Major additions to the Revit 2022.1 API

API additions

View API additions

Duplicating Sheets

The new enum:

- SheetDuplicateOption

allows you to indicate what information should be copied when duplicating a sheet. Its values are:

- DuplicateEmptySheet - Only copies the title block.
- DuplicateSheetWithDetailing - Copies the title block and details.
- DuplicateSheetWithViewsOnly - Copies the title block, details, viewports and contained views. The newly created sheet will reference the newly duplicated views.
- DuplicateSheetWithViewsAndDetailing - Copies the title block, details, and viewports. Duplicates the sheet's contained views with detailing. The newly created sheet will reference the newly duplicated views.
- DuplicateSheetWithViewsAsDependent - Copies the title block, details, and viewports. Duplicates the sheet's contained views as dependent. The newly created sheet will reference the newly duplicated dependent views.

The new methods:

- ViewSheet.Duplicate(SheetDuplicateOption)
- ViewSheet.CanBeDuplicated(SheetDuplicateOption)

allow you to duplicate sheets and identify sheets which can be duplicated.

Schedule API additions

Schedule heights on sheets

The new class:

- Autodesk.Revit.DB.ScheduleHeightsOnSheet

returns the heights of schedule title, column header and each body row on sheet view.

The new method:

- ViewSchedule.GetScheduleHeightsOnSheet()

will return the heights object.

Managing schedule segments

The new methods:

- ViewSchedule.IsSplit()
- ViewSchedule.Split(int segmentNumber)
- ViewSchedule.Split(IList<double> segmentHeights)
- ViewSchedule.SplitSegment()
- ViewSchedule.DeleteSegment()
- ViewSchedule.MergeSegments()
- ViewSchedule.GetSegmentCount()
- ViewSchedule.GetSegmentHeight()
- ViewSchedule.SetSegmentHeight()

provide the ability to split schedules and manage schedule segments.

The new method:

- ViewSchedule.GetScheduleInstances()

will return the schedule sheet instances for a schedule segment.

The new methods:

- ScheduleSheetInstance.SegmentIndex
- ScheduleSheetInstance.Create(Document document, ElementId viewSheetId, ElementId scheduleId, XYZ origin, int segmentIndex)

provide the ability to place a schedule segment on sheet and to get and set the schedule segment instance's segment index.

Worksharing API additions

Delete Workset API

The new method:

- WorksetTable.DeleteWorkset()

supports deleting of worksets from the model. It takes a DeleteWorksetSettings input with options for what to do with elements contained by that workset.

Import and Export API additions

AXM Import

The new class:

- Autodesk.Revit.DB.AXMImportOptions

allows user to determine the import options when importing an AXM file.

The new method:

- Document.Import(String, AXMImportOptions, View)

imports an AXM file into the document.

The new method:

- OptionalFunctionalityUtils.IsAXMImportLinkAvailable()

checks if the Import FormIt function is available.

OBJ & STL import/export

The new method:

- Document.Export(String, String, OBJExportOptions)

supports export of Revit geometry to OBJ format. It uses a new class containing the options available for export:

- OBJExportOptions.TargetUnit
- OBJExportOptions.SurfaceTolerance
- OBJExportOptions.NormalTolerance
- OBJExportOptions.MaxEdgeLength
- OBJExportOptions.GridAspectRatio
- OBJExportOptions.SetTessellationSettings()

The new methods:

- Document.Import(String, OBJImportOptions, View)
- Document.Import(String, STLImportOptions, View)

provide support for importing files of STL and OBJ formats. These methods use new classes representing the options for each of the new formats.

The new property:

- BaseImportOptions.DefaultLengthUnit

supports specification of a default length unit to use during import of unitless STL and OBJ files.

Major changes and renovations to the Revit 2022 API

API changes

Parameter API changes (migration to ForgeTypeId)

In Revit 2021, Revit API functionality for units of measurement migrated from enumerated identifiers to extensible identifiers represented by the ForgeTypeId class. ForgeTypeId is now used as the identifier type for more data structures in the Revit API.

A ForgeTypeId instance holds a string, called a "typeid", that uniquely identifies a Forge schema. A Forge schema is a JSON document describing a data structure, supporting data interchange between applications. A typeid string includes a namespace and version number and may look something like "autodesk.spec.aec:length-1.0.0" or "autodesk.unit.unit:meters-1.0.0". By default, comparison of ForgeTypeId values in the Revit API ignores the version number.

The new classes:

- Autodesk.Revit.DB.DisciplineTypeId
- Autodesk.Revit.DB.GroupTypeId
- Autodesk.Revit.DB.ParameterTypeId

contain properties of type ForgeTypeId, following the pattern established in Revit 2021 with the UnitTypeId, SymbolTypeId, and SpecTypeId classes. Values from the DisciplineTypeId class can be used in code to replace values of the deprecated UnitGroup enumeration. For example, where you previously used UnitGroup.Structural, you would now use DisciplineTypeId.Structural. Values from the GroupTypeId and ParameterTypeId classes can be used to replace values of the BuiltInParameter and BuiltInParameterGroup enumerations, respectively, which have not yet been deprecated.

The SpecTypeId class contains several new nested classes containing ForgeTypeId properties identifying non-floating-point data types, such as integers and strings. ForgeTypeId properties in SpecTypeId and its nested classes can be used to replace values of the deprecated ParameterType enumeration. These tables list the deprecated types and their replacements, as well as some migrated types where the original enumeration is not yet fully deprecated:

Deprecated enumeration	**Replacement**
UnitGroup	DisciplineTypeId

ParameterType	SpecTypeId SpecTypeId.Boolean SpecTypeId.Int SpecTypeId.Reference SpecTypeId.String

Original enumeration	Preferred replacements as ForgeTypeId
BuiltInParameterGroup	GroupTypeId
BuiltInParameter	ParameterTypeId

The mapping of enumerations to different ForgeTypeId members has implications for previously existing APIs related to parameter access, parameter definitions, shared parameter creation and binding, global parameters and other utilities.

Changes to Parameter Access APIs

Original member	Preferred replacement
Document.TypeOfStorage[BuiltInParameter]	Document.GetTypeOfStorage(ForgeTypeId)
Element.Parameter[BuiltInParameter]	Element.GetParameter(ForgeTypeId)
Parameter.Id	Parameter.GetTypeId()

Changes to Parameter Definition APIs

In the Definition class, the ParameterType property and the GetSpecTypeId() method are both deprecated in favor of the new method GetDataType(). GetDataType() is more general than the functions it replaces, providing a data type identifier for any parameter.

