**Revit® Structure API: From Bi-Directional Link to Rebar Detailing**

Mikako Harada – Autodesk Inc.

**DE215-3** Two main areas of interest in the Revit Structure (RST) API are the bi-directional link to an external structural analysis program and reinforced concrete detailing design. In addition to the physical building information model (BIM) defined in Revit Architecture, RST defines an analytical model of the building, composed of geometry, loads, connectivity, release/boundary conditions, material properties, and other project parameters. This session shows how to implement a bi-directional link to a structural analysis program. We will then look at the reinforcement concrete detailing API and show how to automate part of the reinforcement workflow, eliminating repetitive manual tasks. This session assumes basic knowledge of Revit programming.

**About the Speaker:**

Mikako works as a consultant for the Developer Technical Services Team at Autodesk, providing API technical support to Autodesk Developer Network members worldwide for AEC products—primarily AutoCAD® Architecture and Revit® Architecture. Prior to joining Autodesk, Mikako was a researcher at universities and private research labs and worked in the areas of interactive techniques, optimization, and layout synthesis.



## Introduction

In addition to the physical building information model (BIM) defined by Revit Architecture, Revit Structure (RST) defines an analytical model of the building. They are composed of geometry, loads, connectivity, release and boundary conditions, material properties, and project parameters, which can be used to link a model to an external analysis application for calculation and optimization of the building model. In this talk, we will focus on the Revit APIs that are specific to Revit Structure, which allows us to access and modify data and building models to implement a bi-directional link with structural analysis software. We’ll show you how to access necessary information, such as geometry, location, boundary and release conditions, the point, line and area loads as well as the physical building elements (e.g., walls, floors, and framing elements). We’ll then demonstrate a sample implementation of Revit add-in linking Revit with an external analysis package simulated using AutoCAD.

In the second half of the talk, we’ll also look at the reinforcement bar (rebar) APIs, focusing on rebar shape definition. Rebar shape is an analogous to family definition. A rebar shape defines a topology of rebar placement and allows you to implement a parametric form of rebar placement within a framing element.

This presentation is accompanied by additional labs and samples:

* rst\_lab.zip – Labs – sample code for exploration and self learning
* rst\_link.zip – RSLink – a sample link application and analysis simulation in AutoCAD

*Note: We’ll post those to the AU site shortly after the class. They will help you learn Revit API from Revit Structure perspective.*

### Comparison with Revit Architecture

Revit Architecture and Structure have many common features. The differences are visible in the user interface in the command organization in the design bar and menu bars, and in the category filtering. Some functionality has been added to support structure-specific features. Some has also been removed:

General User Interface:

* Command organization
* Category filtering

Functionality added:

* Analytical model (generation and adjustment)
* Load modelling
* API specific model and load objects
* Rebar modelling
* Composite metal deck
* Graphical column schedule

Functionality removed:

* Reflected ceiling plans
* Room related functionalities
* Model text

### Analytical Model

The most important problem of the structural industry is the fact that the information used for structural analysis is physically different than the one used for drawings. Revit Structure solves this problem. The physical model is the model that we use for drawings, coordination and construction; the analytical model is the simplified model used for analysis.

For example, consider a plan view where the walls need to be aligned for the analysis and the beams need to join the centre of columns to end of walls. In other situation, in an elevation view, the analytical model is horizontally projected even if the physical model of the beam has a slope. Or consider a situation where the physical model of the beam is curved and the analytical model is segmented. Revit Structure maintains the coordination between the two models and will keep the analytical model of each element connected. When extracting model information for analysis, you just need to work with the analytical model.

Typically, members that need to participate in the analysis are as follows:

Geometry (location)

* Sections
* Materials
* Release conditions
* Reference to level (optional)

Loads

* Load cases
* Load combinations
* Point, line, area loads

Boundary conditions

* Support data

We’ll explain how to access all this information later when we look at RST labs.

### Data Exchange Workflow

Figure1 shows the typical workflow during the structural analysis. We start off with a physical model. The physical model can be used to automatically generate the analytical model. The analytical model is exported to an external analysis package, which may add its own data. As a result of the analysis, the analytical model may be modified, resulting in changes in the physical model as well. New changes may be added in the physical model as well. All of these processes may continue over many iterations.

