AUTODESK. UNIVERSITY

FAB466294 - Anybody Can Do It! Easily Build Revit Content in Inventor

Pete Strycharske

Implementation Consultant | @petestrycharske

Pete Strycharske
D3 Technologies

Implementation Consultant

- Autodesk Platinum Partner
- Teach classes on Inventor, AutoCAD, Factory Design Utilities and Navisworks
- Provide technical support
- Consult on design workflows and customer content generation
- Love working with kids, serving in church, the beach and playing basketball!

Why is this important?

Construction and Manufacturing Industries are Becoming more Connected and Seamless, so Should our Design Processes

- Modular construction practices are really growing rapidly and greatly improving construction efficiencies
- Companies are constantly looking for ways to better integrate the architectural design and manufacturing data
- Improving the quality and easy-of-use of Inventor to Revit models will prevent Revit users from cursing your name and increase the likelihood of your designs being utilized in building projects

VIEW SLIDESHOW
9 photos

Image: Minneapolis/St. Paul Business Journal

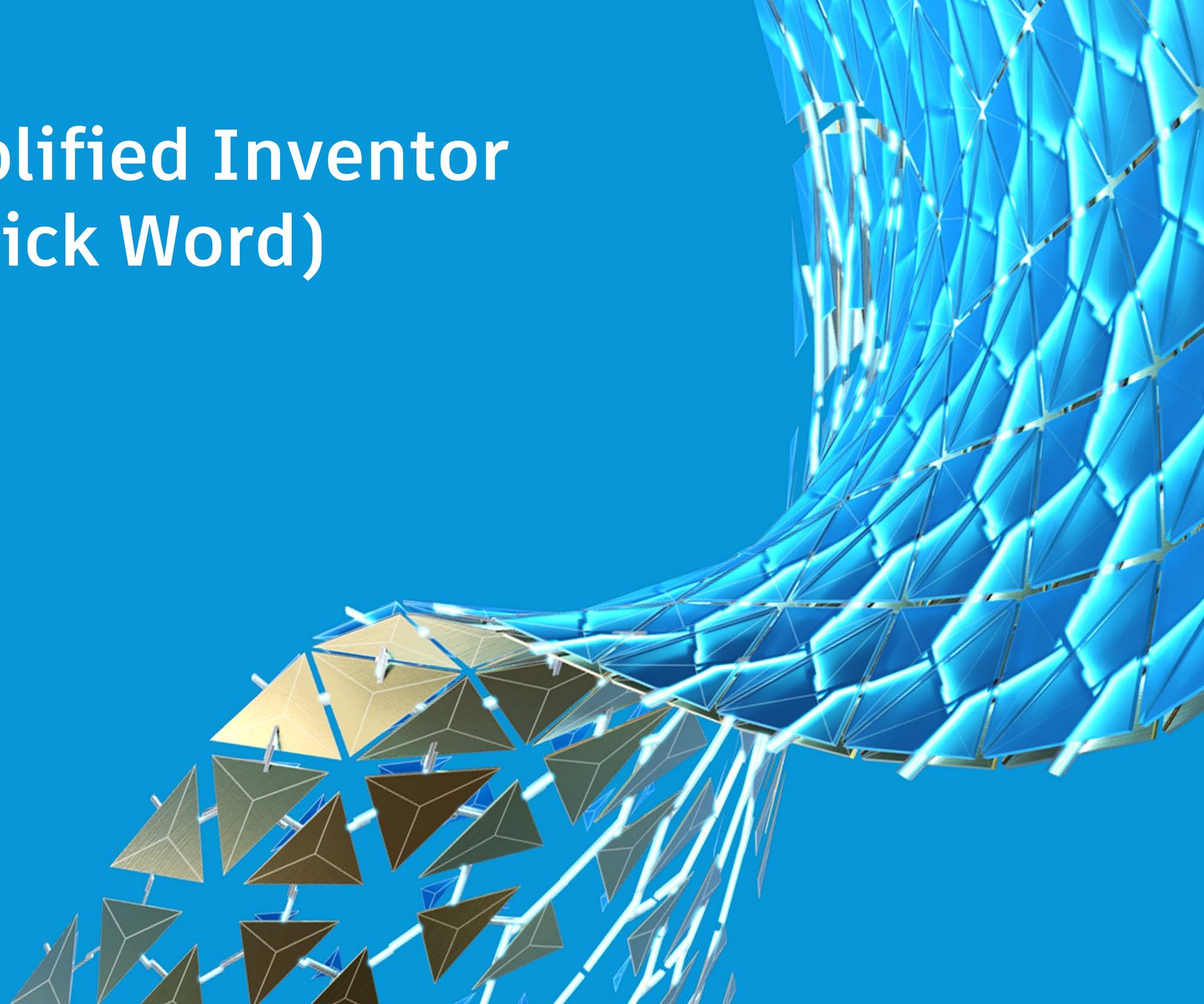
By Nick Halter – Senior Reporter/Broadcaster, Minneapolis / St. Paul Business Journal
Sep 20, 2020, 9:44am EDT

How Do We Make This Happen?

The workflow I'm presenting will utilize models generated withing Inventor that will be simplified, augmented with unique information and converted into Revit-ready content

- (Highly Recommended) Prepare the Inventor models by simplifying them as much as possible
- Apply unique BIM Connector appearances to desired model faces and surfaces
- Utilize a component approach to expedite the assembly building process
- Shrinkwrap the configured design to enhance Revit performance
- Unleash the power of iLogic to add BIM Connectors and content

Prepare Simplified Inventor Models (a Quick Word)



Why Simplified Inventor Models?

BETTER PERFORMANCE

Reducing the complexity of the Inventor models will improve modeling performance inside of Inventor and ultimately / more importantly inside of Revit

EASE OF USE

Simplified models are easier to control parametrically and allow users more ability to get to the desired configuration

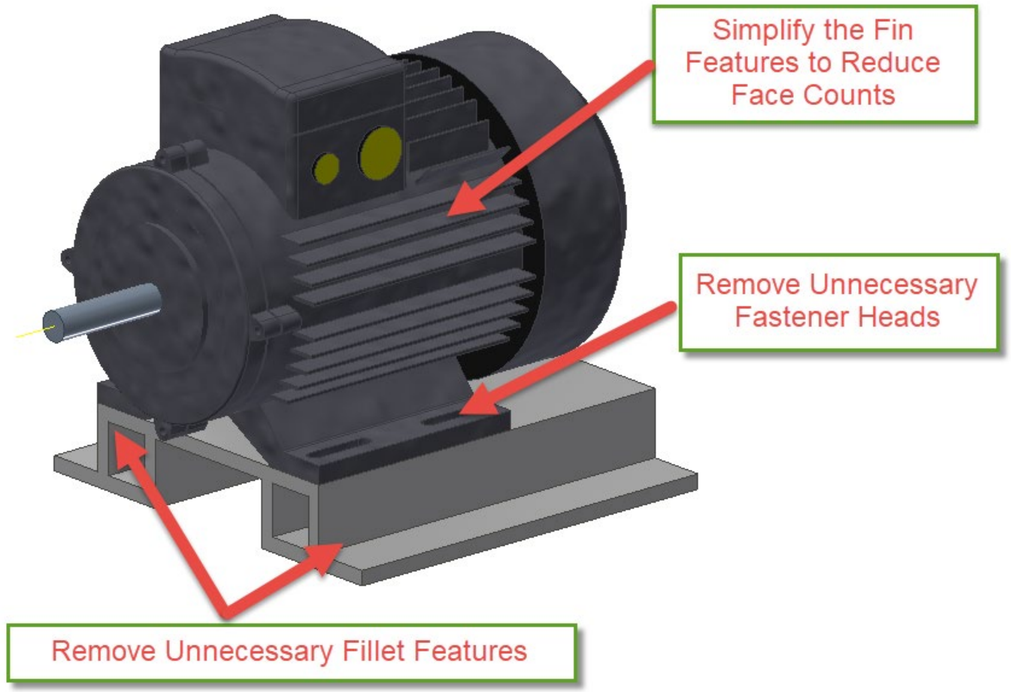
SIMPLIFIED MODELS CAN BE FLESHED OUT LATER

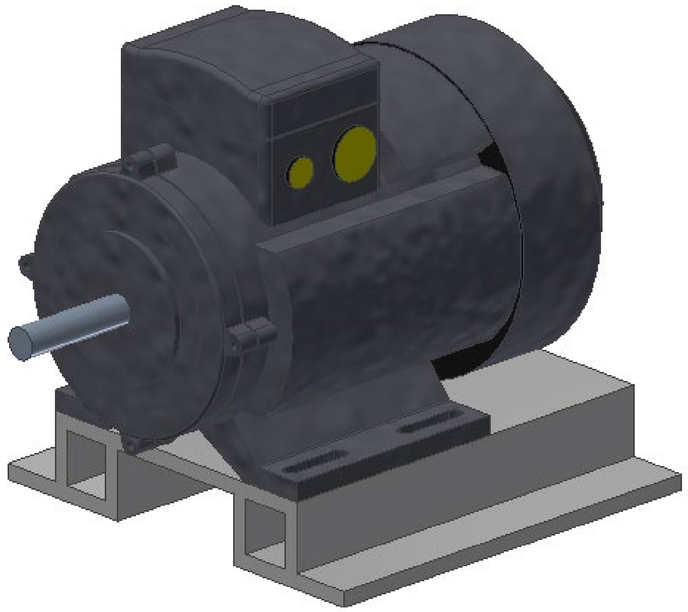
We don’t want to waste time configuring fully engineered models at the sales or concept stage and simplified models can be enhanced or converted when needed (Covered in the Component Approach Section)

Simplify to Improve Performance

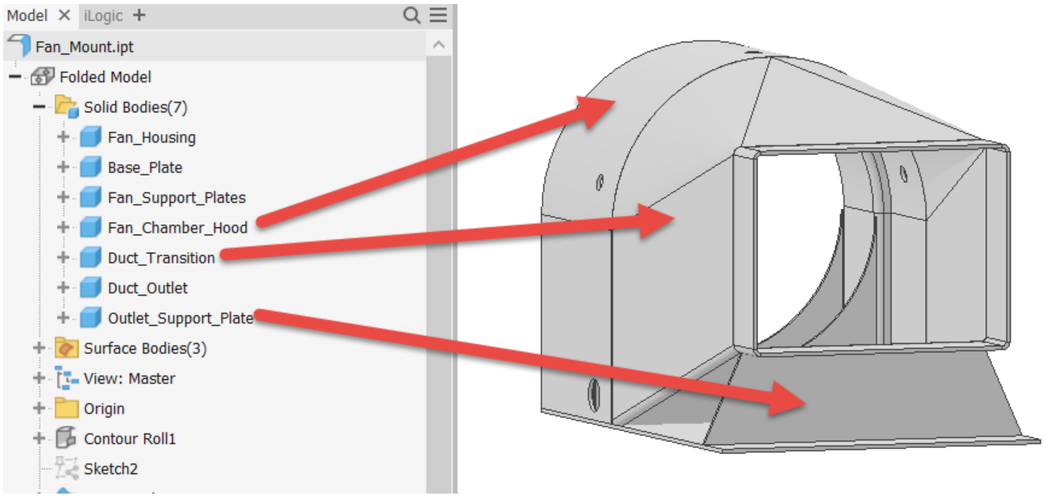
Reducing the amount and complexity of features can be a great way to achieve better performance

- **Remove unnecessary features**
  - Unless a bolt head serves a very specific purpose in Revit, remove them all
  - Fillet features and fine details are generally inappropriate for large models
- **Simplify complex features, such as the fins shown at the right**



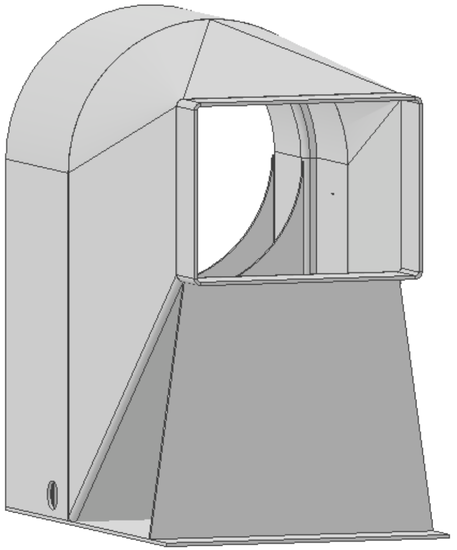


Simplify to Improve Ease of Use



Creating essential, multi-body models allows for better modeling and parametric controls

- Simple models are easier to construct to convey conceptual designs
- Parameters can help to easily shape and resize multi-body designs



