Autodesk University 2012

Autodesk® Revit® Families: Step-by-Step Advanced Concepts

Paul F. Aubin
Paul F. Aubin Consulting Services

AB2083-L The power and potential of the Autodesk Revit Family Editor is vast, and 90 minutes just doesn't do it justice. Dispensing with the basics, this hands-on lab jumps right into the deep end of the pool (if you need to brush up on the basics, video recordings of many Family Editor basics are posted on my website for all attendees). In this lab, we will explore advanced parameters and the use of formulas to drive the geometry. Whether you are new to the Family Editor or just want to use its more advanced features, this lab will give you the tools to begin making more advanced family content. We will explore formulas, family type parameters, materials, and even dabble in some trigonomic functions. So be sure to buckle your seat belt. You are in for an exciting ride!

Learning Objectives

At the end of this class, you will be able to:

- Create advanced family templates such as face-based
- Use formulas to drive parametric values
- Use mathematical relationships instead of geometry to control behavior
- Control nested families with family type parameters

About the Speaker

*Paul F. Aubin is the author of many CAD and BIM book titles including the widely acclaimed: The Aubin Academy Mastering Series: Revit Architecture, AutoCAD Architecture, AutoCAD MEP and Revit MEP titles. Paul has also authored several video training courses for lynda.com (**www.lynda.com/paulaubin**). Paul is an independent architectural consultant who travels internationally providing Revit® Architecture and AutoCAD® Architecture implementation, training, and support services. Paul's involvement in the architectural profession spans over 20 years, with experience that includes design, production, CAD management, mentoring, coaching and training. He is an active member of the Autodesk user community, and has been a top-rated speaker at Autodesk University (Autodesk's annual user convention) for many years. Paul has also received high ratings at the Revit Technology Conference (RTC) in both the US and Australia and he spoke at the inaugural Central States Revit Workshop this year. His diverse experience in architectural firms, as a CAD manager, and as an educator gives his writing and his classroom instruction a fresh and credible focus. Paul is an associate member of the American Institute of Architects. He lives in Chicago with his wife and three children.*

*Contact me directly from the contact form at my website: **www.paulaubin.com***

Introduction

One of the key criteria to being successful in Revit often involves having access to good content. Even with the best procedures, carefully crafted models and attention to detail, if you don't have a well-stocked repository of high quality Families from which to draw, you will find working in Revit frustrating. Revit ships with some items to get you started and there are myriad sites available on the Internet offering all manner of content. However, if you have been using Revit for even a little while, you have no doubt discovered that there is a wide disparity in quality between the various sources of content available. Furthermore, even with all the sources of content available, the chances that you will find all of the items you require readily available are highly unlikely. This reality is the most common reason why most folks begin learning how to create their own custom Family content.

Mastering the Family Editor takes time and dedication. The topic is varied and complex. This paper is not an introduction to the topic. Rather it is focused on a few key (and admittedly more advanced) topics. If you do not yet understand the basics of creating custom Family content, you are encouraged to explore the many resources available on the subject before talking the subjects covered in this paper.

I will assume that you are already comfortable with the following topics:

- Creating a new Family from a template file
- Laying down Reference Planes
- Creating basic dimensional constraints and parameters
- Building the standard geometric forms such as extrusions, blends and sweeps
- Adding Family Types and flexing your model
- Simple arithmetical formulas

This list is not comprehensive and some of the items mentioned may be reiterated in the topics that follow, but if none of the items on that list made you say to yourself "hold on, what is that?" then you are probably in the right place.

> **Note:** Last year at AU, I taught a lab titled: "Autodesk® Revit® Families: A Step-by-Step Introduction". You can find this course at AU Online here: **http://au.autodesk.com/?nd=class&session_id=9060**. I also have the handouts, datasets and video recordings of this session posted on my website at: **http://paulaubin.com/au/**. You are highly encouraged to check out these resources BEFORE attending this lab if at all possible. Particularly if you do not have very much prior Family Editor experience. The lessons in this lab will build on the knowledge from last year's session and even use some of the same datasets.

Family Categories

When you create a new Family, the first thing you must decide is what Family Template to use. There are two basic properties that the template file will impart to your new Family: its Category

and its hosting behavior. There are many other less obvious settings and behaviors that you also inherit from the template. For this lab, we will only consider a few of the available categories such as Furniture System, Furniture, Specialty Equipment and Generic Model. Table 1 summarizes these categories and the behaviors and settings that they inherit from the Family Template. Casework (a similar and related category) is also displayed on the table.

Table 1

Imperial	Host						Massing Environment			Special		Default Settings					
	Non-Hosted (Free Standing)	Face	Wall	Floor	Ceiling	Roof	Free Standing	Adaptive	Pattern Based	Line Based	Dedicated to Particular Host	Cuttable	Work Plane-Based	Always Vertical	Cut with Voids When Loaded	Shared
Furniture System.rft	•													•		
Furniture.rft	•													•		
Specialty Equipment.rft	•													•		
Specialty Equipment wall based.rft			•													
Generic Model.rft	•											•		•		
Generic Model Adaptive.rft	•							•				•				•
Generic Model ceiling based.rft					•							•				
Generic Model face based.rft		•										•				
Generic Model floor based.rft				•								•				
Generic Model line based.rft										•		•				
Generic Model Pattern Based.rft									•			•				•
Generic Model roof based.rft						•						•				
Generic Model wall based.rft			•									•				
Casework.rft	•											•				
Casework wall based.rft			•									•				

When you are planning your Family, think carefully about the settings that the template file imparts. For example, it can is common for family authors to use the Generic Model template and then simply change the category later. There is nothing inherently wrong with this approach as long as the category is established early in the process. If you change the category too late, it can force you to rework parts of your family. For example, any subcategories that you add in object styles and the “cuttable” behavior of your family are determined by its category. If you change the category, it will reset all of your subcategories and object style settings.

The hosting behavior cannot be changed later. You have basically three options here. You can choose the exact host you need such as “Wall-Based” or “Ceiling-Based.” You can choose a freestanding template (one that does not require a host) or you can use one of the Face-Based or Work Plane Based options. A common practice in multi-discipline firms is to use Face or Work Plane Based in lieu of Wall or Ceiling based. This keeps the family more flexible and makes it less likely to become invalid should its host go missing in an update to a linked model.

Another important consideration in choosing your category is how you want the item you are building to appear in schedules. Since Revit does not always have the category you need, or the item you are creating might fit loosely into more than one category, it can be tempting to simply leave it as Generic Model. However, you cannot create a Generic Model Schedule. This category does not appear in the list when choosing your schedule’s category (see Figure 1).

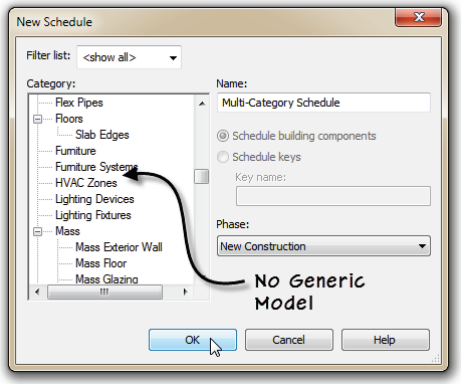


Figure 1—To include Generic Models on a Schedule, you must do a Multi-Category Schedule

So while starting with Generic Model can be convenient, the fact that it cannot be directly scheduled means you should consider this carefully before leaving the category set to Generic Model. It should be noted that we can do a Multi-Category schedule that would include the elements set to Generic Model. However, it would also include all other categories as well. So unless you can find a reliable way to filter such a schedule, it may not prove a very workable solution. The main point is this:

Careful planning is the most important step in Family content creation.

We will look at technique you can use to filter your generic models on a schedule in an exercise below.

With the above issues in mind, most of the content we will work with in this lab uses the categories listed in Table 1.

Family Editor 500

Prerequisites and Setup

If you were going into the shop to build some cabinetry, you'd want to make sure you had all the tools you needed and that the shop was in order before you started. Let's do the same in Revit.

All steps and screen shots here are Revit Architecture 2013, but most should work in other flavors or versions.

The Steps outlined here are meant to supplement the live presentation given in the lab. Steps have been kept brief and much of the explanations accompanying the steps in the live lab have been kept brief.

1. If Revit Architecture is not already running, launch it now.
2. From the Application Menu (big "R"), or on the Recent Files screen, choose **Open > Project**.
3. Browse to the folder containing this lab's dataset files (I will have this posted up on my screen) and open the file named: ***500 Sandbox_A.rvt***.
4. Minimize the active view.

The project file will remain open as we work through the next several lessons. As you build Family content, you will want to test if frequently. Having a project file open in the background is an excellent way to do this. I like to minimize it so that it stays out of my way till I need it. There is nothing special about a "sandbox" except what you put in it. You can open your standard office template, add a few Walls and other items and save it as a sandbox. *It is that simple*.

And now, on with the good stuff! Let's start building some Family content!

Scheduling Family Data

In the previous topic, we discussed the importance of your chosen category as it relates to your schedules. The first part of this paper will discuss how to make your custom Families report useful data to your schedules. In order for a schedule to report a piece of information, there has to be a parameter to store that information. In addition, the parameter has to be either a built-in parameter (a so-called "**system**" parameter) or if it is a custom user-defined parameter, it has to be configured as a "**shared**" parameter.

In addition, you have to consider how the Family's geometry is structured. If you build all of the family geometry in a single RFA file, it is easier to ensure that the data you want to report to your schedules and tags will be available. If you use nested families, they must be configured as "**shared**" families in order for them to report data through their hosts and to the project's schedules.

Getting the picture? "Shared" is the magic word to making families talk to schedules.

Let's start with a look at parameters. There are four kinds:

System Parameters

A System Parameter is the "easiest" of the bunch. As its name implies, it is "built-into" the system. There is no further steps necessary on our part. Some system parameters belong to a single category, some apply to multiple categories. A system parameter is available to all projects, all families (in the category or categories applicable) all schedules and all tags. If you can get away with only system parameters in your content, your task is truly simpler. But alas, it is rarely the case that you can get away with only system parameters.

Project Parameters

A Project Parameter is a custom parameter created in the project environment. It applies to one or more categories that you designate. Since it applied at the category and project level, it applies to *all* elements of that category (or categories) within the project. It is *not* necessary to add it individually to each family. A project parameter can appear in schedules, but cannot appear in tags. Project parameters must be added separately to each project. If you wish you can add them to your project template file so that new projects begin with whatever custom project parameters you require.

If your requirement is only for schedules and not tags, and if you do not need to pre-assign data in the family files, then a project parameter can be a good way to go.

Family Parameters

A Family Parameter is defined in a family file and is only part of that family. A family parameter can drive geometry and behaviors in the family but cannot be scheduled or tagged in a project. Family parameters are most often used to assist in building the family geometry, apply materials and build in other family specific behaviors.

Shared Parameters

Shared parameter is a fancy word for a parameter that you want to "share" among one or more projects, families and or schedules and/or tags. Shared parameters can be defined as both project and family parameters giving them the same benefits and features of each of those parameters respectively. However by being configured as a shared parameter we also gain the ability to schedule and tag the parameter and to use it in more than one project and/or family.

If you have any suspicion that the parameter you are creating will want to appear in a schedule or tag, you should be proactive and set it up as a shared parameter.

Family Editor 501

Creating a Project Parameter to filter Generic Models

Let's do a simple example of a Project Parameter. Project parameters are added at the project level and apply to all families belonging to the category or categories to which the project

parameter is configured to apply. This means you do not have to open each family and add the parameter separately. Above we discussed the Generic Model category and its inability to be scheduled independently. However, you may have situations where you need to use the Generic Model category regardless. This exercise will give you a quick and effective (if a little contrived) way to ensure your schedule includes only Generic Models.

1. You should already have the ***500 Sandbox_A.rvt*** file open, if not please open it now.
2. Place at least one or more instances of both the Family *501 Box_A:My Box* and *501 Box_B:My Other Box* anywhere in the building.
3. Open the Schedule called *Generic Model Schedule*.

Study the Category column and notice that this schedule contains items from several categories. If you Edit the Schedule and look at the fields, you will see that it reads “Multiple Categories” on the fields available from drop-down list. If you want to create your own version of this schedule, simply leave <Multi-Category> selected in the “New Schedule” dialog, then choose your fields. If you scroll through the schedule, you should see your boxes among the other items on the schedule. We want to limit the schedule to *just* the items.

4. On the Manage tab, click the Project Parameters button.
5. Click the Add button.
6. In the “Parameter Properties” dialog, make sure the “Project parameter” is selected at the top. In the Name field, type: **Schedule Filter**, and change the “Type of Parameter” to **Yes/No**.

We can assign the parameter as either Type or Instance just like you can in a Family.

7. Choose the **Type** radio button. On the right, beneath Categories, check the Generic Models button and then click OK (see Figure 2).

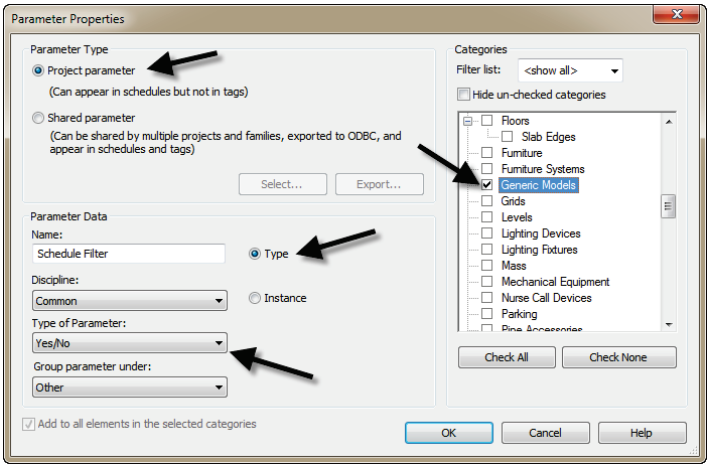


Figure 2—Configure a Yes/No Project Parameter

This parameter is now available to all Generic Models in this current project. We can now add this a field to our schedule and use it to filter the schedule contents.

8. Select the first box, edit its Type properties and make sure that the Schedule Filter property is checked. Repeat for the other box. (Since it is a Type parameter, you only need to do one of each type).
9. Open the schedule and on the Properties palette, click the Fields button. Add the new Schedule Filter field.
10. Click the Filter tab, and add the Schedule Filter property, set to “equals” and “yes.” Click OK to see the results.

