

Revit MEP Programming: All Systems Go

Jeremy Tammik – Autodesk

CP4108 This class discusses working programmatically with Revit MEP models, including an overview of the entire Revit MEP API and specific focus on the major enhancements in Revit 2013. The most important 2013 feature is the routing preference functionality. An overview of the available samples is provided. All MEP domains including HVAC, electrical and plumbing are addressed. We show how to analyse existing systems and create new MEP models from scratch. We cover mechanical and electrical system traversal, display of system hierarchies in a tree view, components such as circuits, ducts, pipes, fittings, connectors, cable trays, and conduits, and automatic calculation and sizing based on room and space requirements. Please note that prior .NET programming and Revit MEP product experience is required and that this class is not suitable for beginners.

Learning Objectives

At the end of this class, you will

- Understand the major enhancements of the MEP API in Revit 2013
- Understand the use of the new routing preferences API
- Manage, analyse, create and modify HVAC, electrical and plumbing Revit MEP models, systems, and components programmatically
- Understand and reuse MEP functionality provided by the Revit SDK and ADN samples

About the Speaker

Jeremy is a member of the AEC workgroup of the Autodesk Developer Network ADN team, providing developer support, training, conference presentations, and blogging on the Revit API.

He joined Autodesk in 1988 as the technology evangelist responsible for European developer support to lecture, consult, and support AutoCAD application developers in Europe, the United States, Australia, and Africa. He was a co-founder 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 in mathematics and physics in Germany, worked as a teacher and translator, then as a C++ programmer on early GUI and multitasking projects. He is fluent in six European languages, vegetarian, has four kids, plays the flute, likes reading, travelling, theatre improvisation, carpentry, and loves mountains, oceans, sports, and especially climbing.

jeremy.tammik@eur.autodesk.com

Learning Objectives	1
About the Speaker	1
Introduction	3
MEP Application Requirements	3
The Generic Revit API	3
Revit MEP API Evolution	3
Analysis	4
EnergyDataSettings and gbXML	4
Conceptual Energy Analysis	4
Detailed Energy Analysis Model	4
Spaces and Zones	4
Model Inspection Utilities	5
Hierarchical Systems and Connectors	5
Hierarchical System Structure	5
MEP Model	6
Connectors	6
Connector Definition Elements	6
Electrical	7
HVAC and Plumbing	8
HVAC and Pipe Systems	8
Duct and Pipes	8
Fittings	8
Connectors	8
Element Creation	9
The Revit MEP 2013 API	9
Routing Preferences	9
Flow Analysis Sections	9
Fluid Viscosity and Density	10
Thermal Properties	10
More Revit MEP API News	10
Sample Applications	10
AddSpaceAndZone	11
AutoRoute	11
AvoidObstruction	12
CreateAirHandler	12
EnergyAnalysisModel	13
PanelSchedule	13
PowerCircuit	14
RoutingPreferenceTools	14
TraverseSystem	15
AdnRme Sample	15
MEP Placeholder Sample	17
Further Reading	17

Introduction

MEP Application Requirements

Revit MEP is a flavour of Revit for work in the mechanical, electrical and plumbing domains. HVAC, i.e. heating, ventilation and air conditioning, is considered part of the mechanical domain. Efficient work in these areas requires strong model analysis tools in addition to read, write and creation access to the components and data used to model the electrical, HVAC and piping systems in Revit MEP:

- Model Analysis Tools
 - Physical, thermal, environmental etc.
 - Building codes and regulations
 - Geometrical relationships
- MEP project information
 - Green Building XML, gbXML
 - Spaces and zones
- BIM access
 - Systems, components, properties and parameters
 - Creation and modification
 - Traversal and analysis

The Generic Revit API

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 .NET assemblies named RevitAPI.dll and RevitAPIUI.dll, respectively providing database and user interface related API access to third-party applications. However, some specific additional features exist for each of the flavours as well, for instance room-related functionality in Revit Architecture and access to the analytical and MEP models in Revit Structure and MEP respectively.

- Basic Revit API is generic
- All flavours use the same .NET assembly RevitAPI.dll
- Specific additional features exist for each flavour

If API functionality not supported by the currently running flavour is accessed, the call simply returns null.

This presentation assumes some basic understanding of the Revit MEP product and the generic Revit API and addresses the API additions and enhancements specific to Revit MEP.

Revit MEP API Evolution

API support specific to the MEP domain is defined in three namespaces within the generic Autodesk.Revit.DB namespace:

- Autodesk.Revit.DB.Electrical
- Autodesk.Revit.DB.Mechanical
- Autodesk.Revit.DB.Plumbing

Even without using the MEP-specific API, it is still possible to analyse and manipulate an HVAC system using generic Revit API methods, for instance by accessing ducts as generic Revit elements and reading and writing their properties using the generic parameter access. This method is demonstrated by the

AdnRme sample application for HVAC air terminal analysis and sizing, first presented way back at at Autodesk University 2007.

The first MEP-specific API support included classes for electrical and mechanical equipment, lighting device and fixture, connector, electrical system, space and zone, the MEP model property on family instances, and the ability to create and modify MEP elements (Revit 2009).

Later, the MEP API was a focal point of the API development and the support for HVAC and piping systems enhanced (Revit 2010).

Revit MEP 2011 introduced cable trays, conduit elements, electrical panel schedules, and consolidated most of the MEP related API classes into the new more specific namespaces.

Revit 2012 API enhancements include placeholder elements, new insulation and lining functionality, and access to pipe settings and sizes.

