Applied Software®

Learn from the Best: iLogic

Written by Carol Dunn & Jason Miles

ASTI.com

Table of Contents

INTRODUCTION

If your product design process would benefit from automating repetitive tasks, iLogic should be in your future. Included with Autodesk Inventor, iLogic adds another level of intelligence in your design work. It provides a simple way to capture and reuse your work with rules-driven design. With iLogic, you can standardize and automate design processes when configuring your virtual products, saving time and boosting productivity.

Inventor is a 3D parametric design application. Using iLogic, you can embed rules as objects directly into part, assembly and drawing documents. The rules determine and drive parameter and attribute values for your design. With this kind of rules-driven design, you can define the behavior of attributes, features and components of a model. Knowledge is saved and stored directly in the documents the same way geometric design elements are stored. iLogic rules can make use of custom parameter types available in Inventor, like text, true/false and multi-value lists. These parameter types allow rules that involve more than numeric input values.

CHAPTER 1 | BASICS

What iLogic Is Used For

If you've used desktop software applications, you've taken advantage of automation. For instance, many of us have used macros in Excel to automate tasks. The automation employed by Autodesk Inventor is similar to that. It uses a series of tools to automatically accomplish a task, process, or function, and iLogic is one of those tools.

iLogic enables users to create logic in the form of Visual Basic VB.net to accomplish tasks. Rules are developed and organized using snippets (small regions of re-usable source code) and other code writing statements. They are set to run at specific intervals to dependably perform some of the tasks that engineers and designers would otherwise need to perform.

Examples of iLogic performance rules include: changing text throughout a model, updating a model's properties and replacing components of an assembly.

In fabrication and manufacturing, there are patterns in production repeatable processes. The simple task is to find the processes that iLogic can help with.

If you have a specific format for an iProperty description in your model that is predictable and standardized, then you can develop logic to collect information from the model, transform the information and overwrite the iProperties with the correct, newly formatted information. It is always correct and consistent.

iLogic can also benefit you in automating drawing generation, dimensioning, adding comments, and updating materials lists.

Components and Management Behavior

iLogic assembly functions make it easier for you to write rule code to add, modify and delete components and constraints:

Components.Add: Ensures the component exists and that it has specified properties.

Components.AddiPart: Creates or updates an occurrence using an iPart factory and specification of a member.

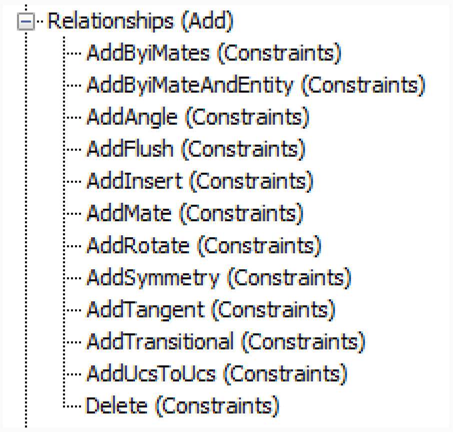
ThisAssembly.BeginManage and ThisAssembly.EndManage: Both allow you to delete components without specifically calling Components.Delete. If you have set the parameter for a component in the assembly to True or False, then it will be added or updated if True and deleted or not updated if False. iLogic ensures that a component is present only if it is labeled as True. Every time a BeginManage/EndManage block runs, you must call a function to add only the components you want included in that block.

Document Units Geometry

There are iLogic functions which use document-units instead of database-units for the coordinate values and objects representing points, vectors, and matrices. These objects can be used to specify the location and orientation of components created by Components.Add and related functions. Use ThisDoc.Geometry.Point() or similar functions to create the objects.

Assembly Add Constraints Functions

The iLogic Relationships (Add) functions allow you to create an iLogic assembly with occurrences and constraints generated by a rule. The rule will regenerate or update the content if any of the relevant parameter values are changed.



Relationships (Add)
- AddByiMates (Constraints)
- AddByiMateAndEntity (Constraints)
- AddAngle (Constraints)
- AddFlush (Constraints)
- AddInsert (Constraints)
- AddMate (Constraints)
- AddRotate (Constraints)
- AddSymmetry (Constraints)
- AddTangent (Constraints)
- AddTransitional (Constraints)
- AddUcsToUcs (Constraints)
- Delete (Constraints)

Capture Current State Commands

The Capture Current State command has been enhanced to support:

Capture Current State (Components.Add): Captures the state of the selected components using Components. Add. Ignores related constraints.

Capture Current State (Components Constraints.Add): Captures the state of the selected components and related constraints using Components.Add.

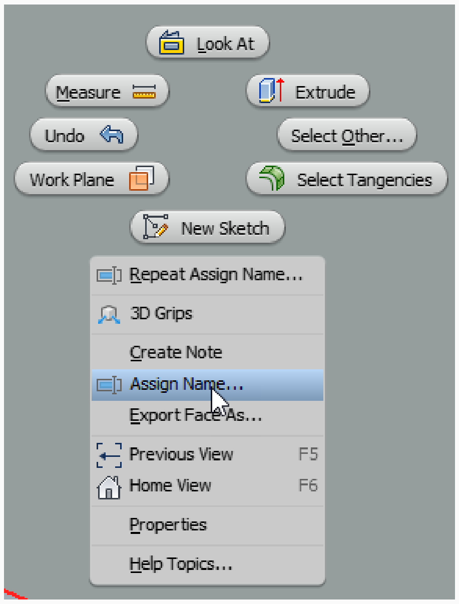
Capture Current State (Constraints.Add): Captures the state of the selected constraints using Constraints.Add. Only available if the file contains a constraint.