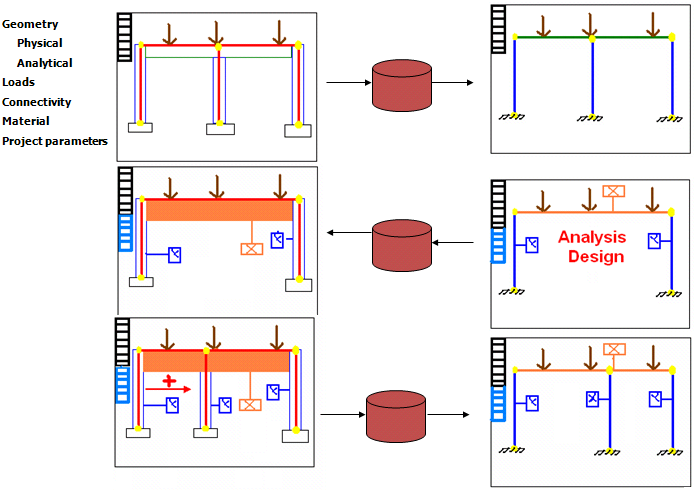


Figure 1: Data exchange workflow

Revit API allows us to implement above mentioned bi-directional link between Revit Structure and an analysis program. To address these needs, the following is an overview of the typical requirements for implementing a link to an external analysis package using the RST API.

* Extract the analytical model geometry
* Extract the load cases, load combinations, loads
* Extract release and boundary conditions
* Update the model with changes and new and deleted elements
* Add load cases and load combinations
* Add loads and reactions
* Add project information, shared parameters
* Add shared parameters on structural objects

We will look at the Revit samples and a series of labs demonstrating how to address these issues one by one. Finally, the results of these labs will be integrated into the sample link application.

### Useful Revit SDK Samples

Currently, information on how to combine Revit API calls to achieve certain complex tasks is contained mainly in the SDK samples. Here is a list of some of the Revit SDK samples, which address areas related to structure.

Parameters

* BrowseBindings
* CreateShared
* FireRating
* InvisibleParam

Analytical Model

* AnalyticalSupportData\_Info
* AnalyticalViewer
* BoundaryConditions
* InplaceFamilyAnalyticalModel3D
* Loads
* RotateFramingObjects
* SlabProperties
* SpanDirection
* TestWallThichness

## Revit Structure Analysis Labs

In this section, we demonstrate step by step instruction of how to implement the functionality required for an application linking Revit Structure with an external analysis package. Below is the organization of the labs: the first group of labs addresses access and modification of load data, the second examines the building structure, and the third addresses the entire analytical model.

Loads:

* 1. Load cases, natures, combinations, usages
  2. Point, line and area loads
  3. Load parameters and modification
  4. Load symbols
  5. Creating load objects

Structural Elements:

2-1 Columns and framing

2-2 Foundations

2-3 Walls, floors, footings

Analytical Model:

3 Analytical model

Loads

Lab 1-1 extracts and lists the load cases, natures, combinations and usages defined in the Revit model. To retrieve these, simply iterate over the document elements and check for the corresponding object class LoadCase, LoadNature, LoadCombination and LoadUsage or use use a filter. To create a new object of these classes, you can use the Autodesk.Revit.Creation.Document methods NewLoadCase(), NewLoadCombination(), NewLoadNature(), NewLoadUsage(), accessible through ActiveDocument.Create.

Lab 1-2 shows how to access load objects (i.e., Point/Line/Area) using the base or specific classes. All three load classes are derived from LoadBase. They are creation through NewAreaLoad() (3 overloads), NewLineLoad() (2), and NewPointLoad(). Just like in the preceding lab, we iterate over all Revit elements and pick out or filter the object types of interest. For each load type object, a number of properties are defined for extracting and modifying data.

Lab 1-3 continues the exploration about loads, for example, shows more details about selected loads, how to access load parameters, and how to modify load objects.

