Everything in Place with Revit MEP Programming
Jeremy Tammik – Autodesk

CP4453 This class discusses working programmatically with Revit MEP models. The main focus is on the MEP API enhancements in Revit 2012 and two new MEP related APIs, the detailed energy analysis model and the conceptual energy analysis API. The major enhancements cover the new placeholder elements, pipe settings, insulation and lining. An overview of the entire Revit MEP API and existing 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, all types of MEP components, 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 be able to:
- Manage, analyze, create and modify HVAC, electrical and plumbing Revit MEP models, systems, and components programmatically
- Understand the major enhancements of the MEP API in Revit 2012: placeholder elements, pipe settings, insulation and lining, detailed energy analysis and conceptual energy analysis
- Understand and reuse functionality provided by the standard Revit MEP SDK and custom 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
# Everything in Place with Revit MEP

## Programming

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Introduction</td>
<td>3</td>
</tr>
<tr>
<td>MEP Application Requirements</td>
<td>3</td>
</tr>
<tr>
<td>The Generic Revit API</td>
<td>3</td>
</tr>
<tr>
<td>Revit MEP API Evolution</td>
<td>3</td>
</tr>
<tr>
<td><strong>Analysis</strong></td>
<td>4</td>
</tr>
<tr>
<td>EnergyDataSettings and gbXML</td>
<td>4</td>
</tr>
<tr>
<td>Conceptual Energy Analysis</td>
<td>4</td>
</tr>
<tr>
<td>Detailed Energy Analysis Model</td>
<td>4</td>
</tr>
<tr>
<td>Spaces and Zones</td>
<td>5</td>
</tr>
<tr>
<td>Model Inspection Utilities</td>
<td>5</td>
</tr>
<tr>
<td><strong>Hierarchical Systems and Connectors</strong></td>
<td>5</td>
</tr>
<tr>
<td>Hierarchical System Structure</td>
<td>6</td>
</tr>
<tr>
<td>MEP Model</td>
<td>6</td>
</tr>
<tr>
<td>Connectors</td>
<td>6</td>
</tr>
<tr>
<td>Connector Definition Elements</td>
<td>6</td>
</tr>
<tr>
<td><strong>Electrical</strong></td>
<td>8</td>
</tr>
<tr>
<td>HVAC and Plumbing</td>
<td>8</td>
</tr>
<tr>
<td>HVAC and Pipe Systems</td>
<td>8</td>
</tr>
<tr>
<td>Duct and Pipes</td>
<td>9</td>
</tr>
<tr>
<td>Fittings</td>
<td>9</td>
</tr>
<tr>
<td>Connectors</td>
<td>9</td>
</tr>
<tr>
<td>Element Creation</td>
<td>9</td>
</tr>
<tr>
<td><strong>The Revit MEP 2012 API</strong></td>
<td>10</td>
</tr>
<tr>
<td>Product Features</td>
<td>10</td>
</tr>
<tr>
<td>Major API Enhancements</td>
<td>10</td>
</tr>
<tr>
<td>Placeholder Ducts and Pipes</td>
<td>11</td>
</tr>
<tr>
<td>Insulation and Lining</td>
<td>11</td>
</tr>
<tr>
<td>Pipe Settings and Sizes</td>
<td>11</td>
</tr>
<tr>
<td>Small Enhancements and Changes</td>
<td>12</td>
</tr>
<tr>
<td><strong>Sample Applications</strong></td>
<td>12</td>
</tr>
<tr>
<td>AddSpaceAndZone</td>
<td>13</td>
</tr>
<tr>
<td>AutoRoute</td>
<td>14</td>
</tr>
<tr>
<td>AvoidObstruction</td>
<td>14</td>
</tr>
<tr>
<td>CreateAirHandler</td>
<td>14</td>
</tr>
<tr>
<td>EnergyAnalysisModel</td>
<td>15</td>
</tr>
<tr>
<td>PanelSchedule</td>
<td>16</td>
</tr>
<tr>
<td>PowerCircuit</td>
<td>16</td>
</tr>
<tr>
<td>TraverseSystem</td>
<td>17</td>
</tr>
<tr>
<td>AdnRme Sample</td>
<td>17</td>
</tr>
<tr>
<td>MEP Placeholder Sample</td>
<td>19</td>
</tr>
<tr>
<td><strong>Further Reading</strong></td>
<td>20</td>
</tr>
</tbody>
</table>
Everything in Place with Revit MEP

