AUTODESK UNIVERSITY 2013

Automate your Autodesk® Revit® Workflows: Let the API Do the Work

David Rushforth – Glumac

MP2153 This class provides a big picture overview of how the API interacts with your projects as well as gives specific training on using the API to automate your workflows. We discuss actual time-saving solutions that are made possible with the API, with a focus on giving participants the general understanding and resources needed to start their own API projects. We examine specific code samples, showing how to extract information from elements in linked models, and we cover some practical uses of the ray trace functions of the Revit API. We also present the use of the API to automate data manipulation and discuss the transfer of data from other sources. The class includes discussion of example applications and discussion of ideas for future development.

Learning Objectives

At the end of this class, you will be able to:

- Recognize the potential for the API to accelerate production in the real world
- Understand the API development process as a whole
- Use the API to query linked models for element information
- Identify methods of data import/export and model manipulation using the API

About the Speaker

David is an electrical engineer for Glumac Engineers with experience designing millions of square feet of projects using the Revit software. In addition to project design, David oversees the integration of BIM technologies into the engineering practices of the firm's 10 offices. His experience includes software development in integrating calculations and office applications with design software such as Autodesk Revit and SKM Power Tools. He shares many of his software solutions at www.rushforthprojects.com . David is a registered professional engineer and LEED accredited professional. He holds a Master's Degree in Electrical Engineering from UNLV and a Bachelor's Degree in Electrical Engineering from BYU.

david@rushforth.org

www.rushforthprojects.com

INTRODUCTION

Autodesk Revit is a powerful tool, yet there are many tasks that still require a great deal of time to accomplish. If you have ever been required to perform a repetitive task in the software and thought, “there must be a better way,” then you are not alone. Often the application programming interface (API) can be used to carry out tasks that you would normally have to do manually.

Of course, the API can be intimidating and take some time to tame, but the benefits are immense. The primary goal of this class is to provide you with the understanding and resources to start using the API today. Once immersed in the API, you have a whole new set of tools with which to confront various problems. No longer are you limited by the way the software was “intended to run”, but you begin to make the software work in ways that compliment your workflows.

The first two sections are geared for beginners to gain a better understanding of what the API is and how it interfaces with your projects. An overview of the whole development process and how to get started are explained. The next sections include explanations and source code to use in your own projects. The macro code given is complete with instructions and can be directly pasted into your code editing software to create a running application. The end sections discuss specific programming tasks, and key lines of code are given to illustrate how to accomplish each task in your own projects. I hope you find this reference helpful, and I look forward to hearing how it has helped any that read it.

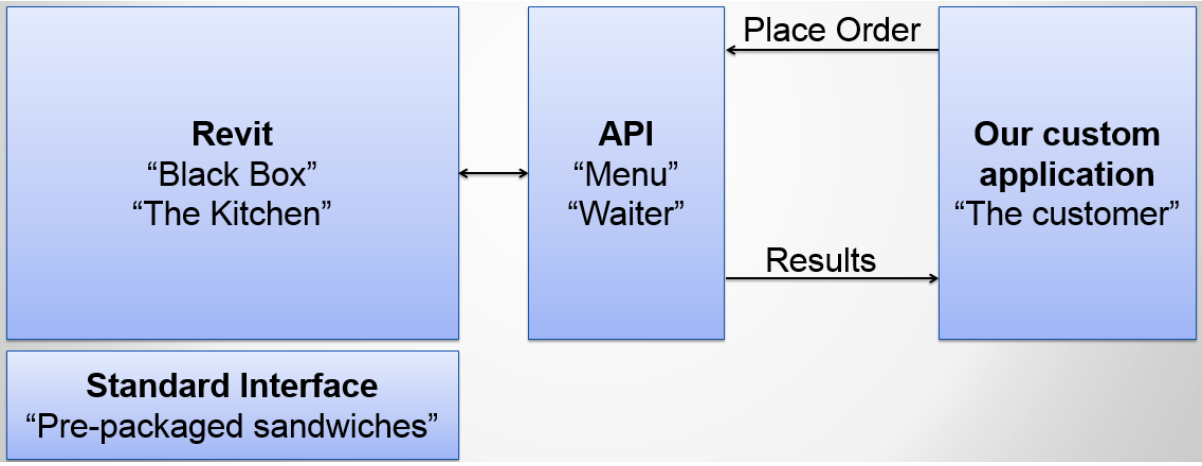
UNDERSTANDING THE API AND ITS POTENTIAL

What is an API?

I start with this basic question because I myself was pretty ignorant to the whole process and had lots of ideas of what an "API" could be. I wondered, 'Is it a new software interface?' 'Do I get to change the source code of the software?' 'Is it a new programming language or script language?'

Basically, the Application Programming Interface (API) is a list of commands or functions that you can call to tell Revit to do something. The list of functions is stored in a couple .dll files that are provided with Revit. You write your own code in C#, VB.NET, Ruby, or Python and 'reference' the API dll's. That gives you access to call the functions that are provided by the API.

Think of Revit like a kitchen that provides sandwiches. Users of the standard Revit interface are able to pick up pre-packaged sandwiches that are generic and mass produced. If you want to have some input like choosing the bread, meat, and toppings for your sandwich, then you need to pick up a menu. The API is like a menu to the kitchen. You get to order items on the menu and select options with your order. Once you order, your custom results are given back to you. You don't, however, get access to the kitchen to see what ingredients they use and how they prepare the results. Once you've learned how to order things from the menu, the pre-packaged sandwiches just aren't the same.



What is a DLL file?

I had seen DLL files for years, but the only thing I knew about them was that they seemed to be related to my computer crashing and that they were essential for some programs to work properly. I found out that a DLL is a collection of compiled code and/or resources (like embedded images, icons, or files). It is the same thing as an application, like an exe, except

that it doesn't have a set start point to launch its code. It is called by another application which tells it which of its functions to execute.

You can compile your own DLL for others to use, and you can reference other DLL files to gain access to their functions. If you want access to manipulating Excel files, for instance, you must first reference its DLL so that you have a list of the available functions. That way, your application will know what you mean by a worksheet or cell and what functions are available to manipulate them.

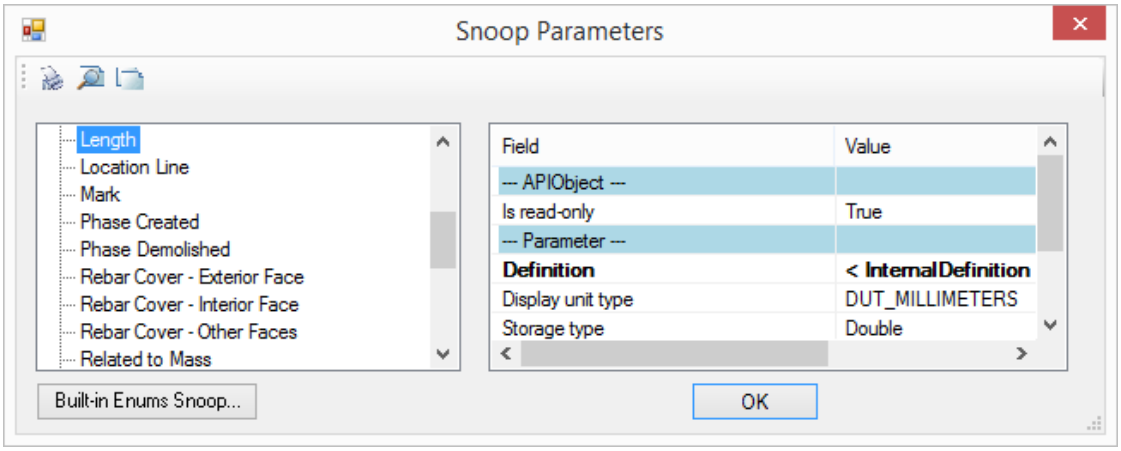
What is the .NET Framework?

The .NET Framework is part of the Windows API and gives you access to a library of functions and preset dialogs. For example, you don't have to figure out how to browse the files system and how to display files and folders to select or create files. You just call a preset dialog that already has the interface and programming built into it.

