Creating Macros with the Revit API

This topic explains how to create macros in Revit. We will describe macro capabilities, the overall workflow, specific installation steps, code examples, frequently asked questions, important caveats, and related information about the Revit SDK.

Getting Started with Macros

First, let's answer the question: "what are macros, and why would you use them"? Macros are programs that are designed to help you save time, by automating repetitive tasks. Each macro performs a series of pre-defined steps to accomplish a particular task. The steps should be repeatable and the actions predictable.

For example, you might define a macro to add a grid to your project, to rotate a selected object, or to collect information about the square footage of all the rooms in your structure. Other general examples include:

- Locating and extracting Revit content to external files
- Automatically tweaking geometry or parameters
- Automatic creation of many types of elements
- Importing/exporting external file formats

Revit provides an Application Programming Interface (API) that allows you to extend the functionality of the product. Experienced developers and Autodesk partners may already know that the Revit API lets them add customized commands to the Tools > External Commands menu, or add new menus and toolbars.

In addition to those API extensions, starting in the Revit 2009 release you can use the API to define macros that run in Revit. Unlike external commands and external applications, the macro functionality is available to Revit after you install an "add-in" called Revit VSTA. We will explain the API differences later in this topic, but for experienced developers, note that you do not need to register the macros in Revit.ini, or add RevitAPI.dll as a reference.

VSTA is an acronym for Visual Studio Tools for Applications. It is a Microsoft technology that provides the .NET framework for creating macros in C# and VB.NET based on specific applications. VSTA is the next evolution of Visual Basic for Applications (VBA) that appears in several existing Autodesk applications.

Expect API Changes

It is very likely that the Revit API will change in subsequent product releases. This means that after installing the next Revit release, you will need to edit and rebuild your macros to reflect the API changes.

Revit VSTA Components

To create macros, you must install Revit VSTA on top of your existing Revit installation. The Revit product and Revit VSTA versions must be the same.

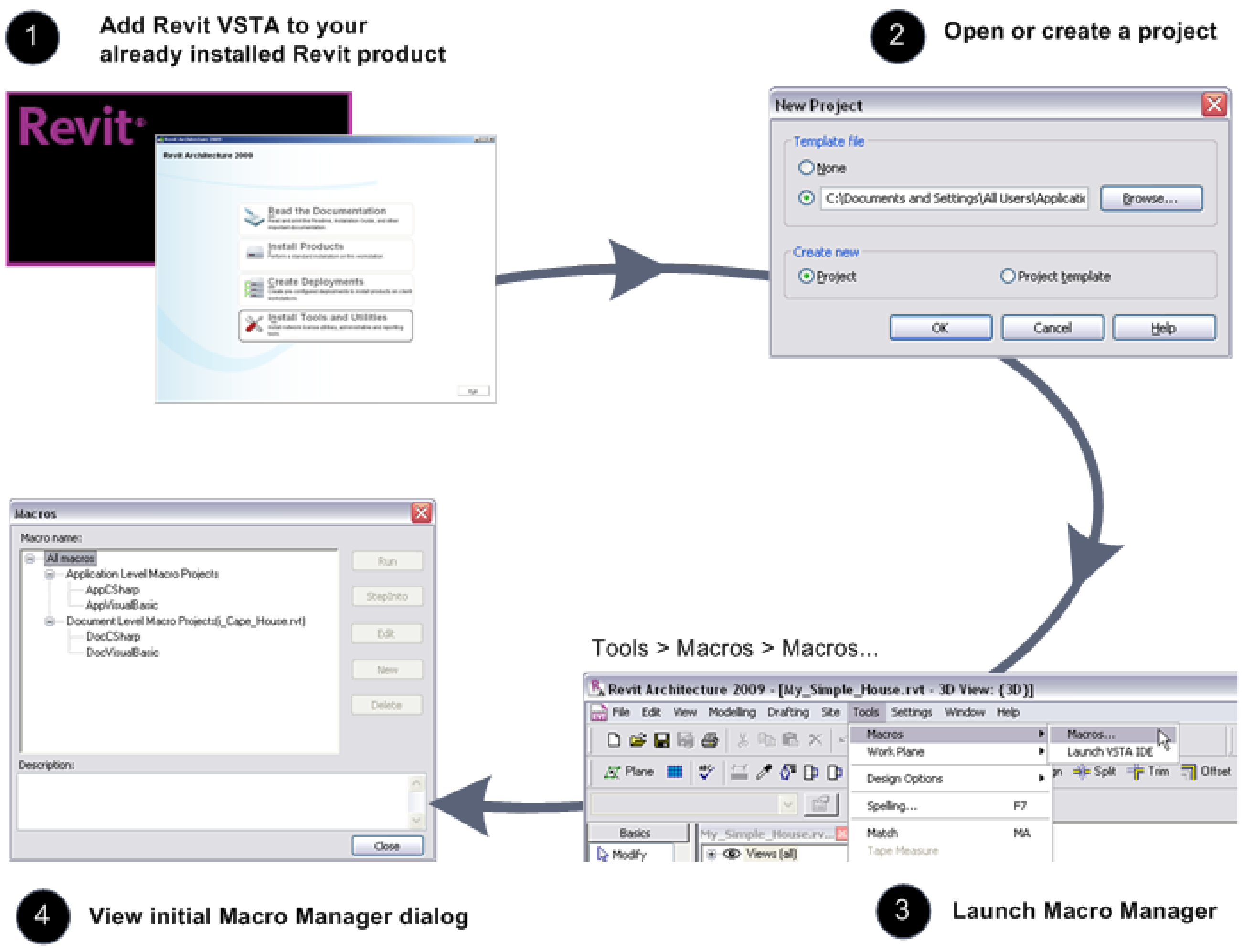
You can use the Revit VSTA macro features in all Revit products: Revit Architecture, Revit Structure, and Revit MEP., however, a separate Revit VSTA install, described below is required for each Revit product that you want to be able to run macros . In this topic, we refer to any of these products generically as "Revit" without a further qualifier.

Once installed, Revit VSTA provides:

- New items on the toolbar's Tools menu:
  - Tools > Macros > Macros…
  - Tools > Macros > Launch VSTA IDE
- Macro Manager, a user interface launched by the Tools > Macros > Macros… menu option. Macro Manager presents a list of macros you built previously that you can run, edit, or debug (StepInto).   Macro Manager also provides options to create new macros using different types of templates.
- An Integrated Development Environment (IDE) built into the product, the Revit VSTA IDE. You can launch it several ways:
  - Tools > Macros > Launch VSTA IDE
  - From the Macro Manager, by selecting the New, Edit, or StepInto buttons
- Full access to the Revit API.

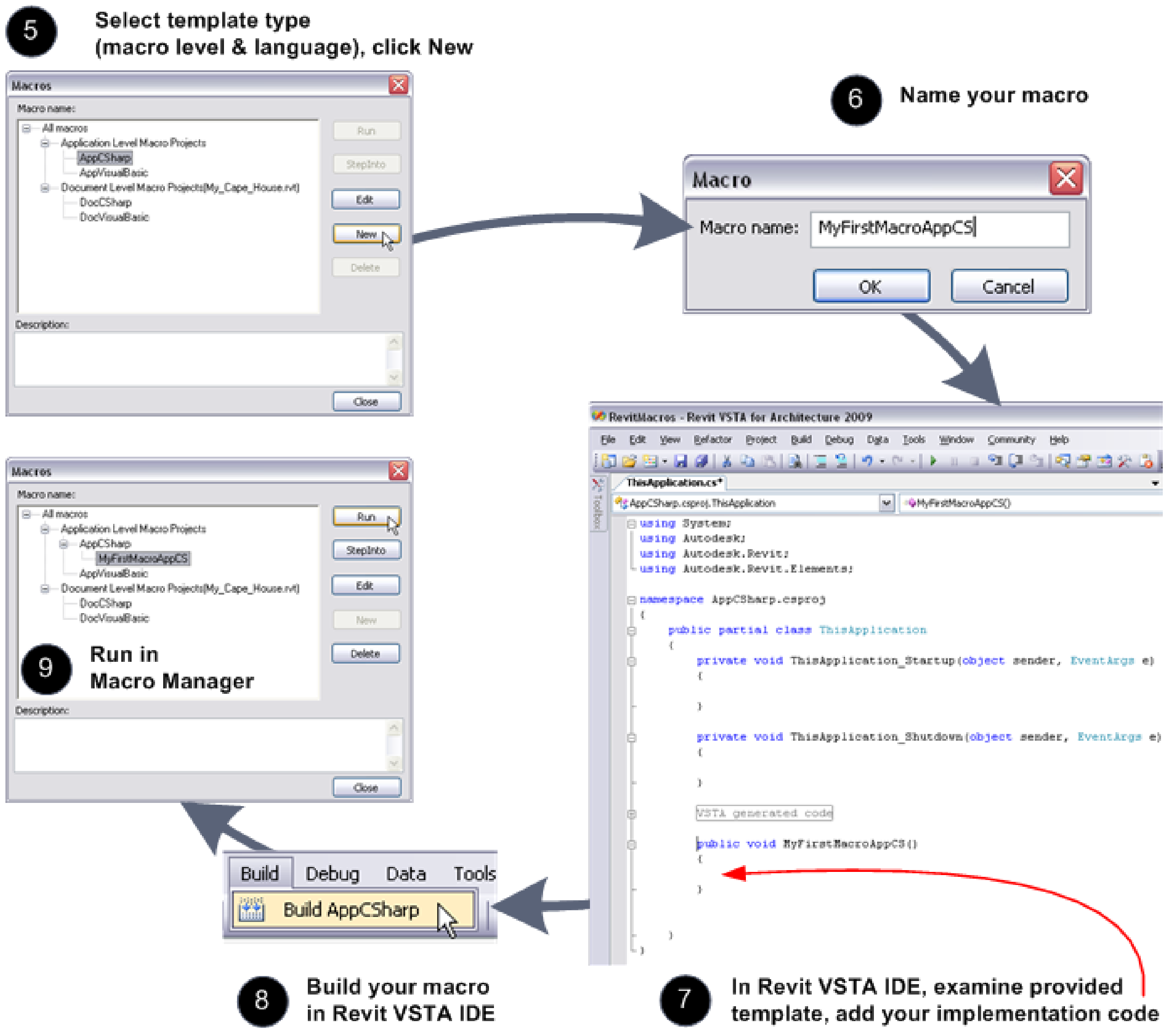
Workflow Overview: The Initial Steps with Macros

The following diagram illustrates the initial steps with Revit VSTA and macro development.



Workflow Overview: Creating, Building, and Running Macros

After installing Revit VSTA and starting Macro Manager, select the type of macro you want to create, click New or Edit, and use the Revit VSTA IDE to define the macro, adding your implementation code. When you are ready, you build the macro in the IDE. After the build succeeds, you can run the macro in Macro Manager and observe the results. The following workflow illustrates the overall process, with the numbering continued from the previous diagram.


5
Select template type (macro level & language), click New
Macros
Macro name:
All macros
Application Level Macro Projects
AppCSharp
AppVisualBasic
Document Level Macro Projects(My_Cape_House.rvt)
DocCSharp
DocVisualBasic
Run
StepInto
Edit
New
Delete
Description:
Close
6
Name your macro
Macro
Macro name: MyFirstMacroAppCS
OK
Cancel
RevitMacros - Revit VSTA for Architecture 2009
File Edit View Refactor Project Build Debug Data Tools Window Community Help
ThisApplication.cs*
AppCSharp.csproj.ThisApplication
MyFirstMacroAppCS()
7
In Revit VSTA IDE, examine provided template, add your implementation code
Build Debug Data Tools
Build AppCSharp
8
Build your macro in Revit VSTA IDE
Macros
Macro name:
All macros
Application Level Macro Projects
AppCSharp
MyFirstMacroAppCS
AppVisualBasic
Document Level Macro Projects(My_Cape_House.rvt)
DocCSharp
DocVisualBasic
Run
StepInto
Edit
New
Delete
Description:
Close
9
Run in Macro Manager

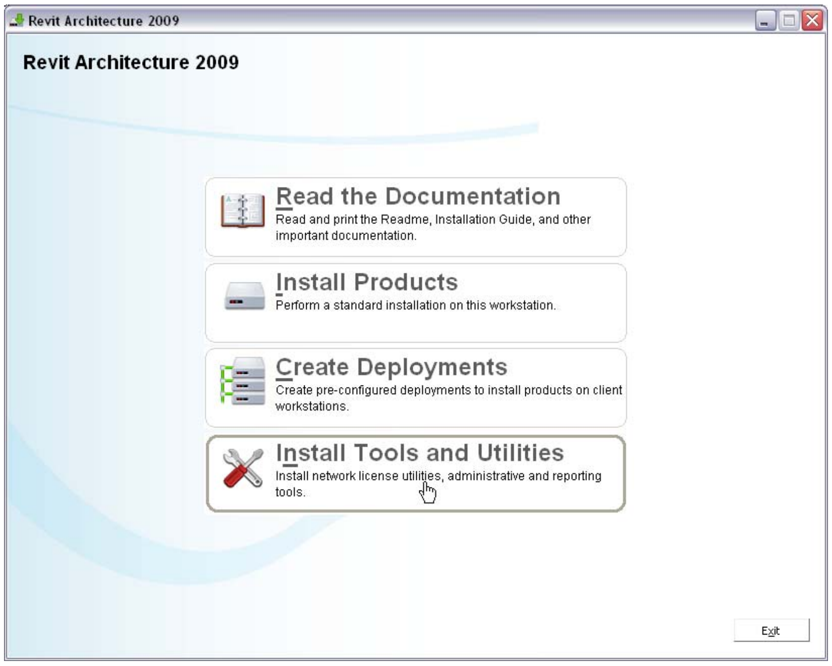
```
using System;
using Autodesk;
using Autodesk.Revit;
using Autodesk.Revit.Elements;

namespace AppCSharp.csproj
{
    public partial class ThisApplication
    {
        private void ThisApplication_Startup(object sender, EventArgs e)
        {

        }

        private void ThisApplication_Shutdown(object sender, EventArgs e)
        {

        }

        VSTA generated code

        public void MyFirstMacroAppCS()
        {

        }


    }
}
```

Now that we have introduced the overall process, let’s look at the specific steps.

Installing Revit VSTA

Follow these steps to install the Revit VSTA on top of your existing Revit product installation:

1. Check that Revit is not running on your computer. If it is, close Revit.
2. Start the Revit installation program again from the product DVD or download kit (the same program used to install the base Revit product).
3. On the main menu, select the option to Install Tools and Utilities, as shown here.



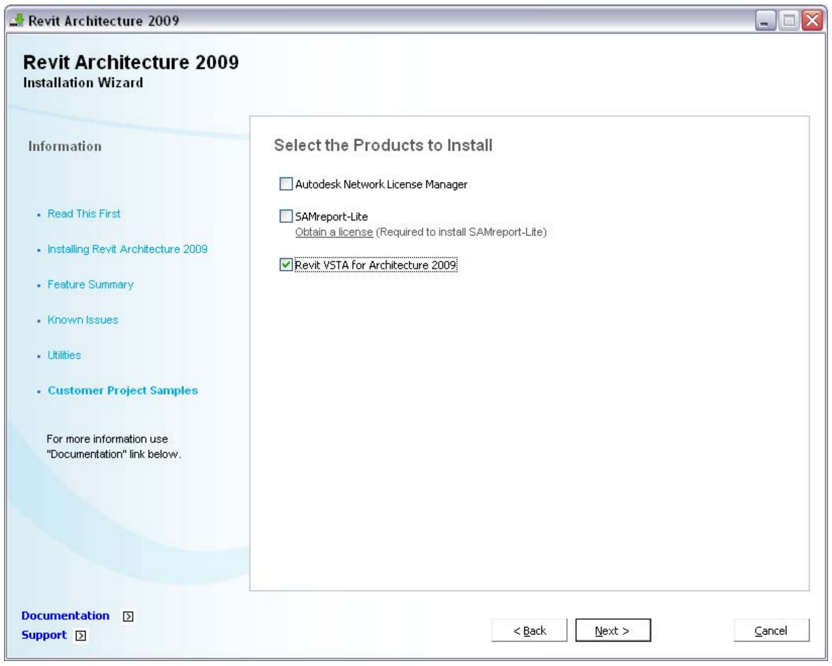
Read and print the Readme, Installation Guide, and other important documentation.

Perform a standard installation on this workstation.

Create pre-configured deployments to install products on client workstations.

Install network license utilities, administrative and reporting tools.

4. Proceed to the screen that lists the option for the Revit VSTA. For example, here is the option from the Revit Architecture installer.



For more information use "Documentation" link below.

