Revit Architecture 2009 User's Guide >

Creating Macros with Revit VSTA

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

Topics in this section

Revit Architecture 2009 User's Guide > Creating Macros with Revit VSTA >

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
- Tweaking geometry or parameters
- Creating many types of elements
- Importing and 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.

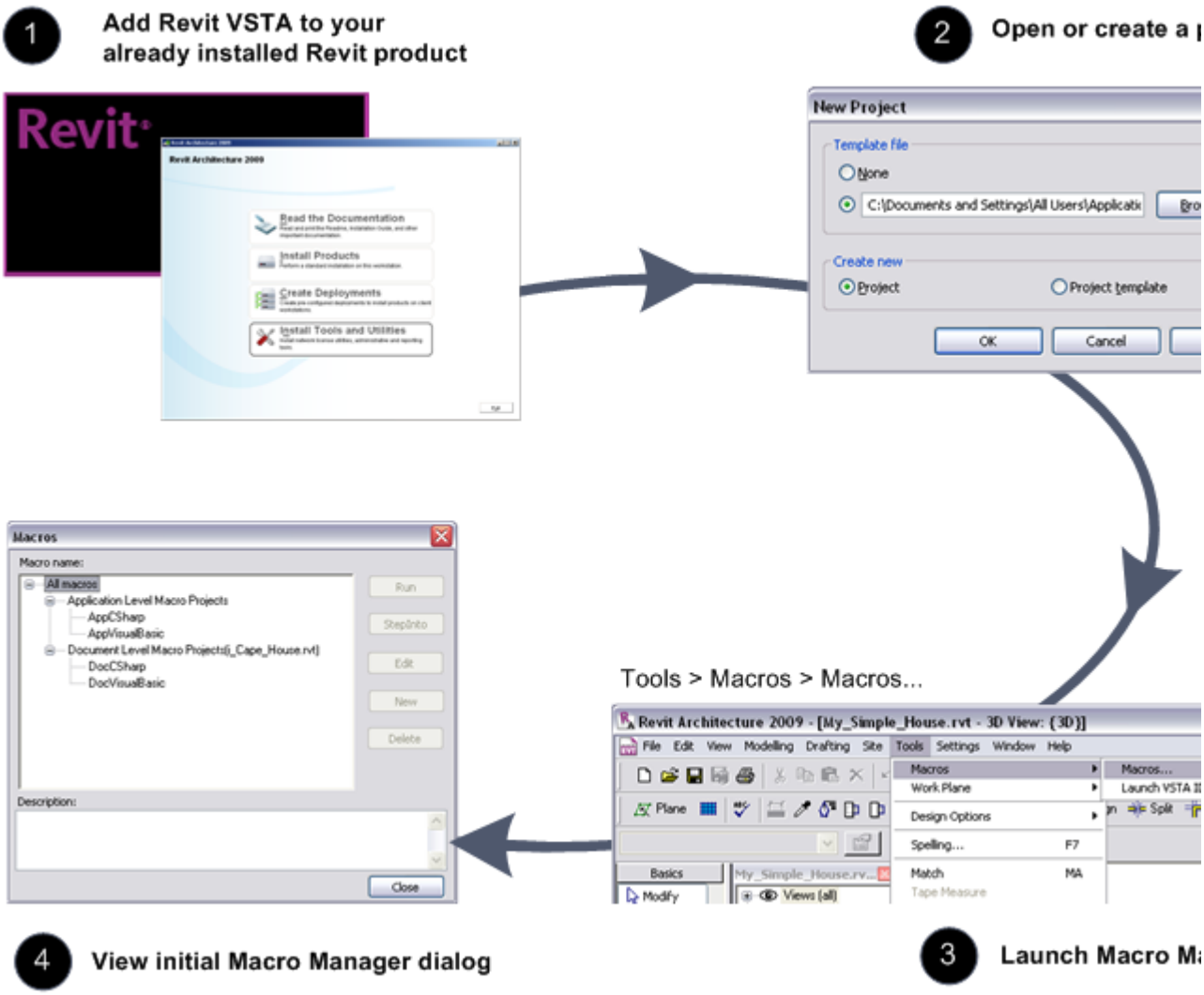
You can use the Revit VSTA macro features in all Revit products: Revit Architecture, Revit Structure, and Revit MEP. However, a separate Revit VSTA install, described in this topic, is required for each Revit product in which you want to run macros. In this topic, we refer to any of these products generically 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.

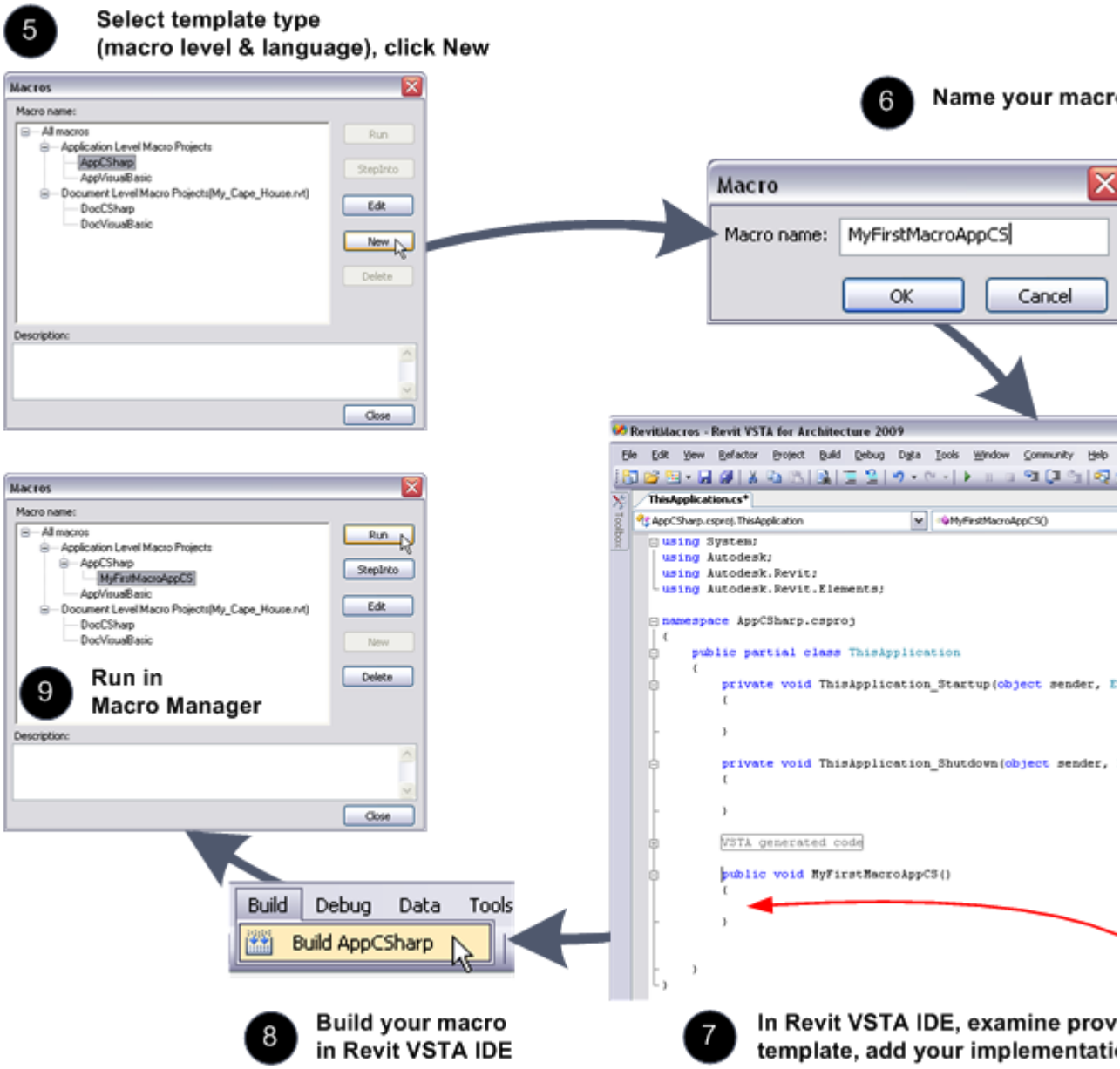
Workflow Overview: Initial Steps with Macros

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:



Now that we have introduced the overall process, let's look at the specific tasks.

Topics in this section

- Installing Revit VSTA

Revit Architecture 2009 User's Guide > Creating Macros with Revit VSTA > Getting Started with Macros >

Installing Revit VSTA

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

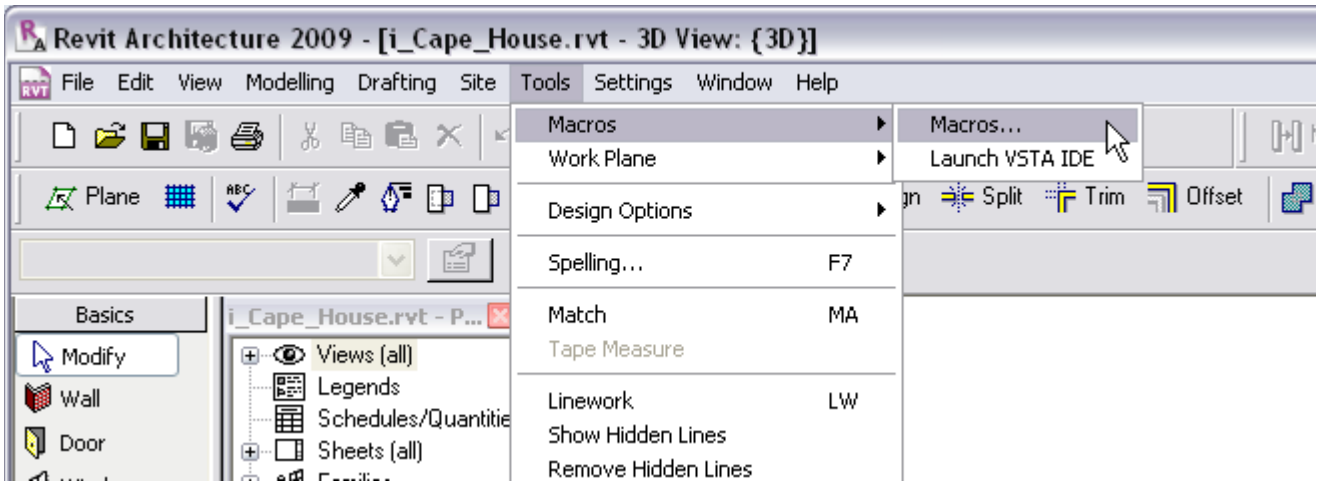
1. Check that Revit is not running on your computer. If it is, close Revit.
2. Start the Revit installation program again from the product DVD or download kit (the same program used to install the base Revit product).
3. On the main menu, select the option to Install Tools and Utilities, as shown here:

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

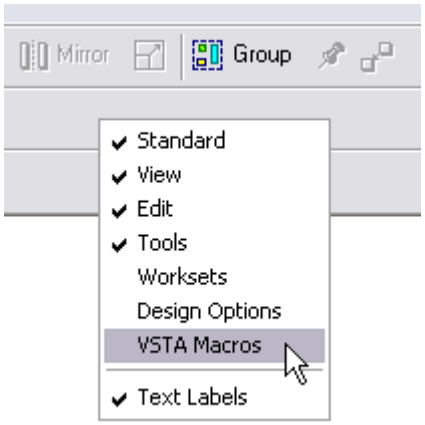
Follow the prompts, click Install, then click Finish on the final Revit VSTA screen.

