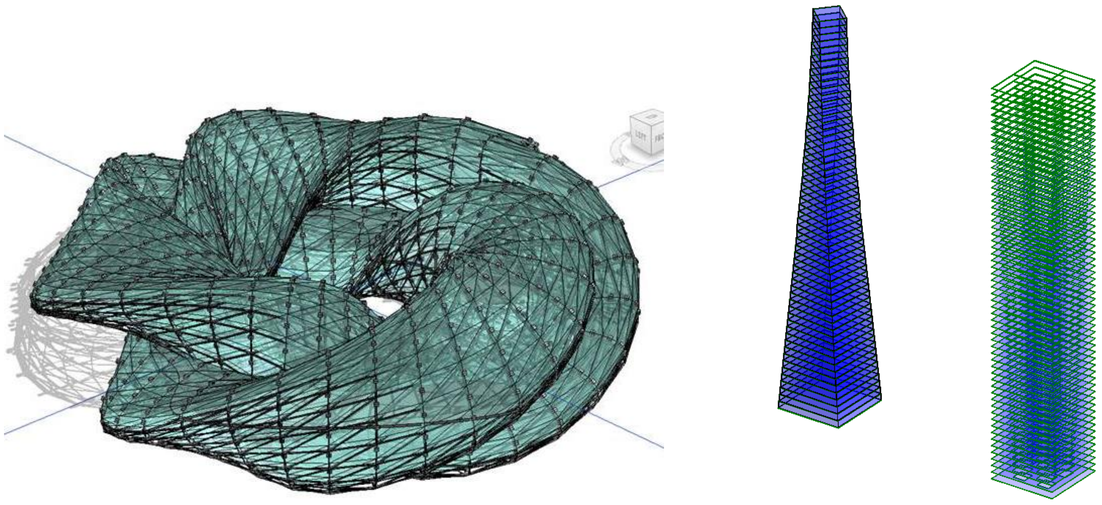
Hummingbird - Revit Addin (ModelBuilder) User Guide

Mario Guttman

2018-05-04

Based on Revit Version 2019



Geometry from Rhino-Grasshopper (left) and Generated Algorithmically in a Custom C# Program (right)

I. Overview

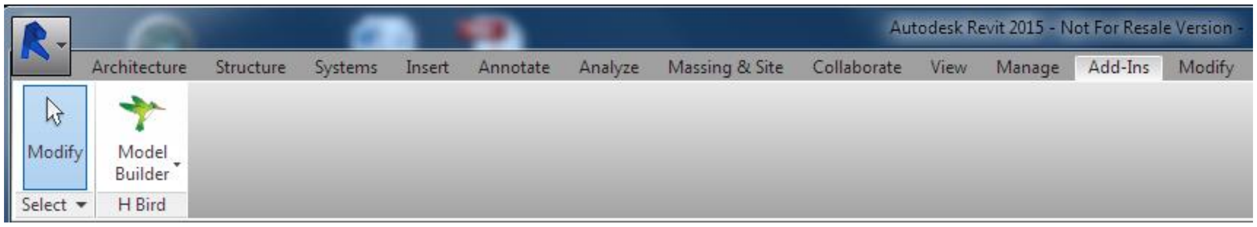
The Revit addin for Hummingbird (Model Builder) is used to build geometry in Revit, based on input from a pseudo-language in a .CSV text file. Its primary purpose is to recreate geometry for another program as native Revit objects, rather than as a linked or imported shape. This is useful when the shapes need to be edited or interact with other Revit objects.

The program can also export Revit objects to the text file. This is currently used primarily for generating sample data but could potentially support workflows into other applications.

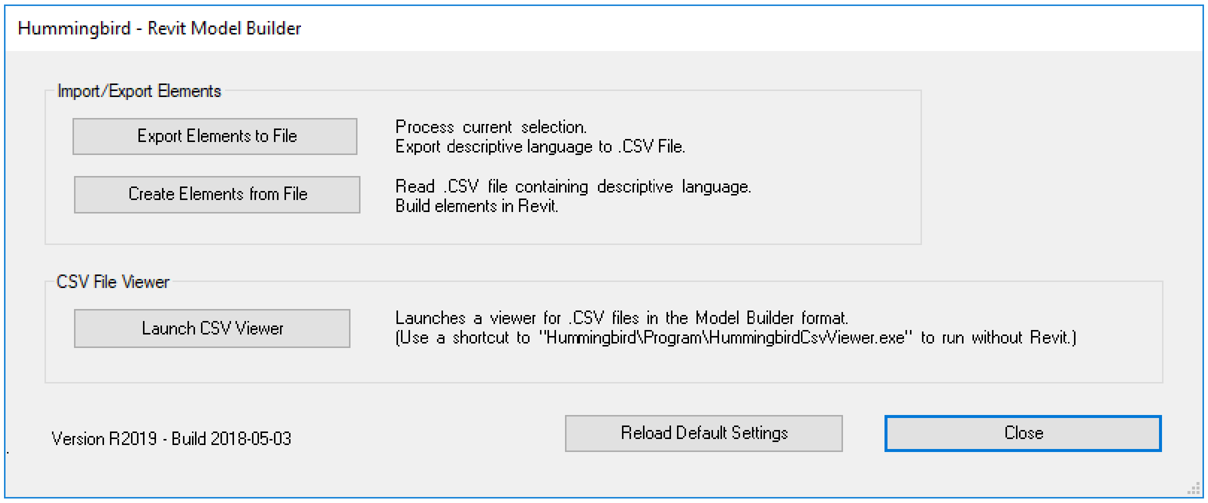
This document includes detail on the syntactical requirements of the text file, and how such a file can be created in Revit; however, it does not address how that file is created in other applications.

The program is focused on creating geometry and utilizes a single "World" coordinate system in Revit. Where possible it does not use the Revit level to place objects or determine their height.

The Revit ModelBuilder add-in is installed to the *Add-Ins* tab in Revit.



The ModelBuilder is installed to the Add-Ins Tab in Revit



Process current selection.
Export descriptive language to .CSV File.

Read .CSV file containing descriptive language.
Build elements in Revit.

Launches a viewer for .CSV files in the Model Builder format.
(Use a shortcut to "Hummingbird\Program\HummingbirdCsvViewer.exe" to run without Revit.)

Version R2019 - Build 2018-05-03

Main Menu

The Main Menu has links to two sub-menus:

Export Elements to File: Using Revit elements to create a new .CSV text file.

Create Elements from File: Using text file data to create new Revit elements.

Each of these is described in a following section.

It also includes a command to launch a specialized .CSV file viewer.

The Revit Model Builder add-in acts as a layer between the Revit modeling environment, including the Family Editor and Conceptual Mass environment, and a .CSV text file.

The text file, which can be viewed in Excel for study if necessary, creates a bridge to other applications. There is no restriction on these other applications since the Model Builder is only aware of the text file. Most of the work to date has been done with data generated from either Rhino-Grasshopper (Hummingbird) or a special C# Model Builder Tool which is a Windows program that serves as a template for custom computational projects. Ports have been proposed for other applications such as Processing, AutoCAD, ArchiCAD, or other applications. If these other applications are used, they will need to customized to create the .CSV output based on their own geometry or other algorithm, and a .DLL library is provided to make this easier.

The primary workflow is for one of these applications to output an Excel worksheet that follows the conventions described in the next section. Then, from an open Revit session, the Model Builder tool reads the Excel data and creates new geometry in Revit.

A secondary workflow is to create geometry in Revit, and then use the Model Builder to export data to a text file. This is useful as a means of creating sample data, which can be edited and then imported to generate new geometry. It can also be used as a way of transferring geometry from Revit objects as drafting or model lines into the Family Editor or another view.

The Hummingbird tools now include an Input tool which can be used to read the CSV data, output from Revit, into Rhino-Grasshopper.