#### One thing to note is that the way we can access parameters are somewhat inconsistent. Take a point load parameters, as an example; some parameters can be accessed via both built-in parameter enum and localized name. But for others, only one or the other, and not both:

|  |  |  |
| --- | --- | --- |
|  | Built-in Parameter | Display Name |
| Equivalent | LOAD\_IS\_REACTION | Is Reaction |
| Equivalent | LOAD\_CASE\_ID | Load Case |
| Similar | LOAD\_USE\_LOCAL\_COORDINATE\_SYSTEM | Orient to |
| Built-in only | LOAD\_USE\_LOCAL\_COORDINATE\_SYSTEM\_HOSTED |  |
| Built-in only | LOAD\_IS\_CREATED\_BY\_API |  |
| Display name only |  | Fz |

Parameters may vary from Revit version to version. A list of 59 different parameters relevant to loads is given in the comment at the end of Labs1.vb.

Some of the point load properties accessible through parameters are also exposed directly as properties on the object class:

LoadBase properties:

* HostElement
* LoadCaseName
* LoadCategoryName
* LoadNatureName

PointLoad properties:

* Force
* Moment
* Point

The HostElement property was added in 2008. Many other properties are also available. Whenever you are looking for a specific property, you will need to explore the Revit API and one or more suitable sample models to determine what exactly you are looking for and how to access it. Please look, for example, help file, reflection, debugger, BuitInParamsChecker (in Rac Labs), and RvtMgdDbg.

Lab 1-4 demonstrates how to retrieve load symbols (also known as types) available in the project, and determine load symbol family. It include one utility for all load symbols (point, line and area), and another utility specifically for point loads. Load symbols are system families, and not standard or component family. In RST 2008, to retrieve load symbols from the current model, we used a standard Revit element iteration loop and pick all the Symbol instances, then check for the structural loads category. In RST 2009, load type is exposed; You can check LoadTypeBase to obtain all the load types, or use Point/Line/AreaLoadType. The built-in parameter SYMBOL\_FAMILY\_NAME\_PARAM that returns the family name can be used to determine a load symbol’s family name. (note: it could be localized.)

Lab 1-5 shows how to create a new load object, specifically a point load. It creates two new point loads by calling NewPointLoad() with somewhat arbitrary parameters. One is as reaction, simulating an analysis result. The other is a standard external load. It also shows how to set a selected symbol for this new load and assign the load to a selected load case. To assign a selected symbol to it, all point load symbols in the document are retrieved and the user is allowed to select one of them. Its id is then assigned to the newly created point load ELEM\_TYPE\_PARAM. Similarly, to assign a load case, all load cases are retrieved and displayed and the selected one's id is assigned to the LOAD\_CASE\_ID parameter.

### Structural Elements

Columns and framing elements are structural elements using standard families. Lab 2-1 demonstrates how to retrieve all structural 'line segment' or 'stick' elements. Columns belong to a separate standard family. Others such as beam, brace, joist, etc. belong to the framing standard family.

To select all columns in the model, we once again iterate over the Revit document elements and select all standard family instances for category OST\_StructuralColumns. We can display generic element and column properties such as element id, symbol name, structural type and analytical model type. We make use of a simple utility method to get all standard family instances for a given category:

...

BuiltInCategory bicSc = BuiltInCategory.OST\_StructuralColumns;

List<Element> columns = RacUtils.GetAllStandardFamilyInstancesForACategory( app, bicSc );

**...**

' Helper to get all Standard Family Instances for a given Category

Shared Function GetAllStandardFamilyInstancesForACategory( \_

ByVal app As Revit.Application, \_

ByVal bic As BuiltInCategory) As List(Of Revit.Element)

' The following code is using the new feature of Element filter to improve

' the performance. It works in Revit 2009 and afterward version

'

Dim elements As New System.Collections.Generic.List(Of Revit.Element)

Dim filterType As Revit.Filter =

app.Create.Filter.NewTypeFilter(GetType(FamilyInstance))

Dim filterCategory As Revit.Filter =

app.Create.Filter.NewCategoryFilter(bic)

Dim filterCombination As Revit.Filter =

app.Create.Filter.NewLogicAndFilter(filterCategory, filterType)

Dim nRetVal As Integer =

app.ActiveDocument.Elements(filterCombination, elements)

Return elements

End Function

Selecting beams and other framing elements is similar and makes use of the OST\_StructuralFraming category.

Foundations are structural elements and also make use of a standard family. Lab 2-2 demonstrates how to retrieve all structural foundation elements. This may not be required for every analysis package. An alternative approach would be to retrieve all standard family instances having structural usage, i.e., having an analytical model.

