Major changes and renovations to the Revit API

1 API changes

1.1 CefSharp upgrade

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 initializations.

- In this version, Revit uses CEFsharp version **v119.4.30**

1.2 Revit now on .NET 8

The Revit API for Revit 2025 is built on .NET 8. This upgrade keeps Revit up-to-date with the latest .NET features, performance improvements, and security fixes.

Addins for Revit 2025 will need to be rebuilt on .NET 8.

1.3 Add-ins and macro changes

1.3.1 MacroManager API

Updated Revit macro tool with modernized UI and code editor that also supports modern .NET API. Following methods were changed.

Deprecated member(s)	Replacement members(s)

MacroManager.AddModule(ModuleSettings moduleSettings)	MacroManager.AddModule(ModuleSettings moduleSettings, IModuleMaker* maker)
UIMacroManager.AddModule(ModuleSettings moduleSettings, MacroEnvironment environment)	UIMacroManager.AddModule(ModuleSettings moduleSettings, MacroEnvironment environment, IModuleMaker* maker)

The following properties were deprecated:

- ModuleSettings.Description
- MacroModule.MacroLevel
- MacroModule.Description
- Macro.Description
- MacroManage.MacroLevel

The following methods were deprecated:

- MacroModule.AddMacro()
- MacroModule.RemoveMacro()
- MacroManager.GetDocumentMacroSecurityOptions()
- MacroManager.SetDocumentMacroSecurityOptions()
- MacroManager.GetMacroManager()
- MacroManager.IsDocLvlMacro()
- UIMacroManager.EditModule()
- UIMacroManager.EditMacro()
- UIMacroManager.StepInto()
- UIMacroManager.GetMacroManager()
- UIMacroManager.GetUIDocumentMacroSecurityOptions()
- UIMacroManager.SetUIDocumentMacroSecurityOptions()

The following enums were deprecated:

- Deprecated values of enum Autodesk.Revit.DB.Macro.MacroLanguageType, only C# is supported.
    - MacroLanguageType.VBNet
    - MacroLanguageType.Python
- Autodesk.Revit.DB.Macros.MacroLevel
- Autodesk.Revit.DB.Macros.DocumentMacroOptions
- UIDocumentMacroOptions

1.4Array API changes

The valid arrays size differs in project and family documents. The replacement method takes the document as an argument so we get an accurate answer.

Deprecated member(s)	Replacement members(s)
LinearArray.IsValidArraySize()	LinearArray.IsValidNumberOfMembers()
RadialArray.IsValidArraySize()	RadialArray.IsValidNumberOfMembers()

1.5 BRepBuilder API changes

BRepBuilder now accepts HermiteSurface as a permitted support surface type for faces.

Changed behavior:

- BRepBuilder.IsPermittedSurfaceType() will now return true for HermiteSurface.
- BRepBuilderSurfaceGeometry.Create() can now accept a HermiteSurface and use it to create the corresponding BRepBuilderSurfaceGeometry.

1.6 Dimension API changes

After the introduction of LinearDimension class, when filtering for Dimensions, linear dimensions will return as type LinearDimension (child class) instead of Dimension (parent class).

Methods affected:

- typeof(), etc. : If using typeof() or .getType().Equals(), to check if a LinearDimension is Dimension, it will not work due to the nature of the derived classes.

1.7 Extensible Storage(Schema) API changes

Fixed two API(s) for the Extensible Storage feature:

Fixed members(s)	Notes
Document.EraseSchemaAndAllEntities()	Fixed the API to ensure it removes entities from all parts of a Revit model.
ExtensibleStorageFilter()	Fixed the API to ensure it filters all elements associated with extensible storage data based on specific Schema IDs.

1.8 Electrical API changes

1.8.1 Distribution system

The following properties were modified to support high-leg delta and single-phase distribution system.

Deprecated member(s)	Replacement members(s)	Notes
AnalyticalPowerSourceData.Voltage	AnalyticalPowerSourceData.AssignedVoltage	
AnalyticalDistributionNodePropertyData.NumberOfPhases	AnalyticalDistributionNodePropertyData.AssignedPhasesNumber	For analytical power source, bus, transformer,

		transfer switch, the number of electrical phases is read only in Revit 2025 and assigned through the distribution system. For analytical equipment load, we suggest you use the AnalyticalEquipmentLoadData.PhasesNumber property.
AnalyticalBusData.Voltage	AnalyticalBusData.AssignedVoltage	
AnalyticalTransferSwitchData.Voltage	AnalyticalTransferSwitchData.AssignedVoltage	

1.8.2 Parameter Naming

Improved naming for the following electrical parameters to remove ambiguity, improve accuracy, and better adhere with industry terminology.

Parameter/String Attribute	Text(before)	Text(after)
RBS_ELEC_PANEL_TOTALESTLOAD_PARAM	Total Estimated Demand	Total Demand Apparent Power
RBS_ELEC_PANEL_TOTALLOAD_PARAM	Total Connected	Total Connected Apparent Power
RBS_ELEC_APPARENT_LOAD_PHASEA	Apparent Load Phase A	Apparent Power Phase A
RBS_ELEC_APPARENT_LOAD_PHASEB	Apparent Load Phase B	Apparent Power Phase B
RBS_ELEC_APPARENT_LOAD_PHASEC	Apparent Load Phase C	Apparent Power Phase C
RBS_ELEC_DEMAND_LOAD_PHASEA	Demand Load Phase A	Demand Apparent Power Phase A
RBS_ELEC_DEMAND_LOAD_PHASEB	Demand Load Phase B	Demand Apparent Power Phase B
RBS_ELEC_DEMAND_LOAD_PHASEC	Demand Load Phase C	Demand Apparent Power Phase C