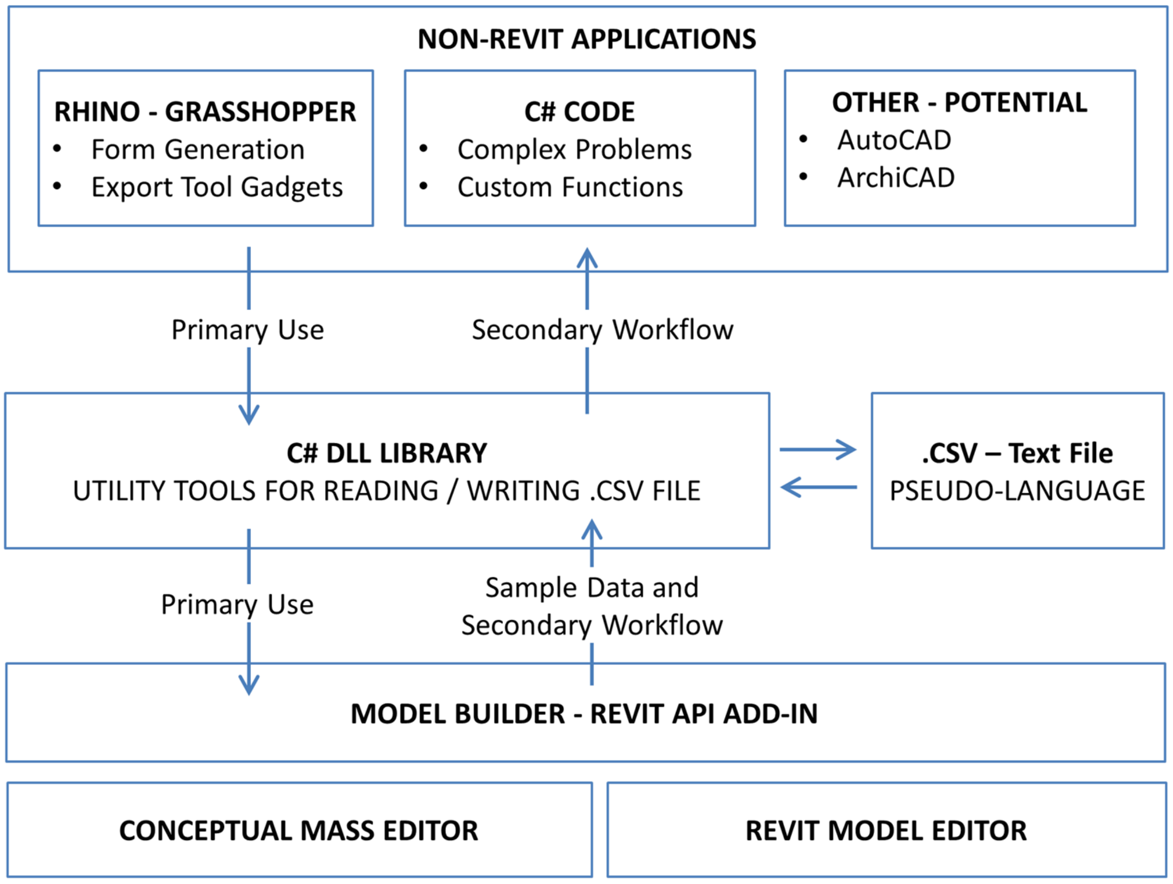
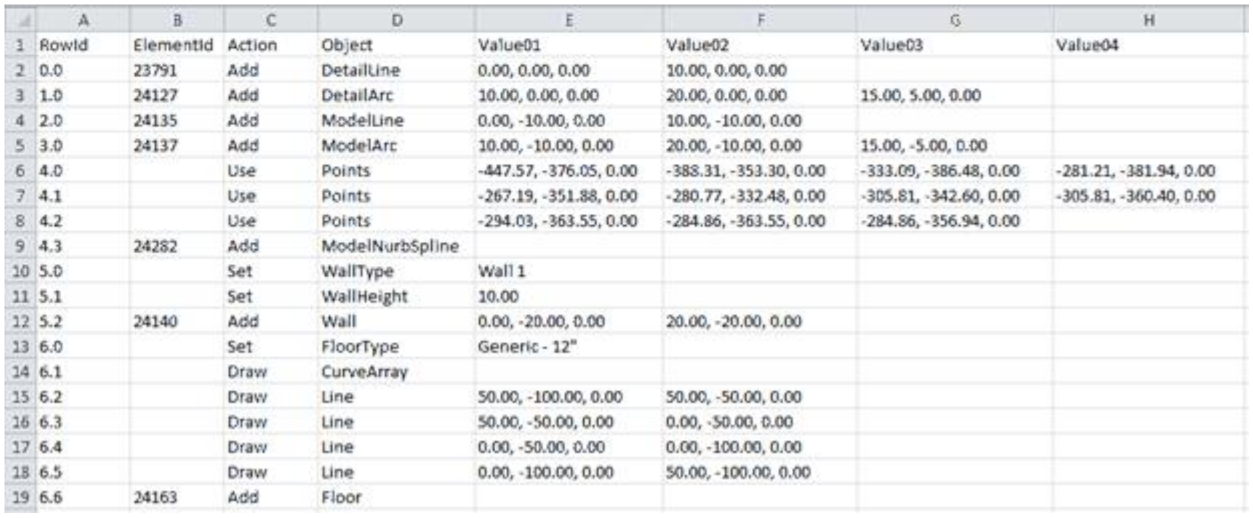


Diagram of the Workflow and Data Model

II. CSV File Data Format

The .CSV file must follow very precise conventions and only limited syntax checking is provided by the program. Generally, if errors are encountered, the program will attempt to continue and report them at the end of the run. This means that some objects or output may not have been created.



	A	B	C	D	E	F	G	H
1	RowId	ElementId	Action	Object	Value01	Value02	Value03	Value04
2	0.0	23791	Add	DetailLine	0.00, 0.00, 0.00	10.00, 0.00, 0.00		
3	1.0	24127	Add	DetailArc	10.00, 0.00, 0.00	20.00, 0.00, 0.00	15.00, 5.00, 0.00	
4	2.0	24135	Add	ModelLine	0.00, -10.00, 0.00	10.00, -10.00, 0.00		
5	3.0	24137	Add	ModelArc	10.00, -10.00, 0.00	20.00, -10.00, 0.00	15.00, -5.00, 0.00	
6	4.0		Use	Points	-447.57, -376.05, 0.00	-388.31, -353.30, 0.00	-333.09, -386.48, 0.00	-281.21, -381.94, 0.00
7	4.1		Use	Points	-267.19, -351.88, 0.00	-280.77, -332.48, 0.00	-305.81, -342.60, 0.00	-305.81, -360.40, 0.00
8	4.2		Use	Points	-294.03, -363.55, 0.00	-284.86, -363.55, 0.00	-284.86, -356.94, 0.00	
9	4.3	24282	Add	ModelNurbSpline				
10	5.0		Set	WallType	Wall 1			
11	5.1		Set	WallHeight	10.00			
12	5.2	24140	Add	Wall	0.00, -20.00, 0.00	20.00, -20.00, 0.00		
13	6.0		Set	FloorType	Generic - 12"			
14	6.1		Draw	CurveArray				
15	6.2		Draw	Line	50.00, -100.00, 0.00	50.00, -50.00, 0.00		
16	6.3		Draw	Line	50.00, -50.00, 0.00	0.00, -50.00, 0.00		
17	6.4		Draw	Line	0.00, -50.00, 0.00	0.00, -100.00, 0.00		
18	6.5		Draw	Line	0.00, -100.00, 0.00	50.00, -100.00, 0.00		
19	6.6	24163	Add	Floor				

Example of Text Data Viewed in Excel

A. General Requirements

Each run uses one file. A folder may contain multiple related files that are run sequentially.

1. Rows

The first row is used for column headings. There are no requirements for these values but the program will assume the columns are in the standard order.

Subsequent rows are read in order. The first empty row is interpreted as the end of the data (irrespective of whether or not additional data or other content follows the blank line.)

2. Columns

The following columns are always used, in this order:

RowId: May be any value or blank; used to help organize the data. The program will ignore except that values, if provided, are included in error messages.

ElementId: Typically blank initially, it will be filled in by the program with the Revit *ElementId* value when an object is created. This value currently has no function but is planned for a future feature that will support the updating of existing objects.

Action: A required field: indicates the process to be initiated.

Object: A required field; indicates that kind of object that will be acted upon.

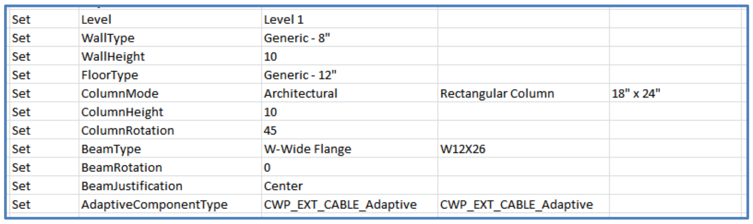
Value01; Value02; Value03; Value04: Required as needed by the specific Action-Object pairs.