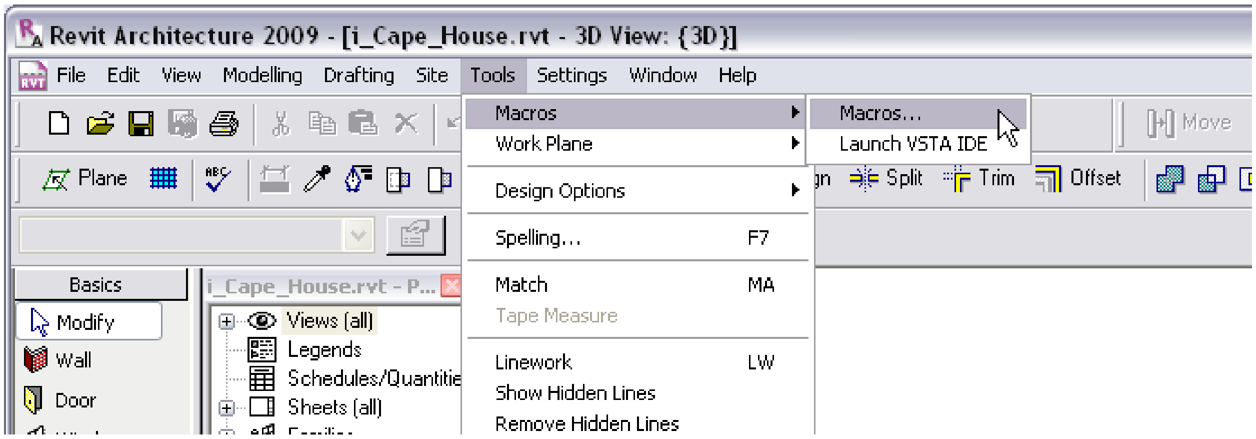
Select the Products to Install

- [ ] Autodesk Network License Manager
- [ ] SAMreport-Lite
  Obtain a license (Required to install SAMreport-Lite)
- [✓] Revit VSTA for Architecture 2009

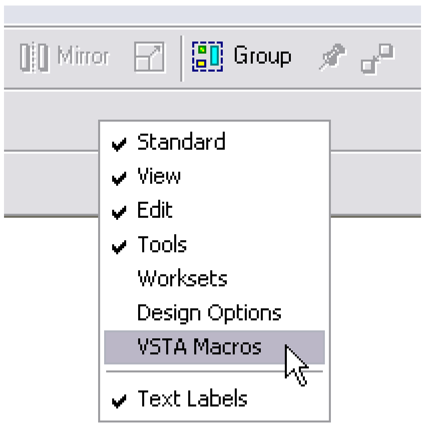
Follow the prompts, click Install, then Finish on the final Revit VSTA screen.

Start Revit and Notice Updated Tools Menu for Macros

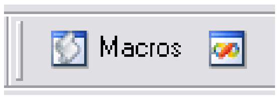
After you install Revit VSTA, start the base Revit product. Open an existing project or create a new one. Notice that the Tools menu contains a Macros submenu. Here is an example from Revit Architecture.



You can also display toolbar shortcuts for Macro Manager and the VSTA IDE by right clicking in any gray region of the docked toolbar and selecting VSTA Macros from the menu.



Clicking the VSTA Macros option results in the addition of the following icons on the Revit toolbar.

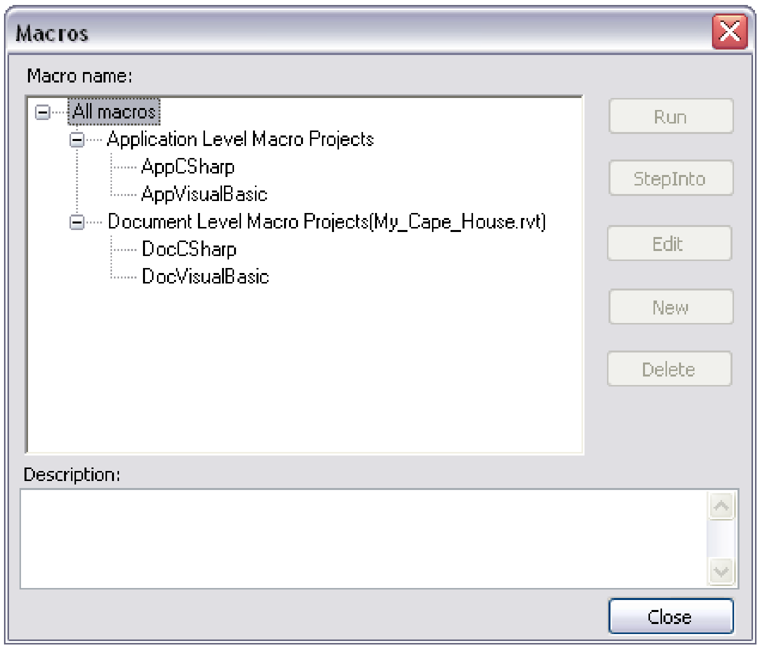


Using Macro Manager and the Revit VSTA IDE

Macro Manager is the user interface for:

- Selecting an option that launches the Revit VSTA IDE, where you can add, edit, build and debug your macros.
- Running a previously built macro from a categorized list.

Shown here is the initial Macro Manager screen.



The buttons are inactive because we have not yet selected two pieces of information:

- The scope or “level” of a macro
- The macro programming language.

For details, see the next section.

Application-Level and Document-Level Macros

In addition to selecting your preferred programming language (C# or VB.NET), you must choose one of two levels for macros:

- **Application-level macros:** Choose this level if you want the macros to be available to all opened Revit projects in the current instance of the Revit application. Note that if you send the .rvt file to a person on another computer, application level macros would not be available.
- **Document-level macros:** Choose this level if you want to define macros that are embedded in the currently open project’s .rvt file. Anyone who opens the .rvt can run the macros, provided the person using the macro and the person who built the macro have both installed VSTA for their respective Revit products.

On the Macros dialog box, select your preferred macro level and the implementation language, C# or VB.NET. When you click the New button, your selection determines the type of source code template generated by Revit. The following table summarizes the options:

If you select this option…	**And then click New, it results in…**
AppCSharp	Application-level macro project source in C#, opened in Revit VSTA IDE
AppVisualBasic	Application-level macro project source in VB.NET, opened in Revit VSTA IDE
DocCSharp	Document-level macro source project in C#, opened in Revit VSTA IDE
DocVisualBasic	Document-level macro project source in VB.NET, opened in Revit VSTA IDE

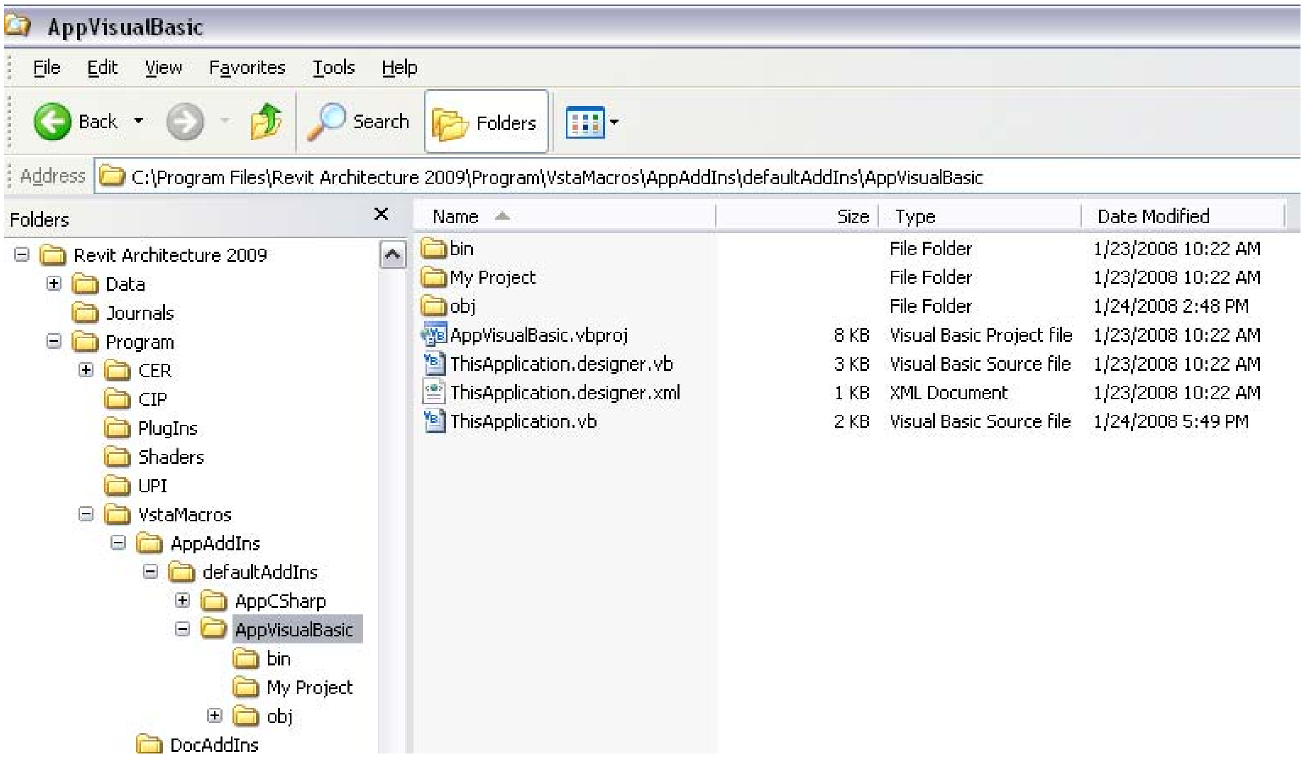
Macro Project File Locations

When you work in the Revit VSTA IDE, you must save and build the macros successfully before they will appear in the Macro Manager's categorized list. Before we look at an example of the initial code loaded into the Revit VSTA IDE, let's discuss where macro project files reside on your computer.

On disk, Application-level macro projects are stored in a subfolder of the Revit installation directory. For example:

```
C:\Program Files\Revit Architecture 2009\Program\VstaMacros\AppAddIns\defaultAddIns\...
```

Here is an example with Windows Explorer and the Application-level macro project's files created for a VB.NET implementation.



C:\Program Files\Revit Architecture 2009\Program\VstaMacros\AppAddIns\defaultAddIns\AppVisualBasic

Name	Size	Type	Date Modified
bin		File Folder	1/23/2008 10:22 AM
My Project		File Folder	1/23/2008 10:22 AM
obj		File Folder	1/24/2008 2:48 PM
AppVisualBasic.vbproj	8 KB	Visual Basic Project file	1/23/2008 10:22 AM
ThisApplication.designer.vb	3 KB	Visual Basic Source file	1/23/2008 10:22 AM
ThisApplication.designer.xml	1 KB	XML Document	1/23/2008 10:22 AM
ThisApplication.vb	2 KB	Visual Basic Source file	1/24/2008 5:49 PM

Document-level macro projects are stored within an .rvt file. On disk, when the associated .rvt project opens, any built and saved macro(s) are stored temporarily in:

```
C:\Program Files\Revit Architecture 2009\Program\VstaMacros\DocAddIns\defaultAddIns\...
```

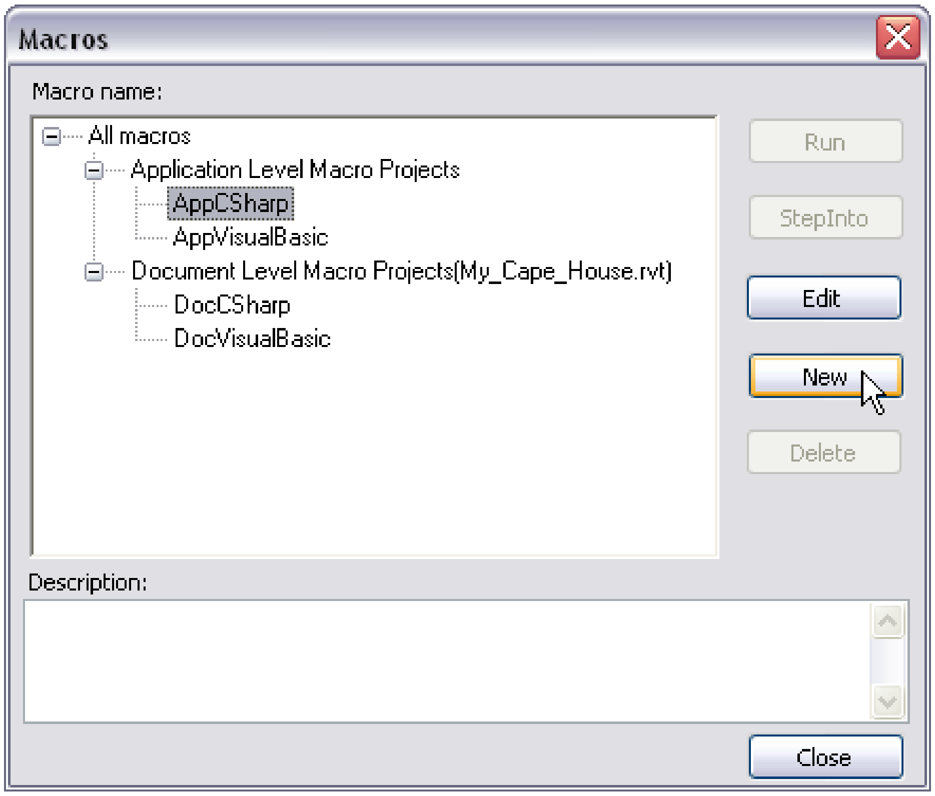
However, these Document-level macro files are deleted from your local computer when their corresponding Revit project document (.rvt) closes. The saved Document-level macros are stored in the .rvt file.

Tip: Remember that you must successfully build and save a macro project in the Revit VSTA IDE, as explained in the next section, before it will appear in the Macro Manager's categorized list.

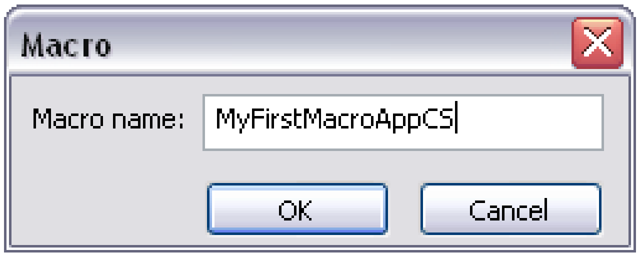
Creating Your First Set of Macros

After you decide on the desired level and language for your macro, as discussed in prior sections of this topic, highlight that type on the Macro Manager list and then click New.