Creating Shared Parameters

Shared Parameters must be configured using the Shared Parameters command and common Shared Parameter file. You can find the Shared Parameters command on the Manage tab of the ribbon (see Figure 3).

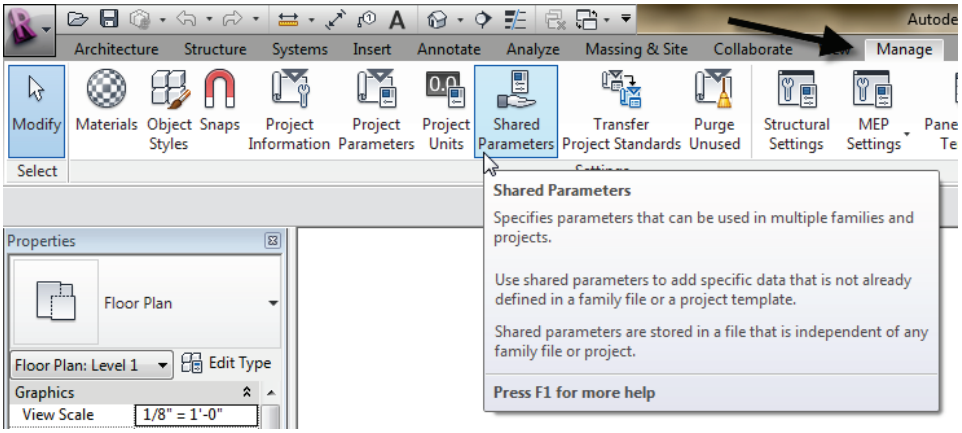


Figure 3—Access the Shared Parameters command on the Manage tab of the ribbon

There are two basic procedures to follow. You can create a new Shared Parameter file or you can browse to and open and existing one.

Important: In most cases, you will want to have only ONE shared parameter file for the ENTIRE office. Do not make multiple shared parameter files.

The shared parameters file is simply a “master list” of available parameters. It is *not* the actual parameters. Rather it is used to define the parameters in project and family files. Once a parameter is defined, it operates independently of the shared parameter file. However, that initial creation from a common shared file *is* critical! If you create multiple shared parameter files, you can end up with inconsistencies and redundant parameters. So remember, regardless of how many offices, team members, projects or actual parameters you need, *create and maintain only ONE Shared Parameters file for the entire organization*.

Family Editor 502

Process to Create or Load a Shared Parameters File

Before you can begin using Shared Parameters, you need to be sure that you have a Shared Parameter file. If your firm does not already have one, you can create one. Otherwise, simply point your copy of Revit to the office standard Shared Parameter file that is typically on your firm's network server. Either way, the steps are similar:

1. If Revit is not already running, launch it now.
2. On the Manage tab, on the Settings panel, click the ***Shared Parameters*** button.

If you need to **create** a new Shared Parameters file, perform these steps:

a. In the "Edit Shared Parameters" dialog, click the Create button.

b. In the "Create Shared Parameter File" dialog, browse to the desired location where you wish to store the file. This is typically on a network server so that all members of the firm can access it in the same location.

c. Give the file a name and then click the Save button (see Figure 4).

If you **already have** a Shared Parameter file, you can load it. Check with your CAD or BIM Manager to see where the file is stored and what it is called.

a. In the "Edit Shared Parameters" dialog, click the Browse button.

b. In the "Browse for Shared Parameter File" dialog, browse to the location of the office standard Shared Parameter file (for the lab it is called: ***OfficeStandardParameters.txt***), select it and then click Open (see Figure 4).

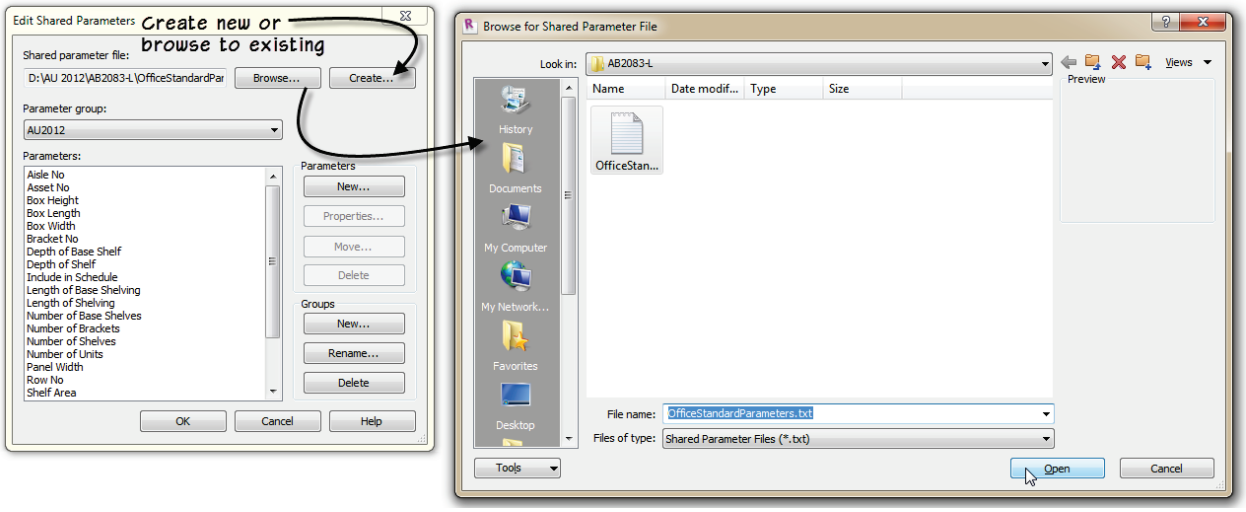


Figure 4—Create or Browse to a Shared Parameter File

***Family Editor 503

Manage Shared Parameter Groups

Shared Parameter files are organized into groups. Groups can be named anything you like and are used simply to organize the file. If you have a file with lots of parameters, you will want to consider your group and parameter naming carefully. If you have just created the file from scratch, you will need to add at least one group before you can add any parameters. If you browsed to an existing file, then it will already have at least one group, but you can certainly edit or create new groups. For the lab, we will use the provided file. But back in the office, you can follow these procedures:

1. In the “edit Shared Parameter” dialog, beneath Groups, click the New button.
2. Give the Group a name and click OK (see Figure 5).

You can also Rename or Delete existing groups, but at least one group is required.

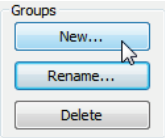


Figure 5—Create or edit Parameter Groups

Once you have a file and at least one group, you are ready to add parameters to the file. You can edit the file anytime and add more parameters, so initially, it is a good idea to just create those parameters you need for the project or family at hand.

To create a new parameter:

3. In the “edit Shared Parameter” dialog, beneath Parameters, click the New button.
4. Input a name for the new parameter and choose the Type of Parameter. Click OK to complete the parameter (see Figure 6).

If you choose a different Discipline, you will get a completely different list of parameter types. So be sure to choose the Discipline first.

Notice that you are only asked to set up name, discipline and parameter type. Any other settings like type vs. instance and group parameter under get established later directly in the family or project.

5. Create as many parameters as you require and then click OK to dismiss the “Edit Shared Parameters” dialog.

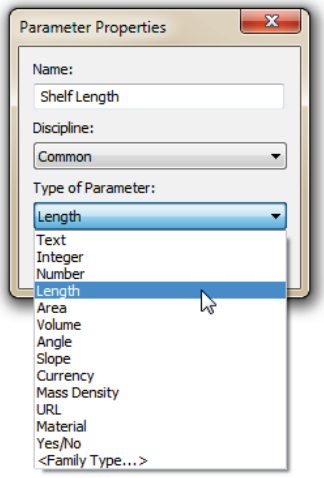


Figure 6—Create a new parameter

The shared parameter is really just the rules to create the parameter. Each shared parameter is completely unique. In the shared parameter file is a unique identifier that Revit maintains for us. This is very important to ensure the proper functioning of the shared parameter and is what makes it "sharable." Think of the Shared Parameter as the recipe for your favorite meal. It establishes the ingredients and steps required to get a tasty dish. But it is not the meal itself. But each time you prepare the recipe, you get similar results. In similar fashion, in each project or family where you use the same shared parameter, you will get the same results in your content, schedules and tags.

Family Editor 504

Adding a Shared Parameter to a Project or Family

Creating the shared parameter file and its groups and parameters is a necessary first step. But just like a recipe by itself does not make a very satisfying meal, until you build some content and use the shared parameters within them, you do not see the full benefit. Let's look at a simple example. Suppose we are working on a retail showroom project. We want to track the shelving units in one or more projects using the client's tracking system. Let's say that the client assigns all shelving items a unique unit number. We can create a custom text field for this as a shared parameter and then use that parameter to create a custom tag and schedule that reports the value of the unit number assigned to each shelving unit item in the projects we do for this client.

The process to create a parameter from the Shared Parameter definition (in the Shared Parameter file) is similar whether you are adding it to a tag, a project or a family.

Tags

Let's start with tags, which of all the items listed, are perhaps the most restrictive. If you want your tag to report a custom parameter, you set it up as a shared parameter. Start by creating the new tag family from the correct template.

1. Continue in the ***500 Sandbox_A.rvt*** file. From the Application menu, choose **New > Annotation Symbol**.
2. From the list of templates, choose *Genetic Tag.rft* and then click Open.
3. On the Create tab of the ribbon, on the Properties panel, click the Family Category and Parameters button.
4. Choose the appropriate category for your tag. In this example, we'll choose Furniture System and then click OK (see Figure 7).

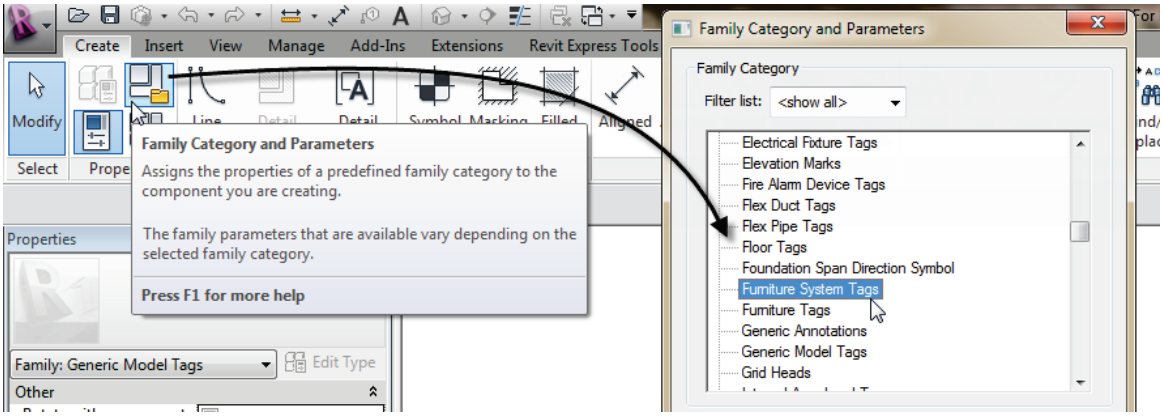


Figure 7—Change the Category of the Tag

This first step is important or you will not get the correct list of System Parameters when adding a label to your tag. If you already know the size of your tag graphics, you can draw them first. But if you are unsure, you can add the Label(s) first to give you a sense of scale and then build the graphics around it.

5. Add a Label at the center of the Reference Planes.

In the dialog that appears, the list on the left is labeled "Category Parameters." This is the list of System parameters available to this category. This is why it is important to set your category first.

6. In the "Edit Label" dialog, at the bottom left, click the Add Parameter icon.
7. In the "Parameter Properties" dialog, click the Select button.
8. In the "Shared Parameters" dialog, choose the Unit No parameter and then click OK twice (see Figure 8).

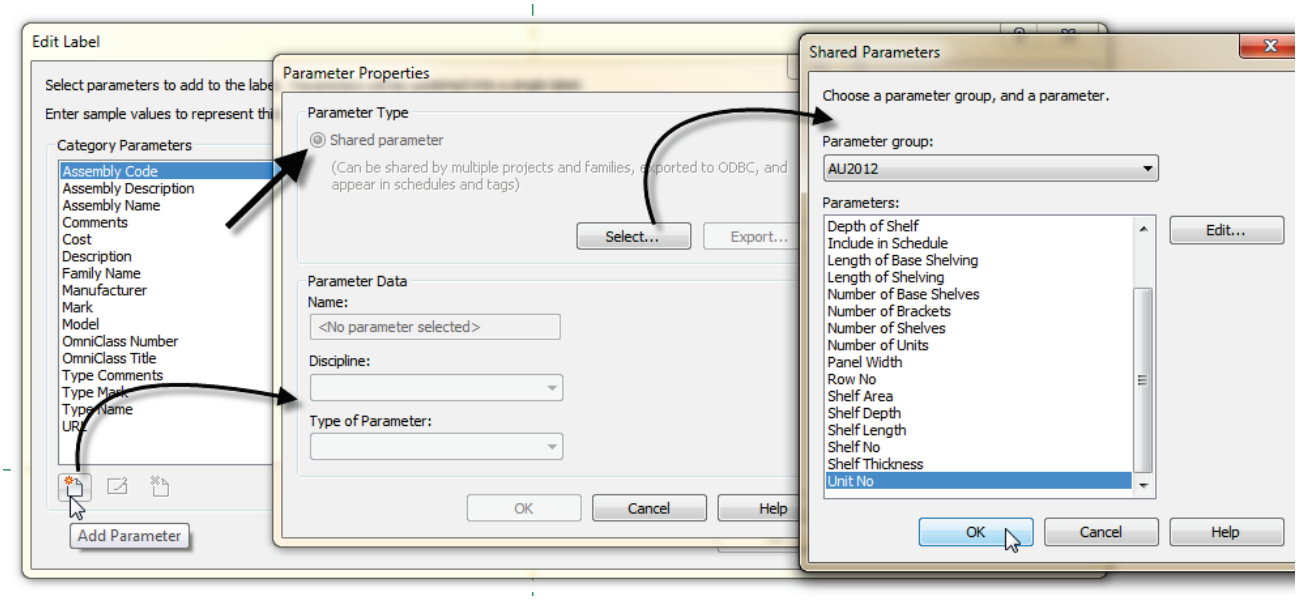


Figure 8—Add a new parameter from the Shared Parameter file

The new parameter will be added to the list on the left. You can now add it to the label on the right.