B. Action – Object Values

The *Action* value must be one of six proscribed key words: *Set*, *Add*, *Use*, *Draw*, *Model*, or *Modify*. Each may be followed by *Object* values as follows.

1. *Set* Actions

Set commands are generally optional. They set a value that will be used by all subsequent *Add* commands and remain in effect until changed by another *Set* command. If a value is not set, the default model/family setting, or one of the defaults set on the dialog box, will be used.



Set	Level	Level 1		
Set	WallType	Generic - 8"		
Set	WallHeight	10		
Set	FloorType	Generic - 12"		
Set	ColumnMode	Architectural	Rectangular Column	18" x 24"
Set	ColumnHeight	10		
Set	ColumnRotation	45		
Set	BeamType	W-Wide Flange	W12X26	
Set	BeamRotation	0		
Set	BeamJustification	Center		
Set	AdaptiveComponentType	CWP_EXT_CABLE_Adaptive	CWP_EXT_CABLE_Adaptive	

Set – Level: (Value01: String: must be a valid level name in the current project.) The level that will be associated with the new element. (Note that elements are generally placed with a Z-value of 0, in the world coordinate system, irrespective of the level assignment.) Any level may be assigned but it must already exist in the project.

Set – WallType: (Value01: String) The family type name (Value01) to be used when creating walls. The family-type must already exist in the project. (Note that the family is not specified.)

Set – WallHeight: (Value01: Double) The wall will be assigned the given value as an unconnected height, and the top of the wall will not be associated with a level. The value must be a number in decimal feet.

Set – FloorType: (Value01: String) The type name (Value01) to be used when creating floors. The family-type must already exist in the project. (Note that the family is not specified.)

Set – FilledRegionType: (Value01: String) The family type name (Value01) to be used when creating filled regions. The family-type must already exist in the project. (Note that the family is not specified.)

Set – FamilyType: (Value01: String; Value02: String) The family name (Value01) and type name (Value02) to be used when creating general family instances. The family-type must already exist in the project.

Set – FamilyFlipped: (Value01: boolean; Value02: boolean) If a general family instance contains Flip Controls, it will be flipped "Hand" if Value01 is true and "Facing" if Value02 is true. If the family cannot be flipped in one or both directions the directive is ignored without a warning. Note that the Flip actions are done before the mirror and rotation transformations.

Set – FamilyMirrored: (Value01: boolean; Value02: boolean) The general family instances will be mirrored in the X direction (about the Y axis) if Value01 is true; they will be mirrored in the Y direction (about the X axis) if Value02 is true. After the mirroring the original placement is deleted. Note that the transformations are done in the following order: mirror-X, mirror-Y, and then rotation.

Set – FamilyRotation: (Value01: Double) The general family instances will be rotated about their Z axis by the given value, which must be a number in degrees.

Set – ColumnMode: (Value01: String; Value02: String; Value03: String):

Mode: The first value (Value01) is a mode that must be one of the following values: "Architectural", "StructuralVertical", "StructuralPointDriven", or "StructuralAngleDriven", which correspond to similar options in the Revit. The selection influences the number of points required in subsequent Set commands. The point-driven and angle-driven columns are placed in the same way; this setting seems to only affect the way they can be edited later by the user.

Type: The family name (Value02) and type name (Value03) to be used when creating columns. The family-type must already exist in the project and it must be of the corresponding column type as set by the mode.

Set – ColumnHeight: (Value01: Double) A vertical column, which is placed with a single point, will be assigned the given value as an unconnected height, and the top of the column will not be associated with a level. The value must be a number in decimal feet. Slanted columns, which are placed with two points, will ignore this value.

Set – ColumnRotation: (Value01: Double) The column will be rotated about its axis by the given value, which must be a number in degrees.

Set – BeamType: (Value01: String; Value02: String) The family name (Value01) and type name (Value02) to be used when creating floors. The family-type must already exist in the project and it must be a beam.

Set – BeamRotation: (Value01: Double) The beam will be rotated about its axis by the given value, which must be a number in degrees.

Set – BeamJustification: (Value01: String) The value must be one of the following values: "Top", "Center", "Bottom", or "Other". This determines whether the placement points for the beam are interpreted as the top of the beam, centerline, bottom, or other (although not clear what that means.) Note that if several connected beams are rotated they will not clean up properly unless they are center-justified (which is the default.)

Set – AdaptiveComponentType: (Value01: String; Value02: String) The family name (Value01) and type name (Value02) to be used when placing adaptive components. The family-type must already exist in the project and it must be an adaptive component. This selection will affect the number of points that must be supplied in the subsequent Set command.

Set – FamilyExtrusionHeight: (Value01: Double) Subsequent FamilyExtrusions will use Value01 as the height. (Note that the base elevation is determined by the profile.)

Set – MullionType: (Value01: String; Value02: String) The family name (Value01) and type name (Value02) to be used when adding mullions to a curtain wall. The family-type must already exist in the project and it must be a mullion.

2. *Add* Actions – Simple

The simple form of the *Add* actions create an element in a single line command. This form is limited to four values, typically four points.

To create objects that require more points and other complex input see the Add actions that are part of the more complex sets, described in the *Use-Add*, *Draw-Use-Add*, or *Model-Use-Add* Action Sets described in the sections that follow.

Add – Grid: (Value01: Point; Value02: Point). Adds a line-based grid between a start point (Value01) and an end point (Value02).

Add – Grid: (Value01: Point; Value02: Point; Value03: Point). Adds an arc-based grid between a start point (Value01) and an end point (Value02) passing through a third point (Value03).

Add – Level: (Value01: double; Value02: string). Adds an level and the elevation given (Value01). If an optional name (Value02) is provided, the level is renamed. The name must not already exist in the project.

Add – DetailLine: (Value01: Point; Value02: Point). Adds a detail line (or symbolic curve in the Family Editor) between a start point (Value01) and an end point (Value02).

Add – DetailArc: (Value01: Point; Value02: Point; Value03: Point). Adds a detail arc (or symbolic curve in the Family Editor) between a start point (Value01) and an end point (Value02) passing through a third point (Value03).

Add – DetailEllipse: (Value01: Point; Value02: Point; Value03: Double; Value04 String). Adds a detail ellipse (or symbolic curve in the Family Editor) where the Value04 (must be either "Full" or "Half") defines a *Mode*. In the full case the Value01 point is the center of a full ellipse. In the half case, it is the start point of a half-ellipse. In both cases the Value02 point is the opposite end of the main axis, which defines both the length and the rotation. The double Value03 sets the radius in the perpendicular direction.

Add – DetailNurbSpline: (Value01: Point; Value02: Point; [Value03: Point]; [Value04: Point]). Adds a detail NURBS spline (or symbolic curve in the Family Editor) using two, three or four points.

Add – ModelLine: (Value01: Point; Value02: Point). Adds a model line between a start point (Value01) and an end point (Value02).

Add – ModelArc: (Value01: Point; Value02: Point; Value03: Point). Adds a model arc between a start point (Value01) and an end point (Value02) passing through a third point (Value03).

Add – ModelEllipse: (Value01: Point; Value02: Point; Value03: Double; Value04 String). Adds a model ellipse where the Value04 (must be either "Full" or "Half") defines a *Mode*. In the full case the Value01 point is the center of a full ellipse. In the half case, it is the start point of a half-ellipse. In both cases the Value02 point is the opposite end of the main axis, which defines both the length and the rotation. The double Value03 sets the radius in the perpendicular direction.

Add – ModelNurbSpline: (Value01: Point; Value02: Point; [Value03: Point]; [Value04: Point]). Adds a model NURBS spline using two, three or four points.

Add – TopographySurface: (Value01: Point; Value02: Point; Value03: Point; [Value04: Point]). Adds a model TopographySurface using three or four points.

Add – Wall: (Value01: Point; Value02: Point). Adds a straight wall between a start point (Value01) and an end point (Value02).

Add – Wall: (Value01: Point; Value02: Point; Value03 Point). Adds an arc-based wall between a start point (Value01) and an end point (Value02) passing through a third point (Value03).

Add – Floor: (Value01: Point; Value02: Point; Value03: Point; Value04: Point). Adds a floor using four corner points.

