**The Ins and Outs of Revit® MEP Programming**

Jeremy Tammik – Autodesk Inc.

**DE301-2** Use the new MEP-specific Revit API features introduced in Revit MEP 2009 as well as the generic Revit platform API to analyze and modify a Revit MEP model. Address both HVAC and electrical domains. Solve tasks such as assigning a specific flow to air terminals based on the room requirements; changing the size of an air terminal to match the required flow; calculating and displaying the room CFM/SF in a colour fill; finding unhosted elements; analyzing the relationship graph between all rooms and air terminals; making use of advanced Revit 2009 element filtering capabilities; iterating over all electrical equipment and lighting fixtures to extract their electrical data and location for electrical load analysis and determine their hierarchical connection relationships for displaying an intuitively appealing system overview in a tree view. This class assumes basic knowledge of Revit programming.

**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 Ins and Outs of Revit MEP Programming

Jeremy Tammik, Consulting Analyst

Developer Technical Services, EMEA

Autodesk SARL

[Introduction 3](#_Toc212621049)

[Acknowledgements 3](#_Toc212621050)

[The Revit API Platform 3](#_Toc212621051)

[The Revit MEP API 4](#_Toc212621052)

[RME Spaces and Zones 5](#_Toc212621053)

[RME API Update in WU2 5](#_Toc212621054)

[Revit MEP SDK Samples 5](#_Toc212621055)

[AddSpaceAndZone 5](#_Toc212621056)

[PowerCircuit 6](#_Toc212621057)

[HVAC Sample 8](#_Toc212621058)

[Engineering Workflow 8](#_Toc212621059)

[Sample Implementation 8](#_Toc212621060)

[Assign Flow to Terminals 9](#_Toc212621061)

[Size Air Terminals 10](#_Toc212621062)

[Flow Density Design Validation 11](#_Toc212621063)

[Find Unhosted Elements 11](#_Toc212621064)

[About and Progress 11](#_Toc212621065)

[Electrical Sample 12](#_Toc212621066)

[Electrical System Browser 13](#_Toc212621067)

[Connection Hierarchy 14](#_Toc212621068)

[Connection Hierarchy 2 14](#_Toc212621069)

[Appendix 15](#_Toc212621070)

[Learning More 15](#_Toc212621071)

[Acronyms 15](#_Toc212621072)

## Introduction

We discuss the use of the Revit API for programming Revit MEP, or RME. First we provide an overview of the generic and the MEP specific Revit API and the SDK samples provided to demonstrate its use. We then discuss two additional samples, one HVAC oriented one making use of the generic Revit API to analyse and manipulate duct air terminals, and an electrical sample for analysis and display of an electrical system in a tree view showing the connection hierarchy.

### Acknowledgements

Many thanks to Martin Schmid, Autodesk, who provided inspiration and support for the analysis and design of the HVAC and electrical sample applications. And as always to my fellow colleagues in DevTech and especially in the AEC workgroup for being such a great team to work with.

### The Revit API Platform

Most of the Revit API is generic and applies to all three flavours of the Revit product, i.e. Architecture, MEP and Structure. All three flavours of Revit also include the same RevitAPI.dll .NET assembly making the API available to third-party applications. However, some specific additional features exist for each of the flavours as well, for instance some room-related functionality in Revit Architecture and support for the analytical model in Revit Structure.

Up until Revit 2008, no MEP-specific functionality was provided by the API. It was however still possible and useful to make use of the generic Revit platform API to access and manipulate a Revit MEP model. This is demonstrated by the HVAC sample using generic parameter access which we discuss below.

Revit 2009 added MEP-specific functionality to the API for the first time. We will discuss the functionality and new SDK samples provided and present an electrical example application that makes use of this to explore the data and hierarchical relationships in a system and present these in an intuitive manner.