- For parameters associated with units of measurement, GetDataType() returns a measurable spec identifier. Use UnitUtils.IsMeasurableSpec(ForgeTypeId) to detect a measurable spec identifier.
- For Family Type parameters, GetDataType() returns a category identifier. Use Category.IsBuiltInCategory(ForgeTypeId) to detect a category identifier.
- For all other parameters, GetDataType() returns a spec identifier. Use Parameter.IsSpec(ForgeTypeId) to detect a spec identifier, including measurable specs.

Use Parameter.IsValidDataType(ForgeTypeId) to detect any parameter data type identifier – i.e., any measurable or non-measurable spec or category.

Deprecated member	Replacement member
Definition.GetSpecTypeId() Definition.ParameterType	Definition.GetDataType()

Original member	Preferred replacement
Definition.ParameterGroup	Definition.GetGroupTypeId()
InternalDefinition.BuiltInParameter	InternalDefinition.GetParameterTypeId()
InternalDefinition.ParameterGroup	InternalDefinition.GetGroupTypeId()
InternalDefinition.ParameterGroup	InternalDefinition.SetGroupTypeId(ForgeTypeId)
InternalDefinition.Id	InternalDefinition.GetTypeId()

Changes to Shared Parameter Creation and Binding APIs

In the ExternalDefinitionCreationOptions class, used to create shared parameters, the implementation now offers better input validation and new capabilities.

The deprecated class constructor and Type property setter taking ParameterType values now throw an exception for input values ParameterType.Invalid or ParameterType.FamilyType. It was not possible to use this interface to create Family Type shared parameters.

However, it is now possible to create Family Type shared parameters using the replacement functions based on ForgeTypeId. The class constructor taking ForgeTypeId and the SetDataType(ForgeTypeId) method accept any spec or category identifier. Using a category identifier produces a Family Type shared parameter of the given category. Use Category.GetBuiltInCategoryTypeId(BuiltInCategory) to obtain the ForgeTypeId identifier for a given category.

Use Parameter.IsValidDataType(ForgeTypeId) to detect any parameter data type identifier – i.e., any measurable or non-measurable spec or category.

Deprecated member	Replacer
ExternalDefinitionCreationOptions(string, ParameterType)	ExternalDefinitionCreationOptions(string
ExternalDefinitionCreationOptions.Type	ExternalDefinitionCreationOptions.GetDa ExternalDefinitionCreationOptions.SetDa

Original member	Preferred replacement
BindingMap.Insert(Definition, Binding, BuiltInParameterGroup)	BindingMap.Insert(Definition, Binding, ForgeTypeId)
BindingMap.ReInsert(Definition, Binding, BuiltInParameterGroup)	BindingMap.ReInsert(Definition, Binding, ForgeTypeId)

Shared parameters can also be created and leveraged in families using members of the FamilyManager class.

Deprecated member	Replacement member

FamilyManager.AddParameter(string, BuiltInParameterGroup, ParameterType, bool)	FamilyManager.AddParameter(string, ForgeTypeId, ForgeTypeId, bool)

Original member	Preferred replacement
FamilyManager.AddParameter(ExternalDefinition, BuiltInParameterGroup, bool)	FamilyManager.AddParameter(ExternalDefinition, ForgeTypeId, bool)
FamilyManager.AddParameter(string, BuiltInParameterGroup, Category, bool)	FamilyManager.AddParameter(string, ForgeTypeId, Category, bool)
FamilyManager.IsUserAssignableParameterGroup(BuiltInParameterGroup)	FamilyManager.IsUserAssignableParameterGroup(ForgeTypeId)
FamilyManager.Parameter[BuiltInParameter]	FamilyManager.GetParameter(ForgeTypeId)
FamilyManager.ReplaceParameter(FamilyParameter, ExternalDefinition, BuiltInParameterGroup, bool)	FamilyManager.ReplaceParameter(FamilyParameter, ExternalDefinition, ForgeTypeId, bool)
FamilyManager.ReplaceParameter(FamilyParameter, string, BuiltInParameterGroup, bool)	FamilyManager.ReplaceParameter(FamilyParameter, string, ForgeTypeId, bool)

Changes to GlobalParameter APIs

Deprecated member	Replacement member
GlobalParameter.Create(Document, string, ParameterType)	GlobalParameter.Create(Document, string, ForgeTypeId)
GlobalParameter.IsValidDataType(ParameterType)	Parameter.IsSpec(ForgeTypeId)

New and changed utility APIs

The LabelUtils class, used to obtain user-visible names for enumerations, has several changes according to the mapping from enumerations to ForgeTypeId. In addition to the replacements listed below, a new method:

- LabelUtils.GetLabelForDiscipline(ForgeTypeId)

returns the user visible name corresponding to a Revit discipline in the current Revit language. The implementation of the method:

- LabelUtils.GetLabelForSpec(ForgeTypeId)

has been expanded to accept a ForgeTypeId argument identifying any parameter data type. For a category identifier input, GetLabelForSpec(ForgeTypeId) returns the label of the Family Type spec for that category, e.g. "Family Type: Wall Tags".

Deprecated member	Replacement member
LabelUtils.GetLabelFor(ParameterType)	LabelUtils.GetLabelForSpec(ForgeTypeId)

Original member	Preferred replacement
LabelUtils.GetLabelFor(BuiltInParameter)	LabelUtils.GetLabelForBuiltInParameter(ForgeTypeId)
LabelUtils.GetLabelFor(BuiltInParameter, LanguageType)	LabelUtils.GetLabelForBuiltInParameter(ForgeTypeId, LanguageType)
LabelUtils.GetLabelFor(BuiltInParameterGroup)	LabelUtils.GetLabelForGroup(ForgeTypeId)

ParameterUtils is a new class with static utility functions related to ForgeTypeId parameter identifiers:

- ParameterUtils.GetAllBuiltInGroups()
- ParameterUtils.GetAllBuiltInParameters()
- ParameterUtils.IsBuiltInGroup(ForgeTypeId)
- ParameterUtils.IsBuiltInParameter(ForgeTypeId)

The ParameterUtils class contains several methods that are new in this release but also deprecated. They are offered only to assist clients in migrating code from the BuiltInParameter and BuiltInParameterGroup enumerations to the ForgeTypeId class.

Introduced for porting assistance but deprecated
ParameterUtils.GetBuiltInParameter(ForgeTypeId)
ParameterUtils.GetBuiltInParameterGroup(ForgeTypeId)
ParameterUtils.GetParameterGroupTypeId(BuiltInParameterGroup)
ParameterUtils.GetParameterTypeId(BuiltInParameter)

SpecUtils is a new class with static utility functions related to ForgeTypeId spec identifiers:

- SpecUtils.GetAllSpecs()
- SpecUtils.IsSpec(ForgeTypeId)
- SpecUtils.IsValidDataType(ForgeTypeId)

The SpecUtils class contains two methods that are new in this release but also deprecated. They are offered only to assist clients in migrating code from the ParameterType enumeration to the ForgeTypeId class.