Programming

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
  - MEP project information
  - Green Building XML, gbXML
  - Spaces and zones
- MEP System Tools
  - Electrical systems, components, properties and parameters
  - Duct and pipe systems, components, properties and parameters
  - Element creation and modification
  - System 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 some 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 will simply do nothing or return 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

In Revit 2008, no MEP-specific API functionality was provided at all. It was still possible to analyse and manipulate an HVAC system using the 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 still works and is demonstrated by the sample
application for HVAC air terminal analysis and sizing, first presented at Autodesk University 2007, previously named mep and now renamed to AdnRme for Revit 2012.

Revit 2009 introduced the first MEP-specific API support, including 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.

In Revit 2010, the MEP-specific API was a focal point of the API development and greatly enhanced, especially the support for HVAC and piping systems. It introduced the Autodesk.Revit.MEP namespace, which replaced by the more specific namespaces listed above.

Revit MEP 2011 introduced cable trays and conduit elements and electrical panel schedules and consolidated most of the MEP related API classes into the new more specific namespaces. At that point, the MEP-specific API was more or less equally evolved as the architectural and structural flavours, and becoming widely used.

The main Revit 2012 API enhancements include placeholder elements, new insulation and lining functionality, and access to pipe settings and sizes. We will look at these topics in greater depth after the general MEP API overview.

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:

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

Conceptual Energy Analysis

Conceptual energy analysis deals with energy analysis on conceptual design models. New classes provide access to the elements and objects created by Revit to perform this kind of analysis. 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

