByCategoryName** node filter all **Mechanical Equipment**. Use the **Dynamo** native nodes **Element.BoundingBox**, **BoundingBox.MaxPoint**, and **BoundingBox.MinPoint** to determine the surrounding area locations of the **Mechanical Equipment**.

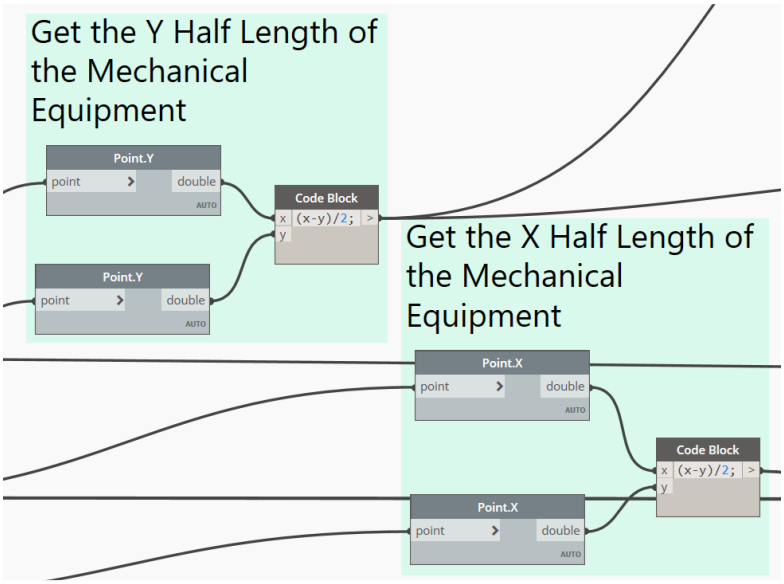


Use the **spring nodes** package node **BoundingBox.MidPoint** to obtain the midpoint for all **Mechanical Equipment**.

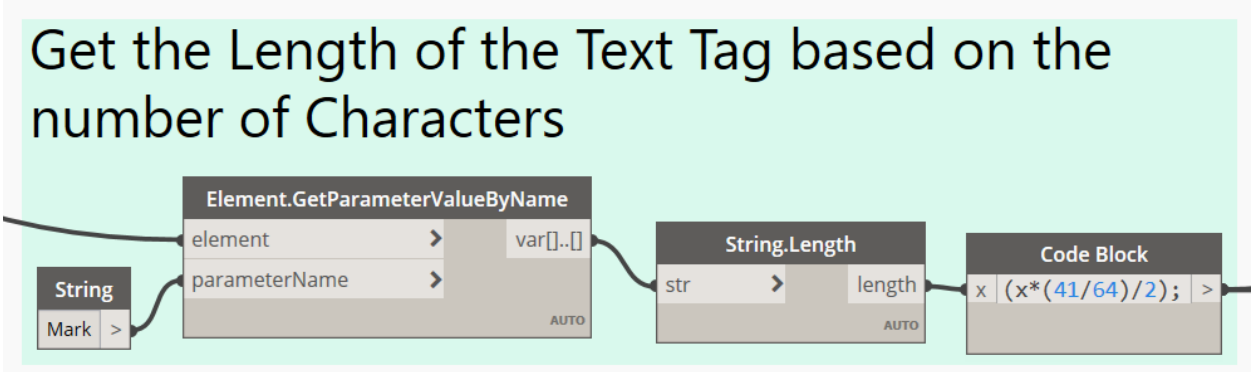


Note: The Blue/Grey nodes are in "Preview" mode to ensure that no additional analytical points are generated by running the Dynamo script which would require additional processing power and increase the Run time of the script.

Next, we will need to get half the length in the X and Y direction for the **Mechanical Equipment**.

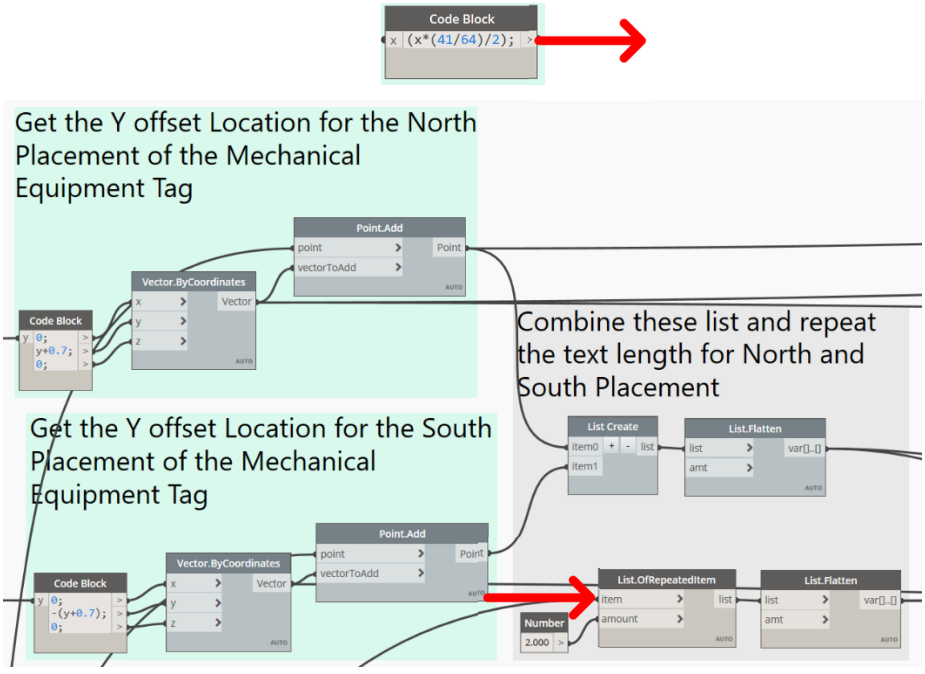


Obtain the approximate size of the Tag Text by estimating the distance of a text string.