Add – FamilyInstance: (Value01: Point). Adds a general family instance at the given point. If the family is workplane-based it will placed on the current work plane with the Z-value of the point as an offset. (Hosted families are not supported.)

Add – Column: (Value01: Point). Adds a vertical column at the given point.

Add – Column: (Value01: Point; Value02: Point). Adds a slanted column between the two given points.

Add – Beam: (Value01: Point; Value02: Point). Adds a beam between the two given points.

Add – AdaptiveComponent: (Value01: Point; Value02: Point; [Value03: Point]; [Value04: Point]). Adds an adaptive component and adjusts the points for two, three, or four points.

Add – AreaBoundaryLine: (Value01: Point; Value02: Point). Adds a straight area boundary line (cannot be used in the Family Editor) between a start point (Value01) and an end point (Value02).

Add – AreaBoundaryLine: (Value01: Point; Value02: Point; Value03). Adds an arc-shaped area boundary line (cannot be used in the Family Editor) between a start point (Value01) and an end point (Value02) passing through a third point (Value03).

Add – AreaBoundaryLine: (Value01: Point; Value02: Point; Value03: Double; Value04 String). Adds an elliptical area boundary line (cannot be used in the Family Editor) where the Value04 (must be "Half"; the "Full" option is not allowed) defines a *Mode*. The Value01 point is the start point of a half-ellipse and the Value02 point is the opposite end of the main axis, which defines both the length and the rotation. The double Value03 sets the radius in the perpendicular direction.

Add – RoomSeparationLine: (Value01: Point; Value02: Point). Adds a straight room separation line (cannot be used in the Family Editor) between a start point (Value01) and an end point (Value02).

Add – RoomSeparationLine: (Value01: Point; Value02: Point; Value03). Adds an arc-shaped room separation line (cannot be used in the Family Editor) between a start point (Value01) and an end point (Value02) passing through a third point (Value03).

Add – RoomSeparationLine: (Value01: Point; Value02: Point; Value03: Double; Value04 String). Adds an elliptical room separation line (cannot be used in the Family Editor) where the Value04 (must be "Half"; the "Full" option is not allowed) defines a *Mode*. The Value01 point is the start point of a half-ellipse and the Value02 point is the opposite end of the main axis, which defines both the length and the rotation. The double Value03 sets the radius in the perpendicular direction.

Add – Area: (Value01). Adds an area (current view must be an area plan) at the given point.

Add – Room: (Value01). Adds a room at the given point.

Add – FilledRegion: (Value01: Point; Value02: Point; Value03: Point; Value04: Point). Adds a filled region of the current type using four corner points. (Note: It is not possible to create a Masking region, which is actually a type of filled region.)

Add – ReferencePoint: (Value01: Point). Adds a Reference Point (only valid in Conceptual Mass family) at the given point.

Add – CurveByPoints: (Value01: Point; Value02: Point; [Value03: Point]; [Value04: Point]). Adds a conceptual model hermite spline (only valid in Conceptual Mass family) using two, three or four points.

3. *Use – Add* Action Sets

Many of the *Add* actions can follow a *Use Set*, which is a format that allows passing more than four points to the command. A Use Set consists of one or more *Use-Points* commands.

Use – Points: (Value01: Point; [Value02: Point]; [Value03: Point]; [Value04: Point]). The values after the first are optional. In the case of multiple lines, only the last one should include missing values. Any empty values should be at the end of the value fields.

It must be terminated with a single *Add DetailNurbsSpline*, *ModelNurbsSpline*, *AdaptiveComponent*, or *Curve By Points* command.

Add – DetailNurbsSpline: (No values.) If the command is run from a Revit Model session it creates a detail line. If the command is run in the Revit Family Editor, it creates a symbolic curve.

Add – ModelNurbsSpline: (No values.) Creates a model NURBS spline.

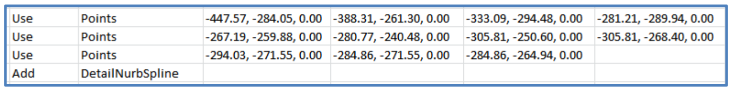
Add – AreaBoundaryLine: (No values.) Creates a NURBS spline-shaped area boundary line (cannot be used in the Family Editor.)

Add – RoomSeparationLine: (No values.) Creates a NURBS spline-shaped room separation line (cannot be used in the Family Editor.)

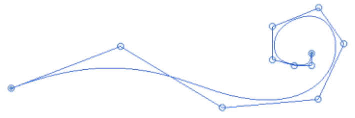
Add – TopographySurface: (No values.) Creates a TopographySurface (cannot be used in the Family Editor.)

Add – AdaptiveComponent: (No values.) Places adaptive component and adjusts the points.

Add – CurveByPoints: (No values.) Creates a 3D hermite spline curve in the conceptual design space.



Use	Points	-447.57, -284.05, 0.00	-388.31, -261.30, 0.00	-333.09, -294.48, 0.00	-281.21, -289.94, 0.00
Use	Points	-267.19, -259.88, 0.00	-280.77, -240.48, 0.00	-305.81, -250.60, 0.00	-305.81, -268.40, 0.00
Use	Points	-294.03, -271.55, 0.00	-284.86, -271.55, 0.00	-284.86, -264.94, 0.00	
Add	DetailNurbSpline				



An Example of a Use Set used with an Add DetailNurbSpline Command

4. *Draw – Use – Add* Action Sets

A *Curve Array Set* consists of a combination of *Draw* and *Use* commands. One or multiple Curve Array Sets are combined into an array of arrays that is used to define the perimeter and openings in a subsequent Floor or Wall command.

a) **Start**

The first line in an Array set must be a *CurveArray* command.

Draw – CurveArray**:** (No values.)

It is followed by additional *Draw* commands or *Use Sets*, which define curves (line, arc, or spline) that together create a closed loop.

b) **Lines and Arcs**

Loops can contain simple curves.

Draw – Line**:** (Value01: Point; Value02: Point.) Adds a line between a start point (Value01) and an end point (Value02) to the current loop.

Draw – Arc**:** (Value01: Point; Value02: Point; Value03: Point.) Adds an arc between a start point (Value01) and an end point (Value02) passing through a third point (Value03) to the current loop.

Draw – Ellipse**:** (Value01: Point; Value02: Point; Value03: Double; Value04 String). Adds an ellipse to the current loop, where the Value04 (must be either “Full” or “Half”) defines a *Mode*. In the full case the Value01 point is the center of a full ellipse. In the half case, it is the start point of a half-ellipse. In both cases the Value02 point is the opposite end of the main axis, which defines both the length and the rotation. The double Value03 sets the radius in the perpendicular direction.

c) **Use Sets**

Loops can also contain NURBS curves, consisting of a *Use Set* followed by a *NurbsSpline* or *HermiteSpline* command.

Use – Points**:** (Value01: Point; Value02: Point; Value03: Point; Value04: Point). The values after the first are optional. In the case of multiple lines, only the last one should include missing values. Any empty values should be at the end of the value fields.

Draw – NurbsSpline: (No values.) This command must follow a *Use Set*. It uses the points of the previous set as input.

Draw – HermiteSpline: (No values.) This command must follow a *Use Set*. It uses the points of the previous set as input.

d) **Curve Array Add Commands**

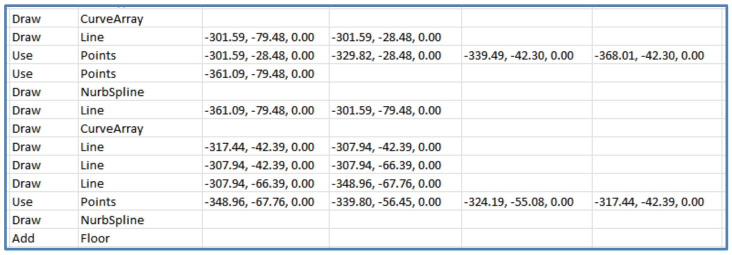
The *Curve Array Set* input is terminated implicitly when an *Add* action is encountered. This action must not have any values and must use the results of the *Curve Array Set* as its input. If more than one *CurveArray* (loop) is defined, the first one is the outer perimeter, and subsequent ones are openings.

Add – Wall: (No values.) Creates a straight wall with a vertical sketch based on curve array set.