Parameter/String Attribute	Text(before)	Text(after)
RBS_ELEC_APPARENT_LOAD	Apparent Load	Apparent Power
RBS_ELEC_APPARENT_LOAD_PHASE1	Apparent Load Phase 1	Apparent Power Phase 1
RBS_ELEC_APPARENT_LOAD_PHASE2	Apparent Load Phase 2	Apparent Power Phase 2
RBS_ELEC_APPARENT_LOAD_PHASE3	Apparent Load Phase 3	Apparent Power Phase 3
RBS_ELEC_TRUE_LOAD	True Load	True Power
RBS_ELEC_TRUE_LOAD_PHASE1	True Load Phase 1	True Power Phase 1
RBS_ELEC_TRUE_LOAD_PHASE2	True Load Phase 2	True Power Phase 2
RBS_ELEC_TRUE_LOAD_PHASE3	True Load Phase 3	True Power Phase 3
RBS_ELEC_TRUE_LOAD_PHASEA	True Load Phase A	True Power Phase A
RBS_ELEC_TRUE_LOAD_PHASEB	True Load Phase B	True Power Phase B
RBS_ELEC_TRUE_LOAD_PHASEC	True Load Phase C	True Power Phase C
RBS_ELEC_DEMANDFACTOR_DEMANDLOAD_PARAM	Estimated Demand Load	Demand Apparent Power
RBS_ELEC_DEMANDFACTOR_LOAD_PARAM	Connected Load	Connected Apparent Power
RBS_ELEC_PANEL_BRANCH_CIRCUIT_APPARENT_LOAD_PHASEA	Branch Circuit Apparent Load Phase A	Branch Circuit Apparent Power Phase A
RBS_ELEC_PANEL_BRANCH_CIRCUIT_APPARENT_LOAD_PHASEB	Branch Circuit Apparent Load Phase B	Branch Circuit Apparent Power Phase B
RBS_ELEC_PANEL_BRANCH_CIRCUIT_APPARENT_LOAD_PHASEC	Branch Circuit Apparent Load Phase C	Branch Circuit Apparent Power Phase C
RBS_ELEC_PANEL_FEED_THRU_LUGS_APPARENT_LOAD_PHASEA	Feed Through Lugs Apparent Load Phase A	Feed Through Lugs

Parameter/String Attribute	Text(before)	Text(after)
		Apparent Power Phase A
RBS_ELEC_PANEL_FEED_THRU_LUGS_APPARENT_LOAD_PHASEB	Feed Through Lugs Apparent Load Phase B	Feed Through Lugs Apparent Power Phase B
RBS_ELEC_PANEL_FEED_THRU_LUGS_APPARENT_LOAD_PHASEC	Feed Through Lugs Apparent Load Phase C	Feed Through Lugs Apparent Power Phase C
RBS_ELEC_PANEL_TOTALESTLOAD_OTHER_PARAM	Other Total Estimated Demand	Other Total Demand Apparent Power
RBS_ELEC_LOADSUMMARY_CONNECTED_LOAD_PARAM	Connected Load (VA)	Connected Apparent Power
RBS_ELEC_LOADSUMMARY_DEMAND_LOAD_PARAM	Estimated Demand (VA)	Demand Apparent Power
RBS_ELEC_PANEL_TOTALESTLOAD_POWER_PARAM	Power Total Estimated Demand	Power Total Demand Apparent Power
RBS_ELEC_PANEL_TOTALESTLOAD_LIGHT_PARAM	Lighting Total Estimated Demand	Lighting Total Demand Apparent Power
RBS_ELEC_PANEL_TOTALESTLOAD_HVAC_PARAM	HVAC Total Estimated Demand	HVAC Total Demand Apparent Power
RBS_ELEC_PANEL_TOTALLOAD_HVAC_PARAM	HVAC Total Connected	HVAC Total Connected Apparent Power
RBS_ELEC_PANEL_TOTALLOAD_LIGHT_PARAM	Lighting Total Connected	Lighting Total Connected Apparent Power
RBS_ELEC_PANEL_TOTALLOAD_POWER_PARAM	Power Total Connected	Power Total Connected

Parameter/String Attribute	Text(before)	Text(after)
		Apparent Power
RBS_ELEC_PANEL_TOTALLOAD_OTHER_PARAM	Other Total Connected	Other Total Connected Apparent Power

1.9 Link Visibility/Graphic Override API changes

The following methods were modified to support custom settings type.

- Revit.DB.RevitLinkGraphicsSettings.LinkVisibilityType
- Revit.DB.View.SetLinkOverrides(ElementId, RevitLinkGraphicsSettings) - Supports RevitLinkGraphicsSettings of Custom type.
- Revit.DB.View.GetLinkOverrides(ElementId) - Removed two validators:
  -
  - AreGraphicsOverridesAllowed()
  - IsLinkOverridesSupported()

1.10 MEP changes

1.10.1 Duct Settings

Two new properties DuctSettings.AirDynamicViscosity and DuctPressureDropData.DynamicViscosity were added to clarify the value used in the duct pressure loss calculation. They replaced the deprecated properties that were kinematic viscosity.

Deprecated members(s)	Replacement member(s)
RbsDuctSettingsElem.AirViscosity	DuctSettings.AirDynamicViscosity
DuctPressureDropData.Viscosity	DuctPressureDropData.DynamicViscosity
DuctFittingAndAccessoryData.AirViscosity	DuctSettings.AirDynamicViscosity

1.11 Reinforcement API changes

1.11.1 Rebar

Deprecated member(s)	Replacement member(s)	Notes
RebarConstraintsManager.SetPreferredConstraintForHandle(Reb	RebarConstraintsManager.SetPreferredConstraint(RebarConstraint constraint)	Use the new replacement method to set RebarConstraint as

arConstrainedHandle handle, RebarConstraint constraint)		preferred constraint target that avoids issues where one could apply the RebarConstraint bounded to handle 'A' to another handle 'B'.
RebarConstraintsManager.GetConstraintCandidatesForHandle(RebarConstrainedHandle handle)	• RebarConstraintsManager.GetConstraintCandidatesForHandle(Structure.RebarConstrainedHandle, ElementId) • RebarConstraintsManager.GetConstraintCandidatesForHandle(Structure.RebarConstrainedHandle, Reference)	• For getting the direct neighbors of a rebar host users may use: Structure.RebarHostData.GetRebarHostDirectNeighbors(Element hostElement). • For getting all the rebars in a host users may use: Structure.RebarHostData.GetRebarsInHost().
Rebar.DistributionType The enum having values: • Uniform 0 - Represents uniform distribution. • VaryingLength 1 - Represents varying length distribution.	RebarShapeDrivenAccessor.UseRebarConstraintsToProduceVaryingBars	

1.12 Slab API changes

The following methods were added for adding split line on SlabShapeEditor.