Capture Current State (Patterns.Add): Captures the state of the selected pattern. Available if the file contains an assembly pattern.

Multi-select components in the Model tree in the Rule Editor to capture the state of multiple components all at once. If you are capturing constraints between components, this ensures they are only captured once.

Assign Name Will Identify Geometry for Constraints

You can assign names to faces and edges and then create a rule that adds the constraint(s) directly to the face or edge with the assigned name. There is no associativity between the assigned name and the Rule Editor. If you change an assigned name in a part file, you will need to recreate the rule or manually change the assigned name in the rule editor.



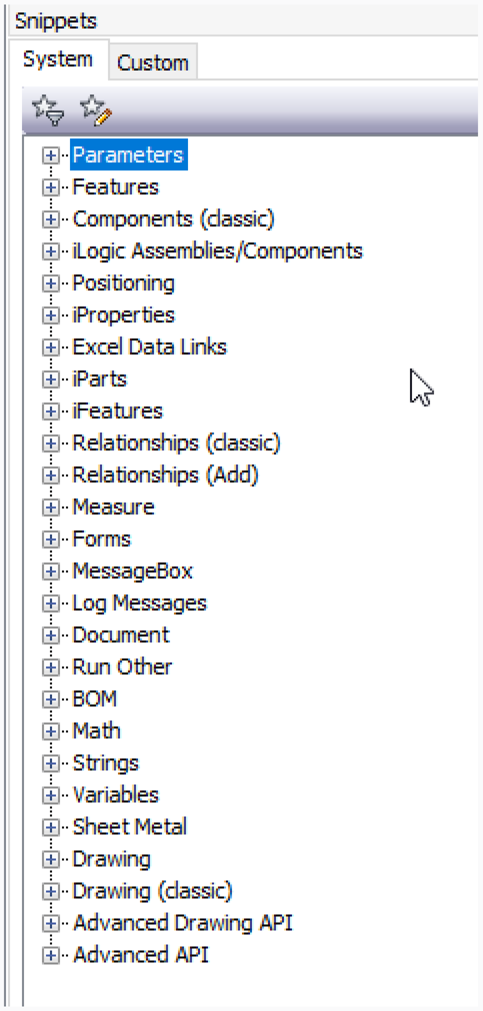
Help for Functions

Help for iLogic functions and arguments are available from the [iLogic API Reference] help node. In the reference, you can also find information on the functions supported in iLogic. For help with commands and rules see [To Work with Rules in iLogic].

***Note**:* iLogic can change your components-documents-assembly workflow dramatically. If it's time to try iLogic, contact the experts of [Applied Software] today for more information.

Glossary of iLogic Terms

Each of the following iLogic rule functions is accessed by expanding the appropriate node under the System tab in the Snippets area of Inventor.



Advanced Drawing API

If you want two applications to communicate with each other, you can use Application Programming Interface rules to achieve that, including the drawing document, the underlying sheet, the underlying view object, or the document for the current view.

Advanced API

Go a step beyond the drawing API to the current Inventor session, document where the rule is stored, part or assembly document, automation, component, or feature.

Assembly Constraint

Define optional configurations for a component in an assembly by constraining or unconstraining the positions of the component.

BOM

Change the bill of materials, override calculated quantities or export BOM data to Excel.

Component

Replace one component with another to change configuration of parts or assemblies.

Document

Access the underlying document or file name for an Inventor part or assembly.

Drawing

Update a model efficiently by customizing the way changes are depicted in the drawing sheets.

Excel Data Links

Read and write to Excel spreadsheets, either from tables or specific cells. Also export bills of materials to Excel.

Feature

Set or read colors, suppression states, and thread and tapped holes properties.

iPart and iAssembly

Change the configuration of an assembly. **Custom** parameters can be used to drive part configuration.

iFeature

Change the active row of a table-driven iFeature where each row contains different parameters to drive the feature.

Math

Add your choice of 13 standard math functions to your function rules, 5 trigonometry functions, 6 comparison functions, 9 others, plus custom iLogic math functions.

Measure

Find and list values for area, perimeter, distance, extents and angles.

Message Box

Create message and data input boxes.

Parameter

Change parameter values of components directly.

Run Other

Run a rule that is set to not run automatically or one that does not have trigger parameters.

Sheet Metal

Read or set the current sheet metal rule in a sheet metal part, and change the rule automatically depending on type or thickness of material.

Standard and String

There are 7 standard string functions for text in your rules, plus 5 specialized string functions.

Variables

Pass data between rules with shared variable or new array functions, which are stored in memory and are not associated with any part or assembly.

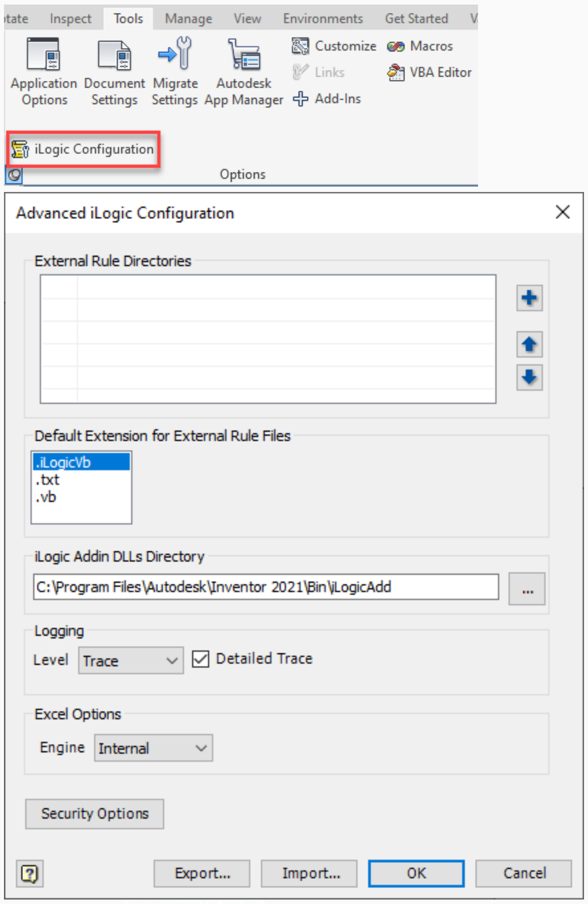
Work Feature

Redefine a sketch, or redefine the workplane by adjusting/flipping the normal vector.