To retrieve all foundations in the model, we can use the same approach as above, getting all standard structural foundation elements, using the category OST\_StructuralFoundation. This excludes the 'Wall Foundation' system type under 'Structural Foundations' category in the browser, because these belong to the 'Continuous Footing' system family, and also the 'Foundation Slab' system type under the 'Structural Foundations' category in the browser, which are internally implemented as Revit 'Floor' system family. These are examined in Lab 2-3.

As an alternative approach, we can also iterate over the document elements and select all FamilyInstance objects having an analytical model.

The structural elements, walls, floors and footings, also all make use of system families, specifically Wall, Floor (Slab), and Continuous Footing, respectively. Lab 2-3 shows how to retrieve all structural elements from each of these system families. Please note that some instances may not have an analytical model, since this property is user definable.

### Analytical Model

The analytical model makes use of separate classes derived from the abstract base class AnalyticalModel. It defines specialised derived classes for various structural elements:

* Wall AnalyticalModelWall
* Floor AnalyticalModelFloor
* Cont.Footing AnalyticalModel3D
* Column, Framing AnalyticalModelFrame
* Foundation AnalyticalModelLocation

To retrieve all Wall elements having an analytical model, we perform the normal Revit document elements iteration/filtering, pick out wall elements, and check for the presence of an analytical model using the Wall.AnalyticalModel property. The same approach can be used for ContFooting elements.

A similar approach as for walls can also be used for floor elements. However, in the case of floors, the analytical model may well be non-null, but still be empty, so we need to add an additional check that the number of curves in it, given by AnalyticalModelFloor.Curves.Size, is actually greater than zero.

Lab 3 implements code to investigate the geometry contained in the analytical model. It retrieves detailed analytical model geometry for selected elements, depending on their family and/or category. Lab 3 iterates over the currently selected elements and examines all of the following: Wall, Floor, ContFooting, FamilyInstance with category, OST\_StructuralColumns, OST\_StructuralFraming, and OST\_StructuralFoundation. For each of these, it queries the analytical model and lists all of its Curves and Support data.For analysis of the support data, please also refer to the Revit SDK sample AnalyticalSupportData\_Info.

## Structural Analysis Link

Now we have worked out all the pieces we might need to extract information from the Revit Structure model, let us put them together into an actual Revit Structure Analysis link application.

There are many different implementation scenarios in which it might be useful to link a Revit building model with an external data, both RST-specific for analysis as we are discussing here, and also generic for other situations. Obviously, there are also many ways in which such a link can be designed. For the RST analysis link, we need to decide which structural elements are passed between the applications and what kind of model changes are automatically supported in either of them. It depends on third party package and its implementer to design, code and document the link. The Revit Structure API provides the required tools to do this, and does not pre-determine the design of the link application in any way.

For data export/import from/to Revit to/from a third party analysis and design (A&D) application, one will generally use a pair of custom Revit external commands. The format used for this communication is once again completely up to the implementer.

When importing and exporting data, we use element id or uniqueId to keep consistency to identify the elements in the data. SDK sample FireRating and RDBLink, also RSLink used in this class are good examples that demonstrate the usage.

RSLink

We have implemented a custom designed practical example of a Revit Structure analysis link application called RSLink to accompany this presentation. We use AutoCAD with an AutoCAD.NET plug-in to “simulate” the external third party analysis and design application and extended entity data (also known as xdata) to store Revit Structure specific information on standard AutoCAD entities. We use the AutoCAD Dynamic Properties COM API and the Object Property Manager OPM for the link-specific user interface within AutoCAD. Please note that our intention here is to get you started with a link program; there is no actual analysis capability in our “mock up” program.

The data exchange in both directions between the Revit model and the A&D simulation in AutoCAD is implemented using an intermediate, custom designed XML format. For simplicity, the XML format is based on .NET SOAP serialization. Some neutral custom .NET classes are utilised by both sides' export and import commands. Before running the demo, the following applications need to be present:

* RSLink - helper dll shared by AutoCAD and Revit clients
* RSLinkRevitClient - implement two commands for export/import
* RSLinkRevitApp - external application (to add menu)
* RSLinkAcadClient - AutoCAD client
* RSLinkAcadClientDynProps - dynamic Revit properties for AutoCAD objects

To run this, first at Revit side:

1. Set up Revit.ini to load either the client or the external application or both.
2. Open or create a sample model in Revit Structure. Run the export command. You are prompted for a file location. A dialogue box pops up, reporting the number of exported elements.

