**The Revit® SDK Sample Smörgåsbord**

Jeremy Tammik – Autodesk Inc.

**DE101-3** Don't miss this sumptuous collection of Revit SDK samples! The collection is a complete knowledge base of working solutions for every conceivable Revit programming task. With over 100 different samples to keep track of and more than a dozen added in 2009 alone, selecting the appropriate one for any given issue can be a daunting task. This session provides a complete overview of all the SDK samples and their functionality, including detailed information on the new samples and classification of all samples by programming topic; level of difficulty; Revit flavour (Architectural, Structural, or MEP); history; and programming language (C# or VB.NET). This session assumes basic knowledge of Revit programming, and is a must for every Revit programmer or anyone aspiring to become one.

**About the Speaker:**

Jeremy is a member of the AEC workgroup of the DevTech team, providing developer support, training, and conferences to the Autodesk Developer Network ADN. He originally joined Autodesk in 1988 as the technology evangelist responsible for European developer support. In this capacity, he wrote articles, consulted, lectured on AutoCAD application programming techniques, and supported AutoCAD application developers in Europe, the U.S., Australia, and Africa. He was a cofounder of ADGE, the AutoCAD Developer Group Europe, and a prolific author on AutoCAD application development. He left Autodesk in 1994 to work as an HVAC application developer, and then rejoined the company in 2005.

Jeremy graduated with a MA in mathematics and physics in Marburg, Germany, in 1984, and worked first as a teacher and translator of both computer and human languages, then as a C++ programmer on early GUI and multitasking projects. He is fluent in five European languages, vegetarian, has four kids, plays the flute, likes reading, travelling, theatre improvisation and carpentry, loves mountains, oceans, sports and especially climbing.

Jeremy writes a blog on the Revit API at <http://thebuildingcoder.typepad.com> and can be contacted at [jeremy.tammik@eur.autodesk.com](mailto:jeremy.tammik@eur.autodesk.com).

The Revit SDK Sample Smörgåsbord

Jeremy Tammik, Consulting Analyst

Developer Technical Services, EMEA

Autodesk SARL

[Introduction 3](#_Toc215928197)

[Revit API History 4](#_Toc215928198)

[The Revit SDK Contents 4](#_Toc215928199)

[Non-SDK Samples 6](#_Toc215928200)

[RvtMgdDbg 6](#_Toc215928201)

[Revit MEP Samples 6](#_Toc215928202)

[Revit Structure Labs, Link, Rebar, and Detailing Samples 6](#_Toc215928203)

[Managing Samples 7](#_Toc215928204)

[Complete overview and readme 7](#_Toc215928205)

[The Main Samples Solution 7](#_Toc215928206)

[Globally Searching all Samples 7](#_Toc215928207)

[Compiling all Samples 8](#_Toc215928208)

[Revit Samples Spreadsheet 8](#_Toc215928209)

[Loading all Samples using RvtSamples 9](#_Toc215928210)

[Enhanced version 10](#_Toc215928211)

[Samples Overview 10](#_Toc215928212)

[Basic Samples 11](#_Toc215928213)

[Revit 9.0 Samples 14](#_Toc215928214)

[Revit 9.1 Samples 14](#_Toc215928215)

[Revit 2008 Samples 15](#_Toc215928216)

[Revit 2009 Samples 15](#_Toc215928217)

[Appendix 24](#_Toc215928218)

[Acronyms 24](#_Toc215928219)

[Learning More 24](#_Toc215928220)

[JHint, the Visual Studio Project HintPath Batch Processor 25](#_Toc215928221)

## Introduction

This paper provides a complete overview of the Revit SDK samples. We start by mentioning some introductory facts on the Revit API history and the rest of the SDK contents, since this is useful for understanding the samples and how to structure their management. For completeness’ sake, in addition to the official SDK samples, we also mention a few important non-SDK samples and RvtMgdDbg, a tool that you really should be aware of when developing your own Revit add-on.

Next, we will discuss a few powerful utilities and information sources for managing the large number of SDK samples provided. Finally, we will dive into the samples themselves and present a short description of some of them, what they do, and how:

* Introduction: SDK history and contents
* Important non-SDK samples: RvtMgdDbg, RME and RST, MidasLink
* Managing samples: Readme, SDK solution, spreadsheet, menu generator
* SDK Samples: Overview and details

We will not provide a full overview of the API here. However, that is a prerequisite for being able to make proper use of this material. For such an overview, please refer to the Revit API Introduction, which is also available as a DevTV presentation from the Autodesk web site.

The techniques presented for managing the SDK samples can be partially applied to the non-SDK ones as well.

The two most important tools for managing the SDK samples are the main SDK Visual Studio solution SDKSamples2009.sln for accessing the source code, and the menu generator RvtSamples to execute them inside the Revit product. This application was previously not included in the SDK, but it was added in the 2009 version. We do provide one additional resource in this session’s materials, the spreadsheet Revit\_SDK\_Samples.xlsx categorising all Revit SDK samples.

### Revit API History

First, a little bit on the history of the Revit API. The API has been and still is evolving strongly since Revit 8.0, and has reached a certain maturity now. We are in the long-term process of restructuring Revit to make it more API driven, i.e. create a kernel providing the API on top of which the various Revit flavours can be implemented, instead of implementing the API as the outermost layer on top of the completed product.

* 5, 6, 7 had no API and no verticals
* 8.0 first public API for Building and Structure
* 9.0 many new objects and creation methods
* 9.1 journal, units, new creation methods
* 2008 major new features
* 2008.2 includes an SDK update and new samples
* 2009 filtering facilities, data access, samples, VSTA
* API size doubled in every release
* Long-term target is an API and kernel-based application
* Still evolving ...

### The Revit SDK Contents

The Revit SDK or Software Developer Kit is included in every Revit product installation CD. You can search the installation for a file named \*sdk\*.zip. In Revit 2009, it is named 'Revit 2009 SDK.zip' and located in a subdirectory 'Utilities\Common\Software Development Kit'. It can also be downloaded from the public Revit developer centre which can be reached via www.autodesk.com > Communities > Developers > Products & Technologies > Revit Architecture and Revit Structure, or from the members-only ADN web site.

The contents of the Revit SDK are basically for documentation purposes only. All that is really needed to allow a Revit plug-in to run is the Revit API .NET assembly RevitAPI.dll, which is always present in every Revit installation and located in the Program subfolder, the same place as Revit.exe itself. Additionally, you need a development environment to compile your plug-in. The standard development environment we normally use is Microsoft Visual Studio. You can also use the free Visual Studio express version with reduced functionality, or any other .NET development environment. Alternatively, in Revit 2009, you can use the Visual Studio Tools for Application or VSTA environment which is included with the Revit product instead of compiling an external .NET plug-in assembly.

Let us look at the contents of the Revit SDK in more detail. Here are the documents present in the top level directory:

* Read Me First.doc
* Getting Started Revit API 2009.doc
* Revit\_2008\_API\_User\_Manual.pdf (on ADN web site)
* Add-On Applications and Non-Power Users.doc
* Guide to placing Family Instances with the API.doc
* Revit API Diagram.dwf
* Revit SDK License.rtf
* Revit VSTA User Manual.pdf
* RevitAPI.chm

'Read Me First.doc' provides a brief overview of the SDK contents and is the first place to start exploring. The next step is 'Getting Started Revit API 2009.doc', which explains the basic paradigms and architecture, how to create a Revit plug-in, and many further details. The 'Guide to placing Family Instances with the API.doc' is a recent addition in the 2009.1 Revit SDK update. A graphical overview of the API class hierarchy is provided by 'Revit API Diagram.dwf', showing the relationships between over 400 classes. The license is provided in a separate file, and a detailed user manual for VSTA, the Visual Studio Tools for Applications, which can be used to create macros inside Revit instead of loading them as external plug-ins separately compiled in an external development environment.

