
November 30 – December 3, 2004 ◊ Las Vegas, Nevada

# Building Stronger Families

David Conant – Product Designer, Autodesk, Inc.
Steve Stafford, WATG

**BD 35-3** **Discuss the best techniques for getting the maximum performance and flexibility from your Autodesk® Revit® families. This course will outline advanced uses of existing functionality and introduce new features now available in Autodesk Revit 7.0. Specific examples will be shown and distributed to attendees.**

**Who Should Attend**

**Intermediate-level users of Autodesk Revit**

**Topics Covered**

*** Adding your own custom data to families**
*** Effective use of formulas**
*** Conditionals in Formulas**
*** Controlling nested families**
*** Using level of detail to add graphic power**

## About the Speaker:

David was the first architect to work for Revit Technology Corp. He helped develop the initial design of the Autodesk® Revit® software and has been a key product designer ever since. He applies his many years of experience as a practicing architect to keep the Revit team focused on real architectural needs and high levels of usability. Prior to working at Revit, David worked as an architect and CAD manager in Cambridge, Massachusetts. He developed and implemented office-wide standards, customized AutoCAD® systems, and implemented an ongoing staff training system. His experience also includes 5 years experience teaching the use of computing in design and modeling as a faculty member in Northeastern University's Department of Art and Architecture.

david.conant@autodesk.com

# Introduction

Families are one of the most important pieces of Autodesk Revit. They give Revit users the ability to define sophisticated parametric building components without programming. The information embedded in families transforms a building model into a building information model. Because they are very powerful much of their potential lies untapped by many users. In this session, we will outline several techniques available to build more sophisticated and powerful families as well as some underlying strategies and methods to maximize quality and performance.

The Revit family goes beyond the blocks, cells, etc. of older CAD systems. It reflects the real world where

# Standards and Testing

## Plan for Success

Your success with family creation depends greatly on planning. With correct planning, you will improve your productivity in family creation. Your families will provide the greatest data and graphic benefits with the lowest performance cost. Before you start building a family, consider the following factors:

1. Where will the family be viewed?
   - Plans only
   - Plans and elevations
   - Plans, elevations, and 3d views
   - 3d views only
2. Will the family ever be seen in 3 dimensions?
3. What scales will the family appear at in deliverable documents?
4. In typical representations, how detailed would the representation be if you were not using Revit?
5. How does this class of components vary in construction practice?
   - Every instance has distinct individual characteristics
   - Instances of the same type are identical
   - Some parts of each instance of a type are identical, and others vary

### Don’t get carried away

It is common for new Revit users to feel that they are supposed to “model everything” in full 3d and that failing to do so is somehow a failure to use the tool properly. Following this road usually leads to frustration and poor performance in the application. It is better to remember that a family is only a means to and. It’s function is to show someone what component is to be used, where it is located and sufficient information to get it placed properly. Developing families beyond that point adequate for those needs is usually a poor time investment.

1. Determine the parameter scheme
   - What are the realistic variations (don’t create parameters to provide for variations that aren’t likely to occur)
   - What values will need to appear on schedules or exports (don’t use shared parameters unless data needs to be on schedules)
2. Determine the necessary level of detail

- Will the element ever be seen in close up 3d views? If not, avoid the temptation to build more than an abstract 3d version. If it is never seen in 3d views, consider making only 2d representations.
- Is the element seen objectively or symbolically in plans? If it is usually symbolic, consider using view specific elements such as detail lines for most of the representation>
- Does the same element appear with different levels of detail at different view scales? Consider adding scale dependent symbolic elements to create the more detailed view rather than adding fully detailed 3d elements.

3. Consider what the truly important parts of the family are. Don't waste time on things that will not be seen on documents and views.
4. Plan what data you will want to get out of a family in schedules and ODBC.
   - Any data you want access to should either use Revit supplied or shared parameters.
   - Construct a parameter scheme in advance mapping out parameter names, types, etc. to help ensure consistency between families.

### Test for Reliability

Good families operate reliably for users. In order to ensure reliability, you should test your families continually as they are being built.

#### Build the bones first

Construct the major references and forms first. Parameterize at this state and test before adding additional elements. Ideally, all major geometric parameters should be built at this stage.

#### Flex flex flex

Test all parameters at each phase as construction proceeds. Look for elements that don't move or vary as planned or pieces that become detached.

#### Know the limits

When testing test settings for parameters that are outside the normal range. In many cases, when parameters greatly exceed normal, failures will occur as parts reach 0 size or faces pass through each other.