Let's walk through an example. In this first screen, select an Application-level macro and C# as the implementation language:

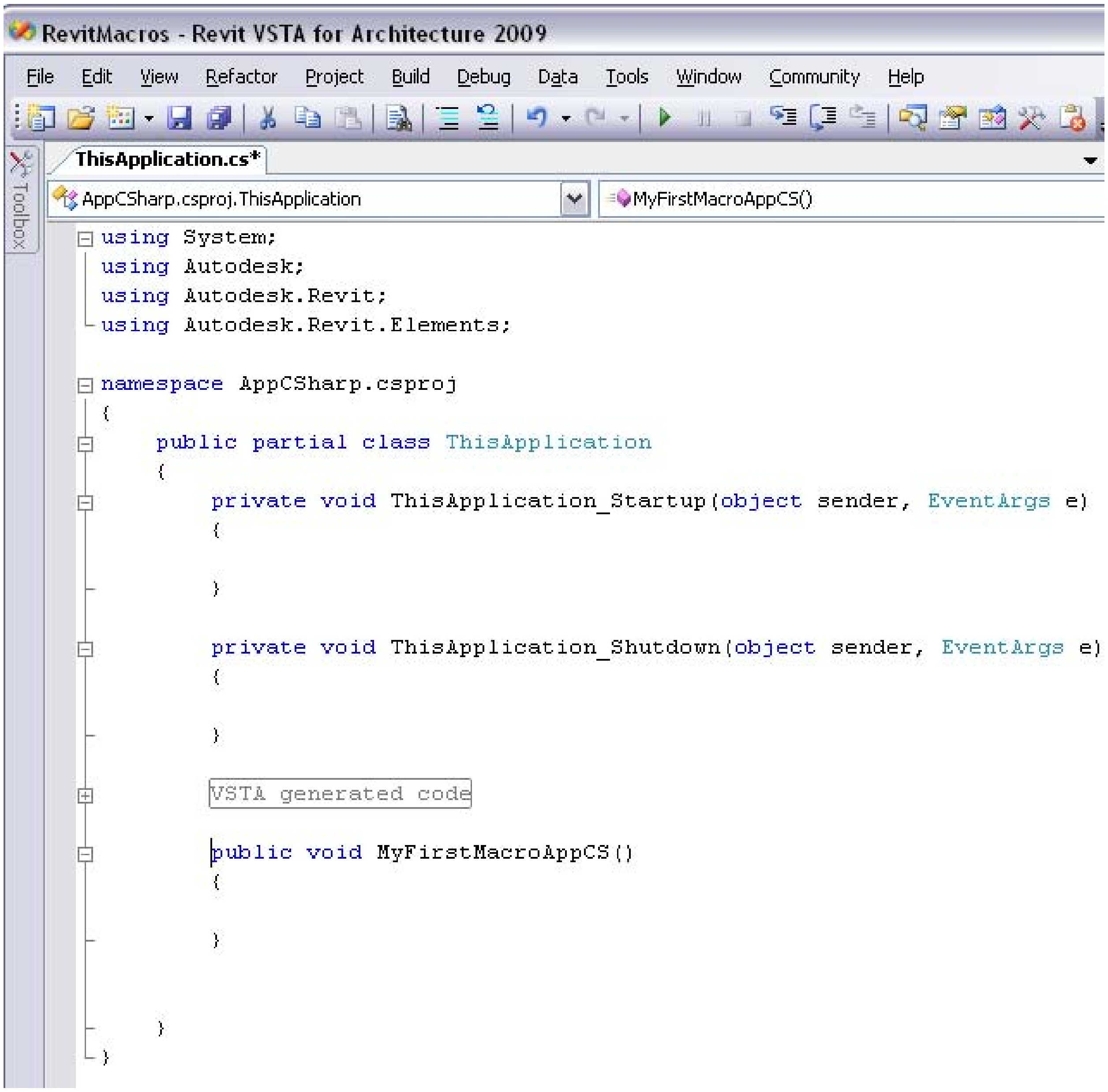


Click the New button. Revit prompts you to name the macro. For example:



Macro name: MyFirstMacroAppCS

The Revit VSTA IDE presents a starting template for Application-level macros in C#. Here is a portion of the screen:

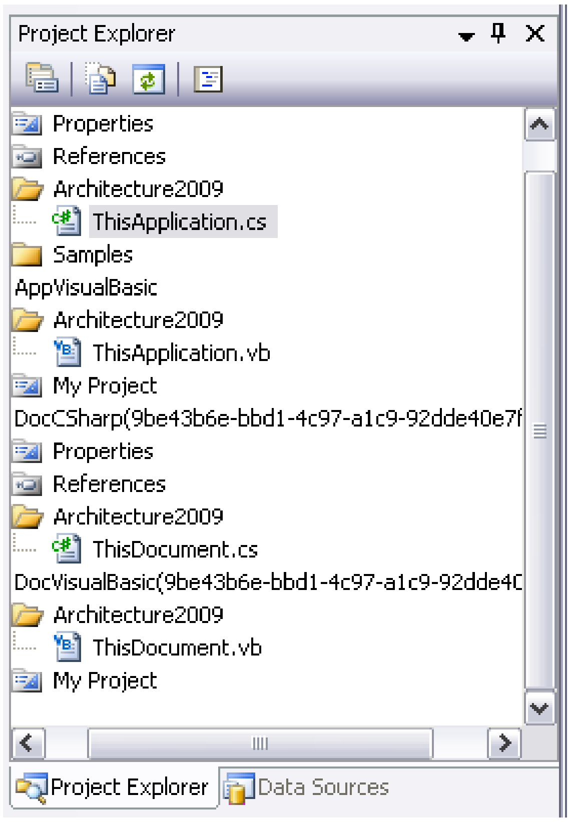

RevitMacros - Revit VSTA for Architecture 2009
File Edit View Refactor Project Build Debug Data Tools Window Community Help
ThisApplication.cs*
Toolbox
AppCSharp.csproj.ThisApplication
MyFirstMacroAppCS()

```
using System;
using Autodesk;
using Autodesk.Revit;
using Autodesk.Revit.Elements;

namespace AppCSharp.csproj
{
    public partial class ThisApplication
    {
        private void ThisApplication_Startup(object sender, EventArgs e)
        {

        }

        private void ThisApplication_Shutdown(object sender, EventArgs e)
        {

        }

        VSTA generated code

        public void MyFirstMacroAppCS()
        {

        }


    }
}
```

Notice that in this C# template for Application-level macros, Revit VSTA has already:

- Included the necessary `using` directives
- Identified the `AppCSharp` namespace
- Started the `ThisApplication` class definition
- Started the methods for `ThisApplication_Startup()` and `ThisApplication_Shutdown()`
- Started your new macro's method, named `MyFirstMacroAppCS`, giving you the opportunity to add your implementation code between the braces

Also note in the following screen how the Revit VSTA Project Explorer shows your context.   You are editing ThisApplication.cs:



Example: Application-Level Macro in C#

In the window for the ThisApplication.cs source code, you can now enter your macro code. For example, here is the `MyFirstMacroAppCS` method in C#:

```
public void MyFirstMacroAppCS()
{
    Autodesk.Revit.Geometry.XYZ baseVec = this.Create.NewXYZ(1.0, 0.0, 0.0);
    Autodesk.Revit.Geometry.XYZ upVec = this.Create.NewXYZ(0.0, 0.0, 1.0);
    Autodesk.Revit.Geometry.XYZ origin = this.Create.NewXYZ(0.0, 0.0, 0.0);

    Autodesk.Revit.Enums.TextAlignFlags align =
        Autodesk.Revit.Enums.TextAlignFlags.TEF_ALIGN_LEFT |
        Autodesk.Revit.Enums.TextAlignFlags.TEF_ALIGN_TOP;

    string strText = "My First Macro, App level, C#!";
    double lineWidth = 4.0 / 12.0;

    View pView = this.ActiveDocument.ActiveView;

    this.ActiveDocument.Create.NewTextNote(pView, origin, baseVec, upVec,
        lineWidth, align, strText);
}
```

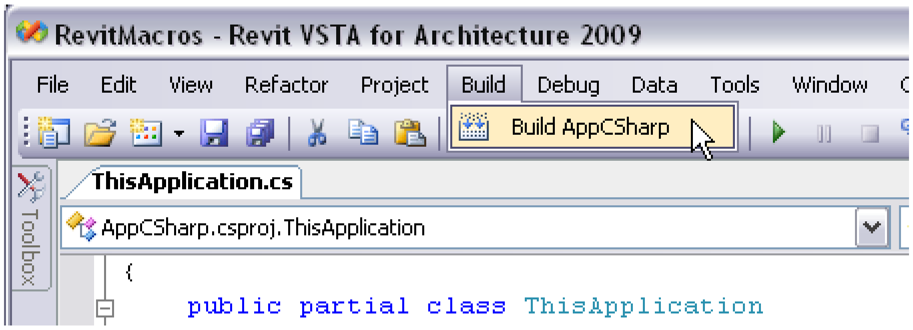
In the example, the Revit API Geometry.XYZ class is used to define a position (with X, Y, Z coordinates) for a Text Note that the macro will add to the active view of the active document.

For now, you can skip any implementation of the startup and shutdown methods.

In your Revit VSTA IDE session, you can copy and paste the code snippet shown previously and insert it into your macro's method. After pasting the code from this documentation, check that no special characters were added.

Build Macros in the Revit VSTA IDE

In the Revit VSTA IDE, select the Build option from the toolbar menu:



```
{
    public partial class ThisApplication
```

In this example, notice that you are building the AppCSharp project. Your Application-level C# macro's code resides in ThisApplication.cs. You can use the IDE's Project Explorer to see its location on disk.

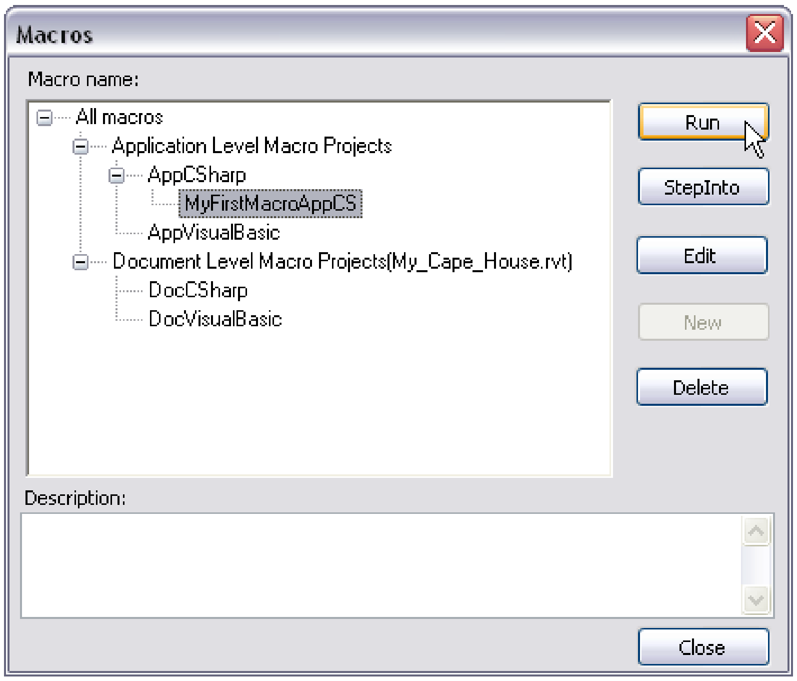
Reminder: After editing a macro's code, remember to build it before opening the Macro Manager again to run it. If your code is set up correctly, you should see a Build Succeeded message in the lower-left corner (by default) of Revit VSTA. You do not build macros in the Macro Manager.

Run Macros in Macro Manager

Next, return to the main Revit product. To see this example macro in action, open one of your views, such as a First Floor plan. As you will note from the code, this Application-level macro adds the Text Note to the active view of the active document.

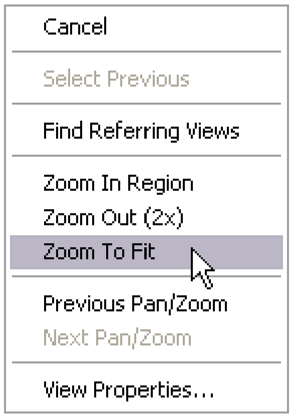
```
View pView = this.ActiveDocument.ActiveView;

this.ActiveDocument.Create.NewTextNote(pView, origin, baseVec, upVec,
    lineWidth, align, strText);
```

Launch Macro Manager again by selecting Tools > Macros > Macros… from the toolbar menus. In Macro Manager, find your macro, select it, and click Run. For example:

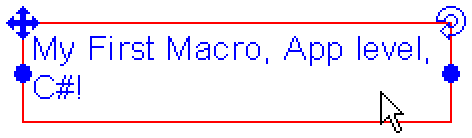


Observe the Macro's Results

After you click run, the macro will execute and should add a Text Note on the current view in Revit. Its location may be difficult to notice. If you do not see the Text Note added by the macro, you can right-mouse click anywhere on your view's drawing area and select Zoom to Fit from the menu. For example:



Then look for the Text Note and zoom-in on it, to verify that the macro worked. In the case of the view we used, the Text Note appeared far to the right of the model. Your location may be different. When we zoomed in and selected the Text Note, it looked like this:



Although you can experiment with the XYZ coordinates in the macro code (including negative numbers like -300 for the X origin), the real point of this example is to notice that the macro really did add the text note.

To remove the Text Note, you can select and delete it, or choose Edit > Undo Macro (Ctrl+Z) from the Revit toolbar menu.

Example: Application-Level Macro in VB.NET

Next let's return to the Macro Manager and add a similar example, this time in VB.NET.

In Macro Manager, select AppVisualBasic, and click New.

Give your macro a name, such as `MyFirstMacroAppVB`.

In the IDE, use the following code for the method:

```
    Public Sub MyFirstMacroAppVB()

        Dim baseVec As Autodesk.Revit.Geometry.XYZ = Me.Create.NewXYZ(1.0, 0.0, 0.0)

        Dim upVec As Autodesk.Revit.Geometry.XYZ = Me.Create.NewXYZ(0.0, 0.0, 1.0)

        Dim origin As Autodesk.Revit.Geometry.XYZ = Me.Create.NewXYZ(0.0, 0.0, 0.0)

        Dim align As Autodesk.Revit.Enums.TextAlignFlags =
Autodesk.Revit.Enums.TextAlignFlags.TEF_ALIGN_LEFT Or
Autodesk.Revit.Enums.TextAlignFlags.TEF_ALIGN_TOP

        Dim strText As String = "My First Macro, App Level, VB.NET!"
        Dim lineWidth As Double = 4.0 / 12.0

        Dim pView As Autodesk.Revit.Elements.View = Me.ActiveDocument.ActiveView

        Me.ActiveDocument.Create.NewTextNote(pView, origin, baseVec, upVec, lineWidth,
align, strText)

    End Sub
```

In this VB.NET example, we used the `Me` keyword pointer (instead of the `this` keyword pointer). Also we used a different `strText` value for the Text Note.

Reminder: Be sure to build your project in the Revit VSTA IDE before trying to run it from Macro Manager.

For this example, when you build the project in the Revit VSTA IDE, notice that you are building the AppVisualBasic project. Your Application-level VB.NET macro's code resides in ThisApplication.vb. You can use the IDE's Project Explorer to see its location on disk. To run your newly built macro, select it in Macro Manager and click Run. Then if necessary, right-click in the active view and select Zoom to Fit from the menu to see the Text Note added by your macro.

Accessing the Application Object in Document-Level Macros

Before we continue with additional `MyFirstMacro*` examples, let's talk about the Revit Application object, and the use of keyword pointers.

Developers know that in C#, you use the keyword `this` to return the current instance of an object. In VB.NET, you use the keyword `Me` to return the current instance of an object.

In Revit, all of the Application-level macros are associated with the `Application` object. Thus using the `this` keyword (C#) or the `Me` keyword (VB.NET) always returns the API `Application` object. This includes all the application-wide data and settings.

In Document-level macros (specific to a .rvt), the `this` keyword (in C#) or the `Me` keyword (VB.NET) returns the API `Document` object. If you need to access the `Application` object from a Document-level macro, use:

`this.Application` … or

`Me.Application`

Example: Document-Level Macro in C#

For the next example, in Macro Manager, select DocCSharp and click New.

Give your macro a name, such as `MyFirstMacroDocCS`.

In the IDE, use the following code for the method:

```
        public void MyFirstMacroDocCS()
        {
            Autodesk.Revit.Geometry.XYZ baseVec = this.Application.Create.NewXYZ(1.0, 0.0, 0.0);
            Autodesk.Revit.Geometry.XYZ upVec = this.Application.Create.NewXYZ(0.0, 0.0, 1.0);
            Autodesk.Revit.Geometry.XYZ origin = this.Application.Create.NewXYZ(0.0, 0.0, 0.0);

            Autodesk.Revit.Enums.TextAlignFlags align =
Autodesk.Revit.Enums.TextAlignFlags.TEF_ALIGN_LEFT |
Autodesk.Revit.Enums.TextAlignFlags.TEF_ALIGN_TOP;

            string strText = "My First Macro, Document level, CS!";
            double lineWidth = 4.0 / 12.0;

            View pView = this.ActiveView;

            this.Create.NewTextNote(pView, origin, baseVec, upVec,
                lineWidth, align, strText);
        }
```

Notice the use of `this.Application` in this Document-level macro. For related information, see "Accessing the Application Object in Document-Level Macros."

Reminder: Be sure to build your project in the Revit VSTA IDE before trying to run it from Macro Manager.

For this example, when you build the project in the Revit VSTA IDE, also notice that you are building the DocCSharp project. Your Document-level C# macro's code resides in ThisDocument.cs. You can use the IDE's Project Explorer to see its temporary location on disk. Recall that the code for successfully built Document-level macros are stored in the .rvt file after you Save the .rvt file; the project files are removed from the temporary location when you exit Revit.

To run your newly built macro, select it in Macro Manager and click Run. Then if necessary, right-click in the active view and select Zoom to Fit from the menu to see the Text Note added by your macro.

Example: Document-Level Macro in VB.NET

For the next example, in Macro Manager, select DocVisualBasic and click New.

Give your macro a name, such as `MyFirstMacroDocVB`.

In the IDE, use the following code for the method:

```
    Public Sub MyFirstMacroDocVB()

        Dim baseVec As Autodesk.Revit.Geometry.XYZ = Me.Application.Create.NewXYZ(1.0, 0.0, 0.0)

        Dim upVec As Autodesk.Revit.Geometry.XYZ = Me.Application.Create.NewXYZ(0.0, 0.0, 1.0)

        Dim origin As Autodesk.Revit.Geometry.XYZ = Me.Application.Create.NewXYZ(0.0, 0.0, 0.0)

        Dim align As Autodesk.Revit.Enums.TextAlignFlags =
Autodesk.Revit.Enums.TextAlignFlags.TEF_ALIGN_LEFT Or
Autodesk.Revit.Enums.TextAlignFlags.TEF_ALIGN_TOP

        Dim strText As String = "My First Macro, Doc Level, VB.NET!"
        Dim lineWidth As Double = 4.0 / 12.0

        Dim pView As Autodesk.Revit.Elements.View = Me.ActiveView

        Me.Create.NewTextNote(pView, origin, baseVec, upVec, lineWidth, align, strText)

    End Sub
```

Reminder: Be sure to build your project in the Revit VSTA IDE before trying to run it from Macro Manager.