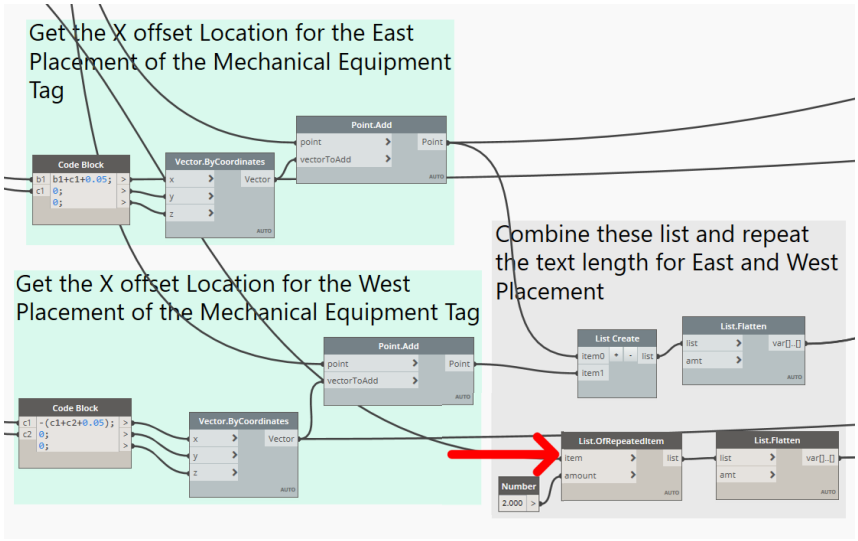


*Note: The text size is not an exact length as each text value is a slightly different in length. The **Code Block** above estimates the distance based on a 3/32" text size.*

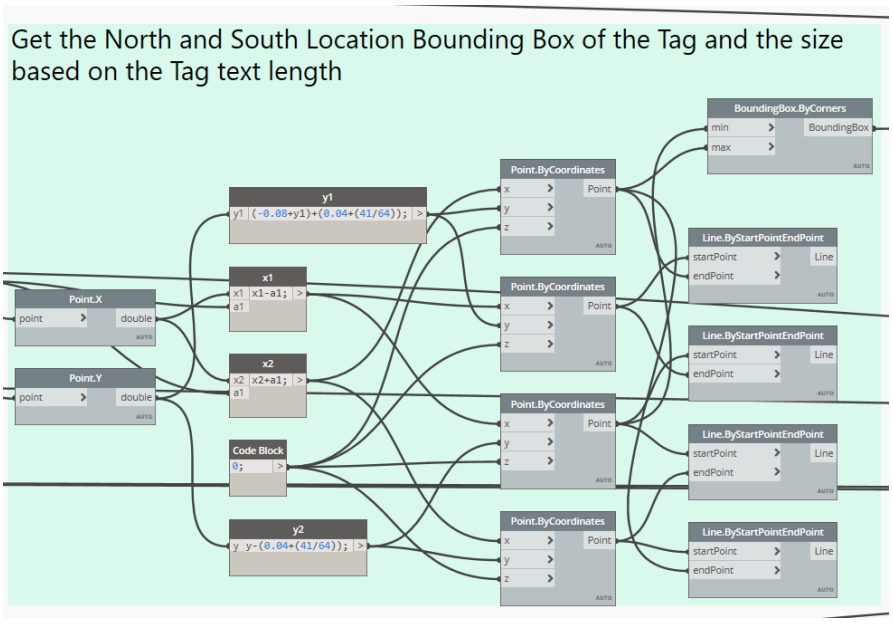
Get the Y offset location for the North and South Placement of the **Mechanical Equipment Tag** and combine these lists. Collect the string length and repeat so each list is equal.



Get the X offset location for the East and West Placement of the **Mechanical Equipment Tag** and combine these lists. Again, collect the string length and repeat so lists are equal.

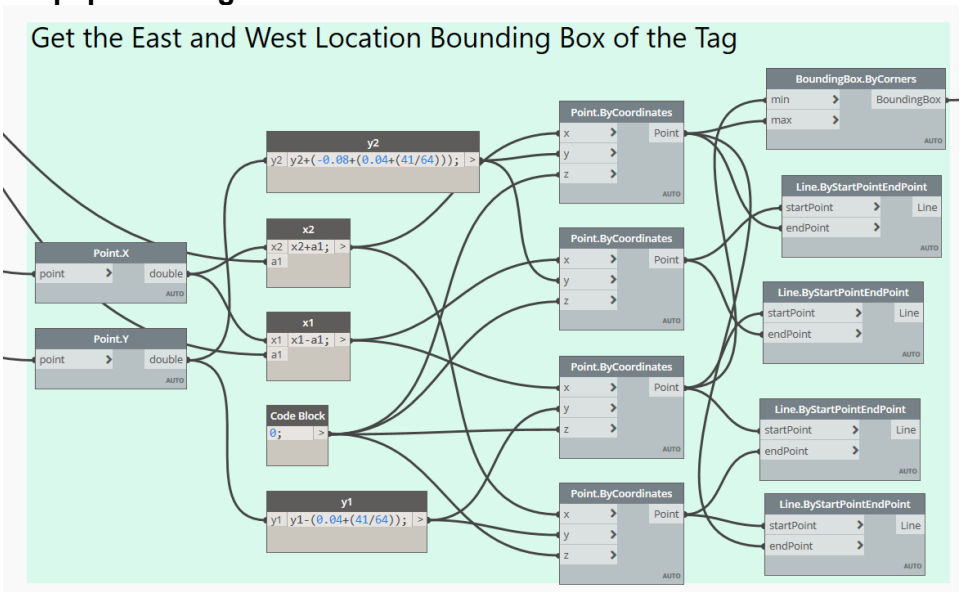


Obtain the North and South placement Bounding Box locations and sizes for the **Mechanical Equipment Tags**.