### Set Standards

Setting and maintaining good standards will make families more reliable and give better data in schedules and exports. The attached standards document is one we use in house to guide family builders, It can serve as a good basis for one of your own.

## Unlock the Power

Once you have mastered the basics of constructing forms and straightforward families, Revit provides many features for building more powerful families.

### Shared Parameters

Shared parameters are the key to extending the data model to embed and report your data. Because Revit is a tightly constructed database, features such as schedules and tags depend on "normalized" data for their correct function. Shared parameters are defined in an external file so they can be accessed by multiple projects and families. They are given unique identifiers so that the application can recognize them as the same wherever they are used.

### Usage

Shared parameters can be used across families and categories. Use them to add your own data to schedules, tag that data, create multi category schedules, and include the data in ODBC exports. For example, an electrical connection parameter can be added to casework, specialty equipment, and plumbing fixtures so that all these elements can appear on a schedule showing what instances have electrical requirements. Since they require extra levels of work, they should not be used unless required.

### Methodology

- Create and manage shared parameters using File|Shared Parameters.
- Shared Parameter definitions are stored in special text files. Select an existing file or create a new one.
- Create Parameter groups. These are user defined logical groups of parameters that make it simpler for users to find the correct ones and load them.
- Set a Parameter Name. Be careful about spelling and case. Once created, a shred parameter cannot be renamed.
- Define the type of data the parameter can store. Like the name, this cannot be altered once the parameter is defined.
- To use a shared parameter in a family, select ADD from the Parameters section of the Family Types dialog, then select the shared parameter option on the Parameter Properties dialog.
- To add a shared parameter to the project, or system families, use Settings|Project parameters and choose to ADD a parameter. Select the categories the parameter will be added to and then select Shared Parameter. Proceed as for families.
- We recommend that you use only one shared parameter file for your firm. Each shared parameter definition has a unique identifier. Parameters with the same name coming from different shared parameter files are regarded by Revit as different parameters. Using a single parameter file helps prevent this duplication.
- To display a shared parameter in single category schedules, add them as a project parameter to the desired category. This ensures that all elements of that category have access to the parameter. (not all need to have a value)
- To show only certain instances or types of a category in a schedule, create a Yes/No shared parameter “Schedule” and add to the desired categories as a project parameter. Instances or types can then be designated to schedule or not. In the schedule filter the schedule to display only items where “Schedule” = Yes.
- To schedule which nested family types are used in a host family, create a shared parameter of type Family/Type. A typical example would be a door schedule that shows which hardware set is used from four types available to each door.

## Formulas

Formulas allow you to create parameters that depend on other parameters for their value. A simple example would be a width parameter set to equal twice the height of an object. In practice, formulas can be used in many ways both simple and sophisticated. While some of the more sophisticated usages may bring back unpleasant memories of high school math class, most are very easy to understand and create.

### Usage

- Embedding design relationships
- Relating a number of instances to a variable length
- Angular relationships

### Syntax

- Formulas are entered in the Formula column of the Family Properties dialog.
- Use normal mathematical syntax
  - Function (arguments)
  - Length = Height + Width + sqrt(Height*Width)
  - ArrayNum = Length/Spacing
- Parameter names are case sensitive
- Formulas can only drive numeric parameters

### Functions

- Addition— +
- Subtraction— -
- Multiplication—*
- Division—/
- Exponentiation—^: x^y, x raised to the power of y
- Logarithm—log
- Square root—sqrt: sqrt(16)
- Sine—sin
- Cosine—cos
- Tangent—tan
- Arcsine—asin
- Arccosine—acos
- Arctangent—atan
- e raised to an x power—exp
- Absolute Value—abs

### Methodology

- In the Formula column next to the appropriate parameter, type the formula for the parameter. Notice that the formula begins with an equal sign (=).
- Since units must be correct for the parameter type chosen, it is sometimes necessary to convert the units of parameters used in formulas. Most commonly lengths and numbers must be interchanged. To convert a length into a number, divide by 1 basic length unit, foot, meter, or mm. To go the other way, multiply by the same unit.
- To convert a continuously variable length to an integer length: Divide by 1 unit length to create a Number. Calculate a new value as a Number. Set an Integer = to the Number. Multiply the Integer by 1 unit length to get a Length
- To convert the length of an angled line to x and y values use sines and cosines: X= L*Cos(Angle), Y = L* Sin(Angle)

### Examples

- Open web joist that has an array of diagonals that are added or removed as the length changes
- Calculate area or volume of geometry
- Convert continuously variable values into integer values giving stepped increases to a dimension
- Add shelves as height of casework increases
- Create a clearance dimension parameter driven by element size.
- Truss Geometry

