

PropertyWizard Help

Release Notes V1-9-0

PropertyWizard is an add-in for Autodesk® Revit® that allows you to control project parameters with formulas, in the same way as you can control family parameters with formulas.

For example, with PropertyWizard you can:

- Set the Sheet Number of all the sheets from their other parameters.
- Set the Door Numbers of all the doors, using data from the door's Level and Type.
- Set visible properties for X and Y location on all pile foundations so you can schedule them.

And best of all, because PropertyWizard is linked into Revit's parametric change engine, the parameter values are kept up-to-date all the time. No need to repeatedly run Dynamo scripts to update the parameters. #ReviseInstantly

Each PropertyWizard formula has 3 parts:

1. The Category of element that you want to apply it to.
2. The Target Property you want to set.
3. The Formula you want to use.

The Category can be any Revit Category. The formula will apply to all elements in the Category (you can use filtering in the formula to affect just a subset). The Target Property can be an Instance Property or a Type Property.

The Formula language is an extension of the normal Revit family formula language, so if you can write Revit family formulas, you can write PropertyWizard formulas. The extensions include string manipulation and concatenation (which you don't have in family formulas), access to hidden properties from the Revit API, properties from related elements, and Project Information properties.

PropertyWizard gives you control of your Revit data, so you can place it just where you need it to be, and generate your data and drawing outputs efficiently, without having to run Dynamo scripts or manually copy data.

Support

There is more help on the website at www.davidwooddesign.com, including examples and walkthroughs.

I want you to get the most out of PropertyWizard, so please, if you need any help or advice in using it, get in touch with me at support@davidwooddesign.com.

Autodesk and Revit are registered trademarks or trademarks of Autodesk, Inc., and/or its subsidiaries and/or affiliates in the USA and/or other countries.

Contents

| | |
|--------------------------------------------------|----|
| PropertyWizard Help | 1 |
| Support | 1 |
| Installation | 3 |
| Getting Started..... | 4 |
| A. Your first formula: Hello Wall | 4 |
| B. Using properties in formulas | 7 |
| C. Joining text together | 9 |
| D. Calculated values | 10 |
| E. More properties | 11 |
| F. Type properties | 12 |
| G. Custom Properties | 13 |
| H. Project Information properties..... | 14 |
| I. Choosing between results | 15 |
| J. Filtering..... | 15 |
| K. Setting Yes/No parameters | 16 |
| L. Using Element Id..... | 17 |
| M. Setting Type Properties (New in V1.4.0) | 18 |
| N. Global Parameters (New in V1.4.0) | 18 |
| Dealing with errors | 20 |
| Licensing errors..... | 20 |
| Syntax errors | 21 |
| Formula errors | 21 |
| Infinite loop error..... | 22 |
| Property name not recognised | 22 |
| Parameter does not have value | 23 |
| Duplicate Properties | 24 |
| Type mismatch errors (New in V1.4.0) | 24 |
| Reference | 25 |
| Running PropertyWizard..... | 25 |
| Activating PropertyWizard in a project..... | 25 |
| Creating a formula | 26 |
| Target Properties | 26 |
| Formulas | 26 |
| Source Properties..... | 27 |
| Operators and Functions | 28 |

| | |
|----------------------------------|----|
| Import/Export | 30 |
| Release Notes (What's New) | 31 |

Installation

Uninstall the Beta version

If you have a Beta version of PropertyWizard older than V1-0-0, you have to uninstall it to enable the new version to work. Download the uninstaller from:

<http://www.davidwooddesign.co.uk/uninstall-propertywizard-beta/>

If you'd prefer to manually uninstall, delete the *PropertyWizard.addin* files and the *DWD* folders from each of the version folders 2016 to 2019 in C:\ProgramData\Autodesk\Revit\Addins.

Autodesk App Store downloads

If you downloaded PropertyWizard from the Autodesk App Store, follow the instructions that came with the package.

DavidWoodDesign downloads

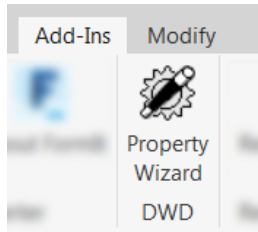
If you downloaded PropertyWizard from www.davidwooddesign.com:

1. Close any Revit sessions before you start.
2. If you received a zip file, unzip it to reveal the DavidWoodDesignPropertyWizard.msi file.
This is a copy of the App Store installer.
3. Run the msi file.

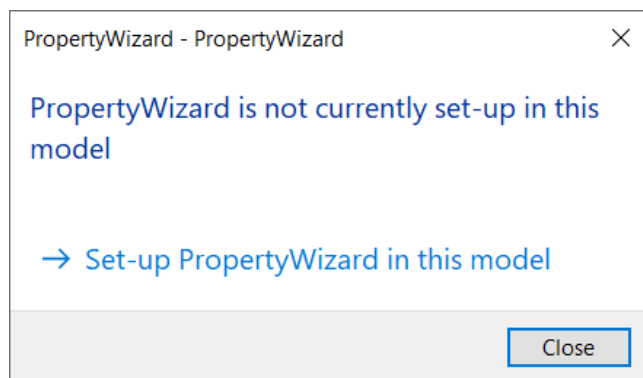
Getting Started

A. Your first formula: Hello Wall

- A.1 Open a new Revit model.
- A.2 Draw some walls.
- A.3 Click the PropertyWizard button. It's on the Add-Ins tab:

