Table of Contents

[Overview 2](#_Toc284490852)

[What Can I Do With the Autodesk Revit API? 2](#_Toc284490853)

[What You Will Need to Get Started 2](#_Toc284490854)

[Understanding Autodesk Revit 2](#_Toc284490855)

[Installation of the Autodesk Revit API 3](#_Toc284490856)

[Development Requirements 3](#_Toc284490857)

[Using the Autodesk Revit API 3](#_Toc284490858)

[Autodesk Revit SDK and Online Help 3](#_Toc284490859)

[Documentation Conventions 4](#_Toc284490860)

[What can you do with the Autodesk Revit API? 4](#_Toc284490861)

[Deployment Options 4](#_Toc284490862)

[1. External commands 4](#_Toc284490863)

[2. External applications 4](#_Toc284490864)

[3. REX addins 5](#_Toc284490865)

[Registration of add-ins 5](#_Toc284490867)

[External Commands 7](#_Toc284490868)

[The IExternalCommand Interface 7](#_Toc284490869)

[Parameters 7](#_Toc284490870)

[Return Value 8](#_Toc284490871)

[Remarks 8](#_Toc284490872)

[Using an Autodesk Revit API External Command 8](#_Toc284490873)

[External Command Object Lifetime 8](#_Toc284490874)

[External Applications 9](#_Toc284490875)

[The IExternalApplication Interface 9](#_Toc284490876)

[Parameters 9](#_Toc284490877)

[Return Value 9](#_Toc284490878)

[External Application Object Lifetime 9](#_Toc284490879)

[Compatibility of API applications with 64-bit Revit 9](#_Toc284490880)

[Debugging Your Application in MS Visual Studio 2019 10](#_Toc284490881)

[The Revit Unit System 11](#_Toc284490882)

[Storing and accessing Custom Data for Applications 11](#_Toc284490883)

[Appendix 1 – Glossary of Autodesk Revit terms 12](#_Toc284490884)

[Appendix 2 - FAQ 12](#_Toc284490885)

[General: 12](#_Toc284490886)

[Autodesk Revit Structure: 14](#_Toc284490887)

# Overview

Autodesk Revit offers an API designed to allow power users and external application developers to integrate their applications with Autodesk Revit, Revit Structure, Revit Architecture, and Revit MEP. The APIs for each of these products are very similar and will be jointly referred to as the Autodesk Revit 2021.1 API or Autodesk Revit API. It is strongly recommended that you become familiar with Autodesk Revit Architecture, Structure or MEP and its features before attempting to use the Autodesk Revit API. Training can be found through the Autodesk Developer Network (ADN).

# What Can I Do With the Autodesk Revit API?

The following are general areas where the API is suitable:

* Creating add-ins to automate repetitive tasks in the Autodesk Revit user interface
* Enforcing project design standards by checking for errors automatically
* Extracting project data for analysis and to generate reports
* Importing external data to create new elements or parameter values
* Integrating other applications, including analysis applications, into Autodesk Revit products
* Creating Autodesk Revit project documentation automatically

# What You Will Need to Get Started

* A working understanding of Autodesk Revit
* An installation of an Autodesk Revit-based product, including the Software Development Kit
* MS Visual Studio 2019 or MS Visual Studio 2019 Express Edition, or a full installation of Microsoft Visual Studio.
* Some experience in a .NET based development language (Autodesk Revit API examples are provided in C# and Visual Basic.NET.)
* 2 days of free time

# Understanding Autodesk Revit

All Autodesk Revit-basedproducts are Parametric Building Information Modeling (BIM) tools. Such a tool can be looked at as a CAD program that is used to build a 3D model rather than a set of individual drawing files. Autodesk Revit modeling is accomplished with real-world elements like columns, walls, doors and windows. The user can create views of the model, including plans, sections and callouts. All these views are directly generated from the 3D physical model so changes made in one view will automatically propagate through all other views. This process virtually eliminates the need to update multiple drawings and details when a change is made in the model.