9. With Unit No still selected, click the icon in the middle to add it to the label. You can add other parameters to the same label if you wish. Click OK when you are finished.

If you wish, you can input a prefix, a suffix or a sample value. The prefix and suffix will appear before or after the value input for the label in all tags. For example, if the client always wanted to include the abbreviation “UN” in front of all unit numbers, we could add UN to the Prefix field. You can also change the Sample Value to anything you like. It is a good idea to use a common generic value here that won’t be mistaken for a “real” value (see Figure 9).

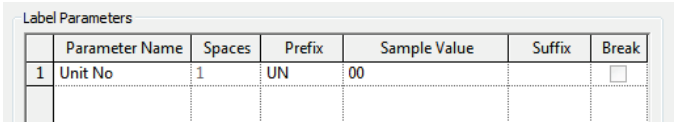


Figure 9—Optionally edit the prefix, suffix and sample value

You can add additional parameters to this label, or you can click OK and create additional labels if required. Add the graphics to your tag and you are ready to test it out.

10. Using the Line tool, draw the graphics for your tag. Delete the red text note, save the file and then click the Load into Project button to load the tag into our sandbox and test it out (see Figure 10).

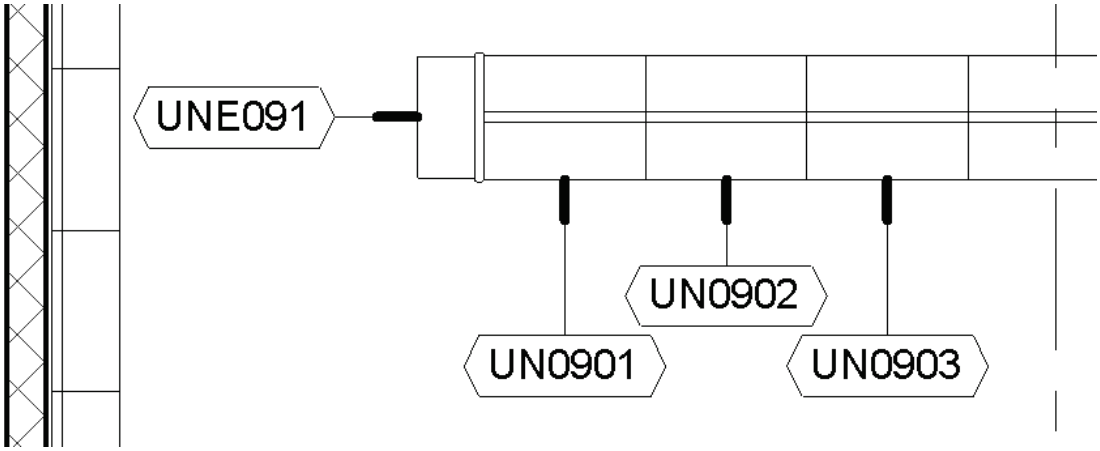


Figure 10—Tags added to the project and reporting the values of shared parameters

***Family Editor 505

Shared Project Parameters

Next we can look at adding Project Parameters using Shared Parameters. The process is similar to what we did to add a shared parameter to a tag and the project parameter we added above. Any project parameter (shared or not) can appear in schedules. So you may be wondering why bother make it a shared parameter? Why not just do it like we did in exercise 501 above? Well, keep in mind that two parameters that have the same name are *not* the same! If you want a consistent definition of a parameter from one project to the next, it needs to be a shared parameter. But more importantly, if you want the parameter to report to both schedules and tags, you *must* set it up as a shared parameter.

When it comes to parameters that are largely used to track data, it is usually safer to make them shared parameters. Non-shared family or project parameters are best used for parameters that perform a unique function in a *particular* family or project and do not need to be scheduled or tagged. The project parameter created above was simply to filter the schedule, but the parameter itself was not important to the schedule content.

Add a Shared Project Parameter

Autodesk Seek includes a huge selection of downloadable content including both generic items and items supplied by product manufacturers. Many Revit families are included among these items. In this example, let's assume that you have several items of the same category like several casework items. Each was acquired from a different source. In other words, some of them are out-of-the-box families provided with Revit, some are from a manufacturer and downloaded from Seek, others might be custom built. While it is possible to open each family and add the custom parameters required to tie into your client's tracking system, this is not very practical. In this case, a project parameter can be a better choice if you don't need to tag it.

Some of the content used in this topic was downloaded from Seek:
http://seek.autodesk.com/product/latest/agg/mcgrawhill/Hamilton-Sorter/HamiltonSorterMCBK362429

1. On the Manage tab, on the Settings panel, click the Project Parameters button.
2. In the “Project Parameters” dialog, click the Add button.
3. In the “Parameter Properties” dialog, choose the **Shared Parameter** radio button at the top. Click the Select button and choose the desired parameter (Asset No in this example).
4. Choose either a Type or Instance parameter and optionally change the Group parameter under setting. Finally, on the right from the category list, select one or more categories (such as Casework) for this parameter (see Figure 11).
5. Save and Close all files.

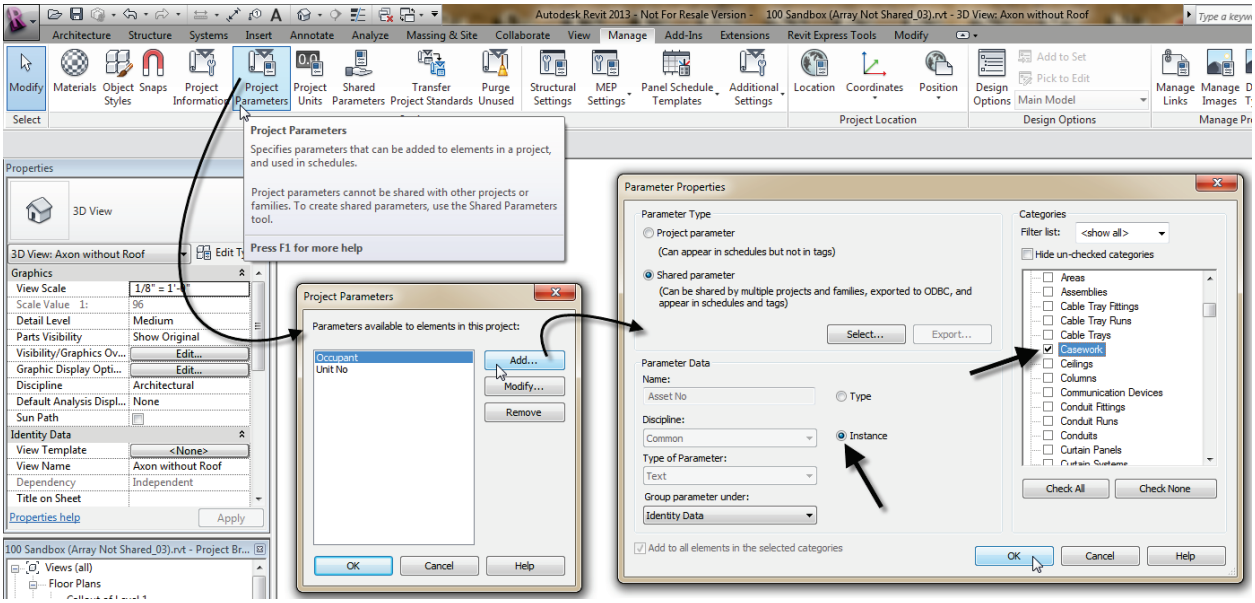


Figure 11—Add a new Project Parameter

When you are finished, you can select any Casework item in the model and on the Properties palette, it will have the Asset No parameter (see Figure 12).

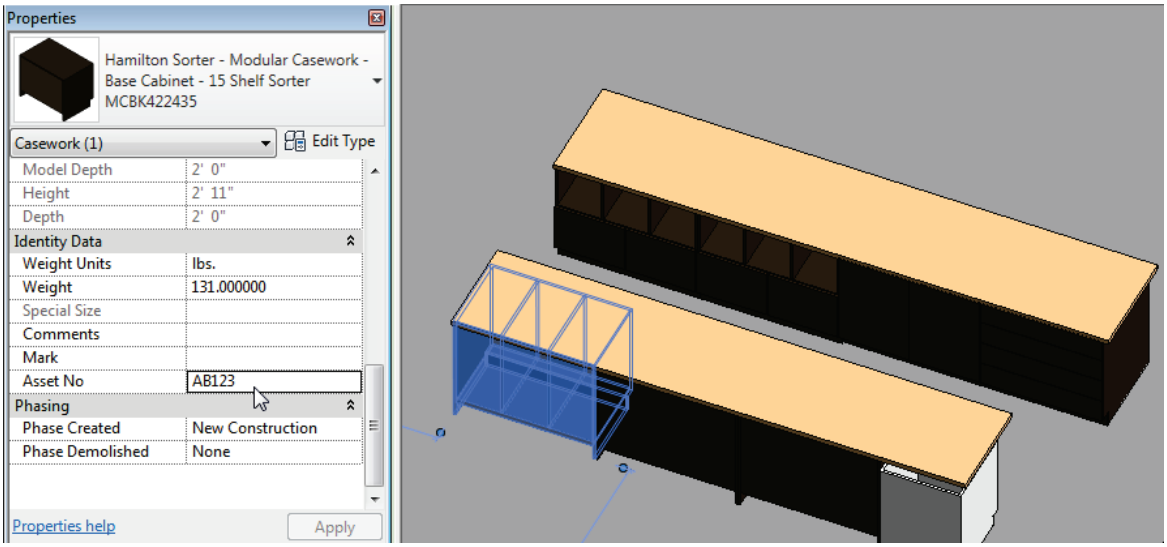


Figure 12—Shared Parameter applied to all elements in the project as a project parameter

Shared Family Parameters

The process to add a shared Family parameter is very similar to adding the previous two examples. The only real difference is where you start the process. In a project file, you will be making a shared project parameter. In the Family Editor, you will be making a shared family parameter. In all other ways, the process is nearly identical.

***Family Editor 506

Add a Shared Family Parameter

To add a Shared Parameter in a Family file, open the Family in the Family Editor. Use the Family Types button on the Properties panel of the ribbon. There beneath the Parameters heading on the right, you can click the Add button. The “Parameter Properties” dialog will appear nearly the same as the examples above, minus the category list on the right. This is because the category of the family file is already established and as noted above, this parameter you are adding will exist *only* in this family file.

1. Open the ***500 Sandbox_B.rvt*** project file.
2. Open the Furniture Systems Schedule and edit the fields.

Notice that there are now parameters for length or width of the shelving.

3. Select one of the shelving units onscreen and on the ribbon, click the Edit Family button.
4. Open the *Ref. Level* floor plan view.
5. Select both Depth labeled dimensions and then on the Options Bar, click the Label drop-down and choose **Add Parameter**.

6. In the “Parameter Properties” dialog, choose Shared Parameter at the top and then click the Select button. Choose the Shelf Depth parameter and then click OK (see Figure 13).

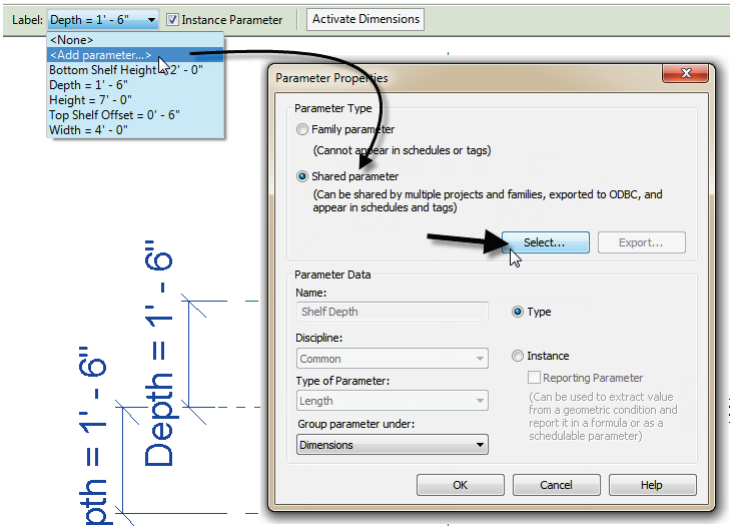


Figure 13—Shared Parameter added in a Family file

7. Repeat by replacing the Width labeled dimension with the Shelf Length shared parameter.
8. Save the file, reload it into the *500 Sandbox_B.rvt* project file and overwrite the existing.
9. Return to the schedule, edit the fields and notice that you can now add Shelf Width and Shelf Length. Feel free to do so and study the results.
10. Save and Close all files when finished.

Other Shared Parameter Considerations

The previous topics give an overview of the shared parameter tools and procedures. In this topic, we will touch on a few additional bits of shared parameter information you might want to keep handy.

- **Deleting Shared Parameters:**

  It is possible to delete a Shared Parameter from the Shared Parameter file. This *does not* affect any existing parameters created from the parameter, but does make it *impossible* to create a new parameter from this shared parameter definition. Think of it this way, if you bake some apple pies from your favorite recipe, and then throw away the recipe while cleaning up, you still have the pies (at least until the kids get home from school…) you will just have a hard time baking another pie the next time.

  If you delete a Shared Parameter, Revit will warn you:

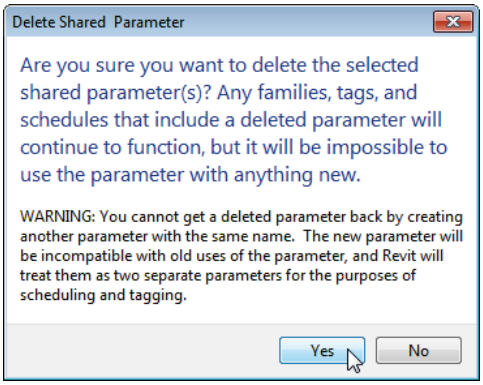


Figure 14—Be careful when deleting a Shared Parameter – there is no undo!

The most important part about the warning is the bottom part telling you that recreating the parameter does not work. Revit will treat the new parameter as new even if it has the same name!

- **Exporting a Shared Parameter**

If you do delete a Shared Parameter, or if you receive a file from someone and do not have access to their Shared Parameter file, you can export the "missing" parameters to your shared parameter file. To do this, edit the parameter in the family or project. In the "Parameter Properties" dialog, the Export button will be lit up indicating that you can export this parameter. (If the parameter is already in your Shared Parameter file, you cannot export it, this button will be disabled). When you click Export, Revit will explain where the parameter will be saved. You can always move it to a different group later (see Figure 15).