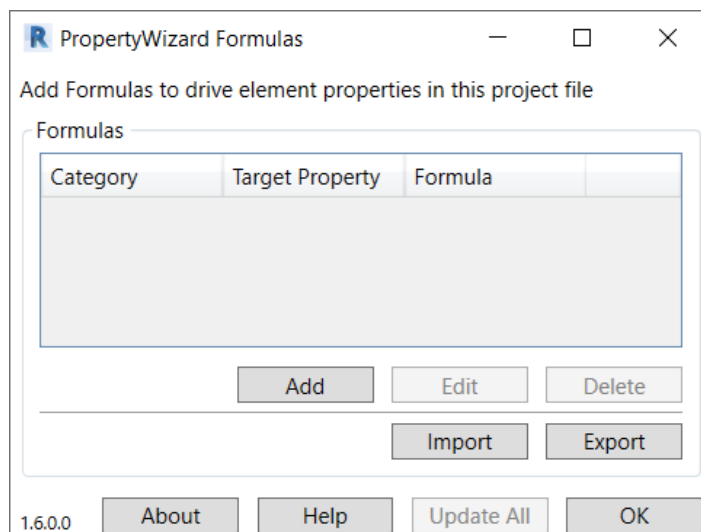


This will show the Set-up dialog:



- A.4 Select *Set-up PropertyWizard in this model*.

PropertyWizard will get set up and then open the main PropertyWizard window:



- A.5 Click *Add* to start adding a new Formula. This will open the Formula window.

PropertyWizard Formula

Category

Target Property

Formula

1.6.0.0

Cancel OK

A.6 Select *Walls* in the Category drop-down. You can type a *W* to get to the value quickly.

A.7 Type *Comments* in the Target Property box.

A.8 Type "Hello Wall" (including the quote marks) in the Formula box.

Your formula should look like this:

Category: Walls

Target Property: Comments

Formula: "Hello Wall"

PropertyWizard Formula

Category

Walls

Target Property

Comments

Formula

"Hello Wall"

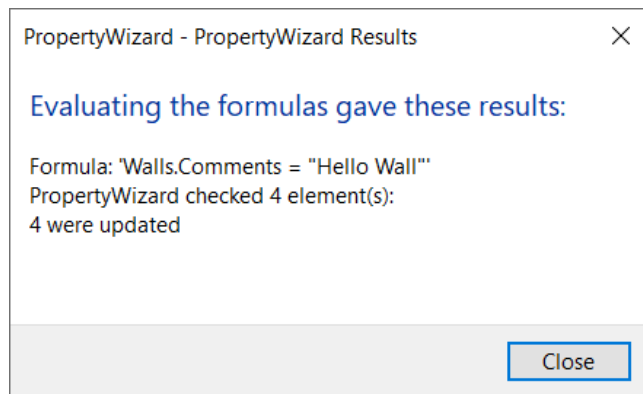
1.6.0.0

Cancel OK

A.9 Click *OK* in the Formula window and then click *Update All* in the main PropertyWizard window.

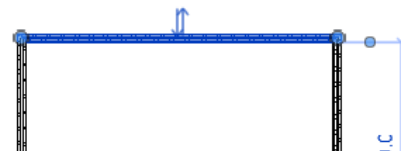
This will update your walls' Comments property using your formula.

PropertyWizard will show a results dialog like this:



The first line of the result shows the formula. The second line shows how many elements (walls) were checked, and the remaining lines will show how many elements were up-to-date, updated, or had errors.

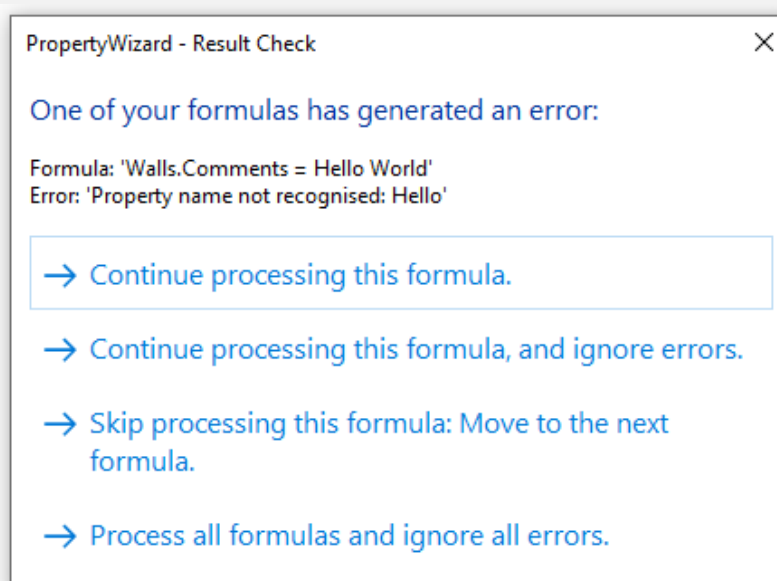
- A.10 Close the results dialog, click *OK* to close the main PropertyWizard Window, and check the properties of your walls:



Every wall in your project now has its Comments set to *Hello Wall*.