The Autodesk Revit API is designed to reflect the same user interaction paradigms as the program’s Graphical User Interface. Therefore, the first step to understanding the API is to learn how to use the program. If you are an Autodesk Revit novice, we suggest you first start by going through the Tutorials which you can access through the program’s Help menu. You may also find it helpful to take a Training class from your local Autodesk reseller. This will help you quickly get up to speed with the program

Autodesk Resources:

* http://www.autodesk.com/products/revit-family/overview
* http://www.autodesk.com/solutions/building-information-modeling/overview <http://forums.autodesk.com/>

Select = Autodesk Revit

Then Select = Autodesk Revit API

External Resources:

* <http://www.augi.com/community/autodesk-revit>
* <http://www.revitcity.com>
* <http://revitinside.com>

# Installation of the Autodesk Revit API

The Autodesk Revit API is automatically installed with the default installation of the Autodesk Revit based product.

The Autodesk Revit API Software Development Kit is installed from the Tools and Utilities section of the Autodesk Revit installation DVD.

# Development Requirements

The Autodesk Revit API requires the Microsoft .NET Framework v4.8. To edit and debug your API applications, you need an interactive development environment such as Microsoft Visual Studio 2019 or one of the MS Visual Studio Express Editions for C# or Visual Basic.NET. When developing with the Autodesk Revit API, ensure that your project references two DLLs: **RevitAPI.dll** and **RevitAPIUI.dll** contained in the Autodesk Revit Program directory.

Some programming skills are required to effectively use the API. If you are a beginner in programming, we strongly advise you to learn Microsoft Visual Studio 2019 and one of the compatible languages like C# or Visual Basic.NET. There are many good books and classes to get you started.

**Resources:**

Online resources

* Free Visual Studio – <http://msdn.microsoft.com/VStudio/Express/>
* <http://www.codeguru.com/>
* <http://devx.com/>
* <http://www.msdn.microsoft.com/>

Books:

* Code Complete, Second Edition, by Steve McConnell
* Software Project Survival Guide, by Steve McConnell

# Using the Autodesk Revit API

## Autodesk Revit SDK and Online Help

Get the Autodesk Revit SDK from ADN or the installation DVD. The Autodesk Revit 2021.1 SDK can be unpacked and installed from the Tools and Utilities option in the installation DVD. In the SDK, there are example files that will help you get a better understanding of the API and its use. Each example file has a sample revit.ini file with the information that you will need to edit and place into the revit.ini file located in your Autodesk Revit Program folder, which Autodesk Revit will access on launch.

* Unzip the SDK onto a local drive
* Read the Read Me First.doc in the SDK

RevitAPI.chm is the Autodesk Revit API reference documentation help file, included with the SDK package in the \Revit 2021.1 SDK\ folder.

**Resources:**

[www.autodesk.com/adn](http://www.autodesk.com/adn) - Autodesk Developer Network home (ADN)

[www.autodesk.com/developrevit](http://www.autodesk.com/developrevit) - Autodesk Revit development resources

<http://discussion.autodesk.com> – Autodesk product discussion groups

Select = Autodesk Revit Architecture

Then Select = Autodesk Revit API

<http://thebuildingcoder.typepad.com/blog/> - The Building Coder, an AND blog dedicated to Revit coding

<http://bimapps.typepad.com/> - Bim Apps, a blog dedicated to BIM applications

### Documentation Conventions

This document will contain names in namespace format, such as **Autodesk.Revit.DB.Element**

## What can you do with the Autodesk Revit API?

### Deployment Options

The Autodesk Revit API supports in-process DLLs only. This means that your API application will be compiled as a DLL loaded into the Autodesk Revit process.

The Autodesk Revit API supports single threaded access only. This means that your API application must perform all Autodesk Revit API calls in the main thread (which is called by the Autodesk Revit process at various API entry points), and your API application cannot maintain operations in other threads and expect them to be able to make calls to Autodesk Revit at any time.