CHAPTER 2 | CONFIGURATION

iLogic is included in Inventor, so you will need to change some Inventor settings first before getting the maximum use from iLogic.

The iLogic Configuration button specifies where Inventor will find external rule directories and their priority order of use. One example is the directory location for dynamic link libraries (DLLs), which originate from Microsoft Visual Studio and trigger iLogic logic and rules.
Users can set file extensions for saving external rules and the default logging level where debugging information will be produced. There are also some security options settings to protect computer systems from potentially hazardous code running within Inventor.
After you set these options, you can begin using the features of iLogic.



Advanced iLogic Configuration

External Rule Directories

Default Extension for External Rule Files

.iLogicVb
.txt
.vb

iLogic Addin DLLs Directory

C:\Program Files\Autodesk\Inventor 2021\Bin\iLogicAdd

Logging

Level: Trace ☑ Detailed Trace

Excel Options

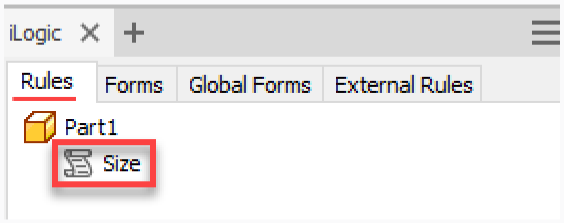
Engine: Internal

CHAPTER 3 | RULES

There are internal and external iLogic rules. They are both created within Inventor in the iLogic browser, and they are both visible within the iLogic browser. Either type of rule can be controlled, suppressed, triggered, and deleted from the list.

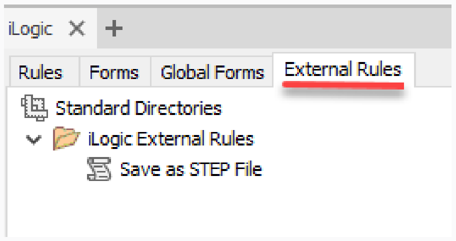
Internal Rules

Internal rules are created and stored within the context of a file. Part, assembly and drawing files all have the capability to store, compile and run rules to affect each file differently.



External Rules

External rules are similar to internal, but they are stored in a directory outside the files – on a server or one user's local system – to make them less accessible to users and less prone to being accidentally altered. Because external rules have a higher level of security, they are preferred in cases where permissions are required to edit rules, such an enterprise setup.



CHAPTER 4 | PARAMETERS

Parameters are named value placeholders of a specific type. Inventor parameters are mostly numeric and associated to dimensions that control geometry. When parameter values change, the dimensions associated to those parameters also change, graphically updating the models.

Inventor Parameters Include:

Model

These are parameters created by normal Inventor behavior. In the Parameters dialog box, they are automatically named as d0, d1, d2. Model parameters are created and deleted as necessary by the Inventor system. It is preferred that users don't rename model parameters.

User

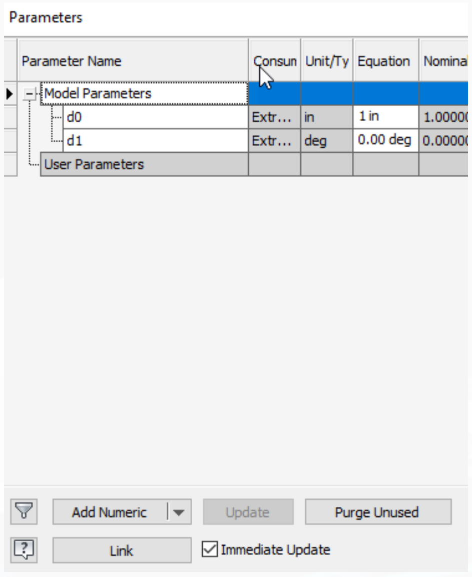
Parameters created by users can range from text/string, Boolean, numeric, and true/false. They are useful because, while they are used by different features and iLogic code, they are not created or deleted by the Inventor system. It is standard operating procedure to create unique names for user parameters.

Reference

These parameters are created when Inventor defines a "driven dimension," which forces the sketch outside its constraints. The parameter can be named to help with the iLogic code, but the value cannot be changed.

Linked

When parameters are connected to Inventor from an application like Excel, they are referred to as linked. When a user updates the names and values in the Excel spreadsheet, the changes reflect in Inventor and control features, dimensions and assemblies.