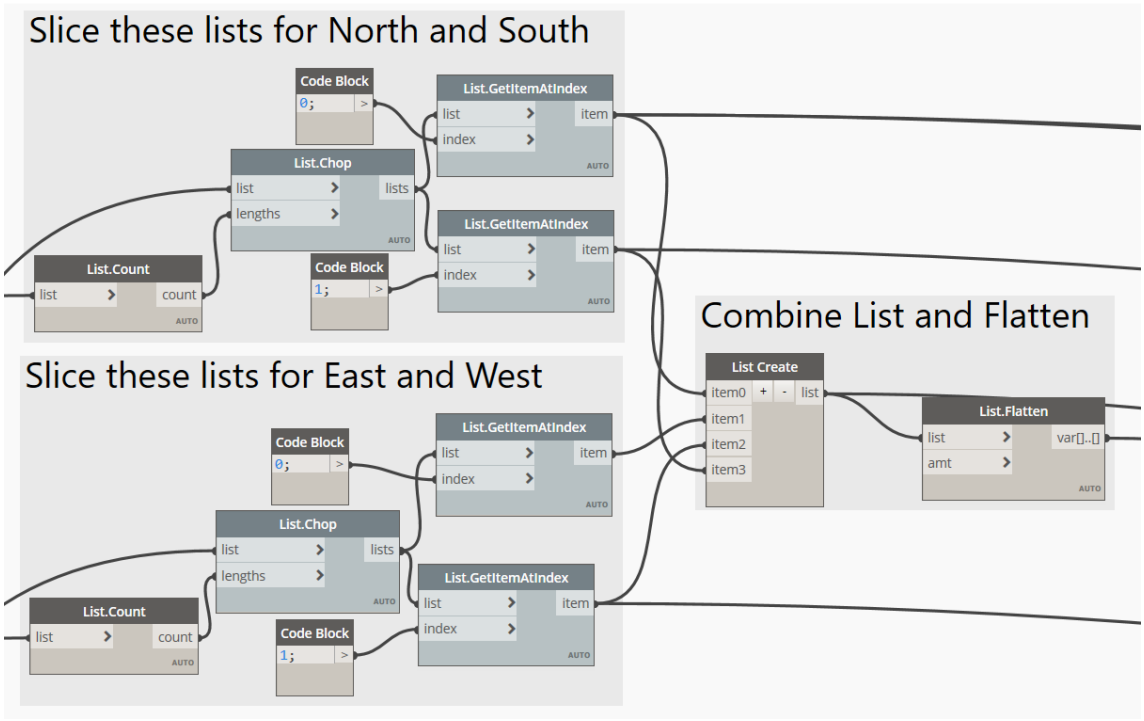


*Note: The **Line.ByStartPointEndPoint** node is not necessary for the script to Run, but should the text size change, these nodes are useful to right click and Preview the potential text size and location of the **Mechanical Equipment Tags**.*

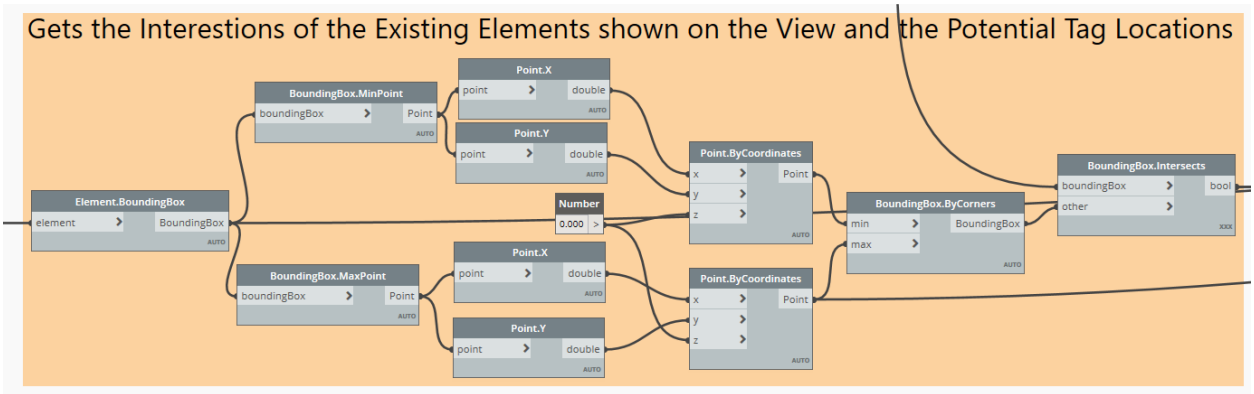
Repeat this for the East and West Placement Bounding Box Location and Sizes for the **Mechanical Equipment Tags**.



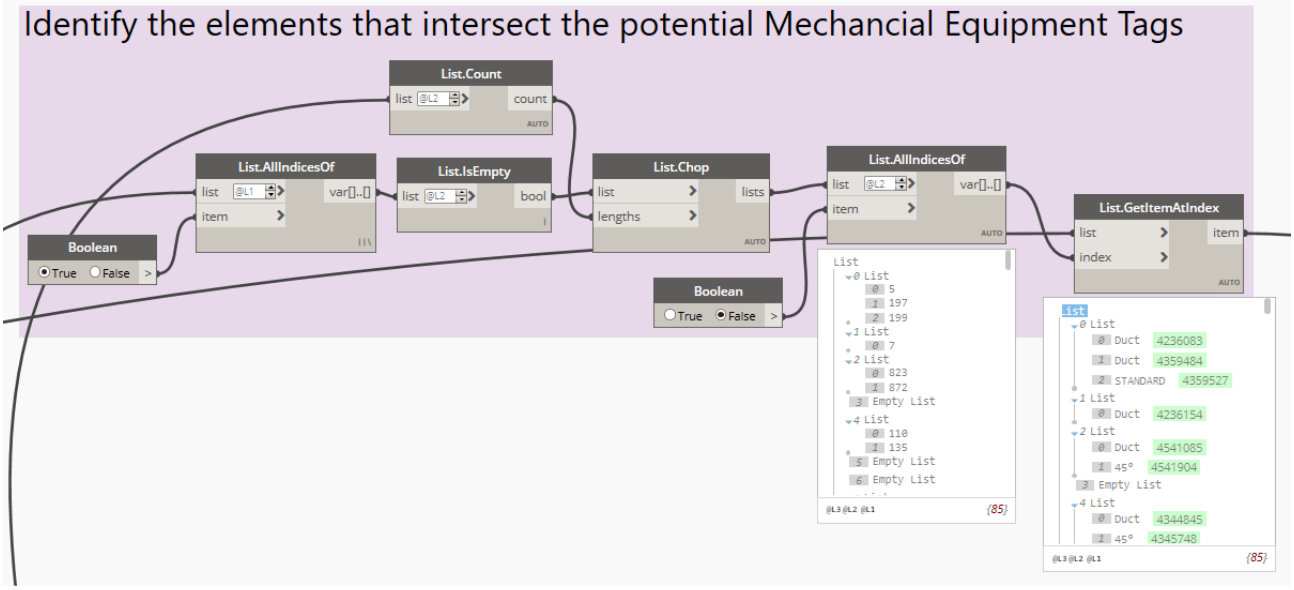
Separate the North, East, West and South tag placement lists into individual lists. Then, combine these lists back to contain sub lists at the appropriate level and reduce the length of the script. These lists will be complied in the preferred order: North, East, West and then South Placement Locations.