Start Revit and Notice Updated Tools Menu for Macros

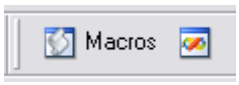
After you install Revit VSTA, start the base Revit product. Open an existing project or create a new one. Notice that the Tools menu contains a Macros submenu. Here is an example from Revit Architecture:



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:



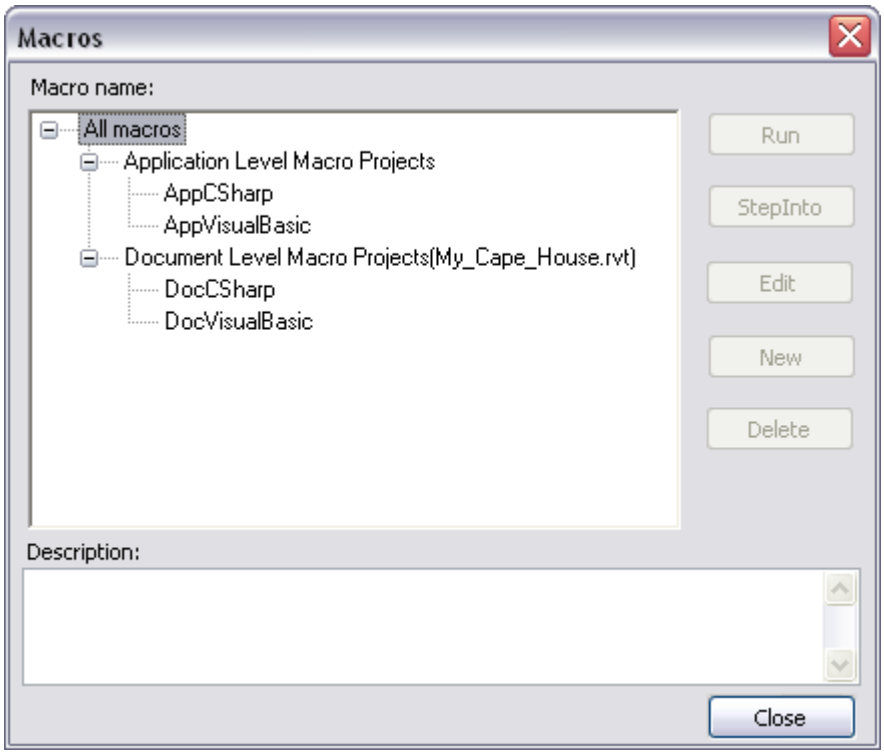
Revit Architecture 2009 User's Guide > Creating Macros with Revit VSTA >

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

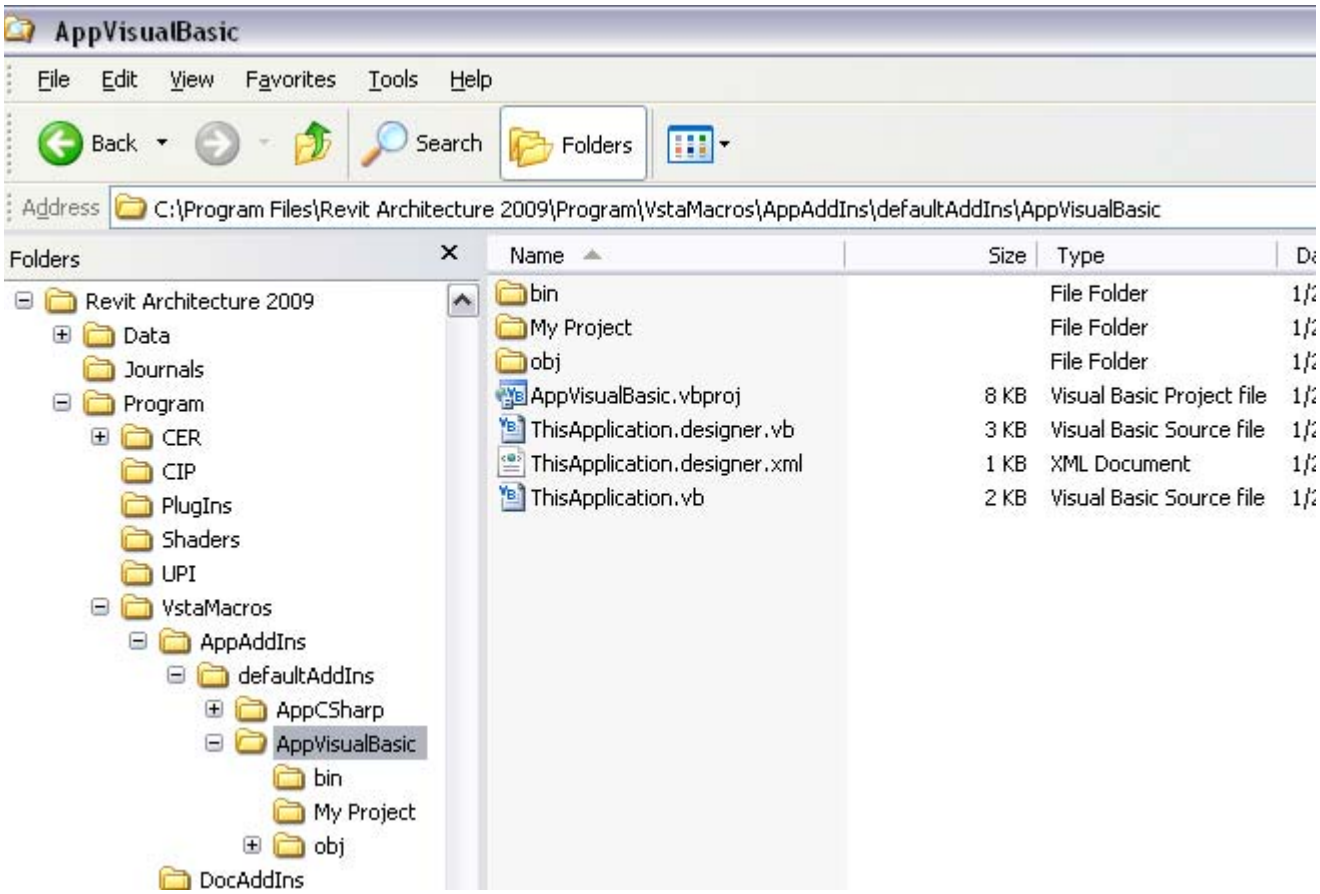
Macro Project File Locations

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.



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.

Topics in this section

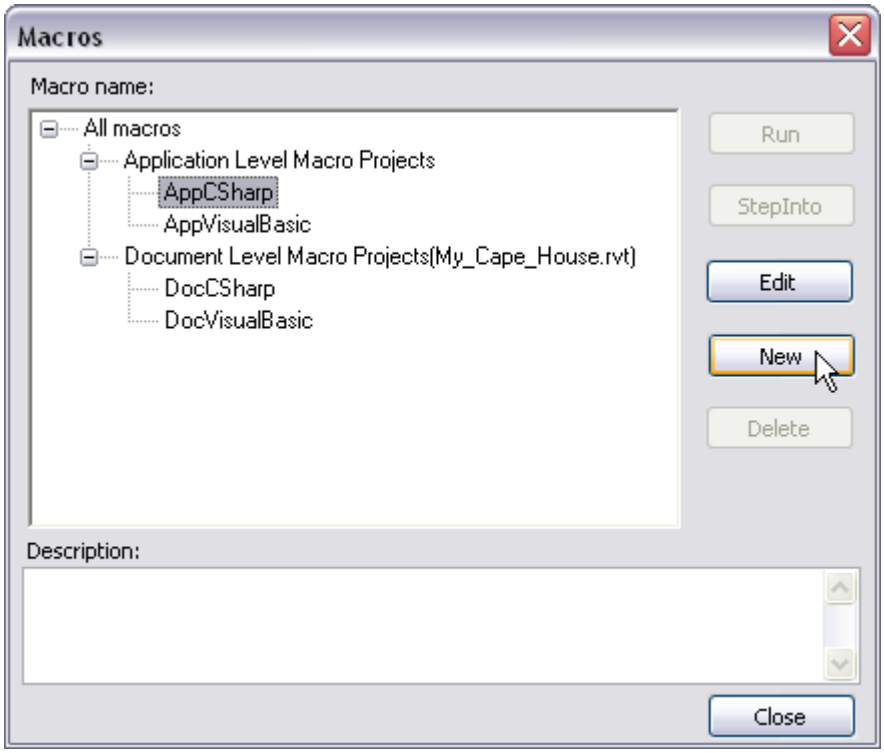
- **Creating Your First Set of Macros**
- **Using the StepInto Option**

