Major changes and renovations to the Revit 2020 API

API changes

Revit automatically initializes CEFsharp

Revit and Autodesk add-ins use the CEFsharp library internally for several features. Some third-party add-ins do so as well. Occasionally, when different versions of the library are used, it leads to instability issues for Revit.In order to avoid version conflicts, we are clarifying what CEFsharp version is being used, and loading it prior to all add-in initialization.

- In this version, Revit uses CEFsharp version 65.0.1.
- In the initialization, legacy JavaScript binding is enabled:
    - `CefSharpSettings.LegacyJavascriptBindingEnabled = true`

MEP Fabrication API deprecation

The following member has been deprecated and replaced:

Deprecated member	Replacement
FabricationConfiguration.GetMaterialThickness()	FabricationPart.MaterialThickness

Structural API deprecation

The following member has been deprecated:

Deprecated member	Replacement
Autodesk.Revit.DB.Structure.StructuralConnectionHandler.GetSubPartIds()	n/a

The subpart functionality was previously removed, so there is no replacement method.

Export API changes

The following member has been deprecated and replaced:

Deprecated member	Replacement
CustomExporter.Export(View3d)	CustomExporter.Export(View)

See the API additions section for more details on the new 2D view support for the custom exporter.

Image API changes

With changes made to support PDF import and extensions to the image API, the following members and classes have been deprecated and replaced:

Deprecated member	Replacement
ImageType.Create(Document, string)	ImageType.Create(Document, ImageTypeOptions)
ImageType.ReloadFrom(string)	ImageType.ReloadFrom(ImageTypeOptions)
Document.Import(string, ImageImportOptions, View, out Element)	ImageType.Create() followed by ImageInstance.Create()
class ImageImportOptions	class ImageTypeOptions

Obsolete API removal

The following API members and classes which had previously been marked Obsolete have been removed in this release. Consult the API documentation from prior releases for information on the replacements to use:

Classes

Methods

- ProjectLocation.IsProjectLocationNameUnique(Document, String)
- FabricationPart.IsStraightSegment(Document, ElementId)
- FabricationPart.CanSplitStraight(Document, ElementId, XYZ)
- FabricationUtils.ExportToMAJ(Document, IList<ElementId>,String, bool, out IList<ElementId>)
- ParameterFilterElement.Create(Document, String, ICollection<ElementId>, IList<FilterRule>)
- ParameterFilterElement.GetRules()
- ParameterFilterElement.SetRules()
- ParameterFilterElement.AllRuleParametersApplicable (IList<FilterRule>)
- ParameterFilterElement.AllRuleParametersApplicable (Document, ICollection<ElementId>, IList<FilterRule>)
- ParameterFilterElement.GetRuleParameter(FilterRule)
- ParameterFilterElement.GetRuleParameters()

Properties

- BasicFileInfo.SavedInVersion
- AssetProperty.Item[String]
- Application.Assets[AssetType]

API additions

Attached detail group API additions

The Group and GroupType API now includes API related to attached detail groups.

The new methods:

- Group.ShowAttachedDetailGroups()
- Group.ShowAllAttachedDetailGroups()
- Group.HideAttachedDetailGroups()
- Group.HideAllAttachedDetailGroups()

control visibility for an element group's attached detail groups as seen in a given view.

The new methods:

- Group.GetAvailableAttachedDetailGroupTypeIds()
- GroupType.getAvailableAttachedDetailGroupTypeIds()

returns the attached detail groups available for this group or group type.

The new method:

- Group.IsCompatibleAttachedDetailGroupType()

checks if the orientation of the input attached detail group matches the input view's orientation. Note that detail groups in perpendicular elevation views (for example, North and East views) are considered compatible. When showing these detail groups, an error (FailureMessage) based on id can be generated if the orientation of the annotations do not match the orientation of the target view (for example, the failure definition DimensionPerpendicularToView). To prevent displaying detail groups in the wrong view, you can check the OwnerViewId of a detail group to make sure it matches the view in which you are trying to display it.

The new properties:

- Group.IsAttached
- Group.AttachedParentId

provide information about whether a group is attached and if so, to what group it is associated.

PDF and Image API additions

ImageType API

The new members:

- ImageType.Create(Document, ImageTypeOptions)
- ImageType.CanReload()
- ImageType.ReloadFrom(ImageTypeOptions)
- ImageType.ExternalResourceType
- ImageType.PageNumber
- ImageType.PathType

add support to loading of specific pages from PDFs as images, loading of images from external resource locations other than paths, and storage of paths as relative or absolute. The API has also been revamped to allow loading of an ImageType in one operation, and placement in a view in a separate operation.