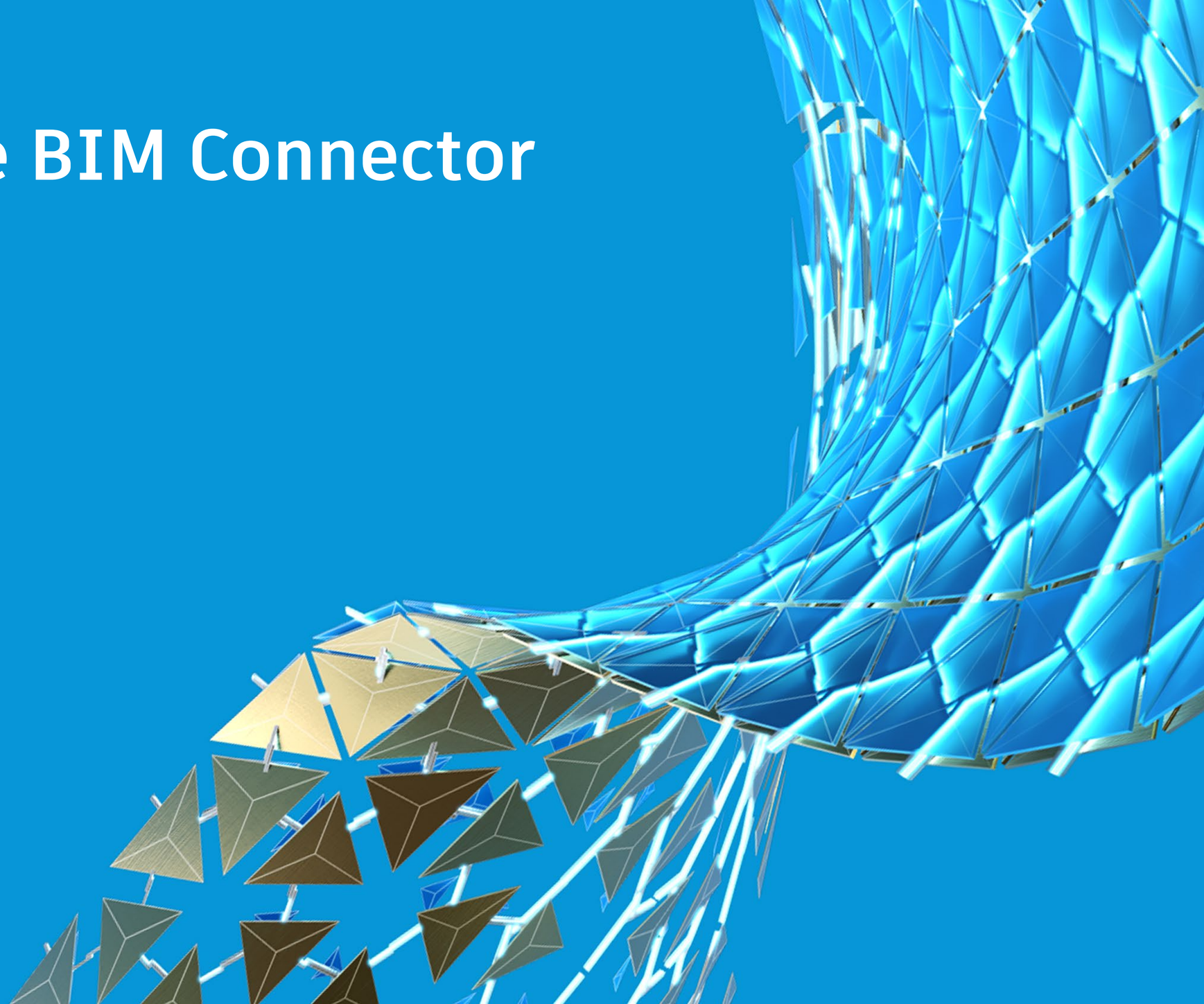
Parameters

Parameter Name	Consumed by	Unit/Type	Equation
Model Parameters			
Reference Parameters			
User Parameters			
Fan_OD	d2	in	10 in
Fan_Inner_ID	d1	in	9.03125 in
Fan_Inner_Width	Fan_Chamber_Length, d0	in	3.1 in
Fan_Height_Position	d23, d13, d3	in	6 in
Base_Plate_Length	d29	in	16 in
Fan_Chamber_Length	d40	in	Fan_Inner_Width * 2 ul
Water_Inlet_Angle	d71, d58	deg	75 deg
Outlet_Duct_Width	d82, d74	in	8 in
Outlet_Duct_Height	d75, d83	in	6 in

Parameters

Parameter Name	Consumed by	Unit/Type	Equation	Nomi
Model Parameters				
Reference Parameters				
User Parameters				
Fan_OD	d2	in	10 in	10.00
Fan_Inner_ID	d1	in	9.03125 in	9.031
Fan_Inner_Width	Fan_Chamber_Length, d0	in	3.1 in	3.100
Fan_Height_Position	d23, d13, d3	in	12 in	12.00
Base_Plate_Length	d29	in	16 in	16.00
Fan_Chamber_Length	d40	in	Fan_Inner_Width * 2 ul	6.200
Water_Inlet_Angle	d71, d58	deg	75 deg	75.00
Outlet_Duct_Width	d82, d74	in	8 in	8.000
Outlet_Duct_Height	d75, d83	in	6 in	6.000

Apply Unique BIM Connector Appearances



How do BIM Connector Appearances Work?

CREATE UNIQUE BIM APPEARANCES

Custom appearances help to differentiate faces and surfaces that will eventually receive BIM Connectors from the rest of the model surfaces. A custom appearance library dramatically improves this process

BUILD SURFACES FOR OPEN CAVITY COMPONENTS

For many models, particularly if solid body faces are not available, surfaces can be created to serve as the basis for BIM Connectors

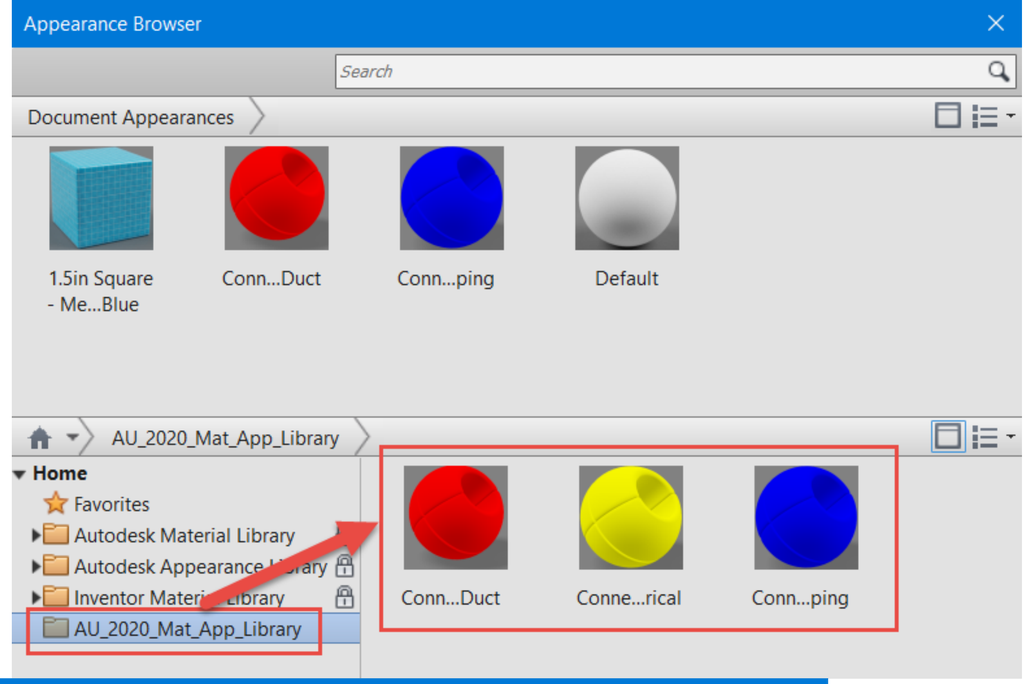
APPLY BIM APPEARANCES TO FACES AND SURFACES

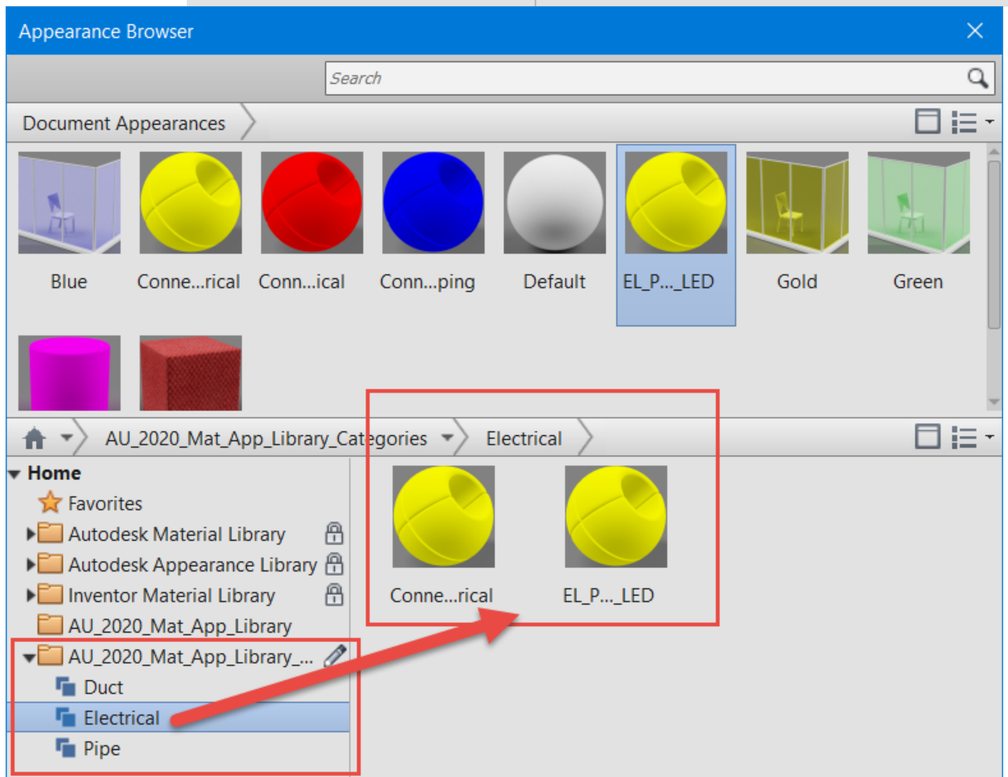
Once the unique BIM Appearances have been created, apply the appearances to the desired faces and surfaces on the simplified models

Create BIM Appearances

The unique appearances that are created will be utilized later in the process to apply specific types of BIM Connectors

- **Appearance names must be CONSISTENT!!!**
  - iLogic will be driving the creation of the BIM Connectors, so they must be IDENTICAL across all components
- **Appearances can be color coded for ease of use**
  - Should give the end users understanding on how to position components
- **Appearances must be stored in a library**
  - This will aid with the consistency requirement and is also necessary for the Shrinkwrap process to produce components with the proper appearances

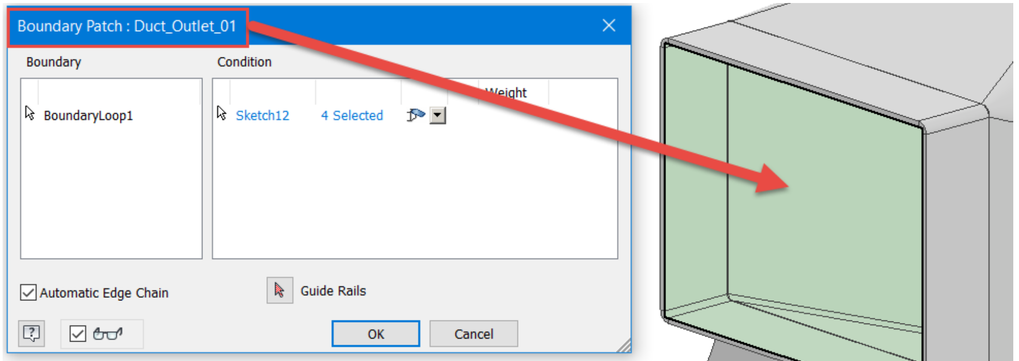


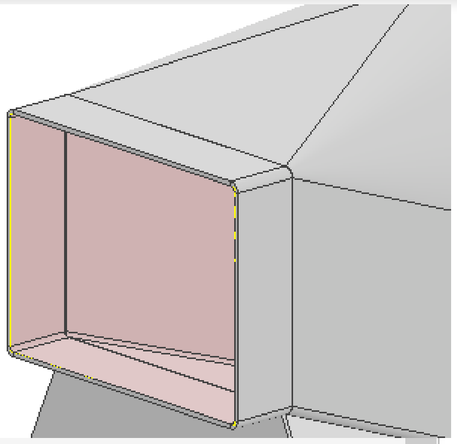


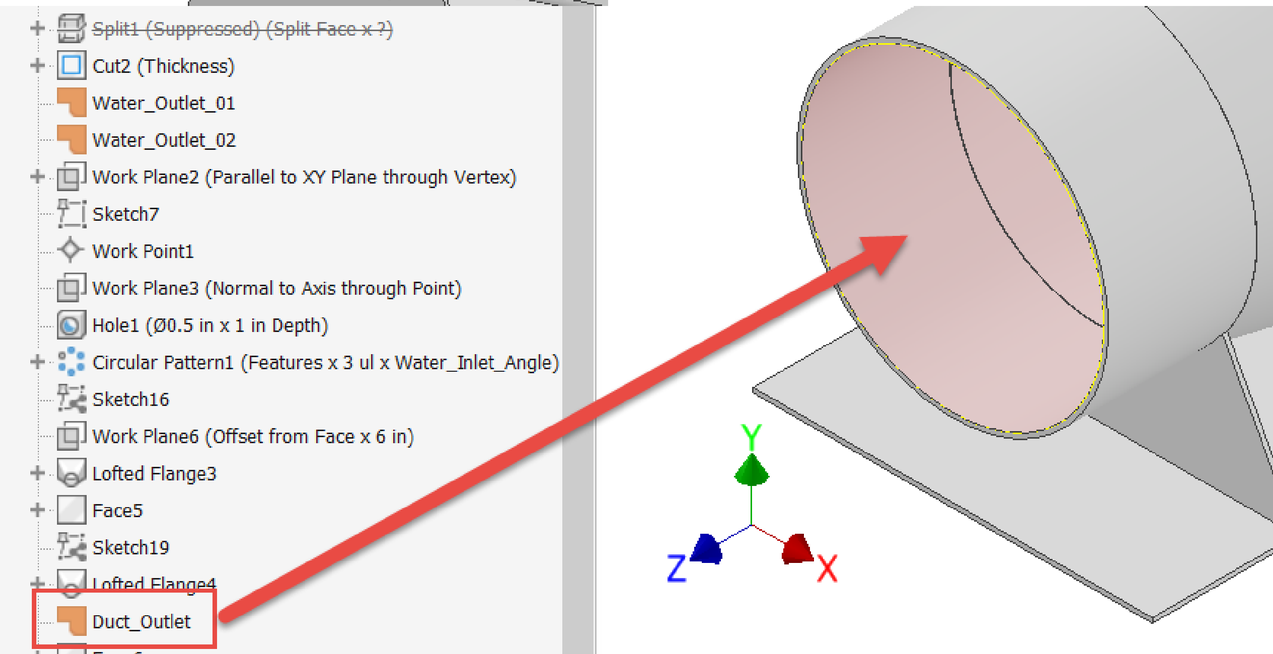
Build Surfaces as Needed

For some components, like ductwork, an open end of the duct is desired, so surfaces can be employed