Deprecated member(s)	Replacement member(s)
SlabShapeEditor.DrawPoint	SlabShapeEditor.AddPoint(XYZ point)
SlabShapeEditor.DrawSplitline	SlabShapeEditor.AddSplitLine(SlabShapeVertex startVertex, SlabShapeVertex endVertex)

1.13 Structure API changes

1.13.1 Bending details in view

The following parameters were renamed without any changes to their functionality.

Deprecated member(s)	Replacement member(s)
BENDING_DETAIL_SEGMENET_REPESENTATION	BENDING_DETAIL_SEGMENT_REPRESENTATION
BENDING_DETAIL_REPESENTATION_FOR_3D_BARS	BENDING_DETAIL_REPRESENTATION_FOR_3D_BARS

1.13.2 Analytical Surface

The following new method was added that allows users to verify if contour is valid for this Analytical Surface.

Deprecated members(s)	Replacement member(s)
AnalyticalSurfaceBase.isCurveLoopValid()	AnalyticalSurfaceBase.IsOuterContourValid()

1.14 Task Dialog API changes

Support added so that FooterText property can contain a hyperlink of the form "rvthelptopic:[topic]" to launch Revit's contextual for the topic specified. Previously, the only option was to specify "#" as the hyperlink and set the Dialog Id, making it difficult to share dialogs with same Id, but different context and associated Help.

1.15 Toposolid API changes

The following parameters were renamed without any changes to their functionality.

Deprecated member(s)	Replacement member(s)
SSE_POINT_EIEVATION_BASE_TYPE	SSE_POINT_ELEVATION_BASE_TYPE
SSE_POINT_EIEVATION	SSE_POINT_ELEVATION
TOPOSOLID_SUBDIVIDE_HEIGNT	TOPOSOLID_SUBDIVIDE_HEIGHT

1.16 Obsolete API removal

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

1.16.1 Classes

- DocumentEntryPoint

1.16.2 Properties

- ViewSchedule.ImageRowHeight
- UIThemeManager.DefaultTheme

1.16.3 Methods

- FamilyManager.AddParameter(ExternalDefinition, BuiltInParameterGroup, bool)
- FamilyManager.AddParameter(string, BuiltInParameterGroup, Category, bool)
- FamilyManager.ReplaceParameter(FamilyParameter, ExternalDefinition, BuiltInParameterGroup, bool)
- FamilyManager.ReplaceParameter(FamilyParameter, string, BuiltInParameterGroup, bool)
- FamilyManager.IsUserAssignableParameterGroup()
- BindingMap.Insert()
- BindingMap.ReInsert()
- FilterNumericLess.Evaluate(int, int)
- FilterNumericLessOrEqual.Evaluate(int, int)
- FilterNumericGreater.Evaluate(int, int)
- FilterNumericGreaterOrEqual.Evaluate(int, int)
- FilterNumericEquals.Evaluate(int, int)
- ParameterFilterRuleFactory.CreateEqualsRule(ElementId, AString, bool)
- ParameterFilterRuleFactory.CreateNotEqualsRule(ElementId, AString, bool)
- ParameterFilterRuleFactory.CreateGreaterRule(ElementId, AString, bool)
- ParameterFilterRuleFactory.CreateGreaterOrEqualRule(ElementId, AString, bool)
- ParameterFilterRuleFactory.CreateLessRule(ElementId, AString, bool)
- ParameterFilterRuleFactory.CreateLessOrEqualRule(ElementId, AString, bool)
- ParameterFilterRuleFactory.CreateContainsRule(ElementId, AString, bool)
- ParameterFilterRuleFactory.CreateNotContainsRule(ElementId, AString, bool)
- ParameterFilterRuleFactory.CreateBeginsWithRule(ElementId, AString, bool)
- ParameterFilterRuleFactory.CreateNotBeginsWithRule(ElementId, AString, bool)
- ParameterFilterRuleFactory.CreateEndsWithRule(ElementId, AString, bool)
- ParameterFilterRuleFactory.CreateNotEndsWithRule(ElementId, AString, bool)

1.16.4 Enums

- Autodesk.Revit.DB.Mechanical.OccupancyUnit - Removed enum value OccupancyUnit.UseDefaultValues
- Autodesk.Revit.DB.BaseLoadOn - Removed enum value BaseLoadOn.kUseDefaultLoad

2 API additions

2.1 Add-ins and macros additions

2.1.1 MacroManager API

The new property:

- MacroModule.ModuleFolder – Returns the folder of the module.

The new interface:

- IModuleMaker

allows users to create new module project by implementing this interface.

The new event:

- MacroUpdated

is raised after the macro modules are updated.

2.2 Analysis API additions

2.2.1 MEP Analytical networks

The new constructor allows users to traverse both sides of the specified analytical segment. In comparison, the existing constructor MEPNetworkIterator(Document, MEPAnalyticalNode, MEPAnalyticalSegment) only traverses one side of the analytical segment.

- Autodesk.Revit.DB.Analysis.MEPNetworkIterator(Document, MEPAnalyticalSegment)

2.2.2 MEP Duct/Pipe Pressure Loss calculation

Users now have access to Duct/Pipe pressure loss characteristic allowing them to correlate the element and understand the calculation logic.