Some of these members reference the new class

- ImageTypeOptions

that contains the options that are used when creating or reloading an image, including:

- ImageTypeOptions.IsValid()
- ImageTypeOptions.PageNumber
- ImageTypeOptions.Path
- ImageTypeOptions.Resolution
- ImageTypeOptions.SetExternalResourceReference()

Resolution is measured in dpi and relates the number of pixels in raster images to their size.

ImageInstance API

The new class

- ImageInstance

represents an image placed in a view. It includes:

- ImageInstance.Create(Document, View, ElementId, ImagePlacementOptions)
- ImageInstance.GetLocation(BoxPlacement)
- ImageInstance.SetLocation(XYZ, BoxPlacement)
- ImageInstance.Width
- ImageInstance.Height
- ImageInstance.EnableSnaps

Snapping capability can be enabled only for PDF-based instances.

ElementLogicalFilter API additions

Setting filters

The new function:

- ElementLogicalFilter.SetFilters()

allows setting ElementFilters that are combined by the given ElementLogicalFilter. This allows an easier method to modify the filter criteria stored in a View filter.

Application API additions

Journal playback

The new function:

- Application.IsJournalPlaying()

determines if Revit is currently in journal playback mode or not.

UI API additions

Add Animated Progress Bar to a TaskDialog

The new property:

- TaskDialog.EnableMarqueeDialogBar

allows the TaskDialog to include a progress bar that has an indeterminate start and stop. The animation continues until the TaskDialog is closed.

Specify minimum width and height for a DockablePane

Minimum width and height can now be set for a DockablePane:

- Autodesk.Revit.UI.DockablePaneState.MinimumWidth
- Autodesk.Revit.UI.DockablePaneState.MinimumHeight

These are specified as part of the InitialState for the registered pane. The default for both properties is 200 pixels.

Document additions

Returning cloud model information for cloud workshared models

The new property:

- Document.IsModelInCloud

indicates if the current document is located in cloud.

- Document.CloudModelGUID

returns the GUID of the model, if it is stored in the cloud.

The new method:

- Document.GetCloudModelPath()

returns the cloud model path.

Open/Save/Modify cloud models on BIM 360

The new method:

- Document.SaveAsCloudModel()

saves the current model as a cloud model in BIM 360. Only non-workshared models can be saved via this method.

The new method:

- Document.SaveCloudModel()

saves the current cloud model.

The new method:

- Document.EnableCloudWorksharing()

converts a cloud model into a workshared cloud model. The method Document.CanEnableCloudWorksharing() can be used to check whether this operation is valid on a given model.

Revit's existing model open functions (Application.OpenDocumentFile and UIApplication.OpenAndActivateDocument) will now throw RevitServerUnauthorizedException if the user tries to open a cloud model without the right entitlements.

Identifying if background calculations are underway

The new method:

- Document.IsBackgroundCalculationInProgress()

identifies if there are any background calculations in progress for this document. When a document has background calculations in progress, users cannot perform the following operations:
- Save/Close the document.
- Export/Print the document.
- Synchronize to central, in worksharing environment.
- Create a steel element.
- Copy/Mirror/Rotate a steel element.
- Edit the sketch of plate element.
- Edit a custom connection.

Category API additions

Category visibility

The new property:

- Category.IsVisibleInUI

returns true if the category should be visible to the user in lists of categories in the Revit user interface (dialogs such as Visibility Graphics or View Filters).

Get string names of categories

The new method:

- LabelUtils.GetLabelFor (BuiltInCategory)

returns the string name of the given BuiltInCategory in the current Revit language.

Parameter API additions

Getting localized parameter names

The new method:

- LabelUtils.GetLabelFor(BuiltInParameter, Autodesk.Revit.ApplicationServices.LanguageType)

returns the localized string name representing the built-in parameter in the input language.

Hiding empty parameters

The new property

- ExternalDefinition.HideWhenNoValue

Indicates if the shared parameter should be hidden from the property palette and Element.GetOrderedParameters() when it has no value.

Similar functionality has been added to other classes:

- ExternalDefinitionCreationOptions.HideWhenNoValue
- SharedParameterElement.ShouldHideWhenNoValue()

The new method:

- Parameter.ClearValue()

can reset the value of a shared parameter which has the HideWhenNoValue flag set back to a cleared state. (This method is not applicable to clear the value of any other parameter type).