The main help file is RevitAPI.chm, which lists all the classes provided in the API and their properties and methods. It does not do much to explain how these classes work together to solve specific programming tasks. For that, the best source of information is the collection of samples, which we will discuss in detail later.

In addition to the documents listed above, the SDK root folder also includes the following subdirectories:

* API Changes
* Add-in Manager
* Revit Structure
* VSTA Samples
* Samples

API Changes lists the changes between the current and the previous version of the API and is useful for migrating existing applications. The changes are documented in an xml file Added.xml and displayed through an XSLT generated view. XSLT stands for XSL Transformation or eXtensible Stylesheet Language Transformation. An XSL style sheet transforms the XML data into a specific view in the browser.

The Add-in Manager is a utility application for loading and managing plug-ins and is accompanied by its own documentation.

A set of additional documentation is provided specifically for Revit Structure in an own folder. Some VSTA samples are provided.

Finally, we have the largest component of the SDK by far, the SDK samples. The SDK samples provide over a hundred real live examples of how the API classes work together to solve specific programming tasks, ranging from the very simple to pretty complex. This constitutes the largest knowledgebase of how to address issues using the Revit API. The API documentation in the help file lists all the classes and their methods and properties, but does not explain how they work together to solve specific tasks. For this, the current main source of information is the samples collection.

The main topic of this presentation is an exploration of how to make optimal use of this knowledge base by providing an in-depth overview and discussing techniques to search and manage the samples.

Besides the many subdirectories containing the source code and readme files for the samples themselves, the SDK samples subdirectory contains the following top level files:

* Converting 2008 applications to Revit 2009 API.doc
* Revit 2009 New Samples.doc
* SamplesReadMe.htm (using SamplesContent.htm and SamplesIndex.htm)
* SDKSamples2009.sln

My blog at <http://thebuildingcoder.typepad.com> and the Learning More section in the appendix at the end provide lists of additional resources for getting started with Revit.

## Non-SDK Samples

Before moving on to an overview and detailed discussion of the standard Revit SDK samples, let me point out a few interesting ones not included in the SDK. First and foremost, RvtMgdDbg is an absolute must-have for every serious Revit programmer. It is discussed in a separate session. We have implemented some specialised labs and samples for RME and RST which are discussed in separate sessions as well. Another more full-fledged RST linking sample is MidasLink, which is available on the ADN site.

### RvtMgdDbg

RvtMgdDbg is a debugging tool which allows you to interactively explore and traverse the Revit database, explore the data attached to building elements and the relationships between them. It is discussed in detail in a separate session

* **DE111-3** A Closer Look at the Revit Database with the Revit API

### Revit MEP Samples

Revit MEP 2009 provided the first MEP-specific API on the Revit platform. Before that, there were no RME-specific SDK samples. However, the generic Revit API can be used to access and modify an MEP model as well. For instance, the generic API can be used to implement an HVAC sample for calculating and resizing air terminals. We made use of it to implement a Revit 2008 MEP sample providing the following functionality and commands:

* Assign Flow to Terminals
* Change Size of Air Terminal
* Calculate Room CFM/SF
* Find Unhosted Elements
* Reset Demo
* About

The RME-specific API added in Revit 2009 makes it possible to implement more powerful applications making use of MEP-specific data. This is demonstrated by the Revit 2009 SDK samples AddSpaceAndZone, PowerCircuit, and RoofsRooms. We also implemented a non-SDK sample to analyse and display the connection hierarchy of an electrical system.

The Revit MEP API and both the SDK and non-SDK samples are discussed in depth in the separate session

* **DE301-2** The Ins and Outs of Revit MEP Programming.

### Revit Structure Labs, Link, Rebar, and Detailing Samples

The important features of the structural API are the exposure of structural elements and their usage. Revit Structure manages a simplified analytical model for analysing the structural behaviour of the building under stress in parallel with the physical model which is of interest to architects and most other parties. In programming for Revit Structure, a developer often faces special requirements. The main areas of interest are linking the parallel analytical and physical models maintained by Revit Structure with an external analysis application, and optimising the generation of detailing drawings. A number of SDK samples are dedicated to the Revit Structure specific API and topics especially interesting in the structural domain. These and additional specialised samples and related topics are discussed in greater detail in the separate session

* **DE215-3** Revit Structure API: From Bi-Directional Link to Rebar Detailing.

Finally, an additional sample available for demonstrating the linking of an external analysis application with Revit Structure is provided by MidasLink. This sample is of special interest even to non-structural developers, because it constitutes a complete professional application and is in productive use. It demonstrates the element handling, export and import forms, localisation, unit handling, and also includes the complete source code for an installer.

All of the samples mentioned above are available including source code in the appropriate AU session materials and from ADN.

## Managing Samples

In this section, we look at overview documentation, utilities and tools for managing the wealth of existing Revit SDK samples effectively. Here is an overview of the main resources to help achieve this:

Complete samples overview and readme

* SamplesReadme.htm

Main samples solution

* SDKSamples2009.sln
* Knowledge base
* Global search

Revit SDK samples spreadsheet

* Revit\_SDK\_Samples.xlsx

External sample manager applications

* RvtSamples
* Generic menu generator

### Complete overview and readme

Every sample in the Revit SDK includes a readme file. An \*.rtf file is always provided, and sometimes also a \*.txt one. This documentation is dispersed, and \*.rtf files are hard to search globally. The Revit 2009 SDK introduced a new overview documentation file SamplesReadme.htm. It is modelled on the ObjectARX samples readme and includes short descriptions for all the samples in one single file. It also provides various categorisation of the samples, for instance by topic or level of complexity, in order to make it easier to find the samples addressing your current issue of interest.

### The Main Samples Solution

The Revit SDK samples include a powerful tool to help manage the over hundred individual samples provided. On one hand, we wish to **compile** all the samples easily in order to load and execute them inside of Revit for testing and debugging purposes. We also want a comfortable method to **search** for solutions to specific tasks. The unified solution file SDKSamples2009.sln addresses both these needs. It includes both C# and VB samples.

Once the samples have been compiled, we also need to **load** them into Revit in order to execute, test and explore them further. In general, loading an external command requires adjusting Revit.ini or manually selecting the sample in the add-in manager. This would constitute a significant effort if we want to be able to test all hundred samples. In fact, it is not even possible to add all the samples to the Revit Tools > External Tools menu in one go. Happily, we implemented an efficient solution to this problem, the RvtSamples external menu generator application, which was included in the standard Revit SDK distribution starting with Revit 2009.

### Globally Searching all Samples

Within this global solution, you can easily use global searching to explore a specific programming task. You can search globally through the entire solution using Visual Studio > Edit > Find and Replace > Find in Files > Look in: Entire Solution. For example, if we are interested in creating a wall, we might search for ‘createwall’ and notice that it does indeed produce a large number of hits. To be more precise, in my installation, 880 files are searched and 74 matching lines found in 18 files. Looking at these in more detail, I notice that some of them contain the string 'CreateWall'. So next, I narrow down the search to this exact string and matching case. This significantly reduces the number of hits, to just 10 hits in 5 files, and one quickly discovers that the Journaling sample defines a method named just so which does exactly what we want. Looking at this method, one sees that the actual API method to create a new wall is NewWall(). Now we can search globally for ‘NewWall’ to immediately determine which seven of the hundred SDK samples can be used to explore wall creation.