There are two types of DLLs that you can create with the Autodesk Revit API:

### External commands

The Autodesk Revit API enables you to add new commands to the user interface of Autodesk Revit. These commands will appear in the Add-ins tab under the ‘External Tools’ pulldown, as seen in Figure 1. Through the API, external tool commands have access to the Autodesk Revit database, as well as the currently selected elements.

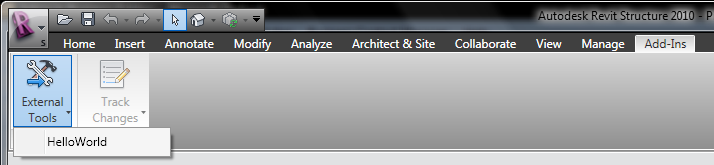


Figure 1 - External Tool added to Revit

### External applications

The Autodesk Revit API enables you to also add external applications. These applications are invoked during Autodesk Revit startup and shutdown. They can create new panels in the Add-ins tab, as seen in Figure 2. They can also register handlers that can react to events occurring in the Autodesk Revit user interface.

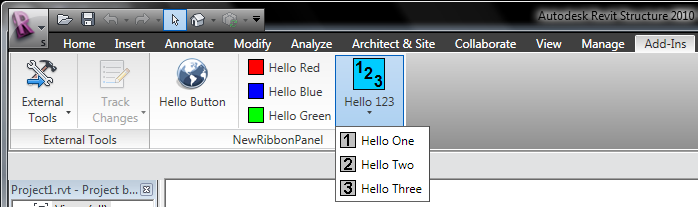


Figure 2 - New panels and controls added to Revit

### REX addins

REX (Revit Extensions) is an API framework that lets you build applications for Revit in .NET similar to classes that implement IExternalCommand. REX is meant to give you a more high-level development environment through built-in resources such as:

* Automatic dialog box creation and display
* Libraries to work with units and geometry
* Built-in command-based architecture to make menu and toolbar development easier.
* A standard mechanism for accessing a reference to the Revit application object.
* Automatic deployment and installation of addins for easy debugging.

Please see the “\Revit 2021.1 SDK\REX SDK\” folder for more details.

## Registration of add-ins

The Revit API offers the ability to register API applications via a .addin manifest file.

Manifest files will be read automatically by Revit when they are places in one of two locations on a user's system:

* **In a non-user specific location in "application data"**
  + C:\ProgramData\Autodesk\Revit\Addins\2021\
* **In a user specific location in "application data"**
  + C:\Users\<user>\AppData\Roaming\Autodesk\Revit\Addins\2021\

All files named .addin in these locations will be read and processed by Revit during startup.

A basic file adding one ExternalCommand looks like this:

<?xml version="1.0" encoding="utf-16" standalone="no"?>

<RevitAddIns>

<AddIn Type="Command">

<Assembly>c:\MyProgram\MyProgram.dll</Assembly>

<AddInId>76eb700a-2c85-4888-a78d-31429ecae9ed</AddInId>

<FullClassName>Revit.Samples.SampleCommand</FullClassName>

<Text>Sample command</Text>

<VisibilityMode>NotVisibleInFamily</VisibilityMode>

<VisibilityMode>NotVisibleInMEP</VisibilityMode>

<AvailabilityClassName>Revit.Samples.SampleAccessibilityCheck </AvailabilityClassName>

</AddIn>

</RevitAddIns>

A basic file adding one ExternalApplication looks like this:

<?xml version="1.0" encoding="utf-16" standalone="no"?>

<RevitAddIns>

<AddIn Type="Application">

<Name>My sample application</Name>

<Assembly>c:\MyProgram\MyProgram.dll</Assembly>

<AddInId>604B1052-F742-4951-8576-C261D1993107</AddInId>

<FullClassName>Revit.Samples.SampleApplication</FullClassName>

</AddIn>

</RevitAddIns>

Multiple AddIn elements may be provided in a single manifest file.