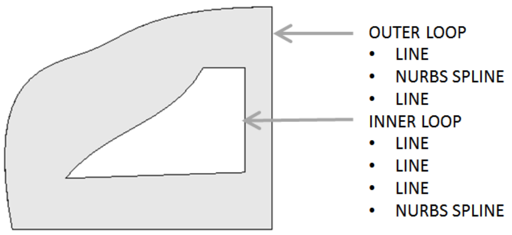
Add – Floor: (No values.) Creates a floor with a perimeter sketch based on the curve array set.

Add – FamilyExtrusion: ([Value01: String]; [Value02: Point]) Creates a new family containing an extrusion, based on the template specified in the ModelBuilder dialog box, using the profile in the curve array set. The new family will be named by Value01 (optional) and this value must not already exist in the model. If not provided a unique name will be generated. The family instance is placed at the point Vlaue02. If the point is omitted it is assumed to be (0, 0, 0). Note that the location of the extrusion is relative to the insertion point.

Add – FilledRegion: (No values.) Adds a filled region of the current type based on curve array set. (Note: It is not possible to create a Masking region, which is actually a type of filled region.)



Draw	CurveArray				
Draw	Line	-301.59, -79.48, 0.00	-301.59, -28.48, 0.00		
Use	Points	-301.59, -28.48, 0.00	-329.82, -28.48, 0.00	-339.49, -42.30, 0.00	-368.01, -42.30, 0.00
Use	Points	-361.09, -79.48, 0.00			
Draw	NurbSpline				
Draw	Line	-361.09, -79.48, 0.00	-301.59, -79.48, 0.00		
Draw	CurveArray				
Draw	Line	-317.44, -42.39, 0.00	-307.94, -42.39, 0.00		
Draw	Line	-307.94, -42.39, 0.00	-307.94, -66.39, 0.00		
Draw	Line	-307.94, -66.39, 0.00	-348.96, -67.76, 0.00		
Use	Points	-348.96, -67.76, 0.00	-339.80, -56.45, 0.00	-324.19, -55.08, 0.00	-317.44, -42.39, 0.00
Draw	NurbSpline				
Add	Floor				



An Example of a Curve Array Set that Includes Two Loops of Lines and Use-Points-NurbSpline Sets

5. ***Model – Use – Add* Action Sets**

A *Reference Array Set* consists of a combination of *Model* and *Use* commands. These are used to establish *Reference Points*, which are used to build *CurveByPoints* curves, which are used to build a *Loft Form*.

a) Start

The first line in a *Reference Array* set must be a *ReferenceArray* command.

***Model – ReferenceArray*:** (No values.)

b) Use Set

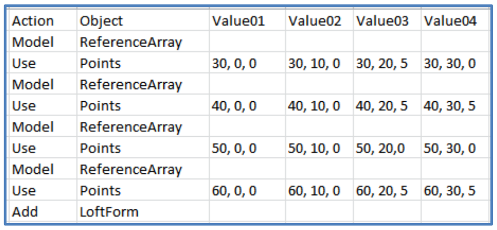
The a *ReferenceArray* command is followed by a *Use Set*, which implicitly defines a *CurveByPoints*. The curve is not explicitly added since it will be absorbed in the final loft form.

***Use – Points*:** (Value01: Point; Value02: Point; Value03: Point; Value04: Point). The values after the first are optional. In the case of multiple lines, only the last one should include missing values. Any empty values should be at the end of the value fields.

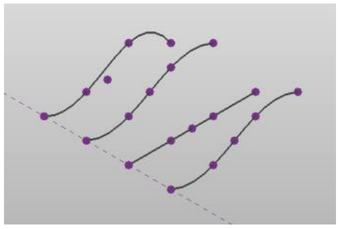
c) Reference Array Add Command

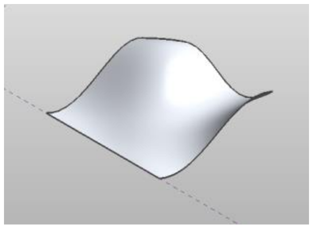
At least two, and possibly more *Reference Array Pairs* must be sequentially contiguous. The set of such pairs is terminated implicitly when an *Add-LoftForm* command is encountered. This must not have any values and must use the results of the *Reference Array Set* as its input.

Add – LoftForm: (No values.) Creates a 3D hermite spline for each *Model-Use* pair, and then uses these curves to create a loft form in the conceptual design space.



Action	Object	Value01	Value02	Value03	Value04
Model	ReferenceArray				
Use	Points	30, 0, 0	30, 10, 0	30, 20, 5	30, 30, 0
Model	ReferenceArray				
Use	Points	40, 0, 0	40, 10, 0	40, 20, 5	40, 30, 5
Model	ReferenceArray				
Use	Points	50, 0, 0	50, 10, 0	50, 20,0	50, 30, 0
Model	ReferenceArray				
Use	Points	60, 0, 0	60, 10, 0	60, 20, 5	60, 30, 5
Add	LoftForm				



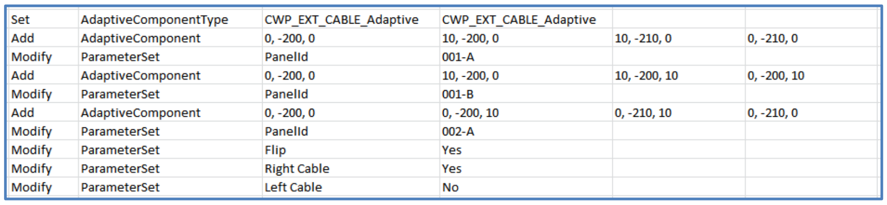


An Example of a Reference Array Set, the Points and CurveByPoints, and the final Loft Form

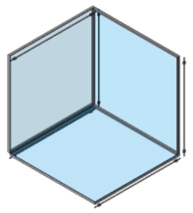
6. *Modify* Action

The *Modify* command is used to set a value on the last element placed.

Modify – ParameterSet: (Value01: String, Value02 Varies). This command (or a set of multiple commands) must immediately follow an Add command. The parameter named by Value01 is set to the value of Value02 in the object created by the previous Add command. The parameter may be of any data type.



Set	AdaptiveComponentType	CWP_EXT_CABLE_Adaptive	CWP_EXT_CABLE_Adaptive		
Add	AdaptiveComponent	0, -200, 0	10, -200, 0	10, -210, 0	0, -210, 0
Modify	ParameterSet	PanelId	001-A		
Add	AdaptiveComponent	0, -200, 0	10, -200, 0	10, -200, 10	0, -200, 10
Modify	ParameterSet	PanelId	001-B		
Add	AdaptiveComponent	0, -200, 0	0, -200, 10	0, -210, 10	0, -210, 0
Modify	ParameterSet	PanelId	002-A		
Modify	ParameterSet	Flip	Yes		
Modify	ParameterSet	Right Cable	Yes		
Modify	ParameterSet	Left Cable	No		



An Example of Placing Three Adaptive Components and Modifying the Last One

Modify – CurtainGridUAdd: (Value01: Double). This command (or a set of multiple commands) must immediately follow an Add command. A curtain grid is added in the U direction with and offset based on Value01, the Primary Offset, which is a number between 0 and 1.0 indicating a proportion of the wall dimension.

Modify – CurtainGridUAdd: (Value01: Double , Value02 Double) Similar to above with Value02, the Secondary Offset, indicating a point along the grid line where the grid is inserted. The resulting grid will only extend as far as the adjacent perpendicular grids.

Modify – CurtainGridVAdd: (Value01: Double). Similar in the V direction.

Modify – CurtainGridVAdd: (Value01: Double , Value02 Double) Similar.

Modify – MullionUAdd: (Value01: Double). Similar but also places a mullion on the grid.