The Revit MEP 2013 API include support for routing preferences, analysis and calculation enhancements.

Analysis

Extensive and detailed model analysis tools are required for efficient modelling in all MEP disciplines. Some of the important features include the MEP project information, support for gbXML, the Green Building XML, spaces and zones for MEP-specific building area definition and management, and various model inspection utility methods.

EnergyDataSettings and gbXML

Global Revit MEP project information and settings are managed through the EnergyDataSettings element and its properties. The settings influence the gbXML export, heating and cooling load calculations, and conceptual energy analysis.

Green Building XML export is initiated by a call to the Document.Export method:

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

Conceptual Energy Analysis

Conceptual energy analysis deals with energy analysis on conceptual design models. Calling the Document.Export method with a MassGBXMLExportOptions argument exports a gbXML file containing conceptual energy analysis elements (mass elements) only.

Detailed Energy Analysis Model

The detailed energy analysis model API enables you to obtain and analyze the contents of a project's detailed energy analysis model, as seen in the Export to gbXML and the Heating and Cooling Loads features. This analysis produces an analytical thermal model from the physical model of a building. The analytical thermal model is composed of spaces, zones and planar surfaces that represent the actual volumetric elements of the building. A new model is created and initialised using the static EnergyAnalysisDetailModel.Create method. The EnergyAnalysisModel SDK sample demonstrates the use of this API.

Spaces and Zones

Architectural rooms are unsuitable for MEP analysis, because they often have the wrong height or are too large for the analysed region. For MEP purposes, Revit uses spaces instead of rooms, and zones to group spaces. For example, this enables a subdivision of a room into exterior and interior subspaces. The

AddSpaceAndZone SDK sample demonstrates the programmatic creation and management of spaces and zones.

The FamilyInstance class has Room and Space properties to determine in which room or space an instance is located. The latter can be used to determine the location of e.g. an air terminal in Revit MEP:

```
FamilyInstance fi; // get a family instance
Space space = fi.Space; // query the space in which it is located
Space space2 = fi.get_Space( phase ); // query space in a specific phase
```

Model Inspection Utilities

The Revit API provides a number of model inspection utilities which are especially useful in the MEP domain. One group helps determine the relationships between and contents of volumes, rooms and spaces:

- Room.IsPointInRoom determines if a 3D point is in the volume of a room.
- Space.IsPointInSpace determines if a 3D point is in the volume of a space.
- GetRoomAtPoint and GetSpaceAtPoint return the room or space containing a given 3D point.

One example use of these methods is to perform a space adjacency analysis.

Filtered element collectors can be used to find 3D elements by intersection with support for Boolean operations. Useful filter classes in this context include BoundingBoxIntersectsFilter, BoundingBoxIsInsideFilter, BoundingBoxContainsPointFilter, ElementIntersectsElementFilter, and ElementIntersectsSolidFilter.

The FindReferencesWithContextByDirection ray intersection method can be used to determine proximity of elements. It returns an array of elements, faces, and references found when moving through the model from a specified point in a specified direction.

The use of this method is demonstrated by the AvoidObstruction, FindColumns, MeasureHeight, and RayTraceBounce SDK samples.

Revit 2013 introduces the ReferenceIntersector class, which provides a simplified and optimized wrapper for this functionality.

Currently these utilities work within the context of the host model, i.e., you cannot use them to analyse elements in linked models.

Hierarchical Systems and Connectors

Many components of an MEP system are represented in Revit by family instances. The Revit API supports MEP-specific properties and data on the generic family instance definition, and defines classes to represent systems and connections between system components.

Hierarchical System Structure

The objects dealt with in the MEP domain are organised into hierarchical structures. The top level node in these structures is an MEP system, represented by the MEPSystem class. This is the base class for the derived classes

- ElectricalSystem
- MechanicalSystem
- PipingSystem

The system has properties to access the base equipment or electrical panel, terminal elements, and the connector manager. The derived classes provide additional domain specific properties.

MEP Model

The MEPModel and Connector classes are fundamental for programmatically representing and accessing the MEP model. MEPModel is a base class for the derived classes

- ElectricalEquipment
- LightingDevice
- LightingFixture
- MechanicalEquipment
- MechanicalFitting

On family instances, it can be accessed through the FamilyInstance.MEPModel property. One of its main purposes is to provide access to the connector manager. This property returns null in a non-MEP flavour of Revit.

Connectors

We have logical and physical connections between MEP components. Logical connections are used in the electrical domain, where wires are annotation elements not intended to represent the real-world model. Physical connections are used in all MEP domains. In the mechanical and plumbing domains, the connectors define and transmit the sizing information from one part to the next.

A physical connector is part of an MEP element such as a duct, pipe, fitting, or piece of equipment. The Connector class is used to represent connectors in the project environment.

The connector defines geometry, e.g. its oval, round or rectangular shape, and dimensions such as width, height and radius. It also maintains a coordinate system and system data such as domain, system type, flow, direction. You obviously cannot read the radius property if the connector shape is not round, so you need to check the shape first before reading some of the dimensions.

Connector Definition Elements

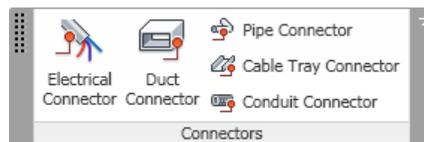
Individual stand-alone connector elements are created inside a family definition and are represented by the DuctConnector, PipeConnector and ElectricalConnector classes. These elements in the family definition are converted to Connectors when a new family instance is inserted into the project.