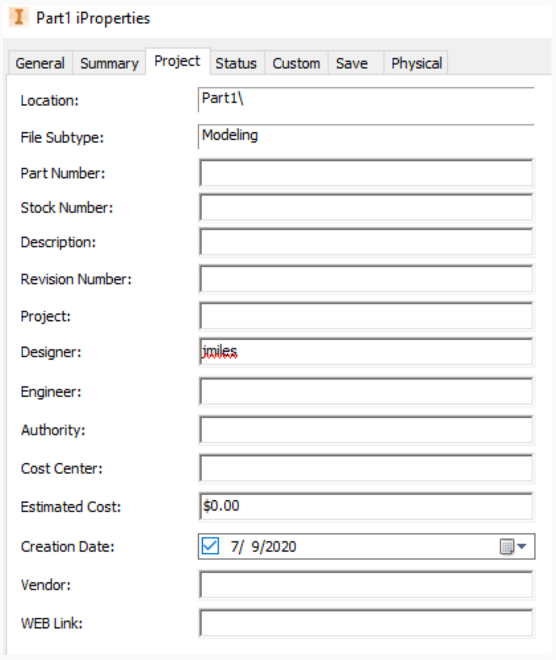


Parameter Name	Consun	Unit/Ty	Equation	Nomina
Model Parameters				
d0	Extr...	in	1 in	1.00000
d1	Extr...	deg	0.00 deg	0.00000
User Parameters				

[✓] Immediate Update

iProperties

Inventor also uses metadata to further define your files. These iProperties include things like file name, file size, file path, author, date modified, part number, description, cost, stock number, and others. They are useful when you want to amass data about your files. iProperties are read-enabled, and most are also write-enabled.



Location:	Part1\
File Subtype:	Modeling
Part Number:	
Stock Number:	
Description:	
Revision Number:	
Project:	
Designer:	jmiles
Engineer:	
Authority:	
Cost Center:	
Estimated Cost:	$0.00
Creation Date:	✓ 7/ 9/2020
Vendor:	
WEB Link:	

CHAPTER 5 | VARIABLES

There are some protocols that all programmers understand. Even though iLogic is simple code, and you don't have to have programming experience to use it, an awareness of code writing best practices will help you get productive faster. Declaring variables and typecasting are examples of those practices, and they will give you a standard for writing logic.

Declaring

Declaring variables in iLogic is as simple as writing a name and giving it a value:

CylinderHeight = 50

Once you create a variable you can manipulate it. The value can be read and processed in a calculation, or you can write to it in order to update something else. Although simply pairing the name and value is acceptable in iLogic, you can leverage a code writing best practice and set the name, give it a "type," and then provide a value:

CylinderHeight As Double = 50

This is referred to as typecasting.

Typecasting

This tells iLogic to create a variable that will only hold a "double" value and then provide the value. Typecasting ensures that only a specific value can be provided to the variable. Values other than height, for instance, will cause the code to fail. This method is useful for constructing logic needed for calculations, manipulating other parameters and passing values to other constructs.

Specifying a type allows you to use more complicated code in your rules and visualize the flow of information at the same time. It also allows you an easy method to debug your code.

Shared

Shared variables are another feature of iLogic. When declaring variables in an iLogic rule, that variable is only accessible within the context of that rule. If you need to create a variable and set its value to use in numerous rules, then shared variables are used.

In the iLogic Snippets panel of the iLogic rule editor, the Shared Variable functions are under the Variables index.

To use shared variables, use a similar process to declaring other variables:

First declare the shared variable, then provide a name and value for it. The value can be a static value, or it can be the value of some other parameter, property or variable.

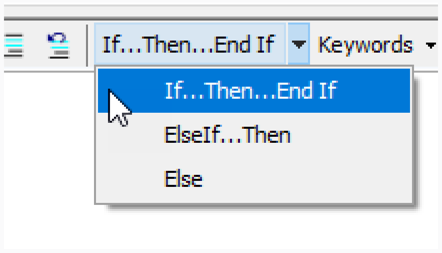
Once a Shared Variable has been declared and a value provided, then it can be consumed and updated as necessary.

Use the other Shared Variable functions to see if a shared variable exists or to remove any and all shared variables from memory.

CHAPTER 6 | DEFINING THE DECISION-MAKING PATH

When looking at iLogic rules, you will need to define the decision-making path by taking into account the different conditions that might exist within a model. Using expressions that define different conditions is a way to accomplish this.

Conditional Expressions



The most common conditional expression is the "If Then" expression:

```
If someValue = True Then
'Do Something
Else
'Do Something Else
End If
```

When using code, you look to see if a condition exists, and if it does, then the code will do something. If the condition does not exist or if a different condition exists, then the code will do something else.

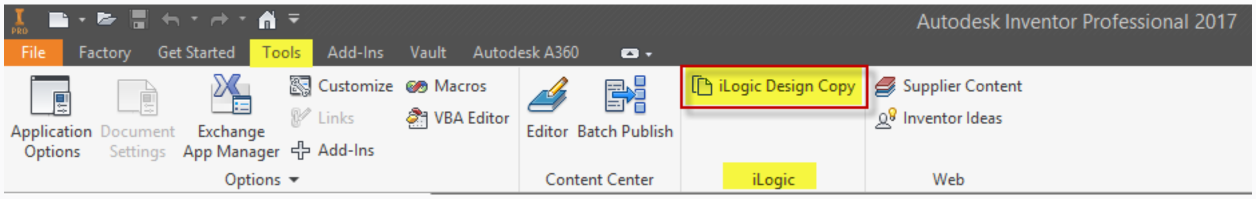
In its simplest form, this is easy to follow, but once you add in several conditions and more operators to explore numerous conditions, it can get confusing. At that point, there are better ways to handle the code.

Loops

An essential method for writing code is the concept of loops. For example, when you are doing iterations of an assembly to get the names of components, loops allow you to go through all the occurrences without having to know how many exist. While constructing code and developing logic involve understanding patterns, consistency and predictability, loops accommodate the unpredictable.