Introduced for porting assistance but deprecated
SpecUtils.GetParameterType(ForgeTypeId)
SpecUtils.GetSpecTypeId(ParameterType)

Changes were made to UnitUtils to clarify the meaning of some operations and to offer functions related to disciplines.

The methods GetAllSpecs() and IsSpec(ForgeTypeId) are renamed to GetAllMeasurableSpecs() and IsMeasurableSpec(ForgeTypeId) to clarify that they apply only to measurable specs, describing floating-point values associated with units of measurement, and to avoid confusion with the new Parameter class static methods GetAllSpecs() and IsSpec(ForgeTypeId), which apply to specs of all value types.

The new method:

- UnitUtils.GetAllDisciplines()

provides a list of all available Revit disciplines, suitable for iteration. Previously iteration would have used the members of the UnitGroup enumeration.

Deprecated	Replacement
UnitUtils.GetAllSpecs()	UnitUtils.GetAllMeasurableSpecs()
UnitUtils.GetUnitGroup(ForgeTypeId)	UnitUtils.GetDiscipline(ForgeTypeId)
UnitUtils.IsSpec(ForgeTypeId)	UnitUtils.IsMeasurableSpec(ForgeTypeId)

Renamed/replaced Parameter Group members

As a part of changes made to support externalized parameter groups, three BuiltInParameterGroup enumeration values were renamed:

Original	Replacement
PG_ELECTRICAL	PG_ELECTRICAL_ENGINEERING
PG_AELECTRICAL	PG_ELECTRICAL
PG_TERMINTATION	PG_TERMINATION

Note that because the name PG_ELECTRICAL now identifies a different group, client applications which do not adjust will find parameters they create now in "Electrical" instead of "Electrical Engineering".

Document & Worksharing API changes

SaveAsCloudModel() now supports save of a new Revit Cloud Worksharing central model

The method:

- Document.SaveAsCloudModel(Guid, Guid, String, String)

has been enhanced to support upload of local workshared file into BIM 360 Design as a Revit Cloud Worksharing central model.

In addition, the exception UnauthenticatedUserException is removed from the documented exceptions for this method.

Revit Links to Cloud Models

The methods:

- RevitLinkType.Create(Document, ModelPath, RevitLinkOptions)
- RevitLinkType.LoadFrom(ModelPath, WorksetConfiguration)

have been enhanced to support creation of new cloud model Revit links. You may use ModelPathUtils. ConvertCloudGUIDsToCloudPath() to create a cloud path to use as an argument to these methods.

These other link methods:

- RevitLinkType.IsLoaded()
- RevitLinkType.Load()
- RevitLinkType.Reload()
- RevitLinkType.Unload()

will work with any cloud Revit link created by a Revit user or through the APIs above.

Worksharing API exception changes

Several exceptions have changed from existing worksharing API.   For the method Document.SaveAsCloudModel():

Situation	Previous exception	Current exception
User doesn't sign in	ArgumentException	RevitServerUnauthenticatedUserException
Duplicated model name	RevitServerInternalException	RevitServerModelAlreadyExistsException
Hub/Project/Folder doesn't exist	ArgumentException	RevitServerUnauthorizedException
Folder is Plans folder or its sub folder	ArgumentException	RevitServerUnauthorizedException
Folder doesn't have "Edit" permission	ArgumentException	RevitServerUnauthorizedException

In addition, the function will now throw RevitServerModelNameBreaksConventionException when the model name breaks the project naming convention.

For the method ModelPathUtils.ConvertCloudGUIDsToCloudPath():

Situation	Previous exception	Current excepti
Region doesn't exist	RevitServerCommunicationException	RevitServerUnauthorizec
Project doesn't exist	CentralModelMissingException	RevitServerUnauthorizec
Model doesn't exist	No exception	RevitServerUnauthorizec
Model exists in a folder that doesn't have the "Edit" permission, but has View above permission	RevitServerUnauthorizedException	No exception

Worksharing API replacements

Some properties have been removed and replaced in this release. Because these were an immediate replacement, applications referencing the replaced properties will need to be updated immediately for this release.

Removed member	Replaced member
DocumentSavingAsEventArgs.IsSavingAsMasterFile	DocumentSavingAsEventArgs.IsSavingAsCentralFile
DocumentSavedAsEventArgs.IsSavingAsMasterFile	DocumentSavedAsEventArgs.IsSavingAsCentralFile

Sketched element API changes

Floor creation API replacements

The following members have been replaced:

Deprecated member	Replacement men
Autodesk.Revit.Creation.Document.NewFloor() (all overloads)	Floor.Create(Document document, IList<CurveLoop> profile
Autodesk.Revit.Creation.Document.NewSlab()	Floor.Create(Document document, IList<CurveLoop> profile bool isStructural, Line slopeArrow, double slope)
Autodesk.Revit.Creation.Document.NewFoundationSlab()	Floor.Create(Document document, IList<CurveLoop> profile

IndependentTag API Changes

Rotation

The new property

- RotationAngle

can be used to get or set the rotation of the tag relative to its view.

Multiple References

IndependentTag now supports multiple references to various elements or sub-elements.

A number of properties were deprecated. The users can now access the desired leader or tagged element using their References.

Deprecated Member	Replacement Members
IndependentTag.TaggedLocalElementId	IndependentTag.GetTaggedLocalElementIds()
IndependentTag.TaggedElementId	IndependentTag.GetTaggedElementIds()
IndependentTag.GetTaggedLocalElement()	IndependentTag.GetTaggedLocalElements()

IndependentTag.GetTaggedReference()	IndependentTag.GetTaggedReferences()
IndependentTag.HasElbow	IndependentTag.HasLeaderElbow()

IndependentTag.LeaderElbow	IndependentTag.GetLeaderElbow()
	IndependentTag.SetLeaderElbow()

IndependentTag.LeaderEnd	IndependentTag.GetLeaderEnd()
	IndependentTag.SetLeaderEnd()

All deprecated members still work in multi leader scenario by using only the first tagged reference.

The new members:

- IndependentTag.AddReferences()
- IndependentTag.RemoveReferences()
- IndependentTag. MultiLeader

are used to bulk add or remove references from a tag, and to query if the tag references multiple elements.

Revision API changes

RevisionSettings API replacements

The following methods have been deprecated and replaced due to terminology changes in the Revisions user interface:

Deprecated API	Replacement
RevisionSettings.GetNumericRevisionSettings()	RevisionNumberingSequence.GetAllRevisionNumberingSequences()
RevisionSettings.SetNumericRevisionSettings()	RevisionNumberingSequence.SetNumericRevisionSettings()
RevisionSettings.GetAlphanumericRevisionSettings()	RevisionNumberingSequence.GetAllRevisionNumberingSequences()
RevisionSettings.SetAlphanumericRevisionSettings()	RevisionNumberingSequence.SetAlphanumericRevisionSettings()

Structural reinforcement API changes

Reinforcement bars now uses two diameter values instead of just one:

- The model diameter which is used to model the reinforcement bars
- The nominal diameter which is used for formula computations

The following members were deprecated and replaced. Note that the deprecated members always reference the bar's "model diameter".