Revit Architecture 2009 User's Guide > **Creating Macros with Revit VSTA** > **Using Macro Manager and the Revit VSTA IDE** >

Creating Your First Set of Macros

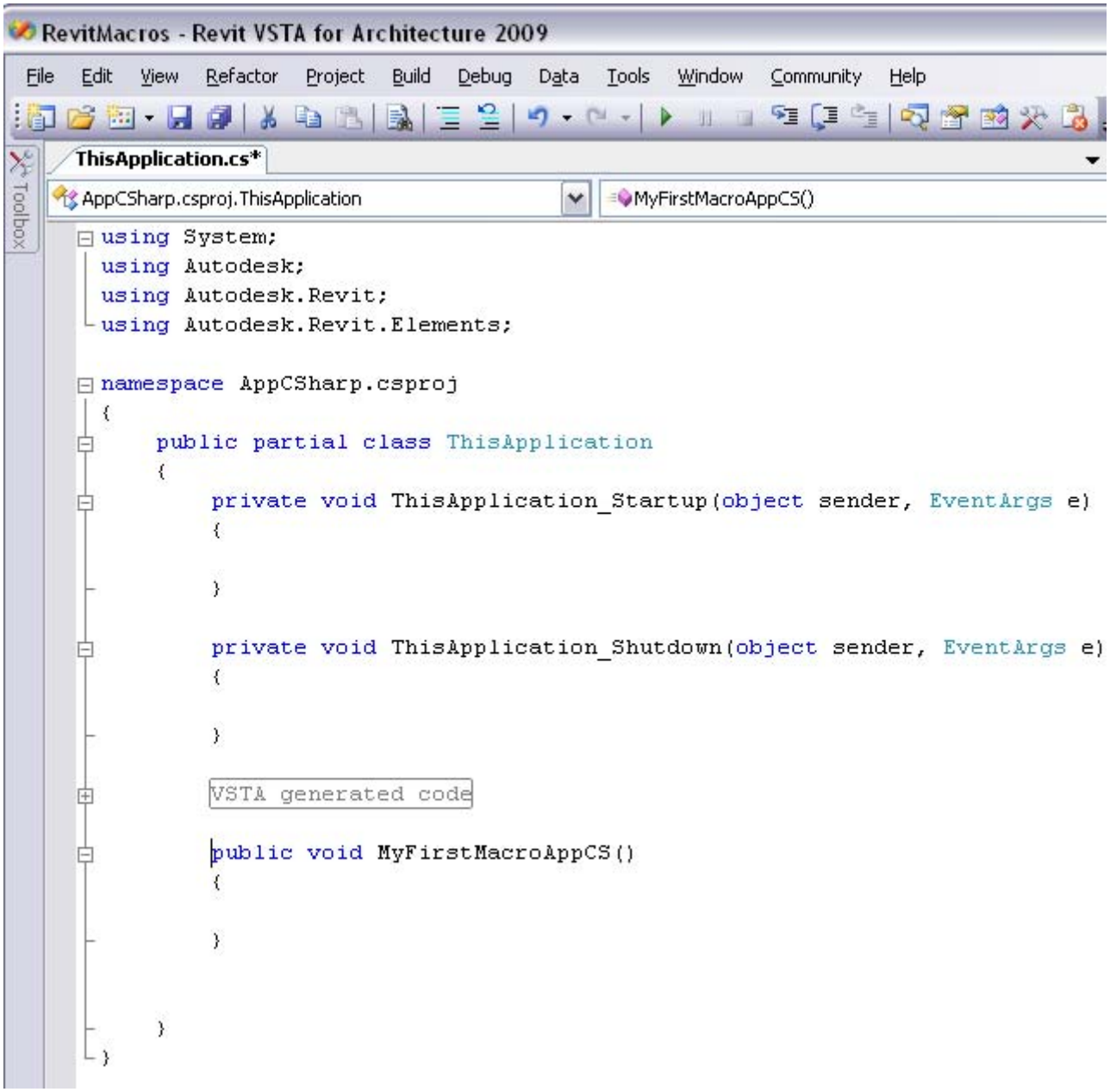
After you decide on the desired level and language for your macro, highlight that type on the Macro Manager list and then click New.

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:

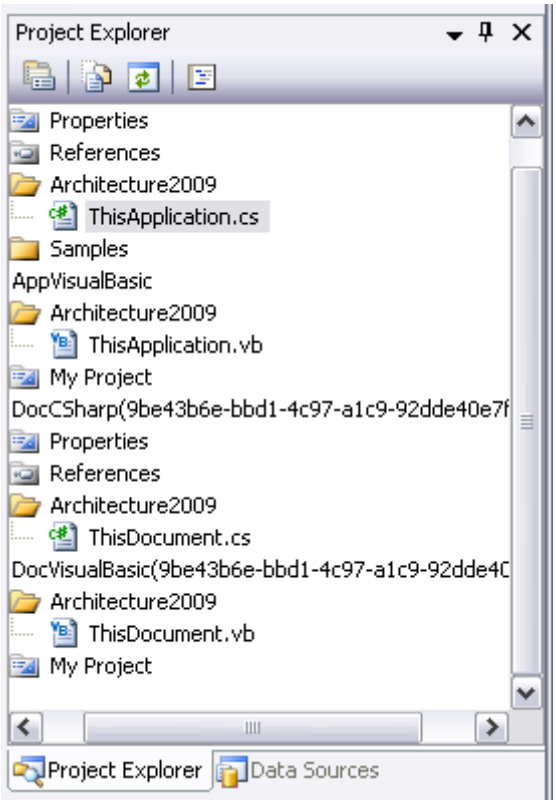
The Revit VSTA IDE presents a starting template for Application-level macros in C#. Here is a portion of the screen.



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 IDE, in the main 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, lineWid
```

```
}
```

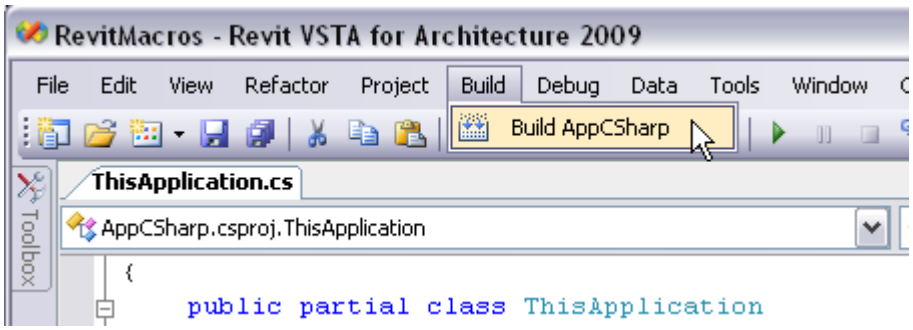
In the example, the Revit API Geometry.XYZ class is used to define a position (with X, Y, Z coordinates) for a Text Note that the macro will add to the active view of the active document.

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:



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.

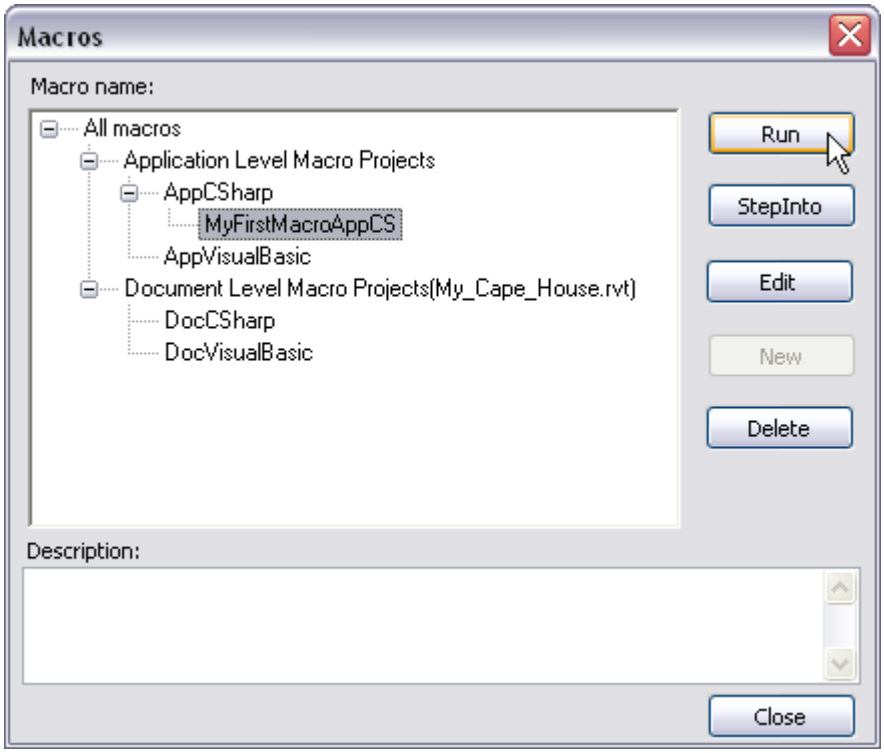
Tip After editing a macro's code, remember to build it before opening the Macro Manager again to run it. If your code is set up correctly, you should see a Build Succeeded message in the lower-left corner (by default) of Revit VSTA. You do not build macros in the Macro Manager.

Run Macros in Macro Manager

Next, return to Revit. To see the 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.

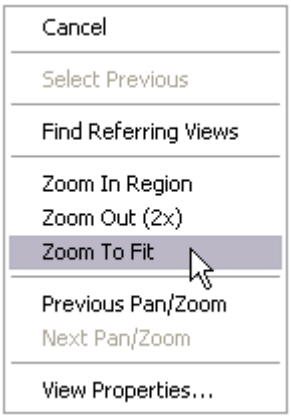
```
View pView = this.ActiveDocument.ActiveView;

this.ActiveDocument.Create.NewTextNote(pView, origin, baseVec, upVec, lineWidth, a
```

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

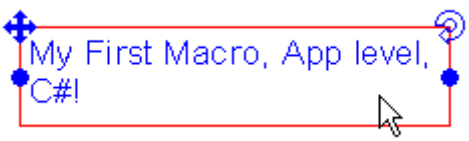


Observe the Macro's Results

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

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.

Tip 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

     Autodesk.Revit.Geometry.XYZ upVec = this.Application.Create.NewXYZ(0.0, 0.0,

     Autodesk.Revit.Geometry.XYZ origin = this.Application.Create.NewXYZ(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, strT

}
```

Notice the use of `this.Application` in this Document-level macro. For related information, see the previous section about accessing the `Application` object from Document-level macros.