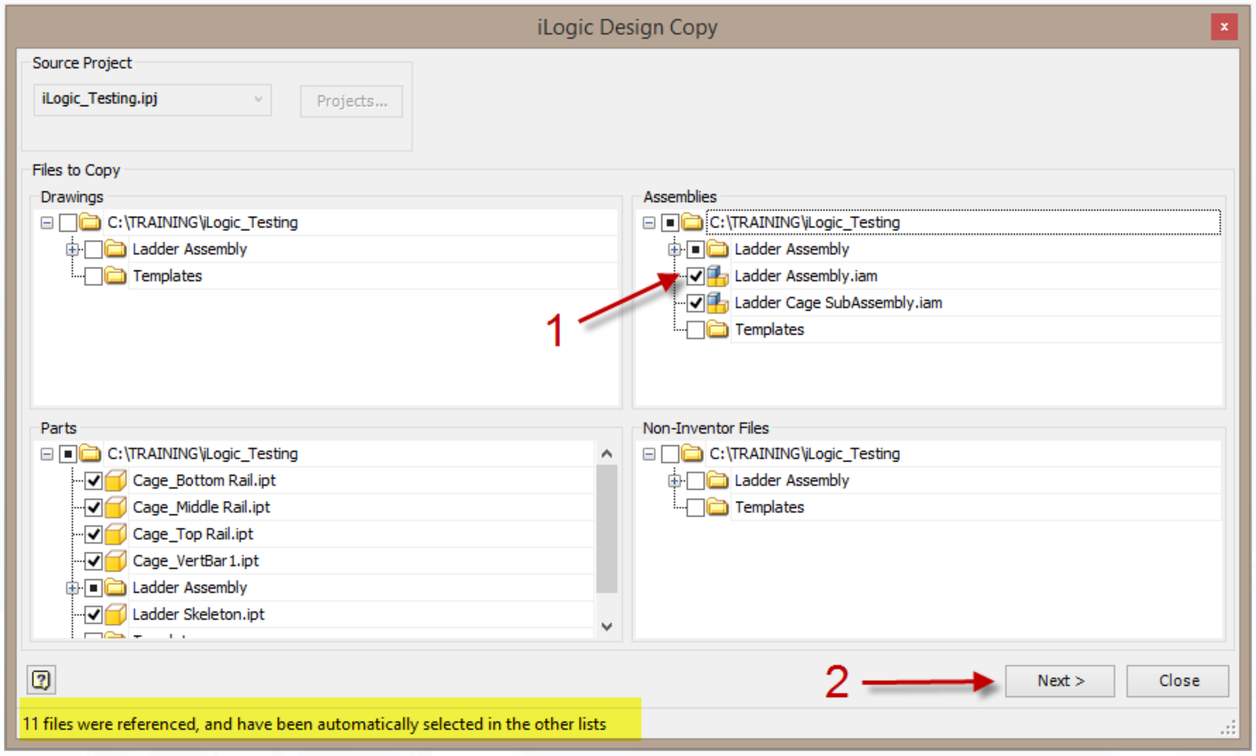
CHAPTER 7 | USE CASE: DESIGN COPY

Using iLogic Design Copy is a great way to copy an assembly that is set up to drive parts from an assembly by using iLogic code. This process is fairly simple.



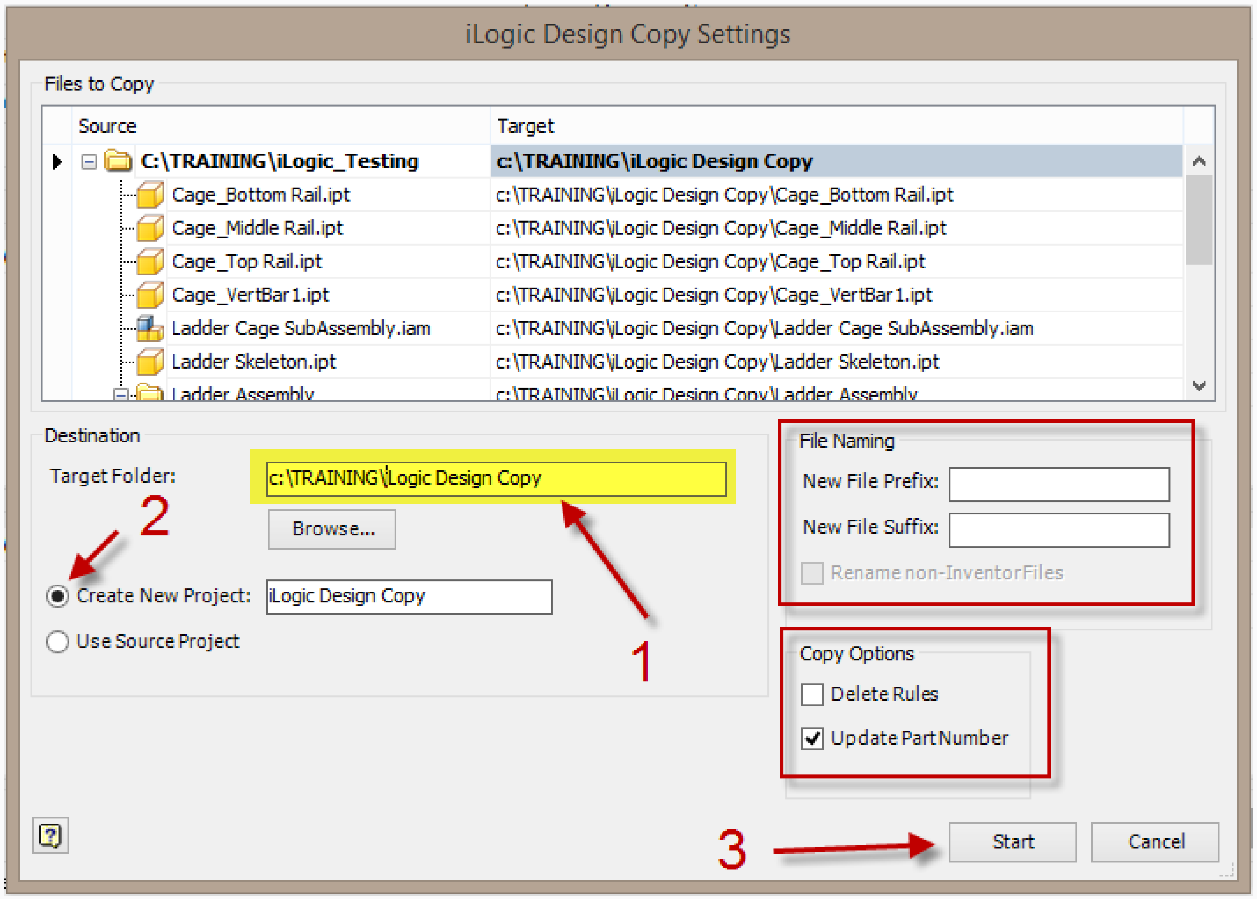
Make sure no files are open in Inventor. The command will not exist in the TOOLS tab if you have any file open.