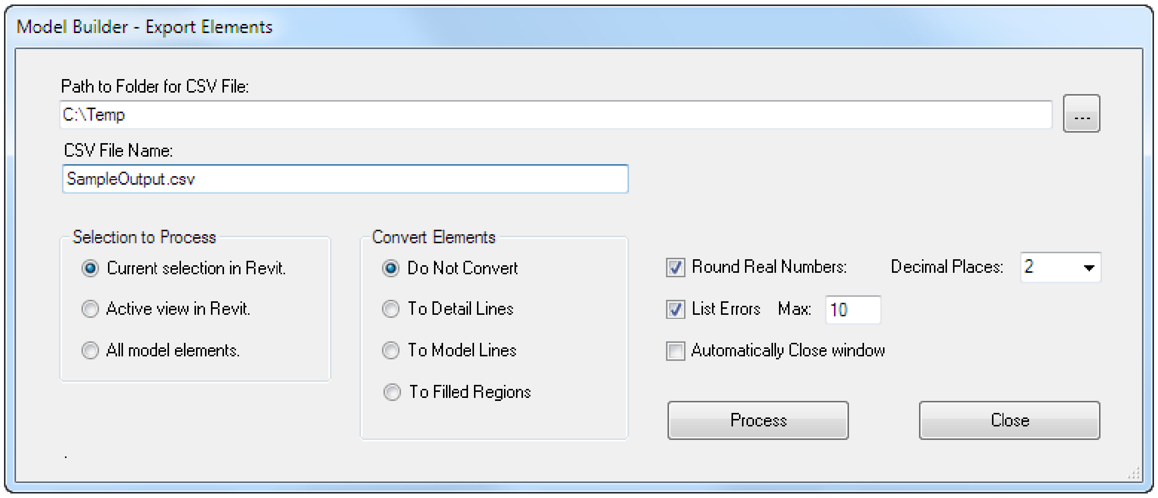
Modify – MullionUAdd: (Value01: Double , Value02 Double) Similar.

Modify – MullionVAdd: (Value01: Double). Similar.

Modify – MullionVAdd: (Value01: Double , Value02 Double) Similar.

III. Exporting Elements to CSV File

The *Export Elements to File* menu is used to create new text data from Revit model elements. When it is run, if the text file already exists it is overwritten. Otherwise a new file is created.



Path to Folder for CSV File: C:\Temp

CSV File Name: SampleOutput.csv

Selection to Process: [✓ Current selection in Revit.] Active view in Revit. All model elements.

Convert Elements: [✓ Do Not Convert] To Detail Lines To Model Lines To Filled Regions

☑ Round Real Numbers: Decimal Places: 2

☑ List Errors Max: 10

☐ Automatically Close window

Export Elements to File Menu

A. Settings

Path to Folder for CSV File: The full path to an existing folder where the .CSV file will be created. Use the Browse button (three dots) to select an existing folder.

CSV File Name: The name of a new or existing .CSV file. If the file already exists it will be overwritten. The extension is optional; it will be added if it is missing.

Selection to Process: The Revit elements to be processed may be pre-selected before running the Model Builder command. Alternatively, everything in the current view, or in the current model may be selected automatically. In either case, elements that the Model Builder cannot export will be ignored.

Convert Elements: The exported elements may retain their original Revit object type, or they may be converted to simpler forms: detail lines, model lines, or filled regions. Generally, with converted complex objects such as walls, the lines used are the definition lines, not the actual geometry. With the filled regions option only elements that can define a region are converted.

Round Real Numbers: The values exported can be rounded to a specific number of decimal places to make the output more legible.

Automatically Closing Window: If the checkbox is selected, the dialog box will close automatically after the command is run.

B. Processing

Once the command is started, all of the selected elements are processed and written to the file.

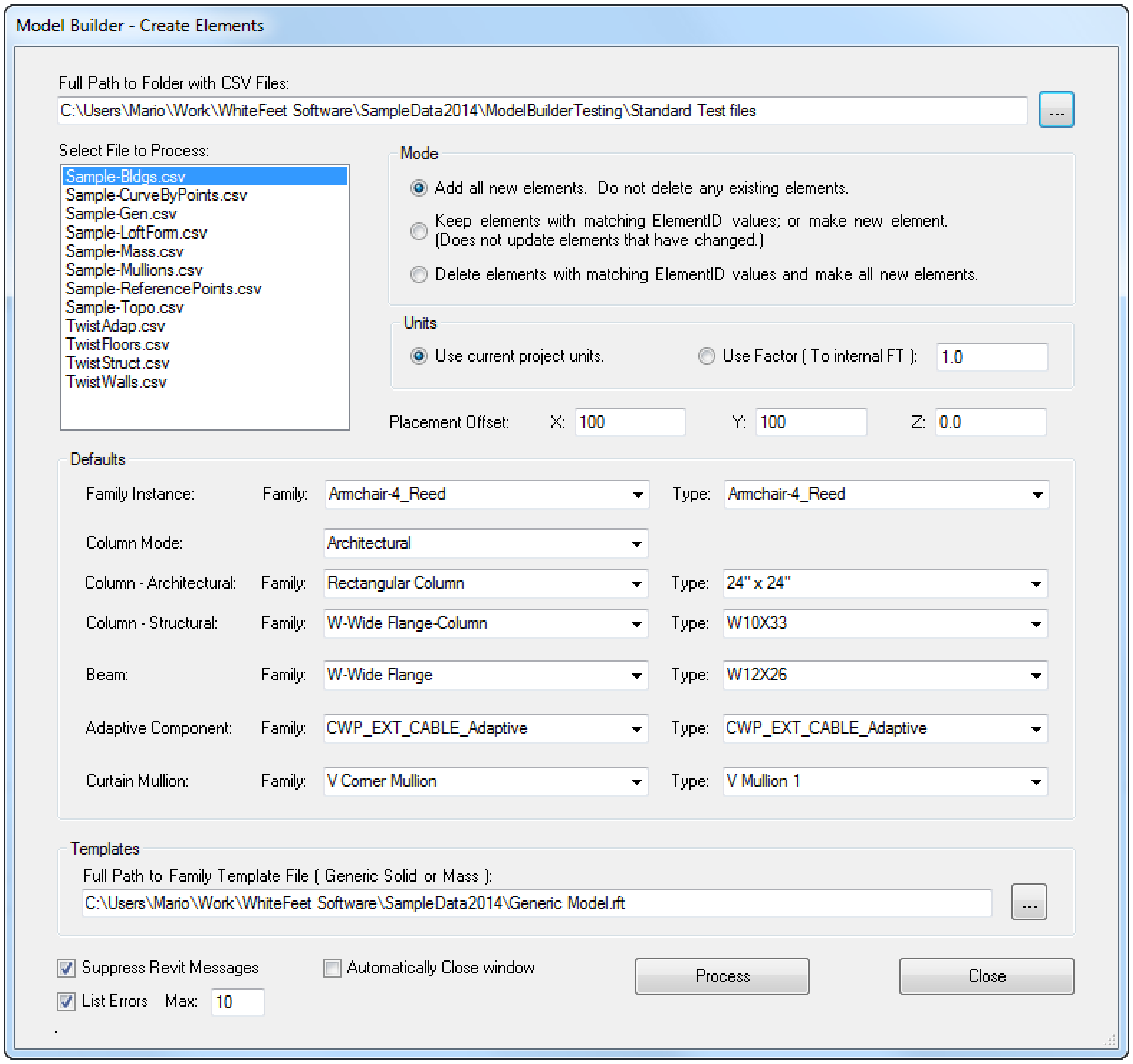
During the export the dialog box disappears and a progress bar is displayed. It includes a button to cancel the process.

When the processing is complete, the Progress Bar will close. If the option is selected, and any errors occurred, a dialog showing them is displayed.

Examples of the similar progress bar and error dialog are shown in the following section.

IV. Creating Elements From CSV File

The *Create Elements from File* menu is used to create new Revit elements from text data. When it is opened, if a folder was previously specified, it is scanned for existing .CSV text files to process.



Model Builder - Create Elements

Full Path to Folder with CSV Files: C:\Users\Mario\Work\WhiteFeet Software\SampleData2014\ModelBuilderTesting\Standard Test files

Select File to Process:
- Sample-Bldgs.csv
- Sample-CurveByPoints.csv
- Sample-Gen.csv
- Sample-LoftForm.csv
- Sample-Mass.csv
- Sample-Mullions.csv
- Sample-ReferencePoints.csv
- Sample-Topo.csv
- TwistAdap.csv
- TwistFloors.csv
- TwistStruct.csv
- TwistWalls.csv

Mode
- [✓] Add all new elements. Do not delete any existing elements.
- Keep elements with matching ElementID values; or make new element. (Does not update elements that have changed.)
- Delete elements with matching ElementID values and make all new elements.