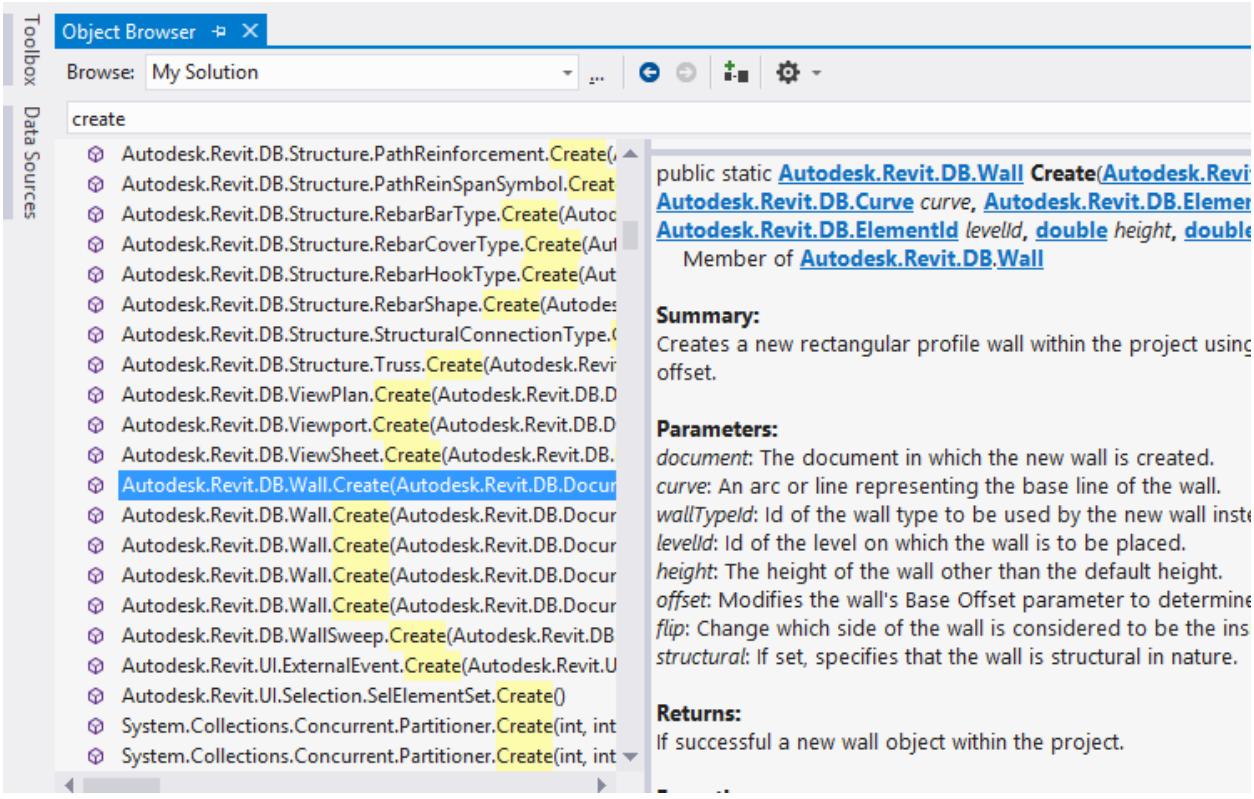
So what do I have access to manipulate in Revit?

You have access to call the functions that are provided in the Revit API DLLs. You also have access to standard programming libraries as well as the .NET framework. That means you can retrieve Revit data, process it however you wish, then you can decide what to do with the results. You can export data, use it to set properties, settings, or parameter data, or even create new elements in Revit. Some basic example of what the API gives access to: family creation, parameter creation, parameter value manipulation, live monitoring of changes/activity, creation of views, sheets and schedules. You can also run certain code based on events such as file saving or dialogs being shown.

When you are programming and want to find out whether you have access to do what you want, the Revit Software Development Kit (SDK) and the Object Browser tool in Microsoft Visual Studio are good places to start. The SDK is basically just some instructions with a help file packaged with a lot of example code for applications. Doing a search in the help file may lead you to what you were looking for. In the SDK there is an addin called Revit Lookup that you can load into your Addins tab of Revit. This addin is useful for identifying the properties and parameters associated with elements or the project in general. It is a quick way to see most of the data that is present in any particular element.



The Object Browser view in Microsoft Visual Studio will let you list and search all of the available classes and functions in the API. It will show you how to call the function and will explain any of the inputs that are used with the function. This can be helpful for finding what is available as well as the path and needed inputs to call various functions.



Which programming language should I use?

The answer is, it really doesn't matter. You have access to all the same functions no matter which language you use. Using the previous restaurant analogy, you are ordering the same food from the same kitchen, even if the menu is in another language. C# and VB.NET are

.NET languages, so you will have more automated error checking and helps using Microsoft Visual Studio. Python and Ruby (newly supported by Revit 2014) are less rigidly structured which can lead to greater flexibility in the way you write the code, but you give up some of the auto-complete and error checking. If you are asking the question, then you should probably start with a .NET language. You can always convert code from one language to another with the click of a button (The SharpDevelop environment that comes with Revit can do the conversion from the Tools menu).

API DEVELOPMENT PROCESS AND RESOURCES OVERVIEW

The API and Your Revit Project

Project Types

When creating code that interfaces with Revit via the API, you basically have three main options:

1. Macro
    a. A command that you manually start and stop
    b. Stored in the project or on a single computer
2. External Command
    a. Has a set start and stop point
    b. Compiled in a DLL file (easy to share)
    c. Example: A user clicks a button to start the command. The command performs its function, like create a 3D view from a selected region, then stops.
3. External Application
    a. Starts with Revit, ends with Revit
    b. Continuously running
    c. Compiled in a DLL file
    d. Example: Revit starts a ribbon application on startup and the ribbon stays visible until Revit closes.

This also means that at this time, you can't write stand-alone applications or code in applications like Excel or AutoCAD that read and write to the database of a Revit project. Revit must initiate the communication. That doesn't mean that you can't trigger the communication externally. It just means that Revit must be running and that the Revit API controls the data being exported and imported.

Letting Revit Know How to Start Your Application

When Revit starts up, it will check a few standard locations to see if there are any .addin text files. Here are some of those locations:

- C:\ProgramData\Autodesk\ApplicationPlugins
- C:\ProgramData\Autodesk\Revit\Addins\2013
- C:\Users\<INSERT YOUR USERNAME HERE>\AppData\Roaming\Autodesk\Revit\Addins\

The .addin file is a simple text file that tells Revit the path to a DLL file that contains an applications and the name of the function to start. It also contains an ID code for your application which must be unique for a given session of Revit. You can generate a unique GUID for your app, or if you prefer, find an example .addin file (like in the SDK) then randomly pick some of the hexadecimal characters in the example ID to change. That should be safe enough to get started with.

If your .addin file points to an 'external command', then your command will be listed in the Add-Ins tab in Revit under 'External Tools'. If your .addin points to an 'external application' that creates its own ribbon buttons, you will see your new buttons on the Ribbon tab name you indicate in your application code (or the Add-Ins tab if you don't specify a new name).

For more information on how to create custom buttons and how to link them to additional commands, refer to the example given in the Revit SDK.

Getting Started and Understanding Visual Studio (and general programming)

Software to Install:

- Visual Studio (Express versions are free)
- Download Revit SDK (not required, but gives examples and documentation)

Namespace, Class Name, and Function

In order to call functions in programming, you must specify the path to the function. Many paths are made up of Namespace.ClassName.FunctionName.
Examples: System.Environment.GetFolderPath() or
System.Windows.Forms.MessageBox.Show() .

Instead of having to write System.Windows.Forms.MessageBox.Show() every time you want a message box, you can just type MessageBox.Show() as long as you have specified "System.Windows.Forms" as one of the known shortcut paths at the top of your code. This is why you see 'Imports' (VB.NET) or 'Using' (C#) at the top of programming projects. I just want to make a couple points on this subject.

1. The Imports/Using statements at the top of the code are optional. If you don't have them, you will just have to specify whole paths when calling functions.
2. Many times when segments of code are shared, people don't include the top shortcut paths, so you may paste in their code only to see a lot of errors because it can't find the functions. You may have to check the object browser or online resources to find the class path you need to include.

In the Macro code example below, I have intentionally use full paths for all functions so no Imports functions are required.

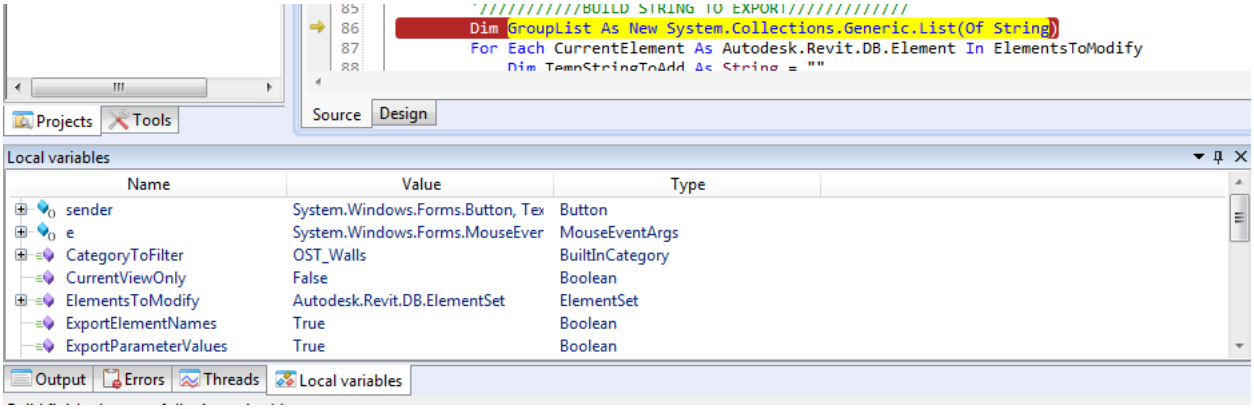
References

As mentioned before, you must add external DLLs as references in order to have access to their functions. This can be done by right-clicking the 'References' section of the Solution Explorer of Microsoft Visual Studio. For a Revit API application, the first thing you need to do is add the RevitAPI.dll and the RevitAPIUI.dll to your programming project. They can be found in the Revit program folder. In the properties of the added references, you will find a property called 'Copy Local' this should be set to False. This should only be set to true for custom DLL references that are not normally found in windows, Revit, or in the same folder as your application. Once you have a reference in your project, you can now use its classes and functions in your code.