### The Revit MEP API

We assume that you already know the Revit programming basics, so we will only discuss the RME specific API at this point. It is included in the standard RevitAPI.dll .NET assembly, which is identical for all three flavours of Revit and documented in the SDK ‘Getting Started Revit API 2009.doc’. If API functionality not supported by the currently running flavour is accessed, the call will simply do nothing. The specific additional features for RME were first introduced in Revit 2009. The new functionality includes

* New classes
* New properties
* New categories
* MEP project information
* Green Building XML support

Some of the new classes and properties are

* ElectricalEquipment, LightingDevice, LightingFixture
* MechanicalEquipment
* FamilyInstance.MEPModel

Among other things, the RME specific API functionality enables an application to create and modify elements of the categories

* Connector
* ElectricalSystem
* Space
* Zone
* Space Tag

Some of the new classes added to the Autodesk.Revit.Elements, MEP, Space and Symbols namespaces are:

* Autodesk.Revit.Elements.ElectricalSystem
* Autodesk.Revit.Elements.ElectricalSystemSet
* Autodesk.Revit.Elements.Space
* Autodesk.Revit.Elements.SpaceSet
* Autodesk.Revit.Elements.Zone
* Autodesk.Revit.MEP.Connector
* Autodesk.Revit.MEP.ConnectorManager
* Autodesk.Revit.MEP.ConnectorSet
* Autodesk.Revit.Space.BoundarySegment
* Autodesk.Revit.Symbols.MEPBuildingConstruction
* Autodesk.Revit.Symbols.SpaceTag
* Autodesk.Revit.Symbols.SpaceTagType

Revit MEP project info is provided through the element Autodesk.Revit.Elements.gbXMLParamElem and its properties, which is accessed through the ProjectInfo.gbXMLSettings property. The available data includes PostalCode, ProjectLocation, BuildingService, BuildingConstruction, GroundPlane, ShadingSurfaces, and ProjectPhase.

A Revit MEP project can be exported to the Green Building XML file format using the new method

Document.Export( string folder, string name, GBXMLExportOptions );

Access to an electrical system is provided by

ElectricalSystem sys;

ConnectorSet connectors = sys.ConnectorManager.Connectors;

### RME Spaces and Zones

In Revit 2009, the workflow between Architecture and MEP was improved. In 2008, the MEP user had to Copy/Monitor rooms from the architect. Rooms are unsuitable for MEP analysis, because they often have the wrong height or are too large for the analysed region. In 2009, one uses spaces instead of rooms, and zones to group spaces. This enables a subdivision of a room into exterior and interior subspaces. If a 2008 project is upgraded to 2009, the Copy/Monitored rooms are automatically converted to spaces. The HVAC sample application we present here uses spaces instead of rooms. More information on this topic is available in Kyle Bernhardt’s blog at <http://inside-the-system.typepad.com> in postings in February and March 2008.

### RME API Update in WU2

The HVAC sample we discuss here uses the current RME API which was provided in the Revit MEP 2009 web update 2. The updated RME API is in RevitAPI.dll and addresses two issues:

1. The FamilyInstance.Space property is implemented:

FamilyInstance fi; // get an instance

// query the space in which the instance is located during the last phase:

Space space = fi.Space;

// query the space in which the instance is located in the given phase:

Space space2 = fi.get\_Space( phase );

2. The ConnectorManager.Connectors property providing access to the electrical system returns logical as well as physical type connectors:

ElectricalSystem sys;

ConnectorSet connectors = sys.ConnectorManager.Connectors;

## Revit MEP SDK Samples

The introduction of an MEP specific API was accompanied by two new SDK samples to demonstrate the new functionality: AddSpaceAndZone demonstrates handling the new space and zone elements, and PowerCircuit demonstrates manipulation of electrical power circuits.