## Conditional Formulas

Conditionals allow you to define actions in a family that depend on the state of other parameters. They allow you to state IF something is true, THEN create a specified condition. While they are particularly useful in several circumstances, they do make families more complex and should not be used unless required. For most type parameters, they are superfluous since the type parameter itself is a form of IF statement: IF this is the type, THEN set this parameter to some value. Instance parameters are a more productive place for conditionals, particularly when they are used to set a parameter that does not vary continuously. Here, you do not know in advance what some driving value will be, and therefore may not be able to set some dependent value even though you know the rules by which it would be set. A formula alone is not sufficient since the driven parameter may have limited values it can take. Conditionals may prove useful for type parameters when the rules for setting a parameter's value are known, but the end user of the family is expected to make most types themselves and may not know those rules.

### Usage

- Conditional visibility
- Upper or lower size limits
- Limiting Array values
- A series of predefined sizes
- Exclusive options (If A then Not B)

### Functions

- Equal: =
- Less Than: <
- Greater Than: >
- IF
- NOT
- AND
- OR
- <= and >= are not implemented yet

"a <= b" can be written as NOT(a > b).

### Syntax

- Conditionals are based on evaluation of a statement for True/False
  - IF (<test condition>, <result-if-true>, <result-if-false>)
  - For simple inversion use: Parameter1 = NOT Parameter 2
- Test conditions can only include valid mathematical statements or YES/NO parameters.

- Formulas and conditionals can only drive numeric and YES/NO parameters

**Methodology**

- Use conditionals in the definition of the parameter whose value is conditional.
- In the Formula column next to the appropriate parameter, type the formula for the parameter. Notice that the formula begins with an equal sign (=).
- IF based on one test statement: =IF (Test Condition, Result if true, Result if false)
  IF ( Length < 35' , 2'-6", 3'-0" )
- IF based on more than one test: =IF (AND/OR (Test1, Test2) Result if true, Result if false)
  IF ( AND (x = 1 , y = 2), 8 , 3 )
  IF ( OR ( A = 1 , B = 3 ) , 8 , 3 )
- IF based on a negative test: =IF ( NOT (Test Condition), Result if true, Result if false)
  IF ( NOT ( A > B ) , 8 , 3 )
- Simple result if True, another IF if false: =IF (Test Condition, Result if true, IF (Test Condition, Result if true, Result if false ))
  IF ( Length < 35' , 2'  6" , IF ( Length < 45' , 3' , IF ( Length < 55' , 5' , 8' ) ) )
- Set one YES/NO parameter to the opposite of another: =NOT *Y/N Parameter*

**Examples**

- Preventing an array parameter taking a value less than 2: Array number = IF (Arrayparam < 2 , 2, Arrayparam)
- Turn off muntin visibility if number of window lights = 1: Muntins = IF (Lights < 2 , False, True)
- If length exceeds X increase depth by a formula: Actual Depth = IF (Length > X, Depth + 2, Depth)

## Ref Lines

Ref lines provide greater control over the location and orientation of portions of families. They create self contained reference systems that operate as part of the overall family. Each can define two work planes for element construction. Elements created in a work plane defined by a reference line will move and orient as the ref line itself is moved and rotated, without unexpected modification. Parametric manipulation of a single reference line can be used to control many objects. In addition, work planes defined by referenced lines can themselves host reference lines allowing the definition of very complex relationships.

**Usage**

- Rotating Parts
- Parts on parts
- Moving parts

**Methodology**

- Use the Reference line tool to draw a Ref line
- From the workplane dialog select the Ref line and TAB to get the desired plane
- Draw elements on that plane.

- Use Remap workplanes to attach already created elements to ref line based planes

**Examples**

- Adjustable swing doors
- Rotating equipment such as cranes, spot lights, etc.
- Moving whole assemblies

## Family Type Parameters

Family Type parameters allow you to create families with interchangeable parts. Using them, an instance of a nested family can be replaced parametrically by any type of any loaded family of the same category. While the detailed information about the nested family will not show in schedules, the Family and Type selected can be shown for either instances or types of the host family.

**Usage**

- Contolling interchangeable parts
- Scheduling which type is used

**Methodology**

Creating the parameter before placing an instance

- In a family, create a Family Type parameter.
- If you want to schedule this value, use a shared parameter
- Select the category of the families you want to change
- Set as either a Type or Instance parameter
- Load a family and place an instance.
- In the Family Properties dialog, select the family and type you want to use.