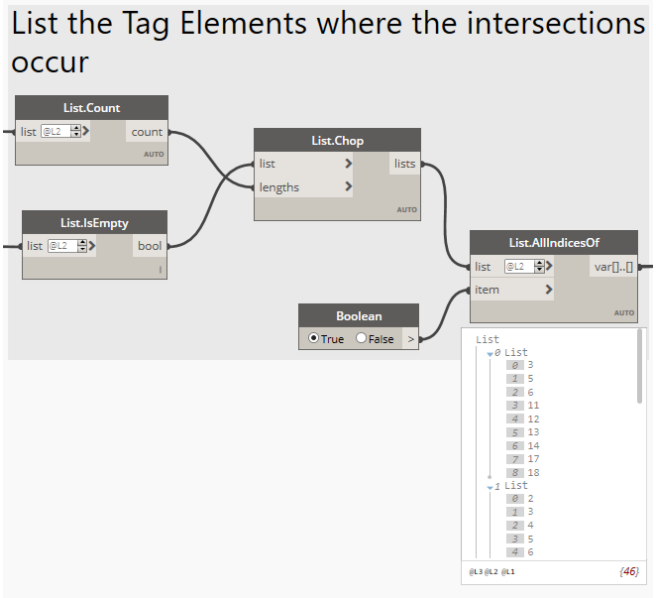
Obtain the intersections for the Bounding Boxes of the visible elements on the floor plan and the potential **Mechanical Equipment Tags**.



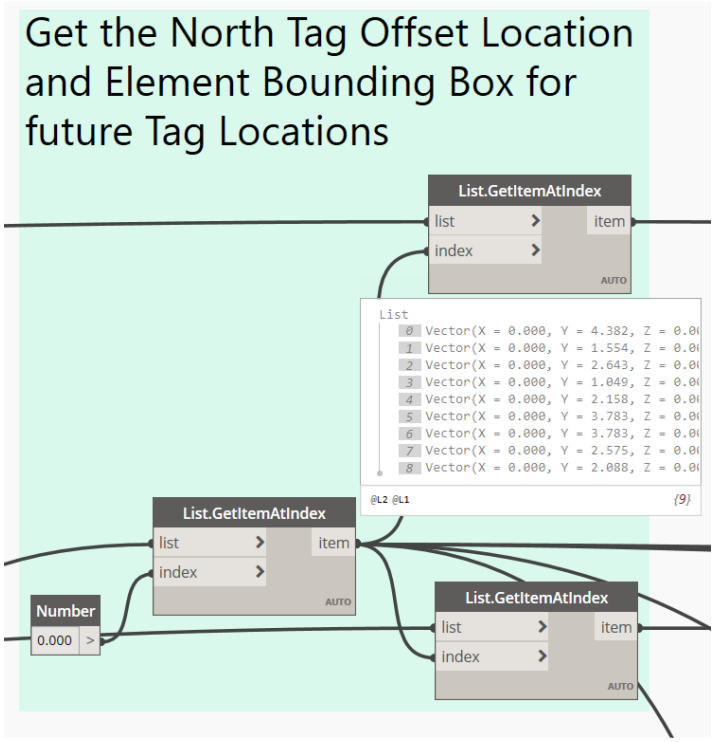
Identify the elements that intersect with the potential Mechanical Equipment Tag location.



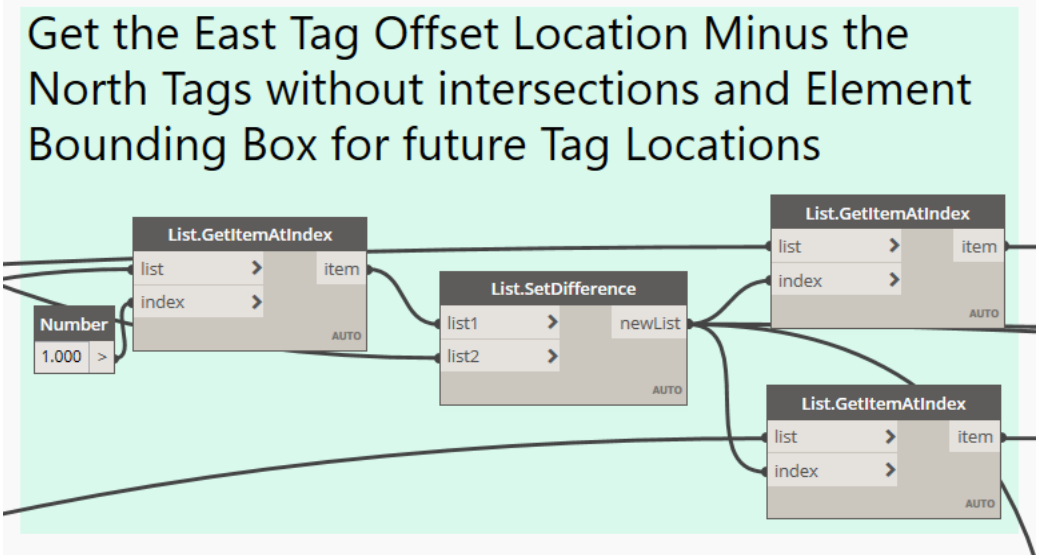
The list the Tag Element List Indices where these intersections occur for each placement.