- **Applying boundary patches is an easy way to create the surfaces**
  - Sketches or model edges can be used
  - Naming the boundary patches is a great way to stay organized



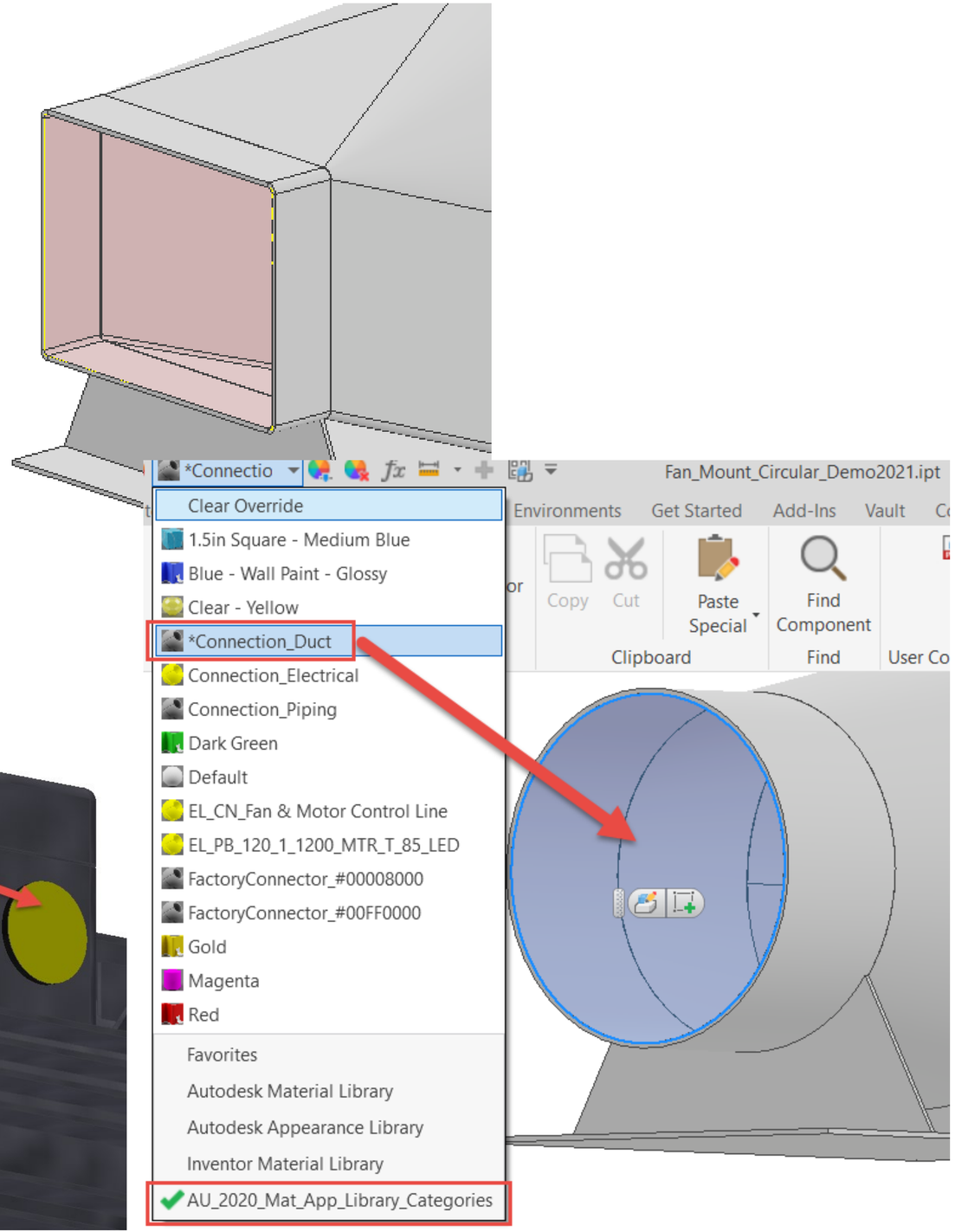


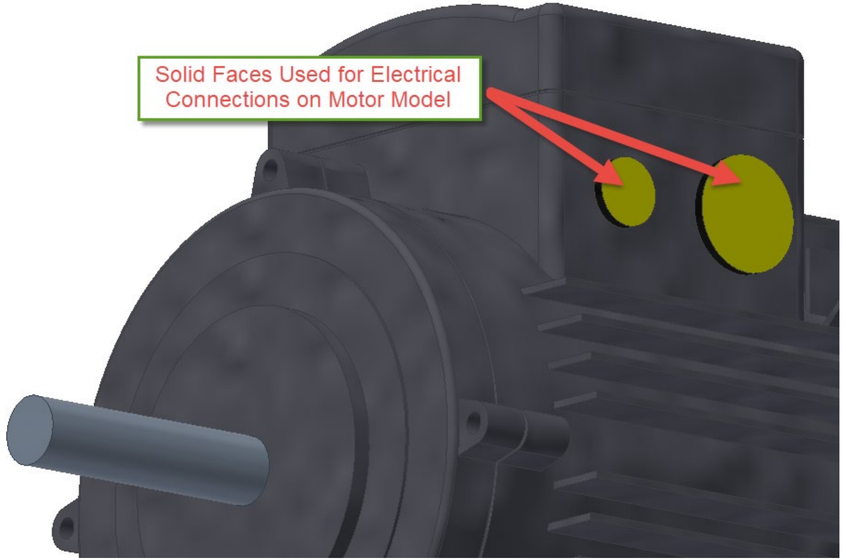


Apply BIM Appearances to Faces and Surfaces

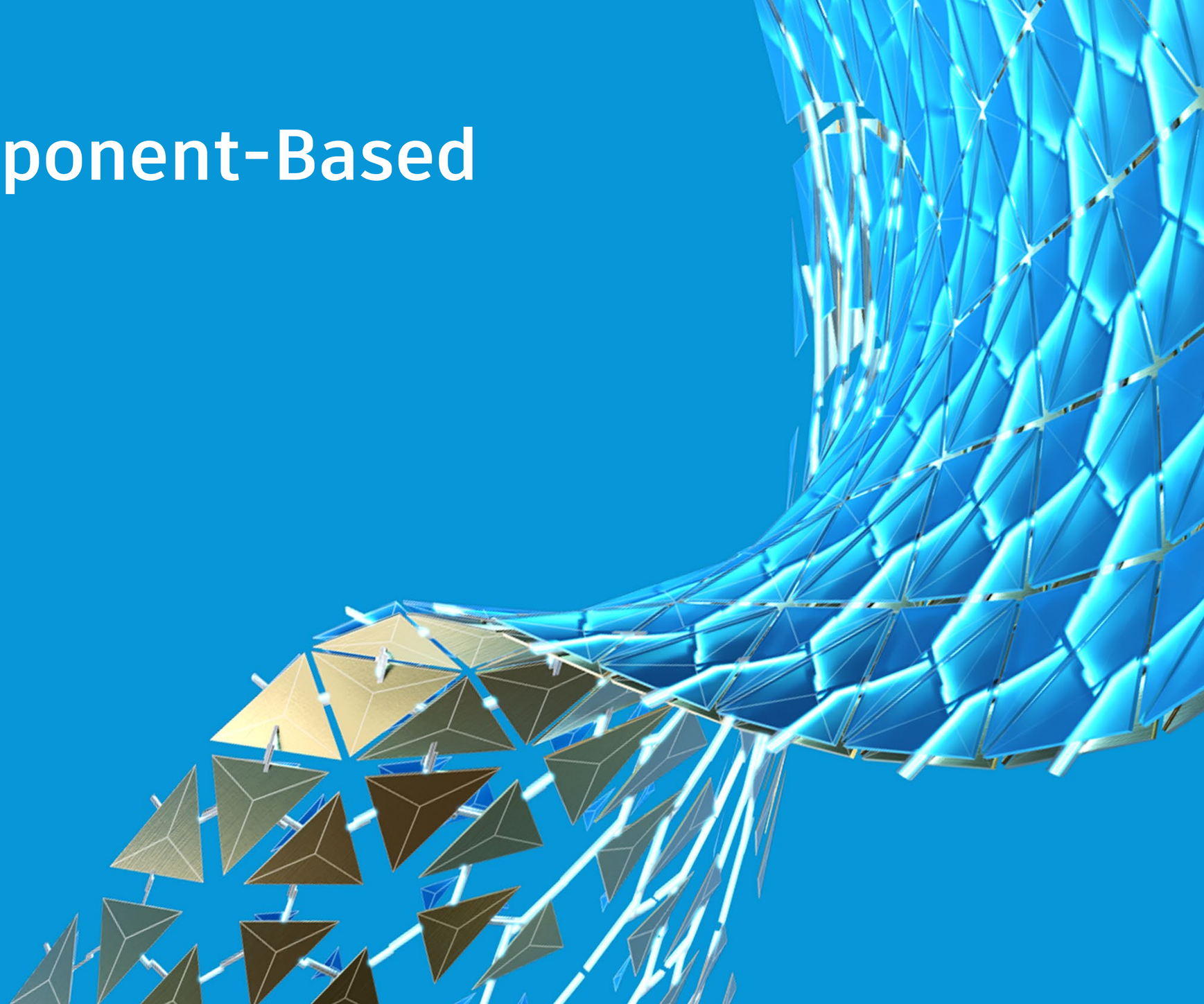
Assign the BIM appearances to each corresponding face or surface that will later be used to drive the creation of BIM Connectors

- Apply the desired BIM Connector appearances to the face or surface
- (Optional) Set the surface to be opaque for consistency with face applications





Utilize a Component-Based Approach



What is a Component-Based Approach?

UTILIZE SIMPLIFIED, PARAMETRICALLY DRIVEN MODELS

Covered earlier, these models will give the users flexibility to configure the desired designs

UTILIZE A MULTI-BODY APPROACH TO ENABLE FULL ENGINEERING LATER ON

Some Top-Down design techniques allow for a simplified, parametrically driven model during the conceptual and sales phases of a project and then help to drive the precise engineering required once the project enters the detailed design phase