The new classes:

- Autodesk.Revit.DB.Analysis.CriticalPathCollector - Provides the calculated flow and pressure drop values of the network critical path, so users can better select their equipment.
- Autodesk.Revit.DB.Analysis.CriticalPathIterator - Allows users to traverse the analytical segments on the network critical path.

2.2.3 MEP Space Engineering Parameters

Users now have access to building operating day and year schedules allowing them to create and edit schedules that describe the usage of buildings (occupancy, power, lighting) for use in energy analysis.

The new classes:

- Autodesk.Revit.DB.Analysis.BuildingOperatingDaySchedule - An element that represents the 24 hour operating schedule.
- Autodesk.Revit.DB.Analysis.BuildingOperatingYearSchedule - An element that represents the 365 day operating schedule.

The new methods:

- BuildingOperatingDaySchedule.Create(Document, Name) - Allows creation of a new BuildingOperatingDaySchedule with that name and adds it to the document. It returns the newly created BuildingOperatingDaySchedule element, with all usages set to 0.

- BuildingOperatingDaySchedule.GetValueForHour(Hour)
- BuildingOperatingDaySchedule.SetValueForHour(Hour, Usage)
- BuildingOperatingYearSchedule.Create(Document, DaySchedule, Name) - Allows creation of a new BuildingOperatingYearSchedule with that name and adds it to the document. It returns the newly created BuildingOperatingDaySchedule element, with all usages set to daySchedule.
- BuildingOperatingDaySchedule.GetScheduleForDay(Day)
- BuildingOperatingDaySchedule.SetValueForHour(Day, DaySchedule)

The new properties:

- BuildingOperatingDaySchedule.ScheduleName
- BuildingOperatingYearSchedule.ScheduleName

2.3 Annotations API additions

Exposing the new class to fix different behavior of span direction symbol between UI and API.

The new class:

- Autodesk.Revit.DB.SpanDirectionSymbol

represents an instance of a Span Direction Symbol in Autodesk Revit.

2.4 Array API additions

2.4.1 Linear Array

The new methods:

- LinearArray.GetMinimumSize() - Allows users to get the minimum size of a linear array based on if the document is a family document.
- LinearArray.GetNumberOfMembersIncludingPlaceholders() - Allows users to get the number of members in a linear array, including placeholders that are still there in families with small array counts.
- LinearArray.IsValidNumberOfMembers() - Indicates whether the input count is a valid size for an array based on the document.

2.5 MEP Fabrication API additions

The new methods in the FabricationConfiguration class now provides support to check for bad connections between fabrication parts prior to reloading the configuration.

- CheckConnectionsForAllFabricationParts() - Allows users to check the connections for all fabrication parts in the current project. It will create reviewable warnings for all bad connections found.

- ValidateConnectionsForAllFabricationParts() - Allows users to validate all fabrication part connections in the current project. Invalid connections found will be added to the connection validation information class. The validation checks for bad alignments or gaps, incompatible connection types, mismatches of size, mismatches of shapes.
- PostReviewableWarningsForBadConnections() - Reviewable warnings are created for all entries contained in the connection validation information.
- GetUpdatedStraightsFromValidateConnections() - Allows users to get the set of element identifiers of fabrication part straights that were previously updated. If no straights were updated, it will return an empty set of element identifiers.

2.6 DirectShape API additions

The new method:

- DirectShapeType.SetShape(IList<GeometryObject>, DirectShapeTargetViewType) - Allows users to set a custom plan view representation of a DirectShapeType.

2.7 Dimension API additions

Users can now create radial, linear and arc length dimensions via the API in a project document.

The new classes:

- RadialDimension
- ArchLengthDimension
- LinearDimension

The new methods:

- RadialDimension.Create(Document, View, Reference, XYZ origin, bool isDiameterDimension)
- ArcLengthDimension.Create(Document, View, Arc, Reference, Reference firstRef, Reference secondRef)
- LinearDimension.Create(Document, View, Line, IList<Reference>)

2.8 Electrical API additions

Added functionality for high-leg delta and single-phase distribution system. User can now create and modify high-leg delta distribution system and single-phase load in electrical analytical distribution system, and get the per phase current on each node.

The new classes:

- Autodesk.Revit.DB.Electrical.ElectricalPerPhaseData - Represents per phase values including current and load.
- Autodesk.Revit.DB.Electrical.AnalyticalPowerDistributableNodeData - Represents the data and parameters of a power distributable node.
- Autodesk.Revit.DB.Electrical.AnalyticalTransformerData - Represents the data and parameters of analytical transformer node.

The new properties:

- Electrical.DistributionSysType.HighLegPhase - Represents the high-leg phase in the 3 phase 4 wires delta distribution system.
- AnalyticalPowerSourceData.ApparentPowerRating - Represents the apparent power rating value of the analytical power source.
- AnalyticalDistributionNodePropertyData.ConnectedPhases - Represents the electrical connected phases of the electrical analytical node to its upstream node.
- AnalyticalEquipmentLoadData.PhasesNumber - Represents the number of electrical phases of the analytical equipment load.
- AnalyticalEquipmentLoadData.PowerFactorState - Represents the PowerFactorState type of the analytical equipment load.
- AreaBasedLoadData.ConnectedPhases - Represents the electrical connected phases of the area based load to its upstream node.
- AreaBasedLoadData.PhasesNumber - Allows users to set the number of electrical phases of the area based load.
- AreaBasedLoadData.PowerFactorState - Represents the power factor state of the area based load.
- AreaBasedLoadType.PowerFactorState - Represents the power factor state of the area based load type.

The new enums:

- Autodesk.Revit.DB.Electrical.ElectricalPhaseLine - Defines the electrical phase.
- Autodesk.Revit.DB.Electrical.ElectricalConnectedPhases - Defines the electrical connected phases of an electrical analytical node.

2.9 Energy Analysis API additions

2.9.1 gbXML export options

We now provide support for legacy export of gbXML based on a conceptual energy analytical model from a conceptual mass model from add-ins/add-ons.