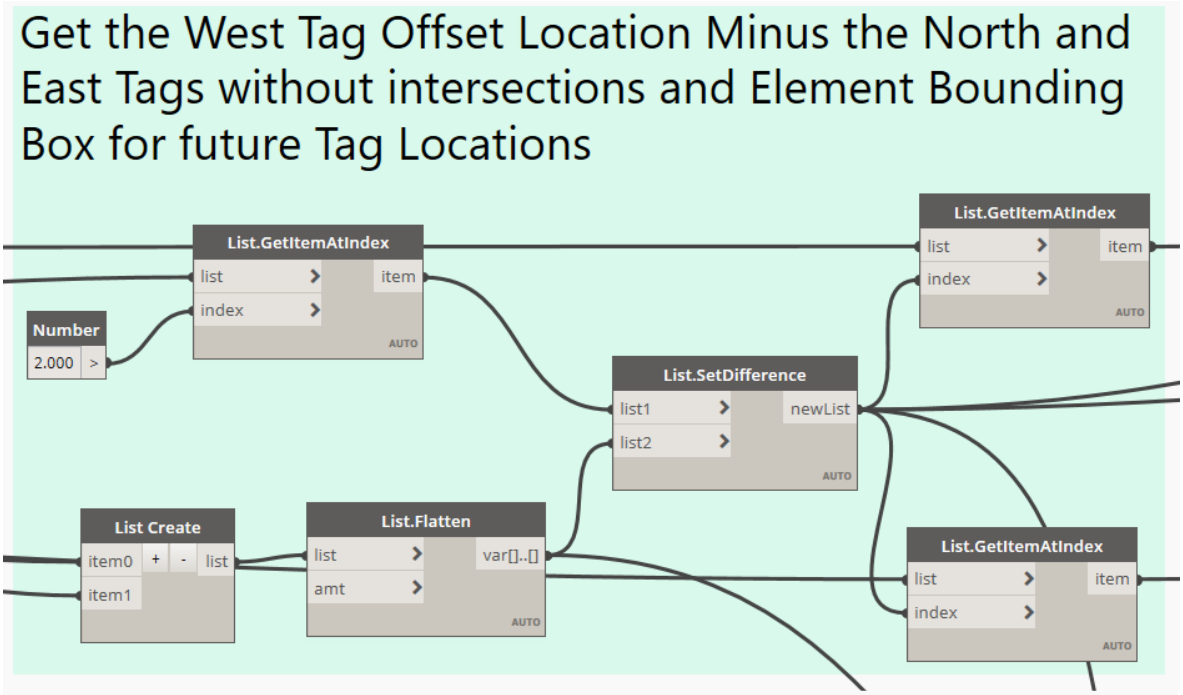
Get the North Tag offset location and Element Bounding Boxes for the future Tag Locations. These offset locations will be added to the list to **Tag** the **Mechanical Equipment** and the Element Bounding Boxes will be stored for additional element locations for future tags of different **Categories**.

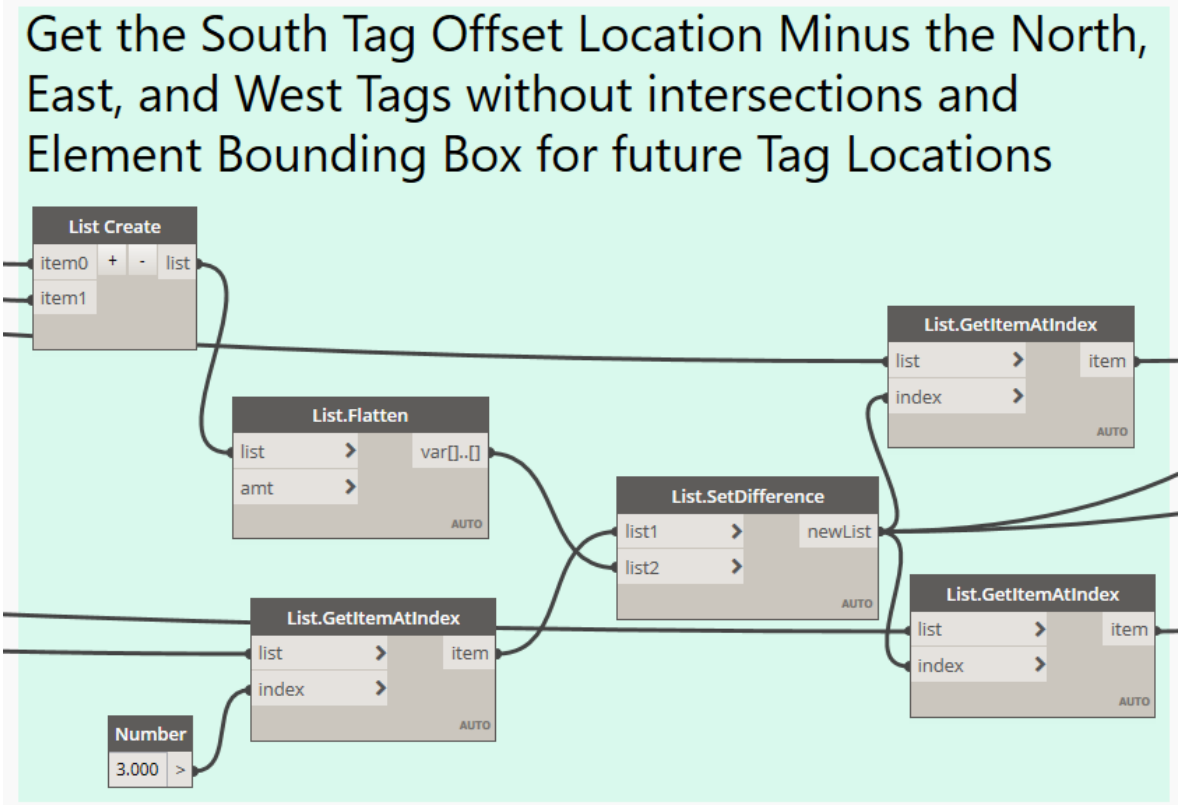


Get the East Tag offset location and Element Bounding Boxes for the future Tag Locations, but first subtract the Tags that have already been placed at the North Location.