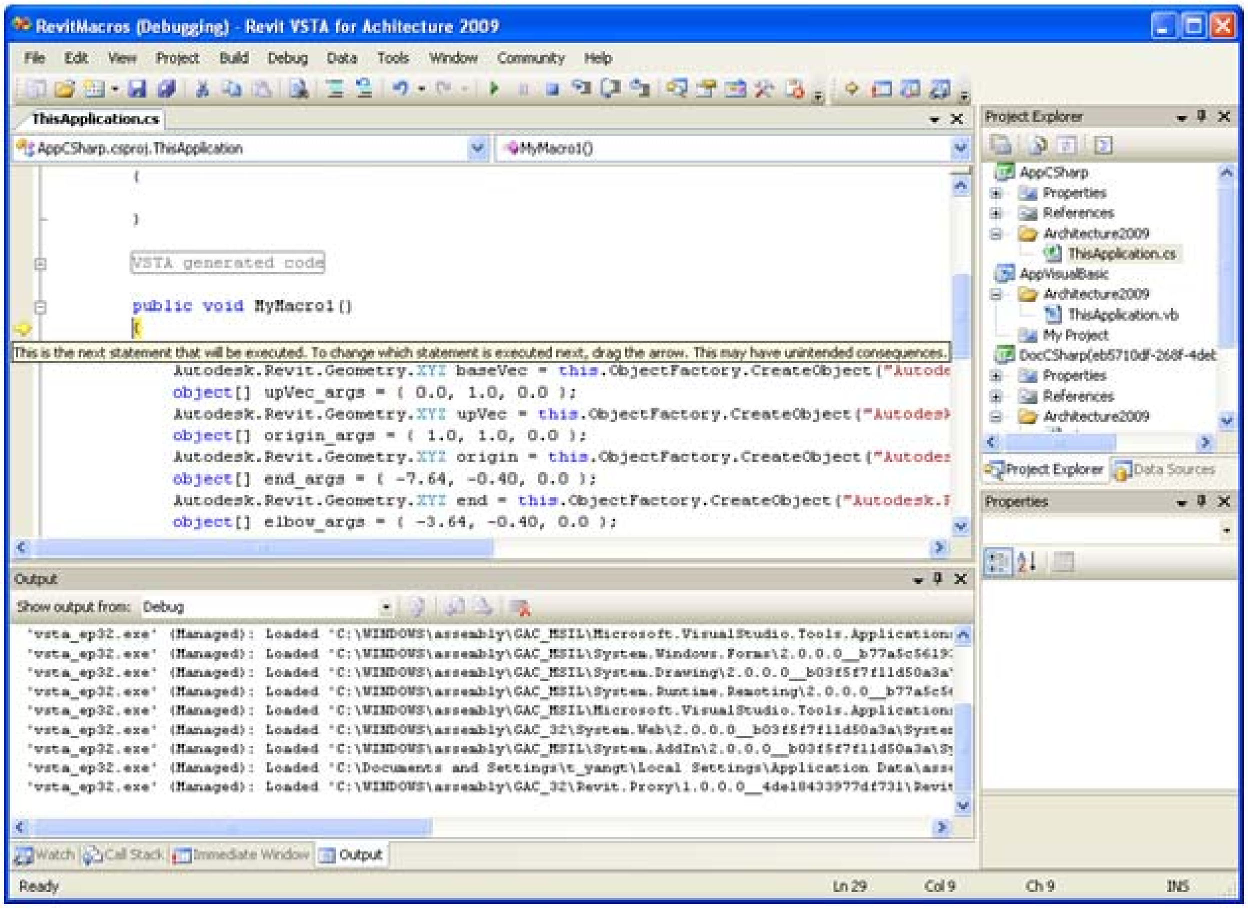
For this example, when you build the project in the Revit VSTA IDE, notice that you are building the DocVisualBasic project, and your Document-level VB.NET macro's code resides in ThisDocument.vb. You can use the IDE's Project Explorer to see its temporary location on disk. Recall that the code for successfully built Document-level macros are stored in the .rvt file after you Save the .rvt file; the project files are removed from the temporary location when you exit Revit.

To run your newly built macro, select it in Macro Manager and click Run. Then if necessary, right-click in the active view and select Zoom to Fit from the menu to see the Text Note added by your macro.

The StepInto Option, and Debugging Macros

In Macro Manager, you can select a macro and click the **StepInto** button to step through the selected macros statements in the Revit VSTA IDE.   Here are the subsequent steps:

- The macro will pause at the first statement. For example:



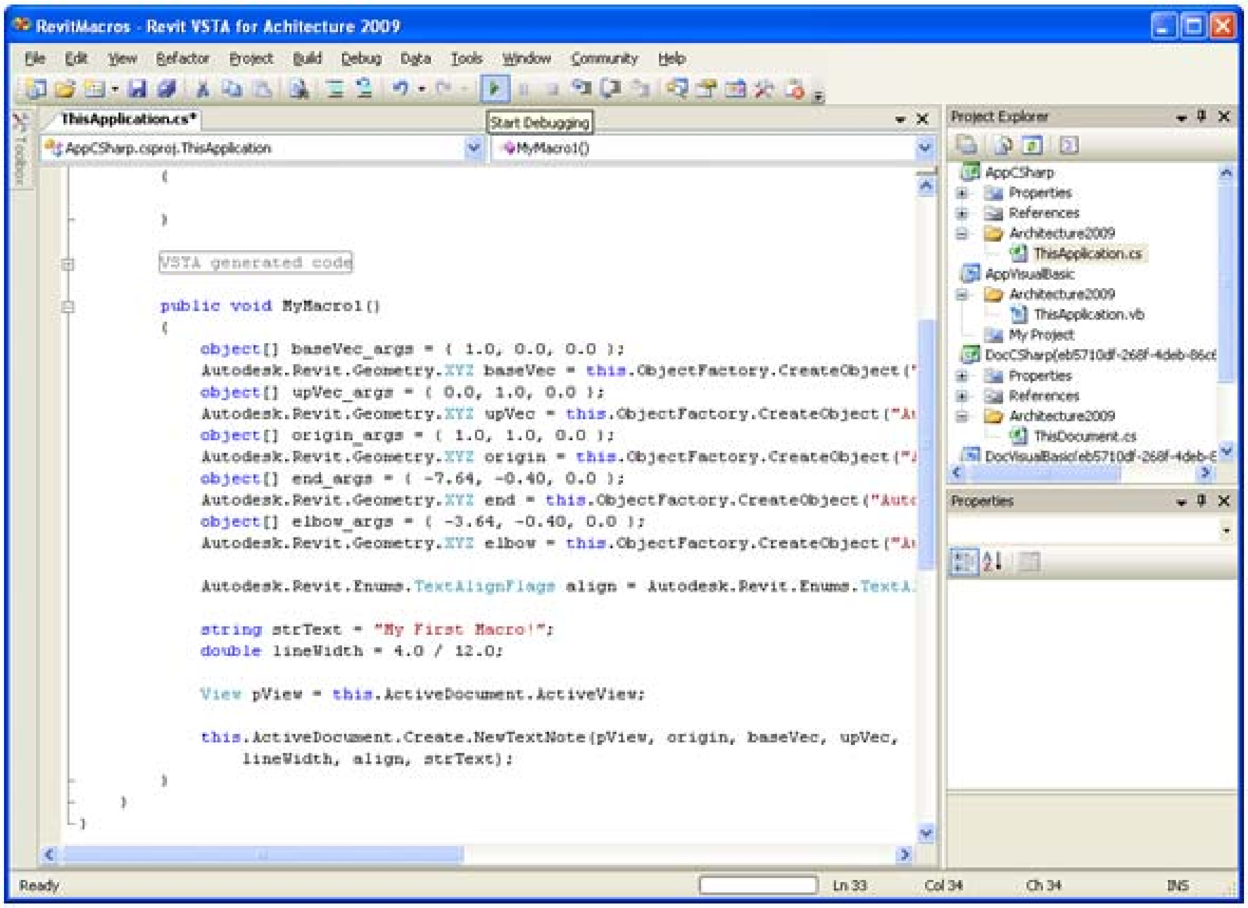
```
        {

        }

        VSTA generated code

        public void MyMacro1()
        {
This is the next statement that will be executed. To change which statement is executed next, drag the arrow. This may have unintended consequences.
            Autodesk.Revit.Geometry.XYZ baseVec = this.ObjectFactory.CreateObject("Autode
            object[] upVec_args = ( 0.0, 1.0, 0.0 );
            Autodesk.Revit.Geometry.XYZ upVec = this.ObjectFactory.CreateObject("Autodesk
            object[] origin_args = ( 1.0, 1.0, 0.0 );
            Autodesk.Revit.Geometry.XYZ origin = this.ObjectFactory.CreateObject("Autode
            object[] end_args = ( -7.64, -0.40, 0.0 );
            Autodesk.Revit.Geometry.XYZ end = this.ObjectFactory.CreateObject("Autodesk.R
            object[] elbow_args = ( -3.64, -0.40, 0.0 );
```

```
'vsta_ep32.exe' (Managed): Loaded 'C:\WINDOWS\assembly\GAC_MSIL\Microsoft.VisualStudio.Tools.Applications
'vsta_ep32.exe' (Managed): Loaded 'C:\WINDOWS\assembly\GAC_MSIL\System.Windows.Forms\2.0.0.0__b77a5c5619
'vsta_ep32.exe' (Managed): Loaded 'C:\WINDOWS\assembly\GAC_MSIL\System.Drawing\2.0.0.0__b03f5f7f11d50a3a'
'vsta_ep32.exe' (Managed): Loaded 'C:\WINDOWS\assembly\GAC_MSIL\System.Runtime.Remoting\2.0.0.0__b77a5c5
'vsta_ep32.exe' (Managed): Loaded 'C:\WINDOWS\assembly\GAC_MSIL\Microsoft.VisualStudio.Tools.Applications
'vsta_ep32.exe' (Managed): Loaded 'C:\WINDOWS\assembly\GAC_32\System.Web\2.0.0.0__b03f5f7f11d50a3a\System
'vsta_ep32.exe' (Managed): Loaded 'C:\WINDOWS\assembly\GAC_MSIL\System.AddIn\2.0.0.0__b03f5f7f11d50a3a\Sy
'vsta_ep32.exe' (Managed): Loaded 'C:\Documents and Settings\t_yangt\Local Settings\Application Data\asse
'vsta_ep32.exe' (Managed): Loaded 'C:\WINDOWS\assembly\GAC_32\Revit.Proxy\1.0.0.0__4de18433977df731\Revit
```

- Press the F10 key to step into each statement.

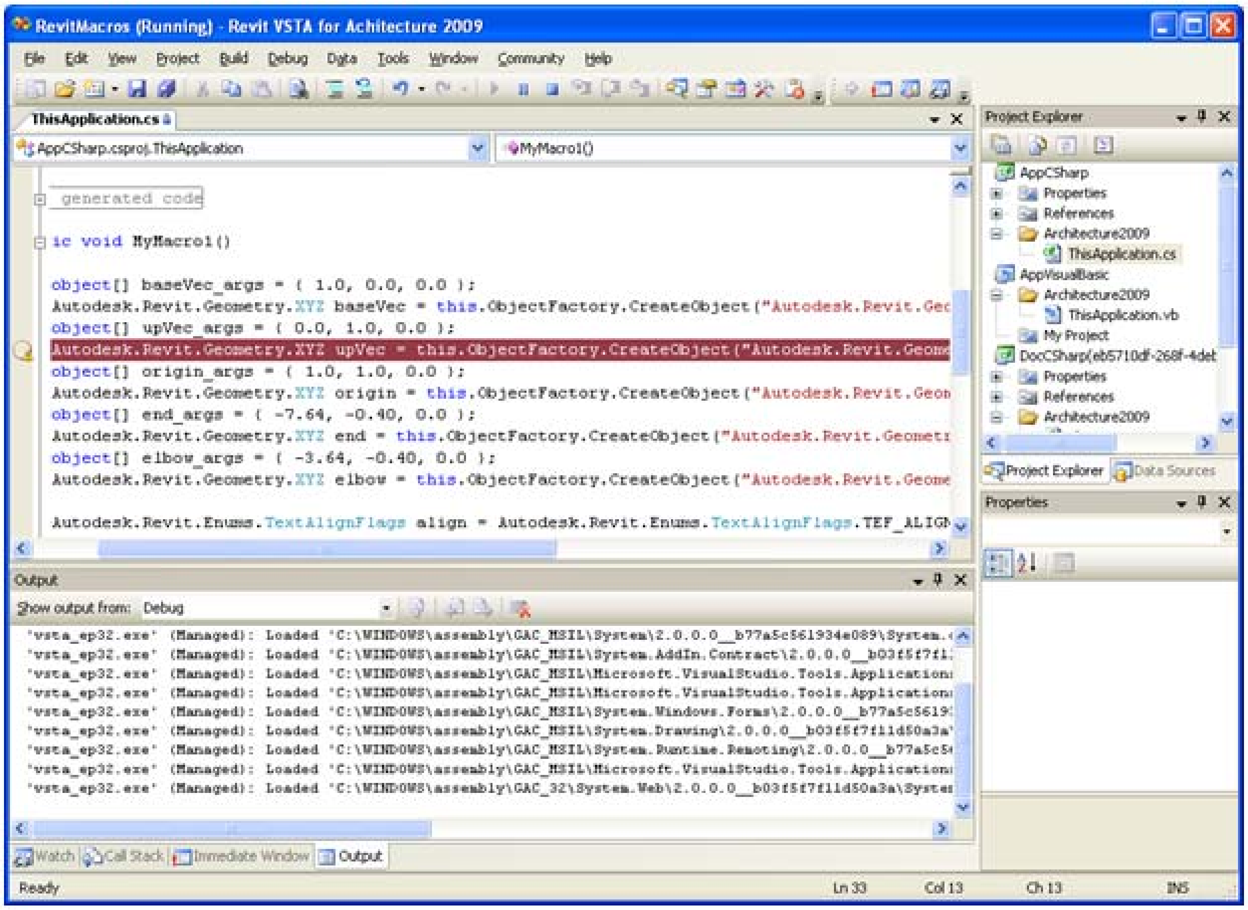
The Revit VSTA IDE provides features that go beyond StepInto, allowing you to debug the macro in the development environment.

