

SD16637

## Streamlining the Revit Process with Custom Add-ins

Nick Kovach  
BOKA Powell

Stephen Faust  
Revolution Design, Inc.

### Learning Objectives

- Learn how to use the sample code examples and create a starting template
- Determine how to best implement new add-ins throughout the company
- Understand the level of difficulty for specific add-in tasks
- Learn how to decide when a custom add-in is worth building or buying

### Description

How does a Building Information Modeling (BIM) manager get started in the wonderful world of the Revit building design software API? More importantly, how do programmers keep from getting overwhelmed with custom add-in requests from employees at their firms? We will discuss ideas regarding how to get started programming and how to beta test your custom add-ins company wide. We will also investigate how to calculate the difficulty level for coding a specific idea, and when it is time to take your code to the next level. This session features Revit, Revit Architecture, and Revit MEP.

### Your AU Expert(s)

Nick Kovach is the Building Information Modeling (BIM) manager at BOKA Powell, and he has a unique blend of experience in architecture, business, technology, and coding with the Revit software API. As a licensed architect in Texas with an MBA and the founder of the Technology in Architectural Practice Committee at American Institute of Architects in Dallas, Kovach contributes this understanding to all that he does as a BIM manager. He has over 15 years of experience as an architect and technology guru at numerous firms, and he understands how the process works and can provide real world solutions in numerous ways.

Stephen Faust brings a unique blend of experience in architecture, coding, and the Revit API. As a licensed architect and former BIM manager/trainer for several firms he understands the needs and processes of the building industry. With 8 years' experience working in the Revit API (6 commercially) he understands the power of the Revit API and how to create solutions that work for the building industry.



## Setting up your Programming Environment

The most important aspect when getting started with Revit Add-ins is to set up your programming environment. There are plenty of resources available including numerous websites and online help files.

### Starting with the Revit SDK website

On the Autodesk Developer Network website, you will find the Revit SDK (Software Development Kit) available for download. There are also numerous videos and blogs that you can follow along with to learn a bit more about how to get started.

<http://usa.autodesk.com/adsk/servlet/index?siteID=123112&id=2484975>

### Downloading Revit 2017 SDK

The first step I recommend is to download and install the most current Revit SDK (currently 2017). This installation will create a folder that contains Sample source code that you can compile and also a few help files (including RevitAPI.chm).

You can also download the updated Revit SDK here:

-  [Revit 2017 SDK \(Update June 23, 2016\).msi](#) (msi - 300068Kb)
-  [Revit 2016 R2 SDK for Subscription Release \(Update October 30, 2015\)](#) (msi - 257092Kb)
-  [Revit 2016 SDK \(Update April 23, 2015\)](#) (msi - 256700Kb)
-  [Revit 2015 SDK for UR4 \(Update September 15, 2014\)](#) (msi - 242811Kb)
-  [Revit 2015 SDK for Subscription Release \(Update September 15, 2014\)](#) (msi - 243040Kb)

*DOWNLOAD REVIT SDK LINK SCREENSHOT*

We will discuss later how to compile and test the Sample Add-ins after we have completed setting up our programming environment.

### Setting up your IDE (Integrated Development Environment)

Now that you have installed the Revit SDK, you will need to install a software to help you compile the source code and debug any errors in your code. The best option that I recommend is to download and install Visual Studio Express (Microsoft).

<https://www.visualstudio.com/downloads/download-visual-studio-vs>

### Downloading Visual Studio Express

Sometimes this download can be a bit tricky because Microsoft is constantly updating their software and renaming the “free version” of Visual Studio. At this point the key to finding the free version is to look for the “Express” version for Windows Desktop.

This download can take a bit of time (usually it downloads a small file that then connects to the internet to download/install the remainder of the file). Trust me, it is worth it and will simplify the process of learning to code and work with the Revit SDK.



## Downloading RevitLookup from GitHub

Brought to you by Jeremy Tammik (from The Building Coder blog) this download is available on GitHub.

<https://github.com/jeremytammik/RevitLookup>

The easiest procedure is to click on the “Clone or download” button and save the contents to a ZIP file. Then you can extract the files to a new folder and you are ready to compile the files and build the DLL file.

## Installing the Add-in Manager

Using the Add-in Manager is my preferred technique of testing Revit add-ins because it doesn't require you to stop/start Revit every time you wish to tweak the code. Luckily this program is precompiled and comes with the Revit SDK.