The new mechanism currently offers the following XML tags:

|  |  |
| --- | --- |
| **Tag** | **Description** |
| Assembly | The full path to the add-in assembly file. Required for all ExternalCommands and ExternalApplications. |
| FullClassName | The full name of the class in the assembly file which implements IExternalCommand or IExternalApplication. Required for all ExternalCommands and ExternalApplications. |
| AddInId | A GUID which represents the id of this particular application. AddInIds must be unique for a given session of Revit. Autodesk recommends you generate a unique GUID for each registered application or command. Required for all ExternalCommands and ExternalApplications. The property UIApplication.ActiveAddInId provides programmatic access to this value, if required. |
| Name | The name of application. Required; for ExternalApplications only. |
| Text | The name of the button. Optional; use this tag for ExternalCommands only. The default is "External Tool". |
| Description | Short description of the command, will be used as the button tooltip. Optional; use this tag for ExternalCommands only. The default is a tooltip with just the command text. |
| VisibilityMode | Provides the ability to specify if the command is visible in project documents, family documents, or no document at all. Also provides the ability to specify the discipline(s) where the command should be visible. Multiple values may be set for this option. Optional; use this tag for ExternalCommands only. The default is to display the command in all modes and disciplines, **including when there is no active document.** Previously written external commands which need to run against the active document should either be modified to ensure that the code deals with invocation of the command when there is no active document, or apply the NotVisibleWhenNoActiveDocument mode. |
| AvailabilityClassName | The full name of the class in the assembly file which implemented IExternalCommandAvailability. This class allows the command button to be selectively grayed out depending on context. Optional; use this tag for ExternalCommands only. The default is a command that is available whenever it is visible. |
| LargeImage | The path to the icon to use for the button in the External Tools pulldown menu. The icon should be 32 x 32 pixels for best results. Optional; use this tag for ExternalCommands only. The default is to show a button without an icon. |
| LongDescription | Long description of the command, will be used as part of the button's extended tooltip. This tooltip is shown when the mouse hovers over the command for a long amount of time. You can split the text of this option into multiple paragraphs by placing <p> tags around each paragraph. Optional; use this tag for ExternalCommands only. If neither of this property and TooltipImage are supplied, the button will not have an extended tooltip. |
| TooltipImage | The path to an image file to show as a part of the button extended tooltip, shown when the mouse hovers over the command for a longer amount of time. Optional; use this tag for ExternalCommands only. If neither of this property and TooltipImage are supplied, the button will not have an extended tooltip. |
| LanguageType | Localization setting for Text, Description, LargeImage, LongDescription, and TooltipImage of external tools buttons. Revit will load the resource values from the specified language resource dll. The value can be one of the eleven languages supported by Revit. If no LanguageType is specified, the language resource which the current session of Revit is using will be automatically loaded. |

# External Commands

* Implement support for a specific interface, in this case Autodesk.Revit.UI.IExternalCommand.
* Adds one or more entries to the Revit.ini file describing the external command.

Technically, an external command is an exposed .NET object that supports the Autodesk.Revit.UI.IExternalCommand interface. Furthermore, in the Revit.ini file there must be one entry for each such object in order for Revit to be able to “see” and to use the commands.

## The IExternalCommand Interface

The declaration (VB.NET) of the interface is as follows:

*Function Execute(ByVal commandData As Autodesk.Revit.UI.ExternalCommandData,*

*ByRef message As String,*

*ByVal elements As Autodesk.Revit.DB.ElementSet)*

*As Result*

### Parameters

* commandData : The object passed in this parameter contains information important to the command that is being executed. This data includes the Autodesk Revit Application object as well as the currently active view.
* message : The message string can be set to supply a specific message to the user when the command terminates. How this message is displayed is dependent upon the return value of the function. See the remarks section for more details.
* elements : Initially this is an empty set that can contain Autodesk Revit elements. When the command terminates, the elements within this set may be displayed, based on the return value. See the remarks section for more details.