Deprecated member	Replacement members
BarTypeDiameterOptions.BarDiameter	BarTypeDiameterOptions.BarModelDiameter BarTypeDiameterOptions.BarNominalDiameter
RebarBarType.BarDiameter	RebarBarType.BarModelDiameter RebarBarType.BarNominalDiameter
RebarBendData.BarDiameter	RebarBendData.BarModelDiameter RebarBendData.BarNominalDiameter
RebarUpdateCurvesData.GetBarDiameter	RebarUpdateCurvesData.GetBarModelDiameter RebarUpdateCurvesData.GetBarNominalDiameter

Import/Export API changes

Import() and Link() methods now treat View as optional

The methods:

- Document.Import()
- Document.Link()

that take a View as an argument have been updated to accept null for that argument. This applies to all overloads of these methods except those that work with DWF. This View argument used to be required, but now null will be used to take an appropriate default action for the import operations.

Export API replacements

The following property has been removed and replaced in this release. Because these were an immediate replacement, applications referencing the replaced property will need to be updated immediately for this release.

Removed member	Replaced member
DGNExportOptions.MasterUnits	DGNExportOptions.WorkingUnits

MEP API changes

Electrical API changes

The following method and property have been deprecated; the old methods referred to temperature in terms on "long", which did not match the capabilities in the UI, and had an error when referring to non-Fahrenheit temperature values.

Deprecated API	Replacement
TemperatureRatingType.AddCorrectionFactor(long temperature, double factor)	TemperatureRatingType.AddCorrectionFactor(double temperature, double factor)
CorrectionFactor.Temperature	CorrectionFactor.GetTemperature()

Fabrication API changes

In a terminology change, the fabrication service "group" was renamed as a "palette" in order to align with changes made in the Revit user interface. This affected the following members:

Deprecated API	Replacement
FabricationNetworkChangeService.SetGroupId()	FabricationNetworkChangeService.SetPaletteId()
FabricationNetworkChangeService.SetRestrictGroup()	FabricationNetworkChangeService.SetRestrictPalette()
FabricationService.GroupCount	FabricationService.PaletteCount
FabricationService.IsValidGroupIndex()	FabricationService.IsValidPaletteIndex()
FabricationService.GetGroupName()	FabricationService.GetPaletteName()
FabricationService.IsGroupExcluded()	FabricationService.IsPaletteExcluded()
FabricationService.SetServiceGroupExclusions()	FabricationService.SetServicePaletteExclusions()
FabricationServiceButton.GroupIndex	FabricationServiceButton.PaletteIndex
FabricationPartSizeMap.GroupId	FabricationPartSizeMap.PaletteId

The following properties have been deprecated. Because they are not currently used and relate to not yet available capabilities, there is no replacement for these members.

Deprecated API	Replacement
FabricationConfigurationInfo.CloudVersion	None
FabricationConfigurationInfo.IsConnected	None

PostableCommand enumeration update

Many updates have been made to the PostableCommand enumeration to match the current set of Revit commands.

A good number of commands were newly added to the enumeration:

Anchors
BatchPrint
Bolts
Collaborate
CollaborateInCloud
ConnectionSettings
ContourCut
CoordinationModel
CoordinationSelectLink
Cope
CopeSkewed
CopyMonitorSelectLink
CopyMonitorUseCurrentProject
CornerCut
CutBy
CutThrough
DeleteEnergyModel
Dynamo
GetAutodeskContent
GlobalParameters
Holes
ImportImage
ImportPDF
LinkImage
LinkPDF
LinkTopography
LoadAsGroupIntoOpenProjects
LoadFamilyIntoProjectAndClose
LoadIntoProject
LoadRebarShapeIntoProjectAndClose
ManageCloudModels
Miter
Multiplanar
MultiPointRouting
NewFamily
NewProject
OpenFamily
OpenIFC
OpenIFCOptions
OpenProject

Optimize
PAndIDModeler
PAndIDSettings
PathOfTravel
Plate
PublishSettings
RepairCentralModel
ResetSharedCoordinates
RevealObstacles
SaveAsCloudModel
SaveAsFamily
SawCutFlange
SawCutWeb
ShapeStatus
SharedViews
ShearStuds
Shorten
ShowWarningsInViews
SpaceNaming
StructuralConnection
StructuralRebarCoupler
SystemsAnalysis
SystemZone
TitleBlock
ToggleHome
TogglePAndIDModelerBrowser
ToggleRevealHiddenElementsMode
Walkthrough
Welds
WorksharingMonitor

Some enumeration values were renamed to match the current naming of the commands in the Revit UI. There is no way to preserve the old name as obsoleted so code referring to these values will need to be updated:

Removed	Replacement
ExportMassModelGBXML	ExportGBXML
ModelBrowserContainer	ToggleHome
PlaceOnHost	PlaceOnStairOrRamp

Playlist	DynamoPlayer
RunEnergySimulation	Generate
ShowEnergyModel	CreateEnergyModel
UseCurrentProject	CoordinationReviewUseCurrentProject

Finally, some commands have been removed from the Revit UI. These commands can no longer be posted, so the values have been removed from the enumeration:

ExportBuildingSite
PublishDGNToAutodeskBuzzsaw
PublishDWFToAutodeskBuzzsaw
PublishDWGToAutodeskBuzzsaw
PublishDXFToAutodeskBuzzsaw
PublishSATToAutodeskBuzzsaw
RecentFiles
ResultsAndCompare
StairBySketch
ViewRange

BuiltInFailures changes

Some changes have been made to existing BuiltInFailure definitions. Code that looks for or handles specific failures may need updating, as it is not possible to obsolete the replaced members, so they have been removed in this release.

Failure submodule	Removed FailureDefinition	Replacement FailureDefinition
WallFailures	ExtrudedWallMustBeVertical	ExtrudedWallDisallowedForCrossSectionType
GroupFailures	FailedToSetMasterGroupId	FailedToSetLeaderGroupId

Filter API change

The constructor for the class FilterStringRule has been obsoleted and replaced. The previous constructor has a Boolean argument for case sensitivity that was not used in View Filters and could not be used properly from the API, as all string comparisons in filters are case-insensitive:

Obsoleted member	Replacement member

FilterStringRule(FilterableValueProvider, FilterStringRuleEvaluator, String, boolean)	FilterStringRule(FilterableValueProvider, FilterStringRuleEvaluator, String)

The method FilterStringRuleEvaluator.Evaluate() still has an argument "caseSensitive", but it's never called by filter code with this argument set to true.

