Revit 2017 API News

Notes from the Revit 2017 API News presentations at the 2015/2016 DevDays conferences at Autodesk University in Las Vegas, Munich and online.

Revit Kepler API

Assembled by Sasha Crotty, Revit Core Product Manager, edited by Jeremy Tammik.

Autodesk Confidential Information

Today's discussion is covered under your ADN agreement with Autodesk.

The information we will be providing may contain confidential information, and is to be shared within your company on "need to know basis" and to no one outside your company.

Autodesk makes no guarantees that anything presented or discussed will actually appear in the future.

Rice – Must Do

Rice | Visual Studio & .NET

All Revit API binaries are now built targeting .NET 4.5.2.

Revit builds with and installs runtime libraries from the Visual C++ Redistributable for Visual Studio 2015.

No guarantee that Revit will install any other runtime on client machines.

Rice | Add-In Security

Code Signing for Add-Ins: To help Revit customers understand the origin of code running on their system, Revit will now check for code signing status of all add-ins during load.

A message will be displayed for each add-in displaying the status of the security certificate (if any).

The Revit team highly encourages signing all add-ins.

The message can be bypassed by adding certificate(s) to certificate store.

The AppStore team is developing a simple process. Ask Mikako. Messages will pop up. You can turn that off by trusting.

Rice | Transaction Changes

TransactionMode.Automatic obsoleted: Use Transaction APIs to manage needed transactions. Use manual mode when making changes to the Revit model, and ReadOnly mode for commands that do not changes to the model.

Rice | Application API Changes

Removed default constructor for Autodesk.Revit.ApplicationServices.Application.

Obtain Application from arguments passed to Revit API callback, e.g.:

- UIApplication via ExternalCommandData
- Document via UpdaterData
- Event arguments

Not available in IExternalApplication callbacks OnStartup and OnShutdown:

- Use ControlledApplication instead
- Use ApplicationInitialized event to work with docs after startup

This is because Revit is not ready to perform operations with Documents while in this initialization state. If you need to start working with Documents immediately after Revit completes start-up, you can subscribe to the ApplicationInitialized event.

Rice | Point Cloud API Changes

PointCloudOverrides:

- Get/SetPointCloudOverrideSettings() methods replaced with Get/SetPointCloudScanOverrideSettings()
- New functions Get/SetPointCloudRegionOverrideSettings() support region overrides

This capability was requested. Now you can override just part of the point cloud. The previous method just did the whole thing.

Rice | MEP Systems

MEP system parameters are now calculated asynchronously.

To handle asynchronous calculation results, define callback methods to react when result is available.

If no callback defined, the calculation becomes synchronous.

Associated built-in parameters are still supported.

These parameters no longer support dynamic model update.

A secondary process called Revit worker is active in the background. Certain properties are taken out of regen and generated in a background process, a separate thread. They no longer block the user. If you are calling them through the API, you need to do so via a callback. If you do not use a callback, they will be forced to act synchronously after all, i.e., you have to wait. The built-in parameters still work. That will force a synchronous access. Obviously, DMU no longer works for these, since they are

asynchronous. The ability to use a background process will be used in other areas as well as we go forward. This is not using the external services framework, although the implementation could be set up to use these as well.

Rice | Obsolete API Removal

All APIs marked as deprecated in Revit 2016 have been removed.

Wine – New Functionality

Main areas of investment:

- Modelling
- Communication
- Efficiency

Efficiency for communicating and modelling designs.

Modelling

Model designs like they will be built while encoding design intent.

Capturing model intent. Automation of knowledge work. Revit does not have to know everything, but the user should be able to define ways for Revit to capture their logic and encode it in the model.

Modelling | Global Parameters

Global Parameters enable the capture of design intent by bringing the parametric power of families into the project environment. GPs can:

- Drive and measure dimensions / constraints
- Use formulas for complex calculations
- Drive parameter values of instances and types in the project

GPs are unique in that they are one value per project. They can be used to define values be pushed into dimensions or directly say the value of an element is driven by the GP. It is like a project setting that can use formulas and measure something in the project. The initial implementation was in R2, with enhancements and expansion. R2 was limited to instance parameters, now type and project parameters are also supported.