### Return Value

result : The return value can be one of the following:

* Success : Is returned if the command succeeded as expected without any unhandled error conditions. The external command will appear as an undoable operation in the Autodesk Revit user interface.
* Cancelled : This value specifies that the user requested that the command be cancelled. Any changes that were made to Autodesk Revit objects during the external commands execution will be undone. A message may be posted, see the Remarks section.
* Failure : Failure signifies that the external command failed in some manner from which it cannot recover. Any changes made to Autodesk Revit objects during the execution of the external command will be undone. A message will be posted, see the Remarks section.

### Remarks

The message and elements parameters are used if the command was cancelled or failed.

* Cancelled : If the external command was cancelled and the message parameter was set by the external command then the message is displayed when execution is returned back to Autodesk Revit. If the message parameter was not set then no message is displayed and the command will exit silently.
* Failed : If the external command failed then the contents of the message parameter will be displayed. If the element set contains Autodesk Revit elements then these elements will be highlighted when the error message is displayed thus giving the developer the ability to show the user the problem elements.

## Using an Autodesk Revit API External Command

1. User opens/creates a project in Autodesk Revit
2. User selects the external command from the External Tools pulldown on the Add-ins tab.
3. The user had the option to select a number of Autodesk Revit elements before invoking the External Tools program. If they did, the program can decide to only perform its function on the selected members.
4. The API program takes focus from Autodesk Revit and performs the required task. Often a dialog box may be required to obtain user input before the application can complete its work.
5. Once the add-on tool has completed its function or has been dismissed by the user the program will update the Autodesk Revit model as required and return from the external command, giving focus back to Autodesk Revit.

|  |  |
| --- | --- |
|  |  |
|  |  |

## External Command Object Lifetime

When no other command or edit modes are active within Autodesk Revit, the registered external command will become enabled. When picked, the command object will be created and the Execute method called. Once this method returns back to Autodesk Revit the command object will be destroyed. Due to this destruction, data cannot persist within the object between command executions. If you wish the data to persist you may use an external file or database to do so. If you wish the data to persist within the Autodesk Revit project you may use the shared parameters mechanism to store this data.

# External Applications

* Implement support for a specific interface, in this case Autodesk.Revit.UI.IExternalApplication.
* Adds one or more entries to the Revit.ini file describing the external application.

Technically, an external application is an exposed .NET object that supports the Autodesk.Revit.IExternalApplication interface. Furthermore, in the Revit.ini file there must be one entry for each such object in order for Autodesk Revit to be able to load these applications when Autodesk Revit starts.

## The IExternalApplication Interface

The declaration (C#) of the interface is as follows:

* Autodesk.Revit.UI.IExternalApplication.Result OnStartup(Autodesk.Revit.ControlledApplication application)
* Autodesk.Revit.UI.IExternalApplication.Result OnShutdown(Autodesk.Revit.ControlledApplication application)

### Parameters

* application : The object passed in this parameter contains information important to the commands OnStartup and OnShutdown that are being called. This object provides limited access methods of Autodesk Revit Application, such as VersionName, VersionNumber; and delegates for some events, such as OnDocumentOpened, OnDocumentSaved.

### Return Value

result : The return value can be one of the following:

* Success : Is returned if the external application succeeded as expected without any unhandled error conditions.
* Failure : Failure signifies that the external application failed in some manner from which it cannot recover.
* Cancelled : This value specifies that the external application be cancelled.

## External Application Object Lifetime

When Autodesk Revit starts, the external application object will be created and the OnStartup method called. Once this method returns back successfully to Autodesk Revit the external application object will be held during the entire Autodesk Revit session. The OnShutdown method will be called when Autodesk Revit shuts down.