The new methods:

- GetMassIds() - Allows users to get a list of masses to use as shading surfaces in the exported gbXML.
- GetMassZoneIds() - Allows users to get a list of mass zones to analyze in the exported gbXML.

The new constructor:

- GBXMLExportOptions(massZoneIds, massIds) - Allows users to construct a new instance of the options used to export a mass model to gbXML.

The new property:

- IsConceptual - Indicates if the exported gbXML is based on a conceptual energy analytical model from a conceptual mass model.

2.10 IFC API additions

2.10.1 IFC Hybrid Import

- IFCHybridImport.UpdateElements() - The new method allows users to update elements previously imported via a Hybrid IFC Link operation.
- IFCHybridImport.GetIFCStepIdToElementIdMap() - The new method allows users to retrieve the association between the original IFC STEP identifiers and the created or updated Elements.

2.10.2 IFC Category Mapping

Added functionality that allows user to control the way how Revit categories are mapped to IFC Classes during IFC Export. User can now create, retrieve and modify IFC category mapping templates.

The new classes:

- Autodesk.Revit.DB.IFCCategoryTemplate - Represents an element that contains IFC category mapping template stored in a Revit document.
- Autodesk.Revit.DB.ExportIFCCategoryKey - Represents a Revit category item stored in a template.
- Autodesk.Revit.DB.ExportIFCCategoryInfo - Represents the mapped IFC information stored in the template.

The new enums:

- CustomSubCategoryId - Represents pseudo sub-categories that can appear in a mapping template. It has the following values:
  - None - Represents the default value for most Revit categories and subcategories.
  - InteriorWall - Represents the custom id for interior walls.
  - ExteriorWall - Represents the custom id for exterior walls.
  - FoundationWall - Represents the custom id for foundation walls.
  - RetainingWall - Represents the custom id for retaining walls.
  - Coreshaft - Represents the custom id for cores/shafts.
  - Soffit - Represents the custom id for soffits.

2.11 Import Export API additions

The new functionality provides support for importing, linking and exporting files of STEP format.

The new class:

- STEPImportOptions

represents the options for STEP formats.

The new methods, uses the new class STEPImportOptions:

- Document.Import(String, STEPImportOptions, View)
- Document.Link(String, STEPImportOptions, View)
- Document.Export(String, String, STEPExportOptions)

2.12 PDF Export API additions

Document export for PDF now allows using a separate Revit Worker to create the PDF in the background, leaving the main Revit process free for other work. FileExporting and FileExported events are triggered at the start and end of the export job respectively. There are new API calls for PDFExportOptions, FileExportedEventArgs, and FileExportingEventArgs. Third party Addins may interfere with the PDF generation using this feature if they change the appearance of Elements in a way that is not serialized.

The new method:

- PDFExportOptions.SetExportInBackground() - When set to true, Document.Export launches a new process to export the PDF that leaves Revit unblocked. Changes to the document made after the export has started are not accounted for in the PDF.

The classes FileExportingEventArgs and FileExportedEventArgsBoth have a new member:

- BackgroundOperation - The value is true if a background process was requested for the background operation that raised the event.

2.13 Reinforcement API additions

2.13.1 Rebar

Users now have the ability to create and modify rebar constraints to surfaces.

The new enum:

- RebarConstraintsStatus

represents the status of the constraints.

The new value for RebarConstraintType enum:

- ToSurface

represents handle is constrained to a surface.

The new methods:

- RebarConstraint.CreateConstraintToSurface() - Allows users to create a constraint of ToSurface type for a given RebarConstrainedHandle.
- RebarConstraint.GetSurfaceForConstraintToSurface() - Allows users to retrieve the surface for a constraint of ToSurface type.
- RebarConstraint.IsValidSurfaceToConstraintHandleTo() - Allows users to check if the surface can be used to create a constraint to it for a given RebarConstrainedhandle.
- RebarConstraint.IsToSurface() - Returns true if the RebarConstraintType of the RebarConstraint is ToSurface, false otherwise.
- RebarConstraint.GetRebarConstrainedHandle() - Allows users to get the RebarConstrainedHandle for which this constraint is.
- RebarConstraint.FlipSideForClearBarSpacingZeroDistanceConstraint() - This method in RebarConstriant is applied for bar-on-bar constraint. It flips the side on which a RebarConstrainedHandle constrained using clear bar distance with zero offset connects to another Rebar target handle.
- RebarConstrainedHandle.IsEqual() - Returns true if the objects are equal, false otherwise.
- RebarConstraintsManager.SetPreferredConstraintsToSurfaceForHandles() - For ShapeDriven rebar it will set a preferred 'ToSurface' RebarConstraint for each input handle. The surface that will be used by the constraint is the current surface that is used to compute the position of the handle. This function applies only for shape driven Rebar, and will throw exception for free form rebar.
- RebarConstraintsManager.SetPreferredConstraint() - Allows users to set the RebarConstraint as preferred constraint for its RebarConstrainedHandle.
- RebarConstraintsManager.GetAutomaticConstraintCandidatesForHandle() - For shape driven rebar returns all possible automatic RebarConstraints that could be used for a specified RebarConstrainedHandle. For free form rebar it returns an empty list.

2.13.2 Rebar splice type options and rules

Added functionality for splicing the rebar. Users can now splice a rebar, remove the splice (keeping the bars separated), unify into one bar, modify the data related to splice and modify the constraints of the spliced bars seeing as splice chain.