Tip 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

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

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

End Sub
```

Tip 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.

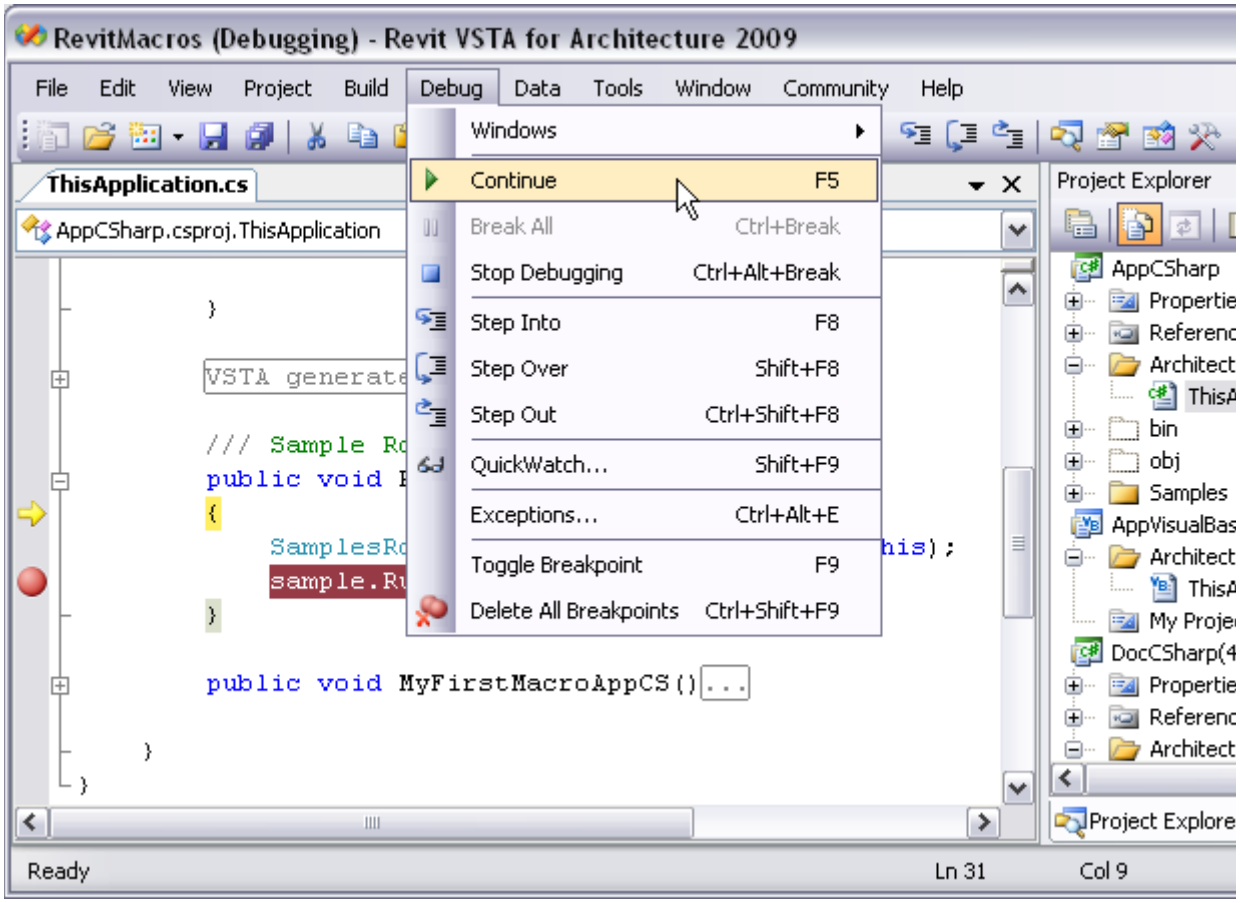
To run your newly built macro, select it in Macro Manager and click Run. Then if necessary, right-click in the active view and select Zoom to Fit from the menu to see the Text Note added by your macro.

Revit Architecture 2009 User's Guide > Creating Macros with Revit VSTA > Using Macro Manager and the Revit VSTA IDE >

Using the StepInto Option

You can debug your macros by using StepInto option in Macro Manager and features in the Revit VSTA IDE:

- Open the code for your macro in the IDE.
- To set breakpoints, click in the left margin of the code window, or click in the code and press F9 or Toggle Breakpoint in the Debug menu.
- Then switch to Macro Manager. Select your macro from the categorized list, and click StepInto. The macro will stop at the first line.
- Press F5 or Continue to move to the breakpoint.



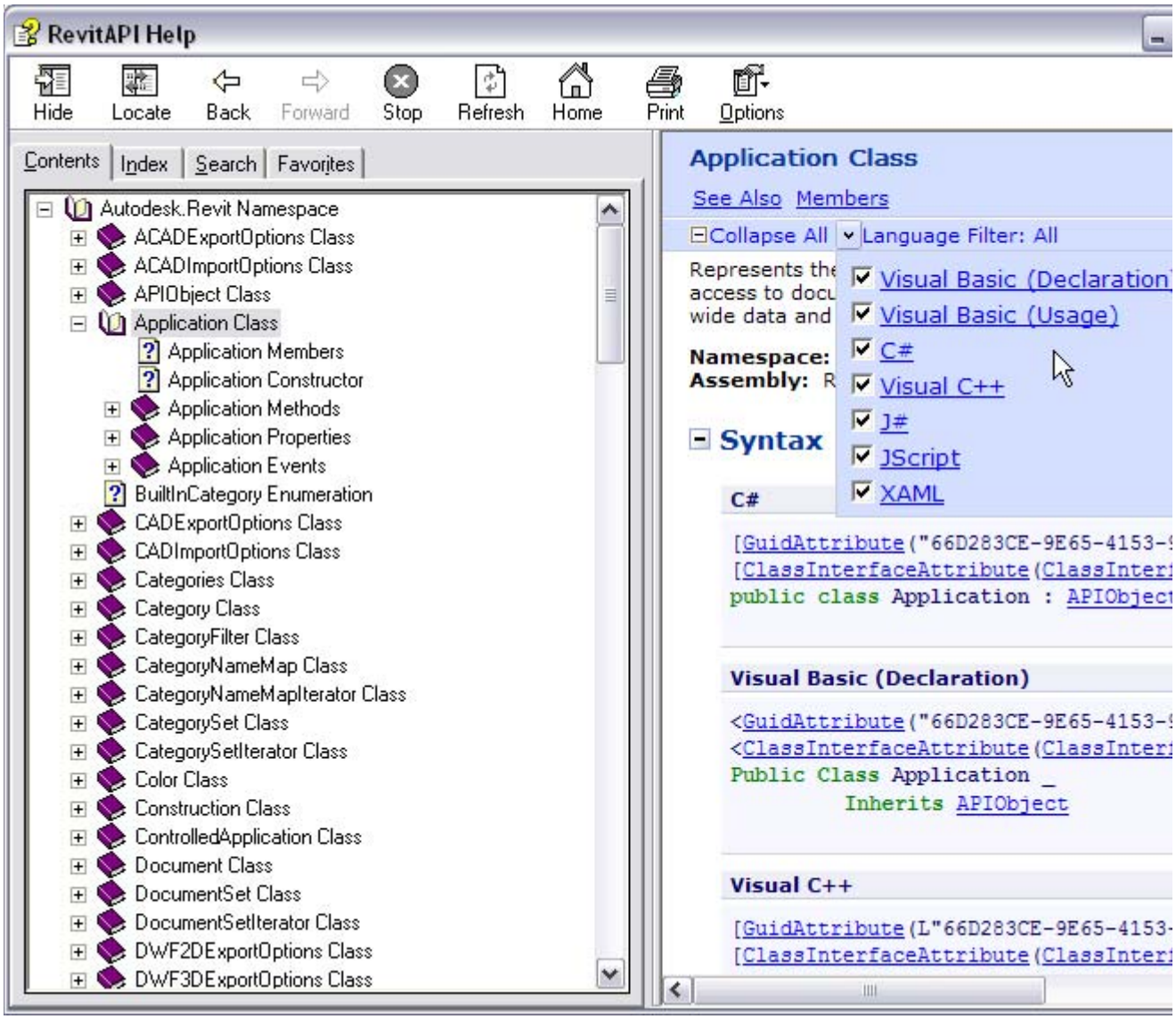
In the Revit VSTA IDE, you can use Watch, Locals, and other options to perform debugging tasks like checking variable values. For more debugging information, see the VSTA IDE Help.

Revit SDK, API Reference Documentation, 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**.
- The Autodesk Developer Network, **http://adn.autodesk.com**. If you are interested, please contact your Autodesk representative for information about getting an ADN account.

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:



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

\Revit <release> SDK\VSTA Samples\...

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

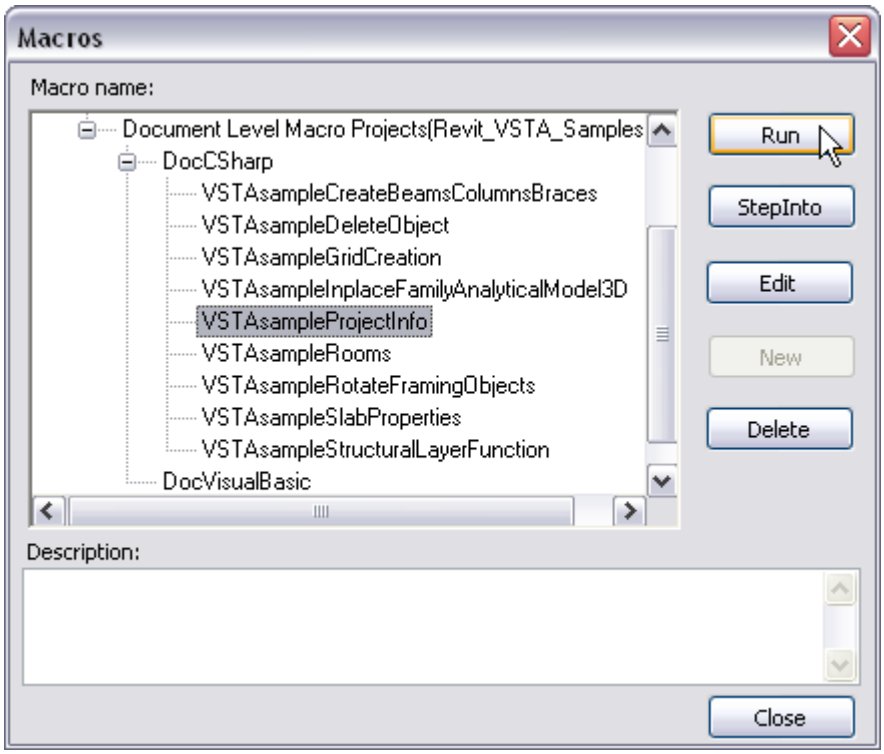
Revit Architecture 2009 User's Guide > Creating Macros with Revit VSTA >

Using the Revit VSTA Samples from SDK

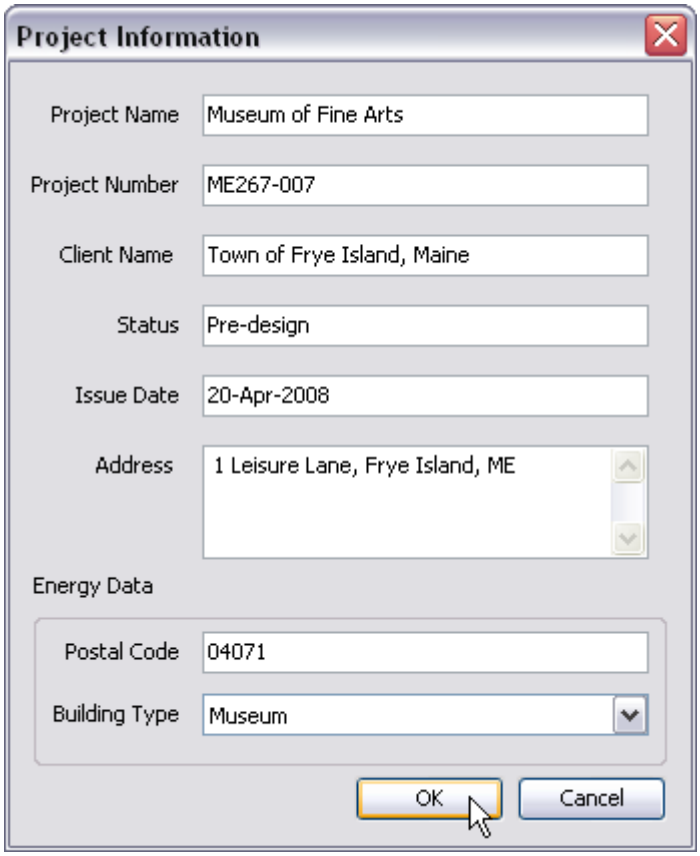