*FOLDER LOCATION OF ADD-IN MANAGER*

## The two important files for Revit Add-in use

This may be a bit of an oversimplification but the two main files you need to make your Add-in work in Revit are the DLL (Dynamic Link-Library) file and the ADDIN (Manifest file). The DLL is the compiled file from Visual Studio and the ADDIN file is a text file that contains the instructions that Revit uses to launch this DLL file.

In my opinion, it is easiest if these two files are copied into a directory that Revit has defined for use with Revit add-ins. There are actually a few locations that Revit searches for these files but the best is **C:\ProgramData\Autodesk\Revit\Add-ins\2017**. Keep in mind that ProgramData is a hidden folder.

## Installing the Revit Lookup Add-in

Another very useful tool that will assist you in building a Revit Add-in is the Revit Lookup Tool. This tool can be a bit more complicated to install because, as mentioned, you will need to download the files and compile them (building the DLL) before you can use it. This may seem like a lot of hoops to jump through but this Add-in will prove it's worth very soon.

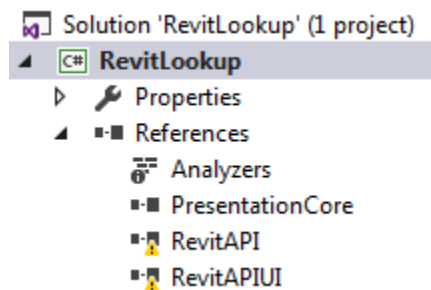


## Compiling the Revit Lookup Add-in in Visual Studio

This will be a similar procedure for how we compile the code samples, so it is important to follow along. Once you have downloaded all the source code from GitHub and placed the files in a folder you are ready to start the process.

The programming language for Revit Lookup is C# (a Microsoft .NET language) and you should find the **RevitLookup.sln** file in the CS folder you just created. Assuming you have installed Visual Studio Express, you can now double click to open this project. Keep in mind that sometimes there will not be a SLN (Microsoft Visual Studio Solution) file and therefore you can use the CSPROJ file (Visual C# Project file).

The Solutions Explorer window in Visual Studio should show you a couple folders but the most important one right now is under References. There are two references that need to be updated (per the Revit version) and we will need to replace both of these.



SCREEN SHOT FROM VISUAL STUDIO

The DLL files that need replacing are RevitAPI.dll and RevitAPIUI.dll. Again, these are Revit version specific, so you will find these files in your specific Revit program folder (For Example: **C:\Program Files\Autodesk\Revit 2017**).

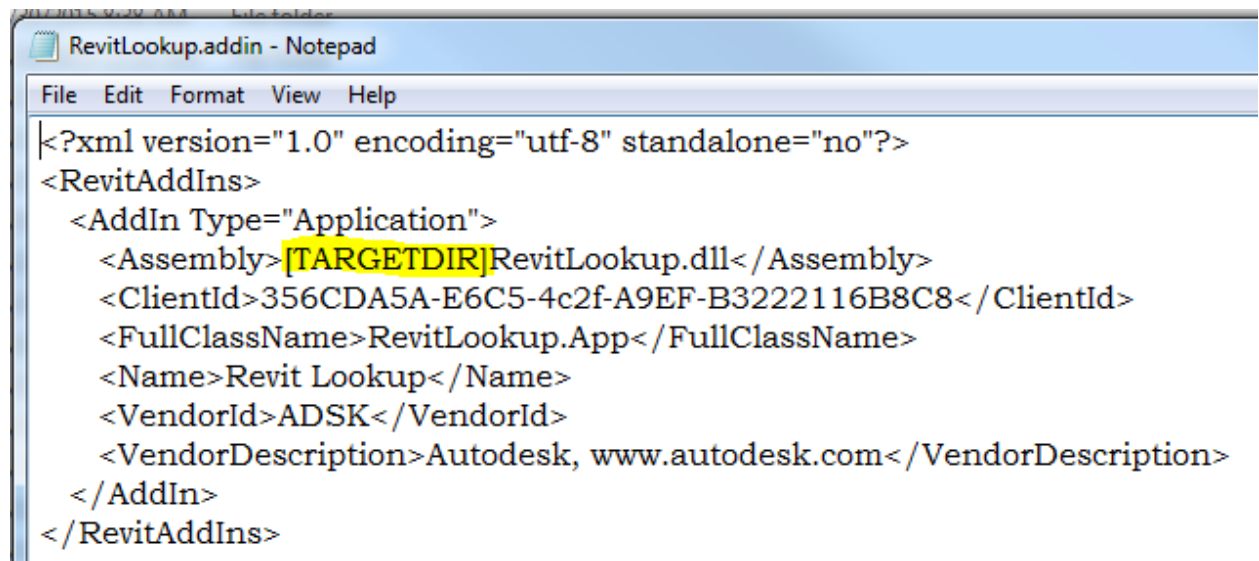
Start by right clicking on RevitAPI and Remove this Reference (so we can replace it with the current 2017 version). Now you can Right click on Reference and choose “Add Reference...” to add it back. Go ahead and click on the “Browse...” button and go to the Revit Program directory suggested above. Now if you select the RevitAPI.dll file and click the Add button, you should see this reference appear (without the Yellow Caution Delta icon). Do this with RevitAPIUI also to replace this one also.