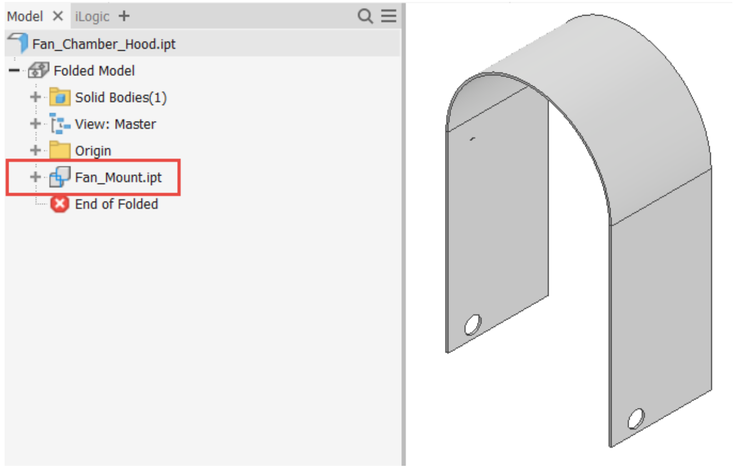
PRECONFIGURE COMPONENT CHARACTERISTICS

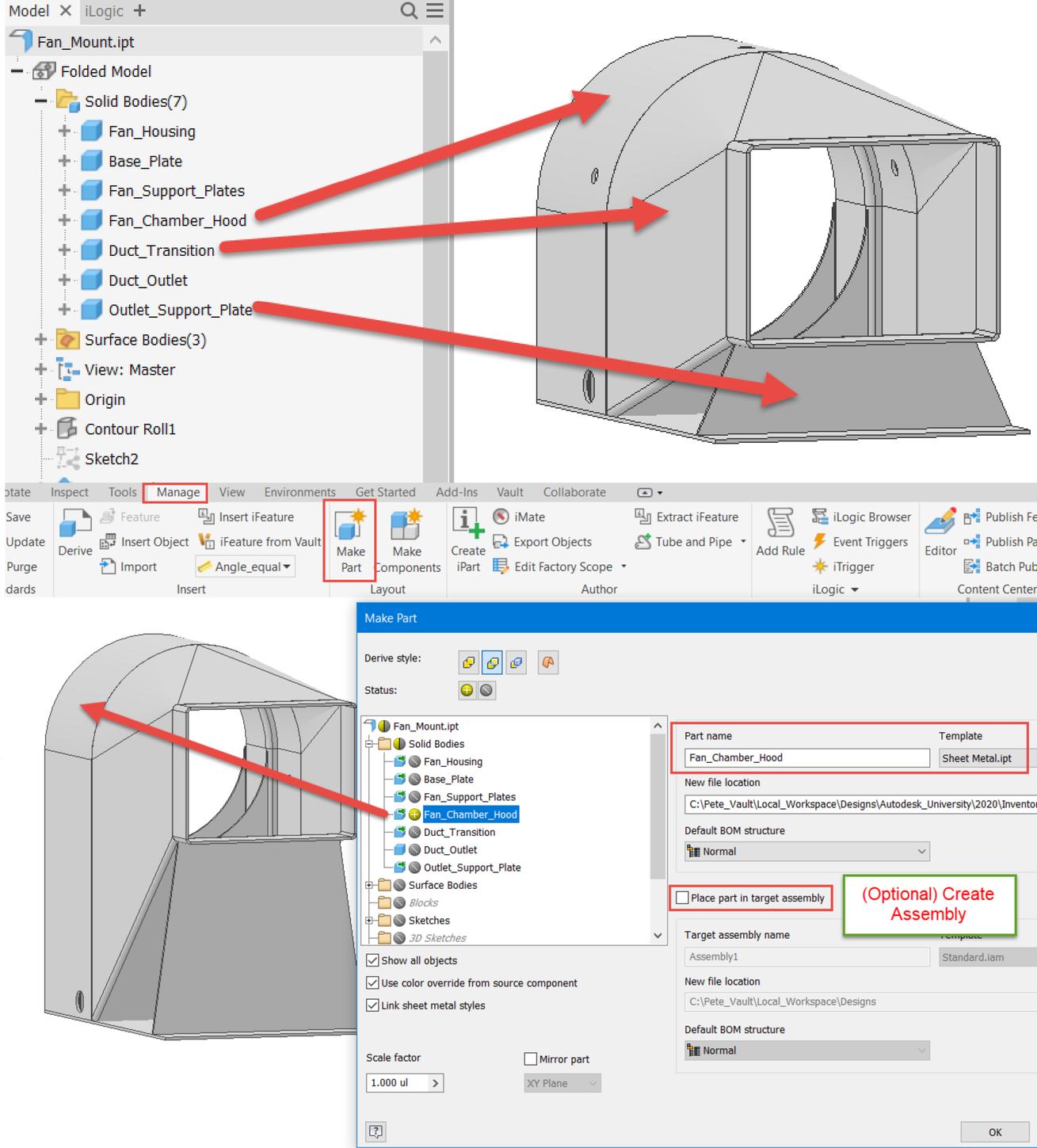
Apply any information or features, such as iMates, that will prove useful when configuring the design in the assembly environment

Utilize a Multi-Body Approach for the Model

Multi-Solid body part models can be used to simulate the final assembly and later converted into fully engineered versions, still tied to the central design file

- Build the model with each “part” as a unique solid body
- Convert these solid bodies to individual parts during the detailed design phase, while maintaining a unified design

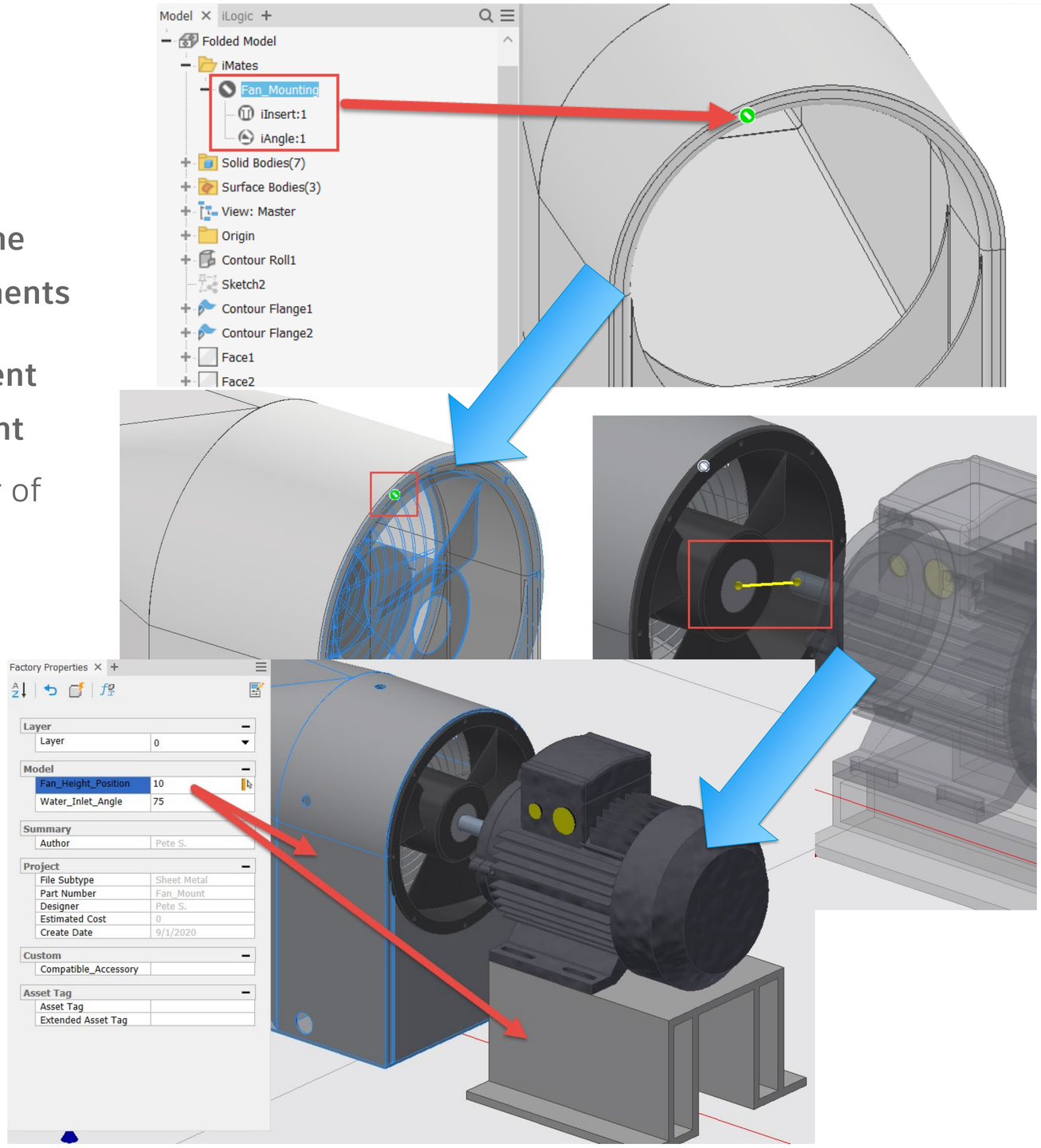




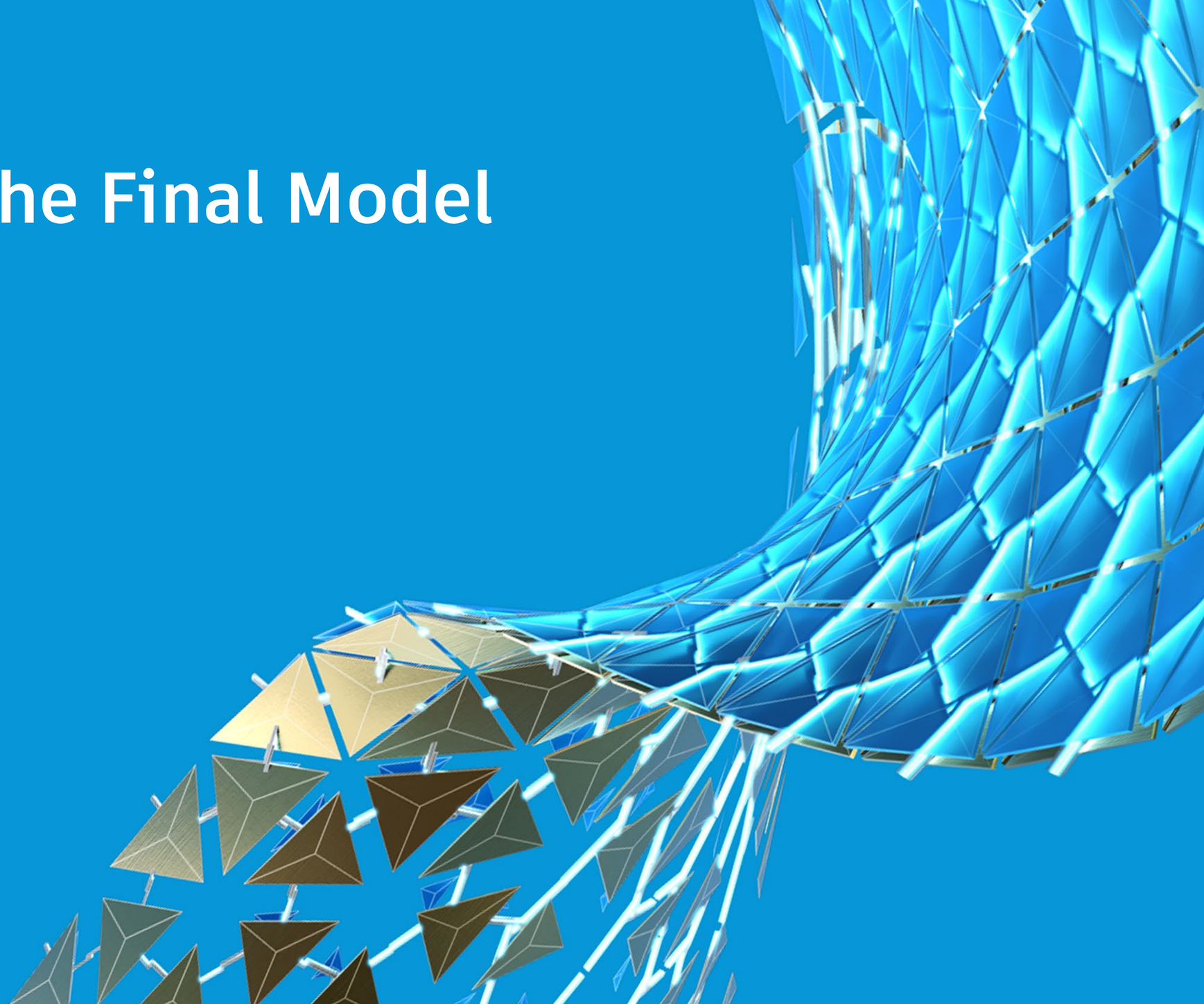
Preconfigure Component Characteristics

Any information or features that will be useful in the assembly environment can be added to the components

- **Utilize iMates to help quickly configure component designs “on-the-fly” in the assembly environment**
  - Ensure that naming conventions and the order of iMates is consistent across components
- **(Optional) Utilize the Factory Design Utilities to quickly configure components AND pass along parameter values between components**



Shrinkwrap the Final Model



What Does Shrinkwrap Bring to the Process?

SIMPLIFIED ASSEMBLY MODELS

Simplified part models greatly enhance the performance of the models and the same benefits can be extended to assemblies by easily removing unnecessary components and features. This improves performance for the final Revit model, but also for internal users as well

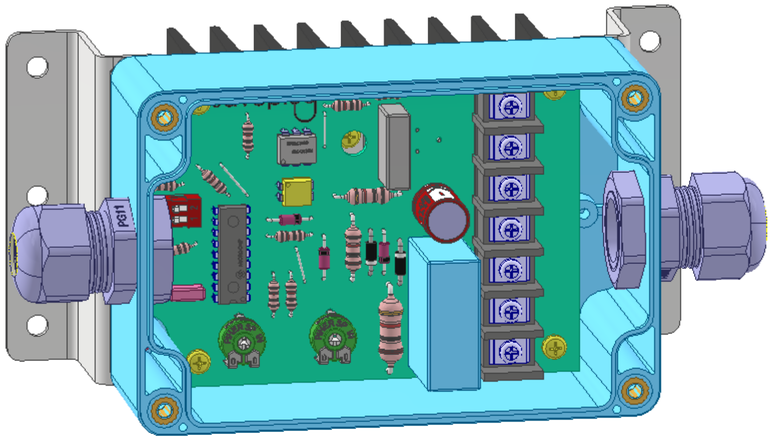
PROTECT INTELLECTUAL PROPERTY

Remove components that customers need or must not view and solid fill voids so end users are unaware of a component's inner workings

UNIQUE BIM APPEARANCES ARE PASSED THROUGH INTO THE SHRINKWRAPPED PART

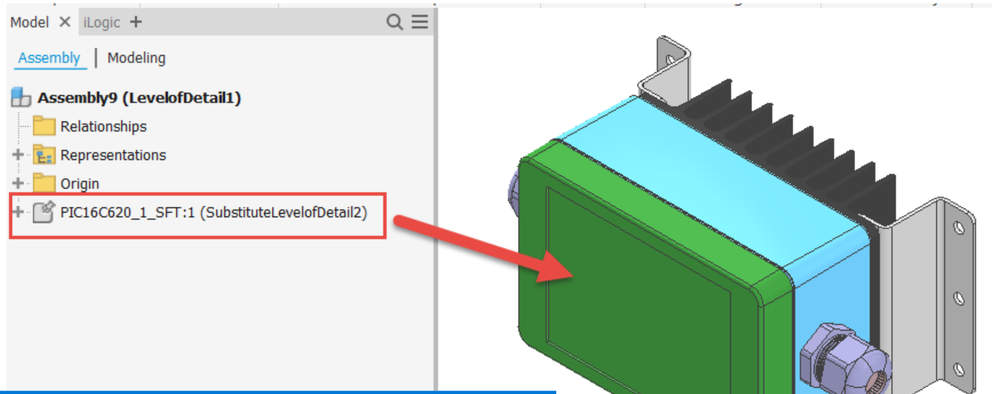
Any special face or surface appearances that we are utilizing to drive BIM Connectors can be passed along to the final Shrinkwrapped part, so long as the required custom appearance library is active