### Compiling all Samples

In order to compile the samples, every project included in the solution needs to reference the appropriate version of RevitAPI.dll. On a default installation of Revit Architecture, MEP, or Structure, the location of RevitAPI.dll is

C:\Program Files\Revit Architecture 2009\Program\RevitAPI.dll

C:\Program Files\Revit MEP 2009\Program\RevitAPI.dll

C:\Program Files\Revit Structure 2009\Program\RevitAPI.dll

They are all identical, more or less, although some of the API functionality is only available when running within the appropriate flavour of Revit.

If you installed Revit to the default location, the reference path to RevitAPI.dll stored in SDKSamples2009.sln will already be correct and no change is required. Otherwise, you will need to change the path to the references to RevitAPI.dll in all your hundred Revit SDK sample projects. Luckily, this can be automated; please refer to the appendix on JHint, the Visual Studio Project HintPath Batch Processor.

By the way, whenever you modify the reference to RevitAPI.dll in order to compile any Revit add-in, you need to make sure that the 'Copy Local' flag maintained by Visual Studio for that reference is set to False. You can see the current setting by right clicking on the reference and selecting its properties in the context menu. If this flag is set to True, Visual Studio will create a local copy of RevitAPI.dll when compiling the plug-in and use this copy when loading it. This confuses the debugger and Revit when running the add-in, as well as unnecessarily polluting your hard disk.

To avoid having to reset this property when modifying an existing reference, simply do not delete the existing reference ... instead, add the new reference to the current assembly, and the old, existing data will be updated, so the new path will be stored, and at the same time the existing 'Copy Local' setting will be preserved.

Once the reference to RevitAPI.dll is set correctly for all the Revit SDK samples, you can build them all at once simply compiling the entire SDKSamples2009.sln solution.

This solution can also be used as a base for running and debugging into any one of the samples. In order to do so, the desired sample needs to be loaded into Revit. This can be achieved by manually editing Revit.ini to add the sample to the Revit Tools > External Tools menu, but that would also be tedious if done manually one by one for each sample. In addition, it is impossible to add all the samples as individual external commands to Revit.ini at the same time, since that would exceed the maximum number of permitted menu entries. See below for the effective alternative provided by the RvtSamples application.

### Revit Samples Spreadsheet

Revit\_SDK\_Samples.xlsx is an Excel spreadsheet database of all the Revit SDK samples including all the EC or external command data required for creating the Revit.ini entries for invoking the samples as external Revit commands. It also manages additional information such as classification into different categories. This data is useful in itself to provide an overview of the samples. It can also be used to automatically generate Revit.ini entries to invoke the commands for test purposes. Last but not least, it is used to generate a text file RvtSamples.txt which is used as an input file by the RvtSamples external application to automatically generate a menu hierarchy including all SDK samples. It includes the following information on each sample:

* Flavour: generic, rac, rst, rme
* Version: 8.0, 8.1, 9.0, 9.1, 2008.0, 2008.2
* Level: basic, intermediate, advanced
* Type: cmd, app, exe
* Language: cs or vb
* Topic: creation, element, parameter, etc.
* Files: number of source files
* Bytes: total source file size

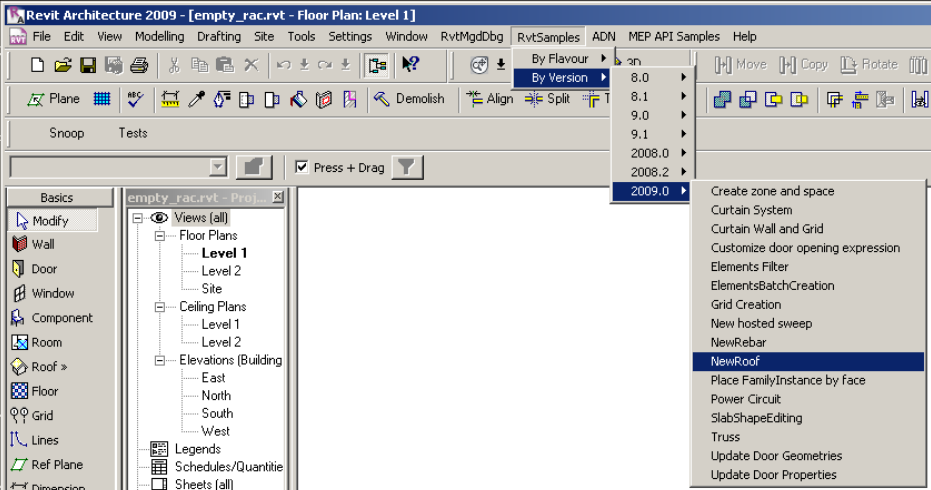
### Loading all Samples using RvtSamples

RvtSamples is a generic Revit menu generator which reads a text file RvtSamples.txt listing all the Revit SDK samples categorised in various ways and creates a new top level hierarchical menu providing simultaneous access to all of them.

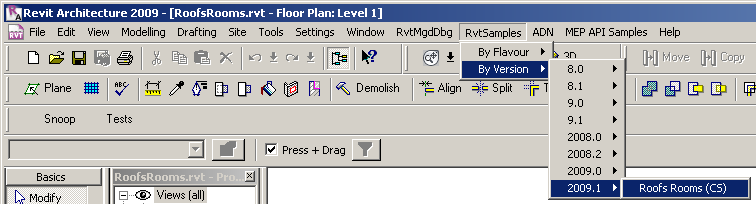
It is not limited to the SDK samples, you can add any other menu entry definitions you like, as well as or instead of the predefined ones for the SDK samples. For instance, in my system, I have added entries to define a second complete menu hierarchy for all the ADN, AU and The Building Coder samples.

RvtSamples is implemented as an external application and thus requires only a minimal edit of Revit.ini, an addition of two or maximally four lines, as opposed to the hundreds of lines required if you want to add the SDK samples individually. It defines its own menu hierarchy, which can be completely freely defined by the driving text file. Using the default database driven input file, it generates entries for all SDK samples, categorised in several different ways, e.g. by Revit flavour and original release version. It circumvents the maximum limit of about 50 entries that can be made to the Tools > External Tools submenu. Open the project, look at how to create menu entries, debug through steps in reading RvtSamples.txt and generating the menu hierarchy.

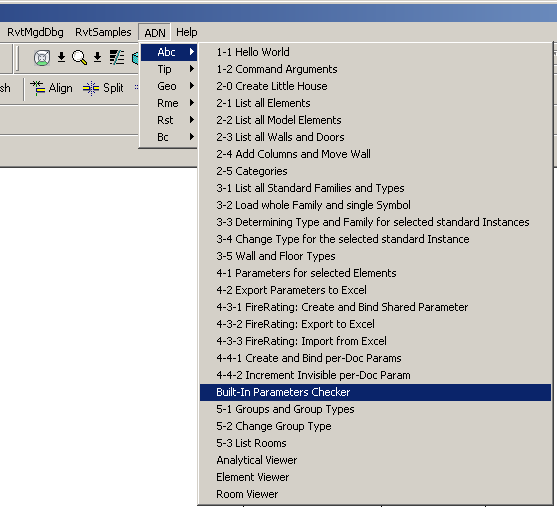
Here is a view of the RvtSamples menu hierarchy displaying the 2009 samples; they are arranged in two separate classifications, by topic and by version:



And here is the view after the addition of the RoofsRooms sample in the 2009 web update 1:



The menu entitled ADN is the additional menu created by me by #including my own additional text files at the end of the original RvtSamples.txt:



For details on the file format used to define the menu entries, please refer to the paragraph on RvtSamples in the Revit 2009 SDK Samples section below.