Now you are ready to compile the source code by pressing **F7** or Selecting the Build Tab and clicking **Build Solution**. Give it a second and try not to worry too much if there are a couple warnings (Errors are a bit more worrisome but they typically describe the issue that caused the error in most cases).

Assuming you have successfully compiled the program, by default you should find the compiled DLL in the bin / Debug (or Release) folder. Go ahead and copy that DLL to the ProgramData folder that I mentioned when installing the Add-in Manager. You should find the ADDIN file in the original folder we started in. You can copy this into the ProgramData folder also but keep in mind we will need to edit this file before it will work.



The final step in getting RevitLookup to work is to edit the RevitLookup.addin (manifest).



```
<?xml version="1.0" encoding="utf-8" standalone="no"?>
<RevitAddIns>
  <AddIn Type="Application">
    <Assembly>[TARGETDIR]RevitLookup.dll</Assembly>
    <ClientId>356CDA5A-E6C5-4c2f-A9EF-B3222116B8C8</ClientId>
    <FullClassName>RevitLookup.App</FullClassName>
    <Name>Revit Lookup</Name>
    <VendorId>ADSK</VendorId>
    <VendorDescription>Autodesk, www.autodesk.com</VendorDescription>
  </AddIn>
</RevitAddIns>
```

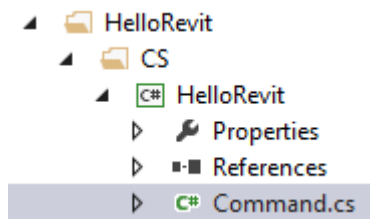
*MANIFEST FILE FOR REVITLOOKUP*

It is not necessary to place the .addin file in the same directory as the .dll file but I typically keep them together for simplicity sake. As a result, typically all you need to do when editing the RevitLookup.addin is remove the [TARGETDIR] and save the .addin file in the ProgramData folder.



## Building a Visual Studio starting Template

Much like Revit, Templates can help simplify the process and get you up and running quickly. In this case, much like my other examples, I find it easier to just start with a pre-built piece of code instead of starting from scratch. For this example, I would recommend copying the entire CS folder from HelloRevit to your desktop.



*SAMPLES.SLN IN SOLUTION EXPLORER*

Go ahead and double click on the copied version of HelloRevit.csproj to look closer at the code. The original code from Command.cs looks something like this:

```
using System;
using System.Collections.Generic;
using System.Text;

using Autodesk.Revit;
using Autodesk.Revit.DB;
using Autodesk.Revit.UI;
using Autodesk.Revit.ApplicationServices;

namespace Revit.SDK.Samples.HelloRevit.CS
{
    /// <summary>
    /// Demonstrate how a basic ExternalCommand can be added to the Revit user interface.
    /// And demonstrate how to create a Revit style dialog.
    /// </summary>
    [Autodesk.Revit.Attributes.Transaction(Autodesk.Revit.Attributes.TransactionMode.Manual)]
    [Autodesk.Revit.Attributes.Regeneration(Autodesk.Revit.Attributes.RegenerationOption.Manual)]
    [Autodesk.Revit.Attributes.Journaling(Autodesk.Revit.Attributes.JournalingMode.NoCommandData)]
    public class Command : IExternalCommand
    {
        [IExternalCommand Members]
    }
}
```

*COMMAND.CS SOURCE CODE*

Notice the main meat of the HelloRevit code is hidden (so click the + symbol to look a little closer). This little trick (#region) helps clean up your code if you want to collapse or expand certain blocks to help organize things. I don't typically use it much myself but you will find it in many of the examples and so you should understand what it does.