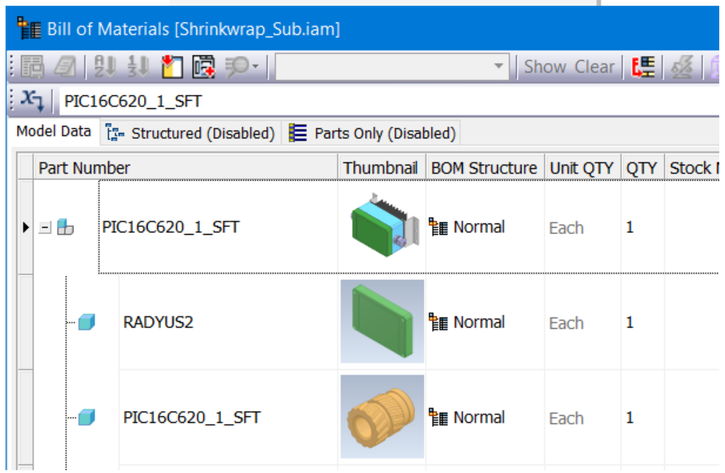
Shrinkwrap to Improve Performance



Reducing the number of components and features can improve the performance of the final Revit design as well as subassembly use

- Shrinkwrap removes unnecessary components and features, along with other settings, to form a streamlined design
- The full subassembly can be replaced by a representative part model, yet the BOM information is still present in the upper level assembly

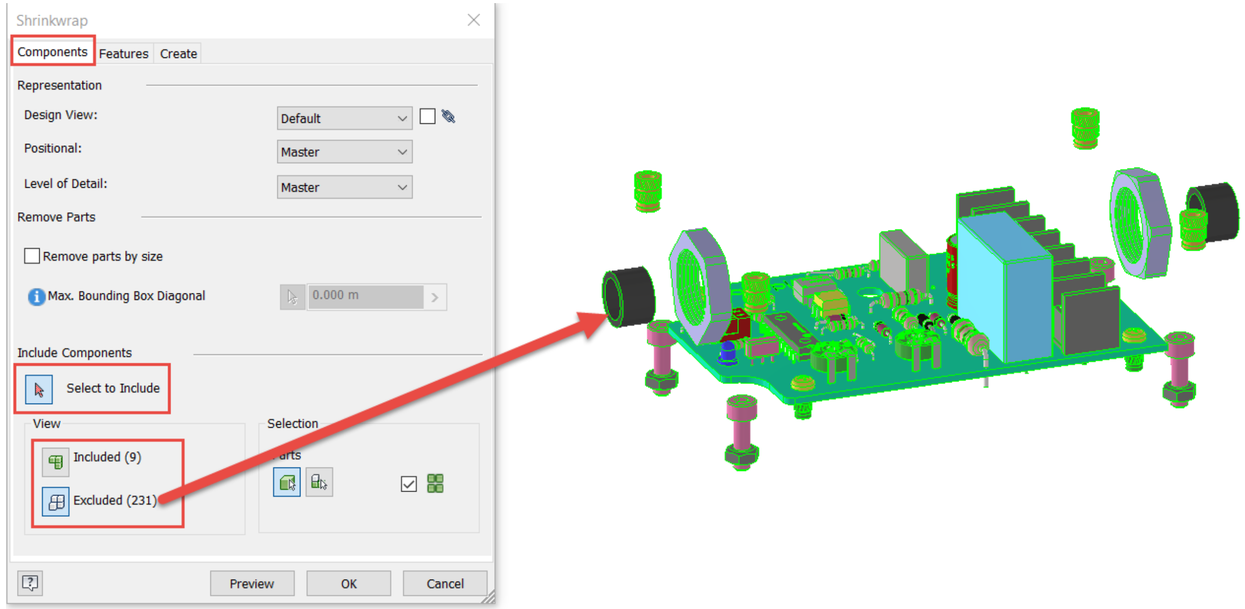


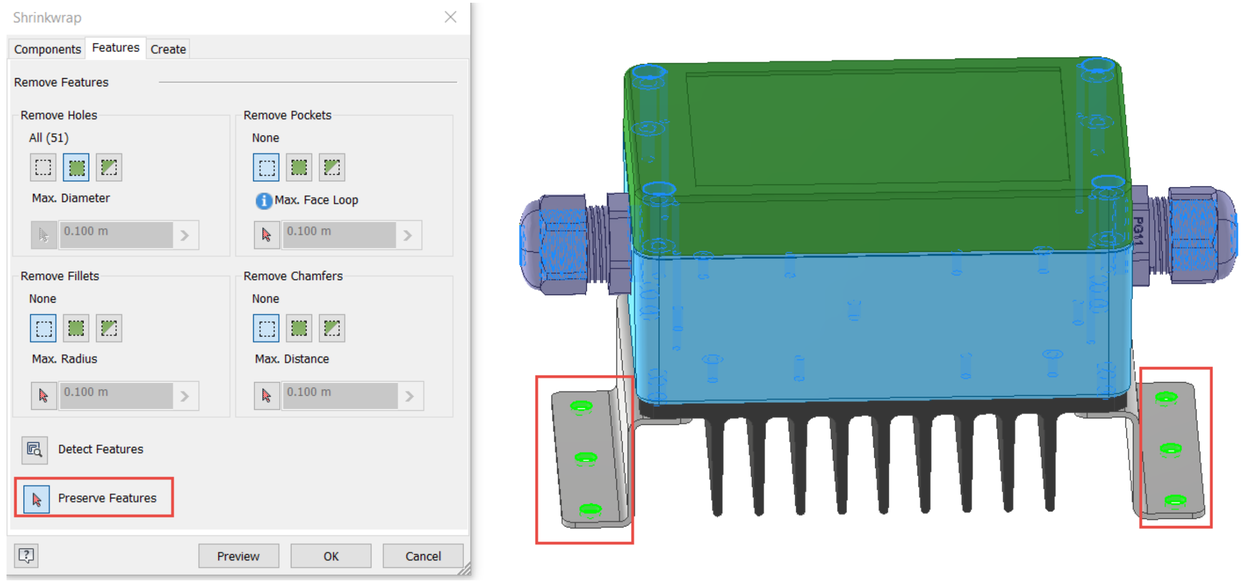


Remove Unnecessary Parts & Features

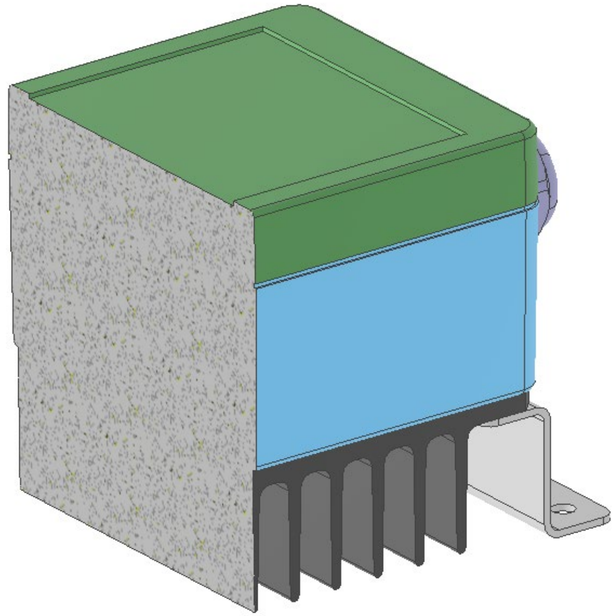
The Shrinkwrap tool makes removing components easy and efficient to do.

- **Select the parts that need to remain or if easier remove all the unnecessary parts**
  - There are tools and techniques that expedite this process
- **Remove unnecessary features**
  - Preserve any that are required



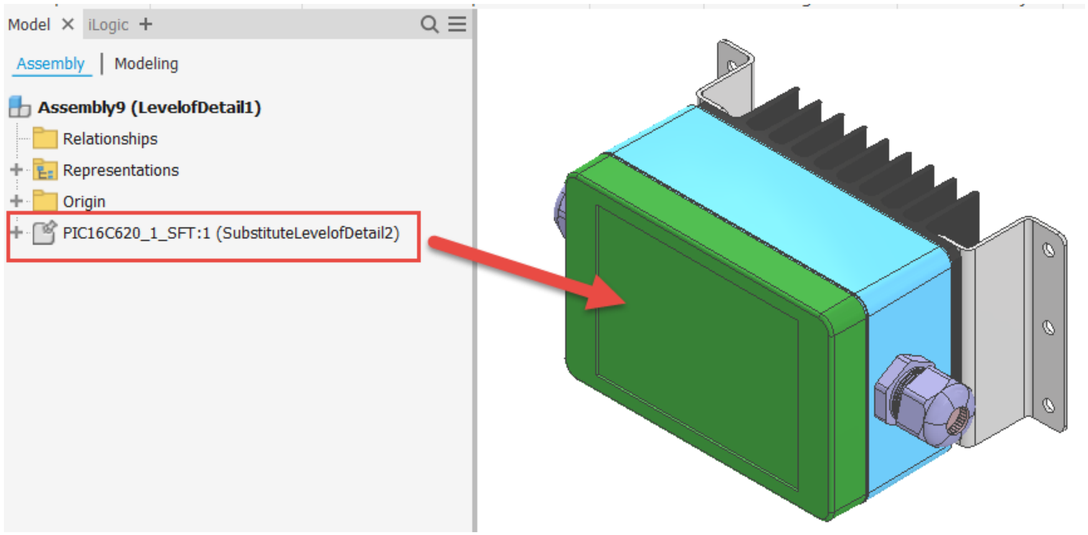


Fill Voids and Publish a Simplified Design

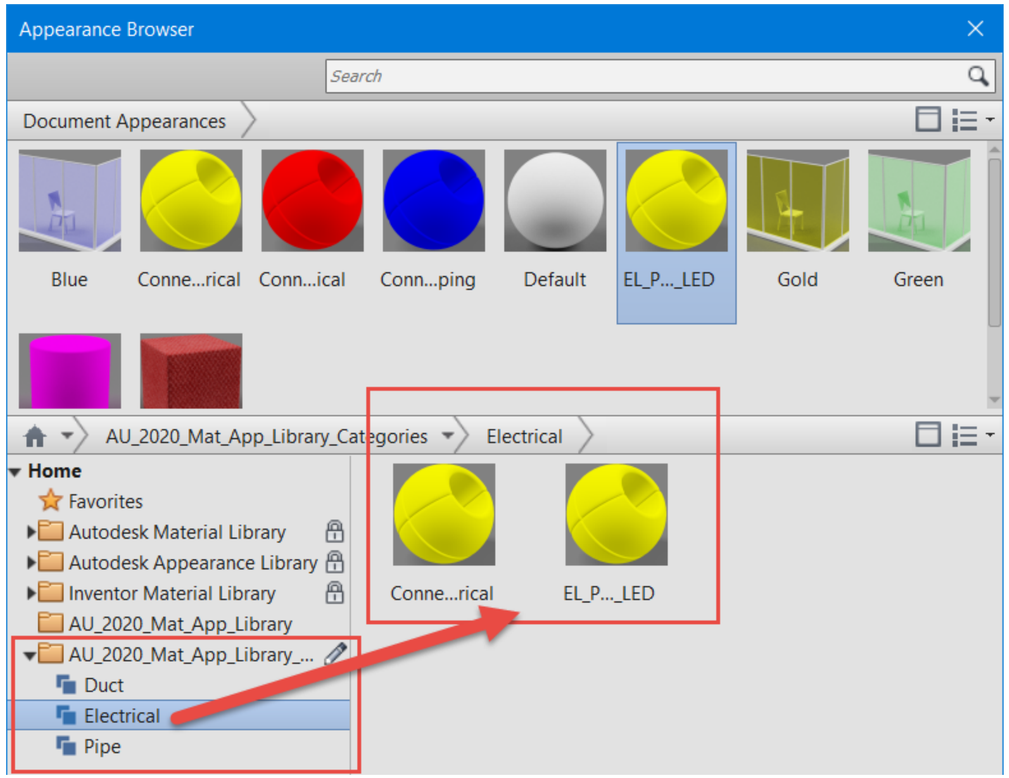


Remove internal voids and fully protect intellectual property when sending models to external contacts

- Remove any further internal components and fill internal voids
- A simplified part model representing the assembly is published for use internally and externally