A new feature added to the Revit API of special interest in the MEP domain is the detailed energy analysis model API. It 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 detailed energy analysis model is created and initialised using the static EnergyAnalysisDetailModel.Create method. The EnergyAnalysisModel SDK sample demonstrates the use of this new API.
Spaces and Zones
Returning to basics again, 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, one 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:

```csharp
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.

The ray intersection method can be used to determine proximity of elements:

- FindReferencesWithContextByDirection 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.

The Revit 2012 API also added new element filters to find 3D elements by intersection, the ElementIntersectsElementFilter and ElementIntersectsSolidFilter, support for Boolean operations, and other useful enhancements for accessing and analysing geometry.

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

This concludes the overview of the model analysis aspect of the Revit MEP API. Next, we explore the MEP system oriented functionality.

Hierarchical Systems and Connectors
Many components of an MEP system are represented in Revit by family instances. The generic family instance definition in the Revit API has been enhanced to provide access to additional MEP-specific properties and data, and provide 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 panel, the terminal elements, and the connector manager. The derived classes have additional domain specific properties.

MEP Model

The Revit 2009 API introduced the MEPModel and Connector classes as fundamentals 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 Revit flavour.

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; they 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.

So we have the elements living 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.
This is achieved using connectors which are not elements, nor family instances, and do not live in the project file. They are contained within the elements that they connect.

On the other hand, within the family definition, the connectors are represented by stand-alone elements.

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.

We thus have one set of classes for defining connector elements, residing in the family document. We have another class to represent the connectors on the resulting family instances, once inserted into a project document. The former are the DuctConnector, PipeConnector and ElectricalConnector classes, the latter the Connector class.

Inside a family document, building your equipment, you can add e.g. a DuctConnector to the model. This defines some information on the equipment hosting it. When you insert that equipment into the model and query it for its connectors, it returns Connector instances to you.

DuctConnector, PipeConnector and ElectricalConnector are the classes you see and use in the family editor. These objects are the MEP connector definition classes used as a part of the building blocks to define an MEP part:

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

This instance contains connectors, which are managed by the connection manager and represented by the Connector class:
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 may not be modelled individually, so the connections between the electrical system components can be defined by logical connectors rather than physical ones.

The classes ElectricalEquipment, LightingDevice, LightingFixture, ElectricalSystem, specialisations of the base MEPModel and MEPSystem classes represent electrical components and systems. These elements all have a connector manager, whose Connectors property returns logical as well as physical type connectors:

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

The connector manager can be used to query both physical and logical connectors to determine how the different components of an electrical system are connected together and construct the system hierarchy from that information. This is 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 classes MechanicalSystem and PipingSystem were introduced in Revit 2010 and provide the following top-level system functionality:

- Access to equipment, connectors and system type.
Programming

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

The elements contained within 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

The 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 are 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 so-called assigned connector properties. These are 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.
Everything in Place with Revit MEP

Programming

- NewDuct, NewFlexDuct, NewPipe and NewFlexPipe to create new elements.
- New*Fitting to create new fittings, * being one of Cross, Elbow, TakeOff, TeeFitting, Transition, Union.
- Newly introduced classes such as Conduit and CableTray provide static Create methods

In the context of a family, different types of connectors are represented by elements and created by the FamilyItemFactory class accessed through the Document.FamilyCreate property, which provides the NewDuctConnector, NewPipeConnector and NewElectricalConnector creation methods for this purpose.

The Revit MEP 2012 API

The Revit 2012 API includes a number of important MEP specific enhancements, some completely new APIs have been added which are especially useful in the MEP domain, and last but not least, all of the many significant Revit platform enhancements are useful for MEP as well. Before looking at the API, here is an overview from a product point of view:

Product Features

Here are some of the key features of Revit MEP 2012 from a product point of view:

- Placeholder elements
- Insulation and lining
- Parallel pipe and conduit runs
- Sloped piping enhancements: settings, tooltips, connection
- System browser enhancements: filtering, hovering and selection
- System enhancements: graphic overrides and warnings, disconnect markers, materials, calculation control, connector labels

The following API enhancements obviously build on and provide access to this functionality.

Major API Enhancements

The major Revit MEP enhancements include:

- 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 MepPlaceholders sample application included in the class materials. It defines the following four commands, matching the three points above:

- CreatePlaceholders and ConvertPlaceholders
- InsulateDuctwork
- GetPipeSettings

A number of minor enhancements have also been made to the Revit MEP API, e.g. spare and space circuits, cable tray and conduit domain, connector, MEP system and space properties, and disconnection warnings.

Completely new APIs which are applicable to the MEP domain include:

- Detailed energy analysis model API
- Conceptual energy analysis API
Furthermore, the Revit platform API enhancements include some features which are especially useful in the MEP domain, such as new classes for the analysis of room and space 3D geometry, new geometry functionality, and new support for handling linked files.

In the following sections, let us take a more detailed look at the most important aspects.

For more information, please refer to the What’s New section in the Revit API help file RevitAPI.chm included in the Revit SDK, which is available from the Revit Developer Center at http://www.autodesk.com/developrevit.

### Placeholder Ducts and Pipes

Revit MEP introduces placeholder elements which can be used to indicate a planned layout when the system itself has not been fully sized. The placeholders can be converted into true duct and pipe elements later on. Placeholders can be assigned a size and slope.

In the API, the following new properties identify placeholder ducts and pipes:

- Duct.IsPlaceholder
- Pipe.IsPlaceholder

These new static methods allow creation of placeholder ducts and pipes:

- Duct.CreatePlaceholder
- Pipe.CreatePlaceholder

New utility methods are exposed to convert a set of placeholder ducts and pipes to real 3D entities:

- MechanicalUtils.ConvertDuctPlaceholders
- PlumbingUtils.ConvertPipePlaceholders

This functionality is demonstrated for ducts by the CreatePlaceholders and ConvertPlaceholders commands in the MepPlaceholder sample application provided in the class materials.

### Insulation and Lining

In previous versions of Revit, insulation and lining had to be modelled separately in the family definitions, practically tripling the amount of geometry that needed to be defined. In Revit 2012, insulation and lining can be added programmatically instead, as demonstrated by the InsulateDuctwork command.

In the user interface, insulation can now be added to an entire duct or pipe run in a single command, and lining to an entire duct run in a similar way. Fittings and flexible ducts and pipes can also be selected and processed.

The new API classes DuctInsulation, PipeInsulation, DuctLining and related types support read, write and create access to duct and pipe insulation and lining. In Revit 2012, these objects are now accessible as standalone elements related to their parent duct, pipe, or fitting.

### Pipe Settings and Sizes

MEP pipe settings are now accessible programmatically. The following new classes provide read and write access to them, as demonstrated by the GetPipeSettings command:

- PipeSettings – The pipe settings class with one singleton object in the document, accessible through the static GetPipeSettings method.
Everything in Place with Revit MEP

Programming

- **PipeSizeSettings** – The main class to access pipe sizes with one singleton object of this type in a document accessible through the static GetPipeSizeSettings method. It stores PipeConnections per each pipe material, using the id of the pipe material element as key to access. PipeSizeSettingIterator defines an iterator over a set of PipeSizeSettings items, each of which is a KeyValuePair<ElementId, PipeConnections>.
- **PipeConnections** – Represents a set of pipe connection types. Stores PipeSchedules per each pipe connection type, using the id of the pipe connection type as the access key. PipeConnectionIterator defines an iterator over a set of PipeConnections items, each of which is a KeyValuePair<ElementId, PipeSchedules>.
- **PipeSchedules** – Represents a set of pipe schedules. Stores PipeSizes per each pipe schedule type, using the id of the pipe schedule type as key to access. PipeScheduleIterator defines an iterator over the PipeSchedules KeyValuePair<ElementId, PipeSizes> items.
- **PipeSizes** – Stores a set of MEP pipe sizes, with the ability to add and remove from the set as needed. PipeSizeIterator defines an iterator over PipeSizes.
- **MEPSize** – Stores the basic size information for an MEP duct, pipe, cable tray, or conduit.

**Small Enhancements and Changes**

A number of small enhancements and changes have been made to the Revit MEP API:

**Spare and space circuits**: The new property ElectricalSystem.CircuitType identifies the type of an electrical circuit (circuit, spare or space). The method ElectricalSystem.AddToCircuit now throws an exception when the system is a spare or space circuit.

**Cable tray and conduit domain**: The Autodesk.Revit.DB.Domain enumeration adds a new value to represent cable tray and conduit.

**Connector**: New read-only properties have been added for the JointType, GenderType and EngagementLength.

**MEPSystem**: The new property MEPSystem.IsEmpty identifies if the system currently contains no components.

**Graphical warnings for disconnects**: The new ‘show graphical warning’ properties on the Application class and the matching setters control whether Revit MEP will highlight disconnects in systems graphically.

**Space properties**: The new property Space.BaseHeatLoadOn indicates if the value of Space.LatentHeatGainperPerson and Space.SensibleHeatGainperPerson properties has the default value or is user-defined.

**Fitting methods**: The Document methods to create new fittings no longer remove unused or dangling curve connectors.

**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.
Everything in Place with Revit MEP

Programming

- **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.
- **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
Everything in Place with Revit MEP

Programming

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.

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.
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:
Everything in Place with Revit MEP

Programming

- 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

- A good starting point for all Revit developers is the resources listed on the Revit Developer Center – [http://www.autodesk.com/developrevit](http://www.autodesk.com/developrevit). These include:
  - The Revit SDK including the API help file RevitAPI.chm and the Developer Guide “Revit 2011 API Developer Guide.pdf”.
  - Training material, recorded presentations, especially the DevTV recordings, and the DevTV Visual Studio Revit add-in templates.
  - Links to the Autodesk discussion groups – [http://www.autodesk.com/discussion](http://www.autodesk.com/discussion). Click on the Revit link to access the Revit API discussion groups.

- Jeremy Tammik’s Revit API blog The BuildingCoder – [http://thebuildingcoder.typepad.com](http://thebuildingcoder.typepad.com) and some Revit MEP API related posts:

- For ADN members, there is a wealth of Autodesk API information on the members-only ADN website – [http://adn.autodesk.com](http://adn.autodesk.com). ADN members can ask unlimited API questions through our DevHelp Online interface.

- Materials from previous MEP API Autodesk University classes.