After a bit of cleaning (removing comments like `///<summary>`), here are the bones that I plan to use for our starting template:

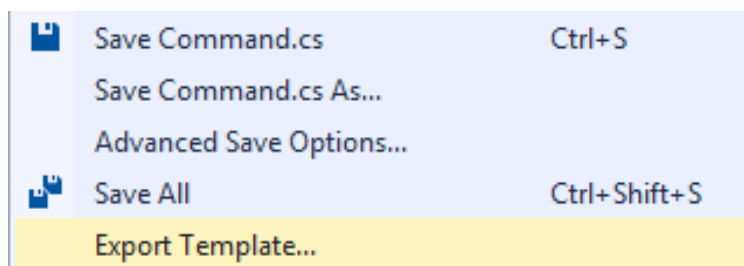
```
Command.cs* X
template.CS.Command Execute(ExternalCommandData)
using System;
using System.Collections.Generic;
using System.Text;

using Autodesk.Revit;
using Autodesk.Revit.DB;
using Autodesk.Revit.UI;
using Autodesk.Revit.ApplicationServices;

namespace template.CS
{
    [Autodesk.Revit.Attributes.Transaction(Autodesk.Revit.Attributes.TransactionMode.Manual)]
    [Autodesk.Revit.Attributes.Regeneration(Autodesk.Revit.Attributes.RegenerationOption.Manual)]
    [Autodesk.Revit.Attributes.Journaling(Autodesk.Revit.Attributes.JournalingMode.NoCommandData)]
    public class Command : IExternalCommand
    {
        public Autodesk.Revit.UI.Result Execute(ExternalCommandData commandData,
            ref string message, Autodesk.Revit.DB.ElementSet elements)
        {
            Application app = commandData.Application.Application;
            Document activeDoc = commandData.Application.ActiveUIDocument.Document;

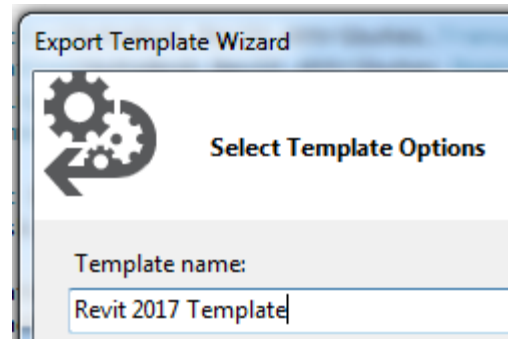
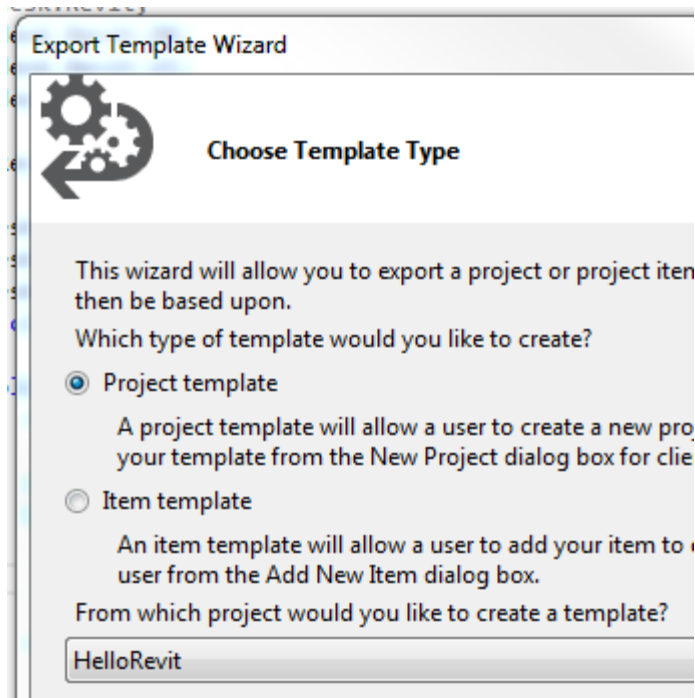
            return Autodesk.Revit.UI.Result.Succeeded;
        }
    }
}
```

COMMAND.CS (CLEANED UP FOR TEMPLATE)