Fixing errors

If PropertyWizard finds an error while processing your formula, it will show a dialog like this:



continued..

You can choose to continue processing this formula for the remaining elements, or to stop processing this formula and skip the remaining elements.

If you get an error, check your typing and try again. Be sure to target *Comments* (not *comments* or *Comment*) and include straight (not curly) quotes around "Hello Wall". In this example, I've forgotten to put quote marks around the *Hello Wall*.

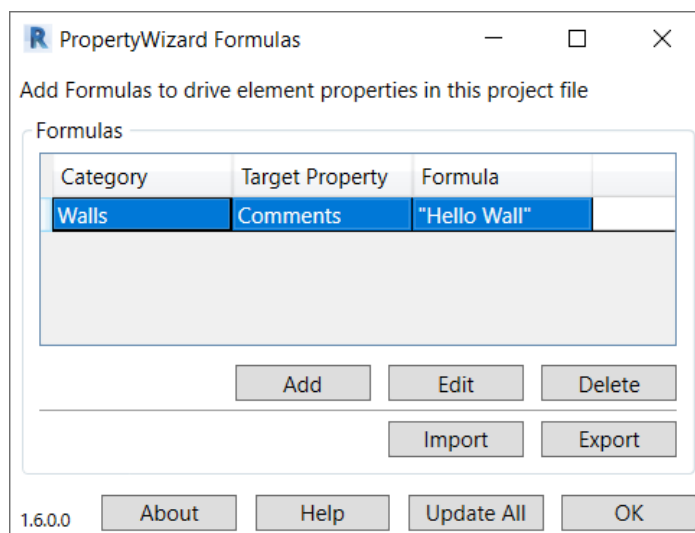
There are more examples of common errors below in 'Dealing with Errors'.

B. Using properties in formulas

Your first formula used the static text "*Hello World*" for every element.

You can *element properties* instead of static text, so that each element gets the relevant property value:

- B.1 Open the PropertyWizard main window by clicking the PropertyWizard button (on the Add-Ins tab).
- B.2 Click your formula to select it:



- B.3 Click the Edit button to open the Formula window.
- B.4 Edit the formula to use the wall's Length property:

Category: Walls

Target Property: Comments

Formula: Length

PropertyWizard Formula

Category
Walls

Target Property
Comments

Formula
Length

1.6.0.0

Cancel OK

This formula is going to get each wall's Length, format it using the project units, and put the resulting text into the wall's Comments property.

- B.5 Click OK in the Formula window to close it, and then OK again in the main window.

PropertyWizard will show a results dialog like this:

PropertyWizard - PropertyWizard Results

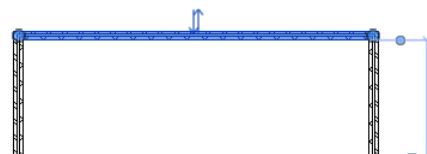
Evaluating the formulas gave these results:

Formula: 'Walls.Comments = Length'
PropertyWizard checked 4 element(s):
4 were updated

Close

- B.6 Close the results dialog and check the properties of your walls:

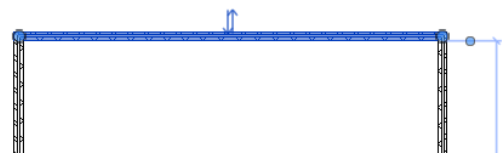
| Dimensions | |
|---------------|-----------------------|
| Length | 11200.0 |
| Area | 91.920 m ² |
| Volume | 26.657 m ³ |
| Identity Data | |
| Image | |
| Comments | 11200 |



All the walls are reporting their Length, using the project's current units setting (mine is set to millimetres for the UK).

- B.7 Change the length of some of your walls, and check the properties.

| Dimensions | |
|---------------|------------------------|
| Length | 13190.0 |
| Area | 107.840 m ² |
| Volume | 31.274 m ³ |
| Identity Data | |
| Image | |
| Comments | 13190 |



As you add and edit your walls, PropertyWizard updates their Comments. This is the power of PropertyWizard: once you have set up your formulas, it will quietly and efficiently keep your properties up-to-date.

C. Joining text together

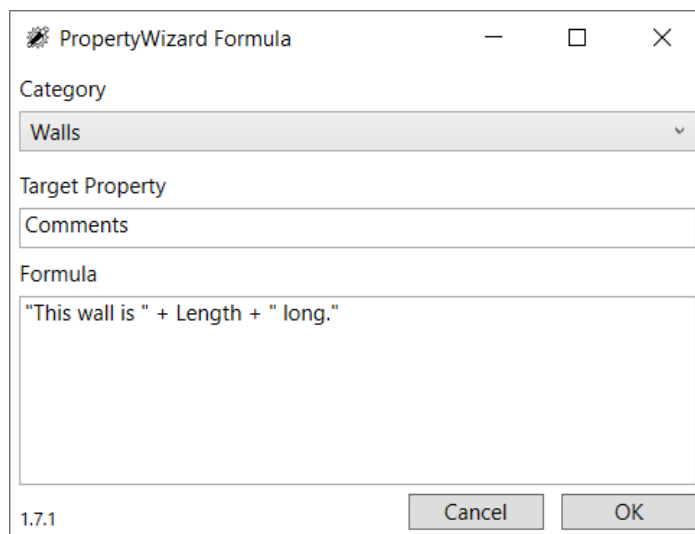
The wall length on its own is not very descriptive. But there are lots of ways you can calculate and combine property values in PropertyWizard. In this case, let's concatenate the Length with some text values using the + operator:

C.1 Open the PropertyWizard main window and Edit your Comments formula:

Category: Walls

Target Property: Comments

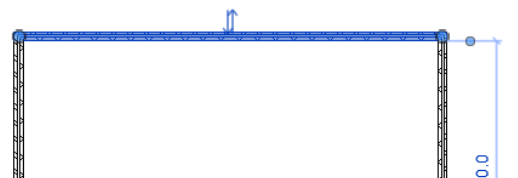
Formula: "This wall is " + Length + " long"



This formula uses two + operators to sandwich the Length value between the pieces of static text. I've included spaces within the quotes, either side of the Length, so that I get spaces next to the length in the output.

C.2 Click OK in the Formula window to close it, and then OK again in the main window. Then check the Comments on your walls:

| Walls (1) | | Edit Type |
|---------------|--------------------------|-----------|
| Dimensions | | |
| Length | 13190.0 | |
| Area | 107.840 m ² | |
| Volume | 31.274 m ³ | |
| Identity Data | | |
| Image | | |
| Comments | This wall is 13190 long. | |
| Mark | | |



You can use the + operator to concatenate any number of text values, like this BS 1192 sheet numbering example: <http://www.davidwooddesign.co.uk/2018/08/06/how-to-use-propertywizard-for-sheet-numbers/>. If your formulas include non-text values (number values, lengths, and areas) PropertyWizard will convert them into text before concatenating them.

The + operator is the simplest of the text (string) operators: PropertyWizard has several useful string functions as well, detailed in the reference section.

D. Calculated values

Your formulas can include mathematical calculations as well as text manipulations. You can calculate with number values, lengths, and areas, and then assign the result to a property with a matching value type.

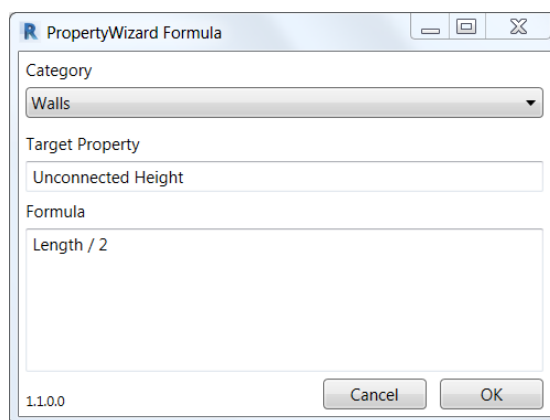
For example, let's control the Unconnected Height of our walls:

D.1 Open the PropertyWizard main window and Add a new formula:

Category: Walls

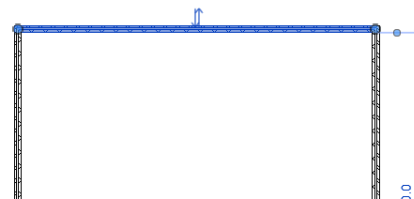
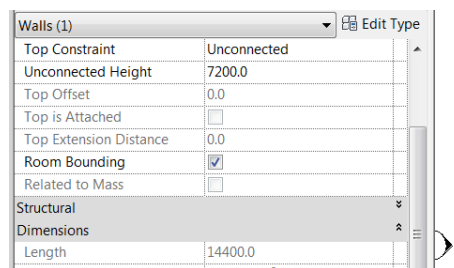
Target Property: Unconnected Height

Formula: Length / 2



This formula sets the wall's Unconnected Height equal to half the Length of the wall.

D.2 Click OK in the Formula window to close it, and then OK again in the main window. Then check the Unconnected Height of your walls:



Unless the tops of your walls are constrained (by being connected to a Level or attached to a Floor), you should see that they alter to match the Unconnected Height. You're using a PropertyWizard formula to control the geometry of your model!

D.3 Change the lengths of your walls and watch how their heights change automatically.

This blog post describes a similar formula, keeping the wall areas fixed by controlling the Unconnected Height: <http://www.davidwooddesign.co.uk/2018/08/22/walls-with-fixed-area/>

Of course, not all properties can be written to: some are 'read-only' like Reporting Parameters. And, just as with family formulas, the value your formula produces has to match the kind of

value the target property expects: otherwise you will get the equivalent of an 'Inconsistent Units' error.