### Enhanced version

I have implemented an enhanced version of RvtSamples which supports an #include statement, similar to the functionality provided by the C and C++ language preprocessor. I use it to include my own regularly updated collection of samples. I have two collections of samples that I regularly work on and add to: the ADN and the Building Coder collections. To load them, I simply add the following two lines at the end of RvtSamples.txt:

#include C:/a/j/adn/train/revit/2009/src/AdnSamples.txt

#include C:/a/j/adn/train/revit/2009/src/bc/BcSamples.txt

## Samples Overview

As already stated, the Revit SDK includes over one hundred different sample applications, demonstrating just about all of the functionality provided by the Revit API. This constitutes a huge knowledge base, and makes it important for every Revit developer to know how to manage and search the information provided. The samples are currently the main source of information on how to solve specific programming tasks in the Revit API. Whenever faced with a new Revit API issue, one of the first places to search for information is in the samples collection. Most tasks that can be accomplished through the API are illustrated by one of them. Obviously we will not be able to discuss each and every one of these in detail in this session. This section presents a chronological categorisation matching the development of the API itself, followed by some details on specific groups of samples.

The rest of the presentation provides a slightly more detailed first impression of selected samples. Since we cannot discuss every one, we will focus on some important basic samples useful for initial exploration of the Revit API features and suitable for beginners, and on the new samples added in the Revit 2009 time frame, which are of interest to developers who have already worked with the API and explored the ones that have been available for longer. The information we present is extracted from the global readme file provided in SamplesReadme.zip and in the individual sample documentation files Readme\*.rtf.

### Basic Samples

These samples existed before Revit 9.0. Many of them were originally written using a COM API, which is no longer supported by Revit. They are all migrated to .NET now. With each sample, we also list the flavours of Revit it applies to, either Rac, Rme, Rst or All.

* ArchSample All
* BrowseBindings All
* CreateShared All
* FireRating All
* HelloRevit All
* MoveLinear All
* PhysicalProp All
* RevitCommands All
* StructSample All
* TypeSelector All
* ElementViewer All
* RoomViewer All
* AnalyticalViewer Rst

The samples listed above are often introductory in nature and provide a good starting point for initial exploration of the API. They are comparable to the Revit API Introduction labs, which constitute an alternative place to start such an exploration (cf. Learning More). The labs are more systematic than the basic samples, whereas the basic samples provide useful stand-alone functionality and more or less complete coverage. The basic samples often demonstrate access and analysis functionality, whereas creation of new element was added later and was a focus of the Revit 9.0 and 9.1 API releases. The following paragraphs highlight some of the important topics to explore when starting work with the Revit API. Remember that all the information presented here as well as a complete overview of all samples including a categorisation listing the basic samples is also available in SamplesReadMe.htm.

#### Interactive Element Selection

Before Revit 2008, explicit interactive element selection functionality was not available through the API. The pre-selected implicit selection set is passed in to the command. From 2008 onwards, the selection set can be changed from an external command. If an external command returns an error, the error set is highlighted and an optional message is displayed in the Revit user interface. This is demonstrated by the RevitCommands Selection sample, accessible by selecting an element and then executing Tools > External Tools > RevitCommands - Selection or RvtSamples > By Version > 8.0 > RevitCommands.Selection.

#### Object Properties

The Revit API uses generic properties to provide access to type and instance parameters. Both may include shared parameters. One sample to begin exploring this topic is the RevitCommands ElementData sample.

#### Import and Export Data

Exporting and importing data to or from a file or other applications such as Excel is a frequent requirement. Data such as element and type properties, location, and geometry may be exported and imported. The simplest sample demonstrating this is ArchSample, which exports to Excel. A more advanced but still basic sample is FireRating, which creates a new shared parameter for storing application specific data in the Revit model, exports it to Excel, and imports modified data from Excel back into and updates the Revit model. Finally, an advanced sample in this area is RDBLink.

#### Custom Data Addition

Application specific custom data, i.e. new parameters, can be added to the Revit model using the shared parameter mechanism. An application can add to new and existing elements and types. The data can be hidden from the user interface. The SDK sample FireRating VB and the Revit API Introduction Lab 4-3 C# commands demonstrate this by defining three commands:

* Apply Parameter
* Export to Excel
* Import from Excel

#### Element Modification

Some basic requirements are to move and rotate elements, swap element types programmatically, manage groups of elements, and generic property access. This is demonstrated by the Move Linear and Type Selector commands, among others.

#### Load Families and Types

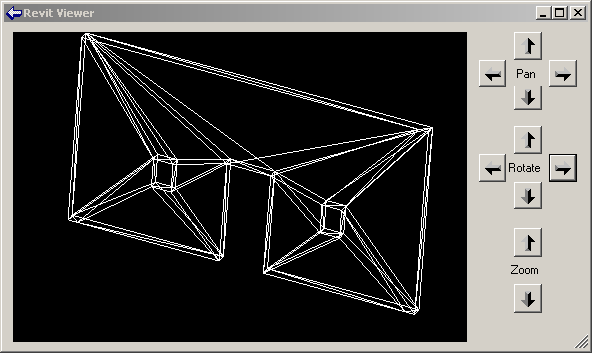
Revit makes strong use of families and types. Loading a whole family and a specific family symbol or type as well as searching for already loaded families and types in the current document is demonstrated by the RevitCommands Load Family and Load Family Symbol commands.

#### Geometry Viewers

A good starting point for exploring the different types of geometry of Revit elements is provided by the geometry viewer samples. The Revit SDK currently includes the following viewers:

* AnalyticalViewer
* ElementViewer
* RoomViewer
* RevitViewer
* ObjectViewer

All the viewers query a selected Revit element for some specific geometry, traverse it, break it down into simple line segments, and render those into a .NET System.Windows.Forms PictureBox. They can all be classified as pretty basic samples. The first four of these are all located in the subdirectory Viewers and are implemented in VB. They were introduced early on in the Revit API, with version 8.0. Of these four, the RevitViewer is a helper class, which implements the actual viewer window which is used by the other three to present three different views of certain subsets of the building model. ObjectViewer was introduced later, in the Revit 9.0 SDK, is written in C#, and defines its own rendering window. Below is an example of the form defined by RevitViewer; in this case, it is being driven by the ElementViewer sample and displaying the wall with a door and two little windows created by the external command defined by Lab2\_0\_CreateLittleHouse in the Revit API introduction labs. The form looks identical when driven by the AnalyticalViewer and RoomViewer samples, only the content displayed differs:

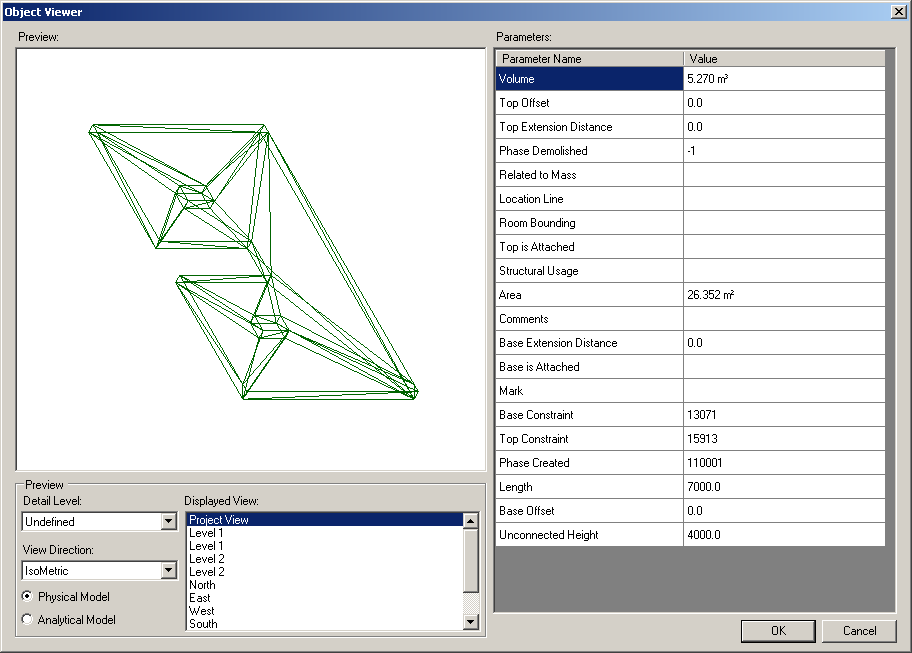


The AnalyticalViewer is for Revit Structure, to view the analytical building model. The analytical model is generally a simplified approximation of the building structure for structural analysis purposes, which is of most interest to the structural engineer. Revit Structure maintains the analytical model in parallel with the physical model, which is what architects and most other people are interested in.

ElementViewer is designed for all flavours of Revit and displays the standard element geometry.

RoomViewer is a viewer for displaying the two-dimensional outline of a room boundary.

Just like the three viewers, just discussed, ObjectViewer demonstrates viewing elements. It allows the user to select between the analytical and physical model, and also which view to display. You might thus say that it is a combination of the AnalyticalViewer and the ElementViewer with some additional features.



ObjectViewer includes both generic functionality usable in all Revit flavours, as well as some Revit Structure specific functionality. It displays a selected single element in a preview window and retrieves geometry from either the analytical or physical model. It also includes some additional useful odds and ends such as defining an own error message exception handling class and some unit handling functionality. ObjectViewer was originally introduced in Revit 9. Another very similar sample named ObjectViewerII was introduced in 9.1. ObjectViewerII was removed in the 2008.2 SDK update, being very similar to ObjectViewer.

For all of the viewers, the steps to run are similar: simply select the element whose geometry or the room whose boundary to display and start the external command. The analytical viewer will check whether any analytical model is available, which requires Revit Structure functionality. In the case of the ObjectViewer, all display views are listed, the detail level can be selected, and analytical and physical model can be switched.

### Revit 9.0 Samples

The sample categorisation into the different flavours of Revit is not always trivial. Some samples demonstrate features that work in all flavours of Revit, such as the creation of walls or section views, but rely on some flavour-specific detail for simplified calculation purposes, such as CreateViewSection and CreateWallsUnderBeams which make use of the RST analytical model.

* AllViews All
* AnalyticalSupportData\_Info Rst
* AreaReinCurve Rst
* AreaReinParameters Rst
* BarDescriptions Rst
* BeamAndSlabNewParameter Rst
* CreateBeamsColumnsBraces All
* CreateComplexAreaRein Rst
* CreateDimensions All
* CreateSimpleAreaRein Rst
* CreateViewSection Rst
* CreateWallinBeamProfile Rst
* CreateWallsUnderBeams Rst
* DeleteDimensions All
* DeleteObject All
* DesignOptionReader All
* GenerateFloor All
* InPlaceMembers Rst
* LevelsProperty All
* Loads Rst
* MaterialProperties All
* ObjectViewer All
* PhaseSample All
* RotateFramingObjects All
* SlabProperties Rst
* StructuralLayerFunction Rst
* VersionChecking All

### Revit 9.1 Samples

Here is an overview of the Revit 9.1 SDK samples, sorted by Revit flavour and alphabetically. ModelLines and Materials were originally provided in two versions each, suffixed with 1 and 2. Later, in the 2008.2 update, these separate variations were united into one each. ObjectViewerII was removed in the 2008.2 SDK update, being very similar to ObjectViewer.

* FrameBuilder All
* Journaling All
* ModelLines All
* Openings All
* ReferencePlane All
* Rooms All
* SharedCoordinateSystem All
* BoundaryConditions Rst
* CreateBeamSystem Rst
* FoundationSlab Rst
* Materials Rst
* ObjectViewerII Rst
* Reinforcement Rst

### Revit 2008 Samples

The samples added in Revit 2008 and in the Revit 2008.2 SDK update are described in a separate document in the Revit 2008 SDK, "Revit 2008 New Samples.doc". The original 2008 SDK included the following new samples:

* APIAppStartup
* ApplicationEvents
* AutoTagRooms
* BlendVertexConnectTable
* CurvedBeam
* FamilyExplorer
* ImportExportDWG, renamed to ImportExport
* InplaceFamilyAnalyticalModel3D
* NewOpenings
* NewPathReinforcement
* PathReinforcement
* ProjectInfo
* ProjectUnit
* ShaftHolePuncher
* SpotDimension
* TagBeam
* TestFloorThickness
* TestWallThickness
* Toolbar
* TransactionControl
* ViewPrinter
* VisibilityControl

ImportExportDWG was renamed to ImportExport in the Revit 2009 timeframe, when support for other file formats was added.

The Revit 2008.2 SDK update added two new samples:

* RDBLink
* RoomSchedule

### Revit 2009 Samples

16 new samples were added to the initial release of the Revit 2009 SDK. One of these is RvtSamples, discussed above, which can be used to load all other samples into Revit.exe for testing and debugging. Another important 2009 sample that every developer should explore is ElementsFilter, because almost every application needs to make use of the new element filtering introduced in the Revit 2009 API. For the first time, we also have some RME-specific samples, as well as the RAC and RST ones.

The RoofsRooms sample was added to the Revit SDK in the 2009.1 SDK update in the summer of 2008, matching the Revit 2009 web update 1.

Here is an overview or the 2009 SDK samples categorised by the different Revit flavours:

All

* CurtainSystem
* ElementsBatchCreation
* ElementsFilter
* GridCreation
* ImportExport
* NewRoof
* PlaceFamilyInstanceByFace
* RvtSamples
* SlabShapeEditing

RAC

* CurtainWallGrid
* DoorSwing
* NewHostedSweep

RME

* AddSpaceAndZone
* PowerCircuit

RST

* NewRebar
* Truss

RAC and RME

* RoofsRooms

Here is a quick first impression of these new samples.

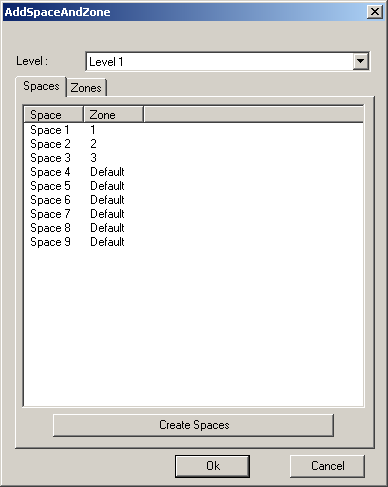
#### AddSpaceAndZone

AddSpaceAndZone is an MEP specific sample implemented in C#. It retrieves and lists all spaces and zones using element filter. It creates new spaces via NewSpaces() method, by automatically detecting and creating a new space for each closed wall loop or closed space separator. It adds and removes spaces in a zone using the Zone class methods AddSpaces() and Remove():