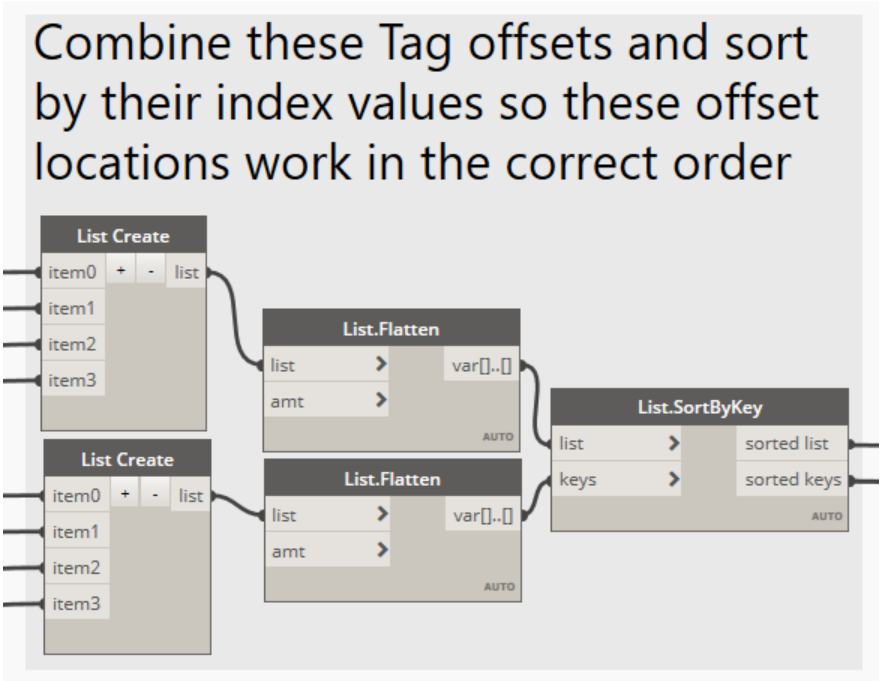


Repeat this process for the West and the South Locations. Again, subtracting the previously placed tags.

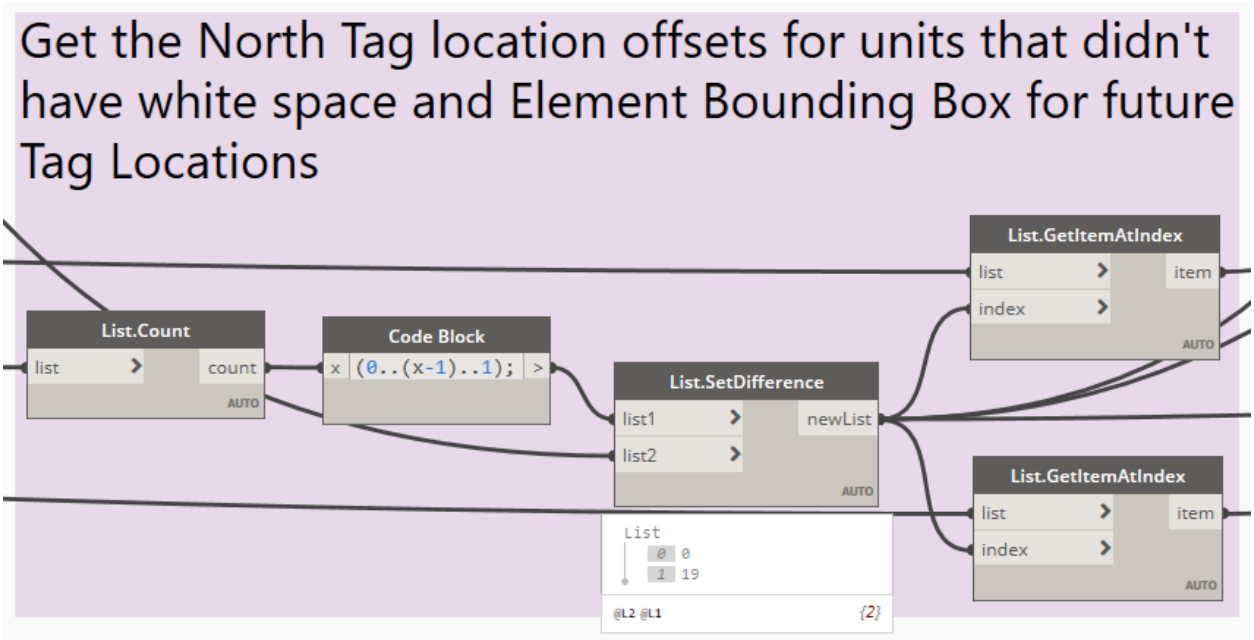




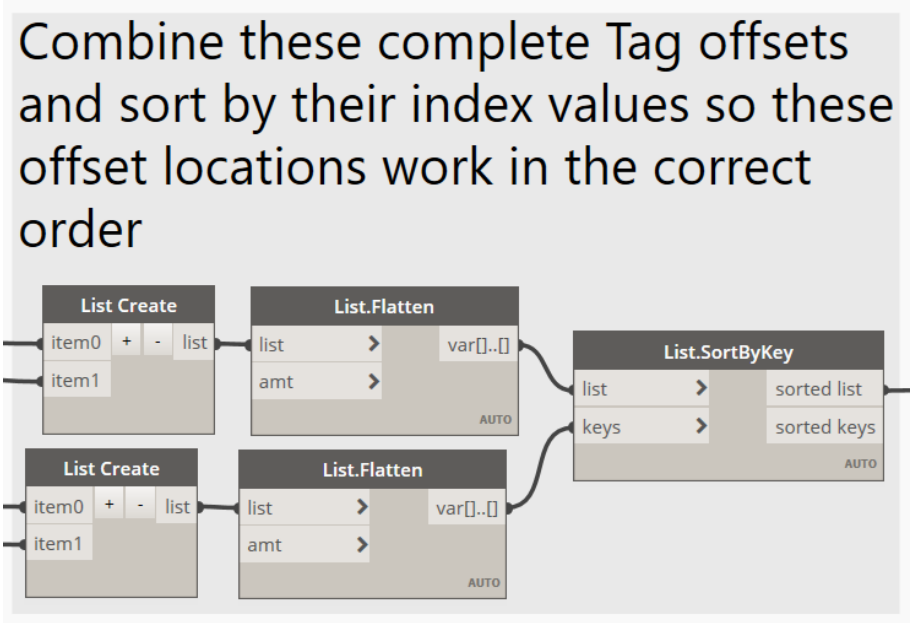
Combine these compiled lists and sort them by their indices.