Once the Dialog Box is open, you can select the assembly you wish to copy. All of the children of the assembly will be automatically selected.



11 files were referenced, and have been automatically selected in the other lists

If you need this assembly design copied to a new project, you can enter that information in this dialog box. It will automatically create the project name based on the location you store it in. You can BROWSE or type in a path. You'll notice that you can change the filenames of the parts with either a prefix or suffix. Keep in mind that you will need to update any code that specifically calls out a filename such as *Parameter("Ladder Skeleton:1", "Ladder_Height") = Ladder_Height .* You can also force the update of the part number.



Source	Target
C:\TRAINING\iLogic_Testing	c:\TRAINING\iLogic Design Copy
Cage_Bottom Rail.ipt	c:\TRAINING\iLogic Design Copy\Cage_Bottom Rail.ipt
Cage_Middle Rail.ipt	c:\TRAINING\iLogic Design Copy\Cage_Middle Rail.ipt
Cage_Top Rail.ipt	c:\TRAINING\iLogic Design Copy\Cage_Top Rail.ipt
Cage_VertBar1.ipt	c:\TRAINING\iLogic Design Copy\Cage_VertBar1.ipt
Ladder Cage SubAssembly.iam	c:\TRAINING\iLogic Design Copy\Ladder Cage SubAssembly.iam
Ladder Skeleton.ipt	c:\TRAINING\iLogic Design Copy\Ladder Skeleton.ipt
Ladder Assembly	c:\TRAINING\iLogic Design Copy\Ladder Assembly

Target Folder: c:\TRAINING\iLogic Design Copy

[✓ Create New Project:] iLogic Design Copy

Use Source Project

New File Prefix:

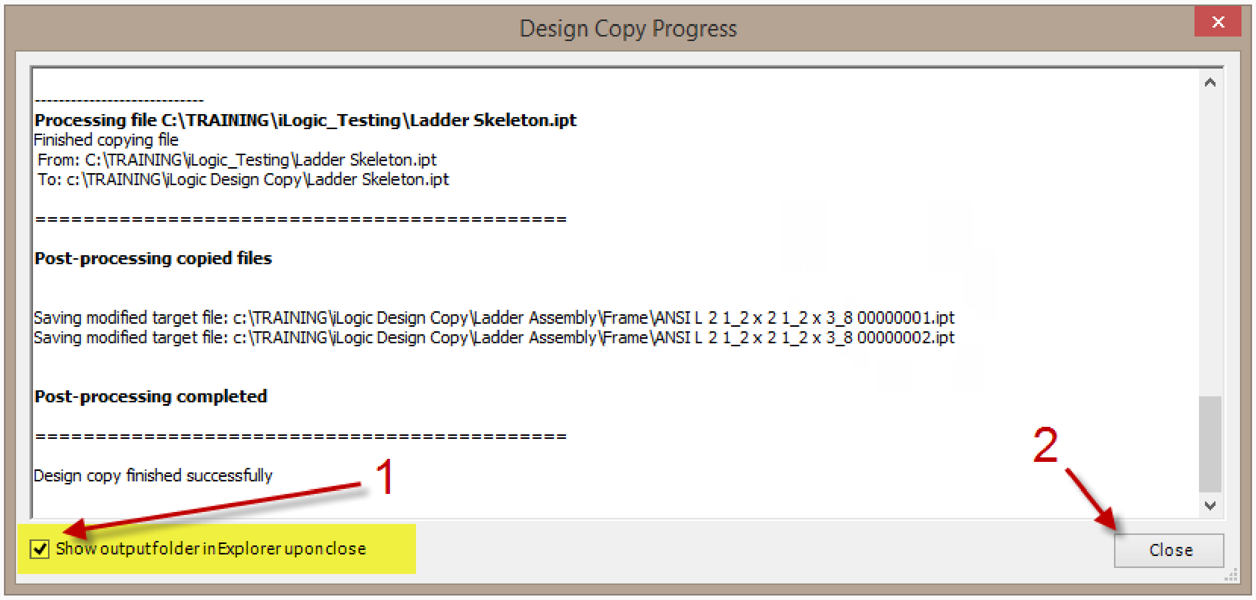
New File Suffix:

☐ Rename non-Inventor Files

☐ Delete Rules

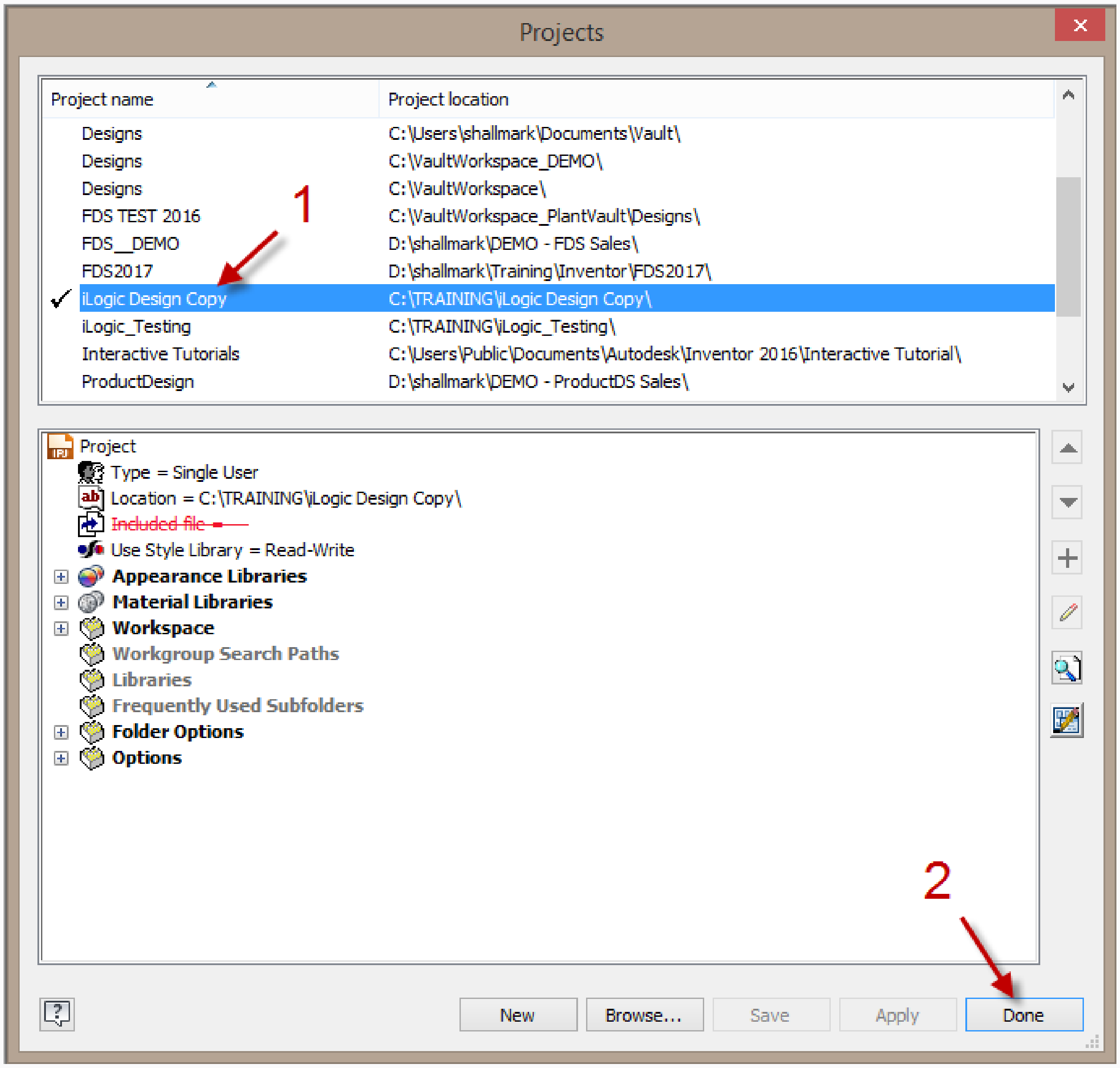
☑ Update Part Number

Once you START the design copy, you will see a progress window. You can uncheck the dialog that allows you to open Windows Explorer to the new location.



```
---------------------------
Processing file C:\TRAINING\iLogic_Testing\Ladder Skeleton.ipt
Finished copying file
 From: C:\TRAINING\iLogic_Testing\Ladder Skeleton.ipt
 To: c:\TRAINING\iLogic Design Copy\Ladder Skeleton.ipt

============================================

Post-processing copied files


Saving modified target file: c:\TRAINING\iLogic Design Copy\Ladder Assembly\Frame\ANSI L 2 1_2 x 2 1_2 x 3_8 00000001.ipt
Saving modified target file: c:\TRAINING\iLogic Design Copy\Ladder Assembly\Frame\ANSI L 2 1_2 x 2 1_2 x 3_8 00000002.ipt


Post-processing completed

============================================

Design copy finished successfully
```

The command is complete. Now you can go open the new project that was created and open the new iLogic Design Copied Assembly.