Locals Window

When stepping through your code while debugging, Microsoft Visual Studio has a window that can show 'Local variables' that displays all the variables being set by your application along with the values stored in the variables. It will even indicate which values have

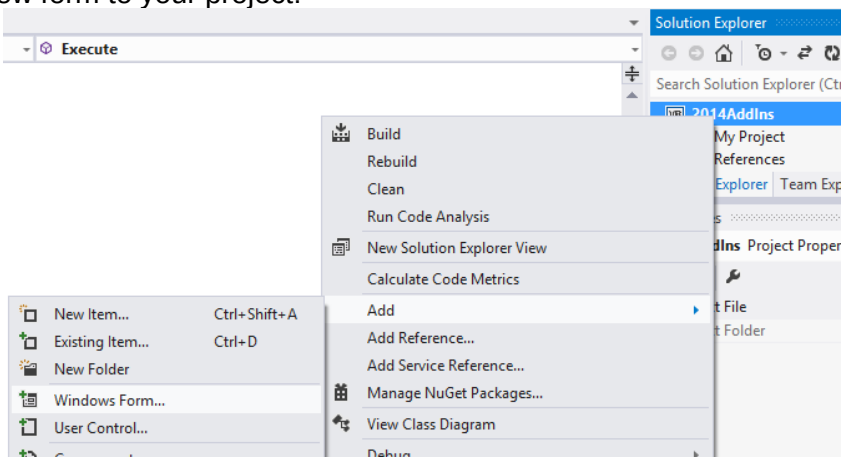
been changed by the previous line of code. This can be very helpful in troubleshooting code and determining which functions or lines of code are affecting your variable values.



Displaying a form

One of the first things you may want to do when creating an application is display a form. This is relatively straight forward and well documented, but if you don't pass your new form access to the Revit database, you may have a hard time doing useful functions with your form.

Add a new form to your project:



In your command code that is being called to start by Revit, tell your form to show and pass it the ExternalCommandData that Revit passed to your command.

```
Dim NewFormToDisplay As New FormIJustCreated
NewFormToDisplay.ShowDialog(MyRevitData)
```

In your newly created form, add an alternate 'New' function that accepts the ExternalCommandData and assigns it to a variable that exists in your form's class outside of all functions.

```
Private RevitDataToUse As ExternalCommandData
Public Sub New(ByVal IncomingRevitData as ExternalCommandData)
   RevitDataToUse = IncomingRevitData
```

```
    InitializeComponent()
End Sub
```

EXAMPLE MACRO CODE

Below are some simple instructions and example code that can be directly pasted into your macro project to begin experimenting with the Revit API.

- **In Revit, open the Macro Manager from the Manage tab**
- **Use the buttons to create a new 'Module' and new 'Macro' in the 'Application' tab of the Macro Manager**
- **In the code editor, create a new form in your macro project**
- **Call a form from your macro:**
  - ```
    Dim MyUIDoc As UIDocument = Me.ActiveUIDocument
    'Dim FormToShow As New MyAUForm
    Dim FormToShow As New MyAUForm (MyUIDoc)
    FormToShow.ShowDialog
    ```
- **Create an alternate 'New' routine in your form to pass in data**
  - ```
    Public Sub New(IncomingUIDoc As Autodesk.Revit.UI.UIDocument)
        Me.InitializeComponent()

        MyUIDoc = IncomingUIDoc
        MyDoc = IncomingUIDoc.Document
    End Sub
    ```
- **Declare variables in the form class, but outside functions, that you want accessible everywhere**
  - ```
    Dim MyUIDoc As Autodesk.Revit.UI.UIDocument
    Dim MyDoc As Autodesk.Revit.DB.Document
    ```
- **Example code for filtering and extracting data**
  - ```
    Dim SelectedItemsOnly As Boolean
        Dim CurrentViewOnly As Boolean
        Dim CategoryToFilter As Autodesk.Revit.DB.BuiltInCategory
        Dim FinalElementResult As System.Collections.Generic.ICollection(Of Autodesk.Revit.DB.Element)
        Dim ElementsToModify As Autodesk.Revit.DB.ElementSet = New Autodesk.Revit.DB.ElementSet
        Dim SelectedElements As Autodesk.Revit.DB.ElementSet = MyUIDoc.Selection.Elements
        Dim ParameterNameToUse As String
        Dim UseSelectionWithoutFilter As Boolean
        Dim ExportElementNames As Boolean
        Dim ExportParameterValues As Boolean
        Dim InstanceTypeEitherOption As String

        '//////////SELECTION INPUT///////////////
        UseSelectionWithoutFilter = True
        SelectedItemsOnly = True
        CurrentViewOnly = False
        CategoryToFilter = Autodesk.Revit.DB.BuiltInCategory.OST_Walls
        ExportElementNames = True
        ExportParameterValues = True
        ParameterNameToUse = "Mark"
        InstanceTypeEitherOption = "Either"
        '//////////END SELECTION INPUT///////////////

        '////////////RUN CATEGORY FILTERS//////////////
        If UseSelectionWithoutFilter = True Then
    ```

```
        For Each currentElement As Autodesk.Revit.DB.Element In SelectedElements
            ElementsToModify.Insert(currentElement)
        Next
    Else
        Dim OtherElementCollector As Autodesk.Revit.DB.FilteredElementCollector
        If CurrentViewOnly = True Then
            OtherElementCollector = New Autodesk.Revit.DB.FilteredElementCollector(MyDoc,
MyDoc.ActiveView.Id)
        Else
            OtherElementCollector = New Autodesk.Revit.DB.FilteredElementCollector(MyDoc)
        End If

        Dim MyOtherFilter As New Autodesk.Revit.DB.ElementCategoryFilter(CategoryToFilter)
        Dim MyInvertedFilter As New Autodesk.Revit.DB.ElementClassFilter(GetType(Autodesk.Revit.DB.Elem
entType), True) 'don't include element types
        Dim NewCombinedFilter As New Autodesk.Revit.DB.LogicalAndFilter(MyOtherFilter, MyInvertedFilter)

        FinalElementResult = OtherElementCollector.WherePasses(NewCombinedFilter).ToElements

        If SelectedItemsOnly = True Then
            For Each currentElement As Autodesk.Revit.DB.Element In FinalElementResult
                If SelectedElements.Contains(currentElement) Then
                    If currentElement.Category.Id.IntegerValue = CInt(CategoryToFilter) Then
                        ElementsToModify.Insert(currentElement)
                    End If
                End If
            Next
        Else
            For Each currentElement In FinalElementResult
                If currentElement.Category.Id.IntegerValue = CInt(CategoryToFilter) Then
                    ElementsToModify.Insert(currentElement)
                End If
            Next
        End If
    End If

    '////////////END RUN FILTERS//////////////

    '////////////BUILD STRING TO EXPORT//////////////
    Dim GroupList As New System.Collections.Generic.List(Of String)
    For Each CurrentElement As Autodesk.Revit.DB.Element In ElementsToModify
        Dim TempStringToAdd As String = ""

        If ExportElementNames = True Then
            TempStringToAdd = "[" & CurrentElement.Name & "] "
        End If
        If ExportParameterValues = True Then
            TempStringToAdd = TempStringToAdd & GetParameterValuebyName(currentElement,
ParameterNameToUse, InstanceTypeEitherOption)
        End If
        GroupList.Add(TempStringToAdd)
    Next
    '////////////END BUILD STRING TO EXPORT//////////////

    '////////////OUTPUT TO FILE//////////////
    Dim writefilename As String
    Dim MyDesktopPath As String
    MyDesktopPath = System.Environment.GetFolderPath(System.Environment.SpecialFolder.Desktop)
```

```
        writefilename = MyDesktopPath & "\MyOutputFile.txt"

        Dim fs As New System.IO.FileStream(writefilename, System.IO.FileMode.Create,
System.IO.FileAccess.Write)
        Dim s As New System.IO.StreamWriter(fs)

        For Each partialmessage As String In GroupList
            s.WriteLine(partialmessage)
        Next

        s.Close()
        '//////////END OUTPUT TO FILE//////////////

        Dim ps As New System.Diagnostics.ProcessStartInfo
        ps.UseShellExecute = True
        ps.FileName = writefilename
        System.Diagnostics.Process.Start(ps)

        System.Windows.Forms.MessageBox.Show("Done.")
```

- **Supporting functions**
    - ```
      Public Function GetParameterValuebyName(currentElement As Autodesk.Revit.DB.Element,
      ParameterName As String, InstanceTypeEither As String) As String
          Dim CurrentParamValue As String
          CurrentParamValue = ""