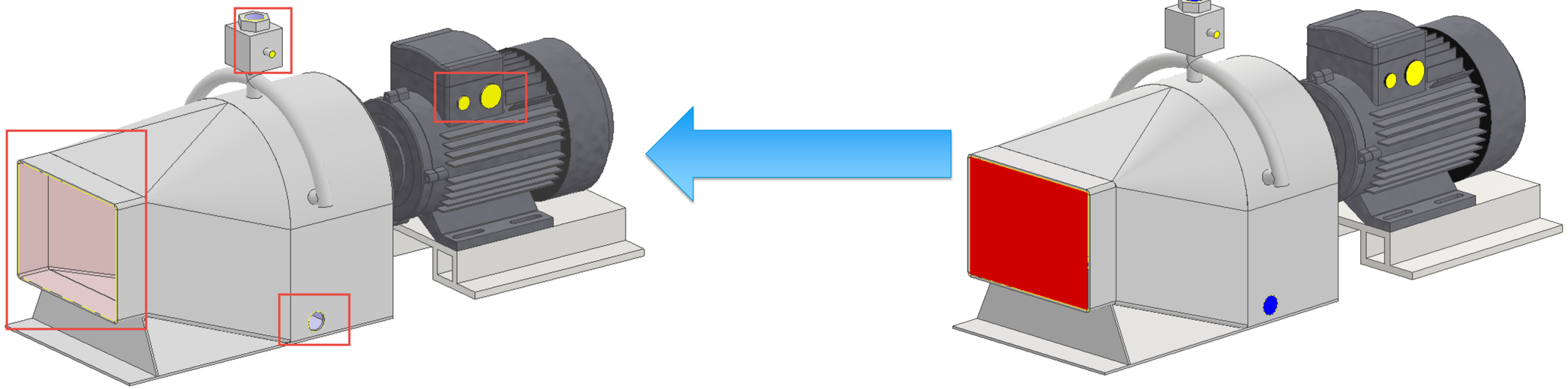


BIM Appearances Pass Into the Shrinkwrap Part

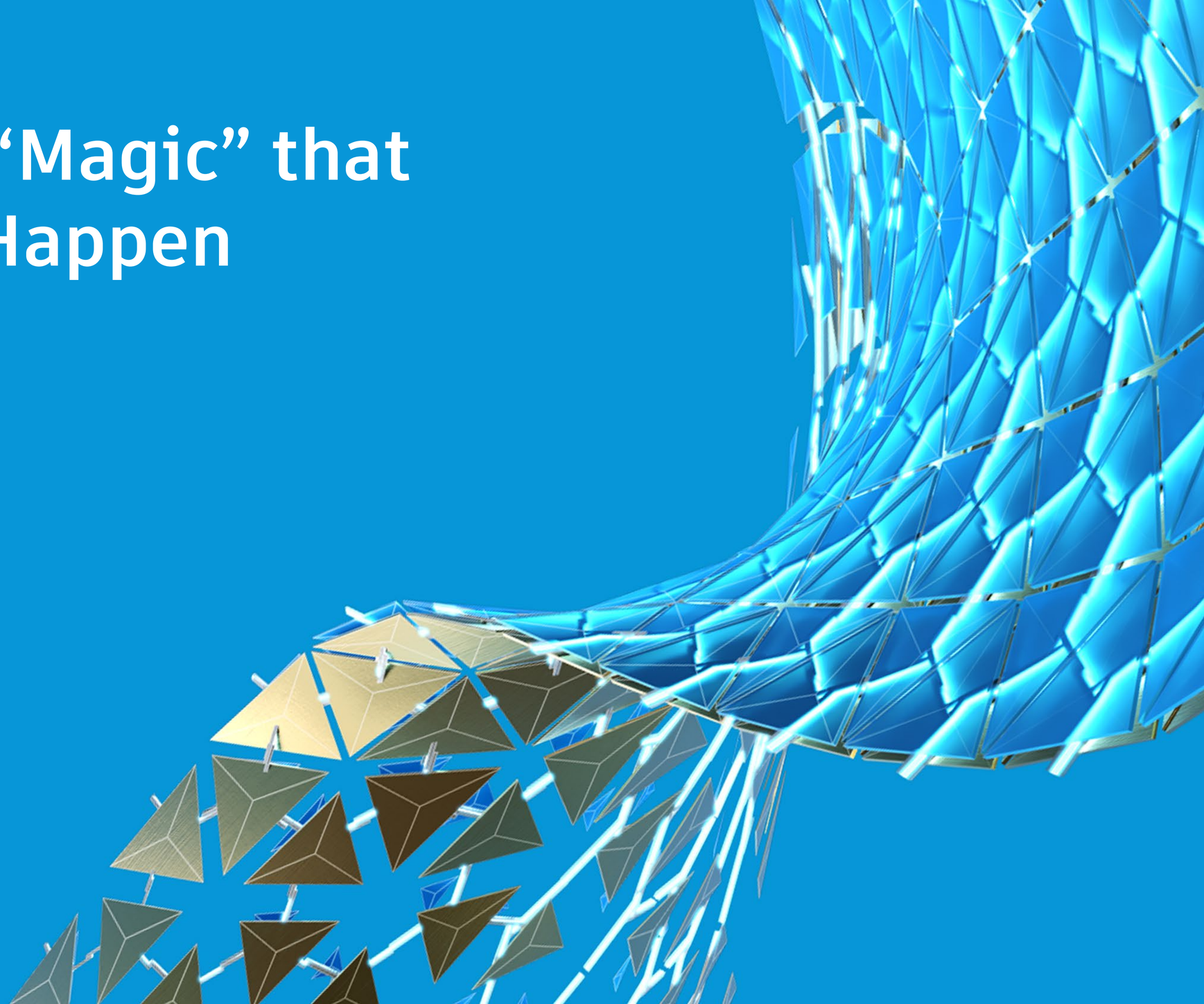


The custom BIM appearances are passed through into the Shrinkwrap Part, allowing a simplified assembly model ready for automated BIM Connectors

- The custom appearance library MUST be active for the BIM appearances to come across into the Shrinkwrap Part



iLogic – The “Magic” that Makes it All Happen



What is iLogic?

RULES BASED DESIGN

iLogic is a VB.NET based coding module that resides natively within Inventor. Conditional logic statements can be used to implement “Rules-Based” design, where specific conditions are used to impact the models

ALLOWS ACCESS TO THE INVENTOR API

The API (Application Programming Interface) is essentially the “guts” of Inventor and iLogic allows the use of higher-level functions, such as creating new parts on the fly, inside of Inventor

MULTIPLE RULES CAN BE USED TO DRIVE CASCADING FUNCTIONALITY

iLogic rules can be written for and sequenced to accomplish specific results. These rules can also reference or launch each other

How is iLogic used in the Process?

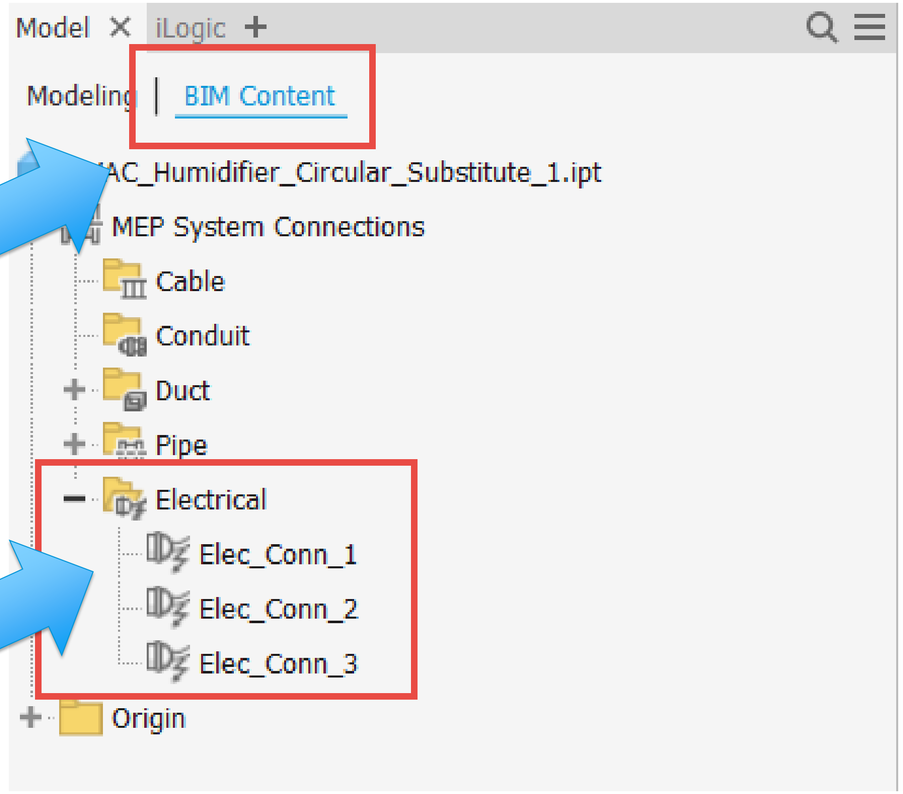
While the simplified models with BIM appearances are the backbone of the conversion process, iLogic is the heart and brains of the operation

- Navigate through the various interfaces and browsers, using the API, to access / set specific information and launch commands
- Create BIM Connectors on the Fly
- Sequence all the iLogic rules using a master control rule
- (Optional) Configure BIM data and export the model for Revit use

Navigate the Interface

Utilizing the API through iLogic is a great way to navigate through the interface and select specific browser nodes

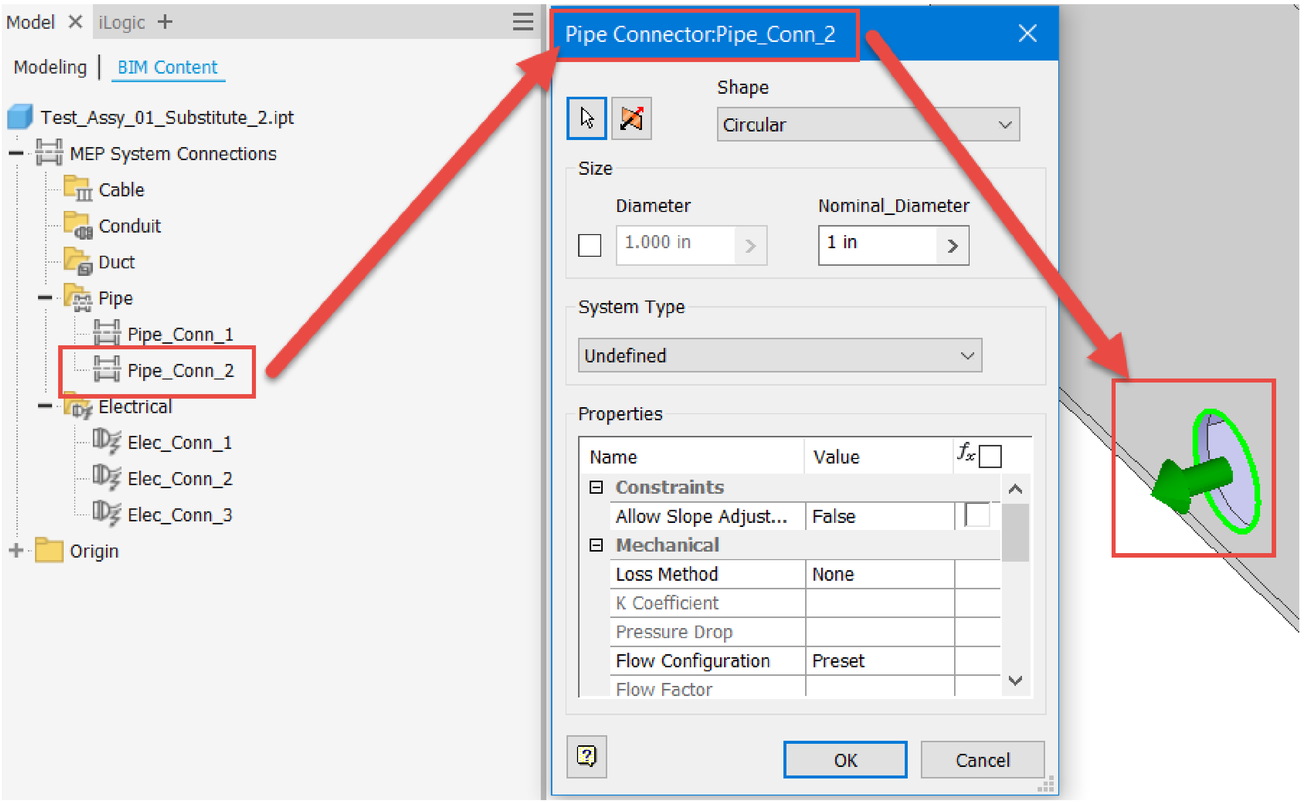
```
'Enter the BIM Content environment
ThisApplication.CommandManager.ControlDefinitions.Item("AEC_Exchange:Environm
ent").Execute()

'Access the Browser and Activate the "Bim Content" Browser Pane
Dim oPartDoc As PartDocument = ThisDoc.Document
Dim bps = oPartDoc.BrowserPanes

Dim bimContentPane As BrowserPane =
oPartDoc.BrowserPanes.Item("AEC_Exchange:Browser")
```




```
'Set the Connection Counters based on the results from the Solid_Faces and
the current counts from the BIM Browser Nodes
Dim systemNode As BrowserNode = bimContentPane.TopNode.BrowserNodes.Item("MEP
System Connections")

Dim electicalNode As BrowserNode = systemNode.BrowserNodes.Item("Electrical")
                electicalNode.Expanded = True

Dim oElecConnCount As Integer = electicalNode.BrowserNodes.Count
```

Build BIM Connectors

Check to see if a face or surface has one of BIM appearances and, if so, create and edit the appropriate BIM Connector




```
'Define the Appearance variables
Dim oElecAppearance As Asset = oPartDoc.Assets.Item("Connection_Electrical")
Dim oDuctAppearance As Asset = oPartDoc.Assets.Item("Connection_Duct")
Dim oPipeAppearance As Asset = oPartDoc.Assets.Item("Connection_Piping")
```

```
'Check through each surface in the part and see if it uses the special
appearances
For Each oWrkSurf As WorkSurface In oPartDoc.ComponentDefinition.WorkSurfaces
        For Each oSrfBody As SurfaceBody In oWrkSurf.SurfaceBodies
                For Each oFace As Face In oSrfBody.Faces

                        If oFace.Appearance.DisplayName =
oElecAppearance.DisplayName Then

                                Dim Elec_Coll As ObjectCollection
                                Elec_Coll =
ThisApplication.TransientObjects.CreateObjectCollection

                                Elec_Coll.Add(oFace)
```

```
' Get the collection of control definitions.
Dim oControlDefs As ControlDefinitions =
ThisApplication.CommandManager.ControlDefinitions

Dim oControlDef As ControlDefinition =
oControlDefs.Item("AEC_Exchange:Command:Connector:Edit")
```

```
'Activate the current electrical node so characteristics can be modified, if
desired.
Dim oConnNode As BrowserNode = electicalNode.BrowserNodes.Item(con.Name)
oConnNode.DoSelect()
oControlDef.Execute()
```

Use a Control Rule to Launch Other Rules

If Rules are required to run in a particular order, a control iLogic rule is a great way to sequence the launching of other rules