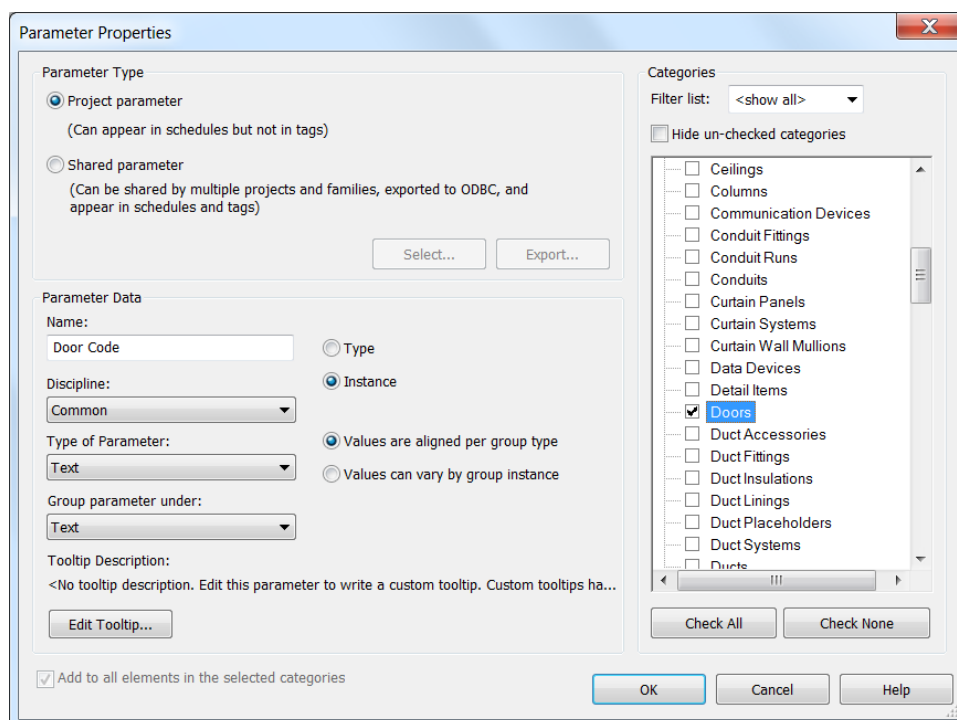
E. More properties

In the last few examples you have used the wall's Length, which is an Instance property. The next few examples show you how to access other kinds of property, including Type properties, API properties and Project Information properties.

As a vehicle for this set of examples, we will create a Door Code property that brings together some location data for doors:

E.1 Add some doors to your project.

E.2 Create a new project parameter in the project. It should be a text parameter, and it should apply to Doors. Name the new parameter *Door Code*.



E.3 Open the PropertyWizard main window and Add a new formula:

Category: Doors

Target Property: Door Code

Formula: Mark

E.4 Update All to run the formula on each of your doors.

This formula simply copies each door's Mark value into your new property. Over the next few examples, we will add more fields to the Door Code.

F. Type properties

Most of the properties you have been using have been Instance properties. But you can also access Type properties:

F.1 Edit the formula to access the door Type Mark:

Category: Doors

Target Property: Door Code

Formula: Type.[Type Mark] + "-" + Mark

To access this Type property you just put the word 'Type', then a full-stop (period), and then the name of the property you want to access.

This is an example of a *Nested Property*, where you can access the *property of a property*. This works because each door has a built-in property named Type that points to the door's Type. And once you have the door's Type, you can access any of that Type's properties.

All nested properties can be accessed using this *dot notation*: Start with the name of the property on the element, then put a dot (full-stop/period), then the name of the nested property. You can access any depth of nested property using this dot notation.

In this case, as an extra complication, the property name *Type Mark* contains a space, so you need to wrap it in square brackets. If the name didn't contain a space, you would just type it straight after the dot.

New in version V1.4.0, you can set Type properties using PropertyWizard – see Getting Started section M.

G. Custom Properties

You can also access custom properties (project parameters or shared parameters) using PropertyWizard:

G.1 Create a new, text, project parameter on Levels. Name the new parameter *Level Code*:

This will allow you to record a short code for each Level, and use it in the Door Code.

G.2 Set the value of the new Level Code for each of the Levels in your project.

In my case, I have Levels 0 and 1, so I have used Level Codes L0 and L1:

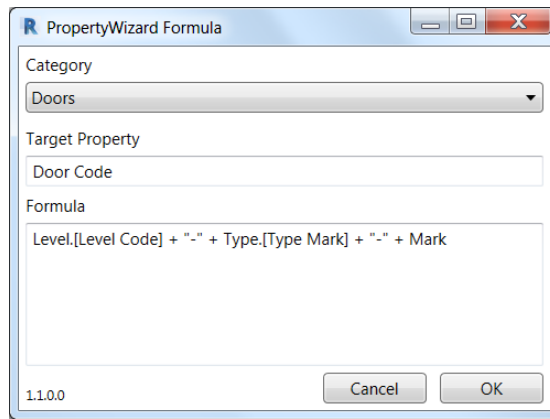
| <Level Schedule> | | |
|------------------|-----------|------------|
| A | B | C |
| Name | Elevation | Level Code |
| Level 0 | 0 | L0 |
| Level 1 | 4000 | L1 |

G.3 Edit your formula to access the Level Code property:

Category: Doors

Target Property: Door Code

Formula: Level.[Level Code] + "-" + Type.[Type Mark] + "-" + Mark



Again, you use dot notation to access the property. And again, because the property name contains a space, you have to wrap it in square brackets.

As you place doors on different levels, their Door Codes will fill-in with the correct Level Codes.