          '///INSTANCE PARAMETERS
          If InstanceTypeEither = "Instance" Or InstanceTypeEither = "Either" Then
              Dim InstanceParams As Autodesk.Revit.DB.ParameterSetIterator =
      currentElement.Parameters.GetEnumerator
              While InstanceParams.MoveNext
                  Dim InstanceParam As Autodesk.Revit.DB.Parameter
                  InstanceParam = InstanceParams.Current
                  If InstanceParam.Definition.Name.ToUpper = ParameterName.ToUpper Then
                      CurrentParamValue = ReturnParameterValue(InstanceParam)
                  End If
              End While
          End If

          '////TYPE PARAMETERS////
          If CurrentParamValue = "" Then 'wasn't set by an instance param
              If InstanceTypeEither = "Type" Or InstanceTypeEither = "Either" Then
                  Dim typeParams As Autodesk.Revit.DB.ParameterSetIterator
                  Dim MyTypeParamsList As Autodesk.Revit.DB.ParameterSet
                  MyTypeParamsList = MyDoc.GetElement(currentElement.GetTypeId()).Parameters
                  typeParams = MyTypeParamsList.GetEnumerator()
                  If typeParams IsNot Nothing Then
                      While typeParams.MoveNext
                          Dim currentTypeParam As Autodesk.Revit.DB.Parameter
                          currentTypeParam = typeParams.Current
                          If currentTypeParam.Definition.Name.ToUpper = ParameterName.ToUpper Then
                              CurrentParamValue = ReturnParameterValue(currentTypeParam)
                          End If
                      End While
                  End If
              End If
          End If
          Return CurrentParamValue
      ```
      ```

```
End Function

Private Function ReturnParameterValue(ParameterToReturn As Autodesk.Revit.DB.Parameter) As String
  Dim CurrentParamValue As String = ""
  Select Case ParameterToReturn.StorageType
    Case Autodesk.Revit.DB.StorageType.[Double]
      Dim doubleValue As Double = ParameterToReturn.AsDouble()
      Try
        Dim unitValue As Double = ConvertFromAPI(ParameterToReturn.DisplayUnitType, doubleValue)
        CurrentParamValue = CStr(unitValue)
      Catch
        CurrentParamValue = ParameterToReturn.AsDouble.ToString
      End Try
      Exit Select
    Case Autodesk.Revit.DB.StorageType.ElementId
      CurrentParamValue = "ID:[" & ParameterToReturn.AsElementId.IntegerValue.ToString & "]"
      Exit Select
    Case Autodesk.Revit.DB.StorageType.[Integer]
      CurrentParamValue = ParameterToReturn.AsInteger.ToString
      Exit Select
    Case Autodesk.Revit.DB.StorageType.None
      CurrentParamValue = ParameterToReturn.AsValueString()
      Exit Select
    Case Autodesk.Revit.DB.StorageType.[String]
      CurrentParamValue = ParameterToReturn.AsString()
      Exit Select
    Case Else
      Exit Select
  End Select

  Return CurrentParamValue

End Function
```

