

# Tips & Tricks

## Welcome

iLogic is the next evolution of parametric design. “Parametrics 2.0” if you will. Parametric design is about defining and driving model geometry using dimensions and equations (or the occasional linked spreadsheet). Logimetric design is about driving the dimensions, equations, attributes, features, components, properties and every other aspect of not only the model geometry but the model document using simple logical expressions that define the logical relationships between all aspects of the design based on design rules. Logimetric design has been developed and is intended for designers and engineers with little or no programming expertise. While it is true that you will have to learn a few basic programming concepts to become proficient at logimetrics, in no way does it require you to be a “true programmer. Below are some do’s and don’ts when you are getting up to speed with iLogic.

### Don’ts

1. Don’t try to “boil the ocean” on your first iLogic project. Pick something you can accomplish in a single day or less. There is a lot you can do and for you to learn about iLogic. Learning and applying the basics in a simple project that returns value to your process quickly will encourage and inspire you to do more, better and bigger!
2. Don’t write rules that reference ambiguous (default) parameter, feature or component names. (ie. “if d0 = 1.25 then FeatureIsActive(“feature1”) = false” ) Something like “if Width = 1.25 then FeatureIsActive(“MountHole”) = false” makes understanding what the rule is actually doing much clearer and easier to understand.
3. Don’t change parameter, feature or component names after the rules are written. Doing so would cause rules that reference these elements (by their names) to fail because changes to the names are not automatically reflected in the existing rules. If you decide to “do” this “don’t” you will have to edit the rules and manually update names before the rules can work properly once again.
4. Don’t start writing rules without a plan. (without really knowing what the rules are)
5. Don’t write really long rules that could otherwise be broken up into much smaller bite size rules. Short rules are much easier for those who didn’t author them to read and understand. They will also be easier for **you** to read and understand when you find yourself looking at them again six months from now trying to remember “how did I do that?”
6. Don’t expect rules to work in a model that you can’t even modify the parameters of manually without it blowing up every time. Remember, it is possible to create a parametric model with such sloppy methods that not even its author could re-use it to define a new configuration!
7. Don’t expect rules to be able to do what you can’t already do manually with inventor. Rules must themselves follow Inventor’s “rules”. Think of it this way; rules don’t change how

Inventor works but rather what “works” Inventor. Instead of an operator doing a bunch of manual, menial design reconfiguration tasks, rules do the work. How nice!

8. Don't use hard coded paths and document names in rules where you could otherwise use relative paths and document names. Design paths can change when copies are made, or when file folders are moved. iLogic supports methods for defining relative paths and filenames so your “logimetric” models don't get confused when such things happen.
9. Don't put rules in iFeature, iPart or iAssembly factory documents. This is not supported. (yet?) Embedded tables drive factory members, not rules.
10. Don't allow people who do not have the iLogic add-in installed to modify your design documents that include rules. If you receive such a design back and continue making changes to it yourself, rules will fire and the result will be .....well.....who knows? Rules won't fire and do what they are supposed to in an Inventor session where the iLogic add-in is not running. Give such a person a copy of the design with the rules deleted if they need to modify the design and then give it back to you and all will be well.

## Do's

1. If you're an iLogic rookie do complete the tutorials that come with the iLogic install and have a look at the iLogic sample designs that also get installed automatically. (**iLogic 2010 Samples.ipj** and **iLogic 2010 Tutorials.ipj**) These projects will help you to quickly familiarize yourself with the basics of the iLogic application as well as simple and common rule methodologies you are likely to employ in your own “real world” logimetric projects.
2. Review the iLogic help documentation before attempting your first iLogic project. I know you all love reading help docs (yuck!) At least familiarize yourself with the table of contents and poke around a bit so you know where to go when basic questions arise. You can access the iLogic Help Docs by clicking the **About iLogic Extension** button from the **iLogic Utilities** dialog and then clicking the **Help** button from the **About Inventor iLogic** dialog. You can also access the iLogic Help Docs by pressing the **F1** key on your keyboard while the iLogic parameter or rule editor dialogs are active.
3. Do give meaningful names to parameters, features, and components in your models that will be directly referenced by rules. Do this before writing rules. Refer to item #3 from the top ten **Don'ts** list if you want to know why.
4. Do change all component names in an assembly that will be referenced by rules. Changing a components name “stabilizes” the name so that future changes to the file name the component points to (such as when copying the design and renaming the files) will not affect the components name. This ensures that rules will continue to work properly if an when file names get changed. You will also need to create a custom level of detail in any assembly document before you can write rules in that document that will drive component suppression states. iLogic will not let you create such rules without a custom level of detail.
5. Write out (in plain language) the rules you plan to write using iLogic. Determine key parameter, feature and component names in advance of building the actual parametric base model. Don't worry about syntax at this point, but rather focus on completely expressing

what all the rules are and try to cover all the cases you want to handle as exhaustively as possible. Napkin and pencil will do just fine.

6. Build (or rebuild) your Inventor model as necessary to support the full scope of modifiability that you hope to achieve using rules. Rules can only do to a model what you could otherwise do manually without update failures. iLogic is not generative (doesn't create anything) but instead modifies, rearranges and reconfigures what is already there. What you want to build is a sort of "super model" if you will that is an overloaded master template containing all things necessary to represent all states or possible (or allowable) configurations of a design.
7. Use rules in assemblies to do assembly level things (ie. Suppress/unsuppress components, change component pattern quantity, replace components, change ipart configuration, etc). Use rules in parts to do part level things (ie. Suppress/unsuppress features, drive parameter values, set iproperties values, etc). Use an assembly level rule (usually the last rule in the list of rules contained in the assembly document) to pass parameters values down to the parts it contains where rules are fired as a result of those parameter values changing. The idea is to have assembly level parameters set by assembly level rules whose values would be pushed down into parts having parameters with the same names that would cause a part level rules to fire that would update the parts. Whew! That was a mouthful!
8. Keep rules as simple and as small and concise as possible. A rule should be dedicated to one or two tasks and its name should give a clear indication about what the rule is all about. Rules that are "short and sweet" make managing the knowledge captured much easier for you the original author of the rules, as well as for those that may inherit your work and try to re-use or even just understand what the rules are doing.
9. Include sufficient comments in the rules you write to make understanding them easier for those who will attempt to read and perhaps even edit them later.
10. Avail yourself of the many good books or on line resources that can help you develop your basic understanding of VB.net programming. You certainly do not need to be an expert VB programmer to write rules that do a lot of really valuable and powerful things with iLogic. I am living proof of that! Keep in mind that you can continually expand the possibilities of what you can do with iLogic by expanding your general understanding of what can be done with VB or programming in general.