Modelling | Global Parameters

This shows a building that is entirely constrained using GP. The proportions of the rooms are relative to each other and driven by a total required area. 1. Assign measure GP to dimension at the bottom, 2. Apply GP to vertical aspect, 3. Plug those values into a formula 4. Entire building is driven by width. When you change dimension length, it reports it and recalculates the entire building. As it gets narrower it gets longer and wider makes it shorter. The total area is hardcoded. Next step: change the hardcoded area, make it bigger, and the entire building resizes. Last part: associate parameters. A couple of GPs measure the size of the top room. A family or array of chairs is placed into the room. The length and width of the family are associated, so that the chairs automatically adjust. This is maybe the most important new functionality. Absolutely unbounded potential!

Modelling | Global Parameters

The GlobalParameterManager manages Global Parameters in a project.

GlobalParameter:

- Project-wide parameter with value
- Value can be driven by a dimension, a formula, or explicitly defined
- IsReporting/IsDrivenbyDimension – is measuring/driving a dimension
- GetAffectedElements() – elements driven in any way by this GP

Not full list. Is this a measuring or a driving one? Reporting can only label one dimension. It could live by itself.

Modelling | Parameters

ParameterValue:

- Contains a value of a corresponding global parameter
- Derived classes for Integer, Double, String, ElementId, and Null
- All derived classes have one property: Value

Parameter: AssociateWithGlobalParameter() makes a GP drive the parameter value of an element.

ParameterType: MultilineText – enables creation and use of multiline text parameters.

This might be used with normal parameters as well in future. Multiline text was added previously in the UI, but got missed in the API. Just a missing enum.

Modelling | IFC4

Enhanced IFC 4 capabilities. Export to IFC 2.x.3 versus export to IFC 4. The video demonstrates importing both files back into Revit. Key thing to note: file size difference due to tessellated shape versus original shape. The important underlying feature is the geometry tool improvements.

Modelling | Geometry API

ShapeImporter: ShapeImporter.Convert() conversion of geometry, materials, and graphics styles stored in external formats (e.g. SAT and Rhino) into Revit geometry objects.

BRepBuilder

- Create Revit geometry (either solids or "open sheets") from input surfaces, edges, and boundary loops
- Use directly in any other Revit tool that accepts geometry, including DirectShape

DirectShape.IsValidCategoryId() now correctly lists categories approved for use with DirectShape.

New functionality, not yet available in the UI. It uses the Autodesk translation framework, so you can open SAT and Rhino files directly in Revit. It does not support all the shapes. Some complicated things... therefore no UI. Central format for defining geometry. Any tool can write

an importer or exporter. This should provide entirely consistent translation. B-rep builder is really cool. Define edges, surfaces and boundary loops and Revit will generate either a solid or a B-rep surface from it. Revit solids can now be generated from much more complicated input than before. The B-rep builder output can be plugged into any place that takes geometry. IsValidCategory no longer returns all top level categories.

Modelling | Geometry API Continued

Surface:

- New subclasses: Cylindrical, Conical, Ruled, Revolved, Hermite
- Plane.CreateByThreePoints()

NurbSpline:

- Curve returned by the new creation methods may be a NURBSpline or a simpler curve such as line or arc
- Revit uses simplest possible representation of curves

TessellatedShapeBuilder:

- Build() no longer returns the build results directly; use GetBuildResult() for that instead
- Set build options on TessellatedShapeBuilder object

Nurb spline methods are re-implemented. Most important: on point input it will return the simplest implementation, e.g. a line for four collinear points, not a spline. This is consistent with what Revit does, reduces file size, improves performance etc. Some reorg was required to achieve this. The build function no longer returns a result directly, you have to call a method to retrieve it, and a few additional new properties can be set on the class.

Modelling | Hosted Railings

Railings can now be hosted on different objects. They adjust to the surface. They can be hosted on floors, roofs, tops of walls, etc.

Modelling | Railing API

BaseRailing

- Create() creates a new railing by specifying its path
- SetPath() sets the railing path
- RailingCanBeHostedByElement() checks if the specified element can be used as a railing host

Straightforward API.

Modelling | Reinforcement – Fabric Sheets

FabricSheet numbering and segment control:

- FabricNumber is a reinforcement numbering value for the fabric sheet
- New methods to identify and specify segment length(s)

Fabric sheet layout control:

- FabricSheetLayoutPattern.QuantitativeSpacing specifies a layout based on spacing and diameter
- FabricSheetType.SetLayoutAsCustomPattern() uses FabricWireItems (single wires) to specify a custom layout

New functionality for numbering, finding segments, specifying segment length, controlling layout of rebar within the sheet or a completely custom pattern, defining location of every single wire.