Ruby macros no longer supported

Due to technology limitations, the Macro Manager no longer supports Ruby language in Revit Macros. The option to create new Ruby macros has been removed from the UI and API, and existing Ruby Macros in previous version documents will be deleted during upgrade.

TransmissionData API change

Autodesk.Revit.DB.TransmissionData.UserData is now used to store the systems analysis reports path, if any report exists in the document. This is to allow eTransmit to include external reports. The API method itself is not changed.

Removal of obsoleted APIs

The following previously obsoleted APIs have been removed:

- ModelPathUtils.ConvertCloudGUIDsToCloudPath(Guid projectId, Guid modelId)
- Document.SaveAsCloudModel(String folderId, String modelName)
- MEPModel.ElectricalSystems
- MEPModel.AssignedElectricalSystems
- StructuralConnectionHandler.Create(Document doc, List<ElementId> idsToConnect)

API additions

Sketched Elements API

Ceiling creation

The new methods:

- Ceiling.Create(Document document, IList<CurveLoop> curveLoops, ElementId ceilingTypeId, ElementId level)
- Ceiling.Create(Document document, IList<CurveLoop> curveLoops, ElementId ceilingTypeId, ElementId levelId, Line slopeArrow, double slope)

create a new instance of a ceiling in a Revit project, with one overload supporting a sloped ceiling.fac

Floor APIs

In addition to the replacement Floor creation methods described above, the new method:

- Floor.GetDefaultFloorType(Document document, bool isFoundation)

returns the id of the default floor or foundation type for a document.

Wall APIs

The new methods:

- Wall.CreateProfileSketch()
- Wall.RemoveProfileSketch()
- Wall.CanHaveProfileSketch()

provide access to add and remove profile sketches to wall elements that support them. Once a sketch is added, the profile sketch can be edited using SketchEditScope.

Slanted and Tapered Walls

The new members:

- Wall.CrossSection
- Wall.IsWallCrossSectionValid()

allow assignment of a cross section type to a given wall, and determine if that type can be assigned.

The new enumerations:

- Autodesk.Revit.DB.WallCrossSection
- Autodesk.Revit.DB.WidthMeasuredAt
- Autodesk.Revit.DB.InsertOrientation

enumerate the values for options related to alternate wall cross section values, especially the options available for Tapered walls.

Sketch APIs

The new methods:

- Ceiling.SketchId
- Floor.SketchId

- Wall.SketchId
- Opening.SketchId

return the id of the element's sketch.

The method:

- Sketch.OwnerId

returns the id of the element that owns this sketch.

The new method:

- Sketch.GetAllElements()

returns the ids of all the elements owned by the sketch, including elements of type:

- Autodesk.Revit.DB.ModelCurve
- Autodesk.Revit.DB.ReferencePlane
- Autodesk.Revit.DB.Dimension

This method works for any Sketch you can obtain from a Sketch-based element (Ceiling, Extrusion, etc.). It is available both in and outside of a SketchEditScope.

Editing Sketches with SketchEditScope

Using the new class:

- Autodesk.Revit.DB.SketchEditScope

allows an application to edit sketch based elements while preserving the unchanged elements that are already present in the sketch. While a Sketch editing session is active, you can add, delete or modify Sketch elements (curves, dimensions, reference planes). When you finish the session, the edited Sketch-based element will be updated. Only certain sketches are currently supported for editing:

- Ceiling sketches
- Floor sketches
- Wall profile sketches
- Opening sketches

using the new members:

- SketchEditScope constructor - creates a new SketchEditScope
- SketchEditScope.Start() - starts editing a particular sketch, including starting the associated transaction. After this is started only elements owned by the Sketch and new elements to be added to the Sketch may be modified.
- SketchEditScope.IsSketchEditingSupported() - checks if a particular sketch can be edited with a SketchEditScope.

Boundary validation for sketched elements

The new method:

- BoundaryValidation.IsValidHorizontalBoundary()

validates input curves form a valid horizontal boundary for Ceiling or Floor creation.

CompoundStructure API

The new method:

- CompoundStructure.CanSplitAndMergeRegionsBeUsed()

checks whether split and merge regions operations can be used for this compound structure. This validator is associated with the use of the existing M ergeRegionsAdjacentToSegment() and SplitRegion() methods.

Color Fill API

Multiple new classes provide access to color fills. These classes expose color fill schemes, entries and color f ill legends. You can also also a ccess the color f ill scheme applied to a particular view.

Color Fill Schemes

The new class:

- Autodesk.Revit.DB.ColorFillScheme

represents a color scheme that can be applied to plan and section views. Key new members of this class include:

- ColorFillScheme.AddEntry()
- ColorFillScheme.RemoveEntry()
- ColorFillScheme.UpdateEntry()
- ColorFillScheme.GetEntries()
- ColorFillScheme.SetEntries()
- ColorFillScheme.SortEntries()
- ColorFillScheme.AreaSchemeId
- ColorFillScheme.CategoryId
- ColorFillScheme.ParameterDefinition
- ColorFillScheme.StorageType
- ColorFillScheme.Title
- ColorFillScheme.Duplicate
- ColorFillScheme.GetFormatOptions()
- ColorFillScheme.SetFormatOptions()
- ColorFillScheme.GetSupportedParameterIds()

Color Fill Scheme Entries

The new class:

- Autodesk.Revit.DB.ColorFillSchemeEntry

represents an entry in a color scheme. Key new members of this class include:

- ColorFillSchemeEntry.Color
- ColorFillSchemeEntry.FillPatternId
- ColorFillSchemeEntry.Caption
- ColorFillSchemeEntry.StorageType
- ColorFillSchemeEntry.IsVisible
- ColorFillSchemeEntry.IsInUse
- ColorFillSchemeEntry.GetIntegerValue()
- ColorFillSchemeEntry.GetDoubleValue
- ColorFillSchemeEntry.GetStringValue()
- ColorFillSchemeEntry.GetElementIdValue()
- ColorFillSchemeEntry.SetIntegerValue()
- ColorFillSchemeEntry.SetDoubleValue()
- ColorFillSchemeEntry.SetStringValue()
- ColorFillSchemeEntry.SetElementIdValue()

View access to Color Fills

The new members:

- View.GetColorFillSchemeId()
- View.SetColorFillSchemeId()

provide access to read and apply the color fill scheme associated with a particular category in the view.

Color Fill Legends

The new class:

- Autodesk.Revit.DB.ColorFillLegend

offers the ability to create, read and modify properties of color fill legend annotation elements in a particular owner view. Key new members of this class include:

- ColorFillLegend.Create()
- ColorFillLegend.GetColumnWidths()
- ColorFillLegend.SetColumnWidths()
- ColorFillLegend.ColorFillCategoryId
- ColorFillLegend.Height
- ColorFillLegend.Origin

Export API additions

Export to PDF

The new method:

- Document.Export(String, IList<ElementId>, PDFExportOptions)

exports one or more views and sheets in PDF format.