- **Conversion code for Revit API (this code is available online and in multiple examples)**
  -
```
Public Shared Function ConvertFromAPI(ByVal dut As Autodesk.Revit.DB.DisplayUnitType, ByVal value As
Double) As Double
  Select Case dut
    Case Autodesk.Revit.DB.DisplayUnitType.DUT_FAHRENHEIT
      Return value * ImperialDutRatio(dut) - 459.67
    Case Autodesk.Revit.DB.DisplayUnitType.DUT_CELSIUS
      Return value - 273.15
    Case Else
      Dim newvalue As Double
      newvalue = value * ImperialDutRatio(dut)
      Return newvalue
  End Select
End Function

Public Shared Function ConvertToAPI(ByVal value As Double, ByVal dut As Autodesk.Revit.DB.DisplayUnitT
ype) As Double
  Select Case dut
    Case Autodesk.Revit.DB.DisplayUnitType.DUT_FAHRENHEIT
      Return (value + 459.67) / ImperialDutRatio(dut)
    Case Autodesk.Revit.DB.DisplayUnitType.DUT_CELSIUS
      Return value + 273.15
```

```
        Case Else
            Dim newvalue As Double
            newvalue = value / ImperialDutRatio(dut)
            Return newvalue
    End Select
End Function

Private Shared Function ImperialDutRatio(ByVal dut As Autodesk.Revit.DB.DisplayUnitType) As Double
    Select Case dut
        Case Autodesk.Revit.DB.DisplayUnitType.DUT_ACRES
            Return 0.0000229568411386593
        Case Autodesk.Revit.DB.DisplayUnitType.DUT_AMPERES
            Return 1
        Case Autodesk.Revit.DB.DisplayUnitType.DUT_ATMOSPHERES
            Return 0.0000323793722675857
        Case Autodesk.Revit.DB.DisplayUnitType.DUT_BARS
            Return 0.0000328083989501312
        Case Autodesk.Revit.DB.DisplayUnitType.DUT_BRITISH_THERMAL_UNITS
            Return 0.0000880550918411529
        Case Autodesk.Revit.DB.DisplayUnitType.DUT_BRITISH_THERMAL_UNITS_PER_HOUR
            Return 0.316998330628151
        Case Autodesk.Revit.DB.DisplayUnitType.DUT_BRITISH_THERMAL_UNITS_PER_SECOND
            Return 0.0000880550918411529
        Case Autodesk.Revit.DB.DisplayUnitType.DUT_CALORIES
            Return 0.0221895098882201
        Case Autodesk.Revit.DB.DisplayUnitType.DUT_CALORIES_PER_SECOND
            Return 0.0221895098882201
        Case Autodesk.Revit.DB.DisplayUnitType.DUT_CANDELAS
            Return 1
        Case Autodesk.Revit.DB.DisplayUnitType.DUT_CANDELAS_PER_SQUARE_METER
            Return 10.7639104167097
        Case Autodesk.Revit.DB.DisplayUnitType.DUT_CANDLEPOWER
            Return 1
        Case Autodesk.Revit.DB.DisplayUnitType.DUT_CELSIUS
            Return 1
        Case Autodesk.Revit.DB.DisplayUnitType.DUT_CENTIMETERS
            Return 30.48
        Case Autodesk.Revit.DB.DisplayUnitType.DUT_CENTIMETERS_PER_MINUTE
            Return 1828.8
        Case Autodesk.Revit.DB.DisplayUnitType.DUT_CENTIPOISES
            Return 3280.83989501312
        Case Autodesk.Revit.DB.DisplayUnitType.DUT_CUBIC_CENTIMETERS
            Return 28316.846592
        Case Autodesk.Revit.DB.DisplayUnitType.DUT_CUBIC_FEET
            Return 1
        Case Autodesk.Revit.DB.DisplayUnitType.DUT_CUBIC_FEET_PER_KIP
            Return 14593.9029372064
        Case Autodesk.Revit.DB.DisplayUnitType.DUT_CUBIC_FEET_PER_MINUTE
            Return 60
        Case Autodesk.Revit.DB.DisplayUnitType.DUT_CUBIC_INCHES
            Return 1728
        Case Autodesk.Revit.DB.DisplayUnitType.DUT_CUBIC_METERS
            Return 0.028316846592
        Case Autodesk.Revit.DB.DisplayUnitType.DUT_CUBIC_METERS_PER_HOUR
            Return 101.9406477312
        Case Autodesk.Revit.DB.DisplayUnitType.DUT_CUBIC_METERS_PER_KILONEWTON
            Return 92.90304
        Case Autodesk.Revit.DB.DisplayUnitType.DUT_CUBIC_METERS_PER_SECOND
```

```
    Return 0.028316846592
Case Autodesk.Revit.DB.DisplayUnitType.DUT_CUBIC_MILLIMETERS
    Return 28316846.592
Case Autodesk.Revit.DB.DisplayUnitType.DUT_CUBIC_YARDS
    Return 0.037037037037037
Case Autodesk.Revit.DB.DisplayUnitType.DUT_CYCLES_PER_SECOND
    Return 1
Case Autodesk.Revit.DB.DisplayUnitType.DUT_DECANEWTONS
    Return 0.03048
Case Autodesk.Revit.DB.DisplayUnitType.DUT_DECANEWTONS_PER_METER
    Return 0.1
Case Autodesk.Revit.DB.DisplayUnitType.DUT_DECANEWTONS_PER_SQUARE_METER
    Return 0.328083989501312
Case Autodesk.Revit.DB.DisplayUnitType.DUT_DECANEWTON_METERS
    Return 0.009290304
Case Autodesk.Revit.DB.DisplayUnitType.DUT_DECANEWTON_METERS_PER_METER
    Return 0.03048
Case Autodesk.Revit.DB.DisplayUnitType.DUT_DECIMAL_DEGREES
    Return 57.2957795130823
Case Autodesk.Revit.DB.DisplayUnitType.DUT_DECIMAL_FEET
    Return 1
Case Autodesk.Revit.DB.DisplayUnitType.DUT_DECIMAL_INCHES
    Return 12
Case Autodesk.Revit.DB.DisplayUnitType.DUT_DEGREES_AND_MINUTES
    Return 57.2957795130823
Case Autodesk.Revit.DB.DisplayUnitType.DUT_FAHRENHEIT
    Return 1.8
Case Autodesk.Revit.DB.DisplayUnitType.DUT_FEET_FRACTIONAL_INCHES
    Return 1
Case Autodesk.Revit.DB.DisplayUnitType.DUT_FEET_OF_WATER
    Return 0.00109764531546318
Case Autodesk.Revit.DB.DisplayUnitType.DUT_FEET_OF_WATER_PER_100FT
    Return 0.109761336731934
Case Autodesk.Revit.DB.DisplayUnitType.DUT_FEET_PER_KIP
    Return 14593.9029372064
Case Autodesk.Revit.DB.DisplayUnitType.DUT_FEET_PER_MINUTE
    Return 60
Case Autodesk.Revit.DB.DisplayUnitType.DUT_FEET_PER_SECOND
    Return 1
Case Autodesk.Revit.DB.DisplayUnitType.DUT_FIXED
    Return 1
Case Autodesk.Revit.DB.DisplayUnitType.DUT_FOOTCANDLES
    Return 1.0000000387136
Case Autodesk.Revit.DB.DisplayUnitType.DUT_FOOTLAMBERTS
    Return 3.1415927449471
Case Autodesk.Revit.DB.DisplayUnitType.DUT_FRACTIONAL_INCHES
    Return 12
Case Autodesk.Revit.DB.DisplayUnitType.DUT_GALLONS_US
    Return 7.48051905367236
Case Autodesk.Revit.DB.DisplayUnitType.DUT_GALLONS_US_PER_HOUR
    Return 26929.8685932205
Case Autodesk.Revit.DB.DisplayUnitType.DUT_GALLONS_US_PER_MINUTE
    Return 448.831143220342
Case Autodesk.Revit.DB.DisplayUnitType.DUT_GENERAL
    Return 1
Case Autodesk.Revit.DB.DisplayUnitType.DUT_HECTARES
    Return 0.000009290304
Case Autodesk.Revit.DB.DisplayUnitType.DUT_HERTZ
```

```
        Return 1
    Case Autodesk.Revit.DB.DisplayUnitType.DUT_HORSEPOWER
        Return 0.00012458502883053
    Case Autodesk.Revit.DB.DisplayUnitType.DUT_INCHES_OF_MERCURY
        Return 0.000968831370233344
    Case Autodesk.Revit.DB.DisplayUnitType.DUT_INCHES_OF_WATER
        Return 0.0131845358262865
    Case Autodesk.Revit.DB.DisplayUnitType.DUT_INCHES_OF_WATER_PER_100FT
        Return 1.31845358262865
    Case Autodesk.Revit.DB.DisplayUnitType.DUT_INV_CELSIUS
        Return 1
    Case Autodesk.Revit.DB.DisplayUnitType.DUT_INV_FAHRENHEIT
        Return 0.555555555555556
    Case Autodesk.Revit.DB.DisplayUnitType.DUT_INV_KILONEWTONS
        Return 3280.83989501312
    Case Autodesk.Revit.DB.DisplayUnitType.DUT_INV_KIPS
        Return 14593.9029372064
    Case Autodesk.Revit.DB.DisplayUnitType.DUT_JOULES
        Return 0.09290304
    Case Autodesk.Revit.DB.DisplayUnitType.DUT_KELVIN
        Return 1
    Case Autodesk.Revit.DB.DisplayUnitType.DUT_KILOAMPERES
        Return 0.001
    Case Autodesk.Revit.DB.DisplayUnitType.DUT_KILOCALORIES
        Return 0.0000221895098882201
    Case Autodesk.Revit.DB.DisplayUnitType.DUT_KILOCALORIES_PER_SECOND
        Return 0.0000221895098882201
    Case Autodesk.Revit.DB.DisplayUnitType.DUT_KILOGRAMS_FORCE
        Return 0.0310810655372411
    Case Autodesk.Revit.DB.DisplayUnitType.DUT_KILOGRAMS_FORCE_PER_METER
        Return 0.101971999794098
    Case Autodesk.Revit.DB.DisplayUnitType.DUT_KILOGRAMS_FORCE_PER_SQUARE_METER
        Return 0.334553805098747
    Case Autodesk.Revit.DB.DisplayUnitType.DUT_KILOGRAMS_PER_CUBIC_METER
        Return 35.3146667214886
    Case Autodesk.Revit.DB.DisplayUnitType.DUT_KILOGRAM_FORCE_METERS
        Return 0.00947350877575109
    Case Autodesk.Revit.DB.DisplayUnitType.DUT_KILOGRAM_FORCE_METERS_PER_METER
        Return 0.0310810655372411
    Case Autodesk.Revit.DB.DisplayUnitType.DUT_KILONEWTONS
        Return 0.0003048
    Case Autodesk.Revit.DB.DisplayUnitType.DUT_KILONEWTONS_PER_CUBIC_METER
        Return 0.0107639104167097
    Case Autodesk.Revit.DB.DisplayUnitType.DUT_KILONEWTONS_PER_METER
        Return 0.001
    Case Autodesk.Revit.DB.DisplayUnitType.DUT_KILONEWTONS_PER_SQUARE_METER
        Return 0.00328083989501312
    Case Autodesk.Revit.DB.DisplayUnitType.DUT_KILONEWTON_METERS
        Return 0.00009290304
    Case Autodesk.Revit.DB.DisplayUnitType.DUT_KILONEWTON_METERS_PER_DEGREE
        Return 0.00009290304
    Case Autodesk.Revit.DB.DisplayUnitType.DUT_KILONEWTON_METERS_PER_DEGREE_PER_MET
ER
        Return 0.0003048
    Case Autodesk.Revit.DB.DisplayUnitType.DUT_KILONEWTON_METERS_PER_METER
        Return 0.0003048
    Case Autodesk.Revit.DB.DisplayUnitType.DUT_KILOPASCALS
        Return 0.00328083989501312
```

```
Case Autodesk.Revit.DB.DisplayUnitType.DUT_KILOVOLTS