At AutoCAD side:

1. Start up AutoCAD and load the client and the dynamic property application.
2. The commands defined by the AutoCAD client are RSImport, RSExport and RSMakeMember. Use RSImport to load the xml file just exported from Revit.
3. Modify some cross sections in the model using the OPM, simulating modifications made by the A&D application, and optionally RSMakeMember to add some new elements.
4. Use RSExport to write the model back out again.

Back in Revit:

1. use the import command to read the modifications and update the Revit model accordingly. Dialogue boxes are displayed to report the number of elements imported and modified. Note that the column types were swapped. Currently, the new elements defined by RsMakeMember are ignored.

Additional Resources: MidasLink is a Revit Structure add-in program that exports and imports the Revit model to and from the MIDAS/Gen structure analysis application. The add-in is available to Revit Structure subscription customers in certain goes. The source code (including installer) is available to the members ADN and found on the ADN site.

An overview of some available analysis link packages is provided on the analysis partners site at: <http://usa.autodesk.com/adsk/servlet/item?id=8447050&siteID=123112>

**Revit Structure for Rebar and Detailing**

Another important part of the Revit Structure (or RST) workflow is the production of high-quality construction documents. In particular, the use the RST API to automate parts of the reinforcement workflow. You can generate the 3D rebar model using the API, either from add-on analysis and design software or from built-in macros. You can further drive the framing modeling, generate section and drafting views, import and export DWG files such as schedules, and add text, dimensioning and annotations. In the second half of the talk, we’ll walk through samples and API feature in the rebar workflow context, and explain rebar and rebar shape API.

Revit Structure has support for a 3D rebar model composed of:

* Single bar,
* Array of bars with rules
* Area reinforcement for slabs and walls
* Path reinforcement for slabs

And here is a list of Revit SDK samples that demonstrates rebar modeling APIs:

* Reinforcement – demonstrates the creation of new beam and column reinforcement element.
* NewPathReinforcement - demonstrate how to create a new PathReinforcement element through the API. The PathReinforcement host is a floor or a wall.
* AreaReinParameters - shows how to use the API to display and modify AreaReinforcement parameters. We have expanded the sample to list rebar elements and their parameters as well in a separate command RebarParas.
* BarDescriptions - shows how to find all BarDescriptions in the project, display their properties in a DataGridView, and export their parameter information to an external comma delimited \*.csv file. BarDescription is a property of an AreaReinforcement element, so some AreaReinforcement elements should be drawn first.

Some rebar information is accessible through built-in parameters and can be determined using BuiltInParameterChecker, some is accessible only through hard-coded localised parameter names. The AreaReinParameters sample only works with floor or wall area reinforcements, not with rebar. Not all rebar parameters are accessible as built-in parameters. We can however retrieve almost all necessary information with its instance parameters.

## Detailing

The following is a list of SDK samples that may help you get started with further automate some of detailing tasks:

* CreateViewSection - CreateViewSection demonstrates the creation of a detail view. Given a linear element such as a wall, floor or beam, it generates a section view across the midpoint of the element using the NewViewSection() method.
* ImportExport – this sample has been exhanced since 2008 version to support more DWG. It includes how to export the current project to dwg files and import a dwg file into Revit. It also demonstrates how to and provides a user interface to set up the import and export options appropriately.
* CreateDimensions – This sample demonstrate how to add a dimension to a basic structural walls from the start to the end of each. Dimensioning is created between graphical references stored in a ReferenceArray instance. The key part is the creation of the reference array.
* TagBeam - demonstrates how to add a tag to a beam element.We extended this sample by adding a new external command class TagRebar, which adds a tag to a rebar element as well, and another external command named CreateText to the TagBeam sample.