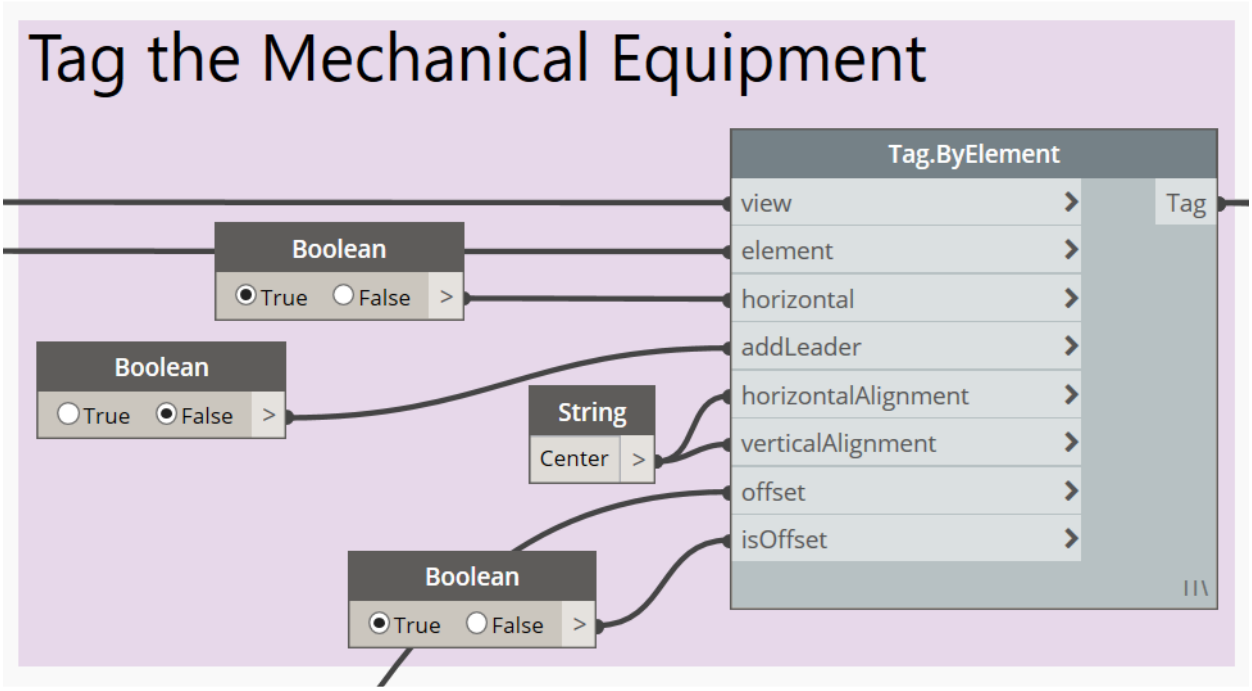
Select the North Tag location offsets and Bounding Boxes that did not have white space in any of the potential locations.



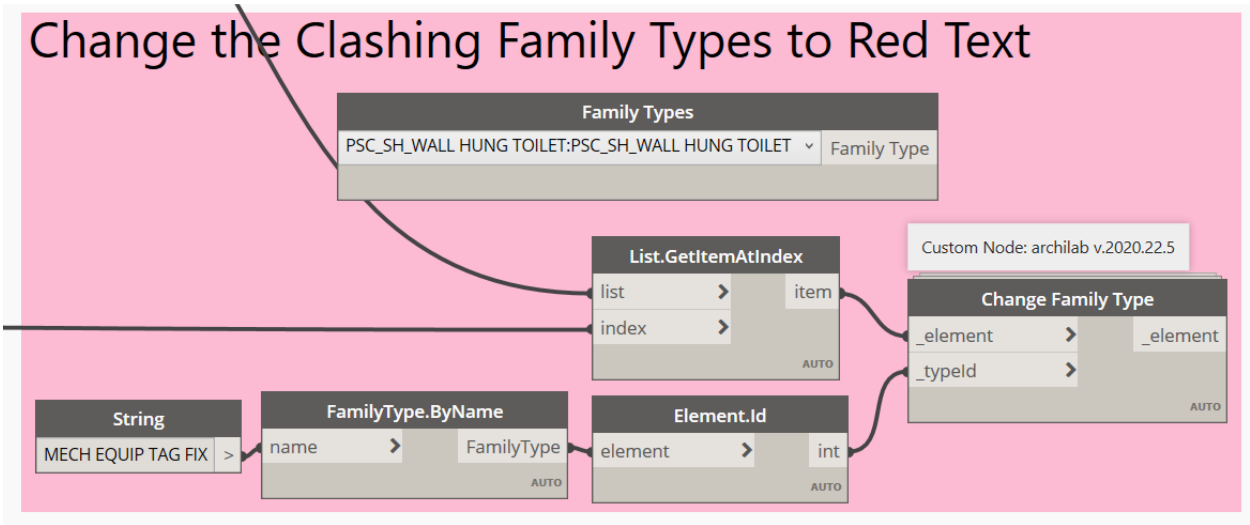
Combine these complete lists of **Tag** offsets and sort by their indices.