The new classes:

- Autodesk.Revit.DB.Structure.RebarSpliceTypeUtils - Utility class for dealing with rebar splice type operations.
- Autodesk.Revit.DB.Structure.RebarSpliceOptions
- Autodesk.Revit.DB.Structure.RebarSpliceGeometry - This class consists of a vector and a point which will be projected to the nearest rebar curve.
- Autodesk.Revit.DB.Structure.RebarSpliceRules
- Autodesk.Revit.DB.Structure.RebarSplice - A class that can used to access the data between two connected rebars.
- Autodesk.Revit.DB.Structure.RebarSpliceByRulesResult
- Autodesk.Revit.DB.Structure.RebarSpliceUtils

The new methods:

- RebarBarType.GetLapLength()
- RebarBarType.SetLapLength()
- RebarBarType.GetAutoCalculatedLapLength()
- RebarBarType.SetAutoCalculatedLapLength()
- RebarBarType.GetStaggerLength)
- RebarBarType.SetStaggerLength()
- RebarBarType.GetAutoCalculatedStaggerLength()
- RebarBarType.SetAutoCalculatedStaggerLength()
- Rebar.GetRebarSplice()
- Rebar.RemoveSplice()
- Rebar.GetLapLength()
- Rebar.GetSpliceStaggerLength()
- RebarConstrainedHandle.GetHandleSurface()
- RebarConstrainedHandle.Move()
- RebarConstrainedHandle.CanSetBehavior()
- RebarConstrainedHandle.GetPossibleHandleBehaviors()
- RebarHostData.GetRebarHostDirectNeighbors() - Returns the first level of neighbors for the provided host that can host reinforcement, i.e. elements that are joined directly to the provided host element and not the neighbors of the joined elements.

The new properties:

- RebarSpliceOptions.SpliceTypeId
- RebarSpliceOptions.SplicePosition
- Rebar.CanHaveVaryingLengthBars
- Rebar.HasVariableLengthBars
- RebarShapeDrivenAccessor.UseRebarConstraintsToProduceVaryingBars
- RebarConstrainedHandle.HandleBehavior

The new enums:

- RebarSplicePosition - Describes the position of the splice. It has the following values:
  - End1 - Lap is towards the start of the splice chain.
  - Middle - Lap goes into both directions.
  - End2 - Lap is towards the end of the splice chain.

- RebarSpliceShiftOption - Describes the way bars are shifted in the splice relation. It has the following values:
  - BarPlane - Represents the bar plane is shifted so that the spliced rebars are not clashing.
  - None - The bars are not shifted at all.

- RebarSpliceByRulesRunOutPosition - Describes the run-out position. It has the following values:
  - Start - Represents the rest will remain at the start of the bar.
  - End - Represents the rest will remain at the end of the bar

- RebarSpliceError - Represents the states for splicing a Rebar. It has the following values:
  - Success
  - Unknown - Represents there is an unexpected error.
  - InvalidRebar - Represents free form rebars or shape driven rebars that are multiplanar or having shape whose definition is RebarShapeDefinitionByArc or rebars part of a group that cannot be spliced.
  - InvalidLineOrLinePlaneNormal - Represents the line length is zero or the line direction is parallel with the line plane normal.

    - LineDoesNotIntersectRebarBoundingBox - Represents the line doesn't intersect the rebar bounding box.
    - SpliceGeometryOnHookOrFillet - Represents if the splice geometry is on a hook or a fillet, the rebar can't be spliced with it.
    - TooSmallSegments - Represents one of the resulting segments is too small to apply the lap.
    - SpliceGeometryDoesNotIntersectAllTheBarsInTheSet - Represents a plane obtained from splice geometry doesn't intersect all the bars in the set.
    - SpliceGeometryAlmostParallelToBarSegment - Represents the plane formed by splice geometry is almost parallel to bar segment plane.

- RebarSpliceByRulesError - Represents the states for splicing a Rebar by rules. It has the following values:
    - Success
    - Unknown - Represents there is an unexpected error.
    - InvalidRebar - Represents free form rebars or shape driven rebars that are multiplanar or having shape whose definition is RebarShapeDefinitionByArc or rebars part of a group that cannot be spliced.
    - TooBigHook - Represents the hook lengths exceed the maximum length.
    - TooSmallRunOut - Represents the run-out is below minimum length, or the lap can't be applied to it.
    - MaximumLengthBiggerThanBarLength - Represents the maximum length exceeds the bar length.
    - TooBigArc - Represents the arc segment exceeds the maximum length.
    - CantSpliceAllTheBarsInSet - Represents some bars in the varying set are not intersected by the resulting splice geometries.
    - LapLengthBiggerThanMaximumBarLength - Represents the lap length is greater than the maximum length.
    - InvalidCombinationOfMaximumMinimumBarLengthAndLapLength - Represents the combination of the maximum bar length, minimum bar length and lap length is invalid.

- RebarHandleBehavior - Represents different behaviors that can be applied to a RebarConstrainedHandle. It has the following values:
    - Default - Represents the default behavior of a RebarConstrainedHandle.
    - SpliceMainEndOnEnd1Position
    - SpliceMainEndOnMiddlePosition - Represents the behavior can be set to a StartOfBar or EndOfBar RebarConstrainedHandle of a bar that is part of splice. On the connected bar there is a ToOtherRebar constraint whose target is the current rebar. The RebarConstrainedHandle's plane in the same position as the splice plane for Middle.
    - SpliceMainEndOnEnd2Position - Represents the behavior can be set to a StartOfBar or EndOfBar RebarConstrainedHandle of a bar that is part of splice. On the connected bar there is a ToOtherRebar constraint whose target is the current rebar. The RebarConstrainedHandle's plane in the same position as the splice plane for Middle
    - SpliceConnectedEndOnEnd1Position - Represents the behavior can be set to a StartOfBar or EndOfBar RebarConstrainedHandle of a bar that is part of splice. On this RebarConstrainedHandle is a ToOtherRebar constraint whose target is the other bar involved in splice. The RebarConstrainedHandle's plane in the same position as the splice plane for End1.
    - SpliceConnectedEndOnMiddlePosition - Represents the behavior can be set to a StartOfBar or EndOfBar RebarConstrainedHandle of a bar that is part of splice. On this RebarConstrainedHandle is a ToOtherRebar constraint whose target is the other bar involved in splice. The RebarConstrainedHandle's plane in the same position as the splice plane for Middle
    - SpliceConnectedEndOnEnd2Position - Represents the behavior can be set to a StartOfBar or EndOfBar RebarConstrainedHandle of a bar that is part of splice. On this

RebarConstrainedHandle is a ToOtherRebar constraint whose target is the other bar involved in splice. RebarConstrainedHandle's plane in the same position as the splice plane for End2
    - SpliceRebarPlaneOnSpliceSetExtent - Represents the behavior can be set to a RebarPlane RebarConstrainedHandle of a bar that is part of splice
    - SpliceOutOfPlaneExtentOnSpliceSetExtent - Represents the behavior can be set to a OutOfPlaneExtent RebarConstrainedHandle of a bar that is part of splice.
    - SpliceEdge - Represents the behavior can be set to an edge segment that is connected to the other rebar of splice.