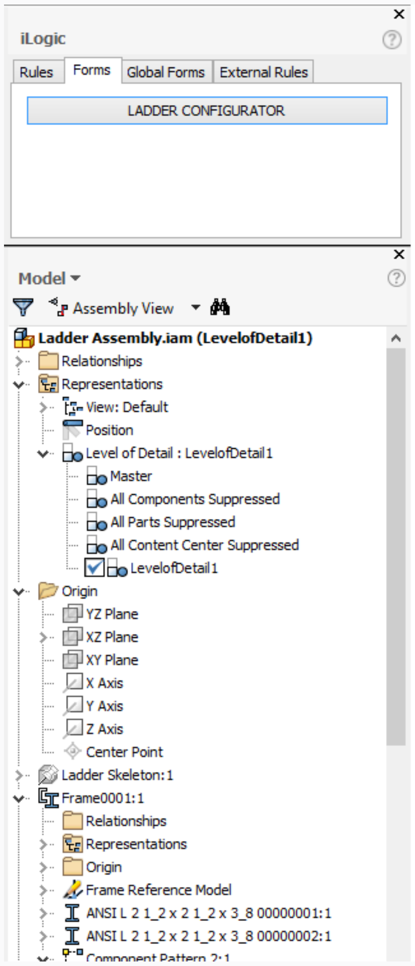


Projects

Project name	Project location
Designs	C:\Users\shallmark\Documents\Vault\
Designs	C:\VaultWorkspace_DEMO\
Designs	C:\VaultWorkspace\
FDS TEST 2016	C:\VaultWorkspace_PlantVault\Designs\
FDS__DEMO	D:\shallmark\DEMO - FDS Sales\
FDS2017	D:\shallmark\Training\Inventor\FDS2017\
✓ iLogic Design Copy	C:\TRAINING\iLogic Design Copy\
iLogic_Testing	C:\TRAINING\iLogic_Testing\
Interactive Tutorials	C:\Users\Public\Documents\Autodesk\Inventor 2016\Interactive Tutorial\
ProductDesign	D:\shallmark\DEMO - ProductDS Sales\

Project
- Type = Single User
- Location = C:\TRAINING\iLogic Design Copy\
- ~~Included file =~~
- Use Style Library = Read-Write
- Appearance Libraries
- Material Libraries
- Workspace
- Workgroup Search Paths
- Libraries
- Frequently Used Subfolders
- Folder Options
- Options

You may have to set/create a LevelOfDetail to see changes made if you are using any Frame Generator parts in the assembly. Hopefully you have created a nice iLogic Form that lets you modify the parameters that need to change.



iLogic

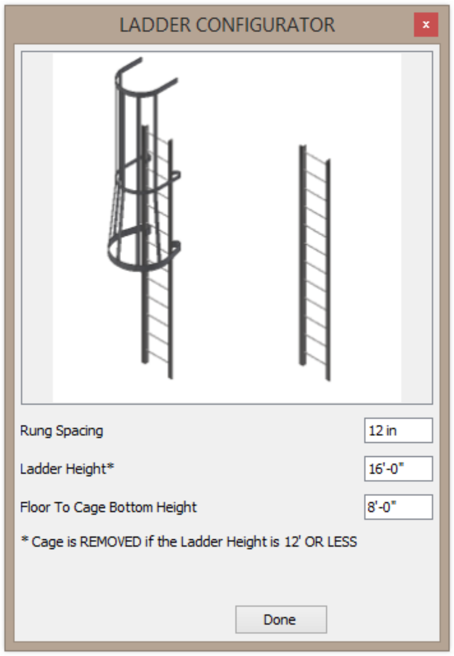
Rules | Forms | Global Forms | External Rules

LADDER CONFIGURATOR

Model

Assembly View

- Ladder Assembly.iam (LevelofDetail1)
  - Relationships
  - Representations
    - View: Default
    - Position
    - Level of Detail : LevelofDetail1
      - Master
      - All Components Suppressed
      - All Parts Suppressed
      - All Content Center Suppressed
      - LevelofDetail1
  - Origin
    - YZ Plane
    - XZ Plane
    - XY Plane
    - X Axis
    - Y Axis
    - Z Axis
    - Center Point
  - Ladder Skeleton:1
  - Frame0001:1
    - Relationships
    - Representations
    - Origin
    - Frame Reference Model
    - ANSI L 2 1_2 x 2 1_2 x 3_8 00000001:1
    - ANSI L 2 1_2 x 2 1_2 x 3_8 00000002:1
    - Component Pattern 2:1

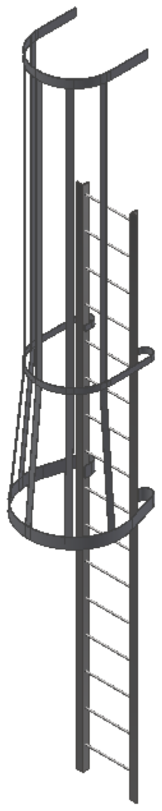


LADDER CONFIGURATOR

Rung Spacing	12 in
Ladder Height*	16'-0"
Floor To Cage Bottom Height	8'-0"

* Cage is REMOVED if the Ladder Height is 12' OR LESS

Done



SUMMARY

The Autodesk Inventor tool iLogic adds a higher level of intelligence to your designs. Its rules-driven process allows you to capture and reuse your design work on virtual products. You’ll save time and increase productivity by standardizing and automating your design processes, especially repetitive tasks.

The iLogic rules determine and drive parameter and attribute values for your designs, so you can define the behavior of a model’s attributes, features and components. Knowledge is saved and stored directly in your documents, and it’s parametric. Rules can be embedded as objects directly into part, assembly and drawing documents and make use of custom parameter types in Inventor. The result is the ability to use input values that are more than just numerical.

ABOUT THE AUTHORS

Jason Miles

With a background in industrial drafting, design, engineering, computer information systems, and marketing, Jason Miles serves as a senior specialist to Applied Software customers in the manufacturing industry.

Jason is an Autodesk Certified Instructor and Autodesk Inventor Certified Professional, with an additional specialized certification in Product Design and Manufacturing.

Carol Dunn

Carol is a professional writer. She joined the Applied Software marketing team in 2018 after 26 years with the top-ranked Autodesk Authorized Training Center in North America – what is now the Denver-metro-based Applied Software office. She began working in the CAD industry in 1992, when AutoCAD was one of only a few products Autodesk marketed. She has held most of the positions a small business offers, including GM, sales, accounting, tech support, customer service, and sometimes the janitor.

Applied Software®

Learn from the Best: iLogic

Written by Carol Dunn & Jason Miles

ASTI.com