Units
- [✓] Use current project units.
- Use Factor (To internal FT): 1.0

Placement Offset: X: 100 Y: 100 Z: 0.0

Defaults

Family Instance:	Family:	Armchair-4_Reed	Type:	Armchair-4_Reed
Column Mode:		Architectural		
Column - Architectural:	Family:	Rectangular Column	Type:	24" x 24"
Column - Structural:	Family:	W-Wide Flange-Column	Type:	W10X33
Beam:	Family:	W-Wide Flange	Type:	W12X26
Adaptive Component:	Family:	CWP_EXT_CABLE_Adaptive	Type:	CWP_EXT_CABLE_Adaptive
Curtain Mullion:	Family:	V Corner Mullion	Type:	V Mullion 1

Templates

Full Path to Family Template File (Generic Solid or Mass): C:\Users\Mario\Work\WhiteFeet Software\SampleData2014\Generic Model.rft

- [✓] Suppress Revit Messages
- [ ] Automatically Close window
- [✓] List Errors Max: 10

Create Elements From File Menu

A. Settings

Full Path to Folder with .CSV Files: The full path to an existing folder. Use the Browse button (three dots) to select a folder.

Select File to Process: An existing .CSV file that will be read. It must be selected from the list.

Mode: Controls whether all new elements will be created or if existing elements will be modified.

Units: Converts the length units used in creating the text file to Revit units. Note that Revit always stores dimensions in feet internally; it only appears to use different units when it displays them. For this reason, the ModelBuilder program always converts the external units to feet. It has two options for doing this:

- Use Current Project Units: Assumes that the source data was created using the same project units set to display in Revit. For example, if Rhino was set to meters when the data was created, and Revit is also set to meters, the Rhino values will be converted from meters to feet.
- Use Factor: If a special conversion is required the program will multiply by the value in the text box. For example, if Rhino was set to meters when the data was created, and Revit is set to some other units, a value of 0.3048 should be used to convert the Rhino units to feet, which will then display according to the current Revit units setting.

Placement Offset: These values are used to offset the placement of the Revit elements as they are created. This can be used to place alternative versions in a row in the same Revit model, or to position the geometry for some other reason.

Defaults: Some of the values that would otherwise be established by a *Set* command can be defined as defaults. This value will be used until it is overwritten by a *Set* command. This helps to avoid the use of *Set* commands that refer to invalid families and other problems. It also provides flexibility in changing settings without editing the text file. However, unlike the use of *Set* commands, only one value can be used during a given run.

Suppress Revit Messages: In some cases, such as when lines that are slightly off of axis, Revit will display a warning dialog and halt processing until the user responds. Checking this box prevents such messages from displaying. If the problem is critical, the element that caused it will not be created. If the problem is not critical the warning will be recorded but not displayed.

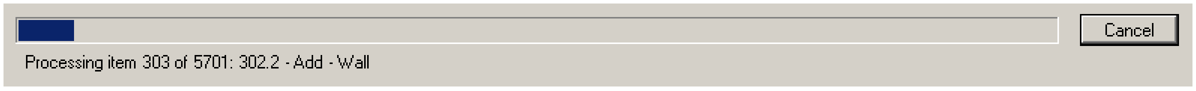
List Errors: If the checkbox is selected, errors, up to maximum number, will be displayed at the end of the run.

Automatically Closing Window: If the checkbox is selected, the dialog box will close automatically after the command is run.

B. Processing

Once the command is started, the text file is read and elements are created in Revit.

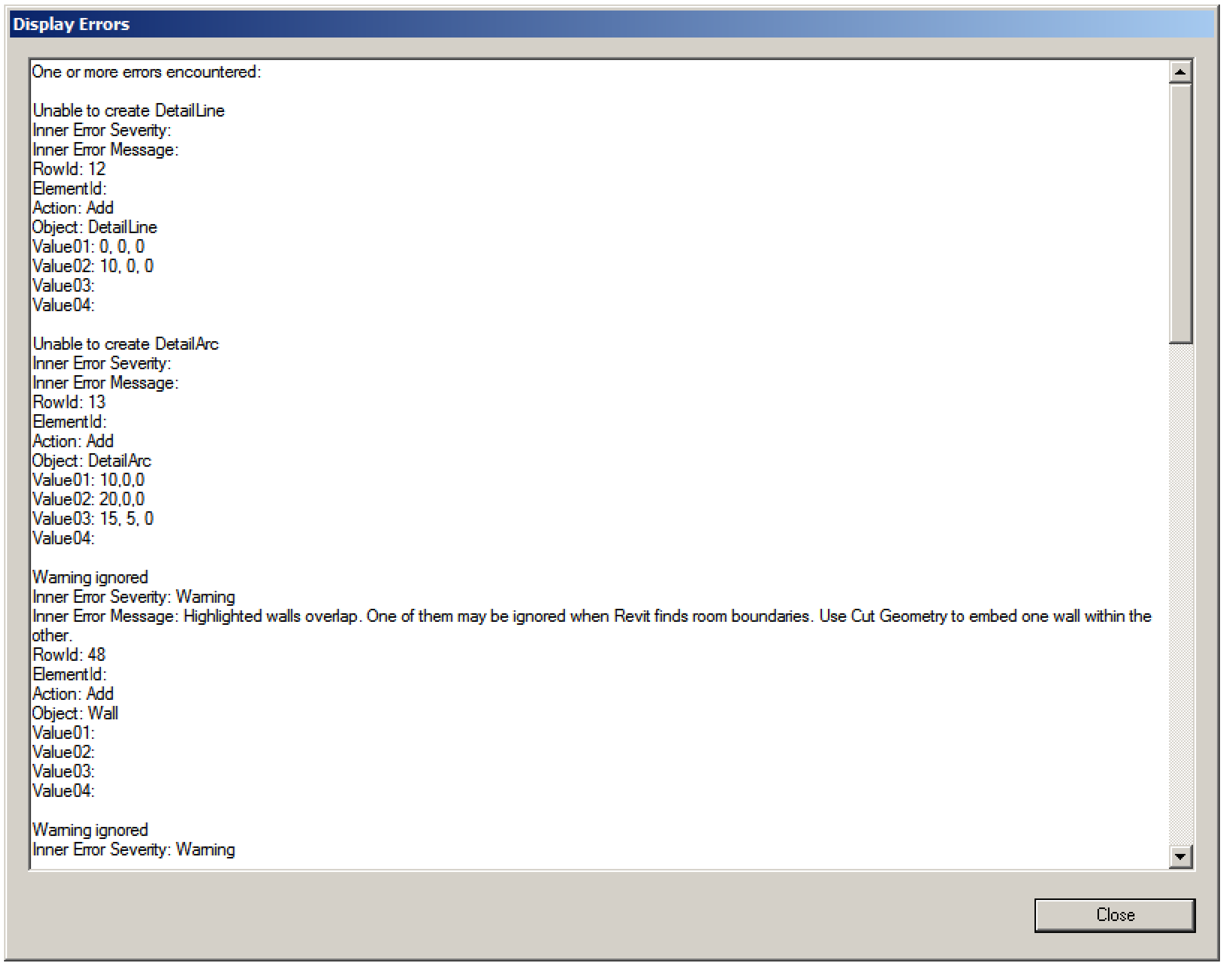
During the export the dialog box disappears and a progress bar is displayed. It includes a button to cancel the process.



Processing item 303 of 5701: 302.2 - Add - Wall

The Progress Bar while Creating Elements

When the processing is complete, the Progress Bar will close. If the option is selected, and any errors occurred, a dialog showing them is displayed.