# Compatibility of API applications with 64-bit Revit

Most API applications can be built with settings that allow them to be used either with 32-bit Revit or 64-bit Revit. Use the following settings to build your project in this manner:

* For C# project, go to project property->Build tag, select platform target as “Any CPU”.
* For VB.Net project, go to project property->Compile tag, select platform as “Any CPU”.
* For C++/CLI project, open property page and go to Configuration properties->General->Common Language Runtime support, select /clr:safe.

Note that there are some Microsoft components (such as OleDB provider for Microsoft Jet) that are not supported for 64-bit. All SDK samples are set up to work with either 32-bit or 64-bit Revit, except those samples that have dependencies to unsupported Microsoft components.

**For more information see this Microsoft resource:**

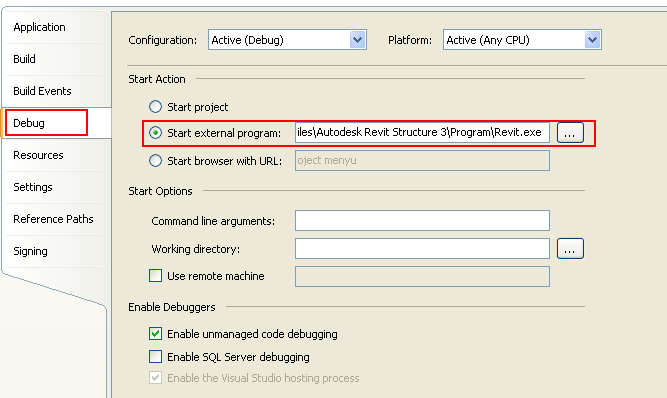
<http://msdn.microsoft.com/en-us/library/ms241064(VS.80).aspx>

# Debugging Your Application in MS Visual Studio 2019

There are a few differences between debugging a standalone application (EXE) and an external application (DLL) that needs another program to launch it. To debug an application that is using the Autodesk Revit API it needs to be activated by Autodesk Revit. To do this in the developer environment for debugging you will need to:

|  |  |
| --- | --- |
| 1. Open up the Visual Studio project for the API application. (For example AnalyticalSupportData\_Info.csproj from the Samples folder) 2. From the Project menu select AnalyticalSupportData\_Info Properties |  |

1. Select the Debug tab on the left
2. Select the “Start external program:” radio button
3. Press the browse button and find the Revit.exe file and press Open



1. Set some break points in your source code.
2. From Visual Studio, select “Start Debugging”. Autodesk Revit will launch.
3. To hit a break point select the option for your program from the External Tools menu. Once the compiler reaches one of your break points it will stop to let you debug your program.

# The Revit Unit System

The Revit Unit System uses the following base units:

|  |  |  |
| --- | --- | --- |
| **Base Unit** | **Unit In Revit** | **Unit System** |
| Length | Feet (ft) | Imperial |
| Angle | Radian | Metric |
| Mass | Kilogram (kg) | Metric |
| Time | Seconds (s) | Metric |
| Electric Current | Ampere (A) | Metric |
| Temperature | Kelvin (K) | Metric |
| Luminous Intensity | Candela (cd) | Metric |

**Note**: Because Revit stores lengths in feet and other quantities in metric, a derived unit involving length uses a non-standard unit using the Imperial and the Metric systems. For example, force is measured in mass-length per time squared and is stored in kg-ft/s2.

# Storing and accessing Custom Data for Applications

Often programs linked to Autodesk Revit require information that is not available in the Autodesk Revit model database. There are a number of ways for the user to enter such additional information. How and where the information is entered depends on its use:

* When the information is of a general type and the user will want to see and edit it inside Autodesk Revit, then it should be stored as a visible Project or Shared Parameter.
* If the information needs to be kept with the Autodesk Revit model as it evolves but does not need to be visible then it can be stored in the Autodesk Revit model as a non-visible Project or Shared Parameter.
* If the information is specific to a single add-on program, and is too large to practically store within the Autodesk Revit model such as specifications for a multitude of building products subject to change, then the best solution may be to create a concurrent model database that stores the program specific information. In this case it may be useful to use the element UniqueId property for each element as a key for the database, because the element’s UniqueId is stable within a model.