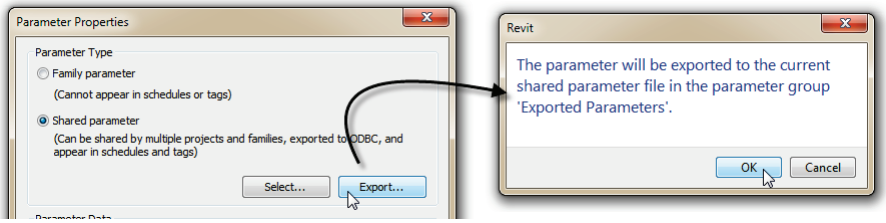


Figure 15—Export missing parameters to your Shared Parameter file

- **Merging Shared Parameters from multiple files**

Despite our best efforts to the contrary we all know that sometimes we end up with variances to the published office standards and procedures. As noted above, it is highly recommended that you establish and maintain a *single* Shared Parameter file for your entire organization. Not one per project, not one per office, but ONE for the entire organization; all offices; all projects. However, this is often tough to accomplish. In fact, you are currently using the Shared Parameter file provided with this lab's dataset, so what do you do if you already had one? **Revit Family Tools** to the rescue. This free tool distributed by the CAD Technology Center has a few very useful features and among them is a tool to merge the

parameters from two Shared Parameter files into one file. It is highly recommended that you download this tool:

http://www.cadtechnologycenter.com/ctc-products.html

- **Shaded Parameter information online**

There are far too many Revit resources online to list in any kind of comprehensive way. Google is your friend! However, any serious Revit user should have Steve Stafford's blog on their reading list. Steve recently posted a summary of all the shared parameter posts he has created over the years. Yikes, I had no idea there were so many! Check it out:

http://revitoped.blogspot.com/2012/09/parameter-related-post-summary.html

Nested Families

If you have been building family content for a while you have likely explored nested families. A nested family is simply an instance of a family inserted as a component in another family. There are obvious places where it makes sense to use this capability. If you have a complex form to model, nesting can be an effective way to break up the object into smaller easier to manage and model chunks. Not everyone agrees on the best times to use nesting. Some family authors rely on it heavily, others avoid it altogether. My aim here is not settle this debate. Rather I would simply like to share some thoughts and perhaps propose a few guidelines when nesting.

- **Have a Plan**—The most important aspect to successful use of nesting is to plan out your family. Do some sketches and have a clear idea of why you are nesting a component. If a clear benefit does not come to light in this planning process, then perhaps you can skip the nesting and build the geometry directly in the host.
- **Nesting helps with rotation, mirroring and arrays**—If you need to rotate, move or mirror an element parametrically or wish to make a parametric array in your family, nesting is almost a must. It is not impossible to achieve these behaviors without nesting, but it can be much more challenging (see Figure 16).

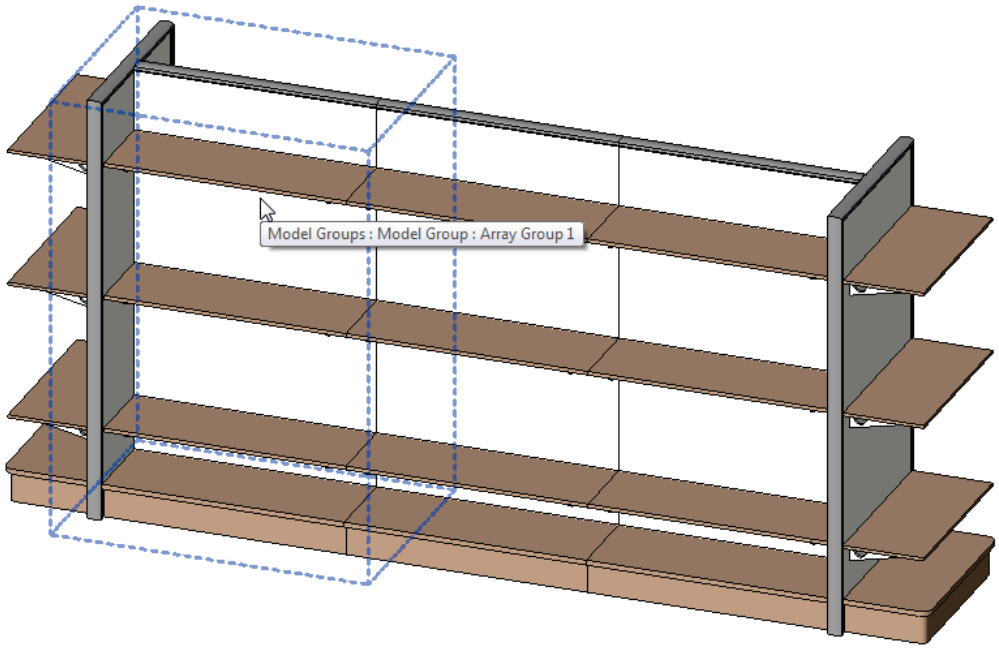


Figure 16—Nested Families simplify the creation of parametric arrays

- **Give careful consideration to how many levels of nesting you introduce**—It is possible to nest multiple layers deep. In other words, you can nest family A into family B and then in turn nest family B into family C and so on. There can be benefits to doing this, but each level of nesting you introduce will increase the complexity of the final family and make it more challenging to track down problems when it does not flex properly. Multi-level families have been created quite successfully, but careful planning and documentation is a must. If you are not sure how deep to nest, start with a manageable rule-of-thumb of say three levels and work from there. If you are building a family that seems to need more than three levels, don't immediately rule it out, just let it be a small red flag that tells you to look for other ways to solve the problem. If no viable alternative can be determined, then "break the rule" for that family (see Figure 17).

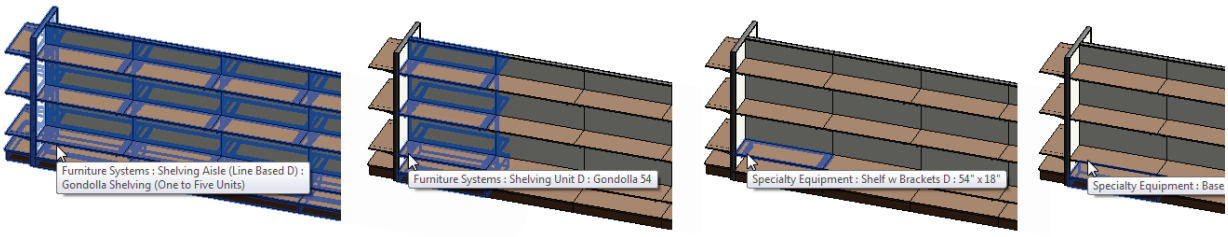


Figure 17—Nesting multiple levels

- **Linking up nested parameters**—If you have a nested Family, you can drive the parameter values in that family from the host family. To do this, look for the small button in the far right column of the Properties palette or "Type Properties" dialog (see the right side of Figure 19 below). You will need to create a parameter in the host family that will drive the value of the nested family parameter. This parameter can have the same name or can be different.

Family Editor 601

Add a Shared Family Parameter

Let's tie the previous two topics together. If you have created Shared Parameters to report in your schedules and tags and you are using nested families, then it is very important that you also consider using "Shared families." A Shared Family enables the nested family to appear as a separate selectable element in the host family or project. If you have parameters in the nested family that you wish to schedule or tag, you *must* use a shared family. You can do this with the "Family Category and Parameters" dialog or directly on the Properties palette (see Figure 18).

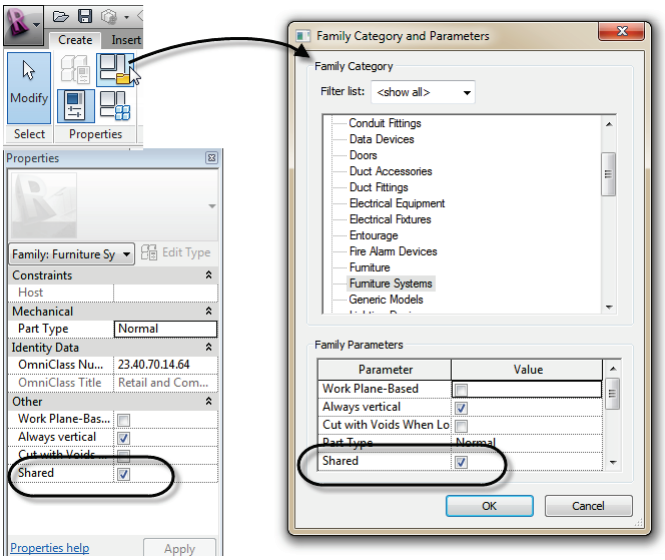


Figure 18—Make a Family Shared on the Properties palette

There is a very important consideration when using shared families. If you have nested families that rely on linked type parameters (see the "Linking up nested parameters" bullet point above), you cannot make the family shared. Shared families cannot have their type parameters driven by a host family. So you either have to avoid nesting, use instance parameters or not use shared families (see Figure 19).

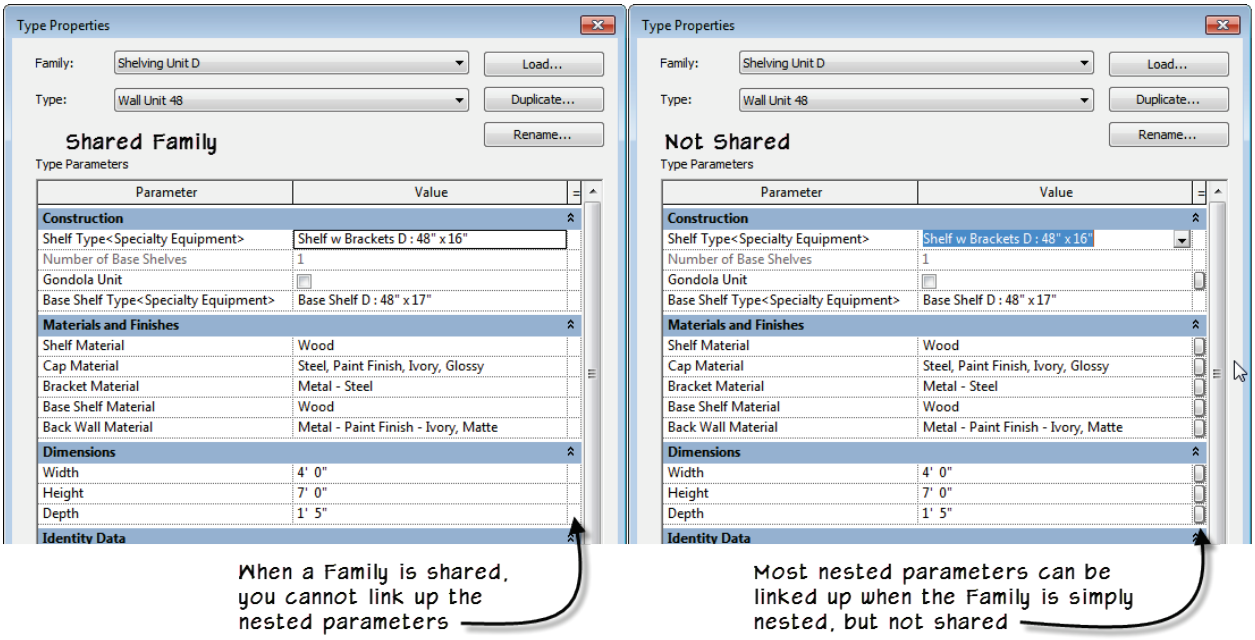


Figure 19—Nested Shared Families cannot have their type parameter values driven by the host Family

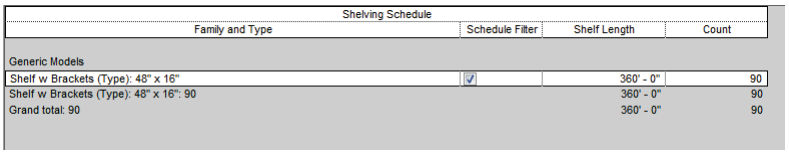
If you don't use shared families, you cannot schedule the nested components directly. However, it is possible to get around this by careful planning and the use of formulas. For example, the driving parameter can be scheduled at the top level host family. This top level family can in turn drive the parameters of the non-shared nested families within it. You can also use instance parameters with shared families. There are some issues in each situation. Let's look at a few of these in an exercise.

1. Open the ***600 Sandbox_A.rvt*** project file.
2. Take note of the shelving units in the file. Use the TAB key to try to select the nested elements. Open each of the schedules.

Notice that you cannot select the nested shelf components and that the overall shelving units do appear on the Furniture Systems schedule, but none of the individual shelves appear on the shelving schedules. In this sequence, we will use shared families to help us begin counting the nested shelves in each unit.

3. Open the family ***601 Shelving Unit_A.rfa***.
4. Open the family ***Shelf w Brackets (Type).rfa***.
5. On the Properties palette, check the Shared checkbox. Save the file and then click the Load into Project button.
6. In the "Load into Projects" dialog, check ONLY the *601 Shelving Unit_A.rfa* checkbox and then click OK. Press ESC before placing an instance.

7. Select one of the shelves (it is a Group). Click the Edit Group button and then select the Shelf again.
8. From the Type Selector, choose *Shelf w Brackets (Type):48" x 16"* and then click the Finish button on the Group edit toolbar.
9. Select an instance of the Group on the other side and repeat. Save the file and then load it into the sandbox file. When prompted, overwrite the original.
10. Open the *Shelving Schedule (No Filter)* schedule. Notice the gray checkbox in the Schedule Filter column. Click this to check it.
11. Open the *Shelving Schedule* (see Figure 20).



Shelving Schedule

Family and Type	Schedule Filter	Shelf Length	Count
Generic Models			
Shelf w Brackets (Type): 48" x 16"	☑	360' - 0"	90
Shelf w Brackets (Type): 48" x 16": 90		360' - 0"	90
Grand total: 90		360' - 0"	90

Figure 20—After loading the shared nested family, the shelves appear on the schedule

If you study the itemized schedule and the 3D view, you will see an outstanding issue. The Shelving Unit family includes two overall sizes: 48" and 54". However, while the shelving unit flexes, the nested shelves currently do not. As we discussed above, if you edit the Type Properties of the nested family (*Shelf w Brackets (Type)*), you cannot link up the nested parameters to make them flex. We have a few alternatives to explore. We can use instance parameters in the nested component or we can explore <Family Types> parameters. Let's take a look at each. To save a few steps, the instance-based version is already saved in the shelving unit family. We only need to swap it into the groups.