You can learn various API techniques by using the Revit VSTA samples provided with the SDK. Copy and open the following Revit project file:

\Revit <release> SDK\VSTA Samples\Revit_VSTA_Samples.rvt

Included in this .rvt are a number of Document-level macros. In Revit, start Macro Manager, select one of the Document-level macros, and choose Run. Note that when you open Revit_VSTA_Samples.rvt from the SDK, it may contain more samples than the ones shown in the following screen, and the macro methods may be named differently.



These macros have been preconfigured to run in Revit, assuming that you have already installed the Revit VSTA add-in. In this example, the ProjectInfo sample prompts you to enter information about the project, as in the following dialog box.



In Macro Manager, you can also select any of the VSTA sample macros and click Edit to see the code created to run that sample.

If you need to integrate an Application-level sample macro into the IDE, or learn how to add new types of references or properties for your macros, see the next section.

Revit Architecture 2009 User's Guide > **Creating Macros with Revit VSTA** >

Integrating Macros into Revit VSTA

Let's walk through the steps to integrate a macro into a Revit VSTA project. You can skip this section if the Document-level samples described in the section **Using the Revit VSTA Samples from SDK** meet your needs.

This section explains where to create folders in the IDE projects that correspond to resources on the file system, how to add required references, and how to define properties. Those steps were done for you in the macros that were built into Revit_VSTA_Samples.rvt, which is provided on the SDK.

Add Required References

If your macro presents a user interface, you will need to add required references to your project. For example, in the Rooms sample, we need to reference:

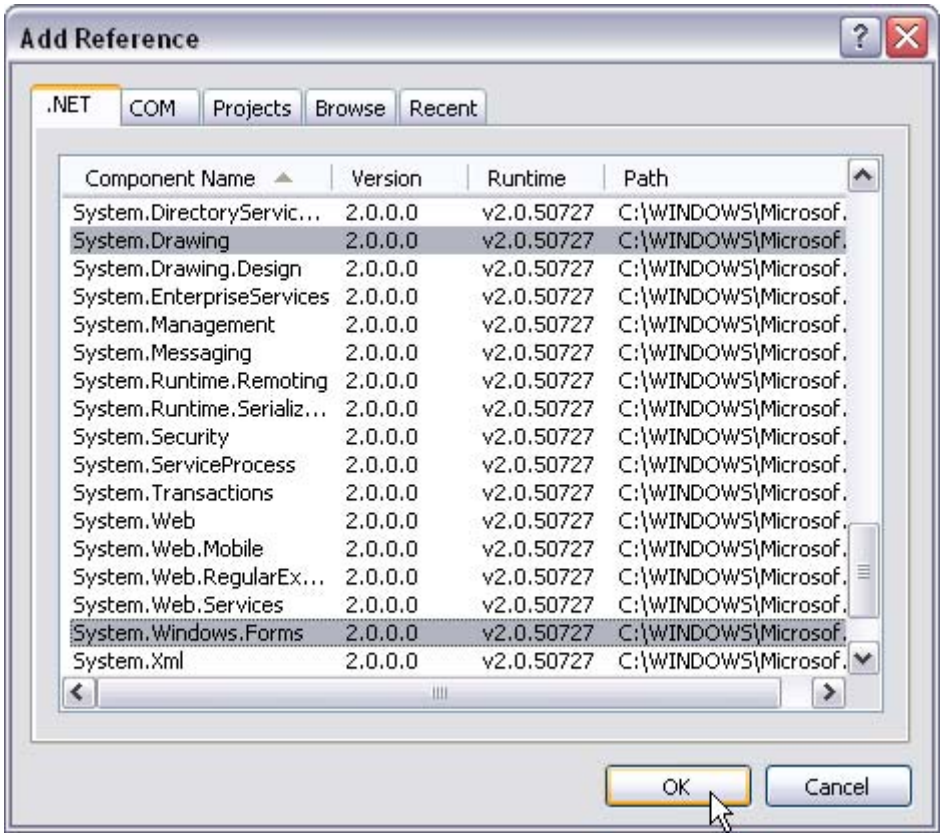
1. `System.Windows.Forms`
2. `System.Drawing`

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

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