Modelling | Reinforcement – Rebar Container

RebarContainer:

- SetItemHiddenStatus() / IsItemHidden() provides access to the option to hide an individual RebarContainerItem in a given view
- PresentItemsAsSubelements identifies if Items should be presented in schedules and tags as separate sub-elements

Ability to hide a single bar within the container. Consider sub-elements. A sub-element can optionally be used in schedules and tags as if it were an element.

Modelling | Structural Connections

StructuralConnectionHandler:

- Defines a connection between a whole set of elements
- Participating categories: structural framing, columns, walls, floors, and foundations
- GetOrigin specifies the connection's origin
- SetDefaultPrimaryElement defines primary element in the connection

New class for advanced steel integration. We acquired this product for steel detailing. Handler enables to define a set of elements that are part of a connection. This is API, not automatic, define origin, define the primary element, and define the elements forming a part of the connection. It basically just stores data and defines a concept.

Modelling | MEP Fabrication

FabricationSettings exposes user-specifiable settings related to placement of FabricationParts.

DesignToFabricationConverter supports the conversion of design elements to FabricationParts.

FabricationPart StretchAndFit() stretches the fabrication part from the specified connector and fits to the target routing end.

~40 other properties & methods.

This is a whole bunch of stuff compressed. We invested a lot of time and effort in MEP fabrication to drive direct fabrication from Revit. Most important: the converter; stretch and fit. Lots more!

Communication

Create stunning visuals and documents to easily communicate designs.

Communication | Autodesk RayTracer

This is a ray tracer with better rendering results, better and faster than mental ray. Look at the reflections of the windows on the tabletop. That took forever in mental ray. The same tool is used for the interactive render tool.

Communication | Rendering API

Autodesk RayTracer is replacing MentalRay. Some settings have been removed.

New enums:

- RenderDuration
- LightAndMaterialAccuracyMode

RenderingQualitySettings are redesigned.

2016 R2 had the two side by side; now MentalRay has been removed. The rendering duration could be infinite. There is no end state. You can cut it off by setting the duration.

Communication | Updated Text Editor

Finally. Lots of painful problems solved. Text is measured just like Mtext in AutoCAD. You can add spaces around bulleted points. A wrapping issue has been fixed.

Communication | Text

Upgrading from 2016 will modify existing text notes.

FormattedText:

- Read and write text and formatting of a text note
- Obtained via TextNote.Get/SetFormattedText()
- Access text & formatting for whole text note or a range via class TextRange

TextEditorOptions control Revit's Text Editor appearance and functionality.

Key thing: existing text notes will change. Please review your drawings. The entire machinery has changed. Wrapping may change. It depends on the font. Arial font works best.

Communication | Depth Cueing

This is an important architectural visualisation style.

ViewDisplayDepthCueing:

- Fade line and surface display intensity between front and back clip plane of a view
- EnableDepthCueing enables/disables depth cueing
- StartPercentage / EndPercentage – percentage offset from front clip plane where fade begins/ends
- FadeTo defines the maximum fade percentage

The View class Get/SetDepthCueing() defines depth cueing settings for a view.

It measures between the front and back clip plane and you can define what ranges you would like to see.

Communication | Schedules

Concatenated schedule fields:

- ScheduleDefinition.InsertCombinedParameterField inserts a schedule field with combined parameters
- ScheduleField.Get/SetCombinedParameters() defines combined parameter fields
- Enum value ScheduleFieldType.CombinedParameter

You can now create a concatenated schedule field.

Efficiency

Leverage the power of the best BIM tool to get projects done.

Efficiency | New Document

Application.NewProjectDocument(UnitSystem) creates a new imperial or metric project document without a specified template.

Efficiency | Revit Link API

Link instance locations:

- MoveBasePointToHostBasePoint() moves a RevitLinkInstance so that the link and host base points coincide
- MoveOriginToHostOrigin() performs a one time move that aligns internal origins of the linked document and host document

Local unload of Revit Links:

- UnloadLocally() unloads a Revit link for the current user only without affecting other users
- RevertLocalUnloadStatus() removes a user's local link override

Rearrange links. Local unload of a link.

Efficiency | Family API

PromptForFamilyInstancePlacement() options:

- SketchGalleryOptions controls types of sketched curves which can be used for a curve-based family
- FaceBasedPlacementType provides options for placement of a face-based family

FamilyInstance.CanSplit/Split() splits a curve-driven family instance.