1. Return to the family *601 Shelving Unit_A.rfa* family.
2. Repeat the Edit Group procedure above, but this time swap in the *Shelf w Brackets (Instance):48" x 16"* type instead.
3. Before finishing the group, select the shelf and on the Properties palette, link up the Shelf Length parameter with the Shelf Length in the host family (see Figure 21).
4. Finish the group and repeat on the other side. Save the family and reload into the sandbox and update.

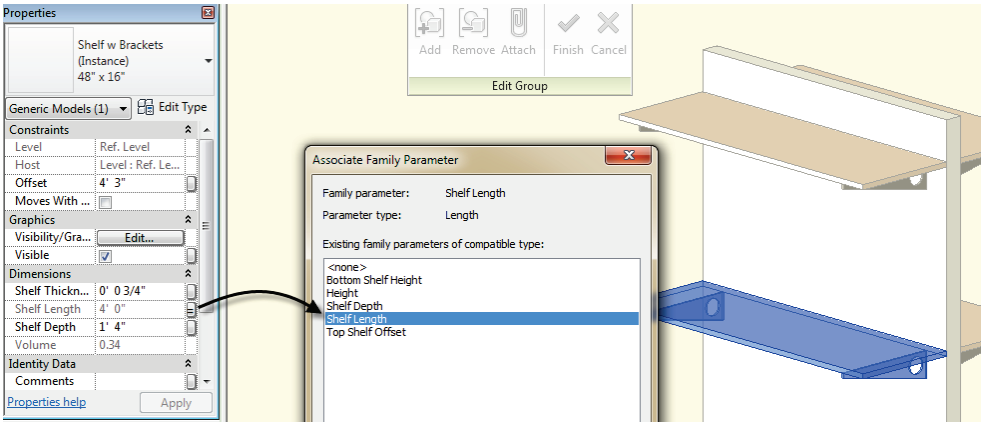


Figure 21—Swap in the instance based family and link up the parameters

The *Shelving Schedule (Itemize)* schedule is sorted to show the two sizes as separate sub groupings. To do this on the *Shelving Schedule*, add a sort criterion for Shelf Length and give it a header and footer.

Understanding <Family Types> Parameters

The family we have now works OK and gives us decent information in our schedule. But we can still improve it. One problem that it has is evident if you study the *Shelving Schedule*. Here we can see that while we are getting two entries: one for the 4 foot shelves and another for the 4 and half foot shelves, examining the names of the families and types carefully reveals the issue. Since we are driving the nested instance parameters, the type names are no longer logical. Now, we could simply reopen the family and rename the types to use a less specific name that did not include the size. This would be recommended if we were satisfied with everything else about this solution. However, if we would like the solution to more closely mimic the actual construction, it is probably likely that the shelves come in certain pre-manufactured sizes. Therefore the way that the types are currently named is more accurate. In fact, we only went to instance properties to overcome the inability to map the nested type-based parameters in a shared family. There is an alternative. Instead of using mapped parameters, we can instead set up a parameter that allows us to swap out the type of nested family instead. In this way, we gain the advantages of both approaches. We can use nested shared families that can appear on the schedule and give us accurate counts, and we can ensure that we can drive the parameters of the shelves by type-based parameters thereby more accurately representing the actual construction.

To achieve this, we will use a <Family Types> parameter. A <Family Types> parameter allows you to swap out the type of family used for an instance nested inside another family. In other words, if you have a component family inserted (nested) within your family, you can select this instance and label it with a parameter. This parameter will then allow you to change the family and type used by the component parametrically from the host.

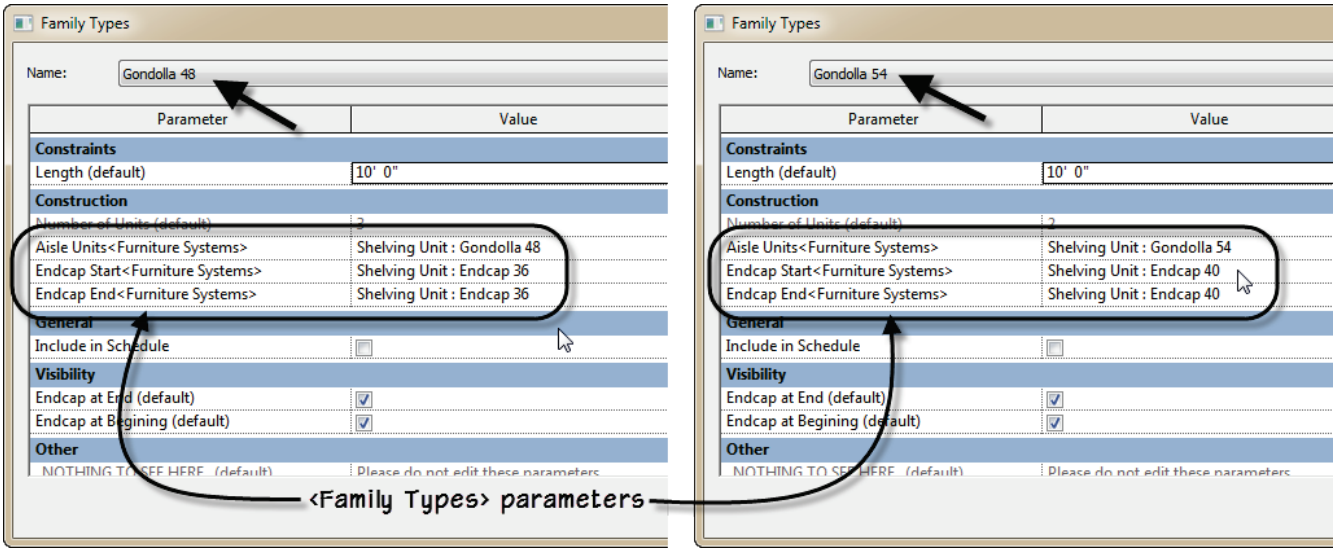


Figure 22—Examples of <Family Types> parameters that change the nested families when a new Type is chosen for the host family

Family Editor 602

Add a <Family Types> Parameter

Adding a <Family Types> parameter is similar to adding other kinds of parameters. You can add one in the "Family Types" dialog and then apply it to an element onscreen, or you can start by selecting an element onscreen and use the Label dropdown on the Options Bar. This is the preferred method as it saves a step. When you create a <Family Types> parameter, you need to assign it to a particular category. If you build the parameter in the "Family Types" dialog, you will be prompted to choose this category. If you start with a selection, the category will be assumed to by the selected object.

1. Save and close all files and then open the ***600 Sandbox_B.rvt*** project file.

This file is the completed version of the previous exercises except that the shelving unit now contains the nested shelving family with type-based parameters.

2. Open the family ***602 Shelving Unit_B.rfa***. (Open from the folder, *not* the project)
3. Open the Left elevation view. Select one of the shelves (it is a Group). Click the Edit Group button and then select the Shelf again.
4. On the Options Bar, click the Label dropdown and choose **<Add Parameter>**.
5. Call it **Shelf Type**, make it a **Type** parameter and group it under **Construction** (see Figure 23). *If an error appears about constraints, click Remove Constraints*.
6. Click the Finish Group button.

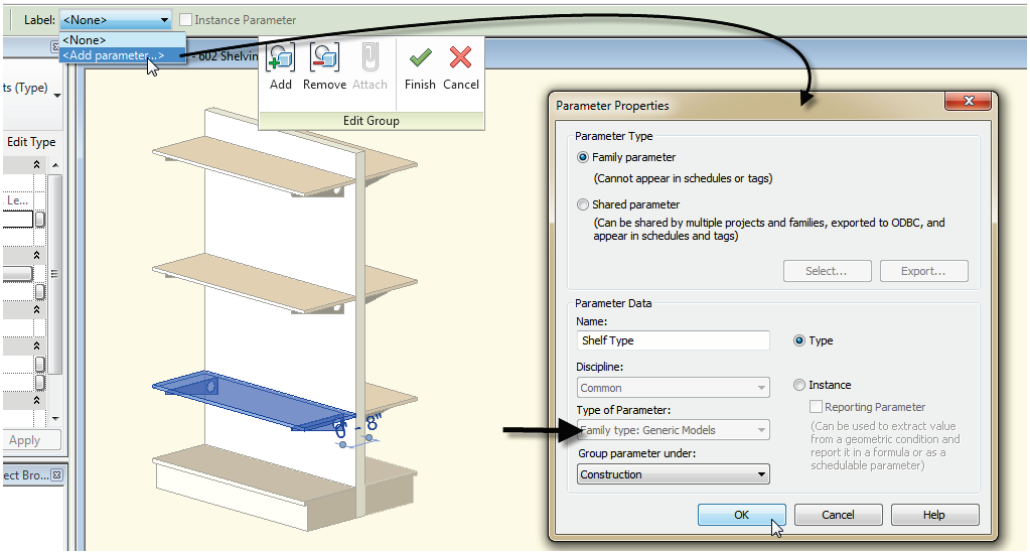


Figure 23—Create a new <Family Types> parameter

Notice that the "Type of Parameter" is automatically set to *Family Type:Generic Model*. This is why it is easier to create the parameter by selecting the element onscreen first and using the Options Bar. When you click OK, notice that this family's instance is labeled with a parameter (on the Options Bar or Properties palette). We can now drive it from the "Family Types" dialog.

7. Add a new dimension at the top to replace the Top Shelf Offset parameter that was just deleted (in the error dialog). Label it with **Top Shelf Offset**.
8. Repeat the procedure on the other side. Use the same <Family Types> parameter and be sure to recreate the Top Shelf Offset labeled dimension.
9. Open the "Family Types" dialog and edit each existing type. For the ones that are named "48", choose the **48"x16"** type for the new Shelf Type parameter. Choose the **54"x20"** type for the "54" types. Flex each one (see Figure 24).

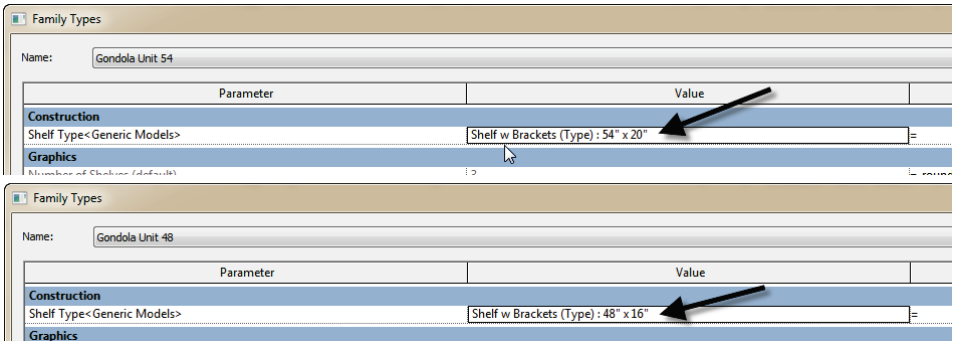


Figure 24—Assign values for the new <Family Types> parameter

10. Reload the family back into the sandbox.

Notice that the shelving units update to the new nested sizes. More importantly, check the Shelving schedule and note that the nested shelves are now listed separately with their correct names describing their actual sizes.

CATCH UP! If you get behind, look for these boxes. I have saved versions of the files at various stages of completion. You can open the file completed to this point named: ***600 Sandbox_C.rvt***.

Conditional Formulas

If you have been building custom family content for a while, then you have no doubt used a formula or two. Formulas can perform simple arithmetic or they can create complex conditional statements that help you trap errors in your families and prevent invalid input. In the AU paper noted in the introduction, there are some simple examples using arithmetic to tie length and width dimensions together proportionally and some more advanced conditional statements to validate the inputs used in arrays. In this topic, let's look at how we can use a formula in conjunction with our <Family Types> parameter discussed in the previous topic.

Consider our retail shelving family from the previous exercise. To expand on this family, it might be nice to create a new family that can place several shelving units along an aisle without having to manually copy them. To do this, we can use a line-based family template. A line-based template has a reference line within it tied to a Length parameter. You click two points along a linear path during placement. You can have geometry flex along this path as the family is placed. In this exercise, we will use the line-based family to place a series of shelving units that are adjacent to one another. Typically, we would use a parametrically driven array for this purpose. However, using a <Family Types> parameter, it is possible to swap out different types for some or all of the units after placement making it more flexible than the traditional array approach. The additional challenge we have is how to ensure that the units stay adjacent to one another when the size is changed using the <Family Types> parameter. In other words, the different family types are different lengths.

Family Editor 701

<Family Types> and Formulas instead of an Array

For this exercise, we will start from scratch. The file is also included at several stages along the way, so look for the catch-up files as we go. There is an Excel file (*Parameter Properties.xlsx*) that contains all the required parameters and their respective settings and formulas. Use this for reference and/or to copy and paste from.

1. Save and close all files and then open the ***700 Sandbox_A.rvt*** project file. Minimize the file.
2. Create a new Family using the ***Generic Model line based.rft*** template.
3. Flex the Length parameter to 20'. Insert a component and when prompted load the ***701 Shelving Unit_A.rfa*** file. Place a single instance near the insertion point.

4. Align and lock the left edge to the Center Left/Right Reference plane and align and lock the center of the horizontal support wall to the Center Front/Back Reference Plane.
5. Make four copies to the right spaced 4'-6" apart.
6. Save the Family as **Aisle Shelving**.

CATCH UP! You can open the file completed to this point named: ***701 Aisle Shelving_B.rfa***.

7. Select the first item, and following the procedures outlined above in exercise 602, label the item with a <Family Types> parameter (line 2 in Table 2).
8. Repeat for each of the other items. Add two additional <Family Types> parameters for a total of 7. Use the Excel spreadsheet for the details (see lines 3 through 8 in Table 2).