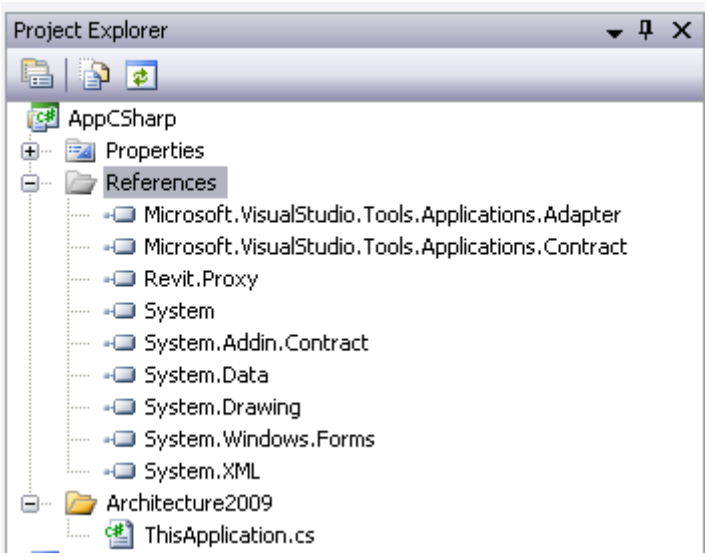
For the macro project (example: `AppCSharp`), right-click on the References section and select Add Reference from the menu.

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



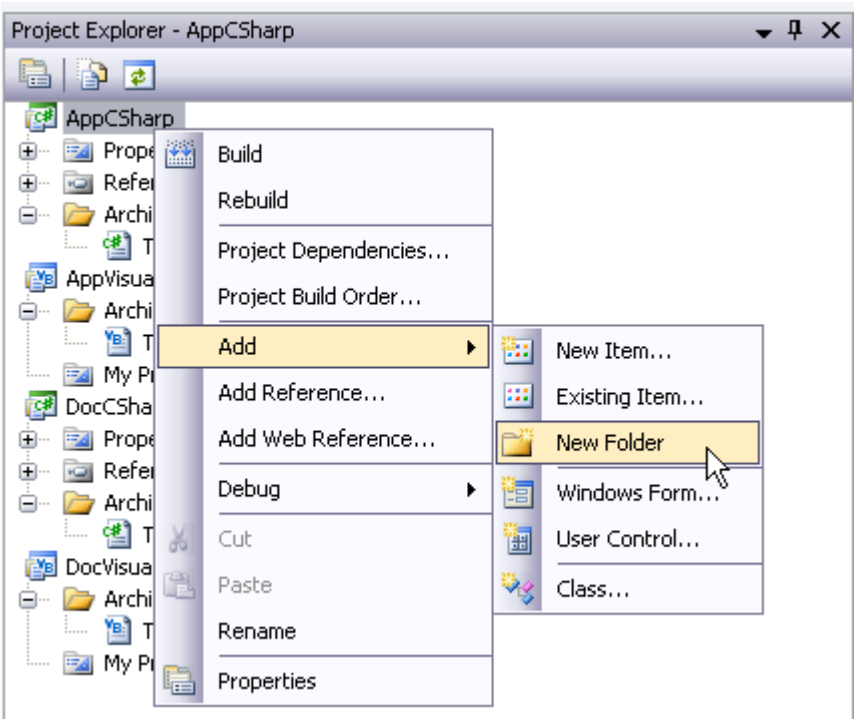
When you are ready, click OK.

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



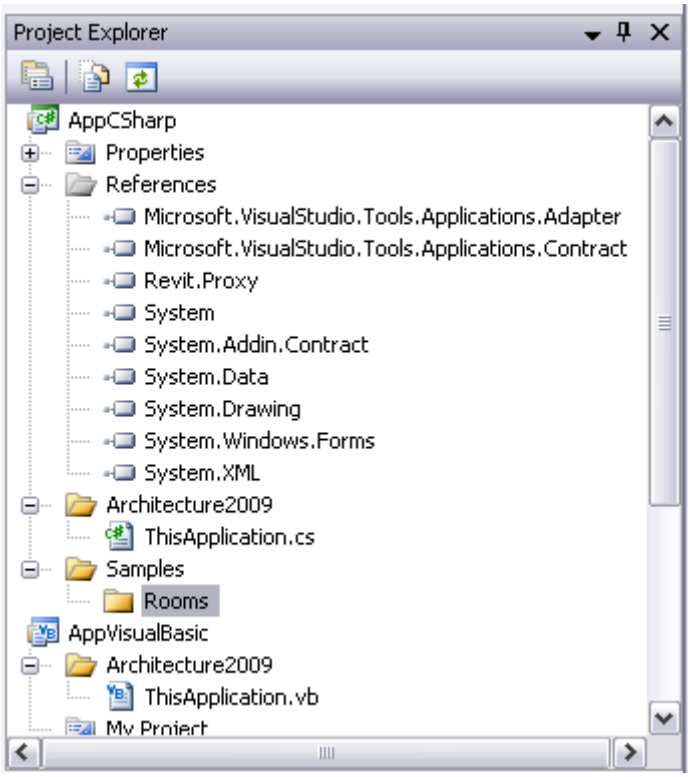
Create Folders in Revit VSTA IDE

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

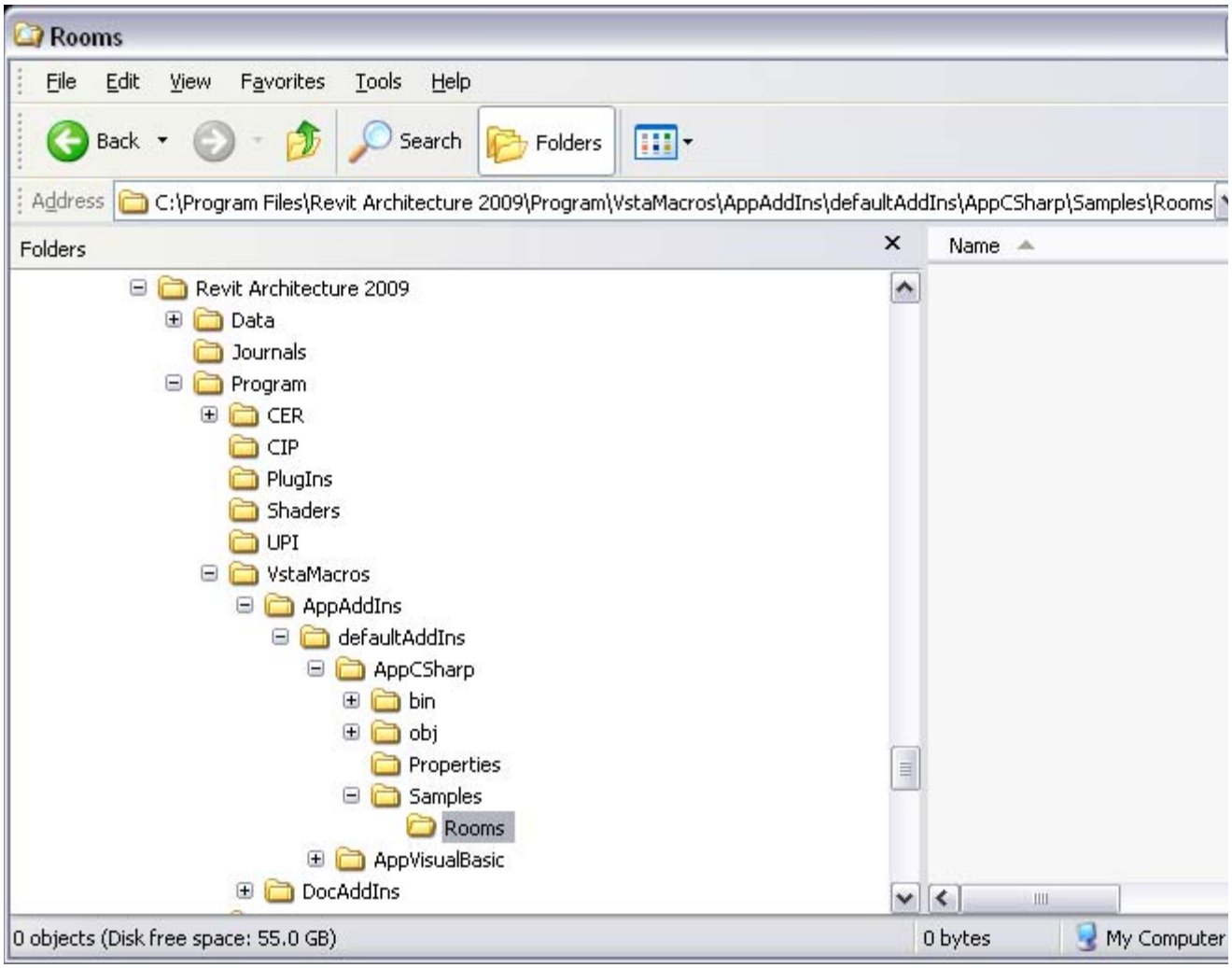


Name the folder; in this example, we call it Samples. Then right click on the Samples folder entry, and click Add ➤ New Folder again to add a secondary folder, such as Rooms.

Here is the resulting Project Explorer screen:

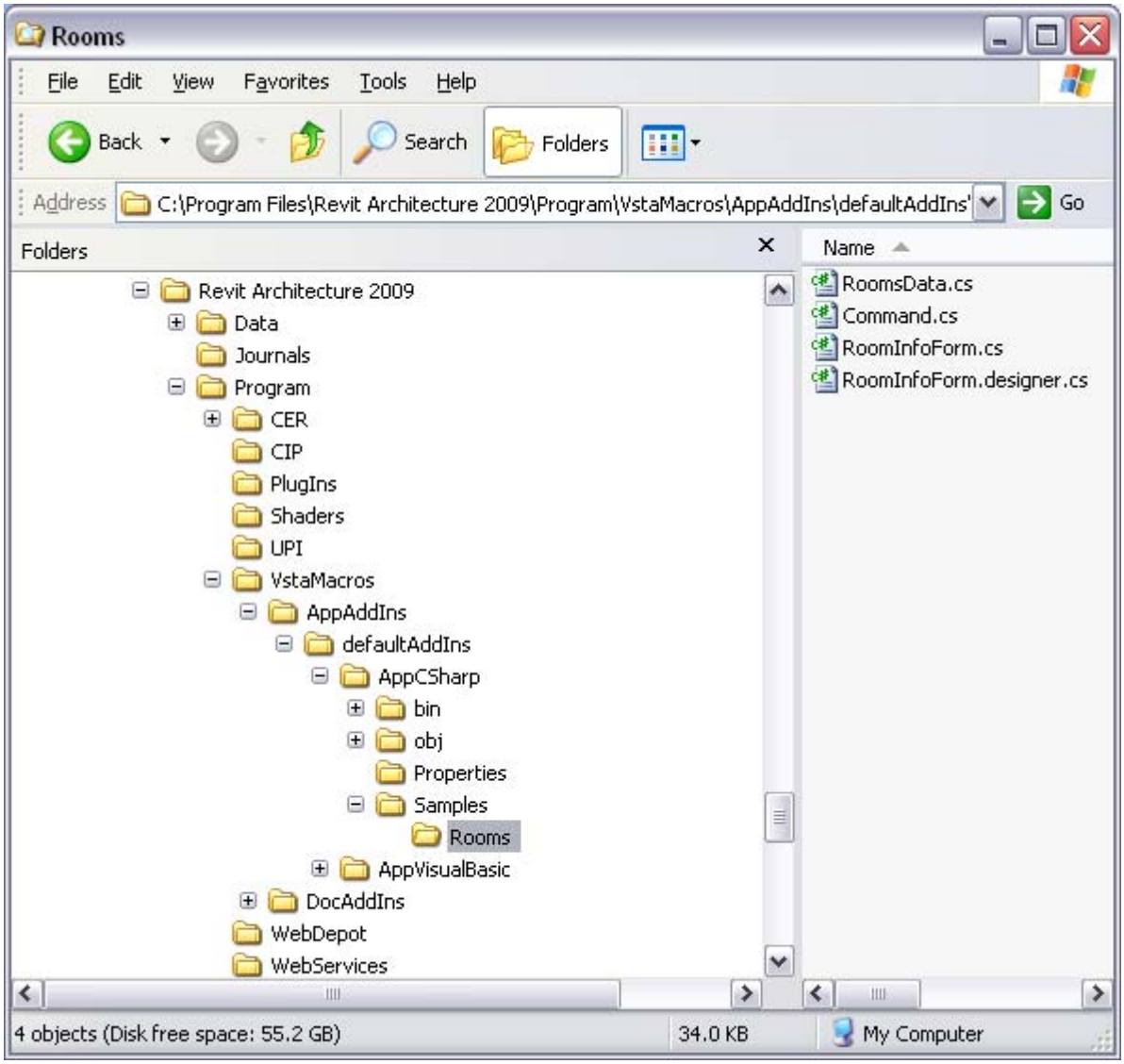


Outside of the IDE, using Windows Explorer, navigate to the Revit installation folders on the file system and find the VstaMacros folder. Notice that the corresponding `\Samples\Rooms` subfolders have been created in this location. For example:



Copy Your Macro to File System Folder

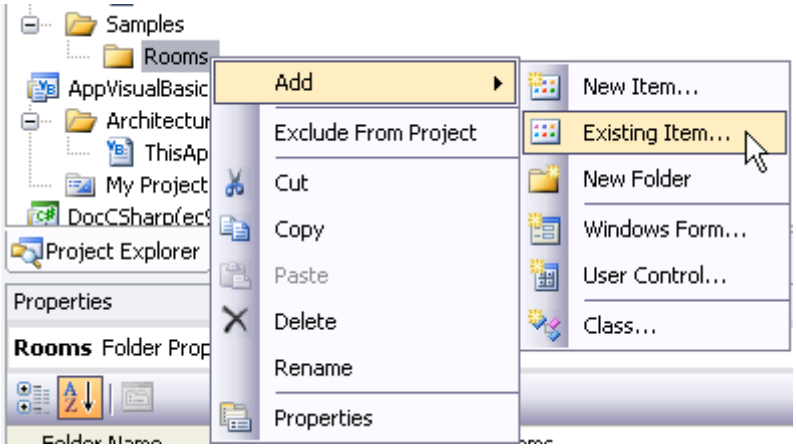
Still outside of the Revit VSTA IDE, copy your macro's files to the subfolder you created on the file system. In our example, we now have:



Note If your macro uses a .resx file, copy it too.

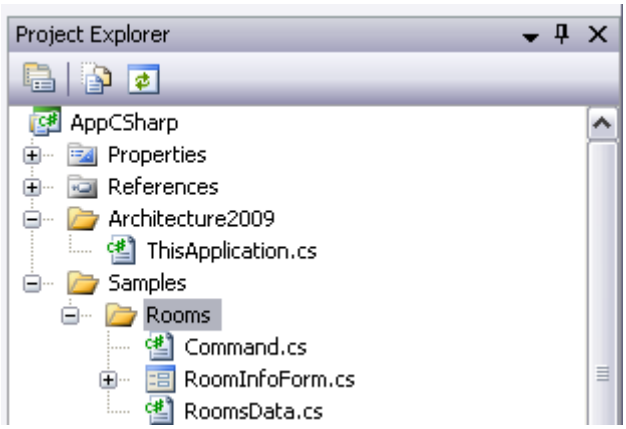
Add Existing Files to Macro Project in IDE

Return to the Revit VSTA IDE. In the Project Explorer (to continue this example) right-click on the folder you created for your macro, and select Add ➤ Existing Item from the menu. For example:



In the resulting IDE dialog, browse to the corresponding subfolder on the file system, under your Revit installation folder, select all the files that comprise the macro, and click Add.

In this example, the refreshed IDE Project Explorer for `AppCSharp` contains:



Create and Build Your Macro

Once the files have been added to your project, you can write a method that runs the macro. For example, in C#:

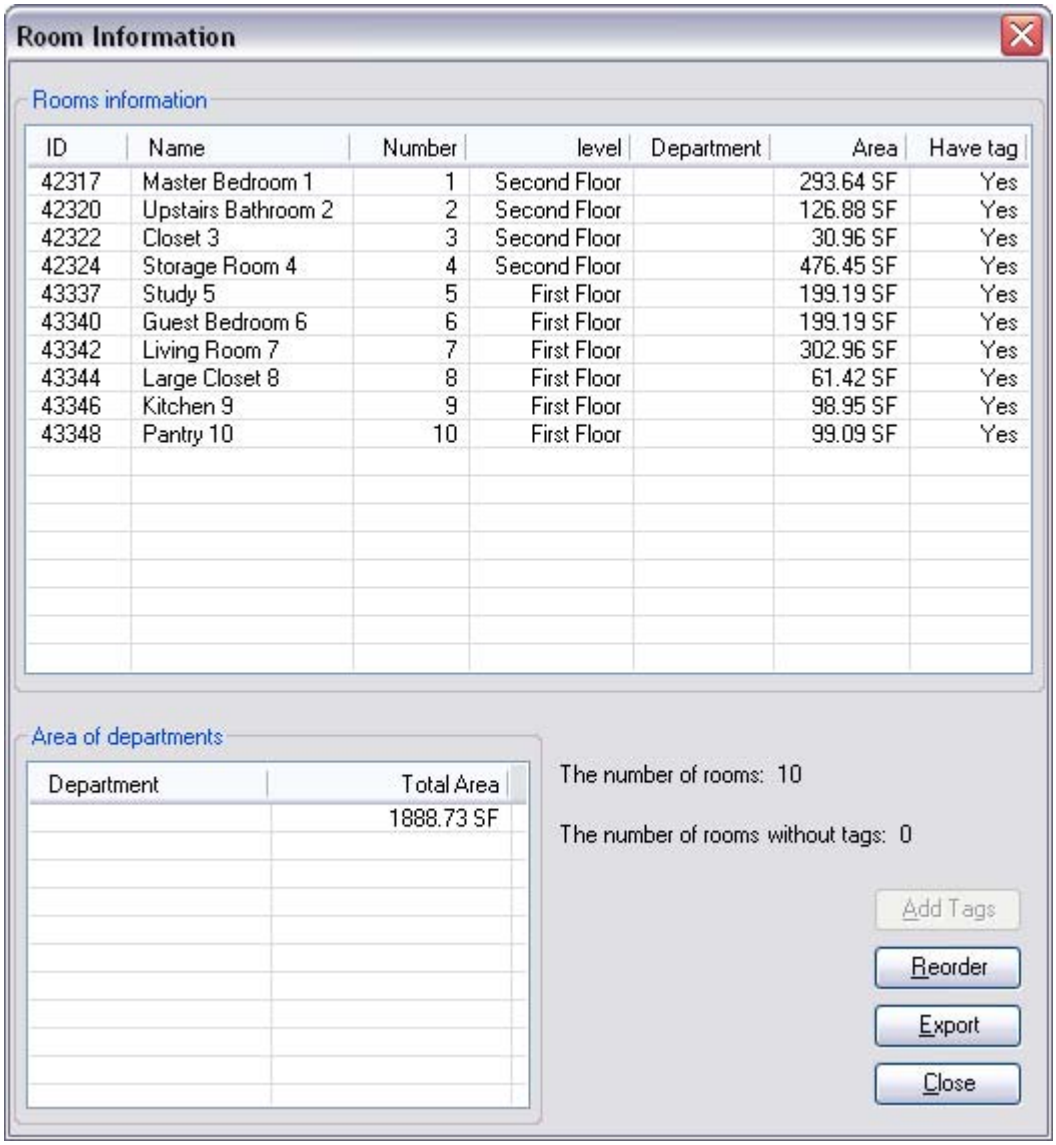
```
/// Sample Rooms test