Elements in the project file, such as equipment, pipes, ducts and fittings, which form parts of connected systems and need to be connected to their neighbour elements achieve this using connectors which are not elements or family instances, but contained within the elements that they connect.

There are different types of connectors for the different types of systems. A single piece of MEP equipment can have none, some or all connector types.

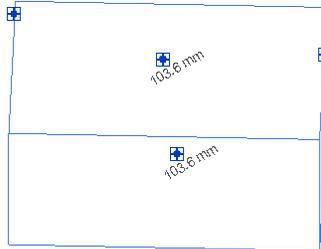
For instance, a piece of HVAC equipment such as a cooler may have electrical connections for power, pipes for cooling water in- and outlets, and ducts for air in- and outlets. These connectors are defined in the family editor. When the equipment is inserted into the model, the connectors are used to hook it up to neighbouring elements, in this case electrical systems, pipes and ducts.

Thus one set of classes defines connector elements residing in the family document. Another class represents connectors on the resulting family instances, once inserted into a project document.



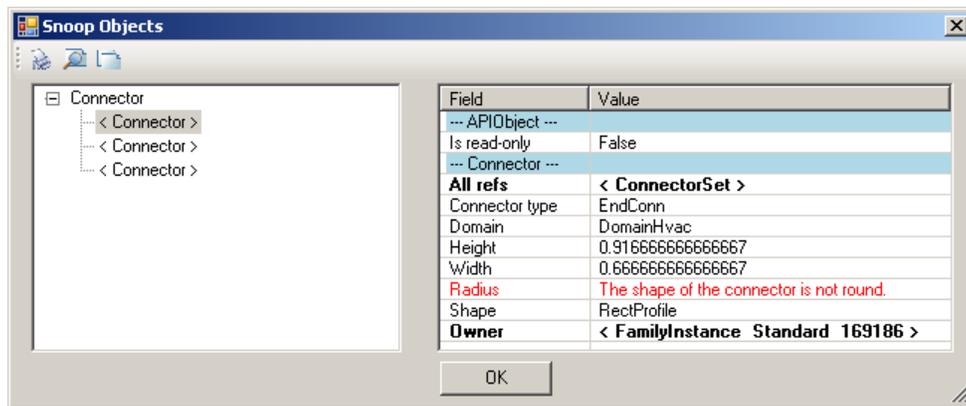
MEP family editor connector definition tools

Once you define a family type and insert it in a Revit project, you have a family instance, for instance this air handler:



Air handler family instance with connectors in a project file

This instance contains connectors, accessed via the connection manager and represented by the Connector class:



Duct Tee Cross connector class instances in RevitLookup

Electrical

Electrical systems are organised hierarchically in Revit MEP. The root elements are panels. Each panel is connected to a number of systems, also known as circuits. Each system or circuit consists of a number of circuit elements, some of which may be further panels, and so on, recursively.

Cables and wires are not always modelled individually, so the connections between the electrical system components may be defined by logical connectors rather than physical ones.

The classes ElectricalEquipment, LightingDevice, LightingFixture, ElectricalSystem, specialisations of the base MEPModel and MEPSysSystem classes, represent electrical components and systems. Their connector manager returns logical as well as physical type connectors via the Connectors property:

```
ElectricalSystem sys;
ConnectorSet connectors = sys.ConnectorManager.Connectors;
```

The connector manager can be used to retrieve the connectors and determine how the different components of an electrical system are attached to each other to construct the system hierarchy from that information, as demonstrated by the AdnRme electrical sample.

HVAC and Plumbing

The main components in HVAC and piping hierarchy are the top level systems, ducts, pipes, fittings and the connectors:

- Systems manage the top level system properties.
- Ducts and pipes define the main flow elements.
- Fittings implement bends and branches in the system.
- Connectors hook up ducts, pipes and fittings with each other.

The following sections present a top-down hierarchical introduction to these classes and their functionality.

HVAC and Pipe Systems

The MechanicalSystem and PipingSystem classes provide the following top-level system functionality:

- Access to equipment, connectors and system type.
- Access to system properties such as flow and static pressure.
- Analysis of system contents through the DuctNetwork or PipeNetwork property.

Elements contained in a system are not automatically returned in the flow direction. The TraverseSystem SDK sample demonstrates how to query the connector managers to implement traversal in direction of flow for a given system.

Duct and Pipes

Ducts and pipes are represented by the classes Duct, Pipe, FlexDuct and FlexPipe, derived from the MEP curve type. They provide the following functionality:

- Read access to duct properties, types, and geometry.
- Move or layout duct or pipe.
- Change duct or pipe type.

The layout functionality can be driven by two points, one point and a connector, or two connectors.

Fittings

The fittings are represented by family instances, which can be inserted in the model using dedicated New*Fitting methods on the Autodesk.Revit.Creation.Document class, where * can be replaced by Elbow, Tee, Cross, Takeoff, Transition, and Union. The fitting properties and their shape and dimensions can be accessed through the FamilyInstance.MEPModel property.

Connectors

The basic features of connectors and the different classes used to represent them in the family and project contexts were discussed above. Additional HVAC-specific connector features include properties to read the flow, coefficient and demand. They also provide access to physical properties like Origin, Angle, Height, Width and Radius.

We have read and write access to the assigned connector properties, connector properties that can be overridden by the user and are dependent on the overall network condition, including characteristics like Flow, Flow Config, Coefficient, Loss etc. These depend on the system being well connected, the configuration of the family within the system, and how the family is constructed.