- From the Revit menu, choose Tools > Macros > Launch VSTA IDE
- From the IDE menu, choose Start Debugging.



```
        {

        }

        VSTA generated code

        public void MyMacro1()
        {
            object[] baseVec_args = { 1.0, 0.0, 0.0 };
            Autodesk.Revit.Geometry.XYZ baseVec = this.ObjectFactory.CreateObject("
            object[] upVec_args = { 0.0, 1.0, 0.0 };
            Autodesk.Revit.Geometry.XYZ upVec = this.ObjectFactory.CreateObject("Au
            object[] origin_args = { 1.0, 1.0, 0.0 };
            Autodesk.Revit.Geometry.XYZ origin = this.ObjectFactory.CreateObject("A
            object[] end_args = { -7.64, -0.40, 0.0 };
            Autodesk.Revit.Geometry.XYZ end = this.ObjectFactory.CreateObject("Autc
            object[] elbow_args = { -3.64, -0.40, 0.0 };
            Autodesk.Revit.Geometry.XYZ elbow = this.ObjectFactory.CreateObject("Au

            Autodesk.Revit.Enums.TextAlignFlags align = Autodesk.Revit.Enums.TextA

            string strText = "My First Macro!";
            double lineWidth = 4.0 / 12.0;

            View pView = this.ActiveDocument.ActiveView;

            this.ActiveDocument.Create.NewTextNote(pView, origin, baseVec, upVec,
                lineWidth, align, strText);
        }
    }
}
```

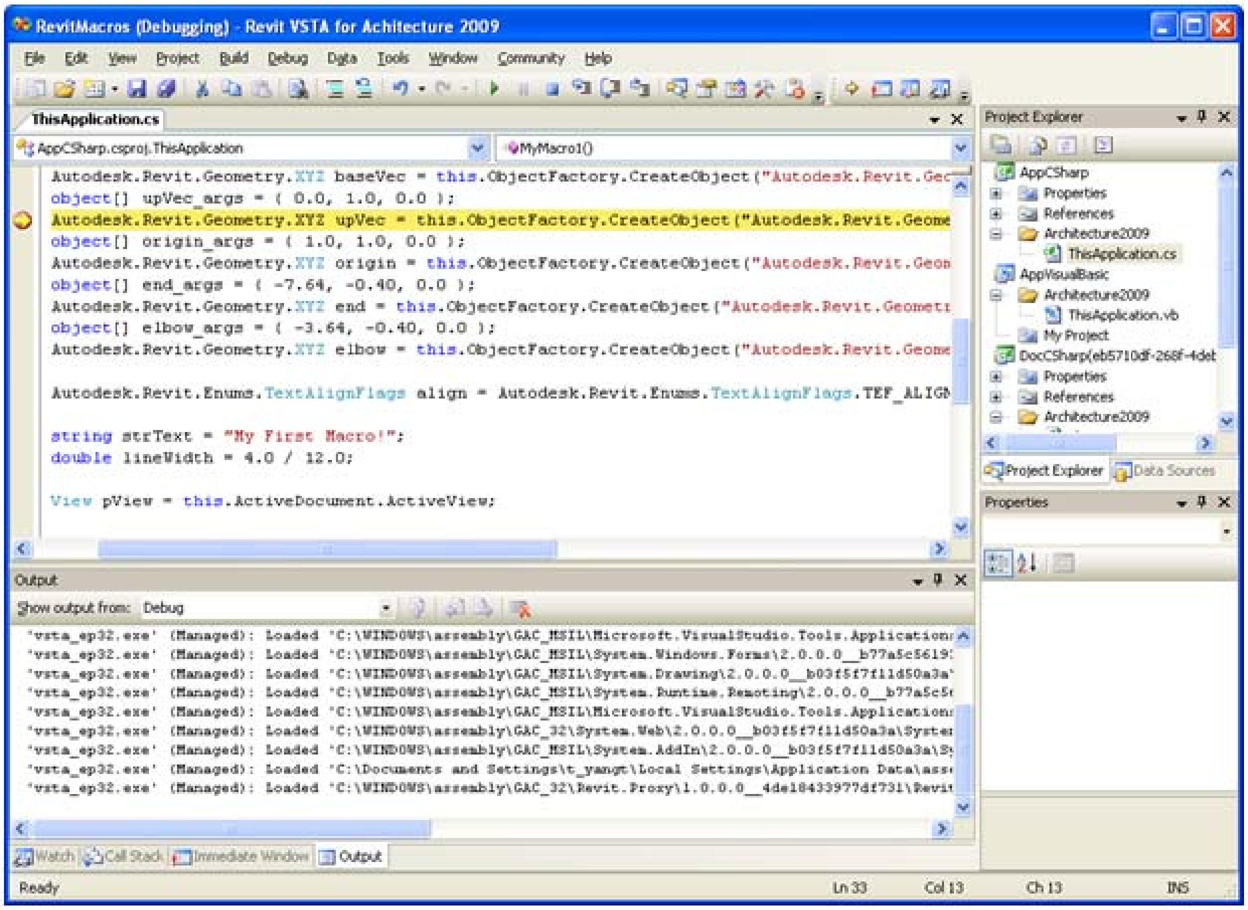
- Press the F9 key to set a break point at the statement you want to debug.



```
 generated code

ic void MyMacro1()

object[] baseVec_args = { 1.0, 0.0, 0.0 };
Autodesk.Revit.Geometry.XYZ baseVec = this.ObjectFactory.CreateObject("Autodesk.Revit.Gec
object[] upVec_args = { 0.0, 1.0, 0.0 };
Autodesk.Revit.Geometry.XYZ upVec = this.ObjectFactory.CreateObject("Autodesk.Revit.Geome
object[] origin_args = { 1.0, 1.0, 0.0 };
Autodesk.Revit.Geometry.XYZ origin = this.ObjectFactory.CreateObject("Autodesk.Revit.Geon
object[] end_args = { -7.64, -0.40, 0.0 };
Autodesk.Revit.Geometry.XYZ end = this.ObjectFactory.CreateObject("Autodesk.Revit.Geometr
object[] elbow_args = { -3.64, -0.40, 0.0 };
Autodesk.Revit.Geometry.XYZ elbow = this.ObjectFactory.CreateObject("Autodesk.Revit.Geome

Autodesk.Revit.Enums.TextAlignFlags align = Autodesk.Revit.Enums.TextAlignFlags.TEF_ALIGN
```

```
'vsta_ep32.exe' (Managed): Loaded 'C:\WINDOWS\assembly\GAC_MSIL\System\2.0.0.0__b77a5c561934e089\System.
'vsta_ep32.exe' (Managed): Loaded 'C:\WINDOWS\assembly\GAC_MSIL\System.AddIn.Contract\2.0.0.0__b03f5f7f1
'vsta_ep32.exe' (Managed): Loaded 'C:\WINDOWS\assembly\GAC_MSIL\Microsoft.VisualStudio.Tools.Application
'vsta_ep32.exe' (Managed): Loaded 'C:\WINDOWS\assembly\GAC_MSIL\Microsoft.VisualStudio.Tools.Application
'vsta_ep32.exe' (Managed): Loaded 'C:\WINDOWS\assembly\GAC_MSIL\System.Windows.Forms\2.0.0.0__b77a5c5619
'vsta_ep32.exe' (Managed): Loaded 'C:\WINDOWS\assembly\GAC_MSIL\System.Drawing\2.0.0.0__b03f5f7f11d50a3a'
'vsta_ep32.exe' (Managed): Loaded 'C:\WINDOWS\assembly\GAC_MSIL\System.Runtime.Remoting\2.0.0.0__b77a5c5
'vsta_ep32.exe' (Managed): Loaded 'C:\WINDOWS\assembly\GAC_MSIL\Microsoft.VisualStudio.Tools.Application
'vsta_ep32.exe' (Managed): Loaded 'C:\WINDOWS\assembly\GAC_32\System.Web\2.0.0.0__b03f5f7f11d50a3a\Syste
```

- Switch back to Revit.
- From the menu, choose Tools > Macros > Macros…
- In Macro Manager, expand the project type and select your macro.
- Choose the Run option. In the IDE, the macro will execute and pause at the statement marked with a break point.



```
Autodesk.Revit.Geometry.XYZ baseVec = this.ObjectFactory.CreateObject("Autodesk.Revit.Geo
object[] upVec_args = { 0.0, 1.0, 0.0 };
Autodesk.Revit.Geometry.XYZ upVec = this.ObjectFactory.CreateObject("Autodesk.Revit.Geome
object[] origin_args = { 1.0, 1.0, 0.0 };
Autodesk.Revit.Geometry.XYZ origin = this.ObjectFactory.CreateObject("Autodesk.Revit.Geom
object[] end_args = { -7.64, -0.40, 0.0 };
Autodesk.Revit.Geometry.XYZ end = this.ObjectFactory.CreateObject("Autodesk.Revit.Geometr
object[] elbow_args = { -3.64, -0.40, 0.0 };
Autodesk.Revit.Geometry.XYZ elbow = this.ObjectFactory.CreateObject("Autodesk.Revit.Geome

Autodesk.Revit.Enums.TextAlignFlags align = Autodesk.Revit.Enums.TextAlignFlags.TEF_ALIGN

string strText = "My First Macro!";
double lineWidth = 4.0 / 12.0;

View pView = this.ActiveDocument.ActiveView;
```

```
'vsta_ep32.exe' (Managed): Loaded 'C:\WINDOWS\assembly\GAC_MSIL\Microsoft.VisualStudio.Tools.Applications
'vsta_ep32.exe' (Managed): Loaded 'C:\WINDOWS\assembly\GAC_MSIL\System.Windows.Forms\2.0.0.0__b77a5c5619
'vsta_ep32.exe' (Managed): Loaded 'C:\WINDOWS\assembly\GAC_MSIL\System.Drawing\2.0.0.0__b03f5f7f11d50a3a'
'vsta_ep32.exe' (Managed): Loaded 'C:\WINDOWS\assembly\GAC_MSIL\System.Runtime.Remoting\2.0.0.0__b77a5c5
'vsta_ep32.exe' (Managed): Loaded 'C:\WINDOWS\assembly\GAC_MSIL\Microsoft.VisualStudio.Tools.Applications
'vsta_ep32.exe' (Managed): Loaded 'C:\WINDOWS\assembly\GAC_32\System.Web\2.0.0.0__b03f5f7f11d50a3a\Syste
'vsta_ep32.exe' (Managed): Loaded 'C:\WINDOWS\assembly\GAC_MSIL\System.AddIn\2.0.0.0__b03f5f7f11d50a3a\Sy
'vsta_ep32.exe' (Managed): Loaded 'C:\Documents and Settings\t_yangt\Local Settings\Application Data\ass
'vsta_ep32.exe' (Managed): Loaded 'C:\WINDOWS\assembly\GAC_32\Revit.Proxy\1.0.0.0__4de18433977df731\Revit
```

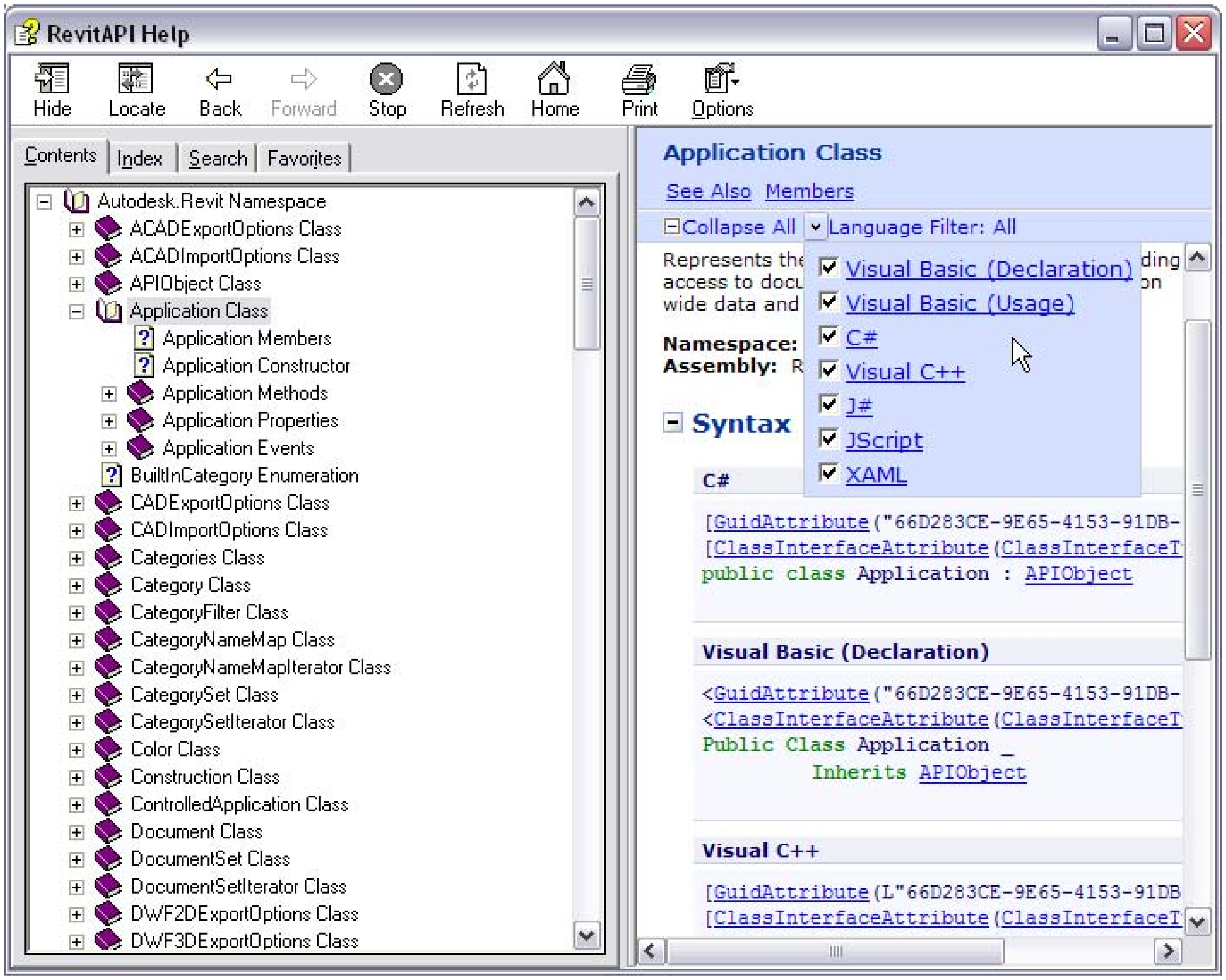
- Press the F10 key to step over each statement.

Revit SDK, API Reference Documentation, and VSTA Samples

The Revit Software Development Kit (SDK) contains useful resources to help you understand the API and create macros. The SDK includes the Revit API reference documentation, the full SDK API samples, and the Revit VSTA samples. The Revit SDK is available on:

- The Revit DVD
- The Autodesk web site, at http://www.autodesk.com/revit-sdk. (**Revit 2009 Beta customers:** this URL is not active yet.)
- The Autodesk Developer Network, http://adn.autodesk.com. If you are interested, please contact your Autodesk representative for information about getting an ADN account.

On the web sites, the SDK is packaged in a ZIP file. After unzipping it and agreeing to the license text, look for the Revit*API.chm Help file. On a Windows computer, open the .chm and refer to the classes, properties, and methods described there. For example:



Autodesk.Revit Namespace
ACADExportOptions Class
ACADImportOptions Class
APIObject Class
Application Class
Application Members
Application Constructor
Application Methods
Application Properties
Application Events
BuiltInCategory Enumeration
CADExportOptions Class
CADImportOptions Class
Categories Class
Category Class
CategoryFilter Class
CategoryNameMap Class
CategoryNameMapIterator Class
CategorySet Class
CategorySetIterator Class
Color Class
Construction Class
ControlledApplication Class
Document Class
DocumentSet Class
DocumentSetIterator Class
DWF2DExportOptions Class
DWF3DExportOptions Class

Application Class

See Also Members

Collapse All Language Filter: All

- [✓] Visual Basic (Declaration)
- [✓] Visual Basic (Usage)
- [✓] C#
- [✓] Visual C++
- [✓] J#
- [✓] JScript
- [✓] XAML

Represents the ... ding access to docu... on wide data and

Namespace:
Assembly: R

Syntax

C#

```
[GuidAttribute("66D283CE-9E65-4153-91DB-
[ClassInterfaceAttribute(ClassInterfaceT
public class Application : APIObject
```

Visual Basic (Declaration)

```
<GuidAttribute("66D283CE-9E65-4153-91DB-
<ClassInterfaceAttribute(ClassInterfaceT
Public Class Application _
    Inherits APIObject
```

Visual C++

```
[GuidAttribute(L"66D283CE-9E65-4153-91DB
[ClassInterfaceAttribute(ClassInterfaceT
```

Also see the Revit VSTA samples that are part of the SDK. You can find them under:

```
\Revit <release-year> SDK\VSTA Samples\...
```

The next section explains how to integrate the VSTA Samples into your Revit VSTA projects.

Integrating VSTA Samples from SDK to IDE

Power users can learn useful API techniques by using the VSTA samples provided with the Revit SDK.

To integrate the samples into your Revit projects, follow the steps described in this section.

Steps to Integrate One VSTA Sample: Rooms

First, let's go through the steps to integrate one sample. We will use the C# Application-level sample called Rooms. This macro finds all the room tags in the current project and presents a dialog box with summary information.

As noted in the prior section "Revit SDK, API Reference Documentation, and VSTA Samples," the VSTA Samples are provided by the separate Revit SDK.

Note that the SDK contains two samples folders:

`\Revit <release-year> SDK\Samples\...`

`\Revit <release-year> SDK\VSTA Samples\...`

For this exercise, refer to the samples under the \VSTA Samples\ folder. Specifically in this section, we will use the sample under:

`\Revit <release-year> SDK\VSTA Samples\CS\Rooms\...`

Add Required References

Before you run the Rooms sample, open an .rvt that contains at least a few room tags.