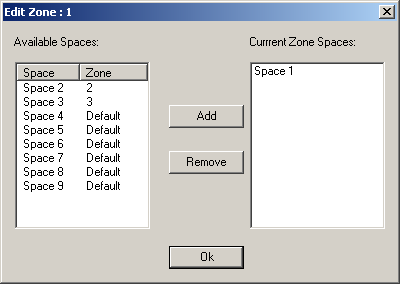
* Use element filtering to get all space and zone elements in a specified level
* Create space elements for each closed wall loop or closed space separation
* Create a new zone element in a specified level and phase
* Add and remove spaces in a zone element

The new element filter classes are used to search for all existing spaces and zones in the project. NewSpaces() identifies all closed wall loops and generates a new space element for each closed loop found. Once the spaces are available, zones can be created and the spaces assigned to appropriate zones. Spaces are created using the method ElementSet NewSpaces( Level, Phase, View). Spaces in a zone are added and removed using the Zone class member methods AddSpaces( SpaceSet ) and Remove( SpaceSet ). Instructions:

* Open a Revit MEP 2009 file containing some enclosed loops made up of walls. A sample project file WallLoopForAddingSpaceAndZone.rvt is available in the sample folder.
* Run the command.
* The AddSpaceAndZone dialog is shown to display all the Space and Zone elements filtered by level. The Spaces are displayed as a list in the Spaces Tag page, the Zones as a tree list to show the spaces each Zone element contains.
* Click the 'Create Spaces' button in Space Tag page to create spaces for each closed wall loop or closed space separation in the selected level.
* Click the 'Create Zone' button in Zones Tag page to create a zone in a current level and phase of the active view.
* Select a zone and click the 'Edit Zone' button. A ZoneEditorForm displays the available spaces and the spaces the selected zone contains in the current level. You can select some spaces in the available spaces list, and add them to the current zone, or select some spaces in the current spaces list and remove them from the current zone.



Creating new spaces for all closed wall loops



Creating new zones and assigning spaces to zones

This sample is also discussed in the RME session mentioned above.

#### CurtainSystem

Applies to all flavours of Revit, implemented in C#. Creates curtain systems on specified faces, deletes selected curtain systems, adds curtain grids to curtain systems, and removes curtain grids from curtain systems. The test element must be a paralleliped mass. Steps to run the command:

* Open or new a Revit project and make sure it contains a paralleliped mass element. A sample project file CurtainSystem.rvt is provided in the sample folder.
* Select a paralleliped mass element.
* Run the command, displaying the main dialogue.
* Click the ‘Create Curtain System’ button to display a new dialog used to create curtain systems.
* Exercise the other buttons.

#### CurtainWallGrid

Create a curtain wall with specified level and wall type. Applies to Revit Architecture and implemented in C#. Retrieves the curtain wall grid and edits its properties, grid lines and segments: add new, lock, unlock, move; add, delete, add all. This command mimics part of the UI operations on segments, but only supports simple cases. You can also retrieve and manipulate attached mullions: add and remove.

#### DoorSwing

Create, maintain and schedule door opening parameters. Applies to Revit Architecture and implemented in C#. Can be customised for country specific standards. Updates schedule information from geometry and vice versa.

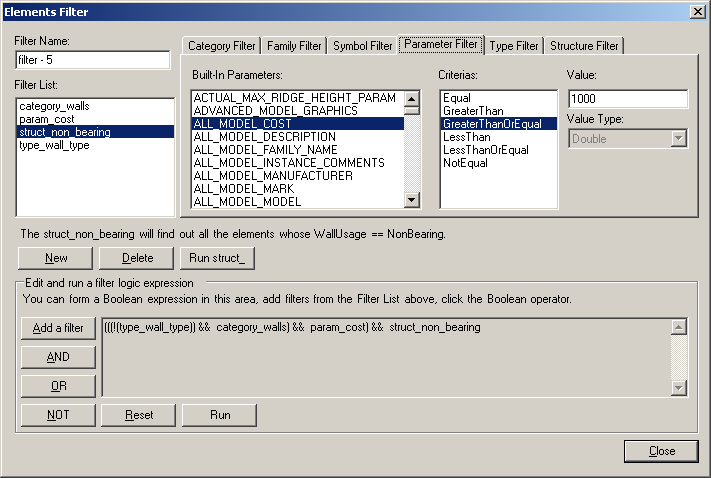
#### ElementsBatchCreation

Demonstrates creation of new elements using the new batch functionality provided in the Revit 2009 API. Supports all flavours of Revit, implemented in C#. Create new elements using the following batch creation methods and arguments:

* Area NewAreas( List<AreaCreationData> )
* FamilyInstance NewFamilyInstances( List<FamilyInstanceCreationData> )
* Room NewRooms( List<RoomCreationData> )
* TextNote NewTextNotes( List<TextNoteCreationData> )
* Profiled Wall NewWalls( List<ProfiledWallCreationData> )
* Rectangular Wall NewWalls( List<RectangularWallCreationData> )

#### ElementsFilter

The new Revit 2009 element filtering provides filtered access to elements. Several different types of filters are available. Filters can be atomic, based on category name, built-in category, family name, symbol name, type, structural classification, or parameter value. When searching for elements based on parameter value, the comparison criterion can be specified as equal, gt, ge, lt, le, ne, etc. The execution speed of the new element filtering is significantly higher than the element iteration in previous versions. All aspects of the functionality are demonstrated by the SDK sample application ElementsFilter:



Here is a very simple example of using the new filtering feature, selecting all walls from the Revit database. In 2008, one would have used a loop to pick out the walls from the collection of all document elements:

public static ElementSet GetAllWalls\_2008( Application app )

{

ElementSet elems = app.Create.NewElementSet();

ElementIterator iter = app.ActiveDocument.Elements;

while( iter.MoveNext() )

{

Element elem = iter.Current as Element;

if( elem is Wall )

{

elems.Insert( elem );

}

}

return elems;

}

In 2009, using filtering, this method is more concise, no loop is required, and only wall elements are returned:

public static List<Element> GetAllWalls( Application app )

{

List<Element> elements = new List<Element>();

Filter filterType = app.Create.Filter.NewTypeFilter( typeof( Wall ) );

app.ActiveDocument.get\_Elements( filterType, elements );

return elements;

}

Here is a slightly more complex example, which includes filtering for a specific parameter value. Here we are selecting all model group types in the database. In 2008, this involves iterating over all elements, selecting GroupType objects, and then checking a parameter for a specific value:

public static ElementSet GetAllModelGroupTypes\_2008( Application app )

{

ElementSet elems = app.Create.NewElementSet();

ElementIterator iter = app.ActiveDocument.Elements;

while( iter.MoveNext() )

{

Element elem = iter.Current as Element;

if( elem is GroupType )

{

try

{

string s = elem.get\_Parameter(

BuiltInParameter.SYMBOL\_FAMILY\_NAME\_PARAM ).AsString();

if( s.Equals( LabConstants.gsGroupTypeModel ) )

{

elems.Insert( elem );

}

}

catch( Exception )

{

}

}

}

return elems;

}

In 2009, this can all be achieved more concisely and efficiently using filtering, by combining a type and a parameter filter with a Boolean and filter:

public static List<Element> GetAllModelGroupTypes( Application app )

{

List<Element> elements = new List<Element>();

Filter filterType = app.Create.Filter.NewTypeFilter( typeof( GroupType ) );

Filter filterParam = app.Create.Filter.NewParameterFilter(

BuiltInParameter.SYMBOL\_FAMILY\_NAME\_PARAM, CriteriaFilterType.Equal,

LabConstants.gsGroupTypeModel );

Filter filterAnd = app.Create.Filter.NewLogicAndFilter( filterType, filterParam );

app.ActiveDocument.get\_Elements( filterAnd, elements );