Connector size and location can be modified. Connectors can be disconnected to insert a new fitting into a network or connect two ducts into a transition.

- The family connector elements define Flow, Flow Configuration, Coefficients, and Loss Method.

- Read duct, pipe, and fitting connector properties such as Flow, Coefficient, and Demand.
- Access physical connector properties e.g. Origin, Angle, Height, Width, Radius.
- Change connector size and location.
- Connect and disconnect.

Element Creation

The creation document provides a number of methods for the creation of HVAC and plumbing elements within the project environment:

- `NewMechanicalSystem` and `NewPipingSystem` to create a new system.
- `NewDuct`, `NewFlexDuct`, `NewPipe` and `NewFlexPipe` to create new elements.
- `New*Fitting` to create new fittings, * being one of `Cross`, `Elbow`, `TakeOff`, `TeeFitting`, `Transition`, `Union`.
- More recent classes such as `Conduit` and `CableTray` provide static `Create` methods instead of using the creation document.

As discussed above, in the family context, different types of connectors are represented by elements and created by the `FamilyItemFactory` class accessed through the `Document.FamilyCreate` property and its `NewDuctConnector`, `NewPipeConnector` and `NewElectricalConnector` creation methods.

The Revit MEP 2013 API

The information provided in this section is based on the Revit API help file `RevitAPI.chm` 'What's New' section. The help file is available as part of the Revit SDK from the Revit developer page (cf. Further Reading at the end).

One of the main Revit MEP 2013 product features is the support for routing preferences. Previously, you could only define one single default elbow, tee, etc. The routing preferences support multiple settings for different materials and sizes.

Routing Preferences

The routing preferences enable you to select your preferred pipe and duct fitting types for various materials and sizes. It can be used to set routing preference policies for end users and query fittings and segments used for given size criteria.

The `RoutingPreferenceManager` class is used to manage routing preference rules for segments and fittings or query the fitting or segment chosen by Revit for a given size condition. The `MEPCurveType` class now provides a `RoutingPreferenceManager` property to access the main routing preferences object for that type. This is currently only supported by the derived `PipeType` and `DuctType` classes.

Routing preferences choose the first symbol in a rule list matching the given criteria. To ensure that a certain symbol is chosen for a given scenario, you can either set appropriate size criteria, or temporarily re-order the rules using the `RemoveRule` and `AddRule` methods.

Programmatic use of routing preferences is demonstrated by `RoutingPreferenceTools` SDK sample discussed below.

Flow Analysis Sections

The `MEPSection` base class has been introduced to represent duct and pipe sections, i.e. a series of connected elements, mainly for pressure loss calculation. A section includes ducts or pipes, fittings, terminals and accessories, where all section members have same the flow analysis properties, e.g. Flow, Size, Velocity, Friction and Roughness. An element can belong to multiple sections. For example, a tee

fitting with three connectors usually belongs to three sections, and a tap will divide a duct or pipe segment into two sections.

Fluid Viscosity and Density

Revit MEP 2013 provides support for more precise temperature dependant friction calculation via the new FluidTemperature class, which can represent viscosity and density properties at a given temperature. The FluidType class has been extended to provide read and write access to a collection of FluidTemperature objects representing the fluid properties at various different temperatures. This collection can be accessed and manipulated using the new AddTemperature, GetTemperature, RemoveTemperature and GetFluidTemperatureSetIteratormethods.

Thermal Properties

Thermal properties have been enhanced in several areas. New properties are provided on the ThermalProperties class, e.g. Absorptance, HeatTransferCoefficient, Roughness, ThermalMass, ThermalResistance. These are available on BIM elements such as wall, floor, ceiling, roof, door, window, etc.

The new ThermalAsset class is provided, representing the properties of a material pertinent to energy analysis. It is specified using the PropertySetElement and SetMaterialAspectByPropertySet classes.

The gbXML export provides more precise control of thermal properties of BIM types. They can be set to use calculated values or pre-defined values from Constructions.xml using the MEPBuildingConstruction methods GetBuildingConstructionOverride and SetBuildingConstructionOverride. The EnergyDataSettings.IncludeThermalProperties flag determines whether thermal information from model assemblies and components is included in the gbXML export.

More Revit MEP API News

Other enhancements include read and write access to spare circuit values, retrieval of more of the localised user-visible display strings for enumeration values.

ConnectorProfileType and PartType enumeration changes, ConnectorElement changes and new static creation methods, more LabelUtils access to localised user-visible display strings, access to panel schedule spare circuit values, a light and light group API,

The new ReferenceIntersector class provides a simpler access to geometric ray-tracing analysis.

Finally, a new external services framework has been put into place. It wraps external service functionality, which can be used to enable encapsulation and replacement of various calculation and analysis workflows, providing a basis for future MEP calculations and structural code checking. It is already in place and fully functional, but not yet used, so no examples are yet provided.

Sample Applications

The MEP API is illustrated by a number of sample applications including those provided in the standard Revit SDK, non-SDK ones created by the ADN DevTech developer support team, and some from The Building Coder blog.

The Revit SDK includes the following MEP specific samples:

- AddSpaceAndZone: handling space and zone elements.
- AutoRoute: route a set of ducts and fittings between a source and two sinks, the air supply equipment and two air outlet terminals.
- AvoidObstruction: detect and resolve obstructions between ducts, pipes, and beams.
- CreateAirHandler: create a family representing an air handler and its connectors.