### AddSpaceAndZone

This sample creates the spaces and zones required for HVAC analysis. Its main features are:

* Retrieve and list all spaces and zones using element filter
* Create new spaces via NewSpaces() method
* Add and remove spaces in a zone:  
  for this, the Zone class provides the member methods AddSpaces() and Remove()

Spaces are created using the method

ElementSet NewSpaces( Level level, Phase phase, View view )

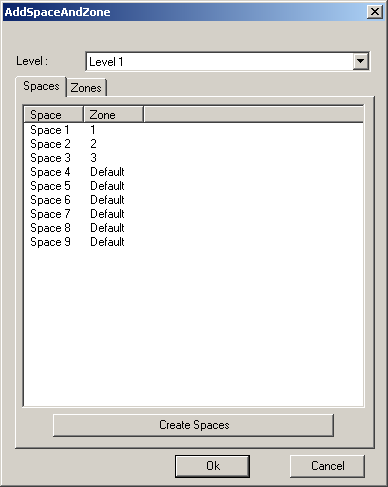
This method automatically detects and creates a new space for each closed wall loop or closed space separator.

Spaces in a zone are added and removed using the Zone class member methods

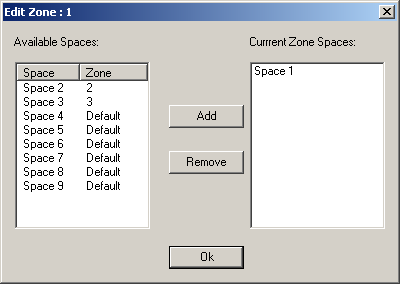
AddSpaces( SpaceSet )

Remove( SpaceSet )

To run the sample, open a Revit MEP 2009 file containing some enclosed loops made up of walls. A sample project file WallLoopForAddingSpaceAndZone.rvt is provided 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. You can click the 'Create Spaces' button in the 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 the Zones Tag page to create a zone in a current level and phase of the active view. If you select a Zone and click the 'Edit Zone' button, a ZoneEditorForm displays the available spaces list and the spaces the selected zone contains in current level.



Select some spaces in the available spaces list and click Add to add them to the current Zone. Select some spaces in the current spaces list and click Remove to remove them from the current Zone.

### PowerCircuit

This Revit MEP electrical sample shows how to operate power circuits, similar to the RME Circuit Editor toolbar:

* Create a new power circuit with selected elements
* Edit circuit and add and remove circuit elements
* Select or disconnect a circuit panel

It supports you in exploring the Autodesk.Revit.MEP namespace and the MEPModel and ElectricalSystem classes.

From a programming point of view, some additional interesting aspects of this sample are that it also demonstrates 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.





In somewhat more detail, its functionality includes creating a power circuit with selected elements, editing a power circuit, or adding or removing an element to or from a circuit. The elements should have unused electrical connectors with same voltage definition and pole numbers. You can also select a panel for a circuit, or disconnect a panel from a circuit if it already has one assigned.

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.

To run the sample, you can open or create a new Revit project and make sure the required electrical elements are placed. A sample project file PowerCircuit.rvt is provided in the sample folder.

To create a power circuit, select elements which all have unused electrical connectors with the same voltage definition and pole numbers in current project, execute the command and click the button. Expected result: A power circuit is created with the selected elements and highlighted. Note: 1. Currently the API provides no good way to check whether the connectors have the same voltage definition or pole numbers, therefore the sample skips validating this information and just displays a message if the creation fails. The user must know whether the selected element connectors' voltage and pole numbers match. 2. This sample does not support creation of a power circuit with elements which are all lighting devices.

To edit a power circuit, select a power circuit or an element belonging to one or more circuits, execute the command and click the button. If the selected element belongs to more than one power circuits, select one of them to edit in the 'Select a Circuit' dialog. Then choose an option to edit the circuit in the 'Edit Circuit' dialog: there are four buttons to add an element, remove an element, select a panel, and disconnect a panel.

Click the Add button to add an element to the circuit. Then select an element which has unused electrical connectors within same voltage definition and pole numbers as the other circuit elements. The sample skips validating the voltage and pole numbers and provides message if creation fails. Expected result: The selected element is added to the circuit and the circuit is highlighted.

Click the Remove button to remove an element from the circuit. Then select the element to remove. Expected result: The selected element is removed from the circuit and the circuit is highlighted.