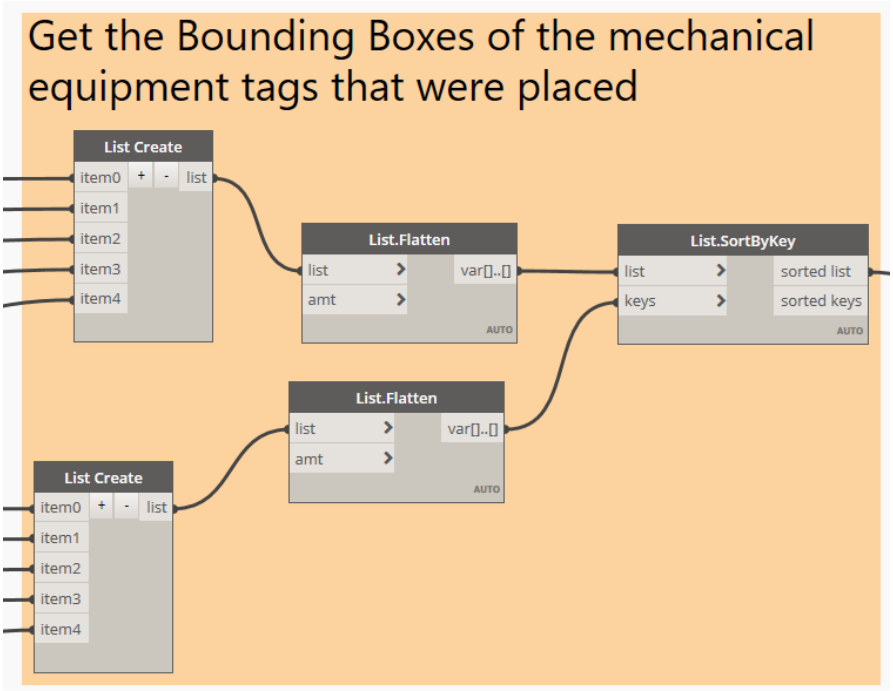
Finally, tag the Mechanical Equipment with their correct offset value location.



Change the **Tag Family Type** for the **Tags** that could not find an adequate white space so that these are clearly defined to Red text.

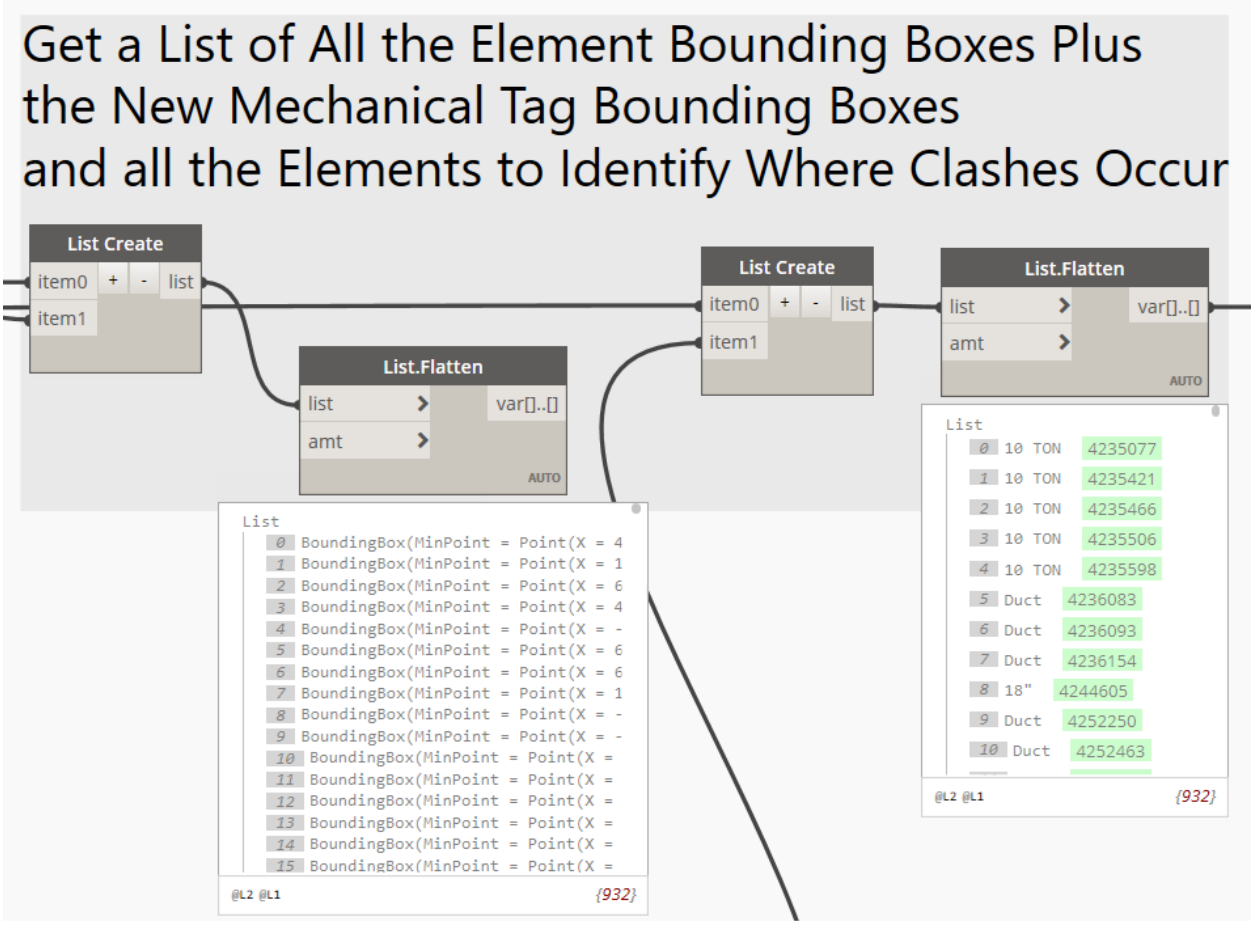


Now obtain the Bounding Boxes for the **Mechanical Equipment Tags** that were placed on the view.



Note: This step is unnecessary should you not plan to include any other tagging categories in your script.

Obtain a now complete list of all the Elements that contain Bounding Boxes on the view and their Bounding Box location.



Useful References

Boehning, J. (2017). Dynamo Design Script for MEP. *Autodesk University*: Las Vegas, NV. https://www.autodesk.com/autodesk-university/class/Dynamo-DesignScript-MEP-2017

Diekmann, A. (2017). Master Graphs. *DynamoAutomation.* Retrieved from https://github.com/andydandy74/DynamoAutomation/wiki/Master-Graphs

Massey, M. (2016). Schedules can't do that in Revit 2017. *Autodesk University*: Las Vegas, NV. https://www.autodesk.com/autodesk-university/class/Schedules-Cant-Do-Revit-2017-2016

Raja. (2017). Atkins14. Read Excel file node Opens Excel file. *Forum DynamoBIM.* Retrieved from https://forum.dynamobim.com/t/read-excel-file-node-opens-excel-file/7813