- EnergyAnalysisModel: create an EnergyAnalysisModel, query and display its face structure in a tree view.
- PanelSchedule: use of the electrical panel schedule API.
- PowerCircuit: manipulation of electrical power circuits.
- RoutingPreferenceTools:
- TraverseSystem: traverse a well-connected mechanical or piping system in the direction of flow.

The main non-SDK sample application is named AdnRme. It implements some HVAC oriented commands making use of the generic Revit API to analyse and manipulate duct elements for automatic air terminal sizing. Others implement an electrical sample for analysis and display of an electrical system in a tree view reproducing the Revit system browser structure or showing the full connection hierarchy tree:

- HVAC air terminal analysis and sizing.
- Electrical system connectivity analysis and hierarchical display.

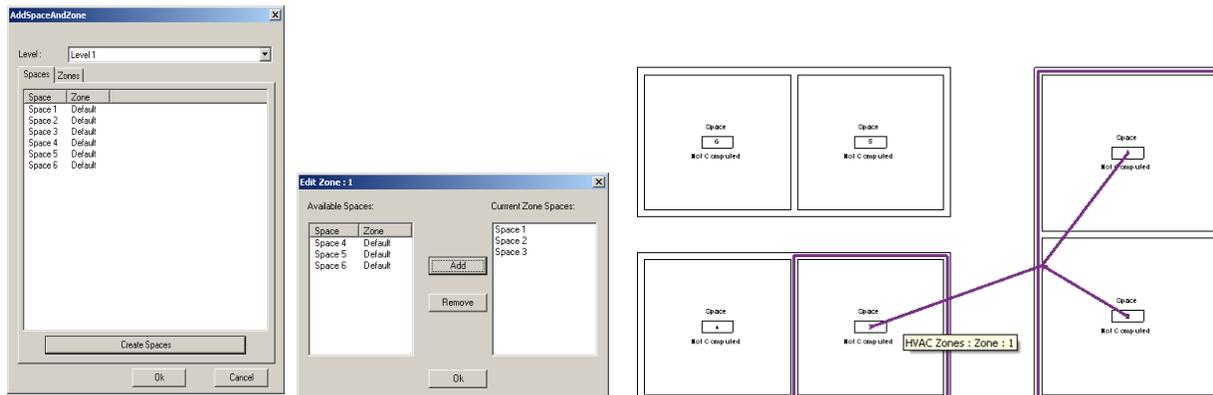
Finally, we already looked at the MepPlaceholders sample included in the class materials, demonstrating the new Revit MEP 2012 API features. It was also partially discussed by The Building Coder.

Here is a more detailed look at these samples:

AddSpaceAndZone

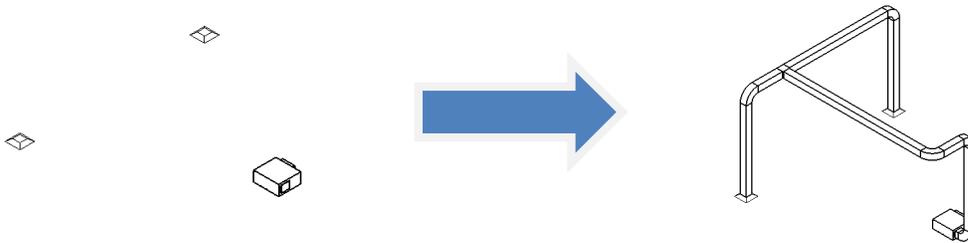
This 2009 sample creates the spaces and zones required for HVAC analysis, and implements the following main features:

- Retrieve and list all existing spaces and zones in a specified level using an element filter.
- Create new spaces for each closed wall loop or space separation with the NewSpaces method.
- Create a new zone element in a specified level and phase.
- Add and remove spaces in a zone using the AddSpaces and Remove methods.



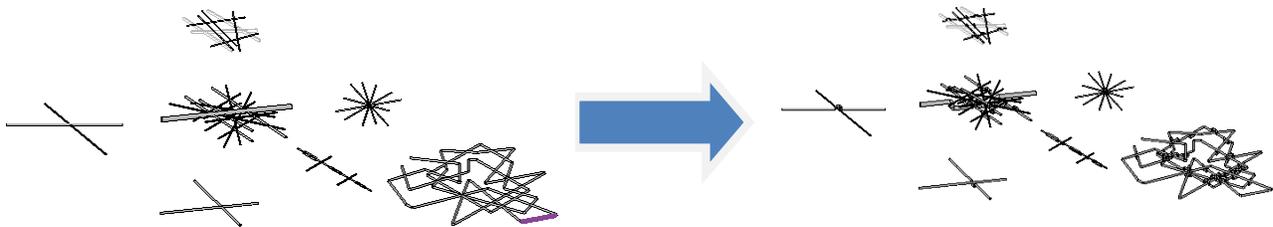
AutoRoute

Automatically create and route a set of ducts and fittings between a source, the air supply equipment, and the sink, two air outlet terminals. It creates a new mechanical system, ducts, fittings and connections using the methods NewMechanicalSystem, NewDuct, NewElbowFitting, NewTeeFitting and Connector.ConnectTo. The routing between the source and sink objects is automatically defined, so these can be moved freely prior to executing the command. To define the route, it first determines the bounding box of all the three elements, then uses the middle line or quarter lines on the X and Y axes as the trunk with a series of ducts that will be connected to the elements with fittings and ducts. This sample also demonstrates use of the .NET framework Trace class to create a log file AutoRoute.log.