Table 2

	Parameter Properties				
	Name	**Type of Parameter**	**Group Parameter under**	**Type or Instance**	**Formula**
1)	_NOTHING TO SEE HERE_	Text	Other	Instance	"Please do not edit these parameters"
2)	Unit 1	Family type: Furniture Systems	Construction	Instance	
3)	Unit 2	Family type: Furniture Systems	Construction	Instance	
4)	Unit 3	Family type: Furniture Systems	Construction	Instance	
5)	Unit 4	Family type: Furniture Systems	Construction	Instance	
6)	Unit 5	Family type: Furniture Systems	Construction	Instance	
7)	CMP1	Family type: Furniture Systems	Other	Instance	
8)	CMP2	Family type: Furniture Systems	Other	Instance	
9)	Unit 1 Loc	Length	Other	Instance	if(OR(Unit 1 = CMP1,Unit 1 =CMP2), 48", 54")
10)	Unit 2 Loc	Length	Other	Instance	Unit 1 Loc + if(OR(Unit 2 = CMP1,Unit 2 =CMP2), 48", 54")
11)	Unit 3 Loc	Length	Other	Instance	Unit 2 Loc + if(OR(Unit 3 = CMP1,Unit 3 =CMP2), 48", 54")
12)	Unit 4 Loc	Length	Other	Instance	Unit 3 Loc + if(OR(Unit 4 = CMP1,Unit 4 =CMP2), 48", 54")
13)	Unit 5 Loc	Length	Other	Instance	Unit 4 Loc + if(OR(Unit 5 = CMP1,Unit 5 =CMP2), 48", 54")
14)	Show Unit 5	Yes/No	Graphics	Instance	Length > Unit 4 Loc
15)	Show Unit 4	Yes/No	Graphics	Instance	Length > Unit 3 Loc
16)	Show Unit 3	Yes/No	Graphics	Instance	Length > Unit 2 Loc
17)	Show Unit 2	Yes/No	Graphics	Instance	Length > Unit 1 Loc

CATCH UP! You can open the file completed to this point named: ***701 Aisle Shelving_C.rfa***.

9. Add a dimension from Reference Plane Center Left/Right to right side of the first shelving unit. Label this Unit 1 Loc.
10. Repeat for the other four units, dimensioning from Center Left/Right to right side of the unit each time and label each one Unit 2 through Unit 5 (see Figure 26).

CATCH UP! You can open the file completed to this point named: ***701 Aisle Shelving_D.rfa***.

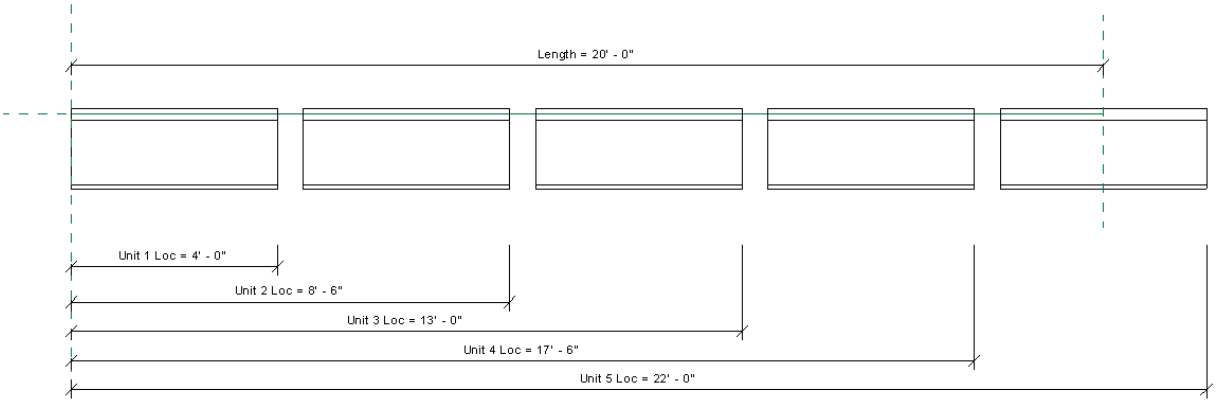


Figure 25—Dimension and label each shelving unit family

We now have <Family Types> parameters driving the kind of shelving unit in each position and a dimension driving each unit's location. Our next task is to instruct the family to calculate the proper distance for each of these dimensions based on the size of the unit chosen for each <Family Types> item. We can do this with a few conditional statements in a formula.

Look at the two parameters **CMP1** and **CMP2**. CMP is short for "Comparison," but you can name them anything you like. Each of these is a <Family Types> parameter. They are placed in the "Other" grouping and you do NOT want anyone to change these values. Simply assign each of these to one of the values you want to compare against. You need one parameter for each comparison possibility. Since we have two 48" types and two 54" ones, or two sizes, we needed two comparison parameters. In this way, if it is not a 48" size, we can reasonably assume it is 54". That is the logic anyhow. *If you have lots more sizes, this solution could become cumbersome* because you would need to add many more comparison parameters and several more nested conditional statements in your formulas.

11. Open the "Family Types" dialog and set CMP1 to: **701 Shelving Unit_A : Wall Unit 48** and CMP2 to: **701 Shelving Unit_A : Gondola Unit 48**.

These values *must not* be changed for any type in this family, or the formulas will fail.

12. Using lines 9 through 13 of the spreadsheet, input the formulas for the dimension parameters.

Two conditional statements are nested here: an IF statement with a nested OR statement. Translated it says to compare the value each Unit <Family Type> parameter (which our end user *can* change) with the two comparison parameters (CMP1 and CMP2, which our user should *never* change). If either one is the same (the OR condition allows either to be true), that tells the formula that the family is 48" wide. If it comes up false, it assumes that it is 54" wide instead. This is then added to the next dimension parameter as it moves down the line.

CATCH UP! You can open the file completed to this point named: ***701 Shelving Aisle_E.rfa***.

13. Add two Family Types named: **Wall Units (One to Five)** and **Gondola Units (One to Five)**. Set the values of each of the "Unit" parameters to the 48" Wall size for the Wall type and the 48" Gondola size for the Gondola type (see Figure 26).

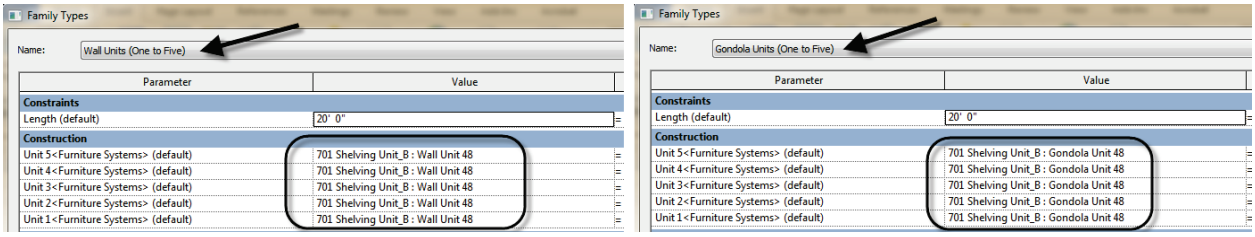


Figure 26—Create Family Types and assign default values for each Family Type parameter

The <Family Type> parameters are instance based, which means that we will ultimately have to set them properly in the project. However, I like to set a default value for each <Family Type> anyhow.

Controlling Visibility

We now have <Family Type> parameters controlling each individual unit and formulas calculating the correct distances between them so that they remain adjacent to one another. The last thing we need to do is have the visibility of each unit tied to the length parameter. In this way, when the line-based family is created in a project, we will only get only one unit if the length is equal or less than the size of one unit, we'll get two if the length is longer than one, but shorter than two units and so on. To do this, we need to create some visibility parameters.

1. Create four visibility Yes/No parameters (see lines 14 through 17 in Table 2). Create the parameters now. We'll add the formulas below.

CATCH UP! You can open the file completed to this point named: ***701 Shelving Aisle_F.rfa***.

2. Open the *701 Shelving Unit_B.rfa* family.
3. Open Family Types and add a formula for the Gondola Vis parameter: **and(Show, Gondola Unit)** (see Figure 27).

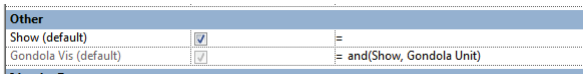


Figure 27—Control visibility in the nested family

When this family was originally configured, the shelves on the wall unit side did not have a visibility parameter: they displayed all the time. The ones on the gondola unit side had a visibility parameter that was toggled on only when a gondola unit type was chosen. This is a simple and effective solution. But the issue here is that we need to allow for an additional visibility condition now. We do not want to see either one if the overall parent family is too short. If you look at the

shelves in this modification of the original family, you can see that there is now a “Show” parameter applied to the wall unit side and a “Gondola Vis” parameter on the gondola side. This formula tells Revit that both have to be true for the gondola units to display. In this way, the “Show” parameter controls the visibility overall and the original “Gondola Unit” parameter still controls only the gondola side.

4. Reload the modified family back into the *701 Shelving Aisle_F* family.
5. Select unit 2. On the Properties palette, link up both the **Visible** and **Show** parameters to the Show Unit 2 parameter.
6. Repeat for units 3 through 5 (see Figure 28).

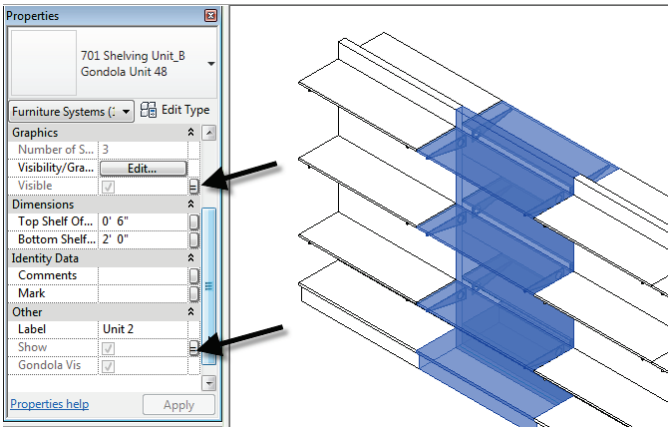


Figure 28—Create Family Types and assign default values for each Family Type parameter

7. Using Table 2 (lines 14 through 17), add the formulas to the four visibility parameters.

These parameters tell each unit to show only if the overall Length of the family is long enough. This in turn triggers the nested “Show” parameter to ensure that the nested items only display if the family is long enough.

8. Load it into the project and test it out.

CATCH UP! You can open the file completed to this point named: ***701 Shelving Aisle_G.rfa***.

An alternate version of the aisle shelving family is provided in the dataset folder. It is called: *Aisle Shelving (Line Based Array).rfa*. It uses an array instead of several nested <Family Types> parameters. The advantage of this one is that it can have more than five units if required. There is likely a practical limit to the number of units you can have along an aisle, so it may be possible and practical to use <Family Types> exclusively. The array example is presented for you to explore as an alternative. This one also incorporates an endcap unit at both ends. This introduces a few new formulas and visibility controls. Feel free to open this family and explore it.

Miscellaneous Techniques

Here are a few additional topics that you should consider when building you custom family content.

Level of detail

When planning out a new piece of family content, level of detail is an important consideration. By managing the level(s) of detail of your family, you can enhance performance and achieve better graphics. Sadly, most families that you come across either out-of-the-box or online use one or maybe two levels of detail. So what you see in Medium is pretty much what you see. With all of the other things that you are likely trying to achieve with your content items, level of detail is easy to overlook. But you are strongly encouraged to look at it anew. If you add two or better yet three well thought out levels detail to your families, it will set you apart as true master family author. Seriously the value of level of detail is hard to overstate. Ask yourself these questions when planning your family:

- What scale is this family most likely to be seen?
- What scale is this family most likely to be printed?
- Are there parts of the family that should only appear under certain circumstances?
- Will this family be rendered?
- Are there times (scales, view types like elevations or sections) when this family should not appear?

Any or all of these issues can be addressed using multiple levels of detail in your family. Here is an example looking at the shelf and brackets (see Figure 29).

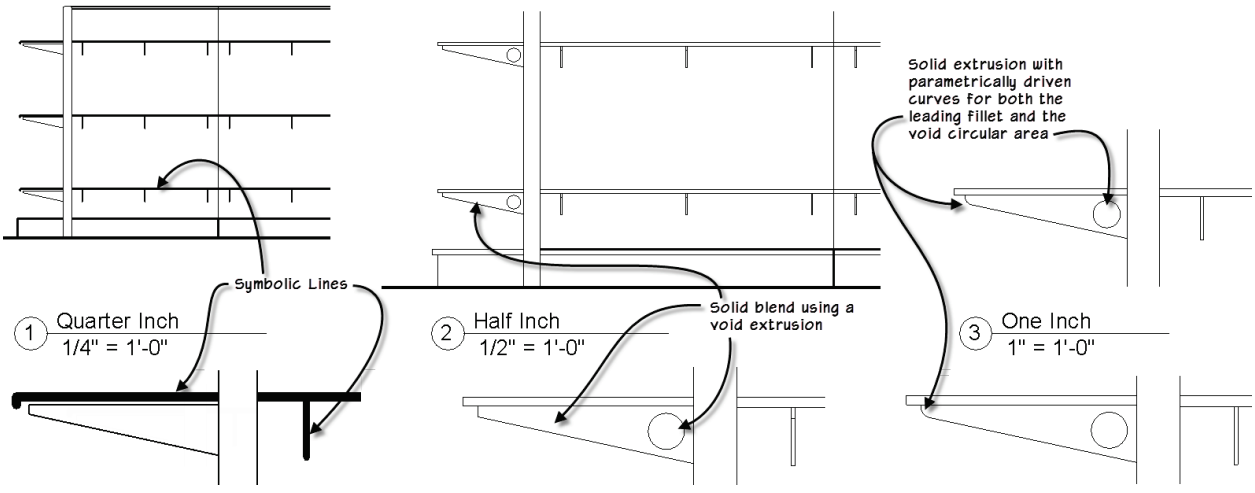


Figure 29—Three levels of detail used on the shelves and brackets

There are two approaches used here. In the Course detail version, Symbolic Lines are drawn in the elevation views to represent the bracket when viewed head on and the shelf's top and leading edge. Since they are Symbolic Lines, they are "self-managing." In other words, the

visibility takes care of its self. By definition, a Symbolic Line only appears in a view parallel to the one in which it was drawn.

A solid blend is used for the basic 3D form of the bracket. It blends from a small rectangle at the leading edge to a taller one at the back. (The steps are covered in detail in the paper for last year's AU class noted in the Introduction to this paper.) This blend is duplicated. One copy is set to Course view only in left and right elevations (see Figure 30).