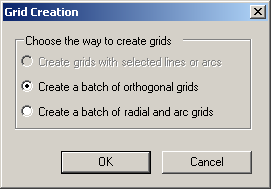
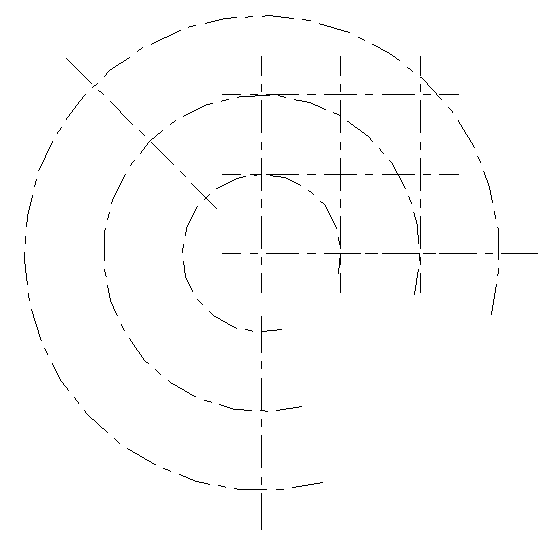
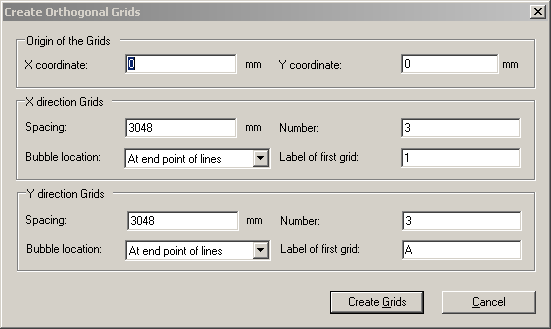
return elements;

}

This sample applies to all flavours of Revit and is implemented in C#. It demonstrates retrieving elements using atomic filters such as category name, built-in category, family name, symbol name, type, structural classification, parameter value eq, gt, ge, lt, le, ne, and also using Boolean combinations of atomic filters created by and, or, not. It demonstrates how to use, populate and display the filters and the results. Additionally, it demonstrates how to make use of the .NET ResourceManager, DataTable and ProgressBar classes.

#### GridCreation

This sample shows how to create grids and modify grid properties in all flavours of Revit and is implemented in C#. Grids can be generated based on selected lines and arcs and support orthogonal linear, radial and arc grids. You can specify the origin, spacing, span, number of lines, labels, and bubble locations.

Grid Creation

#### ImportExport

ImportExport was previously named ImportExportDWG. This sample was enhanced and renamed to ImportExport in Revit 2009 to handle additional file formats. It now exports to DWG, 2D and 3D DWF and DWFx, gbXML and FBX and imports DWG and image files. New API classes to support this were added. This sample is written in C# and applies to all Revit platforms.

#### NewHostedSweep

Create or modify hosted sweep such as fascia, gutter and slab edge. This sample is written in C# and applies to the RAC platform. It demonstrates how to create hosted sweep and modify their properties. There are three types of hosted sweep, fascia, gutter, and slab edge. All the three types of hosted sweep can be created.

#### NewRebar

NewRebar demonstrates how to create a new Rebar instance and define a custom Rebar shape. It applies to Revit Structure and is implemented in C#.

#### NewRoof

Create or edit a footprint or extrusion roof. Implemented in C# and applies to all the three Revit flavours.

* Retrieve and list all footprint and extrusion roofs in the active document.
* Create a footprint roof with default properties by specifying a curve loop, a wall loop, or a combination of walls and curves, a level and a roof type.
* Create an extruded roof by providing a profile which is a series of connected lines or arcs (not closed in a loop), a roof type, a reference plane, and an extrusion start and end value.
* Select a footprint roof in the list to define whether a roof line is slope-defining and other properties such as overhang, offset from base, extend into wall. The roof type can be changed by selecting a different type.
* Select an extrusion roof in the list to change the value of extrusion start, extrusion end, the roof type and reference level.

A sloped roof is not demonstrated by this SDK sample. For an example of editing a sloped roof, please refer to the Revit API Introduction labs. The last parameter of the NewFootPrintRoof method is an ElementIdSet which maps the footprint profile curves to model curves in the Revit sketch, on which you can set the DefinesSlope and SlopeAngle properties.

#### PlaceFamilyInstanceByFace

Create a family instance on a face. Applies to all flavours of Revit, implemented in C#. Before running, load the family definitions Line-based.rfa and Point-based.rfa from the sample folder. Select an element with faces, e.g. a wall to place a family instance on.

#### PowerCircuit

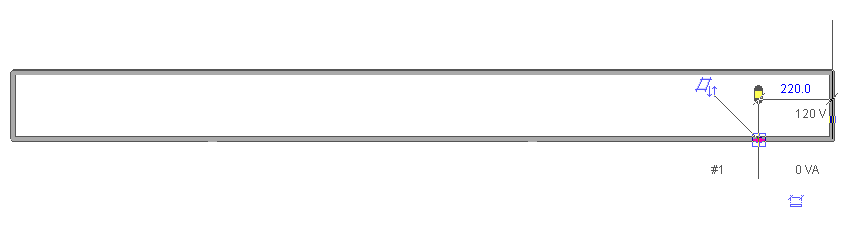
PowerCircuit is an RME C# sample application demonstrating the following functionality, similar to the RME Circuit Editor toolbar:

* Operate power circuits
* Toolbar-based user interface
* Create a power circuit with selected elements
* Add and remove circuit element to and from a circuit
* Select or disconnect a circuit panel

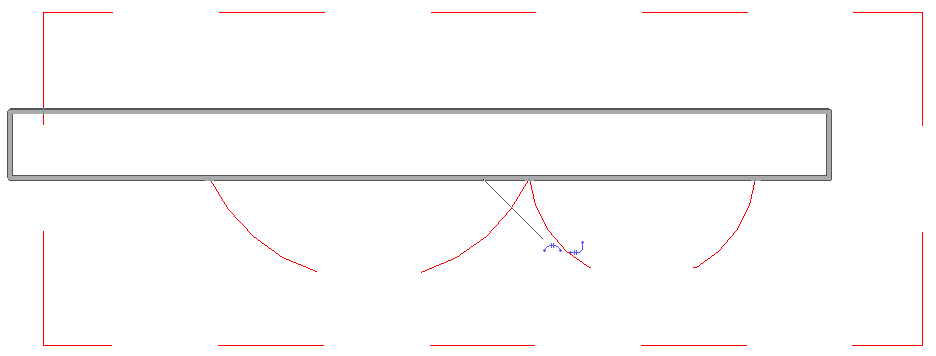
A new electrical system containing selected elements can be created using NewElectricalSystem(). New elements can be added to an electrical system using AddToCircuit(). The elements are family instances. The family instance class has a new property MEPModel, which provides access to the family instance’s MEP model, analogously to the structural analytical model for the family instance handled by the AnalyticalModel class and family instance property in Revit Structure. If a family instance has a non-null MEP model property, it can be used to access the instance’s connector manager and set of electrical systems. The connector manager in turn can return all the connectors as well as all the unused connectors, and the electrical systems provide methods to select and disconnect panels, etc.



The button-based user interface



A wall hosting three electrical elements



The auto-generated circuit connecting the three elements

This sample is also discussed in the RME session mentioned above. It shows how to operate power circuits, as well as how to handle interactive element selection in Revit, use a ResourceManager to manage images and localisable string resources, and implement a toolbar user interface for an external command. It mainly exercises classes from the Autodesk.Revit.MEP namespace. It provides the following functionality:

* Create a power circuit with selected elements, edit a power circuit, or add or remove an element to or from a circuit. The elements should have unused electrical connectors with same voltage definition and pole numbers.
* Select a panel for a circuit, or disconnect panel from a circuit if the circuit has a panel.