The Rooms macro uses Windows Forms, , so we must add two references to our macro project in the Revit VSA IDE:

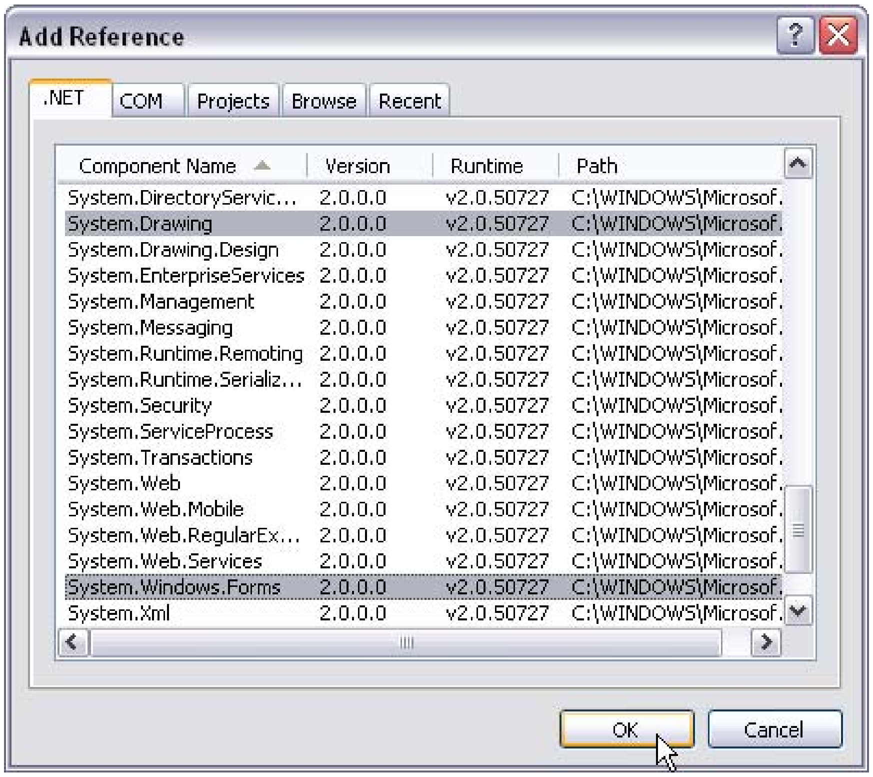
1. `System.Windows.Forms`
2. `System.Drawing`

Starting in Macro Manager, select the AppCSharp type and click Edit.

In the IDE, go to the Project Explorer. Be default it is docked on the right side of the display.

For the AppCSharp macro project, right-click on the References section and select Add Reference… from the menu.

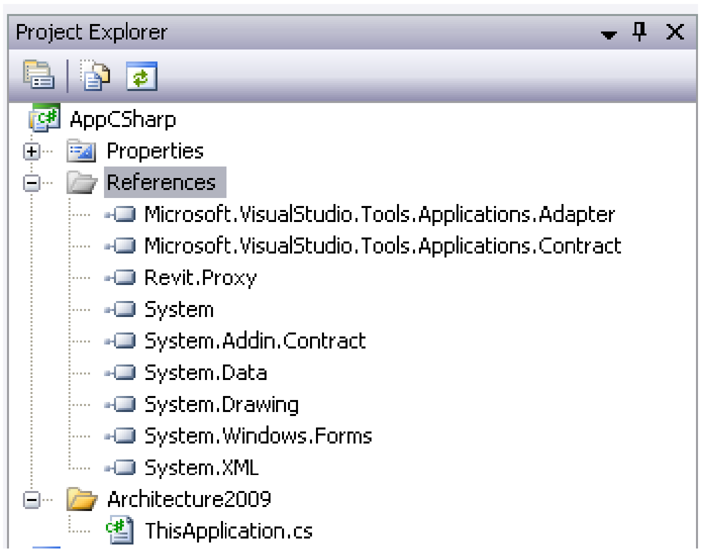
In the Add Reference dialog, find and select `System.Drawing` and `System.Windows.Form` on the list. Hold down the Ctrl key to do the multi-select operation. For example:



Component Name	Version	Runtime	Path
System.DirectoryServic...	2.0.0.0	v2.0.50727	C:\WINDOWS\Microsof.
System.Drawing	2.0.0.0	v2.0.50727	C:\WINDOWS\Microsof.
System.Drawing.Design	2.0.0.0	v2.0.50727	C:\WINDOWS\Microsof.
System.EnterpriseServices	2.0.0.0	v2.0.50727	C:\WINDOWS\Microsof.
System.Management	2.0.0.0	v2.0.50727	C:\WINDOWS\Microsof.
System.Messaging	2.0.0.0	v2.0.50727	C:\WINDOWS\Microsof.
System.Runtime.Remoting	2.0.0.0	v2.0.50727	C:\WINDOWS\Microsof.
System.Runtime.Serializ...	2.0.0.0	v2.0.50727	C:\WINDOWS\Microsof.
System.Security	2.0.0.0	v2.0.50727	C:\WINDOWS\Microsof.
System.ServiceProcess	2.0.0.0	v2.0.50727	C:\WINDOWS\Microsof.
System.Transactions	2.0.0.0	v2.0.50727	C:\WINDOWS\Microsof.
System.Web	2.0.0.0	v2.0.50727	C:\WINDOWS\Microsof.
System.Web.Mobile	2.0.0.0	v2.0.50727	C:\WINDOWS\Microsof.
System.Web.RegularEx...	2.0.0.0	v2.0.50727	C:\WINDOWS\Microsof.
System.Web.Services	2.0.0.0	v2.0.50727	C:\WINDOWS\Microsof.
System.Windows.Forms	2.0.0.0	v2.0.50727	C:\WINDOWS\Microsof.
System.Xml	2.0.0.0	v2.0.50727	C:\WINDOWS\Microsof.

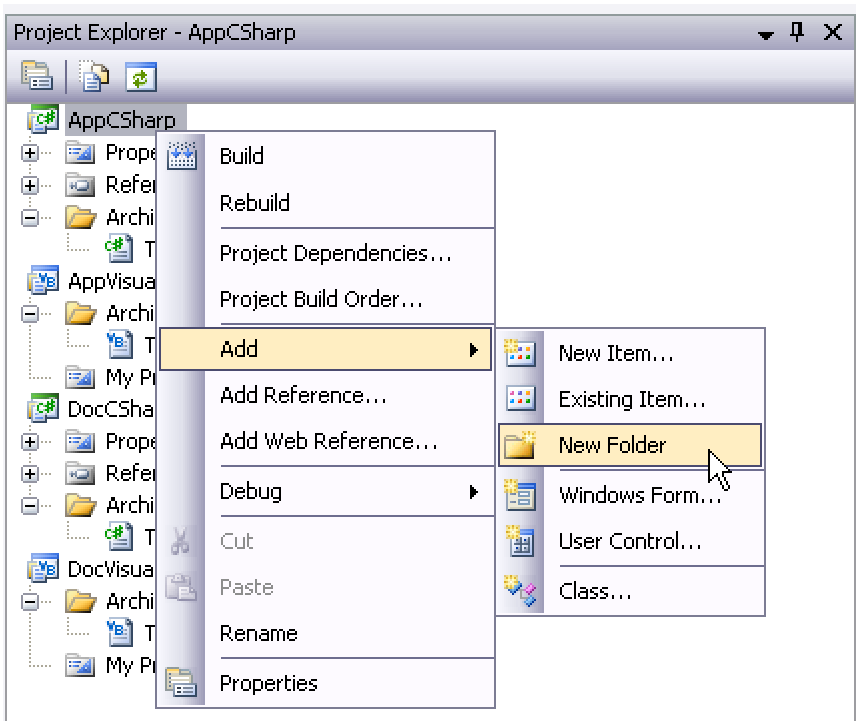
When you are ready, click OK.

The IDE’s Project Explorer is updated with the references:



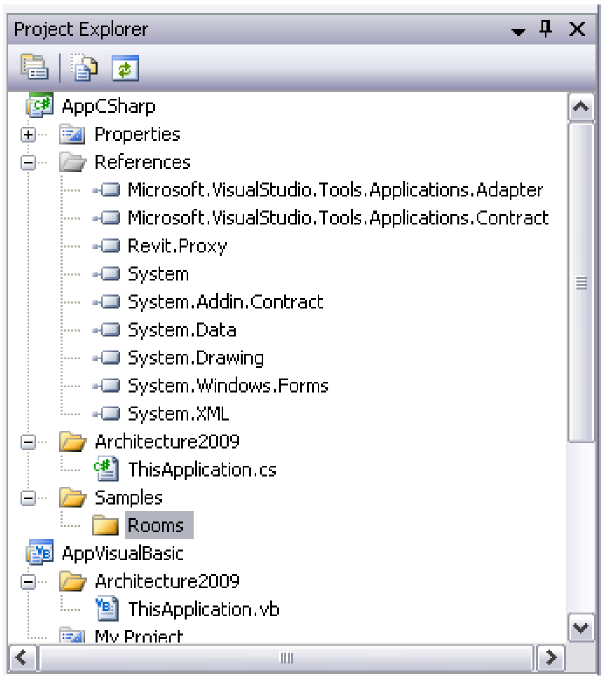
Create Samples Folder in Revit VSTA IDE

In the IDE’s Project Explorer, right-click on the AppCSharp macro project, and select Add > New Folder from the menu. For example:

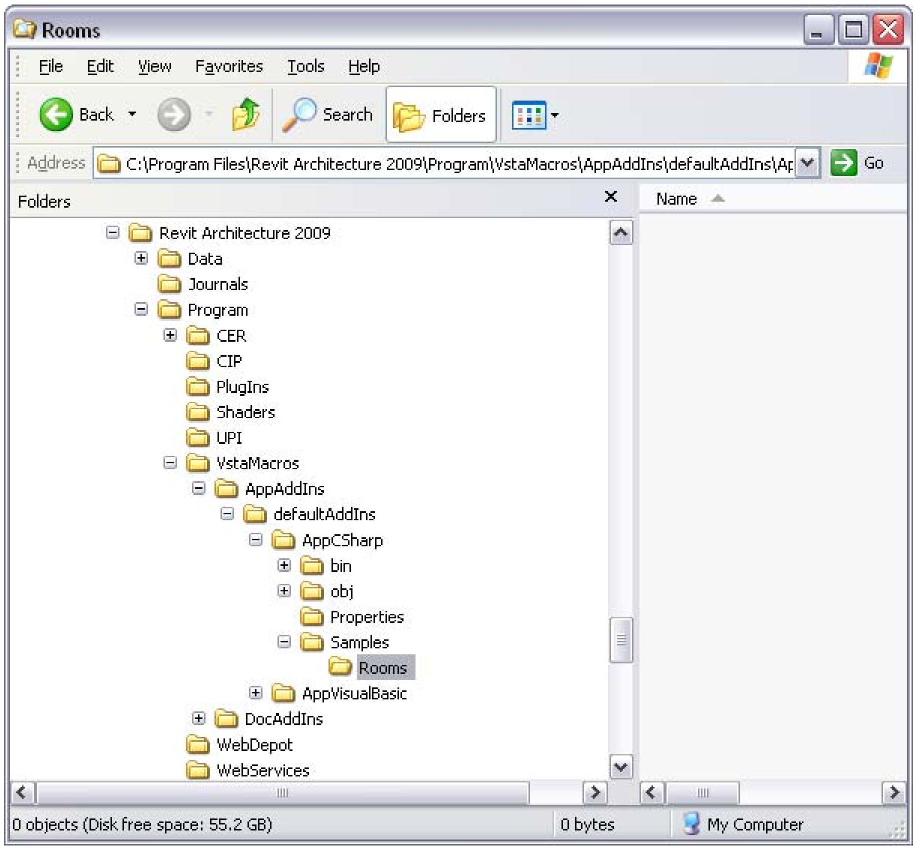


Name the folder Samples. Then right click on the Samples folder entry, and click Add > New Folder again. Name the secondary folder Rooms.

Your Project Explorer should look similar to the following:



Outside of the IDE, using Windows Explorer, navigate to the Revit installation folders on the file system and find the VstaMacros folders. Notice that the corresponding \Samples\Rooms\ folders have been created in this location. For example:



Copy Rooms Sample Files from Revit SDK to File System's Rooms Subfolder

Still outside of the Revit VSTA IDE, copy the Rooms sample's files from the unzipped Revit SDK into the Rooms subfolder created on your file system.   Here is an example for Revit Architecture, but your Revit product's path may be different:

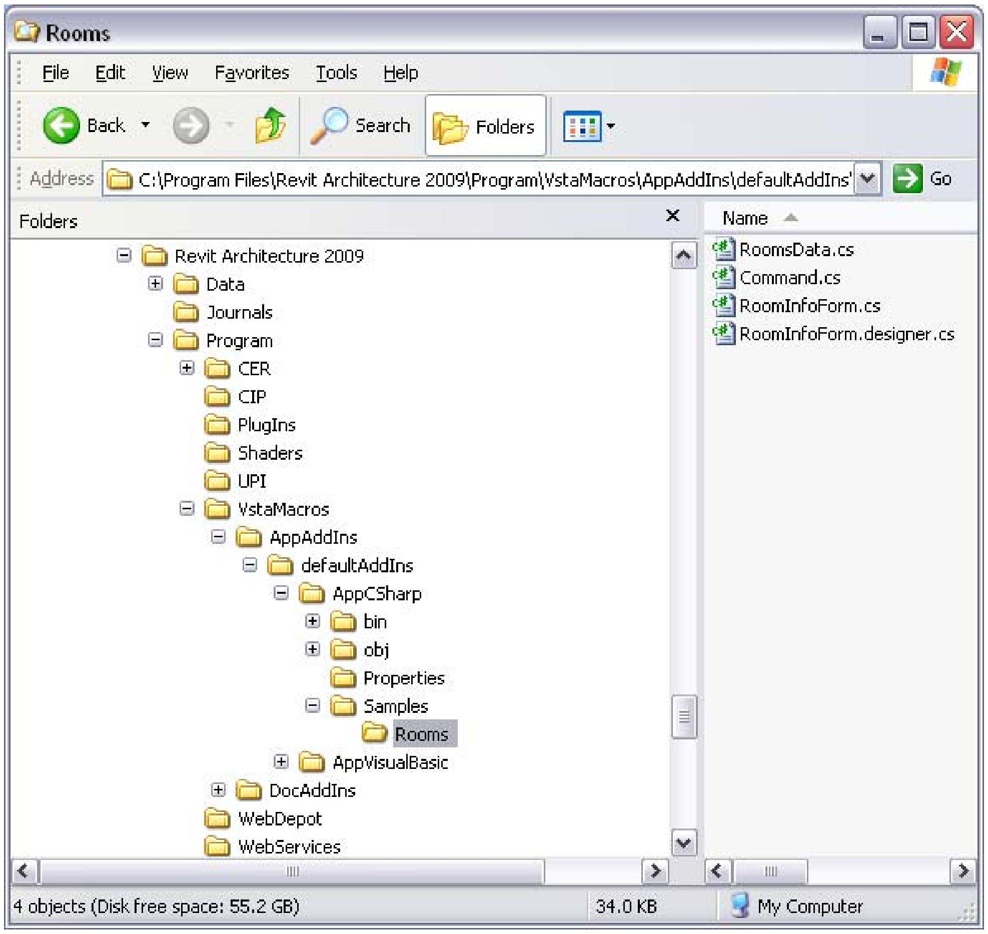
From:

```
C:\My Documents\Revit 2009 SDK\VSTA Samples\CS\Rooms\*
```

To:

```
C:\Program Files\Revit Architecture 
2009\Program\VstaMacros\AppAddIns\defaultAddIns\AppCSharp\Samples\Rooms\*
```

At this point, your Rooms subfolder on the file system should look similar to the following example:



C:\Program Files\Revit Architecture 2009\Program\VstaMacros\AppAddIns\defaultAddIns\

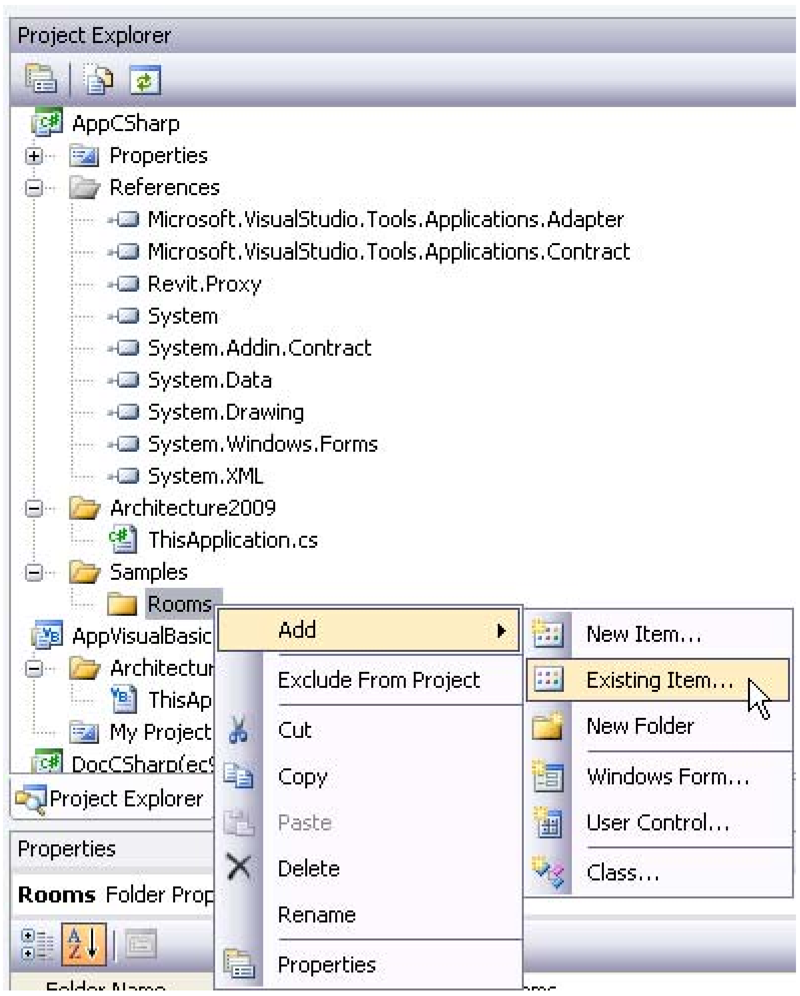
```
Revit Architecture 2009
  Data
  Journals
  Program
    CER
    CIP
    PlugIns
    Shaders
    UPI
    VstaMacros
      AppAddIns
        defaultAddIns
          AppCSharp
            bin
            obj
            Properties
            Samples
              Rooms
          AppVisualBasic
      DocAddIns
    WebDepot
    WebServices
```

```
RoomsData.cs
Command.cs
RoomInfoForm.cs
RoomInfoForm.designer.cs
```