  Return 0.00009290304
Case Autodesk.Revit.DB.DisplayUnitType.DUT_KILOVOLT_AMPERES
  Return 0.00009290304
Case Autodesk.Revit.DB.DisplayUnitType.DUT_KILOWATTS
  Return 0.00009290304
Case Autodesk.Revit.DB.DisplayUnitType.DUT_KILOWATT_HOURS
  Return 0.0000000258064
Case Autodesk.Revit.DB.DisplayUnitType.DUT_KIPS
  Return 0.224808943099711
Case Autodesk.Revit.DB.DisplayUnitType.DUT_KIPS_PER_CUBIC_FOOT
  Return 0.0000685217658567918
Case Autodesk.Revit.DB.DisplayUnitType.DUT_KIPS_PER_CUBIC_INCH
  Return 0.000000396537996856434
Case Autodesk.Revit.DB.DisplayUnitType.DUT_KIPS_PER_FOOT
  Return 0.0000685217658567918
Case Autodesk.Revit.DB.DisplayUnitType.DUT_KIPS_PER_INCH
  Return 0.00000571014715473265
Case Autodesk.Revit.DB.DisplayUnitType.DUT_KIPS_PER_SQUARE_FOOT
  Return 0.0000685217658567918
Case Autodesk.Revit.DB.DisplayUnitType.DUT_KIPS_PER_SQUARE_INCH
  Return 0.00000475845596227721
Case Autodesk.Revit.DB.DisplayUnitType.DUT_KIP_FEET
  Return 0.0000685217658567918
Case Autodesk.Revit.DB.DisplayUnitType.DUT_KIP_FEET_PER_DEGREE
  Return 0.0000685217658567918
Case Autodesk.Revit.DB.DisplayUnitType.DUT_KIP_FEET_PER_DEGREE_PER_FOOT
  Return 0.0000208854342331501
Case Autodesk.Revit.DB.DisplayUnitType.DUT_KIP_FEET_PER_FOOT
  Return 0.0000685217658567918
Case Autodesk.Revit.DB.DisplayUnitType.DUT_LITERS
  Return 28.316846592
Case Autodesk.Revit.DB.DisplayUnitType.DUT_LITERS_PER_SECOND
  Return 28.316846592
Case Autodesk.Revit.DB.DisplayUnitType.DUT_LUMENS
  Return 1
Case Autodesk.Revit.DB.DisplayUnitType.DUT_LUX
  Return 10.7639104167097
Case Autodesk.Revit.DB.DisplayUnitType.DUT_MEGANEWTONS
  Return 0.0000003048
Case Autodesk.Revit.DB.DisplayUnitType.DUT_MEGANEWTONS_PER_METER
  Return 0.000001
Case Autodesk.Revit.DB.DisplayUnitType.DUT_MEGANEWTONS_PER_SQUARE_METER
  Return 0.00000328083989501312
Case Autodesk.Revit.DB.DisplayUnitType.DUT_MEGANEWTON_METERS
  Return 0.00000009290304
Case Autodesk.Revit.DB.DisplayUnitType.DUT_MEGANEWTON_METERS_PER_METER
  Return 0.0000003048
Case Autodesk.Revit.DB.DisplayUnitType.DUT_MEGAPASCALS
  Return 0.00000328083989501312
Case Autodesk.Revit.DB.DisplayUnitType.DUT_METERS
  Return 0.3048
Case Autodesk.Revit.DB.DisplayUnitType.DUT_METERS_CENTIMETERS
  Return 0.3048
Case Autodesk.Revit.DB.DisplayUnitType.DUT_METERS_PER_KILONEWTON
  Return 1000
Case Autodesk.Revit.DB.DisplayUnitType.DUT_METERS_PER_SECOND
  Return 0.3048
```

```
Case Autodesk.Revit.DB.DisplayUnitType.DUT_MILLIAMPERES
  Return 1000
Case Autodesk.Revit.DB.DisplayUnitType.DUT_MILLIMETERS
  Return 304.8
Case Autodesk.Revit.DB.DisplayUnitType.DUT_MILLIMETERS_OF_MERCURY
  Return 0.0246083170946002
Case Autodesk.Revit.DB.DisplayUnitType.DUT_MILLIVOLTS
  Return 92.90304
Case Autodesk.Revit.DB.DisplayUnitType.DUT_NEWTONS
  Return 0.3048
Case Autodesk.Revit.DB.DisplayUnitType.DUT_NEWTONS_PER_METER
  Return 1
Case Autodesk.Revit.DB.DisplayUnitType.DUT_NEWTONS_PER_SQUARE_METER
  Return 3.28083989501312
Case Autodesk.Revit.DB.DisplayUnitType.DUT_NEWTON_METERS
  Return 0.09290304
Case Autodesk.Revit.DB.DisplayUnitType.DUT_NEWTON_METERS_PER_METER
  Return 0.3048
Case Autodesk.Revit.DB.DisplayUnitType.DUT_PASCALS
  Return 3.28083989501312
Case Autodesk.Revit.DB.DisplayUnitType.DUT_PASCALS_PER_METER
  Return 10.7639104167097
Case Autodesk.Revit.DB.DisplayUnitType.DUT_PASCAL_SECONDS
  Return 3.28083989501312
Case Autodesk.Revit.DB.DisplayUnitType.DUT_PERCENTAGE
  Return 100
Case Autodesk.Revit.DB.DisplayUnitType.DUT_POUNDS_FORCE
  Return 224.80894309971
Case Autodesk.Revit.DB.DisplayUnitType.DUT_POUNDS_FORCE_PER_CUBIC_FOOT
  Return 0.0685217658567918
Case Autodesk.Revit.DB.DisplayUnitType.DUT_POUNDS_FORCE_PER_FOOT
  Return 0.0685217658567918
Case Autodesk.Revit.DB.DisplayUnitType.DUT_POUNDS_FORCE_PER_SQUARE_FOOT
  Return 0.0685217658567917
Case Autodesk.Revit.DB.DisplayUnitType.DUT_POUNDS_FORCE_PER_SQUARE_INCH
  Return 0.000475845616460903
Case Autodesk.Revit.DB.DisplayUnitType.DUT_POUNDS_MASS_PER_CUBIC_FOOT
  Return 2.20462262184878
Case Autodesk.Revit.DB.DisplayUnitType.DUT_POUNDS_MASS_PER_CUBIC_INCH
  Return 0.00127582327653286
Case Autodesk.Revit.DB.DisplayUnitType.DUT_POUNDS_MASS_PER_FOOT_HOUR
  Return 7936.64143865559
Case Autodesk.Revit.DB.DisplayUnitType.DUT_POUNDS_MASS_PER_FOOT_SECOND
  Return 2.20462262184878
Case Autodesk.Revit.DB.DisplayUnitType.DUT_POUND_FORCE_FEET
  Return 0.0685217658567918
Case Autodesk.Revit.DB.DisplayUnitType.DUT_POUND_FORCE_FEET_PER_FOOT
  Return 0.0685217658567918
Case Autodesk.Revit.DB.DisplayUnitType.DUT_RANKINE
  Return 1.8
Case Autodesk.Revit.DB.DisplayUnitType.DUT_SQUARE_CENTIMETERS
  Return 929.0304
Case Autodesk.Revit.DB.DisplayUnitType.DUT_SQUARE_FEET
  Return 1
Case Autodesk.Revit.DB.DisplayUnitType.DUT_SQUARE_FEET_PER_KIP
  Return 14593.9029372064
Case Autodesk.Revit.DB.DisplayUnitType.DUT_SQUARE_INCHES
  Return 144
```

```
        Case Autodesk.Revit.DB.DisplayUnitType.DUT_SQUARE_METERS
            Return 0.09290304
        Case Autodesk.Revit.DB.DisplayUnitType.DUT_SQUARE_METERS_PER_KILONEWTON
            Return 304.8
        Case Autodesk.Revit.DB.DisplayUnitType.DUT_SQUARE_MILLIMETERS
            Return 92903.04
        Case Autodesk.Revit.DB.DisplayUnitType.DUT_THERMS
            Return 0.00000000088054745701663
        Case Autodesk.Revit.DB.DisplayUnitType.DUT_TONNES_FORCE
            Return 0.0000310810655372411
        Case Autodesk.Revit.DB.DisplayUnitType.DUT_TONNES_FORCE_PER_METER
            Return 0.000101971999794098
        Case Autodesk.Revit.DB.DisplayUnitType.DUT_TONNES_FORCE_PER_SQUARE_METER
            Return 0.00033453805098747
        Case Autodesk.Revit.DB.DisplayUnitType.DUT_TONNE_FORCE_METERS
            Return 0.00000947350877575109
        Case Autodesk.Revit.DB.DisplayUnitType.DUT_TONNE_FORCE_METERS_PER_METER
            Return 0.0000310810655372411
        Case Autodesk.Revit.DB.DisplayUnitType.DUT_VOLTS
            Return 0.09290304
        Case Autodesk.Revit.DB.DisplayUnitType.DUT_VOLT_AMPERES
            Return 0.09290304
        Case Autodesk.Revit.DB.DisplayUnitType.DUT_WATTS
            Return 0.09290304
        Case Autodesk.Revit.DB.DisplayUnitType.DUT_WATTS_PER_SQUARE_FOOT
            Return 0.09290304
        Case Autodesk.Revit.DB.DisplayUnitType.DUT_WATTS_PER_SQUARE_METER
            Return 1


        Case Autodesk.Revit.DB.DisplayUnitType.DUT_BRITISH_THERMAL_UNITS_PER_HOUR_CUBIC_FO
OT
            Return 0.31699833062815
        Case Autodesk.Revit.DB.DisplayUnitType.DUT_BRITISH_THERMAL_UNITS_PER_HOUR_SQUARE_
FOOT
            Return 0.316998330628151
        Case Autodesk.Revit.DB.DisplayUnitType.DUT_BRITISH_THERMAL_UNITS_PER_HOUR_SQUARE_
FOOT_FAHRENHEIT
            Return 0.176110194261872
        Case Autodesk.Revit.DB.DisplayUnitType.DUT_CUBIC_FEET_PER_MINUTE_CUBIC_FOOT
            Return 60
        Case Autodesk.Revit.DB.DisplayUnitType.DUT_CUBIC_FEET_PER_MINUTE_SQUARE_FOOT
            Return 60
        Case Autodesk.Revit.DB.DisplayUnitType.DUT_CUBIC_FEET_PER_MINUTE_TON_OF_REFRIGERA
TION
            Return 2271305.33644539
        Case Autodesk.Revit.DB.DisplayUnitType.DUT_CURRENCY
            Return 1
        Case Autodesk.Revit.DB.DisplayUnitType.DUT_LITERS_PER_SECOND_CUBIC_METER
            Return 1000
        Case Autodesk.Revit.DB.DisplayUnitType.DUT_LITERS_PER_SECOND_KILOWATTS
            Return 304800
        Case Autodesk.Revit.DB.DisplayUnitType.DUT_LITERS_PER_SECOND_SQUARE_METER
            Return 304.8
        Case Autodesk.Revit.DB.DisplayUnitType.DUT_LUMENS_PER_WATT
            Return 10.7639104167097
        Case Autodesk.Revit.DB.DisplayUnitType.DUT_RATIO_10
            Return 10
```

```
        Case Autodesk.Revit.DB.DisplayUnitType.DUT_RATIO_12
            Return 12
        Case Autodesk.Revit.DB.DisplayUnitType.DUT_RISE_OVER_FOOT
            Return 1
        Case Autodesk.Revit.DB.DisplayUnitType.DUT_RISE_OVER_INCHES
            Return 12
        Case Autodesk.Revit.DB.DisplayUnitType.DUT_RISE_OVER_MMS
            Return 1000
        Case Autodesk.Revit.DB.DisplayUnitType.DUT_SLOPE_DEGREES
            Return 57.2957795130824
        Case Autodesk.Revit.DB.DisplayUnitType.DUT_SQUARE_FEET_PER_TON_OF_REFRIGERATION
            Return 37855.0889407566
        Case Autodesk.Revit.DB.DisplayUnitType.DUT_SQUARE_METERS_PER_KILOWATTS
            Return 1000
        Case Autodesk.Revit.DB.DisplayUnitType.DUT_TON_OF_REFRIGERATION
            Return 0.0000264165275523459
        Case Autodesk.Revit.DB.DisplayUnitType.DUT_WATTS_PER_CUBIC_FOOT
            Return 0.09290304
        Case Autodesk.Revit.DB.DisplayUnitType.DUT_WATTS_PER_CUBIC_METER
            Return 3.28083989501312
        Case Autodesk.Revit.DB.DisplayUnitType.DUT_WATTS_PER_SQUARE_METER_KELVIN
            Return 1
        Case Else
            Return 1
    End Select
End Function
```