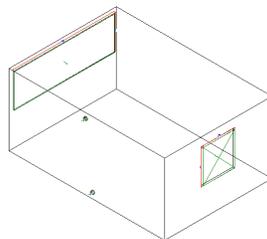
AvoidObstruction

Detect and resolve obstructions between ducts, pipes, and beams. This sample uses the ray casting intersection analysis method `FindReferencesWithContextByDirection`. Only the pipe centre line is used to detect obstructions, the radius is ignored. Only pipes are modified, none of the other elements. The pipe's original direction remains unchanged. It is split into several segments according to the obstructions detected, and elbows are inserted to reroute it around the obstruction. The example project included with the sample demonstrates the automatic resolution of collisions between pipes and beams, ducts, and other pipes.



CreateAirHandler

Use the family API in the family editor context to create an air handler and add connectors.



The sample first checks the family template type using the `OwnerFamily.FamilyCategory` property. It creates the geometrical shape of the air handler using the `NewExtrusion` method on the `FamilyItemFactory` class in the `Autodesk.Revit.Creation` namespace. The extrusions are united by the method `Document.CombineElements`. The methods `NewPipeConnector` and `NewDuctConnector` are used to create the connectors. Their parameters are defined by setting the duct properties.

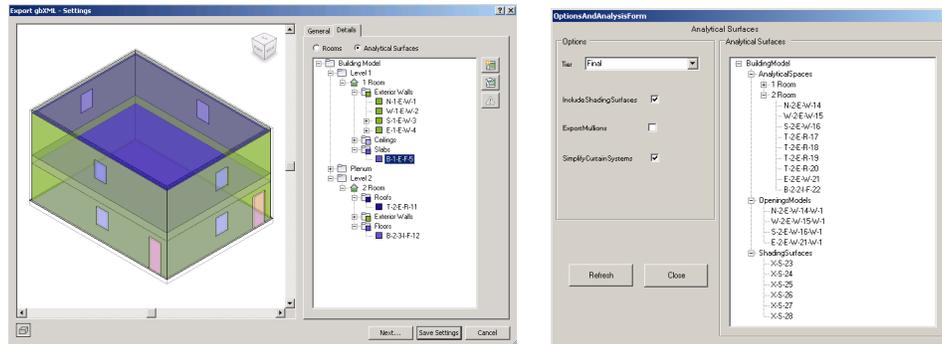
- Create an air handler with its geometry and its pipe and duct connectors
- Check family category to verify valid mechanical equipment starting point
- Use `FamilyItemFactory` class methods `NewExtrusion`, `NewPipeConnector`, `NewDuctConnector`
- Set connector parameters and use `Document.CombineElements` to join the extrusions

The geometric shape creation is obviously generic, and only the addition of the connectors is MEP specific. In spite of this, this sample runs in any flavour of Revit, which means that the creation of connectors in the family API is not restricted to the MEP flavour of Revit.

EnergyAnalysisModel

The new API provides access to the contents of a project's detailed energy analysis model, as generated by and seen in the Export to gbXML and the Heating and Cooling Loads features.

This analysis produces an analytical thermal model from the physical model of a building. The analytical thermal model is composed of spaces, zones and planar surfaces that represent the actual volumetric elements of the building.



New classes in Autodesk.Revit.DB.Analysis namespace for the energy analysis detail model itself, creation options, openings, spaces, surfaces and loops can be used to generate and analyze the contents of the detailed energy analysis model. The EnergyAnalysisDetailModel Create method creates and populates the model with appropriate options selected. The methods GetAnalyticalSpace, GetAnalyticalSurfaces, GetAnalyticalOpenings and GetAnalyticalShadingSurfaces extract entities from the analysis model. The method Destroy cleans up the Revit database after finishing with the analysis results.

- Produce analytical thermal model from physical building model
- Retrieve energy analysis detail model and present as tree view
- Similar to Export to gbXML and Heating and Cooling Loads
- Model is composed of spaces, zones and planar surfaces
- EnergyAnalysisDetailModel.Create()
- GetAnalyticalSpaces, GetAnalyticalSurfaces, GetAnalyticalOpenings, GetAnalyticalShadingSurfaces

The EnergyAnalysisModel SDK sample in the RvtSamples Analysis group generates a detailed energy analysis model, queries it for its surfaces depending on user-specified options, and lists them in a tree view. A sample model EnergyAnalysisModel.rvt is provided in the Samples > GeometryAPI > EnergyAnalysisModel folder.

PanelSchedule

The Revit 2011 SDK includes the MEP specific sample application PanelSchedule. It defines three external commands to demonstrate the use of the new panel schedule API:

- PanelScheduleExport – retrieve the panel schedule view data and export to CSV or HTML.
- InstanceViewCreation – create a panel schedule view instance for a selected electrical panel.
- SheetImport – place panel schedule views on a sheet view.