Click the Select Panel button to select a panel for the circuit. If the selected element belongs to more than one power circuits, select a circuit from its circuits to edit in the 'Select a Circuit' dialog. Then select a panel to assign it to the circuit. The panel must be assigned to distribution system with the same voltage definition as the other circuit elements. Also, the panel must have the same pole numbers as the other circuit elements. Here again, the sample does not validate the voltage and pole numbers and provides a message if the creation fails. Expected result: The selected panel is assigned to the circuit and the circuit is highlighted.

To disconnect the panel from a circuit, select a power circuit or an element belonging to one or more circuits, execute the command, and click appropriate button. If the selected element belongs to more than one power circuits, select a circuit from its circuits to edit in the 'Select a Circuit' dialog. Expected result: The panel of the circuit is removed from the circuit and the circuit is highlighted.

## HVAC Sample

We now discuss our two custom samples. The first is for HVAC air terminal analysis and sizing. Before looking at the HVAC sample itself, let us consider the workflow it supports.

### Engineering Workflow

Some of the main tasks of an HVAC engineer are

* Placement and sizing of air distribution ducts and terminals
* Analysis and verification of results

The more detailed HVAC engineering workflow might look like this:

* Manually lay out air terminals
* Run heating and cooling load analysis
* Based on the load analysis, assign flow to terminals
* Based on the terminal flow, size air terminals
* Validate design

In HVAC design, a common check figure to validate design is the flow density, or air flow per floor area, i.e. the total amount of supply air divided by the size of the room. In the US, this is typically measured in cubic feet per minute per square foot, or CFM / SF.

There are common rules of thumb based on the room type on the result expected from such a calculation. Anomalies can be quickly identified, both in a schedule, as well as in a colour fill. A schedule can calculate the CFM/SF, but this computation is not available to a colour fill, thus the reason for this example application.

#### Room Supply Airflow

Revit MEP maintains several different values related to the air flow through a room:

* Calculated Calculated by external application
* Specified Assigned by design engineer
* Actual Tabulated based on supply air terminals in the room

The calculated value can be set by using IES heating and cooling loads calculation, or importing from gbXML. After the calculation, the ‘specified’ value is set equal to the ‘calculated’ one. The engineer can then adjust manually if necessary.

#### Heating and Cooling Load Calculation

Revit MEP includes the IES Virtual Environment as a built-in module. It also supports export to IES VE, and import and export to other calculation applications via gbXML, the Green Building XML format.

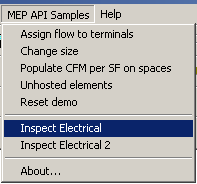
### Sample Implementation

The RME HVAC sample application implements six commands:

* Assign flow to terminals
* Change air terminal size
* Verify design by air flow per surface area
* Find unhosted elements
* Reset demo
* About

The first three commands support the actual HVAC engineering workflow, the fourth is independent, the fifth cleans up and resets the data manipulated by the first three, and the last simply presents the application description and version number.

These commands can be added directly to the Revit Tools > External Tools menu by listing them individually in Revit.ini. The application also implements an external application interface. If you add an entry for that to Revit.ini instead, it adds a new top level menu to the Revit menu, which packages all six commands together. It also includes the commands for the electrical sample:



Information on how to add new external commands and applications to Revit.ini is provided in the Revit SDK “Getting Started Revit API 2009.doc”, and in the other introductory material discussed in <http://thebuildingcoder.typepad.com/blog/2008/08/getting-started.html>.

The assembly name is mep.dll, and it defines the external commands and application in the namespace mep. The external application implementation class name is App.

This HVAC sample was originally implemented for RME 2008. In that version, it was based on Room elements. It was rewritten for 2009 using Space instead. To run it in Revit MEP 2009, be sure to use the updated version of RevitAPI.dll provided in the Revit MEP 2009 web update 2, since previous versions do not provide full API access to the spaces, especially the **FamilyInstance.Space** property.