## 

## Appendix 1 – Glossary of Autodesk Revit terms

###### Family

A Family is a collection of objects called types. A family groups elements with a common set of parameters, identical use, and similar graphical representation. Different types within a family may have different values of some or all parameters, but the set of parameters - their names and their meaning - are the same.

###### Parameter

There are a number of Autodesk Revit parameter types. **Shared Parameters** can be thought of as user defined variables. **System Parameters** are variables that are hard coded in Autodesk Revit. **Family parameters** are variables that are defined when a family is created or modified.

###### Type

A Type is a member of a Family. Each Type has specific parameters that are constant for all instance of the Type that exist in your model. These are called Type Properties. Types have other parameters called Instance parameters, which may vary over the various instances of this type in your model.

## Appendix 2 - FAQ

### General:

**Q:** How do I reference an element in Autodesk Revit?

**A:** Each element has Id and UniqueId properties that can be used to reference the element.

**Q:** Are element IDs universally unique and can they ever change?

**A:** At any given time, element IDs in a Revit model are unique. However, multi-user interactions in Revit can result in an element’s ID changing. Also, another element in a different model may have an element with the same ID. If you need to store an ID (especially an ID of an element that was newly created) in order to find the element later, store UniqueId.

**Q:** Can a model only use one shared parameter file?

**A:** Shared parameter files are used to store a parameter specification. The most important portion of these parameter descriptions is the GUID that is used to insure the uniqueness of a parameter both in a single model file and across multiple models.

Autodesk Revit can work with multiple shared parameter files but can only read parameters from one file at a time. It is then up to you to choose the same shared parameter file for all models or a different one for each.

An API application should avoid interfering with the user’s own parameter file. Your application should ship with a parameter file containing the parameters it requires. To load the parameter(s) into an Autodesk Revit file, an application should:

* Store the name of the user’s parameter file,
* Make the applications parameter file current,
* Load the required parameter(s) and,
* Make the user’s parameter file current.

**Q:** Do I need to distribute the shared parameters file with the model so other programs can use the shared parameters?

**A:** No, you do not need to distribute shared parameter files. The shared parameters file is only used to load shared parameters. After they are loaded the file is no longer needed for that model.

**Q:** Do shared parameter values get copied when the corresponding element is copied?

**A:** Yes they do. Therefore if you have a shared parameter that holds a unique ID for an item in your database it is recommended that you make this ID include the Autodesk Revit element’s UniqueId property. This will allow you to identify elements that were copied from existing elements in your application.

**Q:** Autodesk Revit takes a long time to update when my application sends data back to the model. What do I need to do to speed it up?

**A:** You should use the Manual regeneration mode to prevent the API framework from regenerating after every model level change. Then, you will use the regeneration APIs to force update of the document after a group of changes. Performance of multiple modifications of the Revit document should be faster than RegenerationOption.Automatic. Because this mode suspends all updates to the document, your application should not read data from the document after it has been modified until the document has been regenerated, or it runs the risk of accessing stale data.

**Q:** What do I do if I want to add shared parameters to elements that do not have the ability to have shared parameters bound to them? For example: Grids or Materials.

**A:** If an element type does not have the ability to contain shared parameters, you will need to add a project parameter. This does make it a bit more complicated as you will need different code to access the value of the parameter on these elements.

**Q:** When exporting a model containing groups to an external program, the user receives an error at the end of the export:

"Changes to group "Group 1" are allowed only in group edit mode. Use the Edit Group command to make the change to all instances of the group. You may use the "Ungroup" option to proceed with this change by ungrouping the changed group instances."

**A:** Currently the API does not permit changes to members of groups. You can programmatically ungroup, make the change, regroup and then swap the other instances of the old group to the new group to get the same effect.