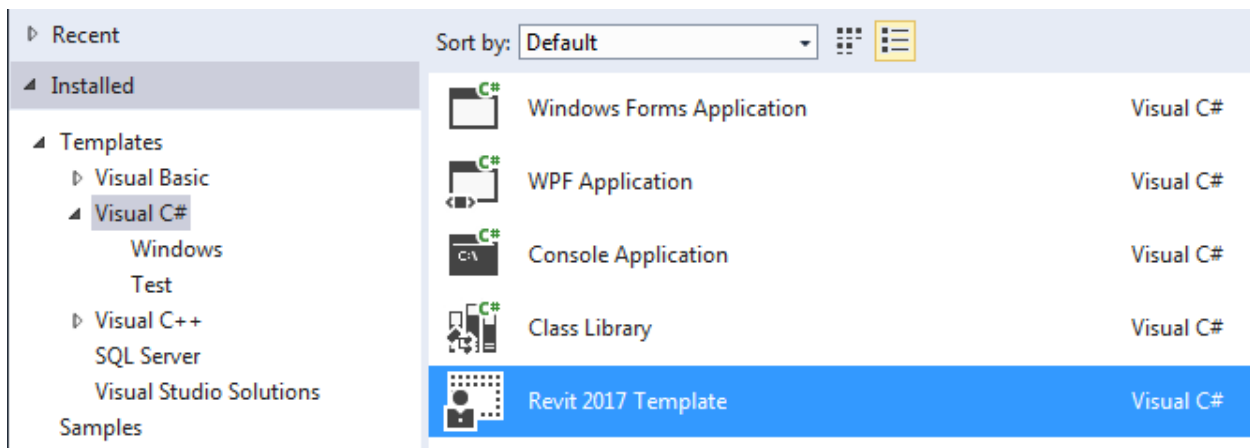


FILE OPTIONS IN VISUAL STUDIO

Your next step will be to Go to the File Tab and select “Export Template...” after saving the changes you have made. Once you have saved your changes, you will be asked to choose the template type (in this case we will want the Project Template). Make sure your current file is selected (HelloRevit) and click the Next button. Now you can give the template a name and select the Finish button.



Go ahead and test out your work by closing your project and then create a new project. You should see a couple options and at the bottom is your new template:



From now on, the beauty of this template is that you can copy/paste the code samples you find from the Revit API help or websites. Make sure to give it a new name and take note of where these projects exist (Usually under your documents\Visual Studio 20xx\Projects)

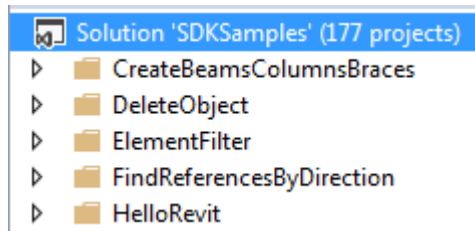




## Compiling the Revit Add-in Samples

Now that you have compiled the RevitLookup example, you may find compiling the Samples to be a bit easier. In the Revit 2017 SDK/Samples folder you should find SDKSamples.sln to get this process started. When you open this file you will notice it is loading every single Sample (so it can be a bit overwhelming at first).

Assuming you have Revit 2017 installed, you should be able to easily right click on the Solution 'SDKSamples' and Build Solution (F7). The code is already looking for the correct Revit API and APIUI DLLs and should start the process of building all of the .DLL files in each of the bin/debug folders.



*SOLUTION EXPLORER IN VISUAL STUDIO*

## Trying out a few Samples or your own code in Revit

The heavy lifting is done and now you are ready to open up Revit with Visual Studio in a separate window (2 monitors is a must for this very reason!) From here on out, we will use the Add-in Manager to test out the samples and RevitLookup to understand exactly what kind of data you have available to you from the Revit API.

## C# Examples from Samples (Revit 2017 API)

These Examples are fairly clean and provide a good starting point:

Example Type	Example Name	Complexity (1-10)
Basic Dialog	HelloRevit	2 - Simple
User Interface	AllViews	5 - Normal
Element Selection	DeleteDimensions	2 - Simple
OnStartup Application	APIAppStartup	1 – Simple
	DisableCommand	4 – Normal
	PrintLog	8 - Complex
InstantUpdater	DynamicModelUpdate	9 - Complex
Element Creation	ElementsBatchCreation	4 - Normal
	GenerateFloor	5- Normal
Reading Data	ParameterUtils	4 - Normal
	PhysicalProp	3 - Simple
	Rooms	7 - Complex
	VersionChecking	2 - Simple
Revit UI – Ribbon	Ribbon	8 - Complex

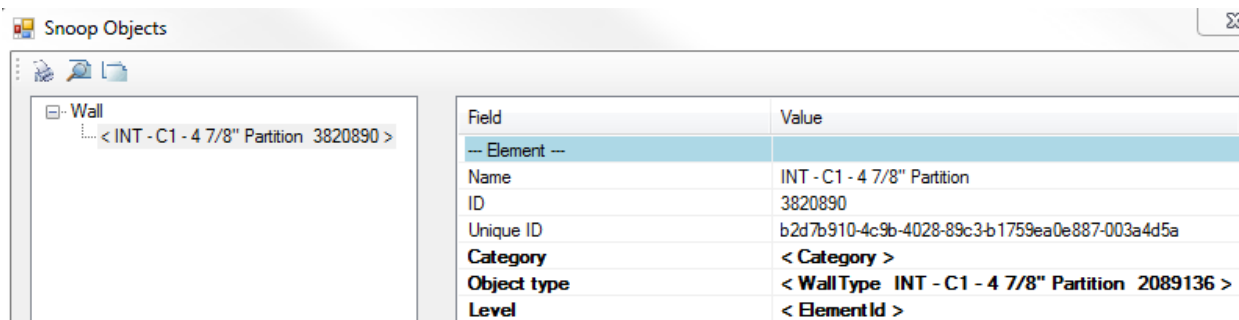
## C# and VB Examples from RevitAPI.chm

These Examples can be copied into the new template easily:

Example Type	Example Name	Complexity (1-10)
Basic Dialog	TaskDialog Class	2 - Simple
Element Selection	Selection Class	2 - Simple
OnStartup Application (Ribbon)	UIApplication. CreateRibbonPanel Method	3 - Simple
	UIControlledApplication. CreateRibbonTab Method	3 - Simple
InstantUpdater	IUpdater Interface	8 - Complex
Element Creation	ViewSheet.Create Method	4 - Normal
	Transaction Class	7 - Complex
Reading Data	FilteredElementCollector. FirstElementId Method	3 - Simple
	FilteredElementCollector. GetElementIterator Method	4 - Normal
	FilteredElementCollector Constructor	2 - Simple
	FamilySymbol Class	3 - Simple

## Using Revit Lookup tool

Obviously since we have spent the time to install this tool, it would be helpful to run through a couple use cases. The beauty of the Revit Lookup tool is that it helps define what parameters and information you can access through the API. For example, if you select a wall in your Revit model and open the Revit Lookup Add-in option "Snoop Current Selection.." you will see parameters such as this:



Field	Value
--- Element ---	
Name	INT - C1 - 4 7/8" Partition
ID	3820890
Unique ID	b2d7b910-4c9b-4028-89c3-b1759ea0e887-003a4d5a
Category	< Category >
Object type	< WallType INT - C1 - 4 7/8" Partition 2089136 >
Level	< ElementId >

REVIT LOOKUP TOOL – SNOOP OBJECTS

Go ahead and click on **< Category >** or any other bold element to see more information about that particular element. One Value that I find most useful is the **< ParameterSet >** as this can help you identify parameter names and whether or not a parameter can be updated or not.

## C# Examples from the Internet

There are numerous examples available on the internet. Below are 3 blogs that I often enjoy:

The Building Coder (Jeremy Tammik) - <http://thebuildingcoder.typepad.com/blog/>

Boost your BIM (Harry Mattison) - <https://boostyourbim.wordpress.com/>

AEC DevBlog - <http://adndevblog.typepad.com/aec/>

## Implement your Add-in inside your company

Now that you have created a useful Revit Add-in that you wish to distribute to the rest of your company, it is time to consider the strategy for doing this. Keep in mind that it is usually smarter to select a few Beta testing employees in your company before you launch the Add-in to the entire firm. Depending on the complexity of your add-in there is always the potential you could have some unforeseen issues occur in Revit files. I typically suggest Beta testing for a couple months and reminding the testers to use the Add-ins in Non-Mission critical projects.

### Sending your .Addin file and .DLL through email

As amateur as this sounds, I often suggest this to BIM Managers that have no access to deployment techniques (Group Policy) or no Network access to individual computers. As mentioned before, to successfully install a Revit Add-in, all you need to do is to place the .Addin file and the .DLL file in the employee's ProgramData folder. If you send these 2 files to your "Beta testers" with instructions to save them to the appropriate ProgramData folder you will find this technique works well enough.

### Modifying the .Addin file to point to a file server (Network drive)

In the interest of being thorough, I must mention that it is also possible to create an .Addin file that points to different location to find the .DLL file. This can be useful when you are planning to update and fix your Add-ins and don't want to keep asking your users to copy files.

However, there are numerous reasons **I don't recommend this technique**. For larger firms, it is typically difficult to get everyone out of Revit to allow you to update the .DLL on the file server. Typically the .DLL will remain read-only if someone is accessing it (with Revit open). As a result, you will need to wait until later in the evening to ensure everyone is out of Revit.

One other difficulty (and this may not work in the newest version of Revit) is that you will need to make a modification to the **revit.exe.config** file on all the users computers. Basically there are security settings that prevent .dll's from loading from remote sources and this can become a bit annoying if you don't adjust this setting.

### Copying files remotely to individual computers

One of the easiest options when you need to copy Revit Add-ins to individual computers is to browse to a specific computer using the [\\username\C\\$](#) technique (obviously you will need Admin rights and a network setup that allows for this). Once you have access to a user's ProgramData folder you can simply copy/paste the .Addin and .DLL to their computer.