- **Rules are launched in a top-to-bottom fashion**
  - Can be internal or External

```
'Check all the Solid Faces for the BIM Custom Appearances and create
connectors
iLogicVb.RunExternalRule("Jelte_BIM_Solid_Faces")

'Check all the Surface Boundary Patches for the BIM Custom Appearances and
create connectors
iLogicVb.RunExternalRule("Jelte_BIM_Surfaces")

'Export the model as a Revit family with the same root filename in the same
file location
iLogicVb.RunExternalRule("Create_Revit_Family_Basic")

iLogicVb.UpdateWhenDone = True
```

Export the Revit Family (RFA) Behind the Scenes (Optional)

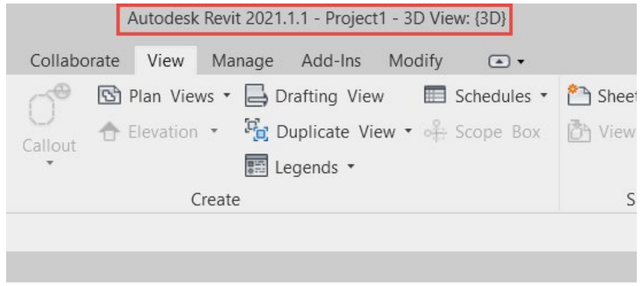
For a more hands-off experience, the creation of the Revit family can be automated, utilizing the API

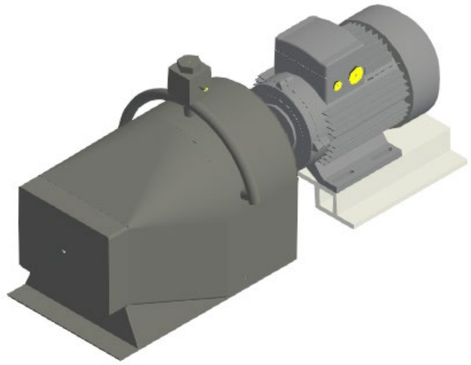
- Using the API, a rule can access the file location and file name of the substitute part, as well as generate the final Revit family

```
'Extract the current filename And parse Out the .ipt file extension
Dim Comp_Name As String = oPartDoc.DisplayName
Dim Comp_Name_Count As Integer = Comp_Name.Length
Dim Final_Comp_Name As String = Left(Comp_Name, Comp_Name_Count - 4
```

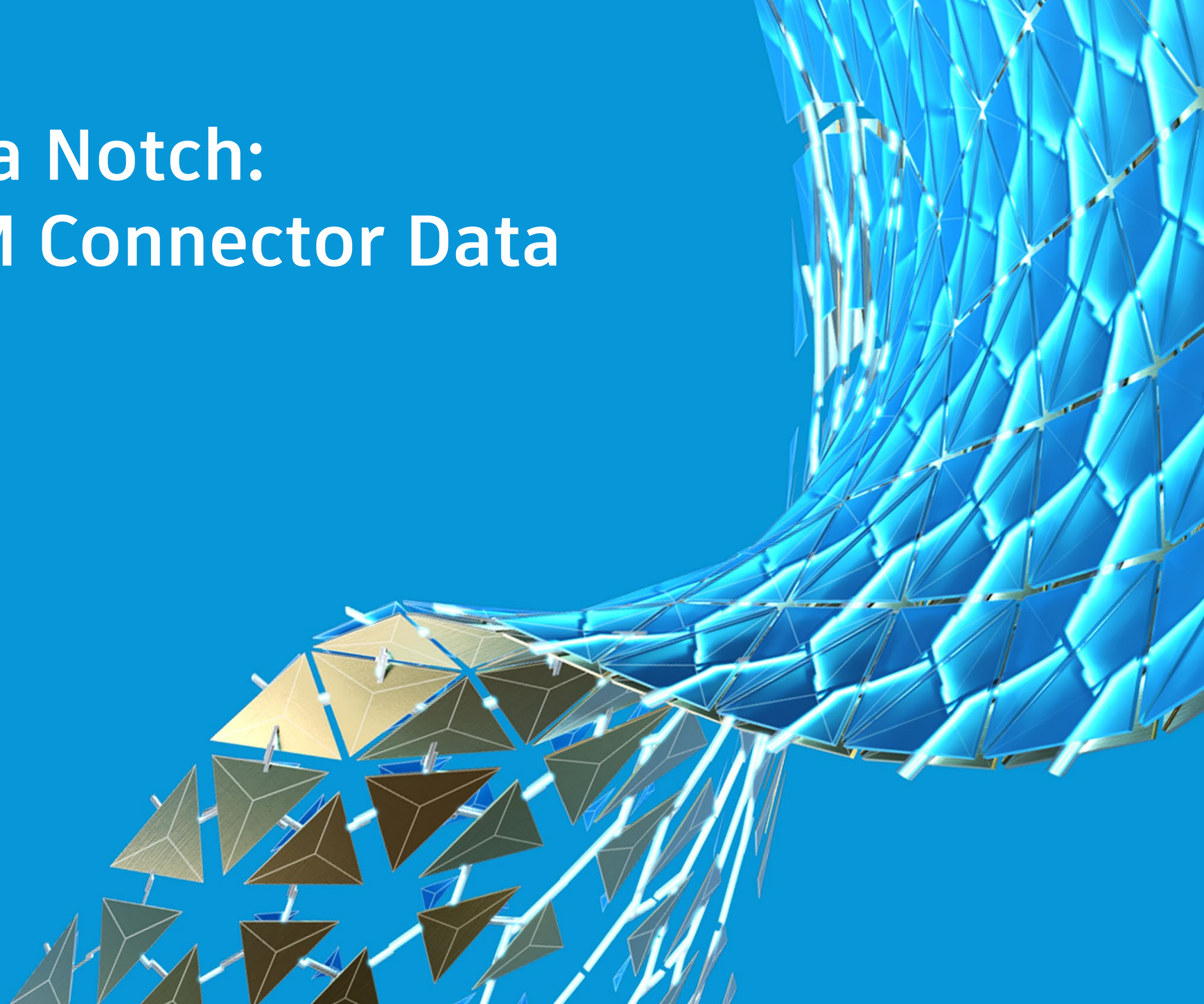
```
'Set up the file path from the original host file
Dim File_Path_Length As Integer = oPartDoc.FullFileName.Length
Dim File_Path As String = Left(oPartDoc.FullFileName, File_Path_Length -
Comp_Name_Count)
```

```
'Create the Revit Family using the extracted file path and formed Revit file
name
oPartDocBIM.ExportBuildingComponent(File_Path & Final_Comp_Name & ".rfa")
```





Taking it Up a Notch: Populate BIM Connector Data



Can BIM Connector Data Entry Be Automated?

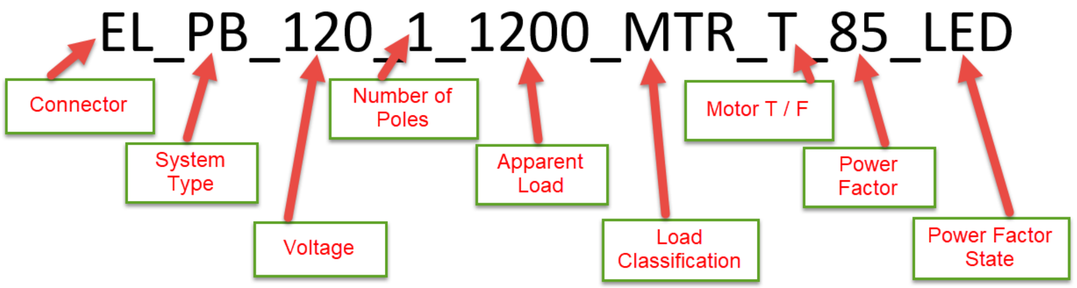
With a handful of small tweaks to the process, and some forethought, the BIM Connector data entry process can be automated, saving lots of time day-to-day

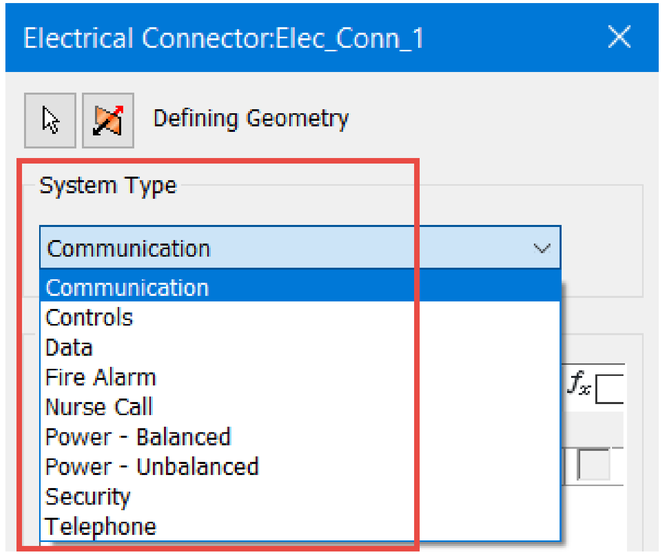
- Implement more precise, information-rich appearance names
- Utilize iLogic code to specify the BIM Connector type, sub-type and additional BIM information

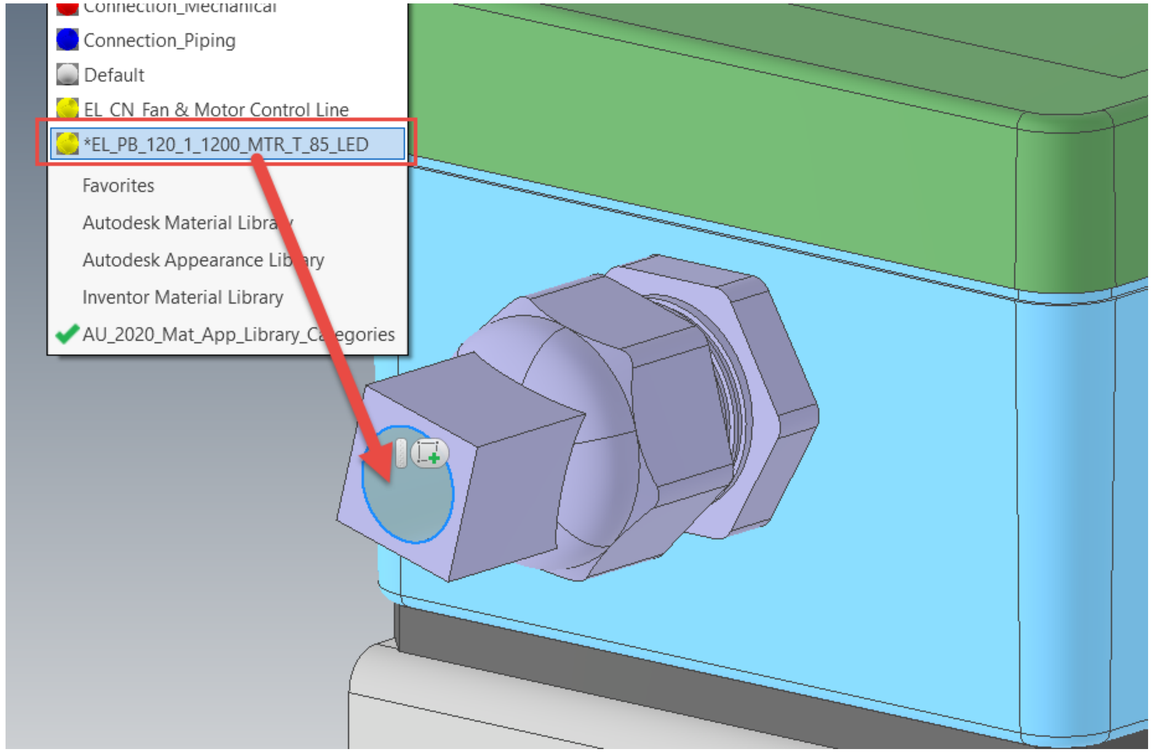
Implement Information Rich Appearance Names

In lieu of generic BIM appearance names, precise character strings can be implemented to aid in specific BIM Connector data entry process

- Implement specifically sequenced characters that iLogic will utilize later
- For consistency's sake be sure to utilize a custom appearance library