H. Project Information properties

You can use a shortcut to access Project Information properties. This can be useful if you need to code elements in your model according to the zone, block or volume that the model file represents:

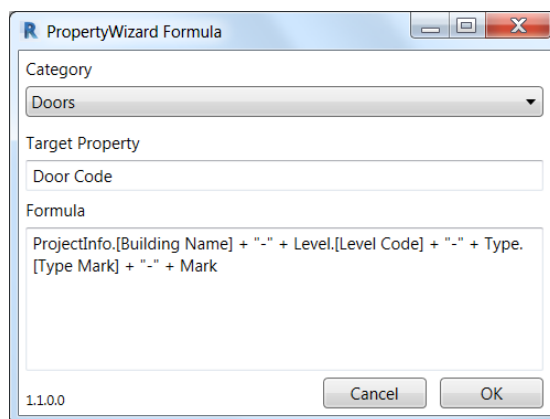
H.1 Set the value of the Building Name parameter in Project Information. I used the value *B1*.

H.2 Edit your formula to access the Building Name parameter from Project Information:

Category: Doors

Target Property: Door Code

Formula: `ProjectInfo.[Building Name] + "-" + Level.[Level Code] + "-" + Type.[Type Mark] + "-" + Mark`



This uses the shortcut *ProjectInfo* to access the project's Project Information, followed by a dot and then the property name in the usual way.

That completes the Door Code: Now, it identifies the Building, Level and Door Type that the door belongs to, as well as reporting the door's unique Mark.

I. Choosing between results

Moving on to look at some other ways of using formulas, you can use an `if()` function to make a formula that chooses between two results:

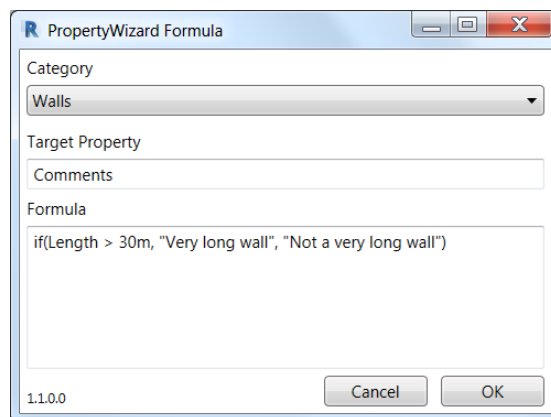
- I.1 Draw a new wall about 20 metres (say 60 feet) long.
- I.2 Open the PropertyWizard main window and Edit your Comments formula.

It's important that you Edit the existing formula (or Delete it and Add a new formula). Otherwise you will have two formulas that are trying to set the same Target Property, which can lead to the infinite loop error (for more details, see the *Dealing with errors* section, below).

Category: Walls

Target Property: Comments

Formula: `if(Length > 30m, "Very long wall", "Not a very long wall")`



If you prefer, you can use 100ft rather than 30m. PropertyWizard will work with either.

Your new wall's Comments will now read *Not a very long wall*.

- I.3 Drag the end of the wall to make it more than 30m long.

The wall's Comments will change to *Very long wall*.

The `if()` function has three inputs: The test, and two results. In this case the test is *Length > 30m*, which is true if the wall's Length is greater than 30m and false otherwise. The `if()` returns the first result if the test is true, and the second result if the test is false.

J. Filtering

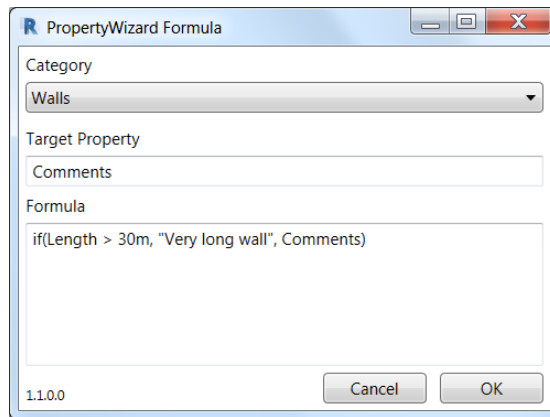
What if you want to set the target property if the test is true, and leave it alone otherwise?

- J.1 Edit your Comments formula to use *Comments* as the second result of the `if()`:

Category: Walls

Target Property: Comments

Formula: `if(Length > 30m, "Very long wall", Comments)`



This formula will only set the Comments property if the wall is longer than 30m. Otherwise, it will leave the Comments property unchanged.