Notes on Dimensioning

For the basic wall, the location line of the wall is used to discover the two vertical edges at each end of one side of the wall. Some calculation and comparison work needs to be done in order to find the two right reference lines. For each of these two curves, a reference is added to the reference array. Note that the geometry options for obtaining the wall geometry have options.ComputeReferences set to true:

ReferenceArray referenceArray = new ReferenceArray();

Options options = m\_revit.Application.Create.NewGeometryOptions();

options.**ComputeReferences** = true;

options.View = m\_revit.Application.ActiveDocument.ActiveView;

Autodesk.Revit.Geometry.Element element = **wall.get\_Geometry**(options);

GeometryObjectArray geoObjectArray = element.Objects;

//enum the geometry element

for( int j = 0; j < geoObjectArray.Size; ++j )

{

GeometryObject geoObject = geoObjectArray.get\_Item(j);

Curve curve = geoObject as Curve;

if (null != curve)

{

// find the two upright lines beside the line

if (Validate(newLine, curve as Line))

{

referenceArray.Append( **curve.Reference** );

}

if (2 == referenceArray.Size)

{

break;

}

}

}

Once the reference array has been set up, the creation of the dimensioning is simple:

XYZ p1 = newLine.get\_EndPoint(0);

p1.X += 5;

p1.Y += 5;

XYZ p2 = newLine.get\_EndPoint(1);

p2.X += 5;

p2.Y += 5;

Line newLine2 = app.Create.NewLine( ref p1, ref p2, true );

Dimension newDimension = doc.Create.NewDimension(

doc.ActiveView, newLine2, referenceArray );

To extend the sample to work with other elements than structural wall, other methods for determining the references need to be developed. An architectural wall cannot get the reference from an edge. Not all geometry can provide a reference. An alternative is to use a face instead:

Solid solid = geoObject as Solid;

if(null != solid)

{

FaceArrayIterator faceItor = solid.Faces.ForwardIterator();

while (faceItor.MoveNext())

{

PlanarFace face = faceItor.Current as PlanarFace;

if (null != face)

{

// find the two upright lines beside the line.

// we pick up the face in term of its normal.

if( Validate( newLine, face.Normal ) )

{

referenceArray.Append( face.Reference );

}

if( 2 == referenceArray.Size )

{

break;

// . . .

For rebar dimensioning, we need to extend the sample further to work with rebar elements. To do so, we need to dig further into geometry elements:

// get rebar reference

if (m\_rebar != null)

{

Revit.Geometry.Element gelement = m\_rebar.get\_Geometry(options);

GeometryObjectArray geoObjectArray = gelement.Objects;

// enum the geometry element

for (int j = 0; j < geoObjectArray.Size; j++)

{

GeometryObject geoObject = geoObjectArray.get\_Item(j);

Line line = geoObject as Line;

if (line != null && line.Reference != null)

{

referenceArray.Append(line.Reference);

}

}

}

### RebarShape API

In Revit Structure 2009, significant new functionalities are added for modeling concrete reinforcement bars. One major change is that a rebar element is now defined as an instance of RebarShape symbol object. RebarShape is analogous to a Revit Family; it defines the topological layout of a rebar with the lengths of segments driven by parameters. Internally, however, the implementation of RebarShape is separate from the Revit Family functionality and its behavior is different. RebarShape can be provided as content (just like normal standard family). You can also create it through API (unlike normal family which we cannot create programmatically with the currently available API).

RebarShape objects are created through the Revit.Creation.Document.NewRebarShape() method. To avoid specifying a large number of arguments to NewRebarShape(), we first create an object of type Revit.Symbols.RebarShapeDefinition, which is further derived into two types: RebarShapeDefinitionByArc (which uses curves) and RebarShapeDefinitionBySegment (which uses straight line). Parameters used to define RebarShape must be Shared Parameters (Revit.Parameters.ExternalDefinition).

There is a rather elaborate sample called “NewRebar” in the SDK, which demonstrates the usage to Rebar Shape definition as well as creation of a Rebar. However, as this sample has many UI components in it, you may find it a little difficult to digest or to get the essence of how those classes are used. (at least, that was our experience). In the following, we’d like to show you how to define a rebar shape in a form of a bare boned version of shape creation sample using RebarShapeDefinitionBySegments.