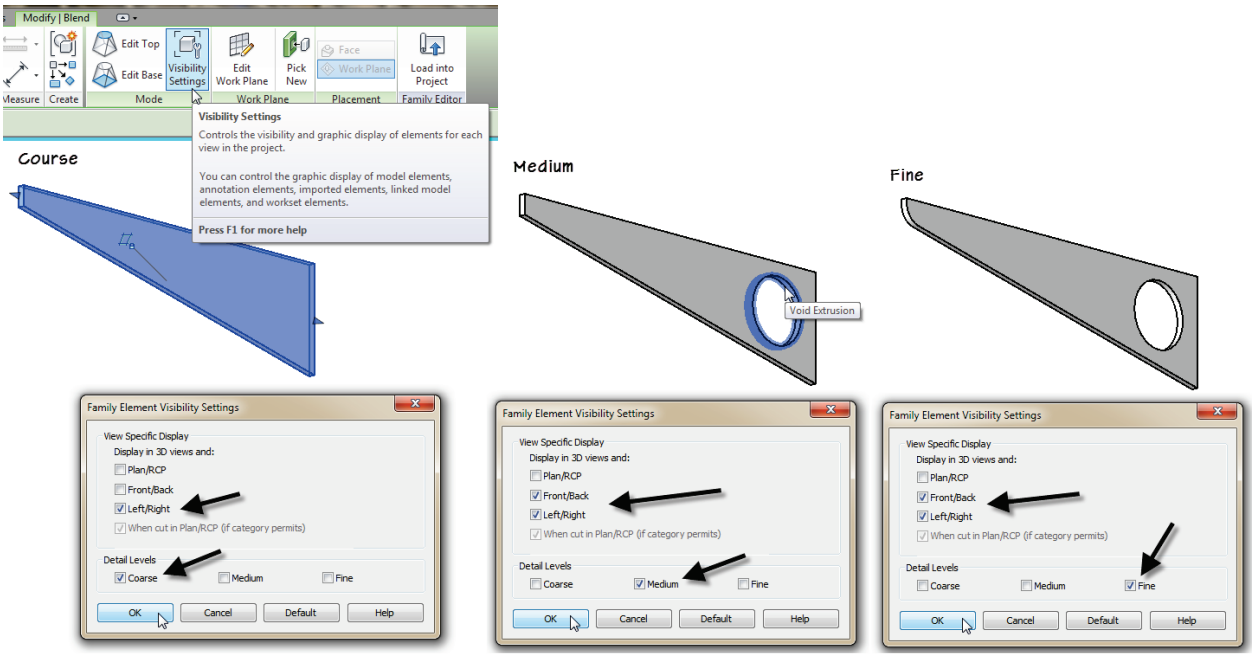


Figure 30—Course, Medium and Fine 3D geometry

For Medium, the other copy is used. It is combined with the void to cut the hole out of the bracket. The joined solid shows in all elevations but only in Medium. Fine uses a different form. Here we have an extrusion instead of a blend and void. This extrusion also introduces a filleted curve to the leading edge.

This is a very simple use of the three levels of detail. It did require three separate 3D forms, which some family authors may not like. But the benefits of having multiple levels of detail should not be underestimated. First keep in mind that the three separate 3D forms do not display at the same time, therefore, you will not suffer a performance hit from them. On the contrary, using simpler forms as we have here for course and medium should actually speed things up. Other ways that levels of detail can be implemented include using Fine views for rendering and course and medium for other types of drawings and even sometimes other disciplines. Just be sure to use them consistently from Family to Family so that the team will come to expect the preferred usage and behavior.

Curves

An entire class could be done on the subject of curves in the Family Editor. Time and space here do not allow such comprehensive coverage, so we will focus on one simple example. To tie it to the previous exercises, let's look at the fillet curve on the leading edge of the Fine detail bracket.

Getting this curve to flex properly when the bracket changes shape is a little challenging. I have explored this quite a bit over the years in various families and have always found constraining curves to be a little challenging. Typically, you must carefully constrain the curve to various reference planes and be sure to flex it thoroughly. If you do not shy away from formulas, and if you remember your high school math, you can add some trigonometry in your formulas which can help. However, trig can add computation time and if you have a lot of instances of the family in a project, can affect performance.

So in choosing an example to show here, I settled on the fillet of the bracket because it was very similar to an example showcased recently on the AUGI (Autodesk User Group International) forums. The example there was a structural family, but the shape is the same kind of shape used on the bracket here. I encourage you to read the whole post and try the different techniques. One of the active participants in the discussion was Dave Baldacchino. Dave is a very talented family author and I have had the pleasure of sharing content creation ideas with him on several occasions. In addition to his contributions on the forum, he and I spoke privately on the matter and he promotes a non-trig solution. You can find a link to his family file on the AUGI forum post. Dave feels that if you can solve the problem without trig, that you will benefit from faster performing families. This is true, so by all means do experiment with his approach. I ultimately decided to show the trig solution here. Partly because my file is slightly different than the one on AUGI and because I was getting more consistent results with the trig. My solution is very similar to one proposed by Alfredo Medina on the AUGI post.

The easiest way to get the whole summary is to visit Steve Stafford's blog again. He waded into the discussion, proposed some alternative solutions of his own and provides a link to the original AUGI post and a link to Alfredo's solution. You can find Steve's post here and links to all others in that post:

http://revitoped.blogspot.com/2012/08/constraining-tangents.html

Family Editor 801

Controlling a Curve with Trigonometry

All of the geometry is provided in this file. We will simply focus on the formulas.

1. Open the file named *Bracket (Start).rfa*.

2. Open the *Left* view select the first blend (squared off corner, not filleted). Click the Visibility Settings button and note the settings. On the View Control Bar, click the temporary hide pop-up (sunglasses) and Hide this element.
3. Repeat for the other blend.

These are the two blend noted above for medium and fine. We will hide them to get them out of our way as we work.

4. Select the Extrusion and then edit it.
5. Using the Align tool, lock the endpoints of the curve in both directions to the Ref Planes. Also lock the endpoints of the diagonal line. The other lines are locked already. Also align and lock the center point of the arc (see Figure 31).

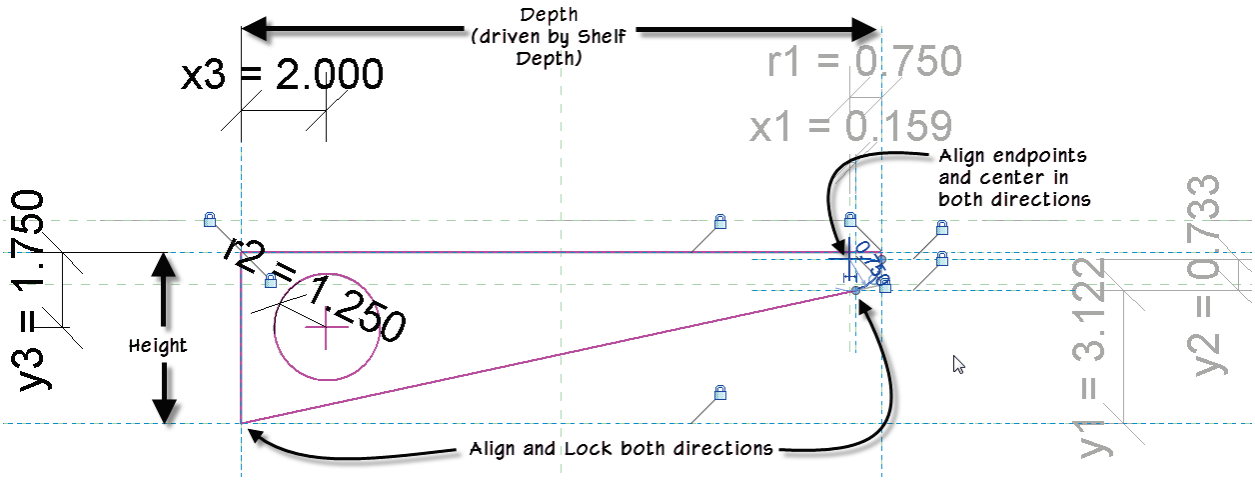


Figure 31—Make sure that all sketch lines are aligned and locked. Lock endpoints of curves and diagonal lines

You can open other views and/or the Family Types dialog to see what we currently have.

Here is what we know: The depth of bracket is the horizontal dimension called **Depth** and it is the main controlling dimension. It is always 1” shorter than the **Shelf Depth**. Shelf Depth is a Shared Parameter and will ultimately be driven by the host family to which it is nested. This will always keep the brackets one inch smaller than the shelves. In this file, you can use the Shelf Depth parameter to flex the family.

Thickness is just the thickness of the material of the bracket and does not impact the shape in this view. **Height** does impact the shape. We can also flex this value to test. It defaults to 4”.

The **r1** parameter is the radius of the fillet curve we are constraining. It defaults to ¾". We can flex this if we want to test, but let’s assume that this value is pretty constant. **Offset** has no impact on our formulas and is used to shift the entire bracket down below the thickness of the shelf material that will sit on top. Three parameters: **r2**, **x3** and **y3** control the circular cutout at

the left and do not impact the fillet. We will ignore these. The formulas simply size them as a multiple of the Height for convenience.

This leaves **x1**, **y1**, **y2** and **A**. Each of these will require a formula that uses trig functions.

Let's start with what we know. We have our **Height** parameter and a constant value of **¾"** that sets the angle of the sloped underside of the bracket (this is angle **A**). The ¾" value could have just as easily been a parameter, but in this case, the two blends used for the medium and course scale geometry use the ¾" fixed value, so no need to change here. We can subtract these two values to arrive at the height of the triangle formed by angle A. The base of the triangle is equal to the parameter **Depth**.

This is all we need to solve for the angle **A**. Also, due to the laws of similar triangles, A is also the angle between the radius (r1) and vertical projected from the center of the arc at the fillet (see Figure 32). Since we know the adjacent and opposite sides (Depth and Height – ¾") we can use the ATAN function.

6. So for the formula of parameter **A**, input: **atan((Height - 0.75") / Depth)**

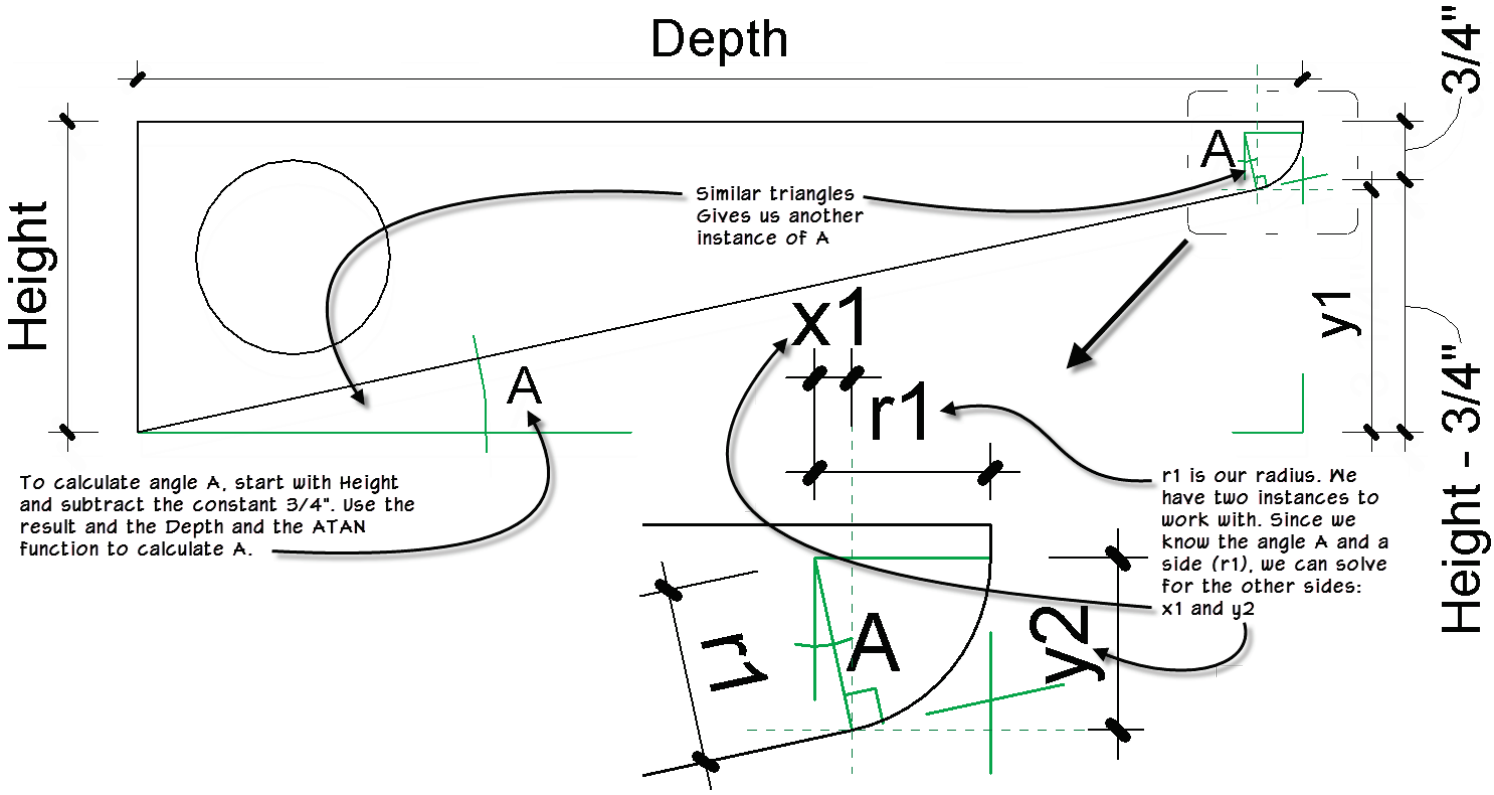


Figure 32—Figuring out the Trig

Next we need the point where the fillet meets the angled edge of the bracket. We will have to calculate both the X and Y position of this point. We have a reference plane at each location

that we need to flex properly. Let's calculate the X value first. This is **x1** or the base of the small triangle formed between the vertical line from the center of the fillet and the radius r1 (see the inset in Figure 32). Which trig function do we need here? Well r1 is our hypotenuse and we have our angle A. x1 is the opposite side. So this means we will use SIN.