Again, if a user has Revit open on their computer at the time you try this, you won't be able to overwrite the DLL file. So, you will either want to do this after hours or just ask the employee to exit out of Revit while you install your Add-ins.

### Pushing out .Addin files and .DLLs through I/T Management Software

This option varies greatly depending on your I/T infrastructure, however this particular example is utilizing my own company's software "ManageEngine Desktop Central". This software allows you to create a File Folder Operation that copies multiple files into the ProgramData folder on specific user's computers. If you have numerous Revit users in your company that would like your Revit Add-ins, this option is a perfect solution. You can even modify the configuration and resend your add-ins out to all of your employees (when you make updates or fixes).

## **The Difficulty level of building your Revit Add-in**

The more experience you have with programming Revit Add-ins, the more accurately you can determine the complexity and time needed for a specific Add-in idea. If programming Add-ins is new to you, it can often be difficult to know if you should spend time on a project or purchase a pre-made Add-in for \$100/Year. Below are a few considerations to make when determining the time needed for programming an Add-in.

### **How complex will your User Interface need to be?**

If you work for a smaller firm or one less concerned with the aesthetics of custom tools that will appear on all the Revit Users computers, you might be able to get away with a bit of a less polished User Interface. If you can get away with this level of simplicity, you won't need to allocate much time to this.

When you need to create a more polished User Interface, this can often double the time required to build this Add-in. Less experienced Revit users will appreciate a good User Interface that can help guide them. Using a WPF Form allows for resizing and more flexibility but will also require a bit more experience than using the traditional Windows Forms.

### **How bug free does your Add-in need to be?**

Most people that are new to programming assume their Add-in will be error free and everyone will use the Add-in properly. The reality is that some users will just push random buttons or enter text that is not formatted properly. If it is important for you, as the programmer, to build a bullet proof Add-in, then you will need to allocate a couple hours for torture testing. As mentioned previously, you will want to have a couple Beta testers in the office before you release your software to the entire office. In addition to Beta testing, you should probably try to throw as much random items at your Add-in to see that it can handle this and exit cleanly.

### **Will your Add-in need to run in the background?**

This is usually the biggest red flag for a complex and potentially buggy Add-in. Although I am not necessarily advising you to avoid this type of Add-in, I have plenty of horror stories about Add-ins going rogue and damaging live projects (crashing or worse). In some cases the tricky part relates to trying to update elements that are attempting to synchronize to the Central file. Sometimes you can even run into issues trying to do too much updating in Revit and it will slow the project down to a crawl.

My typical suggestion for Add-ins that need to automatically update elements is that you keep the updating narrowed down to certain events (such as Save As / Synchronize). This will help prevent some of the slow down issues or potential crashing that might happen if an Add-in is trying to do too much real-time instant adjusting.

### **Manipulating elements using Extensible data?**

Don't worry if you have no idea what Extensible data is, the simple explanation is a storage type that can be attached to a Revit element. In other words, it is sort of like invisible data that you can use in your Revit Add-in without the Revit user seeing any parameters or visible data. This seems like a wonderful concept except it can get fairly angry when this data is formatted differently (perhaps in a newer update of the Add-in). Needless to say, you need to plan ahead and be very careful not to accidentally corrupt data in your Add-in. This can add a new level of complexity to your code to ensure you don't blue screen or crash Revit on accident.

### Just how much customization do you need in this Add-in?

In most cases you know the exact use and settings needed for your Add-in but there are occasions when a user might have a preference that isn't common. You can deal with this by creating a settings dialog or additional dialog screens to help accommodate this. Unfortunately, as you would expect, this can really add some complexity to your programming and should therefore be factored in.

#### Complex tasks

- Instant Updaters and Extensible Storage
- 3D element manipulation (Transform coordinates)
- Complex User Interface – Real-time drag-drop

#### Simple tasks

- List all elements from Revit project
- Update specific parameters
- Simple data shown in Dialog Boxes

*REVIEW OF TASK DIFFICULTY LEVEL*

### Your Revit Add-in: To Build or Buy?

After you have done a bit of an estimate on the complexity of your particular Add-in, you should have some idea of what an equivalent Add-in would be worth. Excuse the business side of me for talking numbers but sometimes you need to do a cost analysis before you undertake a large project. Below are a few suggestions for making this decision.

#### Does your firm expect a custom branded solution?

The larger the firm, the more likely it is that you will be asked to add Company logos or color schemes into your Add-in. There are a few more expensive Add-ins available on the market that will happily modify the look of their add-in (usually for a price). If this price tag seems a bit too harsh, perhaps making your own custom branded Add-in yourself is the way to go.