The following shows the basic steps to define RebarShapeDefinitionBySegments:

1. Create a RebarShape, e.g.,

RebarShape myShape = rvtDoc.Create.NewRebarShape()

1. Create a RebarShapeDef, e.g.,

RebarShapeDefinitionBySegment myDef   
 = myShape.NewDefinitionBySegments(segmentCount)

1. Add shared parameters by external def, e.g.,

myDef.AddParamater( externalDef, value )

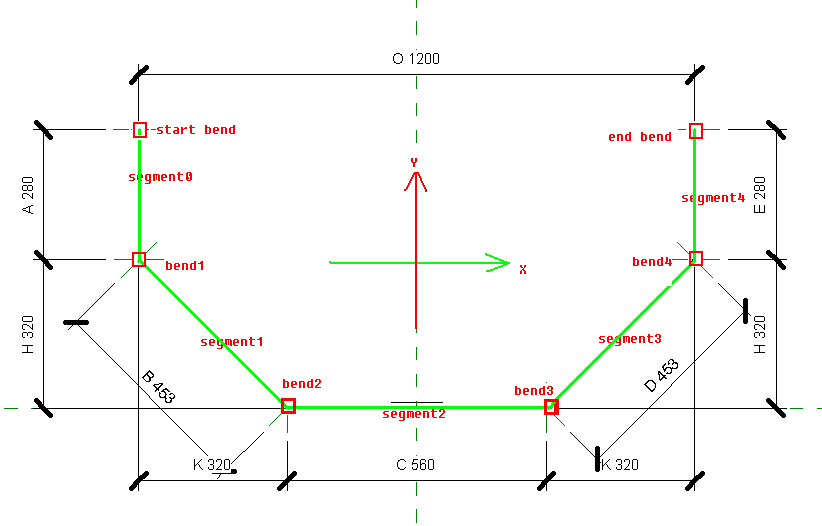
1. Add constraints to shape definition, e.g.,

myDef.AddConstraintToSegment  
myDef.AddConstraintParallelToSegment  
myDef.AddBendDefaultRadius  
myDef.SetSegmentFixedDirection  
myDef.AddListingDimentionBendToBend

1. Set Hook to RebarShape (optional) , e.g.,

myShape.set\_HookAngle  
myShape.set\_HookOrientation

Let’s take an example of a rebar shape like the following picture shows (which is equivalent of a shape documented in “Revit 2009 SDK\Samples\NewRebar\CS\Customize RebarShapes Step by step\ Customize a Segments Shape.doc”.) :



The bear boned version of the code looks something like this:

Public Function Execute(ByVal commandData As ExternalCommandData, ByRef message As String, ByVal elements As ElementSet) As IExternalCommand.Result Implements IExternalCommand.Execute

m\_rvtApp = commandData.Application

m\_rvtDoc = m\_rvtApp.ActiveDocument

' 5 segment sample from SDK (see the illustration in the SDK) Metric.

'

' A | | E \_

' B \\_\_\_\_\_/ D \_ H

' C

' | |

' K

' create a newrebarshape.

'

Dim oRebarShape As RebarShape = m\_rvtDoc.Create.NewRebarShape()

oRebarShape.Name = "MyRebarSegmentSample"

' the real meat of information is under rebarshape definition.

' with # of segment.

Dim shapeDef As RebarShapeDefinitionBySegments = oRebarShape.NewDefinitionBySegments(5)

' add parameters with default value.

Dim def As ExternalDefinition = GetOrCreateSharedParameter("A") ' this is my parameter "A"

Dim pA As Parameter = shapeDef.AddParameter(def, 280)

def = GetOrCreateSharedParameter("B")

Dim pB As Parameter = shapeDef.AddParameter(def, 453)

def = GetOrCreateSharedParameter("C")

Dim pC As Parameter = shapeDef.AddParameter(def, 560)

def = GetOrCreateSharedParameter("D")

Dim pD As Parameter = shapeDef.AddParameter(def, 453)

def = GetOrCreateSharedParameter("E")

Dim pE As Parameter = shapeDef.AddParameter(def, 280)

def = GetOrCreateSharedParameter("H")

Dim pH As Parameter = shapeDef.AddParameter(def, 320)

def = GetOrCreateSharedParameter("K")

Dim pK As Parameter = shapeDef.AddParameter(def, 320)

' here is the one with a formula. search valid formula syntax in product help.

def = GetOrCreateSharedParameter("O")