Filtering by presence or absence of parameter value

The new filter classes:

- ParameterValuePresenceRule
- HasValueFilterRule

- HasNoValueFilterRule

define filter rules evaluating whether or not a parameter has a value for a specific element.

Use the static methods:

- FilterRule.CreateHasValueParameterRule()
- FilterRule.CreateHasNoValueParameterRule()

to create an instance of these rules for use in a parameter filter.

In schedules, use the new enumerated values:

- ScheduleFilterType.HasValue
- ScheduleFilterType.HasNoValue

along with the method:

- ScheduleDefinition.CanFilterByValuePresence()

to filter based on the presence or absence of a value assigned to that parameter.

Geometry API additions

CurveLoop API

The new property:

- CurveLoop.NumberOfCurves

returns the number of curves in the curve loop.

The new method:

- CurveLoop.CreateViaOffset()

creates a new curve loop that is an offset of the existing curve loop. This method differs from the previously exposed API in that it takes as input a double array of offset distances instead of one double offset distance. The size of this array must match the size of the curve loop. The curve at position *i* will be offset with the double at position *i* in the array.

View API additions

View Template API additions

The new methods:

- View.CreateViewTemplate()

- View.isViewValidForTemplateCreation()

allow you to turn an existing view into a view template. The created view template has an automatically generated unique name.

The Revit API, for a few releases, has featured a fairly fully-featured capability to set up and modify view templates and their properties. These capabilities are now demonstrated in the new sample "ViewTemplateCreation".

View Filter API additions

Validation restrictions for the ElementFilter stored by a ParameterFilterElement (the class that represents a View Filter) has been relaxed to enable more flexible creation of OR filters where criteria can reference parameters only associated to specific categories. Currently the ElementFilter must be either an ElementParameterFilter or an ElementLogicalFilter representing a Boolean combination of ElementParameterFilters. In addition, we check that each ElementParameterFilter satisfies the following conditions:

- Its array of FilterRules is not empty and contains:
    - Any number of FilterRules of type FilterValueRule, FilterInverseRule, and SharedParameterApplicableRule or
    - Exactly one FilterCategoryRule containing only one category from categories stored by this ParameterFilterElement or
    - Exactly two rules: the first one is a FilterCategoryRule containing only one category from categories stored by this ParameterFilterElement and the second one is a FilterRule of type FilterValueRule, FilterInverseRule, or SharedParameterApplicableRule.

Note that cases in the second and third bullet are currently allowed only if the parent node of ElementParameterFilter is LogicalOrFilter.

In addition to the change in validation, the new method:

- ParameterFilterElement.GetElementFilterParametersForCategory()

retrieves a list of the parameters associated with all rules in the filter that are combined (using logical AND) with a FilterCategoryRule corresponding to single categoryId.

Schedule API additions

The new property:

- ScheduleDefinition.ShowGridLines

indicates if the schedule grid lines will be visible on a sheet.

The new property:

- TableData.ZoomLevel

allows the user to set the zoom level of a schedule in a tabular view. This setting will not change the size of text, rows, or columns in a sheet view. Additionally, the setting is temporary and only applies to the current session.

Site API additions

TopographySurface additions

The new method:

- Autodesk.Revit.DB.Architecture.TopographySurface.Create(Document, IList<XYZ> points, IList< IList<int> > facets, String)

creates a topography surface from a given list of triangulation facets.

The new method:

- Autodesk.Revit.DB.Architecture.TopographySurface.IsValidFaceSet()

checks whether the given set of triangulation data is valid.

TopographyLinkType additions

The new class:

- Autodesk.Revit.DB.Architecture.TopographyLinkType

represents a site file brought into the current Revit model as a link.

The new method:

- Autodesk.Revit.DB.Architecture.TopographyLinkType.Reload()

reloads the TopographyLinkType from its current location.

Shared Coordinates API additions

SiteLocation addition

The new method:

- SiteLocation.SetGeoCoordinateSystem()

sets the Geo coordinate system for the current document.

BasePoint additions

The new read-only property:

- BasePoint.Position

gets the XYZ value corresponding to the base point's position in Revit's internal coordinates.

The new read-only property:

- BasePoint.SharedPosition

gets the XYZ value corresponding to the base point's position in the transformed (shared) coordinates.