Autodesk Revit MEP 2011 - Microsoft Internet Explorer provided by Autodesk EIS

C:\Program Files\Autodesk\Revit\Autodesk Revit MEP 2011\PanelSchedule\PanelScheduleEP-2.html

Branch Panel:	EP-2	Volts:	120/208 Wye	A.I.C. Rating:	
Location:	Doctor 68	Phases:	3	Mains Type:	
Supply From:	PP-2B	Wires:	4	Mains Rating:	100 A
Mounting:	Surface			MCB Rating:	
Enclosure:	Type 1				

Notes:

CKT	Circuit Description	Trip	Poles	A	B
1	HVAC Corridor 56 30 A		3	1761 VA	2260
3	--	--	--	--	--
5	--	--	--	--	--
7	HVAC Corridor 56 30 A		3	1761 VA	2260
9	--	--	--	--	--
11	--	--	--	--	--
13	HVAC Corridor 57 30 A		3	2260 VA	1761

Branch Panel: PP-3B

Location: PP-3A
 Supply From: PP-3A
 Mounting: Surface
 Enclosure: Type 1

Volts: 120/208 Wye
 Phases: 3
 Wires: 4

Notes:

CKT	Circuit Description	Trip	Poles	A	B
1	Power Hall 84	20 A	1	1440	1080
3	Power Hall 82	20 A	1		1440
5	Power Hall 87	20 A	1		
7	Power Physics 89	20 A	1	1440	1440
9	Power Auditorium 91	20 A	1		1440
11	Power WC 76	20 A	1		
13	Power Hall 81	20 A	1	1440	1440

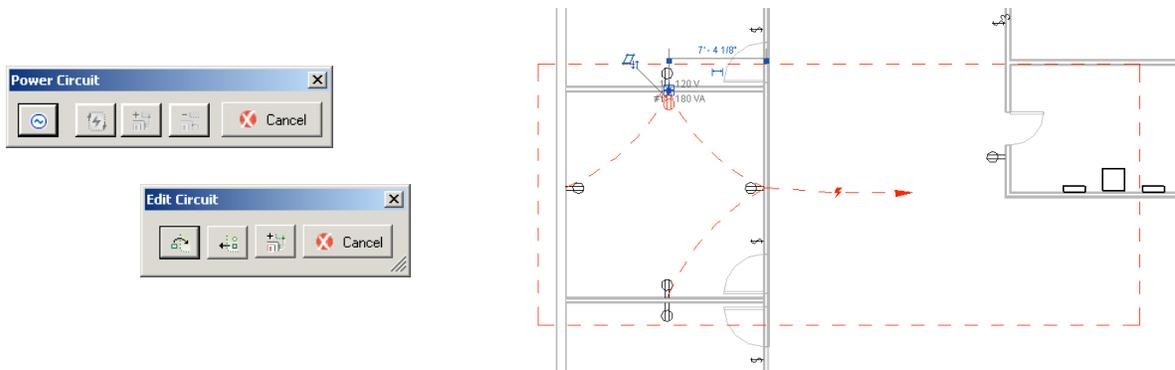
PowerCircuit

This RME electrical sample provides functionality similar to the legacy RME Circuit Editor toolbar. It shows how to operate power circuits, as well as how to handle interactive element selection in Revit, implement a toolbar user interface for an external command, and use the .NET ResourceManager class to manage the images and localisable string resources. It mainly exercises classes from the Autodesk.Revit.MEP namespace and the MEPModel and ElectricalSystem classes. 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.
- 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 its connector manager and electrical systems. The elements to be added to a circuit should have unused electrical connectors with same voltage definition and pole numbers. The entire user interface is presented by two toolbars with buttons representing the operations listed above.



RoutingPreferenceTools

- Routing preference analysis and reporting
 - Analyse routing preferences of a given pipe type
 - Look at all rules and criteria for a given pipe type
 - Check for common problems
- Routing preference builder XML import and export

- CommandReadPreferences and CommandWritePreferences
- Set pipe type, fitting, and routing preferences in a project
- Export for archival, documentation, and collaboration purposes
- Enable users to work with RP data in a shareable XML format
- Suitable for reuse in a wide variety of BIM management environments

TraverseSystem

Traverse a well-connected mechanical or piping system in the direction of flow and dump the traversal into an XML file. The MEP electrical sample presented below demonstrates a similar traversal of electrical systems.

The TraverseSystem sample defines two classes `TreeNode` and `TraversalTree`. A `TreeNode` object represents an element in the system and contains traversal connection clues. The class `TraversalTree` traverses the specified system in the direction of flow by generating a tree-like structure of `TreeNode` objects.

A connected mechanical or piping system is retrieved from a selected element which can be the system itself or any of its elements. If the selected element belongs to more than one well-connected system, the system with the most elements will be chosen. The traversal begins at the system base equipment, if found. If the system has no base equipment, it will have an open connector which is the only point that can connect to a base equipment. The owner of the open connector will be chosen as the starting point. The connected elements relationships in the system are analysed by their connectors' flow directions and generate the traversal tree-like structure.

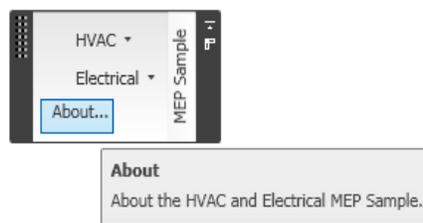
How is the system retrieved? If the selected element is the system itself, it is either a `MechanicalSystem` or `PipingSystem` instance, both derived from the `MEPSystem` class. Otherwise, it may be a fitting, such as a T join or elbow, represented by a family instance, or a duct or pipe, represented by a `MEPCurve` instance. Both fittings and MEP curves have an associated connector manager,

The well-connectedness is reported by the `MechanicalSystem` `IsWellConnected` property, the number of elements in a system by `Elements.Size`, and the base equipment through `BaseEquipment`. If base equipment is null, an open connector is searched for by starting at an arbitrary element and recursively iterating through all its connectors and neighbouring system elements. Once the starting element has been defined, the traversal tree is built by a second recursive iteration through all connectors, neighbouring elements, their connectors, etc.