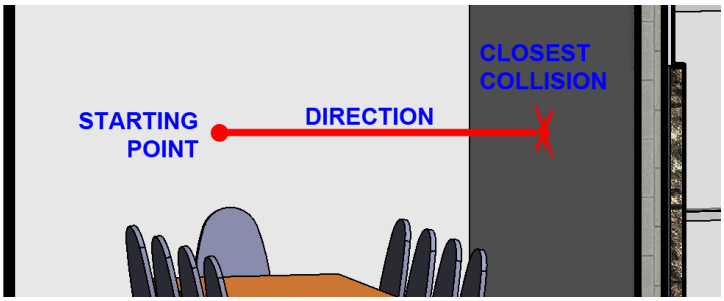
API TO QUERY LINKED MODEL/RAY TRACE ('ray casting')

The following sections include the key lines of code to accomplish the specified task. All code given is in the VB.NET language but can easily be automatically converted to other programming languages. In these sections, some basic variable declarations and programming syntax have been omitted for conciseness, and it is expected that the reader has an understanding of programming fundamentals or can obtain basic documentation from other sources. Also, some basic Revit API functions (like filtering, parameter creation, etc.) that are well documented in the software development kit (SDK) material are not discussed in detail here, and I would encourage readers to use the SDK documentation as a resource for these types of functions.

Find Linked Documents from Document List

```
Dim ListOfLinkedDocs As New Autodesk.Revit.DB.DocumentSet
Dim currentLink As Element
Dim LinkCollector As New FilteredElementCollector(MyDoc)
Dim LinkedElems As IList(Of Element) =
LinkCollector.OfCategory(BuiltInCategory.OST_RvtLinks).OfClass(GetType(RevitLinkType)).To
Elements()
For Each currentLink In LinkedElems
   Dim linkType As RevitLinkType = TryCast(currentLink, RevitLinkType)
   For Each CurrentDoc As Document In MyApp.Application.Documents
      If CurrentDoc.Title.Equals(linkType.Name) Then
         ListOfLinkedDocs.Insert(CurrentDoc)
         Exit For
      End If
   Next
Next
```

Ray trace to find location of intersection




```
'[SET UP A SOURCE POINT AND A DIRECTION]
Dim FoundCollisions As System.Collections.Ilist
FoundCollisions = MyLinkedDoc.FindReferencesWithContextByDirection(OriginalLocation,
RayDirection, My3Dview)
```

```
‘NOTE: REFERENCEINTERSECTOR CLASS IN 2013 API
‘[FILTER COLLISIONS BY CATEGORY, THEN FIND CLOSEST]
For j = 0 To FoundCollisions.Count - 1
   Dim tempReferenceWithContext As ReferenceWithContext = TryCast(FoundCollisions
.Item(j), ReferenceWithContext)
   Dim TempReference As Reference= tempReferenceWithContext.GetReference
   NewLocation = TempReference.GlobalPoint
…
```

Perimeter of room/space

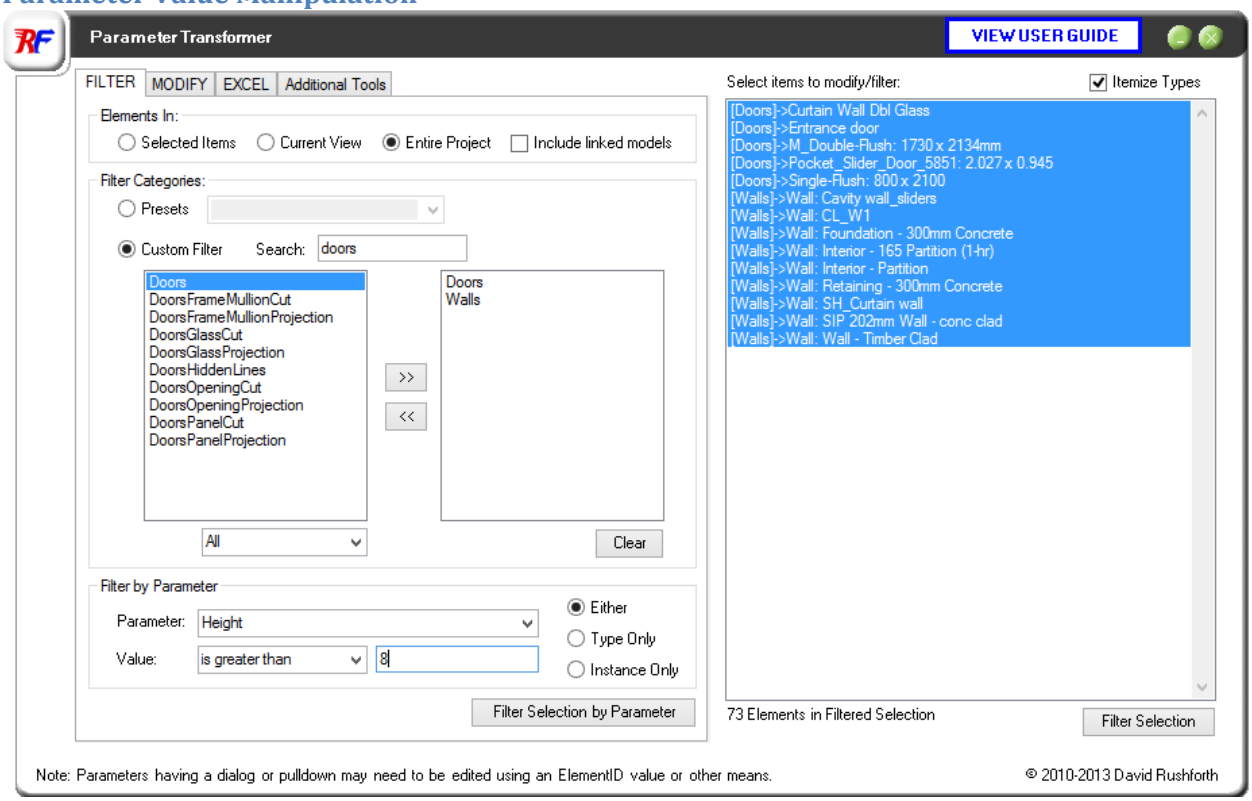
```
Dim TempBoundaryOptions As New SpatialElementBoundaryOptions
myBoundaryArray = TempSpace.GetBoundarySegments(TempBoundaryOptions)
```

Create element on wall vs. linked model

```
If MyDoc.PathName = MyLinkedDoc.PathName Then ‘In this model
   fi = MyDoc.Create.NewFamilyInstance(TempReference , NewLocation,
MyReferenceDirection, MyFamilySymbol) ‘MyReferenceDirection = MyNormalRotated90
Else 'IF LINKED MODEL: CREATE REFERENCE PLANE
   Dim MyReferencePlane As ReferencePlane
   Dim PlaneFreeEnd As XYZ = NewLocation - MyReferenceDirection
   Dim PlaneCutVector As XYZ = XYZ.BasisZ
   MyReferencePlane = MyDoc.Create.NewReferencePlane(NewLocation, PlaneFreeEnd,
PlaneCutVector, MyDoc.ActiveView)
   fi = MyDoc.Create.NewFamilyInstance(MyReferencePlane.Reference, NewLocation,
MyReferenceDirection, MyFamilySymbol)
   MyDoc.Delete(MyReferencePlane.Id)
End if
```