Part API additions

Part creation from DirectShapes

Parts can now be created from DirectShape instances, either in the same host document or in a link. Since various reasons might prevent parts from being created from an element, the new methods

- DirectShape.CanCreateParts()
- DirectShapeType.CanCreateParts()

indicate whether it is possible to create parts from an instance of one of those classes.

Returning the entities that create a divided Part

The new methods:

- PartUtils.GetSplittingCurves(Document, ElementId)
- PartUtils.GetSplittingCurves(Document, ElementId, out SketchPlane)

identify and return the curves that were sketched to create the part division and, optionally, also outputs also the sketch plane for those curve(s).

The new method:

- PartUtils.GetSplittingElements(Document, ElementId)

identifies and returns the elements (ReferencePlane, Level or Grid) that were used to create the division.

Railing API additions

BalusterInfo reference name

The new functions:

- BalusterInfo.GetReferenceNameForHost()
- BalusterInfo.GetReferenceNameForTopRail()

allow access to the name string to be used as a reference to either the Host or Top Rail in the current Revit language.

Export API additions

CustomExporter now supports some 2D views

The class CustomExporter can now export 2D views. The only 2D view types supported at this time are plans, sections and elevations. The new method:

- CustomExporter.Export()

accepts either a 3D or 2D view.

The method:

- CustomExporter.Export(IList<ElementId>)

can now accept either 3D or 2D views, with the limitation that views in the collection must be either all 3D or all 2D.

For both Export() calls, the exporter context must correspond to the views' type; use IModelExportContext or IPhotoRenderContext for 3D views and IExportContext2D for 2D views.

Several new properties expand the options for the CustomExporter to support 2D objects:

- CustomExporter.Export2DGeometricObjectsIncludingPatternLines - Indicates whether pattern lines of geometric objects should be exported in a 2D context. Defaults to false.
- CustomExporter.Export2DIncludingAnnotationObjects - Indicates whether annotation objects should be exported in a 2D context. Defaults to false.
- CustomExporter.Export2DForceDisplayStyle - Forces a display style for the export. If the style is DisplayStyle.Undefined, then export uses DisplayStyle.Wireframe for wireframe views and DisplayStyle.HDR for other views. Defaults to DisplayStyle.Undefined.

The new interface:

- IExportContext2D

should be used for exporting 2D views. It has the following methods in addition to the method inherited from IExportContext:

- IExportContext2D.OnElementBegin2D()
- IExportContext2D.OnElementEnd2D()
- IExportContext2D.OnFaceEdge2D()
- IExportContext2D.OnFaceSilhouette2D()

The new classes:

- ElementNode
- FaceEdgeNode

- FaceSilhouetteNode

contain data for various 2D exported objects.

Some notes on 2D export in DisplayStyle.Wireframe:

1. Geometric object methods (OnCurve, OnFaceEdge2D, OnFaceSilhouette2D) are called regardless of the object being eventually output, i.e. even if it's occluded by another element.

And in DisplayStyle.HDR:

1. Tessellated geometry methods (OnLineSegment and OnPolylineSegments) are called regardless of the return value of the respective geometric object methods (OnCurve, OnFaceEdge2D, OnFaceSilhouette2D).
2. None of these methods are called between the respective pairs of calls OnInstanceBegin/OnInstanceEnd or OnLinkBegin/OnLinkEnd. They are called between OnElementBegin2D/OnElementEnd2D and OnViewBegin/OnViewEnd.

For an example of the use of the API for custom export of 2D views, see the new sample CustomExporter/Custom2DExporter.

New Navisworks export options

Three new options have been added to the class NavisworksExportOptions:

- NavisworksExportOptions.FacetingFactor
- NavisworksExportOptions.ConvertLights
- NavisworksExportOptions.ConvertLinkedCADFormats

Note that you will need to get an updated Navisworks Exporter to successfully use these options.

Multi-Reference Annotation additions

Additional References

MultiReferenceAnnotations can now pick additional references from the model. The following new methods support this:

- MultiReferenceAnnotationOptions.SetAdditionalReferencesToDimension()
- MultiReferenceAnnotationOptions.GetAdditionalReferencesToDimension()
- MultiReferenceAnnotation.AreElementsValidForMultiReferenceAnnotation()

Note that the additional references to dimension cannot come from the same category as the MultiReferenceAnnotationType's reference category. The following function can be used to verify this:

- MultiReferenceAnnotationOption.ReferencesDontMatchReferenceCategory()

3D view placement restrictions

The new method:

- MultiReferenceAnnotation.Is3DViewValidForDimension()

is used by MultiReferenceAnnotation.Create() to verify that the dimension is valid for the view. LinearFixed dimensions cannot be created in 3D views. Linear dimensions can only be created in 3D views if the view direction is perpendicular to the current work plane normal.