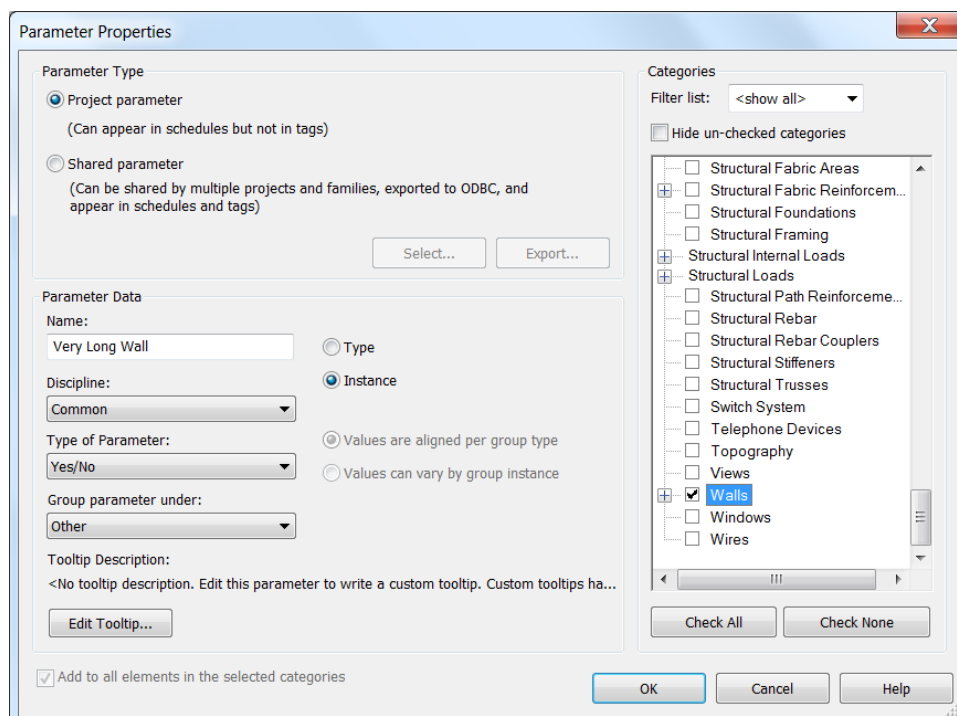
- J.2 Change the Comments on a wall shorter than 30m: Your change will 'stick', because the formula is no longer controlling the Comments value.
- J.3 Change the Comments on a wall longer than 30m: The value will be reset to *Very Long Wall* automatically by the formula.

This filtering formula works because before PropertyWizard sets the Target Property, it checks whether the new value is different from the existing. If the two are the same, it does nothing. One way of ensuring that the value of your formula is equal to the Target Property (in your chosen conditions) is by including the Target Property name in an if() function as you did here.

K. Setting Yes/No parameters

You can set Yes/No parameters using a logical (true/false) value from PropertyWizard:

- K.1 Create a new Project Parameter in the project. It should be a Yes/No parameter, and it should apply to Walls. Name the new parameter *Very Long Wall*.

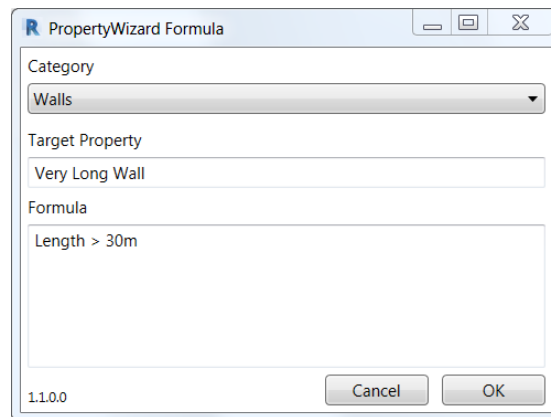


K.2 Create a new formula to set your new parameter:

Category: Walls

Target Property: Very Long Wall

Formula: Length > 30m



This will set your new property to Yes (true) on the very long walls, and No (false) for the shorter ones. And as you alter the lengths of the walls, the property will update automatically.

L. Using Element Id

You can easily access the Element Id property, which can be useful for Warnings workflows:

L.1 Create a new Project Parameter in the project. It should be a Text parameter, and it should apply to Walls. Name the new parameter *ElementId*

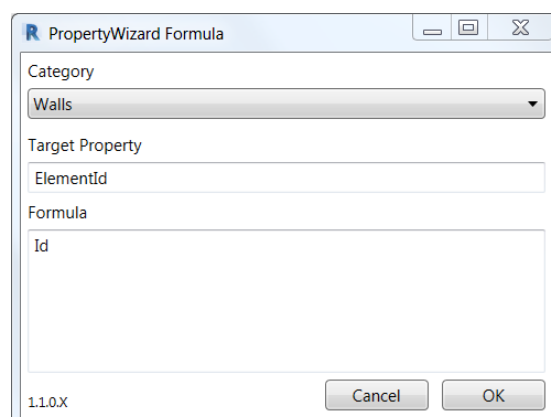
It is important not to name your new project parameter as *Id*, to avoid the problem described under Duplicate Properties in the *Dealing with errors* section below.

L.2 Create a new formula to set your new parameter:

Category: Walls

Target Property: ElementId

Formula: Id



This will set your new ElementId property on all the walls in your project.

M. Setting Type Properties (New in V1.4.0)

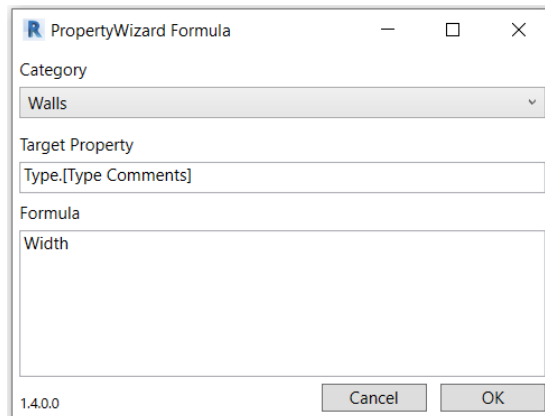
You can now set Type properties using PropertyWizard. Just write the name of the type property in the Target Property box, prefixed with 'Type.' (including the dot).