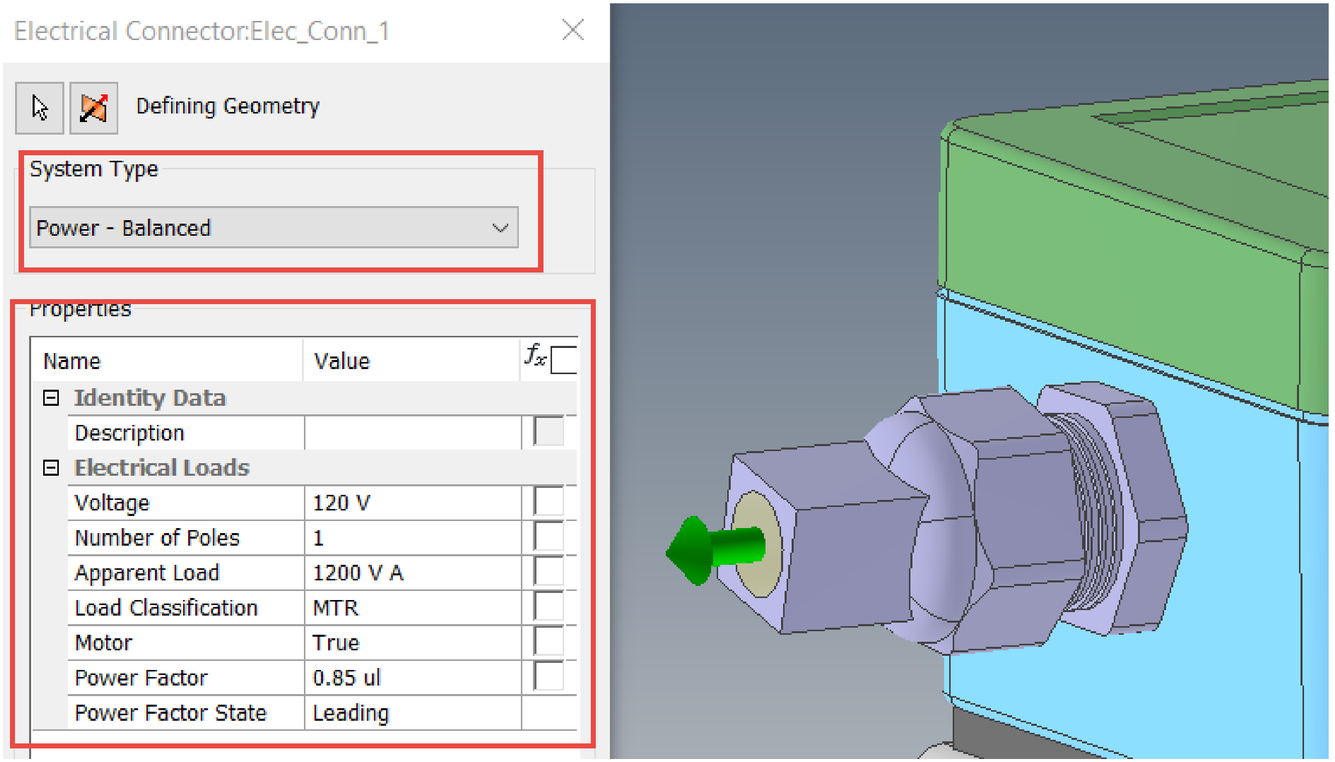


Utilize Precise iLogic Code to Reduce Data Entry

With more precise appearance naming, comes more precise iLogic coding that takes advantage of the character sequencing

- Determine the BIM Connector type and sub-type using a consistent character string

```
'Define a variable to determine which type of Connector Surface is present
Dim BIM_Conn_Type As String = Left(oFace.Appearance.DisplayName, 3)
'Determine if this is an Electrical Connector
If BIM_Conn_Type = "EL_" Then

'Create a variable to determine what kind System Type is being utilized
Dim Elec_Sys_Type As String = Mid(oFace.Appearance.DisplayName, 4, 2)
'Determine which Electrical System Type is being used and adjust accordingly
Select Case Elec_Sys_Type

Case "PB"
        oElecConn.SystemType =
BIMElectricalSystemTypeEnum.kPowerBalancedElectricalSystemType

        oElecConn.Voltage = Mid(oFace.Appearance.DisplayName, 7, 3)

        oElecConn.NumberOfPoles = Mid(oFace.Appearance.DisplayName, 11, 1)

        oElecConn.ApparentLoad = Mid(oFace.Appearance.DisplayName, 13, 4)
```



Future Research / Things to Watch For

What Next?

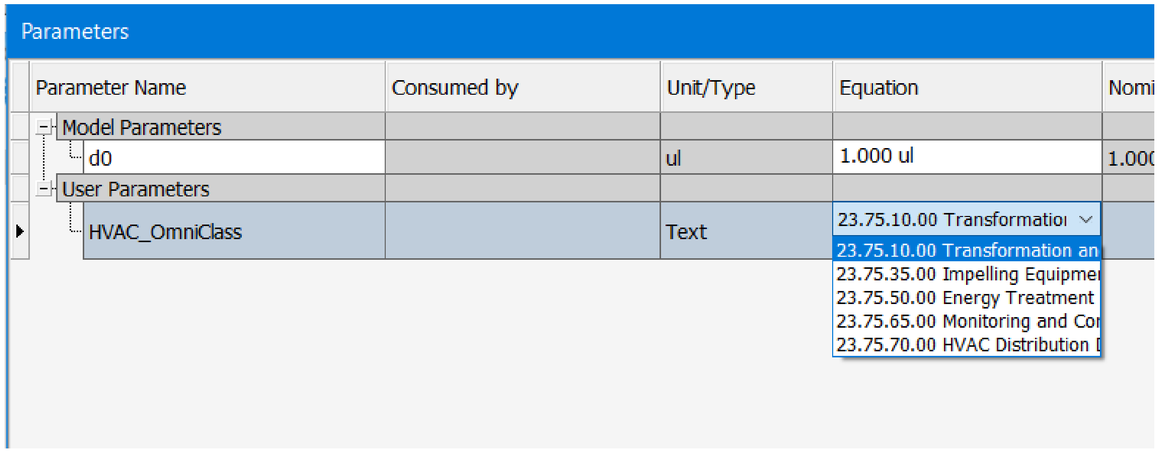
No process is perfect. Sometimes challenges exist that force a particular methodology to be employed or technology improves allowing process to be more efficiently revised. Here are some items that can be improved in the future.

- Incorporating the OmniClass and other BIM properties more efficiently into the models
- 3rd party components don't always display surfaces
- Revit conversion challenges

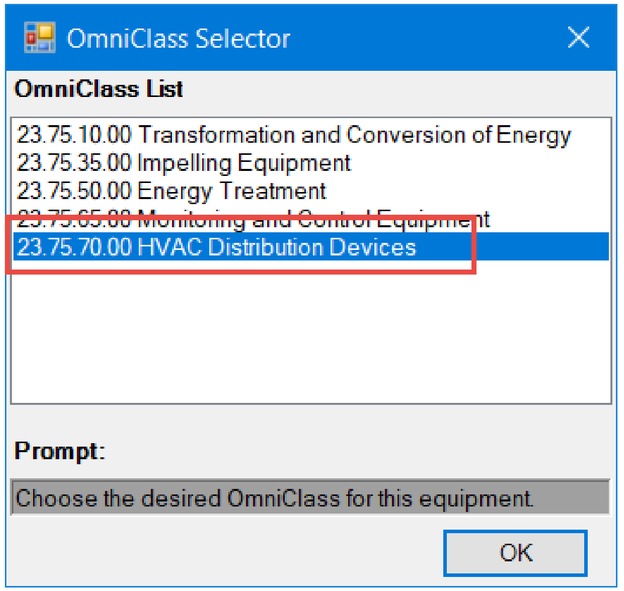
Efficiently Incorporating the OmniClass

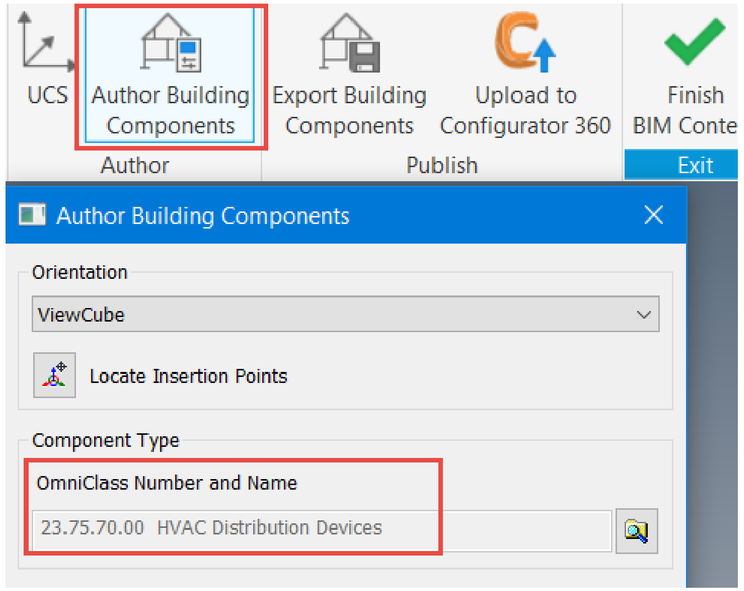
Other BIM data, such as OmniClass can be added to the Inventor model, prior to the Revit publishing process, which presents some challenges

- OmniClass data is vast!
- Currently utilizing specific equipment templates
  - User parameters that list specific OmniClass data
  - iLogic code provides users with a selection list during the authoring process




```
HVAC_OmniClass = InputListBox("Choose the desired OmniClass for this
equipment.", MultiValue.List("HVAC_OmniClass"), HVAC_OmniClass, Title :=
"OmniClass Selector", ListName := "OmniClass List")

'Push the OmniClass value to the BIM Component Properties
oPartDocBIM.ComponentDescription.ComponentType = Left(HVAC_OmniClass,11)
```

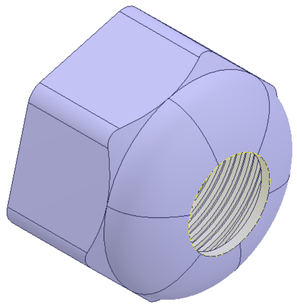


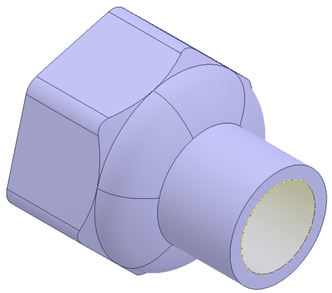


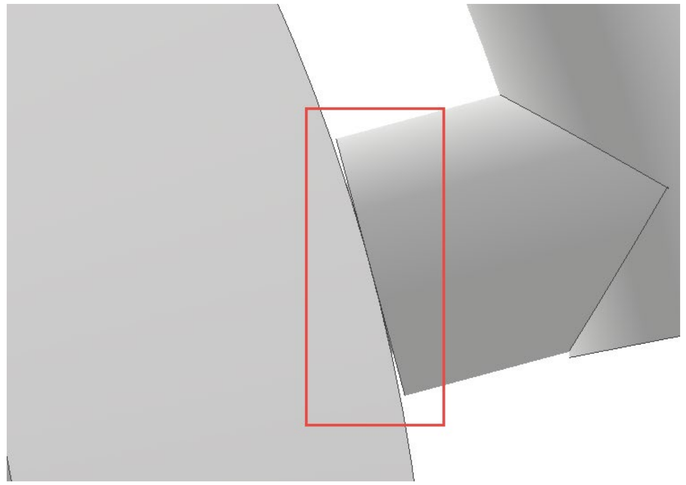
Challenges

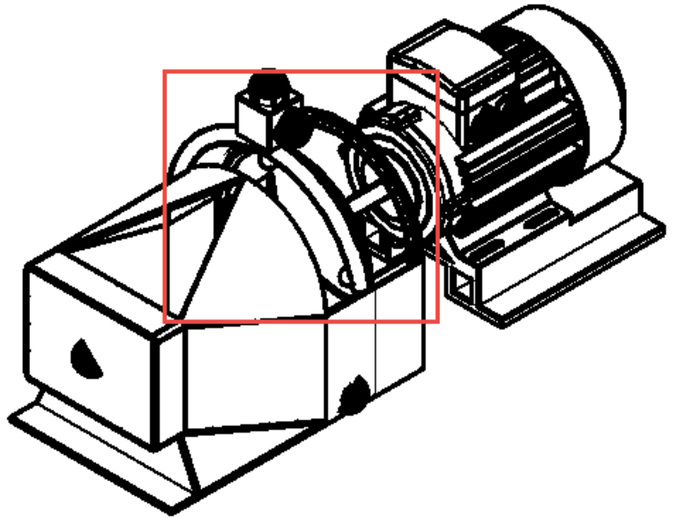
Some portions of the process did not initially work well and some still need to be fixed.

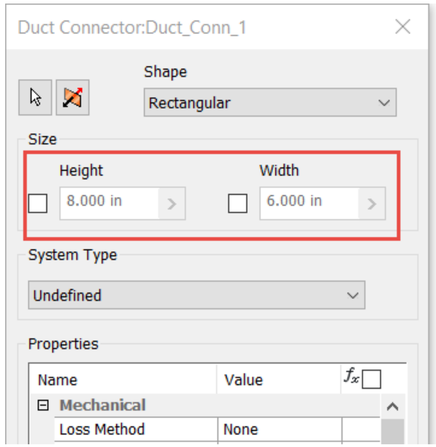
- **3rd party surfaces didn't display after the shrinkwrap**
- **Revit surfaces were missing after the conversion**
  - If small gaps or overlapping surfaces occurred, the geometry didn't convert properly into Revit
- **Rectangular ductwork authored in Inventor has connection troubles in Revit**
  - Autodesk development teams are looking into this issue

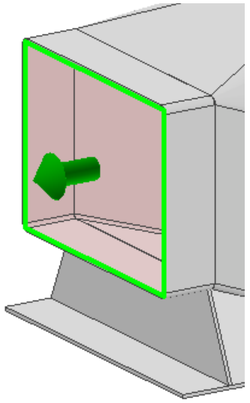


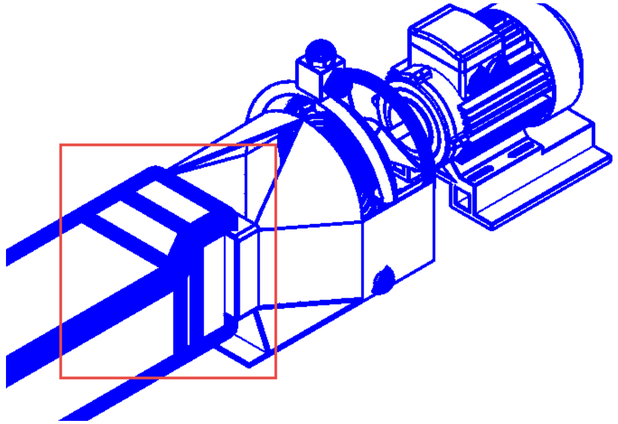












Special Thanks!

- God for this wonderful opportunity and literally every breath I take
- Scott Dibben for allowing me to explore my zany ideas at work
- Jelte de Jong who’s help was invaluable to getting my iLogic code sorted out
- Hakan Yildirim who provided the controller model
- All of you for being great sports and participating in this rather unique AU experience