4 objects (Disk free space: 55.2 GB) 34.0 KB My Computer

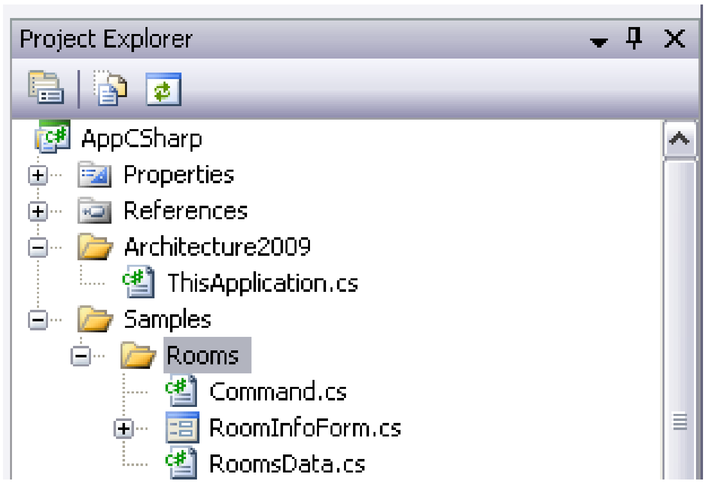
Add Existing Files to AppCSharp macro Project in Revit VSTA IDE

Return to the Revit VSTA IDE. In the Project Explorer, right-click on the AppCSharp > Samples > Rooms folder, select Add > Existing Item… from the menu:



Browse to the Rooms subfolder on the file system, under your Revit installation folder, select all the files that comprise the Rooms sample, and click Add.

The refreshed Project Explorer for AppCSharp contains:



Create and Build a RunSampleRooms Macro

You are now ready to create a macro that runs the Rooms sample. If you have been following the steps in prior sections, you are in the Revit VSTA IDE, and ThisApplication.cs is open in the editing window. You can add code for a macro that will run the Rooms sample.

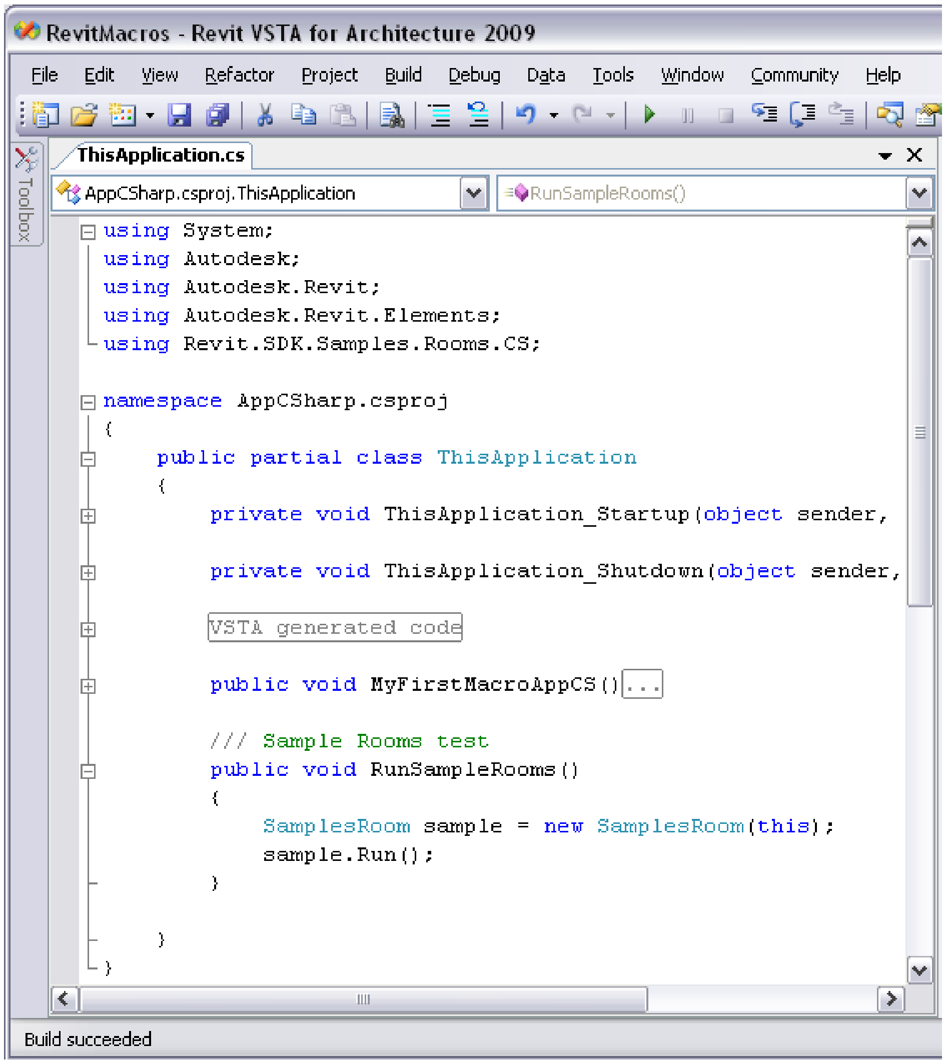
First, add the `using` directive for the sample's namespace:

```
using Revit.SDK.Samples.Rooms.CS;
```

Then add a method that is similar to the following:

```
        /// Sample Rooms test
        public void RunSampleRooms()
        {
            SamplesRoom sample = new SamplesRoom(this);
            sample.Run();
        }
```

Save the project and then click Build from the IDE toolbar menu. At this point, your ThisApplication.cs might look like the following example. We have collapsed the display of the `MyFirstMacroAppCS` created earlier:

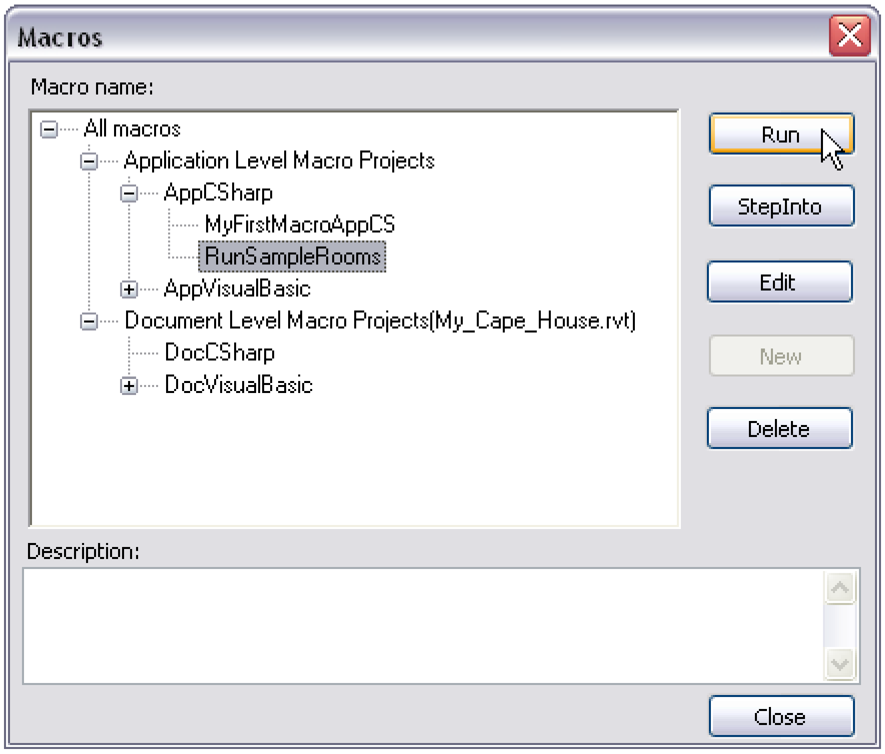

RevitMacros - Revit VSTA for Architecture 2009
File Edit View Refactor Project Build Debug Data Tools Window Community Help
ThisApplication.cs
Toolbox
AppCSharp.csproj.ThisApplication
RunSampleRooms()

```
using System;
using Autodesk;
using Autodesk.Revit;
using Autodesk.Revit.Elements;
using Revit.SDK.Samples.Rooms.CS;

namespace AppCSharp.csproj
{
    public partial class ThisApplication
    {
        private void ThisApplication_Startup(object sender,

        private void ThisApplication_Shutdown(object sender,

        VSTA generated code

        public void MyFirstMacroAppCS()...

        /// Sample Rooms test
        public void RunSampleRooms()
        {
            SamplesRoom sample = new SamplesRoom(this);
            sample.Run();
        }

    }
}
```

Build succeeded

Run the RunSampleRooms Macro

In Revit, launch Macro Manager and select RunSampleRooms from the categorized list. For example:

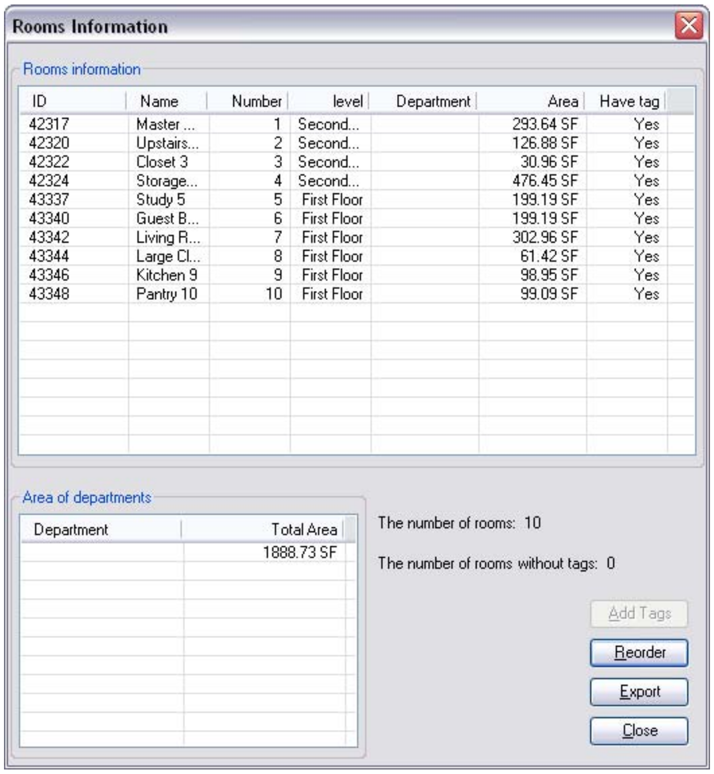


Macros

Macro name:

```
⊟ All macros
   ⊟ Application Level Macro Projects
      ⊟ AppCSharp
           MyFirstMacroAppCS
           RunSampleRooms
      ⊞ AppVisualBasic
   ⊟ Document Level Macro Projects(My_Cape_House.rvt)
        DocCSharp
      ⊞ DocVisualBasic
```

The macro may take several seconds to complete its processing and display the results in a form. Sample output:



Rooms Information

Rooms information

ID	Name	Number	level	Department	Area	Have tag
42317	Master ...	1	Second...		293.64 SF	Yes
42320	Upstairs...	2	Second...		126.88 SF	Yes
42322	Closet 3	3	Second...		30.96 SF	Yes
42324	Storage...	4	Second...		476.45 SF	Yes
43337	Study 5	5	First Floor		199.19 SF	Yes
43340	Guest B...	6	First Floor		199.19 SF	Yes
43342	Living R...	7	First Floor		302.96 SF	Yes
43344	Large Cl...	8	First Floor		61.42 SF	Yes
43346	Kitchen 9	9	First Floor		98.95 SF	Yes
43348	Pantry 10	10	First Floor		99.09 SF	Yes

Area of departments

Department	Total Area
	1888.73 SF

The number of rooms: 10

The number of rooms without tags: 0

Steps to Integrate Multiple VSTA Samples

Now that you know the steps to integrate one VSTA sample from the SDK to a macro project in the Revit VSTA IDE, this section describes the steps to integrate all the Revit VSTA samples.

These instructions assume that you completed the steps in the prior section, “Steps to Integrate One VSTA Sample: Rooms.” For example, it is assumed that you will work in the AppCSharp project, and that the required references (`System.Windows.Form` and `System.Drawing`) have been added already.

As noted earlier, the VSTA Samples are packaged with the Revit SDK. If you skipped a step, see the section, “Revit SDK, API Reference Documentation, and VSTA Samples,” in this topic.

Copy the Samples from the SDK to Your Revit Installation

First, outside of the Revit VSTA IDE, copy the remaining VSTA samples subfolders from the unzipped Revit SDK to the Samples subfolder on your file system, under the Revit installation. For example, copy the folders:

From:

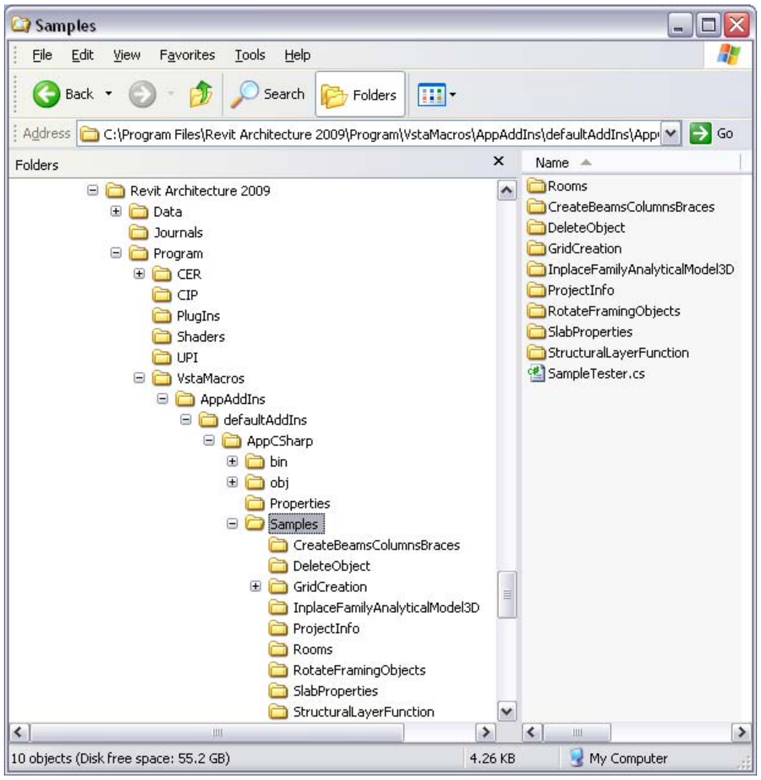
```
C:\My Documents\Revit 2009 SDK\VSTA Samples\CS\*
```

To:

```
C:\Program Files\Revit Architecture
2009\Program\VstaMacros\AppAddIns\defaultAddIns\AppCSharp\Samples\*
```

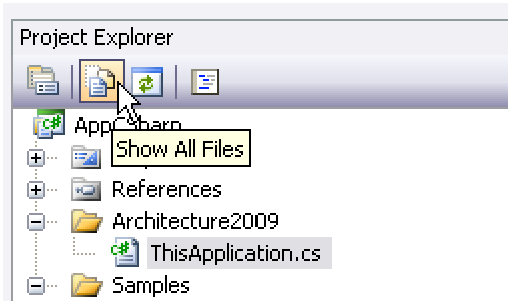
(Skip the copy operation for the Rooms subfolder that was copied in the prior example.)