Material API additions

Glazing schema API

To assist in creating code accessing and manipulating the properties of a given schema, predefined properties have been introduced to allow a compile-time reference to a property name without requiring you to transcribe it as a string in your code. New predefined properties are available in the new class:

- Autodesk.Revit.DB.Visual.AdvancedGlazing

whose contents represent the appearance asset of type "advanced glazing".

Analysis API additions

Path of Travel API

The new class:

- Autodesk.Revit.DB.Analysis.PathOfTravel

represents the element that calculates the shortest navigable path between two points in a plan view.

Some of the new members of this class include:

- PathOfTravel.Create() - creates a single PathOfTravel element given a start and end point.
- PathOfTravel.CreateMultiple() - creates multiple PathOfTravel elements given arrays of start and end points.
- PathOfTravel.CreateMapped() - creates multiple PathOfTravel elements by mapping each of a set of start points to each of a set of end points.
- PathOfTravel.GetCurveLoop()
- PathOfTravel.PathStart
- PathOfTravel.PathMidpoint
- PathOfTravel.PathEnd

The new class:

- Autodesk.Revit.DB.Analysis.RouteAnalysisSettings

represents a new settings element which contains project-wide settings for route calculations in plan views. Currently, these settings are only used by the Autodesk.Revit.DB.Analysis.PathOfTravel element, but in the future they can be used by any other functionalities which do route calculations.

Some of the new members include:

- RouteAnalysisSettings.EnableIgnoredCategoryIds() - allows for the API user to enable ignoring certain categories of elements during calculations.
- RouteAnalysisSettings.SetIgnoredCategoryIds() - sets the categories to be ignored during calculations.
- RouteAnalysisSettings.AnalysisZoneTopOffset - the analysis zone top, specified as an offset above the level of the plan view.
- RouteAnalysisSettings.AnalysisZoneBottomOffset - the analysis zone bottom, specified as an offset above the level of the plan view.

Structural API additions

RebarConstraint additions

The new method:

- RebarConstraint.IsReferenceValidForConstraint()

checks if the reference provided can be used in creating Rebar constraints.

Updating free-form rebar based on shared parameters

Free-form rebar can now regenerate based on one or more shared parameters. Several methods have been added to support this behavior:

- Autodesk.Revit.DB.Structure.RebarFreeFormAccessor.AddUpdatingSharedParameter()
- Autodesk.Revit.DB.Structure.RebarFreeFormAccessor.RemoveUpdatingSharedParameter()
- Autodesk.Revit.DB.Structure.RebarFreeFormAccessor.GetUpdatingSharedParameters()
- Autodesk.Revit.DB.Structure.RebarUpdateCurvesData.GetChangedSharedParameterGUIDs() - Returns an array containing the shared parameters which were changed since the last regeneration.
- Autodesk.Revit.DB.Structure.RebarUpdateCurvesData.GetRebarId() - Returns the ElementId of the rebar being regenerated.

Note that the shared parameter(s) must already be bound to the Rebar category before setting up the dependency.

StructuralConnectionHandler additions

The new property:

- Autodesk.Revit.DB.Structure.StructuralConnectionHandler.OverrideTypeParams

allows or disallows overriding the parameters for a connection type.

When set to true, a set of instance parameters is created for this connection by copying the type parameter set. The user can change these instance parameters in order to make this connection different from the others of the same type. Any further modification on type parameters will not affect this instance (until the "override" is turned back off).
When set to false, the connection instance parameters are discarded and the type parameters are used again.

MEP API additions

FabricationPart additions

The new method:

- FabricationPart.GetInsulationLiningGeometry()

returns any insulation or liner geometry for a fabrication part. If there is no insulation or liner applied the return value will be null.

FabricationNetworkChangeService

The new method:

- FabricationNetworkChangeService.ChangeService()

changes the service of a selection of element identifiers to the passed in service, group identifiers, and the option to restrict the placement of parts that only exist in the group identifier.

ElectricalSystem API additions

The new properties:

- ElectricalSystem.CircuitConnectionType
- ElectricalSystem.IsBasePanelFeedThroughLugsOccupied

provide access to the the circuit connection type of the electrical system, and information about whether or not the feed through lugs of the base panel is already occupied.