7. For the formula for **x1**, input: **r1 * sin(A)**

This leaves us with **y1**. To solve for this, let's look at what we have. We have the **Depth** and **r1** and we now have **x1**. Subtracting these gives us the adjacent side of the large triangle at the bottom. We already have angle A. The opposite side is y1. Since we have adjacent and angle, we need the TAN function.

8. For the formula for **y1**, input: **(Depth - (r1 - x1)) * tan(A)**

Finally we can solve **y2** using COS.

9. For the formula for **y1**, input: **r1 * cos(A)**

The completed Family Types dialog showing all formulas is shown in Figure 33.

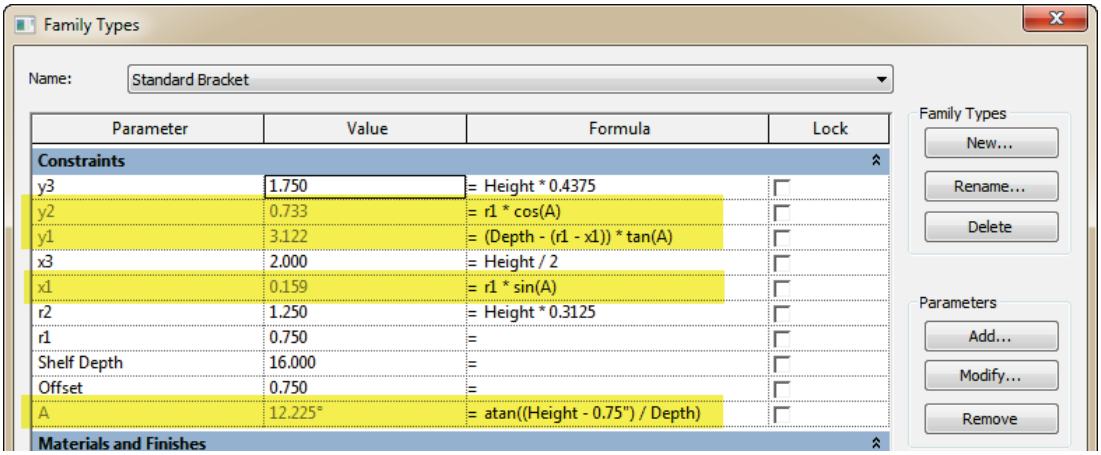


Figure 33—Completed formulas in Family Types

10. When finished, be sure to flex the family to ensure that it works. Remember, the Shelf Depth and/or Height parameters can be used to flex.

If you like, you can load this modified version into one of the Shelf w Brackets families and try it out in the sandbox. We will leave that for you to try on your own.

Sub categories

If you use the above procedures to control levels of detail, you might want to consider implementing some subcategories in your family content. Subcategories can be an effective way to manage lineweights and even materials across multiple families. They should be used sparingly however. If you are not careful, you can end up with many poorly thought our

subcategories that actually work against the original intent which is to help you manage standards. But when used carefully, subcategories give you a convenient way to make a global modification to the lineweight, color, linestyle or material setting of all elements in a project that use that subcategory. For example, in our bracket showcased here, we could create a custom subcategory for the Symbolic Lines used in the course scale views. In this way, you can simply adjust the lineweight of the subcategory in Visibility/Graphics and have it update all families that use that subcategory. Add or modify subcategories in the Object Styles dialog (Manage tab).

Tie it all together with a classical twist

Here is a challenge exercise for you. If you follow my blog: www.paulaubin.com/blog you may have seen some recent post I did on creating families in Revit for classical orders of architecture. Despite being traditional forms, they are VERY complex. My goal is to create fully parametric classical orders that can be used as component families. It is mostly a challenge exercise for me as it is not tied to a client request. The Corinthian capital is proving quite a challenge indeed! As noted in my blog posts, I do intend to publish a book or other resource detailing the steps necessary to create such families. Please check the blog for details as they become available.

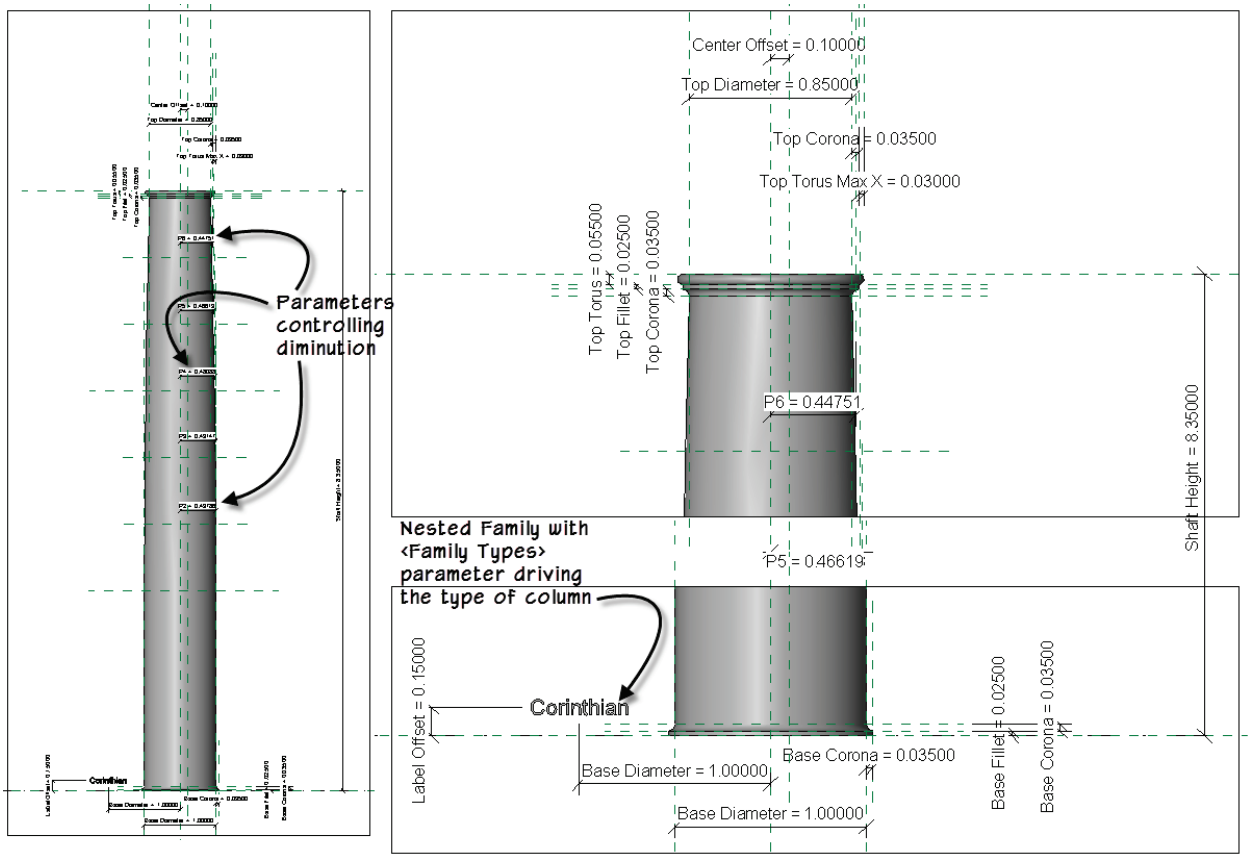


Figure 34—A Classical column shaft for you reverse engineer

In the meantime, I have provided one of the components in the dataset for you to reverse engineer. I will give you a few tips here. Provided in the dataset is a Generic Model Family for the Shaft of any classical order. It is called: ***Shaft All Orders (GM).rfa***. This family is flexible. I have provided several types in it already, one each for the orders Tuscan, Doric, Ionic and Corinthian (I did not do Composite, as its shaft is the same as Corinthian). I also provided four sizes of each: .75, 1.0, 1.5 and 2.0.

Everything is driven from two parameters: Base Diameter is a length parameter that sets the size of diameter of the column measured at the base not including the moldings. Column Type is a <Family Types> parameter that tells it which order. This is my way of overcoming the inability of doing a "list" parameter. So hopefully at some future release, we will get a list parameter (and while we're at it, concatenation would be nice as well…) The Column Type parameter is compared to each of the four comparison parameters to determine the type of column. This in turn triggers different sizes for various moldings and measurements.

Finally, if you dig a little deeper, you see that this column uses accurately calculated diminution (sometimes referred to as entasis). The source material I use preferred the term diminution over entasis siting differences in result with entasis actually forming a bulge above the base making the diameter at this maximum point larger than the base diameter. As I understood it, this is actually the way some Greek versions of the orders do it. I opted for diminution instead (perhaps a more Roman/Renaissance interpretation). I tried this a few ways, and to make it completely accurate, I would have needed to use separate curves or a spline. This would have complicated the construction or forced me to use an adaptive component. I choose to stay in the traditional family editor and approximate the result of the diminution with a single large arc instead. If you zoom in very closely, you can see that it does not completely match the construction points, but since the variance is in thousands of a module, I felt that I could safely accept this tolerance.

The column geometry is a sweep. It could also be done with a revolve. I chose a sweep because in some iterations I used a nested profile family. In the one I gave you here, it is just a sketched profile. So feel free to try a revolve if you prefer. (Be nice to have profiles for Revolves now that you mention it… anther wish, I got a bunch of 'em).

This example brings together most of the topics covered in this paper including: <Family Types> parameters, driving formulas with <Family Types> comparisons, conditional formulas like IF, OR, AND and NOT and trigonometry. I do not have any nested families in here other than the driving Column Type family and I did not use Shared Parameters, but I certainly could have.

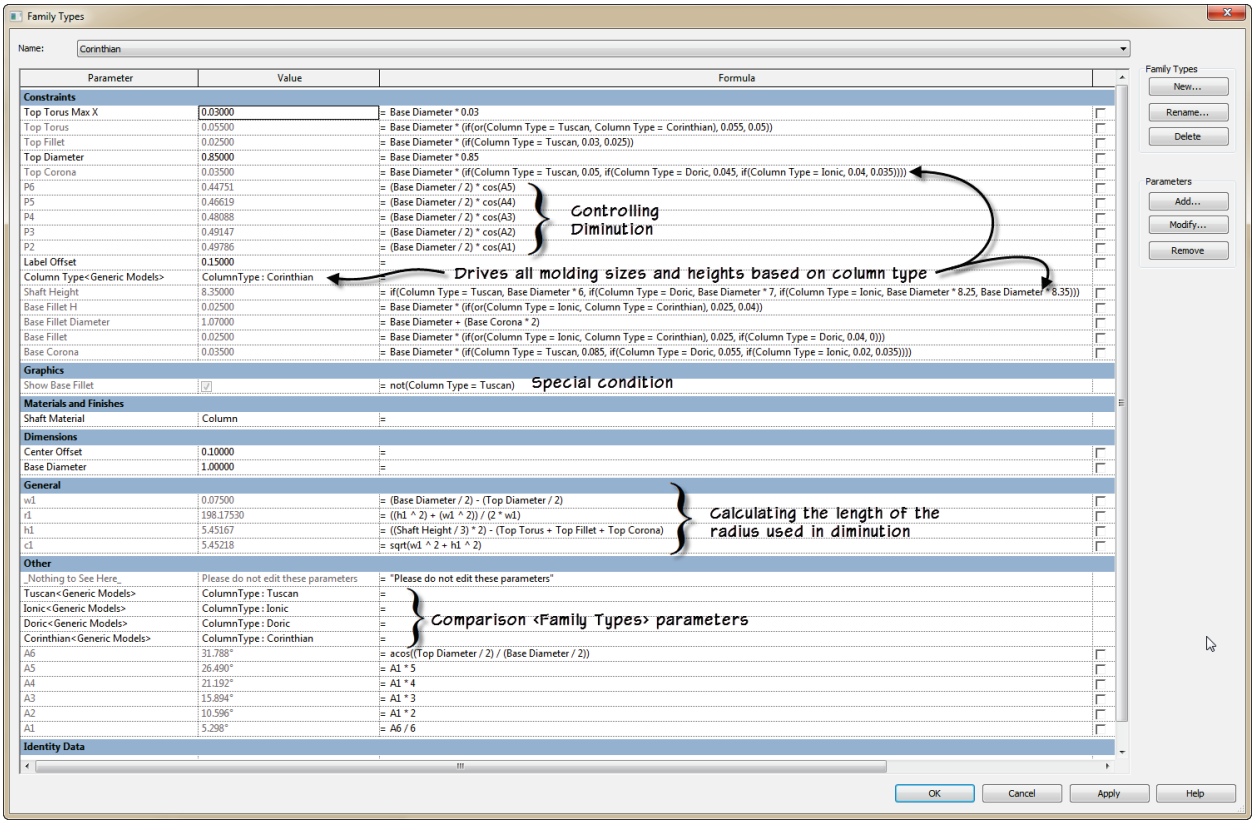


Figure 35—The formulas and parameters for the Classical column shaft family

Finally, there are 16 types in this family. Since I did not cover Type Catalogs in this lab, I chose to embed the types directly in the family. However, best practice dictates using a Type Catalog when you exceed about half a dozen types. I will let you explore Type Catalogs and try your hand at creating one for this family if you wish.

That's all I have for now. Thank you for attending!

Please feel free to experiment further. Thank you for attending.

Further Study

You can find more information and tutorials in *The Aubin Academy Master Series: Revit Architecture*.

Past Autodesk University Class:

"Autodesk® Revit® Families: A Step-by-Step Introduction"

I also have Revit video training available at: **www.lynda.com/trial/paubin**. I have five courses at lynda.com: *Revit Essentials (2011 and 2013)*, ***Revit Family Editor***, *Revit Architecture Rendering* and *Advanced Modeling in Revit Architecture*.

The Revit Family Editor course is devoted entirely to the Family Editor and content creation. The Advanced Modeling course covers the Massing Environment as well as many other related topics.

If you have any questions about this session or Revit in general, you can use the contact form at **www.paulaubin.com** to send me an email.

Follow me on titter: **@paulfaubin**

All of your lab monitors from today also have resources available and have all taught classes here at Autodesk University. Do a search for:

Matt Dillon, D|C|CADD, Desiree Mackey, Martin / Martin Inc., Matthew Stachoni, CADapult Ltd, Don Bokmiller, Clark Nexsen, James Smell, Autodesk

Thank you for attending. Please fill out your evaluation.