#### How flexible is the Add-in licensing for your firm size?

Even if you find an Add-in available for purchase at a reasonable price, sometimes the real expense comes with how the licensing is structured. If you have a small firm, it may not be a big deal buying a few licenses but if you have 100 or more users, buying licenses for each user can quickly add up. Much like Autodesk licenses, some software can also be stand-alone or network (floating). If your Add-in is not used by all employees 24/7 then you might feel you are losing money by paying for unneeded licenses. This isn't always an easy calculation but it is important to consider.



Also important to understand about licensing is if there is a yearly renewal. Because Revit has yearly upgrades it is usually standard practice to charge for Add-ins yearly (and perhaps they will offer a discount for existing customers). Upgrades are handled in many different ways but you need to factor that into your pricing also. It is very unlikely that you would purchase an Add-in and not upgrade for 2 or 3 years but if that is your situation you might factor the price appropriately.

### **Can you make suggestions or get support easily?**

This is sometimes hard to know until you find yourself in a situation where you have questions or problems with a software. Not all Add-ins come with support teams and even fewer Add-ins will actually listen to your suggestions for improvements. Obviously if you create the Add-in yourself, your only limit is your own programming ability.

### **Is a trial version available or perhaps a refund offer?**

The issue I have run into in the past is software that shows you a demo but won't let you test drive the software on your own computer until you purchase it. Most of these Add-in providers tend to allow for a refund if the software doesn't work as advertised but some don't. Usually Add-ins with a trial version available allow you to get a good feel for how support is going to treat you (in the previous question). If you are in the middle of a trial and you don't get your questions answered then you can guarantee that you won't get them answered when you purchase the software either.

### **Conclusion: Where do I go next with my Add-in?**

As a final note, you may actually discover that your Add-in is quite useful and perhaps worth the effort to try and sell it on the market yourself. The following questions are important to consider before you take out a loan or try to get rich with your great idea.

### **Did you design this Add-in while at your office?**

This is a bit of a slippery slope because it is certainly wrong to spend time and effort at the office starting your own business. Some firms might try to take legal action to get some profit if they see you are successful selling your Add-in on the market. Clearly you need to draw a line between work that you are doing for your own firm and work that you intend to market yourself. In general it is just best to be clear and honest with the firm you are working for and maybe even sign an agreement to keep you out of trouble.

### **Do you plan to market and sell this online?**

There is quite a bit of work involved in setting up a website (with PayPal or some credit card purchase option). It takes even more work to actually market your product (or set up a booth at the Expo). Not to discourage your ambitions but there is quite a bit of upfront expense in setting this up and should be factored into your equation. It is also becoming difficult in some cases to deal with online transactions and the requirements for sales tax in different states/counties.

### **Is your Add-in already available?**

You would think this is an obvious question but you should always look at your competition (or lack of competition). If your idea is truly unique and seems to be highly in demand then you might find yourself very successful. If you are just improving on a competition's Add-in then you could be on an uphill climb and might need to flex your marketing muscle a bit.





### **How much should you charge for your Add-in?**

As most business people will tell you, this is the biggest and most difficult question to answer sometimes. You have a couple strategies to attempt – low-cost and high volume, or high-cost and low volume sales. It is important to understand that the price you set initially can drive the future of your business (later in the game you will have a hard time changing the price). If you have a highly complex and very specific Add-in you might attempt the High-cost and low sales volume approach. Typically the less complex but more highly utilized Add-ins are better as Low-cost High volume. Spend a bit of time thinking through what you want to accomplish.

### **Can you get away with just copying the .Addin or .DLL manually?**

This seems like a bit much to ask of your potential customers so it is usually more common to create a setup.exe (installer). To create an executable installer you will need to purchase the more expensive Visual Studio Professional software (not the free Express version).

### **How do you set up a licensing system?**

Unless you intend to sell your Add-in at a very low price and just trust your users not to pass around the software, you will need to set up a licensing system. This isn't the easiest process and can add a bit to the cost of doing business also. You might have to flex your programming muscle a bit designing this licensing system into your Installer system. Also, you need to decide if you want to disable the software after a specific time or just let it stop working with newer updates of Revit.

### **Thank you for your time and participation!**

This topic really is my passion and I appreciate your attendance and review of this material. If you have any additional questions or just want to chat with me, I welcome the discussion but cannot promise you an immediate response (always busy with the family / work). It has been my pleasure to write this up and I wish you all the best of luck!