One interesting thing to note about this application is the following: although it indirectly manipulates the MEP duct system, it only makes use of generic Revit API functionality to manipulate parameters and type of the duct system elements. Therefore, it can be run in Revit Architecture as well as in Revit MEP. Interestingly, it runs significantly faster in the Architecture flavour of Revit than in the MEP flavour, because there is no MEP machinery running in the background trying to clean up and recalculate the duct system every time the application updates some parameter value.

Some implementation details are interesting to mention from a programming point of view: we perform some manipulation of paths and command names in the external application method **App.AddMenu()**, called from the **OnStartup()** method, to determine the names and path of the external command class implementations when adding them to the menu. We also implement a useful generic about box for the application, discussed below.

### Assign Flow to Terminals

The first command in the sequence assigns flow to terminals. Before it is executed, we assume that the air terminal layout has been completed, so as a starting point all air terminals have been arranged in the building model for a uniform air distribution. The application then automatically assigns the appropriate flow to the terminals by performing the following steps:

* Calculate required air flow in each space
* Determine air terminals for each space
* Assign the flow to the terminals

The required air flow in each space can be calculated using an analysis tool, which populates the calculated supply airflow on each space.

To determine the air terminals for each space, the command can proceed as follows: the spaces do not maintain a list of the terminals they contain, but the terminals do know which space they live in. Therefore, we can determine this child-parent relationship and invert it to obtain the parent-child one. This is similar to the algorithm for inverting the relationship between the wall host elements and the doors and windows hosted by them described in the post on “[Relationship Inverter](http://thebuildingcoder.typepad.com/blog/2008/10/relationship-in.html)” in <http://thebuildingcoder.typepad.com>. The implementation is simple:

* Define a dictionary mapping each space number to list of air terminals contained by that space
* Select all air terminals in the model
* For each terminal, determine its space number and add it to the dictionary

To assign flow to its terminals, we can iterate over the spaces in the model, or just the spaces with an entry in the dictionary, determine the required air flow of that space, divide it by the number of terminals in the space, and distribute the space's required air flow evenly across all contained terminals.

Both steps can make use of Revit 2009 filtering, to select all air terminals and all spaces in the model respectively. In the code, we have left the 2008 implementation of the element selection in place in a separate method GetSupplyAirTerminals2008(), so you can compare that with the new method GetSupplyAirTerminals2009() making use of the filtering feature.

Another issue that requires some attention here is the unit conversion. In Revit, the required air flow in each space is stored as cubic feet per second, whereas in the US, the convention is to display the value in CFM, cubic feet per minute. Also, per convention, it is rounded up to nearest multiple of 5 CFM.

So, to summarise some generic aspects, this sample demonstrates how to find instances of a particular family category in a room or space and how to assign a value to a family instance parameter.

### Size Air Terminals

Once the required air flow has been determined for each individual air terminal in the first step, the air terminals can be sized appropriately. Air terminals are commonly sized based on a table of air flows. The greater the flow required, the larger the neck size of the air terminal. Selecting a different neck size for an air terminal family instance means determining which symbol to use from the air terminal family. Within the family, each air terminal symbol has been assigned parameter values specifying the maximum and minimum air flow it is suited for.

A completely different approach would be to implement a much more complex family definition making use of a set of nested if-then statements which switch its neck sizes depending on the flow assigned, without a need to select a different symbol. One additional problem with such an approach is that to manually override the selection for any reason would require additional complexity in the family definition.

This sample demonstrates the first approach, interrogating the min and max flow characteristics of each type, and then assigning the proper size based on the required flow. In this case, overriding the selection is a simple matter of selecting an alternate type.

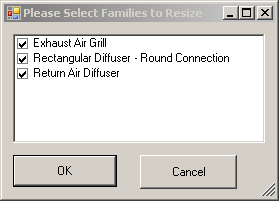
First of all the application sets up a table mapping the various ranges of air flow to specific air terminal type to use for that range. It does so by iterating over selected air terminal families. These are identified within the Revit database by their category, which is BuiltInCategory.OST\_DuctTerminal. All such families are identified and presented to the user to select which of them to process. For each family, a list of all its types specifying its min and max flow is constructed.