public void RunSampleRooms()

{

     SamplesRoom sample = new SamplesRoom(this);

     sample.Run();

}
```

Be sure to add a using directive for the macro's namespace. For example:

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

Save the project and then click Build from the IDE toolbar menu. In Revit, launch Macro Manager and select your macro from the categorized list. On the SDK, the Rooms sample may be coded as a Document-level macro. In which case, code changes would be needed to run it in the `AppCSharp` project shown in this section's screens.

When run from Macro Manager, the macro collects data from your model and presents summary information. Here is an example:



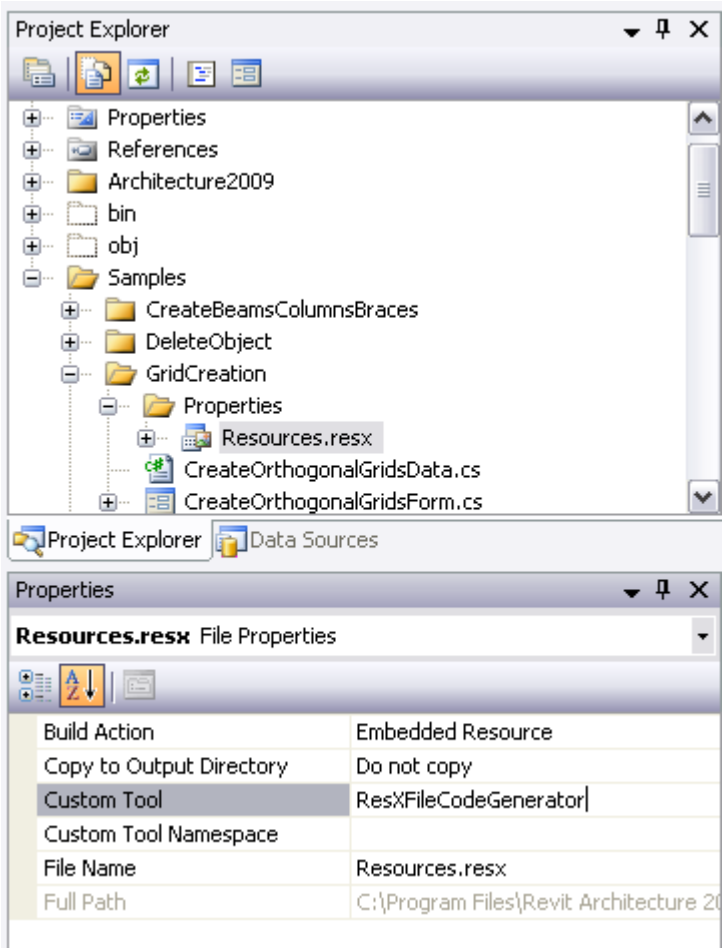
About resources.resx Properties

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. This was already done for you in the Document-level samples built into Revit_VSTA_Samples.rvt (from the SDK). However, for macros you develop yourself, you may need to define properties in the project's resources.resx file. This section shows an example.

In Project Explorer, navigate to the Properties folder for your macro. Example: 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:



Revit Architecture 2009 User's Guide > Creating Macros with Revit VSTA >

Revit API Differences

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

Feature or Capability	Standard Revit API	Revit Macro API
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`	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. 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.

Revit Architecture 2009 User's Guide > Creating Macros with Revit VSTA >

Migrating SDK Samples to Revit VSTA

The Revit SDK contains two samples folders:

\Revit <release> SDK\Samples\...

\Revit <release> SDK\VSTA Samples\...

The programs in the SDK's `\Samples\` folder use the standard Revit API. We refer to these samples as the "SDK sample code," as distinct from the "Revit VSTA samples."

If you want to use the SDK sample code for macros, modifications are needed. Follow the steps in this section. For a programming language, we will show C# examples. However these instructions also apply to the VB.NET versions of the SDK samples.

Initial Steps

The initial steps to migrate standard API samples from the SDK into your Revit VSTA macro project are similar to the section **Integrating Macros into Revit VSTA**. Except instead of copying files from the SDK's *\VSTA Samples\<sample-name>\...* folders, you will copy files from the SDK's *\Samples\<sample-name>\...* folders.

To review, the steps are:

1. In the IDE, add required references
2. In the IDE, create folders for the SDK samples you want to migrate
3. In Windows Explorer, copy the SDK standard API samples' files to the corresponding file system folders
4. In the IDE, add existing files to the macro project

Update the SDK Samples' Code

In the Revit VSTA IDE, the `IExternalCommand` interface is not available or used. In the SDK standard API sample code, you must update the class that inherits from this interface:

- Remove the method parameters and the return of the `Execute` method
- Update other code related to `ExternalCommandData`

Code Example Before Edits

The following code snippet is from a program that uses the standard Revit API:

```
/// the operation.</returns>
public IExternalCommand.Result Execute(Autodesk.Revit.ExternalCommandData commandData,
                                       ref string message, ElementSet elements)
{
    try
    {
        // create a new instance of class data
        RoomsData data = new RoomsData(commandData.Application);

        // create a form to display the information of rooms
        using (roomsInformationForm infoForm = new roomsInformationForm(data))
        {
            infoForm.ShowDialog();
        }
        return IExternalCommand.Result.Succeeded;
    }
    catch (Exception ex)
    {
        // If there are something wrong, give error information and return failed
        message = ex.Message;
        return IExternalCommand.Result.Failed;
    }
}
```

Code Example After Edits

In the Revit VSTA IDE, we need to update the code as follows. This example shows an Application-level macro. The method `RunSampleRooms()` is the entry for this VSTA sample. Notice that we used the `this` pointer to replace `commandData.Application`.

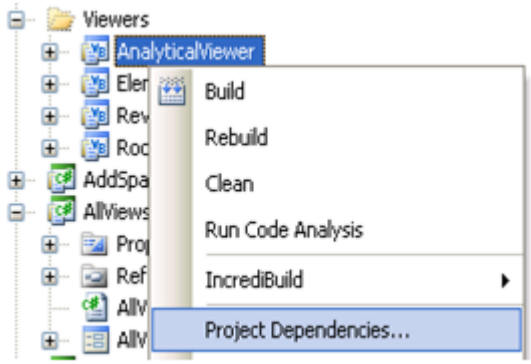
```
public void RunSampleRooms()
{
    try
    {
        // create a new instance of class data
        RoomsData data = new RoomsData(this);

        // create a form to display the information of rooms
        using (roomsInformationForm infoForm = new roomsInformationForm(data))
        {
            infoForm.ShowDialog();
        }
    }
    catch (Exception ex)
    {
        MessageBox.Show("Failed to run sample: " + ex.ToString());
    }
}
```

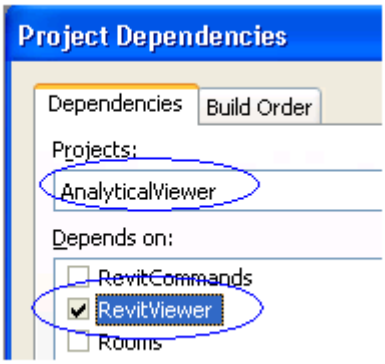
Additional Migration Notes for SDK Standard API Samples

Here are some additional notes:

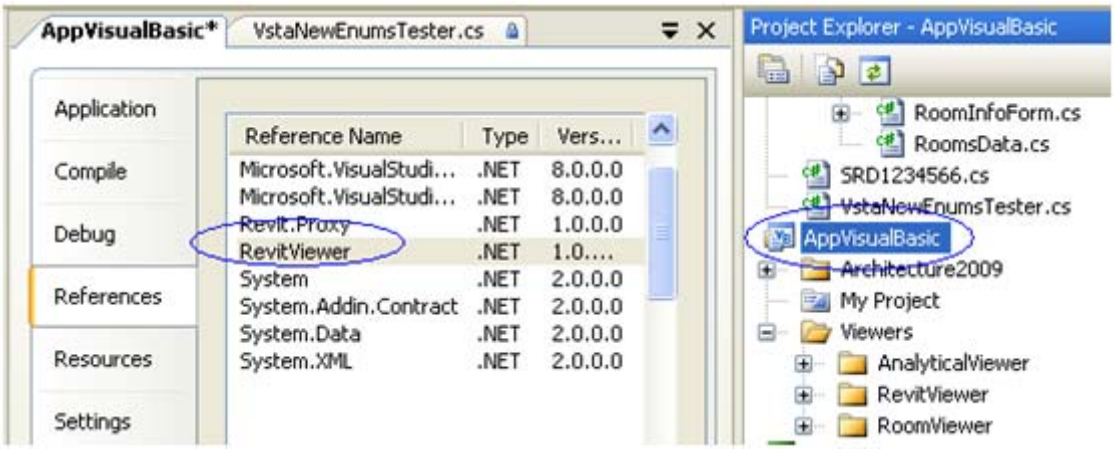
- By default, the SDK sample namespace is: `Revit.SDK.Samples.<SampleName>.CS`. As you edit the sample code that came from the SDK standard API samples, be sure to change the namespace for Revit VSTA. For example, in ThisApplication.cs:
  `namespace AppCSharp.csproj`
- The default project references in Revit VSTA projects only consist of basic references, such as `Revit.Proxy`. Remember to add other required references (as shown earlier in this topic). For example, you have to add `System.Windows.Forms` and `System.Drawing` references when running samples that present a user interface.

- Some samples have resources that must be defined in the resources.resx file. GridCreation is one such sample. If you migrate the SDK standard API sample for GridCreation to Revit VSTA, be sure to set the `resx` property.
- Project build dependence is not supported within the Revit VSTA IDE. If you want to use other DLLs, you must compile that dependent sample project outside of Revit VSTA IDE. For example, in the SDK standard API samples, there are several samples related to Viewers, such as AnalyticalViewer and RoomViewer, which depend on references to RevitViewer. In the SDK sample's solution, you can set the project dependences as shown here:





However, we cannot set the project dependency due to a limitation in Revit VSTA. We cannot migrate the RevitViewer sample to VSTA. Therefore, you need to compile RevitViewer as an independent SDK sample, and then add its DLL as a reference in the Revit VSTA project:



- When you migrate SDK standard API samples to Revit VSTA, do not copy in any Solution files (*.sln*) or existing project files (*.csproj or *.vbproj).
- Toolbar-related samples are unavailable in Revit VSTA.

Revit Architecture 2009 User's Guide > Creating Macros with Revit VSTA >

Revit Macros FAQ

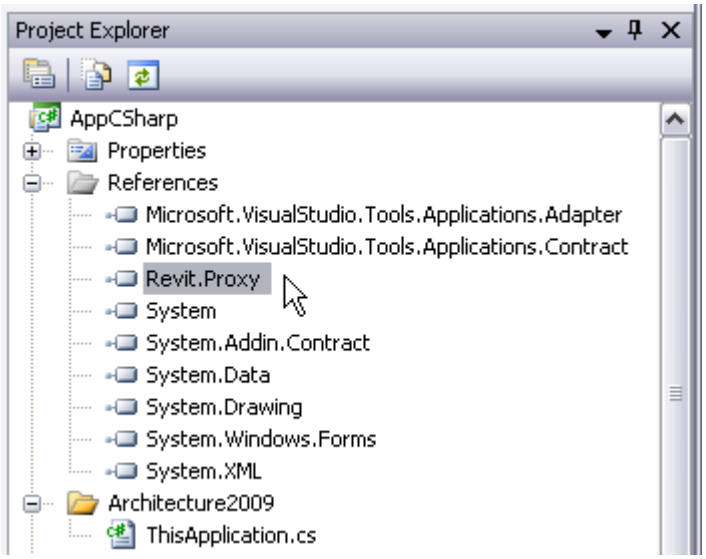
This section answers frequently asked questions about Revit macros.

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:



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 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 is created. Note: One of the VSTA samples provided on the Revit SDK may show an example of a Document-level event handler's startup and shutdown. This FAQ shows Application-level event handler examples.

VB.NET example, Application-level:

```
Private Sub ThisApplication_Startup(ByVal sender As Object, ByVal e As System.Even

     AddHandler Me.OnDocumentNewed, AddressOf Me.ThisApplication_OnDocumentNewed

End Sub
```

```
Private Sub ThisApplication_Shutdown(ByVal sender As Object, ByVal e As System.Eve
     RemoveHandler Me.OnDocumentNewed, AddressOf Me.ThisApplication_OnDocumentNewe
End Sub
Private Sub ThisApplication_OnDocumentNewed(ByVal document As Autodesk.Revit.Docum
     System.Windows.Forms.MessageBox.Show("VB.NET Application event OnDocumentNewe
End Sub
```

C# Example, Application-level:

```
private void ThisApplication_Startup(object sender, EventArgs e)
{
     this.OnDocumentNewed += new Autodesk.Revit.Events.DocumentNewedEventHandler(T
}
private void ThisApplication_Shutdown(object sender, EventArgs e)
{
     this.OnDocumentNewed -= new Autodesk.Revit.Events.DocumentNewedEventHandler(T
}
void ThisApplication_OnDocumentNewed(Document document)
{
     System.Windows.Forms.MessageBox.Show("C# Application event OnDocumentNewed");
}
```

Revit Architecture 2009 User's Guide > Creating Macros with Revit VSTA >

Related Information about Revit Macros

To learn more, please refer to the following resources:

- The Revit*API.chm Help file 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. 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.”
- The Autodesk Developer Network (ADN), **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.