The options provided, in the new class:

- Autodesk.Revit.DB.PDFExportOptions

include graphical options, paper and output format options, and the option to combine output views and sheets into a single PDF or to produce a different PDF for each view and sheet with a specific naming convention.

PDF Export Settings

The new element class:

- Autodesk.Revit.DB.ExportPDFSettings

represents a named stored PDF export option available in a document.   You can use this class to extract the options from this element to use for exporting to PDF, or modify or create new stored settings with different options.

Civil Alignments API

Revit now provides support for Civil Alignments and their associated annotations. Alignments are imported from InfraWorks as a part of the workflow to transfer Civil Structures. The API supports read of alignment properties and geometric information, along with read/write and create of associated annotations. All new classes for the Alignments API are exposed through a different assembly in the Revit installation, located at: **Addins\CivilAlignments\Autodesk.CivilAlignments.DBApplication.dll**

The new class:

- Autodesk.Revit.DB.Infrastructure.Alignment

represents an alignment and can be used to find alignments in a document, and to query a particular alignment's properties and to analyze alignment geometry. This object is not an Element, but the underlying Element can be obtained from this object if needed.

The new class:

- Autodesk.Revit.DB.Infrastructure.AlignmentStationLabel represents an alignment station label annotation and can be used to find such labels in a document as well as to create and modify such labels. This object is not an Element, but the underlying Element (which is a SpotDimension instance) can be obtained from this object if needed.

The new classes:

- Autodesk.Revit.DB.Infrastructure.AlignmentStationLabelOptions
- Autodesk.Revit.DB.Infrastructure.AlignmentStationLabelSetOptions

provide options for creating a single alignment label or for creating a set of alignment labels.

DirectShape API Additions

Reference Curves, Planes, and Points

The new methods:

- DirectShape.AddReferenceCurve()
- DirectShape.AddReferencePlane()
- DirectShape.AddReferencePoint()
- DirectShapeType.AddReferenceCurve()
- DirectShapeType.AddReferencePlane()
- DirectShapeType.AddReferencePoint()

enable the creation of reference curves, planes and points inside DirectShape elements. Explicit bounds can be provided for direct shape reference curves and planes. Revit tools that can use named references within families will also be able to select the references inside the DirectShape elements.

The overloads for these methods include an optional DirectShapeReferenceOptions input. Use:

- DirectShapeReferenceOptions.Name

to set the assigned name for the reference. If the name is specified, it is visible when picking the reference's geometry. Otherwise, the default DirectShape element name is displayed.

The new validator:

- DirectShapeReferenceOptions.IsValidReferenceName()

validates the name assigned to DirectShapeReferenceOptions.Name.

The new validators:

- DirectShape.IsValidReferenceCurve()
- DirectShape.IsValidReferencePlaneBoundingBoxUV()
- DirectShapeType.IsValidReferenceCurve()
- DirectShapeType.IsValidReferencePlaneBoundingBoxUV()

validates the inputs needed for specifying a plane or curve explicit reference in the DirectShape.

External Geometry

The new class:

- ExternalGeometryId

represents an identifier for geometry from external sources.

The new members:

- ExternalGeometryId.IsValidExternalGeometryId()
- ExternalGeometryId.Id

provide validation of an identifier and a means to access the identifier as a String.

The new class:

- ExternallyTaggedGeometryObject

serves as a base class for externally tagged geometry.

The new members:

- ExternallyTaggedGeometryObject.ExternalId

provides the ExternalGeometryId associated with the geometry.

The new class:

- ExternallyTaggedNonBRep

represents non-BRep geometry with an external identifier.

The new class:

- ExternallyTaggedBRep

represents BRep (Solid) geometry with external identifiers for the BRep itself and any subset of Faces or Edges.

The new class:

- BRepBuilderPersistentIds

Provides a mechanism to associate external identifiers with geometry created by a BRepBuilder.

The new members:

- BRepBuilderPersistentIds.AddSubTag()
- BRepBuilderPersistentIds.IsValidExternalGeometryIdForNewCorrespondence()
- BRepBuilderPersistentIds.IsValidBRepBuilderGeometryIdForNewCorrespondence()
- BRepBuilderPersistentIds.IsAssociatedBRepBuilderValid()
- BRepBuilderPersistentIds.IsBRepBuilderGeometryIdFaceOrEdge()

provide an ability to associate “tags” with Faces and Edges created by BRepBuilder.

A new member:

- BRepBuilder.GetResult()

accepting a BRepBuilderPersistentIds has been added which returns an ExternallyTaggedBRep.

The new class:

- ExternallyTaggedGeometryValidation

provides validators

- ExternallyTaggedGeometryValidation.IsValidGeometry
- ExternallyTaggedGeometryValidation.IsNonSolid
- ExternallyTaggedGeometryValidation.IsSolid

to help determine validity of a GeometryObject for use with the new externally tagged geometry.

The new members:

- DirectShape.AddExternallyTaggedGeometry()
- DirectShape.GetExternallyTaggedGeometry()
- DirectShape.HasExternallyTaggedGeometry()
- DirectShape.UpdateExternallyTaggedGeometry()
- DirectShape.RemoveExternallyTaggedGeometry()
- DirectShape.ResetExternallyTaggedGeometry()
- DirectShapeType.AddExternallyTaggedGeometry()
- DirectShapeType.GetExternallyTaggedGeometry()
- DirectShapeType.HasExternallyTaggedGeometry()
- DirectShapeType.UpdateExternallyTaggedGeometry()
- DirectShapeType.RemoveExternallyTaggedGeometry()
- DirectShapeType.ResetExternallyTaggedGeometry()

provide a means to add, remove, update, or retrieve an ExternallyTaggedGeometryObject to/from a DirectShape or DirectShapeType.

Custom Family names

The new member:

- DirectShapeType.SetFamilyName()

provides the ability to set a custom Family name for a DirectShapeType.

The new validator:

- DirectShapeType.CanChangeFamilyName()

provides the ability to check if a DirectShapeType supports a custom Family name. Certain categories do not support custom Family names.

Document & Worksharing API additions

Cloud Model API additions

Four new methods allow access to identifying forge IDs for Cloud Models:

- Document.GetHubId(): ForgeDM hub id where the model locates.
- Document.GetProjectId(): ForgeDM project id where the model locates.
- Document.GetCloudFolderId(bool forceRefresh): ForgeDM folder id where the model locates.
- Document.GetCloudModelUrn(): A ForgeDM Urn identifying the model.

These methods all return strings, which will be empty for a document which is not a cloud model.