Dim pO As Parameter = shapeDef.AddFormulaParameter(def, "2\*K+C")

' add constraints

shapeDef.AddConstraintParallelToSegment(0, pA, False, False)

shapeDef.AddConstraintParallelToSegment(1, pB, False, False)

shapeDef.AddConstraintParallelToSegment(2, pC, False, False)

shapeDef.AddConstraintParallelToSegment(3, pD, False, False)

shapeDef.AddConstraintParallelToSegment(4, pE, False, False)

shapeDef.SetSegmentFixedDirection(0, 0.0, -1.0)

shapeDef.SetSegmentFixedDirection(2, 1.0, 0.0)

shapeDef.SetSegmentFixedDirection(4, 0.0, 1.0)

shapeDef.AddConstraintToSegment(1, pH, 0.0, -1.0, 1, False, False)

shapeDef.AddConstraintToSegment(1, pK, 1.0, 0.0, -1, False, False)

shapeDef.AddConstraintToSegment(3, pK, 1.0, 0.0, 1, False, False)

shapeDef.AddConstraintToSegment(3, pH, 0.0, 1.0, -1, False, False)

shapeDef.AddBendDefaultRadius(1, 1, RebarShapeBendAngle.Acute) ' second argument: 1 left, -1 right turn.

shapeDef.AddBendDefaultRadius(2, 1, RebarShapeBendAngle.Acute)

shapeDef.AddBendDefaultRadius(3, 1, RebarShapeBendAngle.Acute)

shapeDef.AddBendDefaultRadius(4, 1, RebarShapeBendAngle.Acute)

' this will add a read-only (gary out) dimention.

shapeDef.AddListeningDimensionBendToBend(pO, 1.0, 0.0, 0, 0, 4, 1)

' set hooks (optional).

oRebarShape.HookAngle(0) = 180

oRebarShape.HookAngle(1) = 180

oRebarShape.HookOrientation(0) = RebarHookOrientation.Left

oRebarShape.HookOrientation(1) = RebarHookOrientation.Left

' finally, call commit. without this, it won't show up in the browser.

Try

shapeDef.Commit()

Catch ex As Exception

MsgBox("failed to commit rebar def: " + ex.ToString)

End Try

Dim result As Boolean = shapeDef.CheckDefaultParameterValues(0, 0)

MsgBox("shapeDef check result = " + result.ToString)

Return IExternalCommand.Result.Succeeded

End Function

Here, GetOrCreateSharedParameter() is a helper function to define a share parameter, similar to what you find in the RAC labs, LabsUtils.GetOrCreateShaedParameterDefinition(). RAC labs are posted together with webcast materials on the ADN site. If you are not familiar with shared parameter, please take a look at labs4 materials or FireRating samples in SDK.

As you can see, the way we defined a rebar shape is quite different from conventional, geometry based approach. In conventional approach, we would simply calculate the location of each vertex to draw lines. In Revit, models are parametric; we will need to think about topology and how each parameter is constrained to define a shape, and then depending on the specific geometrical envelop, Revit will place rebar automatically adjusting to fit within it.

### Learning More

To learn more, please refer to the additional presentations at AU from this and last year. The Revit online help offers a wealth of product related information, and the reference material included in the SDK documentation lists all the classes provided by the API and their methods and properties. The largest Revit programming knowledgebase is provided by the SDK samples. Finally, there are several useful online resources available:

Discussion Groups: [http://discussion.autodesk.com](http://discussion.autodesk.com/) 🡪 Revit Architecture 🡪 Revit API

API Training Classes: [www.autodesk.com/apitraining](http://www.autodesk.com/apitraining)

Autodesk Developer Network: [www.autodesk.com/joinadn](http://www.autodesk.com/joinadn)

DevHelp Online for ADN members: [adn.autodesk.com/](http://www.autodesk.com/joinadn)

### Acknowledgement

The training labs, the structural link sample program and the analysis package simulation using AutoCAD were originally developed by Miroslav Schonauer of Autodesk Consulting. He designed the materials for Revit Structure API training classes he conducted in various countries. We thank Miroslav for sharing the materials with larger Revit API community. Thanks also to Nicolas Mangon, Structural Product Manager, Autodesk, for his input. We thank you Tim Culver of Revit engineering team for his help on understanding RebarShape API.