After copying the files from the Revit SDK to the subfolder indicated above, your Windows Explorer view might look similar to the following:

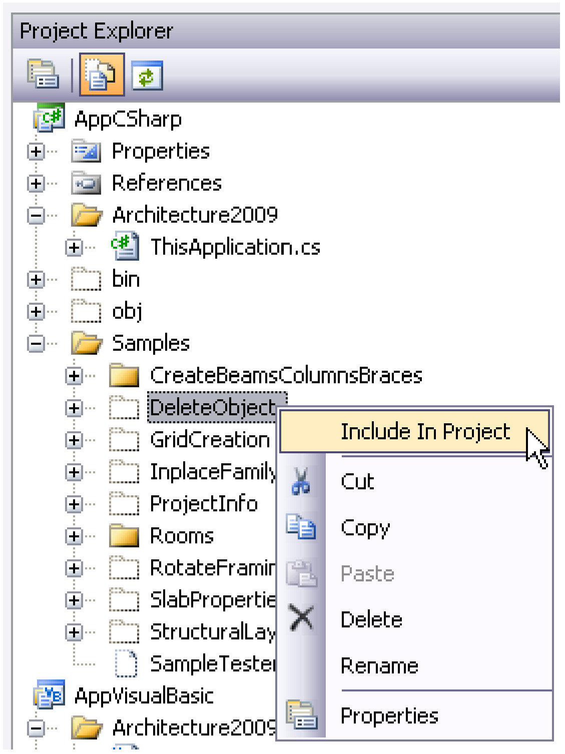


Include the Samples in the Macro Project

In the Revit VSTA IDE, enable the Show All files button in the Project Explorer. For example:



The newly copied samples subfolders are shown in Project Explorer. Because you already have the Rooms folder under Samples, you will need to right-click on each of the other sample subfolders and select Include In Project from the menu. For example:



(If the Rooms subfolder were not already under Samples, you could have right-clicked on the parent Samples folder in Project Explorer to use a single "Include In Project" option.)

Also include the SampleTester.cs class into the AppCSharp project.

Create a Macro to Use SampleTester and Build the Project

After you include all the VSTA Samples into the project, you can use `SampleTester` to run them individually. The following code shows one way to do this.

```
/// Sample Rooms test
public void SamplesTest()
{
    SampleTester samples = new SampleTester(this);
    //samples.RunCreateBeamsColumnsBraces();
    //samples.RunDeleteObject();
    samples.RunProjectInfo();
    //samples.RunRotateFamilyObjects();
    //samples.RunSampleRooms();
    //samples.RunSlabProperties();
    //samples.RunStructuralLayerFunction();
}
```

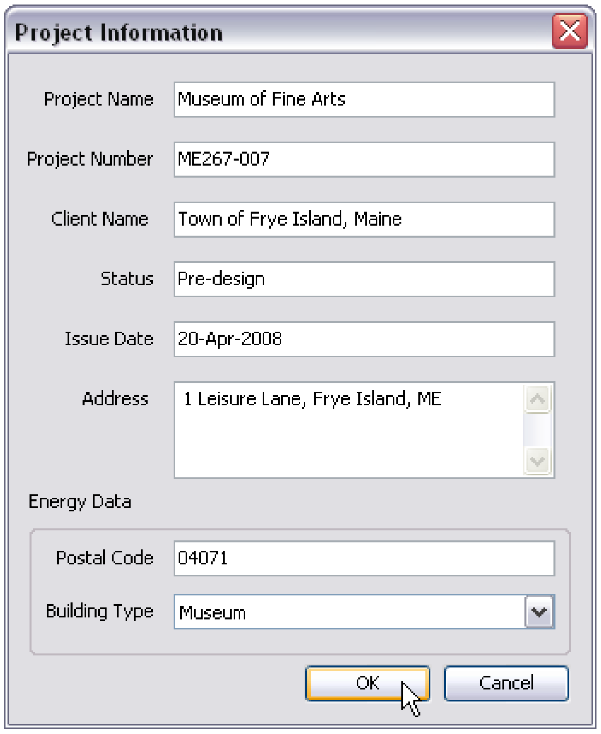
Again, this code assumes that you already followed the steps in the prior section, "Steps to Integrate One VSTA Sample: Rooms." Note how we are not running all the VSTA sample macros simultaneously; you would uncomment the particular sample that you want the macro to run.

Each sample may have an assumed element type (such as structural elements with Revit Structure), or a prerequisite action (such as selecting an object to be removed in your model, before trying the DeleteObject method). See the source code for each sample to understand the context for each macro.

In the Revit VSTA IDE, build the macro project. If you encounter errors, you may need to comment out the NewPathReinforcement method and namespace defined in the SampleTester.cs class. Then try building the AppCSharp macro project again.

Run the SamplesTest Macro

To run the SamplesTest macro, return to Revit, start Macro Manager, select SamplesTest from the AppCSharp list of project macros, and click Run. Here is sample output from the ProjectInfo macro, which loads a form that prompts for information about the current project:



Project Name: Museum of Fine Arts
Project Number: ME267-007
Client Name: Town of Frye Island, Maine
Status: Pre-design
Issue Date: 20-Apr-2008
Address: 1 Leisure Lane, Frye Island, ME

Energy Data
Postal Code: 04071
Building Type: Museum

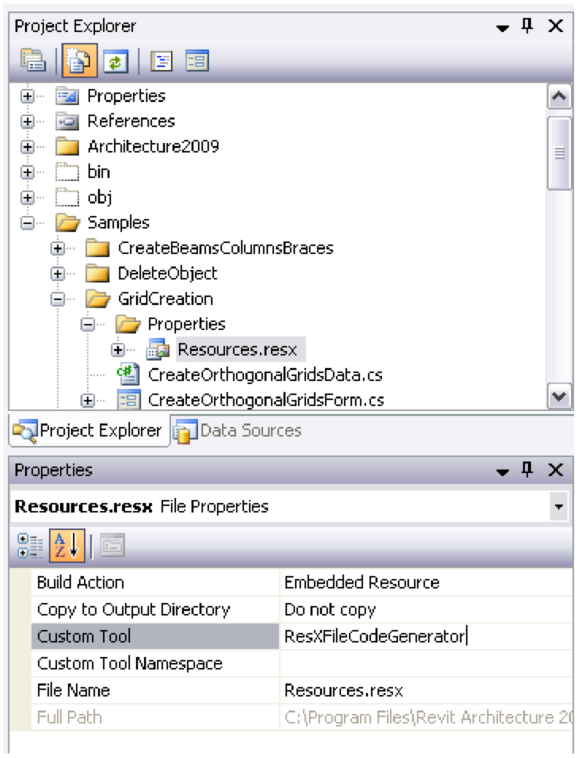
Caveat: GridCreation Sample

Note that one of the Revit VSTA samples, GridCreation, has a dependency on a resources.resx file. Before experimenting with the GridCreation sample, set the ResX file in the Revit VSTA IDE.

In Project Explorer, navigate to AppCSharp > Samples > GridCreation > Properties.

Highlight the resources.resx file.

In the Properties pane, select the Custom Tool property, and enter `ResXFileCodeGenerator` in the value column. For example:



Build Action	Embedded Resource
Copy to Output Directory	Do not copy
Custom Tool	ResXFileCodeGenerator
Custom Tool Namespace	
File Name	Resources.resx
Full Path	C:\Program Files\Revit Architecture 20

Frequently Asked Questions about Revit Macros

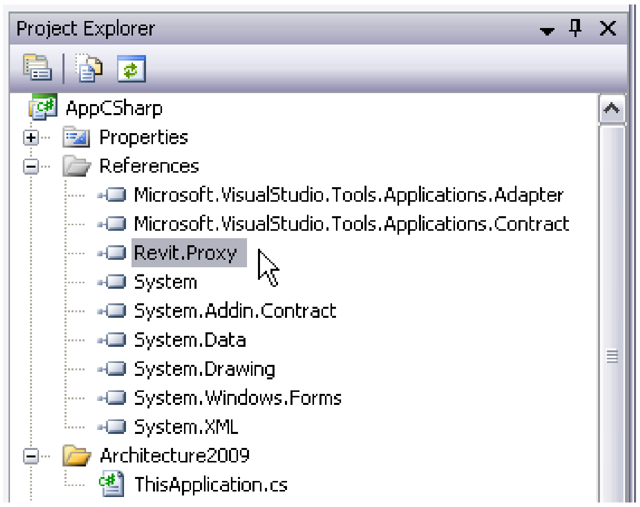
This section covers frequently asked questions about Revit macros.

Q: I was expecting to see my newly created macro listed in Macro Manager's categorized list, but it is not there. Why?

A: You must successfully build the macro project in the Revit VSTA IDE (use the Build menu) before your new macros will appear in Macro Manager.

Q: Do I need to add RevitAPI.dll as a reference when writing a new macro?

A: No. You do not need to reference Revit DLL files because this step was completed for you. A Revit VSTA macro project uses Revit.Proxy.dll as a required reference. Revit macros will fail if you delete this reference in the IDE. For example:



Project Explorer

```
AppCSharp
  Properties
  References
    Microsoft.VisualStudio.Tools.Applications.Adapter
    Microsoft.VisualStudio.Tools.Applications.Contract
    Revit.Proxy
    System
    System.Addin.Contract
    System.Data
    System.Drawing
    System.Windows.Forms
    System.XML
  Architecture2009
    ThisApplication.cs
```

Q: Do I need to edit my Revit.ini files?

A: No. Once you install Revit VSTA on your Revit product installation, Revit knows the API for the macro support.

Q: In the Revit VSTA IDE, I deleted a macro by removing its method in the This*.cs file or This*.vb file. However, the deleted macro's name still appears when I open the Macro Manager's categorized list again. How do I clear the name from the list?

A: You must build your edited project successfully before Macro Manager recognizes the removal.

Q: I opened a new Revit Project and tried to start the Macro Manager, but the menu item is disabled. How do I enable it?

A: Because macros can be embedded in documents, before starting the Macro Manager or launching the Revit VSTA IDE, you must first name and Save the project file.

Q: Why didn't anything happen when I selected File > New Project…" in the Revit VSTA IDE?

A: The IDE does not support creating a new macro project. The IDE provides four default macro projects:

- AppCSharp.csproj
- AppVisualBasic.vbproj
- DocCSharp.csproj
- DocVisualBasic.vbproj

You can write macros only in these four projects. Therefore, File > Open Project… in the IDE is disabled.

Q: What are the differences between Application-level and Document-level macros?

A: Application-level macros can be run on all opened Revit projects within a single instance of the Revit application. Document-level macro projects are stored within an .rvt file. They can be loaded from within the current active document and run on that document.

Q: How do I access the `Application` object or the `externalCommandData` equivalent?

A: All of the Application-level macros are associated with the `Application` object. In Application-level macros, the `this` keyword pointer (in C#) or the `Me` keyword pointer (in VB.NET) returns the API `Application` object.

In Document-level macros, the `this` keyword or the `Me` keyword returns the API `Document` object. To access the `Application` object from a Document-level macro, use `this.Application` or `Me.Application`.

Q: What should I include in the startup and shutdown methods: `ThisApplication_Startup`, `ThisApplication_Shutdown`, `ThisDocument_Startup`, and `ThisDocument_Shutdown`?

A: The `ThisApplication_Startup` method is called when Revit starts up and `ThisApplication_Shutdown` is called when Revit shuts down. Correspondingly, `ThisDocument_Startup` is called when a Revit project opens and `ThisDocument_Shutdown` is called when the project document closes. You can add your some initializing code in the `*_Startup` methods and do the cleanup work in `*_Shutdown` methods. For example, you can register event handlers in `*_Startup` methods and unregister them in `*_Shutdown` methods (this actually is the recommended way).

Q: How and why should I register and unregister my Revit event handler?

A: As noted previously, the recommended way to do this in Revit VSTA is to register event handlers in the `*_Startup` method and unregister them in `*_Shutdown` method. Every VSTA macro will be loaded and unloaded dynamically. When you debug (StepInto) a macro, if the event handler is not properly unregistered, Revit may call into a wrong method (maybe an invalid memory address). Although Revit VSTA may prevent Revit from crashing in this scenario, any event handlers that are not properly unregistered may cause performance issues during your current Revit session.

Q: I want to experiment with the Startup and Shutdown methods and an event handler. Can you show me an example?

A: The following sample code shows how to register an `OnDocumentNewed` event handler, which will automatically launch a message box when a new Revit project.

Important: This example requires that you have added `System.Windows.Forms` as a reference in the macro project, as shown in the instructions for the MyFirstMacroAppCS example. See "Example: Application-Level Macro in C#," but be sure to add the required reference for your language's corresponding macro project type.

VB.NET example, Application-level:

```
    Private Sub ThisApplication_Startup(ByVal sender As Object, ByVal e As System.EventArgs)
Handles Me.Startup

        AddHandler Me.OnDocumentNewed, AddressOf Me.ThisApplication_OnDocumentNewed

    End Sub


    Private Sub ThisApplication_Shutdown(ByVal sender As Object, ByVal e As System.EventArgs)
Handles Me.Shutdown

        RemoveHandler Me.OnDocumentNewed, AddressOf Me.ThisApplication_OnDocumentNewed

    End Sub


    Private Sub ThisApplication_OnDocumentNewed(ByVal document As Autodesk.Revit.Document)

        System.Windows.Forms.MessageBox.Show("VB.NET Application event OnDocumentNewed")

     End Sub
```

C# example, Application-level:

```
        private void ThisApplication_Startup(object sender, EventArgs e)
        {
            this.OnDocumentNewed += new
Autodesk.Revit.Events.DocumentNewedEventHandler(ThisApplication_OnDocumentNewed);
        }

        private void ThisApplication_Shutdown(object sender, EventArgs e)
        {
            this.OnDocumentNewed -= new
Autodesk.Revit.Events.DocumentNewedEventHandler(ThisApplication_OnDocumentNewed);
        }

        void ThisApplication_OnDocumentNewed(Document document)
        {
            System.Windows.Forms.MessageBox.Show("C# Application event OnDocumentNewed");
         }
```

Revit API Differences

The following table summarizes differences between the standard Revit API and the Revit Macro API.

Feature or Capability	ExternalCommand	Macro
Declaration	Must implement `IExternalCommand` interface and its `Execute` method.	Declare a public method with no parameters and void return type in the `ThisApplication` or `ThisDocument` class.
Application object	Access the Application object through `externalCommandData.Application`	As noted earlier, the `this` keyword in C# points to the `Application` object for Application-level macros, and points to the `Document` object for Document-level macros. (Use the `Me` keyword in VB.NET). For Document-level macros, `this.Application` points to the `Application` object (`Me.Application` in VB.NET).
`new` operator	Some Revit API objects can be created directly by `new` operator, such as `Geometry.XYZ`. For example: `XYZ a = new XYZ(1.0, 0.0, 0.0)`	The Revit API objects that can be created by `new` operator in external commands cannot be created by `new` operator in macros. Instead, they must be created by `Application.Create.NewXXX` methods.

Feature or Capability	**ExternalCommand**	**Macro**

		For example, in a C# Document-level macro:
		
		`XYZ a = this.Application.Create.NewXYZ(1.0, 0.0, 0.0);`
		
		In a VB.NET Document-level macro:
		
		`Dim a As Autodesk.Revit.Geometry.XYZ = Me.Application.Create.NewXYZ(1.0, 0.0, 0.0)`
		
		See the MyFirstMacro* examples in this topic.

Menus and toolbars	API external applications can create menus and toolbars for each external command through an external application.	Not supported.

Related Information about Revit Macros

To learn more, please refer to the following resources:

- Revit*API.chm is a Help file that contains the Revit API .NET reference documentation. The API reference documentation is provided with the Revit SDK, which is on the product DVD and the Autodesk web site: http://www.autodesk.com/revit-sdk. (**Revit 2009 Beta customers:** this URL is not active yet. See the Revit*API.chm that ships with the beta Revit 2009 SDK. Thank you.) Be sure to access the Revit API SDK for your release of Revit. As noted earlier in this topic, the SDK also includes the Revit VSTA samples.
- "DevTV: Introduction to Revit Programming," is a video on autodesk.com that you can download. The narrated video covers the Revit API for external commands and external applications. It does not describe the macros functionality with Revit VSTA, but should be of interest to developers who are looking for more details about the full Revit SDK API and its samples. See http://usa.autodesk.com/adsk/servlet/index?siteID=123112&id=2484975, and look for the section that starts with "DevTV:..." (**Revit 2009 Beta customers:** do not use the Revit 2008 SDK from this web page.)
- Autodesk Developer Network, http://adn.autodesk.com, has information and expert advice about the full Revit API. If you do not already have an ADN login account, please contact your Autodesk representative.