Efficiency | Family Preview Visibility

In the family environment, an overlapping geometry filter view setting. Almost like preview, you can filter out geometry, similarly to setting a temporary view mode.

Efficiency | View API

Temporary view modes:

- New class TemporaryViewModes
- Identify and control on/off state of temporary view modes

Plan view underlay provides full access to underlay levels and settings via View.

Dependent views: View.ConvertToIndependent().

Assembly view creation with template information: AssemblyViewUtils.Create defines overloads with new arguments viewTemplateId and isAssigned.

Working towards a consistent API.

Efficiency | Tangency Locks

Two curves can be locked tangentially to each other.

Efficiency | Tangency Lock API

For a CurveElement, the new method GetAdjoinedCurveElements() provides info about elements joined to this curve element.

New methods to access and modify tangent constraints:

- HasTangentJoin() – is the current join tangent
- Get/SetTangentLock() – enforce tangency at curve join

Efficiency | UI API

You can now launch the ColorSelectionDialog, FileOpenDialog and FileSaveDialog from the API.

TaskDialog ExtraCheckBoxText / WasExtraCheckBoxChecked() provides access to an extra checkbox shown in a TaskDialog.

Dockable Panes: DockablePaneProviderData.VisibleByDefault controls the default visibility of dockable panes. Default is true.

Return user selection. Extra checkboxes can be used for things like 'don't show again'.

Efficiency | Journaling Support

You can now use JournalData in overridden commands.

- BeforeExecutingEventArgs.UsingCommandData
- ExecutedEventArgs.Get/SetJournalData()
- Store journal data associated to an overridden command similar to External Commands

Overridden commands previously had no possibility to define journaling data for replay purposes. Now they can.

Lots More!

Global Parameters*, Railing Hosts, Import Shapes, Thermal Zoning*, Copy and Paste in perspective*, IFC4 import/export support for "advanced BRep"*, Family preview visibility*, Tangency Locks, Isolate solids/voids with filter tool*, Architectural Depth Cueing, In-Canvas Reference Plane Naming*, Autodesk Raytracer Rendering, Text Editor, Calculated Values in Tags, Pinned tags, Tag Leader improvements, Schedule View Templates, Occlusion Culling*, Smart Views, Faster Display of Walls and Ducts, Reference plane subcategories, Per-user RVT Unload*, View Underlay and Orientation improvements*, Project Browser scroll bar remembers location*, Cancel Multi-sheet Print/Export*, Project Base to Project Base Point*, Reference plane subcategories, Per-user RVT Unload*, View Underlay and Orientation improvements*, Project Browser scroll bar remembers location*, Cancel Multi-sheet Print/Export*, Project Base to Project Base Point*, Add-in Security, Manage Links dialog improvements, Energy Settings dialog improvements, Railing Preview*, Family types dialog enhancements, Filters UI*, Revision Enhancements*, View Range enhancements*, Project creation API, Family APIs, View APIs, Text APIs, Geometry APIs, Global Parameter APIs, CurveElement APIs, Railing APIs, UI APIs, Link APIs, Category APIs, Dynamo integration, etc.

Over 50 new features and APIs are part of Architecture and Core for the 2017 release of Revit. The key features are Global Parameters, Depth Cueing, Text, Occlusion Culling, and Calculated Values in Tags.

Hidden Slides

Efficiency | Electrical

Wire: GetMEPSystems() lists system(s) to which the wire belongs.

ElectricalSetting:

- CircuitRating – default circuit rating for a newly created circuit
- CircuitLoadCalculationMethod – specify method Revit will use to calculate the apparent circuit load

Efficiency | Duct & Pipe

DuctSettings & PipeSettings:

- FlatOnTop/Bottom – abbreviation of the Flat On Top / Bottom strings
- SetUp/Down – abbreviations of the Set Up/Down strings
- Centerline – abbreviation of the Centerline (=) string

PipeScheduleType:

- Create(Document, String) – creates a new pipe schedule type with given name
- GetPipeScheduleId(Document, String) – returns an existing pipe schedule type with given name

Rice | MEP, Structure and General API Changes

Deprecated APIs have been replaced by improved new ones. The areas with renamed and replaced classes and methods include:

Structure:

- Foundation Wall
- Rebar
- Fabric Sheet
- Load Case

MEP:

- Ducts
- Pipes

Miscellaneous:

- Planes
- Category Visibility
- Text in Custom Export
- Energy Data Settings