**Q:** Why do two different FamilySymbols have the same name in my Autodesk Revit document?

**A:** In the Autodesk Revit API, Symbols represent geometry.

Consider that two window instances of the same type are inserted into walls of different types. Now, these two window instances refer to the same FamilySymbol and count as being of the same type, but it is clear that they cannot possibly use the same symbol’s geometry, since inserting them into walls of different thickness changes each window’s geometry to match. To track individual instances of FamilySymbols, Autodesk Revit employs a “primary” and “subordinate” functionality with “subordinate” FamilySymbols localizing their geometry to surrounding conditions according to the “primary” FamilySymbol template.

If there is no door instance in a model, there is only one “primary” FamilySymbol for a 36”x84” door, but if a door is inserted, Autodesk Revit will create a “subordinate” FamilySymbol for the new 36”x84” Door.

Subsequent doors placed in the model will cause Autodesk Revit to generate additional “subordinate” FamilySymbols.

**Q:** Is the order of vertices or the triangles in an Autodesk Revit geometry mesh object reliably sorted in counter-clockwise direction?

**A:** Yes.

**Q:** Why after flipping a wall or door (both external and internal) do its properties FromRoom and ToRoom do not change?

**A:** External Doors & External Windows have only either a FromRoom or a ToRoom. Users can decide which property exists when locating doors or windows. If external doors or windows are flipped their FromRoom or ToRoom properties do not change automatically, but can be manually changed.

### Autodesk Revit Structure:

**Q:** Sometimes the default end releases of structural elements render the model unstable.

**A:** In some cases the user may not have paid attention to the element end release when modeling the elements. The Analytical Model Check feature will find some of these issues. When exporting the analytical model, ask the user whether they want to retain the release conditions from RST or they want to set all beams and columns to be fixed. When re-importing the model to RST, always update the end releases and do not overwrite the end releases on subsequent exports to analysis programs.

**Q:** I'm rotating the orientation of the beam so it is rotated in the weak direction, i.e. the I of a W14X30 is rotated to look like an H by a 90 degree rotation. How is that rotation angle accessed in the API?   
Because the location is a LocationCurve not a LocationPoint I do not have access to the Rotation value so what is it I need to check? I've got a FamilyInstance element to check so what do I do with it?

**A:** Take a look at the RotateFramingObject example in the SDK. It has examples of how to get and change the rotation angle of beams, braces and columns.

**Q:** How do I add new concrete beam and column sizes to a model?

**A:** Take a look at the FrameBuilder sample code in the SDK.

|  |  |
| --- | --- |
| **Q:** How to view true deck layer?  **A:** There is an example called DeckProperties that provides the information on how to get the layer information for the deck. The deck information is reported in exactly the same way as the UI. The deck dimension parameters are shown in the following figure. |  |

**Q:** How do I tell when I have a beam with a cantilever?

**A:** There is no direct way in the Autodesk Revit database to tell if a beam has a cantilever. However one or more of the options below can give you a good guess at whether a beam is a cantilever:

Options

1. There are two parameters called Moment Connection Start and Moment Connection End. If the value that is set for these two is not None then you should look and see if there is a beam that is co-linear and also has the value set to something other than None. You could also ask the user to make sure to select the Cantilever Moment option rather than Moment Frame option.
2. Trace the connectivity back beyond the element say one or two elements.
3. Look at element release conditions.

**Q:** How do I model a foundation with end overhangs under a wall?

**A:** The wall foundation capability in Autodesk Revit Structure does not support an overhang at its ends. The best solution is to use the foundation slab feature in the GUI or API. You can create them using the Autodesk.Revit.Creation.Document.NewFloor() methods. For the STRUCTURAL\_FLOOR\_ANALYZES\_AS. You will want to use AA\_Mat or AA\_SlabOnGrade for foundation or AA\_Slab if it is a floor.