Creating a new circuit is achieved by the creation document method NewElectricalSystem(), passing in a list of circuit elements and an electrical system type from the ElectricalSystemType enumeration, which can currently be one of Data, PowerCircuit, Telephone, Security, FireAlarm, NurseCall, Controls, and Communication. For editing the circuit, the elements make use of the FamilyInstance MEPModel property, which provides access to the elements connector manager and electrical systems.

#### RoofsRooms

Determine the roof elements bounding a room or space from above. Applies to Revit Architecture and MEP, implemented in C#. Separate sample files are provided for RAC and RME. All rooms and spaces are found and their roofs are determined and listed. This sample was provided in the Revit 2009 SDK update 1. If you have not yet downloaded this version of the SDK, it is available from the Revit download page on the Autodesk product and the ADN web sites.

It demonstrates how to check whether a room or space has a bounding roof. To do so, it queries the roof and the room or space for its geometry and then checks for intersections between the various elements. It retrieves the room or space closed shell as a GeometryObjectArray, and extracts every solid element from it. It also extracts the roof solid, then feeds the two pairs of solids into the method AreSolidsCut(). This checks whether any of the solid faces overlap each other.

#### RvtSamples

As mentioned above in the section on Managing Samples, RvtSamples is an external Revit application providing direct access to load and execute all external command SDK samples from an own Revit menu named "RvtSamples". It applies to all flavours of Revit, and is implemented in C#. Makes use of the classes Autodesk.Revit.ControlledApplication and Autodesk.Revit.MenuItem. RvtSamples is actually a generic menu generator, and can be used to manage other external commands as well as or instead of the Standard SDK ones. It reads a text file containing sample menu definitions that may look like this:

/RvtSamples/By Flavour/Rme/AddSpaceAndZone

Create space and zone elements and edit zone in Revit MEP

Z:\SDK2009\Samples\AddSpaceAndZone\CS\bin\Debug\AddSpaceAndZone.dll

Revit.SDK.Samples.AddSpaceAndZone.CS.Command

/RvtSamples/By Flavour/All/All Views

Generate a new sheet for user selected views

Z:\SDK2009\Samples\AllViews\CS\bin\Debug\AllViews.dll

Revit.SDK.Samples.AllViews.CS.Command

This sample provides the following functionality: Extract samples information from a provided text file and parse them into menu items. The file can be customized and it must follow a fixed format. Every menu item consists of four lines.

The first line represents the popup menu hierarchy. Each item in the hierarchy is separated by "/" or "\". If the line starts with "external tools" (case insensitive), the menu item will be added under the menu [Tools] - [External Tools], otherwise a new top menu will be added to Revit following the hierarchy. For instance, "/external tools/RvtSamples/By Flavour/All/About Revit" will add a menu item RvtSamples > By Flavour > All > About Revit under the Revit standard Tools > External Tools menu. However, "/RvtSamples/By Flavour/All/About Revit" will add a new top level menu [RvtSamples] to Revit and add By Flavour > All > About Revit under that.

The second line represents the description of the menu item, the third the path of the assembly location of the menu item, and the last the class name of the menu item.

#### SlabShapeEditing

Edit a slab shape by using the SlabShapeEditor class in Revit Architecture or Structure, implemented in C#. Creates SlabShapeVertex and SlabShapeCrease instances. Demonstrates how to edit a slab shape by adding more vertices or creases to the slab surface. Steps to run the sample:

* Draw a slab and select it.
* Run the command.
* Click the "Create vertex" button and select a point in the picture box to create a vertex.
* Click the "Create crease" button and select a point in the picture box to create a crease.
* Click the "Select" button, and select a newly created vertex or crease in the picture box.
* Input a distance to move the selected vertex or crease and click "Update".
* Click "Reset" to restore the original shape of slab.
* To rotate the slab in the picture box, click and drag using the right mouse button.

#### Truss

Create a truss of a selected truss type, change beam types, edit the profile of the truss. Applies to Revit Structure and implemented in C#.

## Appendix

### Acronyms

* ADN Autodesk Developer Network
* AEC Architecture, Engineering, Construction
* API Application Programming Interface
* BIM Building Information Model
* HVAC Heating, Ventilation, and Air Conditioning
* MEP Mechanical, Electrical, Plumbing
* RAC Revit Architecture
* RME Revit MEP
* RST Revit Structure
* SDK Software Development Kit

### Learning More

Here are some suggestions on where to go for further information. First of all, of course, we suggest you learn to know and use the online help and SDK samples which are already at your disposal. After that, here are some further suggestions:

DevTV Introduction to Revit 2008 Programming  
<http://adn.autodesk.com/adn/servlet/item?siteID=4814862&id=10194238&linkID=4901650>

* Recording of Revit 2009 Programming Introduction Webcast  
  <http://adn.autodesk.com/adn/servlet/index?siteID=4814862&id=5475217&linkID=4901650>  
  <http://www.adskconsulting.com/adn/cs/api_course_sched.php>
* Discussion Groups  
  <http://discussion.autodesk.com> > Revit API
* API Training Classes  
  <http://www.autodesk.com/apitraining>
* ADN, The Autodesk Developer Network  
  <http://www.autodesk.com/joinadn>
* DevHelp Online for ADN members  
  <http://adn.autodesk.com>
* The Building Coder, a Revit API blog  
  <http://blogs.autodesk.com/thebuildingcoder>  
  <http://thebuildingcoder.typepad.com>

### JHint, the Visual Studio Project HintPath Batch Processor

If you need to change the path to the references to RevitAPI.dll in all your hundred Revit SDK sample projects and want to do so by editing them manually one by one, you are in for a daunting task. Luckily, this can be automated.

The Visual Studio project file is actually an XML file, so all relevant project information is encoded in tags and attributes. The location of RevitAPI.dll on your system is encoded within the project file, inside a <HintPath> tag. You can easily use search and replace to redefine the location of RevitAPI.dll by modifying the hint path globally across hundreds of project files. One way to do this would be to use Visual Studio itself, either straight search and replace if you have one specific hint path value that you would like to modify, or by using regular expressions for more flexibility. I wrote a command line utility to automate this task, the HintPath Batch Processor jhint.exe:

HintPath Batch Processor 1.0.0.1

Copyright (C) 2007 by Jeremy Tammik, Autodesk Inc.

usage: jhint [-?bhqrv] [-n new\_hintpath] [-o old\_hintpath\_regex] filespecs

Options:

-b --backup create backup files (no)

-h --help display help message (no)

-n --newhintpath specify new HintPath replacement (null)

-o --oldhintpath specify old HintPath regular expression pattern (null)

-q --quiet quiet mode (no)

-r --recursive recursively traverse subdirectories (no)

-v --verbose print verbose messages (no)

filespec may include both root directory and filename pattern, for example:

C:\a\j\pro\jhint\test\\*.csproj

.\test\Project1.vbproj

Project2.csproj

old HintPath regular expression search pattern examples:

mgd.dll matches all files containing the substring "mgd.dll"

\\AutoCAD 2007\\ matches all files in some "AutoCAD 2007" subdirectory

If you are interested in more information about this tool, please contact me by mail or drop me a note at my blog <http://thebuildingcoder.typepad.com>.

Actually, the simplest solution is probably to manually create copies of RevitAPI.dll in the three locations that the project file expect them :-)