MODEL MANIPULATION AND DATA IMPORT/EXPORT

Parameter Value Manipulation



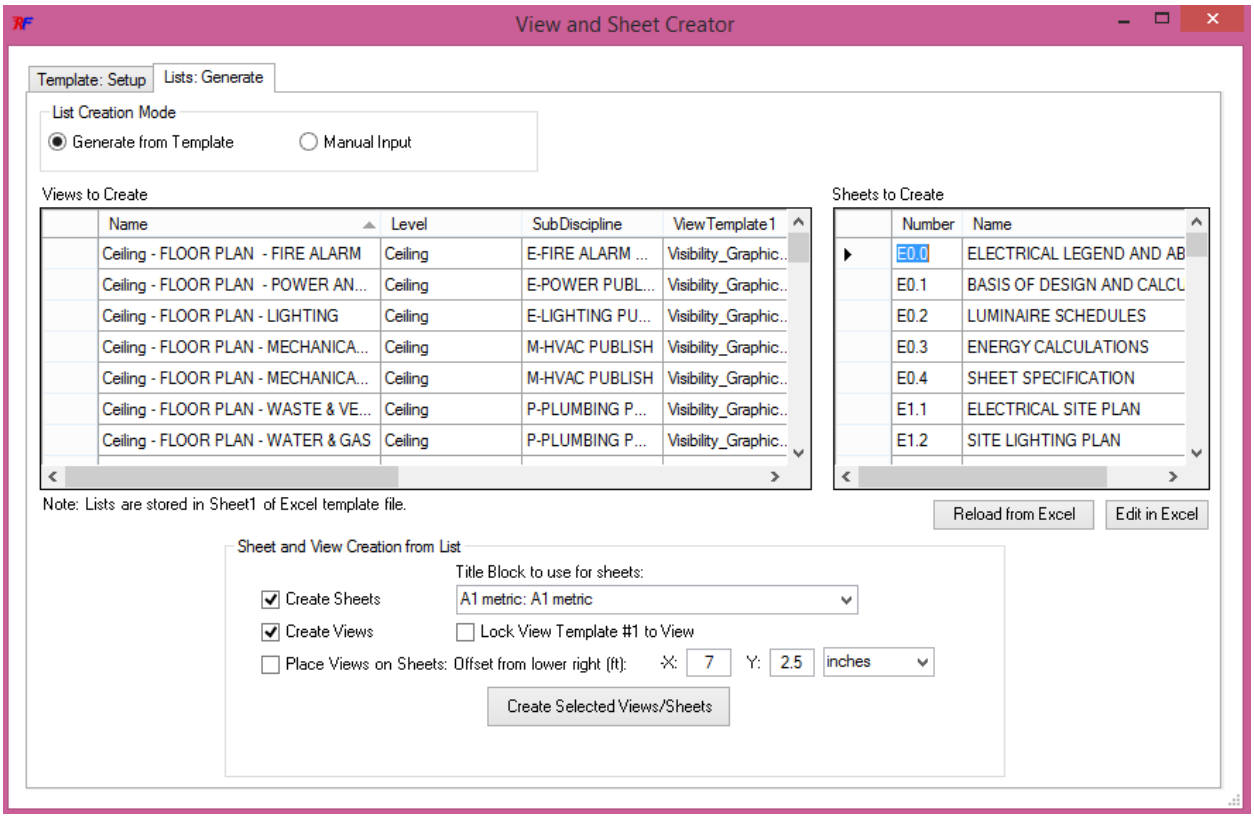
Get and set parameter values ('for each' loop)

```
currentTypeParam.Set(ParamValueToSet)
Select Case ParameterToReturn.StorageType
   '[Remember ConvertFromAPI and ConvertToAPI]
   CurrentParamValue = ParameterToReturn.AsString()
```

Element filters

```
Dim OtherElementCollector As New Autodesk.Revit.DB.FilteredElementCollector(MyDoc)
Dim MyOtherFilter As New ElementCategoryFilter(BuiltInCategory.OST_ElectricalCircuit)
Dim MyOrFilter As New ElementCategoryFilter(BuiltInCategory.OST_ElectricalInternalCircuits)
Dim NewPreCombinedFilter As New LogicalOrFilter(MyOtherFilter, MyOrFilter)
Dim MyInvertedFilter As New ElementClassFilter(GetType(Autodesk.Revit.DB.ElementType),
True) 'don't include element types
Dim NewCombinedFilter As New LogicalAndFilter(NewPreCombinedFilter, MyInvertedFilter)
'MULTI-STAGE FILTER EXAMPLE
FinalElementResult = OtherElementCollector.WherePasses(NewCombinedFilter).ToElements
```

Create sheet




```
Dim CurrentSheetToCreate As ViewSheet = MyDoc.Create.NewViewSheet(TitleBlockForSheet)
‘ITERATE TITLBLOCKS PRIOR
CurrentSheetToCreate.Name = NewSheetToCreateName
CurrentSheetToCreate.SheetNumber = NewSheetToCreateNumber
```

Create view

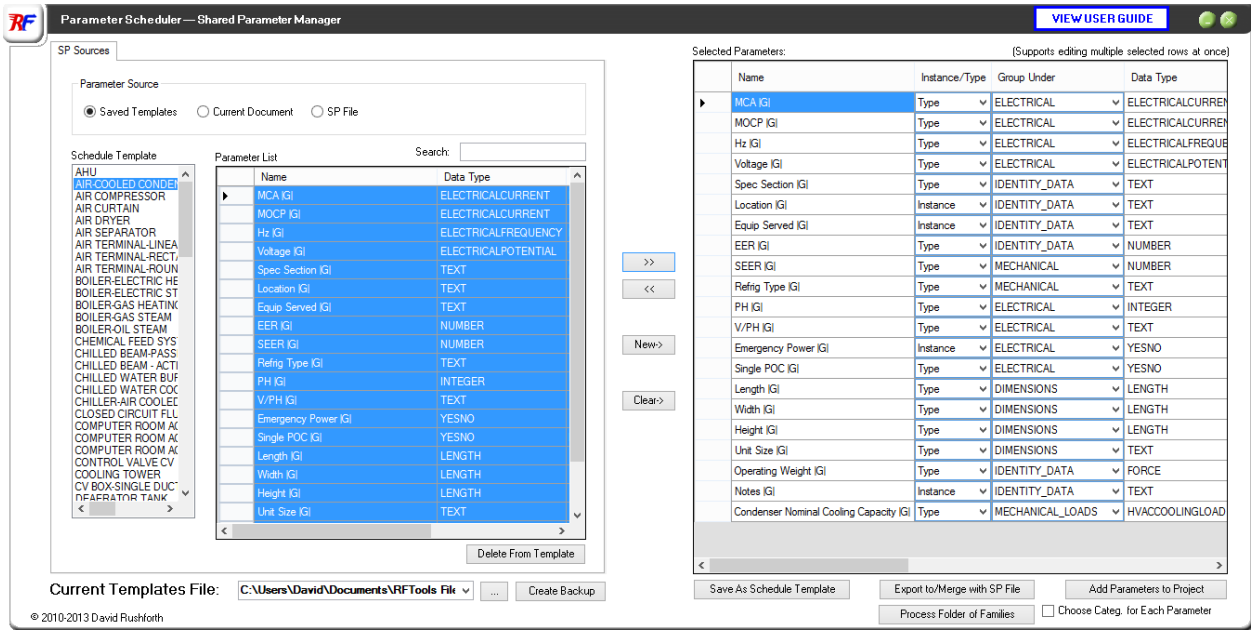
```
CurrentViewToCreate = ViewPlan.Create(MyDoc, CurrentViewType.Id, CurrentLevel.Id)
CurrentViewToCreate.Name = CurrentViewName ‘ITERATE LEVELS PRIOR
```

Apply View Template

```
If TypeOf elem Is Autodesk.Revit.DB.View Then ‘ITERATE VIEWS AND CHECK FOR NAME
MATCHING TEMPLATE
        Dim MyTempview As Autodesk.Revit.DB.View = TryCast(elem,
Autodesk.Revit.DB.View)
        If MyTempview.IsTemplate() = True and MyTempview.Name = TemplateName Then
            CurrentViewTemplate = MyTempview
        End If
End If
If LockToView = True Then
   CurrentViewToProcess.ViewTemplateId = CurrentViewTemplate.Id
```

```
Else
   CurrentViewToProcess.ApplyViewTemplateParameters(CurrentViewTemplate)
End If
```

Create Parameters in the Family Editor




```
For Each group As DefinitionGroup In mysharedFile.Groups
   For Each SharedParamDef As ExternalDefinition In group.Definitions
      If SharedParamDef.Name = ParametersToCreate(i).ParameterToCreateName Then
      '[DETERMINE GROUP AND WHETHER INSTANCE OR TYPE]
         MyFamilyDoc.FamilyManager.AddParameter(SharedParamDef, MyParamGroup,
IsInstanceParameter)
      End If
   Next
Next
```

Excel Connection (refer to 2010 class "Opening the Lines of Communication between Revit and Third-Party Applications" for Access, SQL, Excel, Text, etc.)

```
Dim strXlsFile As String
strXlsFile = "C:\Temp\test.xls"
Dim mExcelApplication As New Microsoft.Office.Interop.Excel.Application
Dim mExcelWorkbook As Microsoft.Office.Interop.Excel.Workbook =
mExcelApplication.Workbooks.Open(strXlsFile)
Dim mExcelWorksheet As Microsoft.Office.Interop.Excel.Worksheet =
mExcelWorkbook.Worksheets(1)
Dim readValue As String
readValue = mExcelWorksheet.Range("A1").Value
mExcelWorksheet.Range("A2").Value = "New Value"
mExcelWorkbook.Windows(1).Visible = True
```

```
mExcelWorkbook.Save()
mExcelWorkbook = Nothing
mExcelApplication.Quit()
mExcelApplication = Nothing
```

THANK YOU

I hope that the material presented will not only motivate you to consider ways in which using the API can increase your productivity but also enable you to have the understanding and resources to get started today. I am interested in hearing comments and success stories from any that read this document. Feel free to contact me at the email address below with questions or comments.

THANK YOU!

Email me: david@rushforth.org