Once the families have been selected and their lists of flow ranges with associated types are set up, we can iterate over all air terminal family instances. For each terminal, we retrieve its flow and find and assign a symbol from the list whose range matches the required flow.

This sample thus demonstrates how to swap a family type based on user input, the flow, and characteristics of the type, the min and max flow. To summarise, the following steps are executed:

* Retrieve all air terminal families.
* User selection of families to process.
* Set up list of symbols with min to max flow ranges.
* Retrieve all supply air terminal instances.

For each terminal, set the appropriate type so that its required flow falls within the symbol’s min/ and max flow range.



From the Revit product side, a schedule can be set up as a powerful tool to simplify checking whether all terminals indeed have been assigned the correct type. By comparing the required air flow with the min and max range assigned to the type, invalid values falling outside the type range can be automatically highlighted.

Once the first two steps have been executed, the design is basically complete: the air terminals have been assigned a suitable required flow based on the space requirements, and then assigned a suitable symbol based on that required flow. The next step serves to validate the design.

### Flow Density Design Validation

As explained above in the section on the engineering workflow, the HVAC duct design can be validated by checking the flow density, i.e. the total amount of supply air divided by the size of the room, commonly measured in CFM/SF in the US. We also mentioned that a schedule can calculate the flow density, but this computation is not available to a colour fill, thus the reason for this example.

To calculate the flow density, we again retrieve all space elements. For each space, we retrieve its area and actual supply air flow parameter data. From this, we can calculate the flow per area and set a custom parameter. We use a project parameter named "CFM per SF"

In 2009, Revit MEP predefines a parameter named "Calculated Supply Airflow per area", with a corresponding built-in parameter enumeration value named ROOM\_CALCULATED\_SUPPLY\_AIRFLOW\_PER\_AREA\_PARAM. This is based on the heating and cooling load analysis computation. This sample demonstrates the computation and colour fill of the "Actual Supply Airflow per Area", which is representative of the flow density based on the *actual* flow assigned to air terminals in the space; such a built-in-parameter does not presently exist in Revit MEP.

From the user interface, we can define an area fill based on the CFM per SF parameter value, giving us the desired graphical flow density colour fill display.

This command demonstrates how to assign a calculated value parameter to a built-in family, in this case, a room or space.

### Find Unhosted Elements

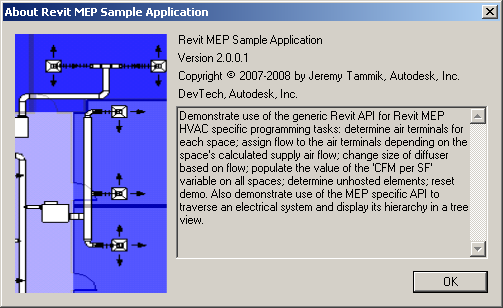
The fourth command provided in this sample demonstrates how to interrogate the model to find unhosted elements. Such functionality is beneficial to MEP designers to find elements that no longer are associated with architectural elements. For example, if a hosted element such as a receptacle, air terminal, plumbing fixture or sprinkler is placed on a wall, and the wall is later shortened or deleted, the receptacle may be orphaned, i.e. unhosted. This command helps the designer detect such situations so they may be addressed.

To find all unhosted elements, we search for all family instances with a built-in parameter INSTANCE\_FREE\_HOST\_PARAM whose value is "<not associated>" or "None", to signify unhosted. The search for these elements is again implemented in two different versions, iterating over all database elements and checking the desired parameter one by one in DetermineUnhostedElements2008(), and using the more efficient 2009-style element filtering in DetermineUnhostedElements2009().