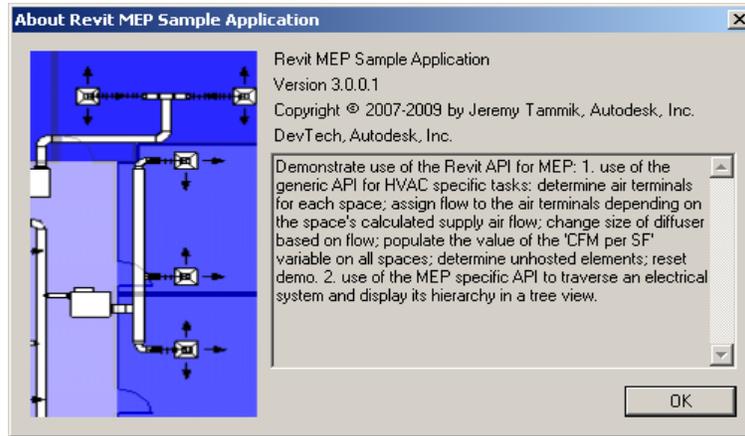
AdnRme Sample

This non-SDK sample implements a number of external commands addressing HVAC air terminal analysis and sizing as well as the hierarchical display of an electrical system. Some aspects of the sample not directly related to the MEP API are the external application implementation, about box, and progress bar.

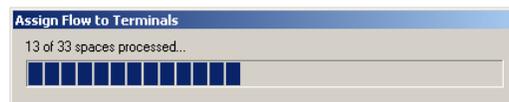
The HVAC part of the MEP sample was originally implemented based on Revit 2008. Later the electrical part and an external application to define a custom user interface to access the sample commands was added:



The application includes an about box command presenting the application description and version number. The version number and description displayed in the form is queried from the executing .NET assembly:



The HVAC sample implements a progress bar, since the processing of a large number of elements may take a noticeable amount of time:



The application resides in the assembly AdnRme.dll and defines the namespace AdnRme. Its external commands can either be accessed by installing the external application interface defining the custom ribbon panel, or by setting up command buttons in the Revit ribbon to access them individually.

The main aspects of the electrical and HVAC sample are described below. For full details, please refer to the materials from the Autodesk University 2010 class on the Revit MEP API, **CP316-3U All Systems Go** in Autodesk Revit MEP Programming.

AdnRme Electrical Sample

The AdnRme electrical sample determines and displays the hierarchical structure of the electrical systems in the model in several different ways. Its focus is to:

- Traverse the electrical system
- Reproduce the system browser data structure in a tree view
- Determine and display the complete connection hierarchy in a tree view

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. This sample inspects the electrical systems and reproduces the structure and information displayed by the system browser as well as the full connection hierarchy. It implements the following commands:

- CmdElectricalConnectors
- CmdElectricalSystemBrowser
- CmdUnhostedElements

The first two analyse the electrical system hierarchical structure and present it to the user in a tree view in a modeless dialogue. Being modeless, 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 will

not be reflected in the dialogue until the command is re-executed. A sample model elec_project.rvt is provided.

- Connection hierarchy tree
- Obsolete pre-connector approach
- System browser structure
- Find un-hosted elements

AdnRme HVAC Sample

The MEP HVAC sample supports a typical HVAC engineering workflow including the following tasks:

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

In a little more detail, the workflow might look like this:

- Manually lay out air terminals
- Run heating and cooling load analysis
- Based on load analysis, assign flow to terminals
- Based on terminal flow, size air terminals
- Validate design; a common check figure uses air flow per surface area

The MEP HVAC sample was originally implemented for Revit 2008 and based on room elements, since spaces had not yet been introduced. It was modified for RME 2009 to use spaces. It implements the following four commands to support the HVAC engineering workflow described above. The first three support the actual workflow itself, the fourth cleans up and resets the data manipulated by the first three:

- Assign flow to terminals
- Change air terminal size
- Calculate and display flow density as colour fill to verify design by air flow per surface area
- Reset demo

MEP Placeholder Sample

The MEP placeholder sample included in the class materials demonstrates the main features of the new Revit MEP 2012 API functionality discussed above:

- Placeholder ducts and pipes
- Duct and pipe insulation and lining
- Read and write access to MEP pipe settings and sizes

These three areas of functionality are demonstrated by the following four commands, matching the three points above:

- CreatePlaceholders and ConvertPlaceholders
- InsulateDuctwork
- GetPipeSettings

Further Reading

Here are a few places to go to find background material and further reading:

- Revit Developer Center: DevTV and My First Plugin introductions, SDK, Samples, API Help
<http://www.autodesk.com/developrevit>
- Developer Guide and Online Help
<http://www.autodesk.com/revitapi-wikihelp>

- Revit and Revit MEP API Trainings, Webcasts and Archives
<http://www.autodesk.com/apitraining> > Courses, Schedule, Webcast Archive > Revit API
- Discussion Group
<http://discussion.autodesk.com> > Revit Architecture > Revit API
- ADN AEC DevBlog
<http://adndevblog.typepad.com/AEC>
- The Building Coder, Jeremy Tammik's Revit API Blog
<http://thebuildingcoder.typepad.com>
- ADN, The Autodesk Developer Network
<http://www.autodesk.com/joinadn> and <http://www.autodesk.com/adnopen>
- DevHelp Online for ADN members
<http://adn.autodesk.com>
- Learning Autodesk Revit MEP 2012 video training
<http://cad-notes.com/2011/12/learning-autodesk-revit-mep-2012-training-video-is-available>