OpenOptions API

The new property:

- OpenOptions.IgnoreExtensibleStorageSchemaConflict

allows your application to ignore a schema conflict error on open. The default is false. Setting this to true will ignore the exception of schema conflict while opening the model. Data in the existing schema with the conflicting ID will be erased from the model.

View API additions

Label for views on sheets (Viewport)

The new property:

- Viewport.LabelOffset

controls the two-dimensional label offset from left bottom corner of the viewport (as established with Rotation set to None) to the left end of the viewport label line.

The new property:

- Viewport.LabelLineLength

controls the length of the viewport label line in sheet space.

Callout view

The new method:

- View.GetCalloutParentId()

returns the ID of a view which this callout references, or InvalidElementId if there is not parent.

The new property:

- View.IsCallout

identifies if the view is a callout view.

Grids in 3D views

Several new methods in View3D control grid visibility in 3D Views:

- View3D.GetLevelsThatShowGrids()
- View3D.ShowGridsOnLevel(ElementId levelId)
- View3D.HideGridsOnLevel(ElementId levelId)
- View3D.ShowGridsOnLevels(ElementIdset levelIds)

Graphics API additions

Temporary in-canvas graphics

The new class:

- TemporaryGraphicsManager

allows the creation of temporary graphics in the canvas and can be used to create in-canvas controls.

It has the following members:

- TemporaryGraphicsManager.GetTemporaryGraphicsManager()
- TemporaryGraphicsManager.AddControl()
- TemporaryGraphicsManager.UpdateControl()
- TemporaryGraphicsManager.RemoveControl()
- TemporaryGraphicsManager.SetVisibility()
- TemporaryGraphicsManager.GetAll()
- TemporaryGraphicsManager.Clear()

The new class:

- InCanvasControlData

stores a reference to the image used in the temporary graphics. It has the following properties:

- InCanvasControlData.ImagePath
- InCanvasControlData.Position

Label API additions

The new methods:

- Level.GetNearestLevelId(Document document, double elevation)
- Level.GetNearestLevelId(Document document, double elevation, out double offset)

return the id of the Level which is closest to the specified elevation. The level can be at, above or below the target elevation. If there is more than one Level at the same distance from the elevation, the Level with the lowest id will be returned.

Parameter API additions

The new class:

- Autodesk.Revit.DB.MultipleValuesIndicationSettings

allows access to the custom value for used in instances of the Properties Palette, Tags and Schedules when multiple elements are referenced and the value of the parameter is different.

The new members:

- MultipleValuesIndicationSettings.GetMultipleValuesIndicationSettings()
- MultipleValuesIndicationSettings.Custom
- MultipleValuesIndicationSettings.CustomValue
- MultipleValuesIndicationSettings.Value

provide access to the settings in the document, the option for the custom value, and read-only access to the value that will be used (either the custom value or the default).

Dimension API additions

Dimension layout

Several new properties allow access to positioning for various parts of a dimension:

- Dimension.TextPosition
- Dimension.LeaderEndPosition
- Dimension.HasLeader
- SpotDimension.LeaderElbowPosition
- SpotDimension.LeaderHasElbow

DimensionType prefix/suffix

The new properties:

- DimensionType.Prefix
- DimensionType.Suffix

allow access to the corresponding values in the dimension type.

Category API additions

Built in Categories

The new methods:

- Category.GetBuiltInCategoryTypeId(BuiltInCategory)
- Category.GetBuiltInCategory(ForgeTypeId)
- Category.IsBuiltInCategory(ForgeTypeId)

allow access to BuiltInCategories.

Shared Coordinates API additions

Reset Shared Coordinates

The new method:

- Document.ResetSharedCoordinates()

resets the shared coordinates for the host model. It provides the same functionality as the UI Command "Reset Shared Coordinates".

After resetting coordinates, the following changes will take place:

1. GIS coordinate system will be erased
2. Shared coordinates relationships with other linked models will be eliminated.
3. The Survey Point will be moved back to the startup location, where it coincides with the Internal Origin.
4. The angle between Project North and True North will be reset to 0.

Note: There will be no changes to linked models.

Clipped state of BasePoint

The new property:

- bool BasePoint.Clipped

will get or set the clipped state of the survey point BasePoint based on the active ProjectLocation of its Document. For the project base point, the get method will always return false, and the set method will throw an exception.

Revision API additions

Revision sequences and API additions

The new class:

- Autodesk.Revit.DB.RevisionNumberingSequence

defines the sequences by which numbers are assigned to Revisions. Revision numbering is either numeric or alphanumeric. Revision numbering is assigned to Revisions to control the numbering scheme of that Revision. Alphanumeric from the API corresponds to the UI concept of "Custom".

The new members:

- RevisionNumberingSequence.CreateNumericSequence()
- RevisionNumberingSequence.CreateAlphanumericSequence()
- RevisionNumberingSequence.GetSequenceName()
- RevisionNumberingSequence.SetSequenceName()
- RevisionNumberingSequence.GetNumericRevisionSettings()
- RevisionNumberingSequence.SetNumericRevisionSettings()
- RevisionNumberingSequence.GetAlphanumericRevisionSettings()
- RevisionNumberingSequence.SetAlphanumericRevisionSettings()
- RevisionNumberingSequence.GetAllRevisionNumberingSequences()
- RevisionNumberingSequence.HasValidRevisionSettingsForNumberType()
- RevisionNumberingSequence.HasValidNumericRevisionSettings()
- RevisionNumberingSequence.HasValidAlphanumericRevisionSettings()
- RevisionNumberingSequence.NumberType

provide the ability to create, read and modify the settings related to Revision numbering sequences.

The new methods:

- Revision.GetRevisionNumberingSequenceId()
- Revision.SetRevisionNumberingSequenceId()

provide access to the id of the revision numbering sequence which controls this revision's numbering:

The new methods:

- NumericRevisionSettings.GetMinimumDigits() - get the minimum number of digits for a revision number.
- NumericRevisionSettings.SetMinimumDigits() - set the minimum number of digits for a revision number.

provide access to the setting controlling the minimum number of digits for a revision number.

Phasing API additions

The new methods:

- Element.IsDemolishedPhaseOrderValid()
- Element.IsCreatedPhaseOrderValid()

validate the order of phases on a given element with respect to the input phase id, ensuring that an object is not assigned a phase where it is demolished before it was created.

Geometry API additions

The new property:

- Mesh.IsClosed

checks if each edge in the mesh belongs to at least two faces.

Point Cloud API additions

The new method:

- PointCloudType.GetReCapProject()

provides a direct entry point to get access to an object from the ReCap SDK (ReCapWrapper.RCProject) from Revit. This object represents the point cloud from the RC file path stored in PointCloudType. The ReCap assembly ReCapWrapper.dll will need to be included into code using this method.