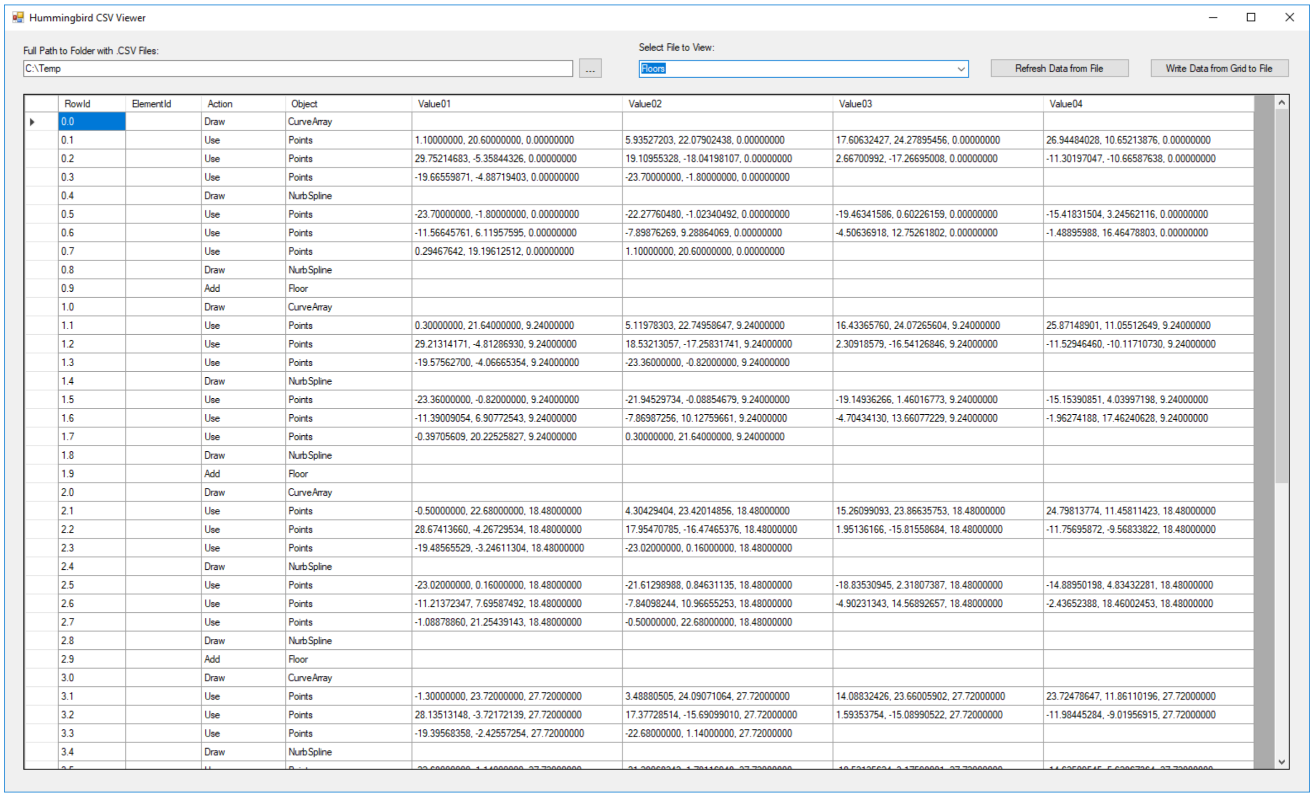


```
One or more errors encountered:

Unable to create DetailLine
Inner Error Severity:
Inner Error Message:
RowId: 12
ElementId:
Action: Add
Object: DetailLine
Value01: 0, 0, 0
Value02: 10, 0, 0
Value03:
Value04:

Unable to create DetailArc
Inner Error Severity:
Inner Error Message:
RowId: 13
ElementId:
Action: Add
Object: DetailArc
Value01: 10,0,0
Value02: 20,0,0
Value03: 15, 5, 0
Value04:

Warning ignored
Inner Error Severity: Warning
Inner Error Message: Highlighted walls overlap. One of them may be ignored when Revit finds room boundaries. Use Cut Geometry to embed one wall within the
other.
RowId: 48
ElementId:
Action: Add
Object: Wall
Value01:
Value02:
Value03:
Value04:

Warning ignored
Inner Error Severity: Warning
```

The Display Errors Dialog

V. CSV File Viewer

The Hummingbird CSV File Viewer is a stand-alone Windows application that can be used to view the .CSV files created during the Hummingbird process.



Full Path to Folder with .CSV Files: C:\Temp

Select File to View: Floors

RowId	ElementId	Action	Object	Value01	Value02	Value03	Value04
0.0		Draw	CurveArray				
0.1		Use	Points	1.10000000, 20.60000000, 0.00000000	5.93527203, 22.07902438, 0.00000000	17.60632427, 24.27895456, 0.00000000	26.94484028, 10.65213876, 0.00000000
0.2		Use	Points	29.75214683, -5.35844326, 0.00000000	19.10955328, -18.04198107, 0.00000000	2.66700992, -17.26695008, 0.00000000	-11.30197047, -10.66587638, 0.00000000
0.3		Use	Points	-19.66559871, -4.88719403, 0.00000000	-23.70000000, -1.80000000, 0.00000000		
0.4		Draw	NurbSpline				
0.5		Use	Points	-23.70000000, -1.80000000, 0.00000000	-22.27760480, -1.02340492, 0.00000000	-19.46341586, 0.60226159, 0.00000000	-15.41831504, 3.24562116, 0.00000000
0.6		Use	Points	-11.56645761, 6.11957595, 0.00000000	-7.89876269, 9.28864069, 0.00000000	-4.50636918, 12.75261802, 0.00000000	-1.48895988, 16.46478803, 0.00000000
0.7		Use	Points	0.29467642, 19.19612512, 0.00000000	1.10000000, 20.60000000, 0.00000000		
0.8		Draw	NurbSpline				
0.9		Add	Floor				
1.0		Draw	CurveArray				
1.1		Use	Points	0.30000000, 21.64000000, 9.24000000	5.11978303, 22.74958647, 9.24000000	16.43365760, 24.07265604, 9.24000000	25.87148901, 11.05512649, 9.24000000
1.2		Use	Points	29.21314171, -4.81286930, 9.24000000	18.53213057, -17.25831741, 9.24000000	2.30918579, -16.54126846, 9.24000000	-11.52946460, -10.11710730, 9.24000000
1.3		Use	Points	-19.57562700, -4.06665354, 9.24000000	-23.36000000, -0.82000000, 9.24000000		
1.4		Draw	NurbSpline				
1.5		Use	Points	-23.36000000, -0.82000000, 9.24000000	-21.94529734, -0.08854679, 9.24000000	-19.14936266, 1.46016773, 9.24000000	-15.15390851, 4.03997198, 9.24000000
1.6		Use	Points	-11.39009054, 6.90772543, 9.24000000	-7.86987256, 10.12759661, 9.24000000	-4.70434130, 13.66077229, 9.24000000	-1.96274188, 17.46240628, 9.24000000
1.7		Use	Points	-0.39705609, 20.22525827, 9.24000000	0.30000000, 21.64000000, 9.24000000		
1.8		Draw	NurbSpline				
1.9		Add	Floor				
2.0		Draw	CurveArray				
2.1		Use	Points	-0.50000000, 22.68000000, 18.48000000	4.30429404, 23.42014856, 18.48000000	15.26099093, 23.86635753, 18.48000000	24.79813774, 11.45811423, 18.48000000
2.2		Use	Points	28.67413660, -4.26729534, 18.48000000	17.95470785, -16.47465376, 18.48000000	1.95136166, -15.81558684, 18.48000000	-11.75695872, -9.56833822, 18.48000000
2.3		Use	Points	-19.48565529, -3.24611304, 18.48000000	-23.02000000, 0.16000000, 18.48000000		
2.4		Draw	NurbSpline				
2.5		Use	Points	-23.02000000, 0.16000000, 18.48000000	-21.61298988, 0.84631135, 18.48000000	-18.83530945, 2.31807387, 18.48000000	-14.88950198, 4.83432281, 18.48000000
2.6		Use	Points	-11.21372347, 7.69587492, 18.48000000	-7.84098244, 10.96655253, 18.48000000	-4.90231343, 14.56892657, 18.48000000	-2.43652388, 18.46002453, 18.48000000
2.7		Use	Points	-1.08878860, 21.25439143, 18.48000000	-0.50000000, 22.68000000, 18.48000000		
2.8		Draw	NurbSpline				
2.9		Add	Floor				
3.0		Draw	CurveArray				
3.1		Use	Points	-1.30000000, 23.72000000, 27.72000000	3.48880505, 24.09071064, 27.72000000	14.08832426, 23.66005902, 27.72000000	23.72478647, 11.86110196, 27.72000000
3.2		Use	Points	28.13513148, -3.72172139, 27.72000000	17.37728514, -15.69099010, 27.72000000	1.59353754, -15.08990522, 27.72000000	-11.98445284, -9.01956915, 27.72000000
3.3		Use	Points	-19.39568358, -2.42557254, 27.72000000	-22.68000000, 1.14000000, 27.72000000		
3.4		Draw	NurbSpline				

There is a button on the Hummingbird-Revit menu to open it. Alternatively, users may create a Windows shortcut to it and run it outside of Revit. (The file to use for the shortcut is:

HummingbirdCsvViewer.exe

It is located in the folder:

C:\Users\<UserName>\AppData\Roaming\Autodesk\Revit\Addins\****\Hummingbird|Program\

Viewing the .CSV file can be a very important step in resolving problems. By examining this file, and editing it manually, one can determine if the problem is with the output from the source program, or the in put to the target program.