The resulting element ids of unhosted elements are displayed in a dialogue and can be copied to the clipboard for further analysis, for instance to highlight and zoom in to them using the Revit menu entry Tools > Element Ids > Select by ID.

### About and Progress

The implementation of the About box deserves a short mention as well. It is implemented as a separate external command, so it can be invoked from its own menu entry. It demonstrates how you can maintain all the application data such as name, title, description, company, version number and copyright in the .NET assembly information specified in AssemblyInfo.cs, and suck that out from the executing assembly itself to reuse it in the About box as well. This approach is completely generic and can be reused in other applications as well:



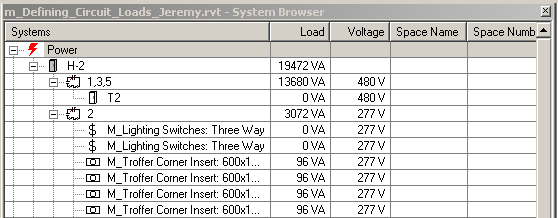
Finally, the sample implements an own little progress bar:



It is only important when running the sample in Revit MEP. In RAC, the processing is too fast for the progress bar to be seen.

## Electrical Sample

The purpose of this MEP electrical sample is to determine and display the hierarchical structure of the electrical systems in the model. The reason why this is slightly challenging is that the Revit MEP API does not provide a top down access for traversing the electrical system elements. Instead, we have to collect all the candidate leaf node elements ourselves, and for each child element, determine its parent element from its properties and parameter values. The method to select the relevant elements and to determine the parent node varies for different element types. Once we have determined the relationship from child to parent, we can add it to a dictionary with a key entry for each parent associated with a value listing all of its children. This is again similar to the algorithm used in the HVAC application described above for inverting the relationship between the spaces of a building and the air terminals located within each of them, or the wall host elements and the doors and windows hosted by them described in the post on “[Relationship Inverter](http://thebuildingcoder.typepad.com/blog/2008/10/relationship-in.html)” in <http://thebuildingcoder.typepad.com>.



The standard RME system browser displays electrical components in a three-level flat list, and the complete hierarchical structure of the connection tree is not immediately apparent.

In this sample, we inspect the electrical system, and explore how to reproduce the structure and information displayed by the system browser as well as the full connection hierarchy. To retrieve the Revit elements associated with these nodes, we obviously make use of the Revit 2009 API filters. We have two different sources of data:

* Electrical equipment
* Circuit elements

Electrical equipment always has a system name and is either unassigned or has a panel name. To retrieve these elements, we can filter for the ElectricalEquipment type.

Circuit elements have a non-empty circuit number parameter value, so we can retrieve those by filtering for RBS\_ELEC\_CIRCUIT\_NUMBER.

It is possible to determine the hierarchical connection structure between the elements from properties and parameters. These values are present on the child elements and can be used to deduce the respective parent element. There are no such properties immediately available to determine the inverse relationship.

We sort the selected elements into dictionaries mapping each parent element, a panel or circuit = system, to a list of connected elements, which can be circuit elements or nested lower level systems. After that, the dictionaries can be merged into an appropriate data structure.

We implement three commands:

* CmdElectricalSystemBrowser
* CmdElectricalHierarchy
* CmdElectricalHierarchy2

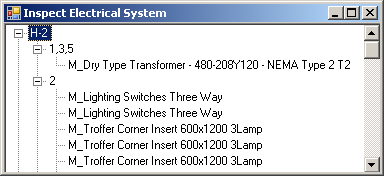
All of them implement the same basic approach:

* Collect the relevant Revit elements.
* Determine their relationships from child to parent by examining child element properties and parameter values.
* Sort this data into a dictionary.
* Use the dictionary structure to determine the inverted relationship from parent to children.
* Display the hierarchical structure in a tree view in a modeless form.
* Display additional nodes listing the electrical loads.

The tree view is displayed in a modeless dialogue, so it remains visible and can be navigated after the command has terminated. You can leave the form open and switch back and forth between it and Revit to explore the electrical system simultaneously from both points of view. Note, however, that updates to the model won’t be reflected in the dialogue until the command is re-executed.