Creating the parameter after placing an instance

- Load a family and place an instance.
- Select the instance
- On the Options Bar, select the Label dropdown and select <add parameter>
- The new parameter dialog will open with the parameter type set to Family Type and the category set to that of the selected instance.

Scheduling

- To show the Family and Type selected for particular type or instance of the host family use a shared parameter.
- To schedule information about the nested family, it needs to be loaded into the project independently

**Examples**

- Doors with schedulable hardware sets
- Doors with variable Sidelights
- Counters with sinks
- Cabinets with a range of doors
- Muntins and grillage

# Revit Content Creation Standards

**Author: David Conant**



## Categories:

- All families generated using a generic template must be assigned to a category.
- Do not change the category of families without approval

## Family/type Naming

- Use title case for family and type names
- Do not repeat family name in type names
- Type names should mirror actual usage. Size should be indicated by use of specific dimensions in name rather than non-specific descriptions like “large”.
- Imperial units in names should appear as a’ – b c/d” x a’ – b c/d”. In most cases, sizes should be in inches i.e. aa” x bb”.
- Metric units in names should appear as ZZZZ x YYYYmm
- Nominal sizes should not use a units indicator in names i.e. 2 x 4 not 2”x 4” for dimension lumber

## Units

- Units neutral families should have at least one type for each unit system unless the family represents an item that is manufactured and sold only in one system.

## Parameter Names

- Parameter names should be as close as possible to natural English (minimize abbreviation/truncation)
- Standard parameter names should be used whenever possible.
- Do not change label names present in templates.
- Use title case for parameter names. Parameters are case sensitive.

## Parameter Usage

- Create parameters only when variation creates meaningfully differentiated types that represent real world possibilities.
- Parameter names reused to create equalities should be carefully checked for name coherence.

## Complexity

- Keep 3D content as simplified as possible. Avoid the temptation to build all detail in 3d. Most can be handled better by adding detail to the plan or elevation representations.

- Use scale dependent representation only where appearance change is meaningful at scales other than typical whole building scale and would be expressed on actual projects
- Do not depict any detail that users would not typically represent.
- Check different view scales to ensure that only the correct elements are visible.

## Element Visibility

- View specific representations must be drawn for any views in which an element is to be represented in a manner different than as a cut or projection of the 3d element.
- Check element visibility so that symbolic elements do not duplicate cuts or projections of 3d elements visible in the same view.
- Check all views to ensure that the symbol is displayed appropriately.
- Do not modify pen weight table in the family editor except to return it to standard.

## References

- All major edges, tops, bottoms, and meaningful centerlines should be references.

## Hosts

- Host objects must be large enough to accommodate all reasonable variations of the family.
- Floor hosted elements cannot be installed in a project until a user has placed a floor plate. Most floor-based families will perform better if built in a level based template. Use floor hosts only where the element makes a change in the floor, or where it is anticipated that the element will usually be installed on a sloping floor.

## Element Subcategories

- Assign elements to appropriate sub-categories as required to allow appropriate visibility control.
- Assign lines to the proper representation style (cut vs projection) of their subcategory by selecting from the properties dropdown list.
- Do Not Rename sub-categories

## Material Assignments

- Materials should only use textures present in the _Accurender or Revit libraries without modification unless a special material library is provided with families.
- Do not modify existing materials!
- Use materials from standard material table whenever possible.
- Maintain consistent material names between families unless independent materials are required.

## Predefined types

- All families should have at least one pre-defined type unless a type catalog is used.
- Where real world examples come in typical sizes, pre-defined types should be generated.
- Where there are to be more than 6 predefined types in a family, use a type catalog to organize the types.

## Model Elements:

- Lock all extrusion ends to any surface they must move with. Test by varying surface position or host dimensions.
- Dimension the depth of extrusion for all extrusions that must maintain a constant depth or where depth is to be parametrically controlled.
- All lines must be locked to any surface they must move with. Test by varying surface position.
- Sketch lines must be locked to any edges that they must move with.

## QA

## In family editor

- Test all hosted families to see what happens if the host's dimensions change both larger and smaller.
- Test all family parameters to ensure that the model behaves correctly as they are modified.
- Check all views to ensure that the symbol is displayed appropriately.

## In project

- Load family
- Instantiate all types
- Inspect in all views
- Dimension to all references
- Snap all references to walls
- Appearance in wireframe, hlr, shaded hlr, raytrace (when available)
- Modify host thickness for hosted families 25% - 400%
- Modify all family parameters from 0 to +400%
- Copy/Paste, Rotate, Mirror
- Ray trace