2.14 Selection API additions

2.14.1 UI Application

The new enum:

- ThemeType - Represents the application frame theme type. It has the following values:
    - CanvasTheme - Indicates that the ThemeChanged event is triggered by canvas theme change.
    - UITheme - Indicates that the ThemeChanged event is triggered by the global UI theme change.

The new read-only property for ThemeChangedEventArgs:

- ThemeChangedType - Returns ThemeType enum that indicates the kind of change triggered the current event.

2.15 Sketched Element API additions

2.15.1 Wall APIs

The new methods now allow users to selectively enable or disable end wrapping for a specific wall end when the wall's end wrap is activated in the Wall Type dialog.

- Revit.DB.Wall.GetWrappingLocationAsReferences(int locationIndex) - Allows users to get an array of references to faces at the location.
- Revit.DB.Wall.GetWrappingLocationAsCurveParameter(int locationIndex) - Allows users to get the non-normalized (actual) curve parameter of the location.
- Revit.DB.Wall.GetValidWrappingLocationIndices()
- Revit.DB.Wall.AllowWrappingAtLocation(int locationIndex)
- Revit.DB.Wall.DisallowWrappingAtLocation(int locationIndex)
- Revit.DB.Wall.IsWrappingAtLocationAllowed(int locationIndex)

2.16 Structure API additions

2.16.1 Analytical Elements

The new methods:

- AnalyticalElement.IsValidTransform() - Allows users to check whether the value set for Local Coordinate System is valid for an Analytical Element.
- AnalyticalElement.SetTransform() - Allows users to set the transform of Analytical Element Local Coordinate System.

2.16.2 Analytical Surface

The new method:

- AnalyticalSurfaceBase.IsOuterContourValid() - Allows users to verify if contour is valid for this Analytical Surface.

2.16.3 Bending Details on Drawings

Added new functionality for enhancing bending details. These include the ability to create and customize the size of schematic bending detail boxes, set specific tag types for bending details, adjust the position and rotation of tags relative to the bending detail, customize the tag alignment option, and add multiple hosts for the schematic bending detail. This update offers greater flexibility and precision in the representation and annotation of bending details in Revit models.

The new methods:

- RebarBendingDetail.AddHosts()
- RebarBendingDetail.GetHosts()
- RebarBendingDetail.RemoveHosts()
- RebarBendingDetail.SetTagRelativePosition()
- RebarBendingDetail.SetTagRelativeRotation()
- RebarBendingDetail.GetTagRelativePosition()
- RebarBendingDetail.GetTagRelativeRotation()
- RebarBendingDetail.IsSchematicBendingDetail()
- RebarBendingDetail.IsRealisticBendingDetail()
- RebarBendingDetail.ResetTagRelativePosition()
- RebarBendingDetail.ResetAnnotationPositions()

The new properties:

- RebarBendingDetailType.SchematicHeight - Represents the height of the box where the schematic Bending Detail will be represented.
- RebarBendingDetailType.SchematicWidth - Represents the width of the box where the schematic Bending Detail will be represented.
- RebarBendingDetailType.TagTypeId - Represents the Id of the tag type which is used in the Bending Detail representation.
- RebarBendingDetailType.DisplayMode - Represents the display mode for the bending detail.

The new enum:

- Autodesk.Revit.DB.Structure.BendingDetailDisplayMode

represents the mode in which the Bending Detail will be represented.

2.16.4 Radial Array

The new methods:

- RadialArray.GetMinimumSize() - Allows users to get the minimum size of a radial array based on if the document is a family document.
- RadialArray.GetNumberOfMembersIncludingPlaceholders() - Allows users to get the number of members in a radial array, including placeholders that are still there in families with small array counts.
- RadialArray.IsValidNumberOfMembers() - Indicates whether the input count is a valid size for an array based on the document.

2.17 Tag/Keynotes API additions

We now provide functionality to align multiple text, tags and keynotes with new contextual alignment tools in the ribbon.

The new class:

- Autodesk.Revit.DB.AnnotationMultipleAlignmentUtils

allows users to align annotation elements to one another. Currently supports TextNotes, Tags and Keynotes.

The new methods:

- AnnotationMultipleAlignmentUtils.ElementSupportsMultiAlign() - allows users to check whether the element type can be aligned using the multiple allignment commands.
- AnnotationMultipleAlignmentUtils.GetAnnotationOutlineWithoutLeaders() - allows users to get the four corners of the annotations bounding box, not including leaders. Outline calculations include leader/border offsets wherever applicable. (Eg. in the case of TextNotes).
- AnnotationMultipleAlignmentUtils.MoveWithAnchoredLeaders() - allows users to move the given element to the position specified by the input moveVec, while keeping the leader end points anchored.

2.18 Toposolid API additions

Added functionality for Toposolid smooth shading.

The new methods:

- Toposolid.ExcavateBy() - Allows users to excavate toposolid by a given element.