### Electrical System Browser

CmdElectricalSystemBrowser was used for the initial exploration. In the first stage of implementation, we explored how to retrieve the information displayed in the Power section of the RME System Browser and replicate its three-level node structure. Later, the same information is used in CmdElectricalHierarchy to display the full tree view representing the entire connection hierarchy. The only difference between the two implementations, is the choice whether to call the PopulateLikeSystemBrowser() or PopulateFullHierarchy() helper methods in the CmdInspectElectricalForm constructor.



The initial challenge here was to reproduce all the system browser nodes, and the exact sorting order. The system browser has three levels of nodes, panel > system > element, as well as one 'Unassigned' top level node on the same level as the panel nodes. The system entry may also contain a circuit number.

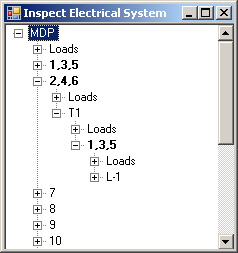
The elements used to establish the entire connection hierarchy are the electrical equipment and the circuit elements described above. Each of these two sets is sorted into dictionaries using one of the following the keys:

* Panel:circuit number
* Panel:system name
* Unassigned:system name

Each such key in the dictionary is mapped to a list of child nodes that the corresponding parent element is connected to. These two dictionaries, merged together and appropriately sorted, represent the same structure as the System Browser.

### Connection Hierarchy

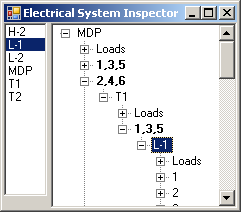
To display the entire connection hierarchy instead of the three-level structure shown in the System Browser, we can use the exact same information, and simply sort the items into the tree differently, making use of a recursive algorithm. Due to the design of the initial dictionary structure, the algorithm used in PopulateLikeSystemBrowser() is significantly simpler than the recursive multi-tiered PopulateFullHierarchy() one.



The CmdElectricalSystemBrowser command was implemented in parallel with the exploration of how the system is actually hooked up, and CmdElectricalHierarchy makes use of the same data collection code. Once that is fully understood, a more straightforward and efficient algorithm for traversing and displaying the system can be implemented. This code also includes some experimental sections that have been commented out but are useful for further exploration. Please don't simply reuse this code, understand it first and then rewrite it. This led to the second implementation described next.

### Connection Hierarchy 2

CmdElectricalHierarchy2 displays the same tree view as CmdElectricalHierarchy, but it uses slightly different methods to identify the relationships. In addition, it also displays a list of systems on the left hand side. Selecting one of the systems on the left side automatically collapses and reopens the tree on the right hand side to display the nodes belonging to that system, with no need to manually navigate to them through the tree.



This implementation directly assembles the tree hierarchy from the element relationships, mapping parent element to child nodes using a dictionary-derived class MapParentToChildren. It also uses element ids wherever possible, instead of key strings of the form "panel name : circuit or system name". The dictionary directly maps each parent element id to a list of child element instances. For the different levels of the tree, it maps

* Null > root panels
* Panel > systems
* System > circuit elements, lower-level panels, etc.

We also define a helper dictionary mapPanel which contains all the electrical equipment and panels and maps their name to their family instance. The electrical equipment is identified by the built-in category OST\_ElectricalEquipment.

The following steps performed by the command determine all system elements, add them to the appropriate place in the map:

* Retrieve panels with GetElectricalEquipment(), add to mapPanel and main map
* Retrieve ElectricalSystem elements, determine parent panels and add to map
* Iterate panels, determine parent systems or root node and add
* Retrieve circuit elements, determine parent circuit and add

To display the complete connection hierarchy in the tree view, we populate it using recursion. During the addition, the equipment nodes are identified for highlighting, and also for adding them to list box for the quick navigation support.

## Appendix

### 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>

### Acronyms

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