The coordinate system in RCProject is defined by the Point Cloud. Please refer to ReCap SDK documentation for RCProject.getCoordinateSystem(). If you need points converted to the modeling coordinate system in Revit, you can obtain the transformation matrix from PointCloudInstance.GetTransform().

Schedule API additions

Multiple Value Indication - schedule customization

The existing class:

- ScheduleField

has several new properties which allow customization of the multiple value indication per field:

- ScheduleField.MultipleValuesDisplayType
- ScheduleField.MultipleValuesText - The text to be used when a field has multiple values
- ScheduleField.MultipleValuesCustomText - The text to be used when a field has multiple values and the display type is set to ScheduleFieldMultipleValuesDisplayType.Custom

The new Enum:

- ScheduleFieldMultipleValuesDisplayType

defines how the schedule field's multiple value indication is displayed (using the project setting, a custom text or a predefined text "<varies>").

Import API additions

Import/Link Rhino

The new methods:

- Document.Import(String, ImportOptions3DM, View)
- Document.Link(String, ImportOptions3DM, View)

import or link a 3DM file into the document.

Structural API additions

Moving individual rebar in a Rebar Set

The new methods:

- Rebar.MoveBarInSet()
- Rebar.GetBarIndexFromReference()
- Rebar.GetMovedBarTransform()
- Rebar.ResetMovedBarTransform()

allow an application to move an individual bar, and to read and reset the transform of the individual bar in a Rebar Set.

Removing individual bars from a Rebar Set

The new function:

- Rebar.SetBarIncluded()

allows you to designate a particular bar at a given index to be included or not.

The existing properties:

- Rebar.IncludeFirstBar
- Rebar.IncludeFirstBar

continue to function as before, and setting these to true are equivalent to removing the bar at the first or last position index.

The existing function:

- Rebar.DoesBarExistAtPosition()

now supports returning false for any bar position index and not just the first or last bar.

Moving and Removing bars owned by PathReinforcement

The new methods:

- RebarInSystem.SetBarIncluded()
- RebarInSystem.MoveBarInSet()
- RebarInSystem.GetMovedBarTransform()
- RebarInSystem.ResetMovedBarTransform()
- RebarInSystem.CanEditIndividualBars()

provide access to move, include or remove individual bars only for RebarInSystem elements that are owned by PathReinforcement. (This is not supported for RebarInSystem owned by AreaReinforcement).

Area Reinforcement Layers

The new methods:

- AreaReinforcement.GetNumberOfLines()
- AreaReinforcement.GetLineFromLayerAtIndex()
- AreaReinforcement.SetLineIncluded()

- AreaReinforcement.IsLineIncluded()
- AreaReinforcement.MoveLine()
- AreaReinforcement.GetMovedLineTransform()
- AreaReinforcement.ResetMovedLineTransform()
- AreaReinforcement.SetLayerActive()
- AreaReinforcement.IsLayerActive()
- AreaReinforcement.GetLayerDirection()

provide access to the layers of Area Reinforcement elements, and to manipulate (move and remove) the individual lines exposed through those layers.

Rebar conversion

The new methods:

- AreaReinforcement.ConvertRebarInSystemToRebars()
- PathReinforcement.ConvertRebarInSystemToRebars()

convert all of the RebarInSystem elements owned by the input element into equivalent Rebar elements.

Rebar geometry

The new methods:

- Rebar.GetTransformedCenterlineCurves()
- RebarInSystem.GetTransformedCenterlineCurves()

return the centerline curves for a given bar, where the geometry of the curves are in the actual transformed position. The BarPositionTransform (representing the relative position of any individual bar in the set - a translation along the distribution path) and MovedBarTransform (representing the movement of the bar relative to its default position along the distribution path) will be applied to the returned curves.

Freeform Rebar

The new properties:

- RebarFreeFormAccessor.RebarStyle
- RebarFreeFormAccessor.StirrupTieAttachmentType

provides read and write access to the corresponding properties of freeform rebar elements.

The new method:

- RebarFreeFormAccessor.SetReportedShape()

changes the rebar shape of a freeform rebar that is currently using the RebarWorkInstructions.Straight option to the provided rebar shape.

MEP API additions

Building and Space Type additions

Several new properties have been added for Building and Space Types:

- HVACLoadType.HeatingSetPoint - The heating temperature set point in unit Kelvin (K).
- HVACLoadType.CoolingSetPoint - The cooling temperature set point in unit Kelvin (K).
- HVACLoadType.HumidificationSetPoint - The humidification set point as a number between 0 and 1.
- HVACLoadType.DehumidificationSetPoint - The dehumidification set point as a number between 0 and 1.

Zone additions

The new property:

- GenericZone.LevelOffset

gets or sets the offset distance from this zone to the associated level.

Mechanical Systems Analysis Set Points

The new properties:

- HVACLoadType.HeatingSetPoint
- HVACLoadType.CoolingSetPoint
- HVACLoadType.HumidificationSetPoint
- HVACLoadType.DehumidificationSetPoint

provide access to the heating temperature, cooling temperature, humidification and dehumidifcation set point values. Temperature set points are in Kelvin, while humidity set points are percentages ranging from 0 to 1.

Analysis Report Style

The new property:

- ViewSystemsAnalysisReport.ReportStyle

accesses the report style used in a particular analysis report view.

MEP Hidden Line Settings

The new class:

- Autodesk.Revit.DB.Mechanical.MEPHiddenLineSettings represents the settings of the mechanical hidden line display (e.g. ducts and pipes). It can be obtained from the static method:

- MEPHiddenLineSettings.GetMEPHiddenLineSettings(Document)

It offers the following properties:

- MEPHiddenLineSettings.DrawHiddenLine
- MEPHiddenLineSettings.LineStyle
- MEPHiddenLineSettings.InsideGap
- MEPHiddenLineSettings.OutsideGap
- MEPHiddenLineSettings.SingleLineGap

Electrical Panel Schedule

The new property:

- PanelScheduleData.IsAutoShadingForLoadDisplay

indicates if a panel schedule will display shading for Load cells automatically.

Conversion to Fabrication

The new method to the DesignToFabricationConverter class:

- DesignToFabricationConverter.SetMapForFamilySymbolToFabricationPartType()

The method will set a map used when the design to fabrication tool is running. The map relationships will be used to replace any family instance that is based upon the family symbols found in the map with the corresponding fabrication part for the fabrication part type set in the mapping.

Energy Analysis API additions

The options used to produce an energy model or export to gbXML have been unified.

The new enumeration value:

- AnalysisMode.RoomsOrSpaces

indicates that the energy analysis model should be based on volumes from rooms or spaces defined in the building model.

The new enumeration values:

- EnergyModelType.AnalysisMode
- ExportEnergyModelType.AnalysisMode

allow specification of the energy model production based on the given analysis type.