- Toposolid.RemoveExcavationBy() - Allows users to remove the excavation between the given element and the toposolid.
- Toposolid.CanBeExcavatedBy() - Allows users to check if the given element can be used to excavate the toposolid.
- Toposolid.SetSmoothedSurface() - Allows users to set the smoothed surface setting of toposolid category in the given document.
- Toposolid.IsSmoothedSurfaceEnabled() - Allows users to check if smoothed surface setting of toposolid category is enabled in the given document.
- Toposolid.GetIntersectingElementData()
- ToposolidType.SetContourSetting() - Allows users to set the contour setting for the current toposolid type by copying from an existing contour setting object.
- ContourSetting.IsItemEnabled()
- ContourSettingItem.GetContourSettingItemType()
- FaceToposolid.Create()
- FaceToposolid.UpdateToFace() - Allows users to reset the face toposolid to its defining face.
- FaceToposolid.GetReferencedFaces()
- FaceToposolid.SetReferencedFaces()

The new classes:

- Autodesk.Revit.DB.IntersectingElementData - Stores information of an element that intersects with another element.
- Autodesk.Revit.DB.FaceToposolid - represents a face-based Toposolid within the Autodesk Revit project.

The new enum:

- Autodesk.Revit.DB.IntersectionType - It has the following values.
  - Cut
  - Excavate
- Autodesk.Revit.DB.ContourSettingItemType - Represents the type of contour setting item. It has the following values:
  - Single
  - UnboudedRange
  - BoundedRange.

The new properties:

- IntersectingElementData.IntersectionType
- IntersectingElementData.IntersectingElementId
- IntersectingElementData.IntersectedElementId
- IntersectingElementData.IntersectionVolume

2.19 UI API additions

2.19.1 Context Menu

We now provide functionality to create context menu from an add-in.

The new Interface:

- Revit.UI.IContextMenuCreator - The interface users need to implement to build context menu.

The new Constructor:

- Revit.UI.ContextMenu() - Allows users to create a new instance of Context Menu.
- Revit.UI.CommandMenuItem(name , className, assemblyName) - Allows users to create a new instance of command menu item with name, external command class name and external application assembly name.
- Revit.UI.SubMenuItem(name , ContextMenu) - Allows users to create a new instance of flyout menu with name and sub menu instance.
- Revit.UI.Separator() - Allows users to create a new instance of separator menu item.

The new methods:

- ContextMenu.AddItem() - Allows users to add a specific type of MenuItem object to context menu.
- CommandMenuItem.SetAvailabilityClassName() - Allows users to set availabilityClassName of CommandMenuItem.
- CommandMenuItem.SetToolTip() - Allows users to set tooltip of CommandMenuItem.
- IContextMenuCreator.BuildContextMenu() - Allows users to add menu items to the passed in ContextMenu object.
- UIControlledApplication.RegisterContextMenu() - Allow user to add new context menu creators with application name.

2.20 View API additions

2.20.1 SheetCollection

The new class:

- Autodesk.Revit.DB.SheetCollection - Represents a sheet collection in Autodesk Revit.

The new methods:

- Autodesk.Revit.DB.SheetCollection.Create(document, name) - Allows users to create a new instance of sheet collection with a specified name and adds it to the document. It returns the newly created sheet collection element.
- Autodesk.Revit.DB.SheetCollection.Create(document) - Allows users to create a new instance of sheet collection with an auto-generated name and adds it to the document. It returns the newly created sheet collection element.

The new property:

- Autodesk.Revit.DB.ViewSheet.SheetCollectionId - Represents the Id of the sheet collection this sheet is associated with.

2.21 Link Visibility/Graphic Override API additions

The new functionality allows Revit Link Visibility/Graphic Overrides, for the 'Custom' option.

The new methods in class Autodesk.Revit.DB.RevitLinkGraphicsSettings:

- RevitLinkGraphicsSettings.IsViewRangeSupported(View) – Allows users to check if the input view supports view range settings for RevitLinkGraphicsSettings graphic overrides.
- RevitLinkGraphicsSettings.GetPhaseId()
- RevitLinkGraphicsSettings.GetPhaseType()
- RevitLinkGraphicsSettings.SetPhase(LinkVisibility, ElementId) – Allows users to configure phase and phase type of RevitLinkGraphicsSettings. Accepts LinkVisibility and ElementId of the phase from the linked document or ElementId.InvalidElementId.
- RevitLinkGraphicsSettings.GetPhaseFilterId()
- RevitLinkGraphicsSettings.GetPhaseFilterType()
- RevitLinkGraphicsSettings.SetPhaseFilter(LinkVisibility, ElementId) – Allows users to configure phase filter and phase filter type of RevitLinkGraphicsSettings. Accepts LinkVisibility and ElementId of the phase filter from the linked document or ElementId.InvalidElementId.
- RevitLinkGraphicsSettings.GetViewDetailLevel()
- RevitLinkGraphicsSettings.GetViewDetailLevelType()
- RevitLinkGraphicsSettings.SetViewDetailLevel(LinkVisibility, ViewDetailLevel) – Allows users to configure detail level and detail level type of RevitLinkGraphicsSettings. Accepts LinkVisibility and ViewDetailLevel types.
- RevitLinkGraphicsSettings.GetDiscipline()
- RevitLinkGraphicsSettings.GetDisciplineType()
- RevitLinkGraphicsSettings.SetDiscipline(LinkVisibility, ViewDiscipline) – Allows users to configure discipline and discipline type of RevitLinkGraphicsSettings. Accepts LinkVisibility and ViewDiscipline types.

The new properties in Autodesk.Revit.DB.RevitLinkGraphicsSettings:

- RevitLinkGraphicsSettings.ViewFilterType
- RevitLinkGraphicsSettings.ViewRange
- RevitLinkGraphicsSettings.ColorFill
- RevitLinkGraphicsSettings.ObjectStyles
- RevitLinkGraphicsSettings.NestedLinks

2.22 RevitServer Enterprise / Revit Cloud Worksharing API additions

The following events have been supported for file-based worksharing since 2021. We now support them in RevitServer Enterprise/Revit Cloud Worksharing.

- DocumentReloadingLatest - Subscribe to the DocumentReloadingLatestEventArgs event to be notified when Revit is just about to reload latest changes from a central model.
- DocumentReloadedLatest - Subscribe to the DocumentReloadedLatestEventArgs event to be notified immediately after Revit has finished reloading a document with central model.