M.1 Create a formula to copy the Width of each wall type into the Type Comments:

Category: Walls

Target Property: Type.[Type Comments]

Formula: Width



The PropertyWizard Formula dialog box is shown. It has a title bar with a blue 'R' icon and the text 'PropertyWizard Formula'. Below the title bar is a 'Category' dropdown menu set to 'Walls'. Below that is a 'Target Property' text box containing 'Type.[Type Comments]'. Below that is a 'Formula' text box containing 'Width'. At the bottom left is the version number '1.4.0.0'. At the bottom right are 'Cancel' and 'OK' buttons.

All the wall types in the project will have their Type Comments set to match their Width. If you change the wall build-up, the Type Comments will change to match.

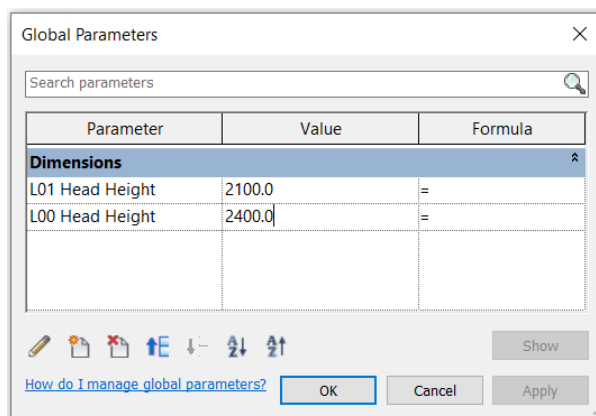
Note: The square brackets around 'Type Comments' are not essential – You don't need to use square brackets in the Target Property box, even if the property name includes a space.

N. Global Parameters (New in V1.4.0)

Also new in version V1.4.0, you can use global parameters in your formulas. Access them by writing 'GlobalParameter.' and then the name of the global parameter.

In this example, we're going to use global parameters to control the head heights of windows on two different levels of the building.

N.1 Create two global parameters, one for each level. These are Length parameters, and I've set them to two different values :

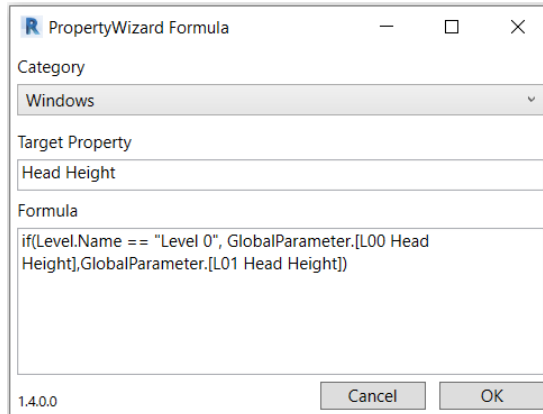


The Global Parameters dialog box is shown. It has a title bar with a close button. Below the title bar is a search bar labeled 'Search parameters'. Below the search bar is a table with three columns: 'Parameter', 'Value', and 'Formula'. The table has a header row and two data rows. The first data row is for 'L01 Head Height' with a value of '2100.0' and a formula of '='. The second data row is for 'L00 Head Height' with a value of '2400.0' and a formula of '='. Below the table is a toolbar with icons for adding, deleting, and editing parameters. At the bottom right are 'Show', 'OK', 'Cancel', and 'Apply' buttons.

| Parameter | Value | Formula |
|-------------------|--------|---------|
| Dimensions | | |
| L01 Head Height | 2100.0 | = |
| L00 Head Height | 2400.0 | = |

N.2 Write a PropertyWizard formula that controls the Head Height of Windows:

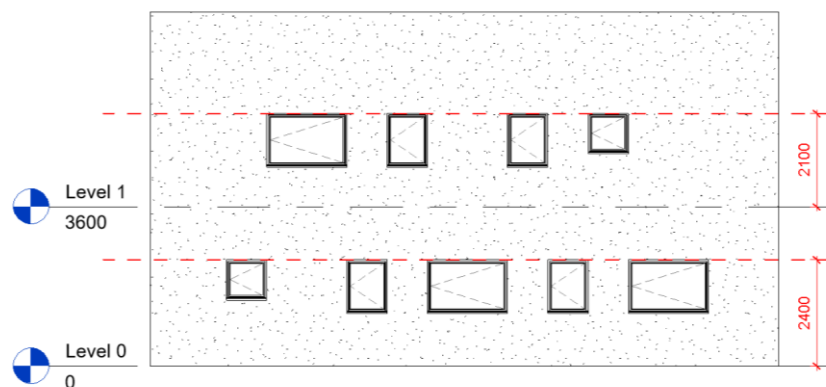
Category: Windows
Target Property: Head Height
Formula: `if(Level.Name == "Level 0", GlobalParameter.[L00 Head Height],GlobalParameter.[L01 Head Height])`



Breaking the formula down, it uses an 'if' function – which works just like the if function in standard Revit formulas and in Excel. It has three inputs – a condition and two possible results. If the condition is true, it returns the first result; and if false, the second.

In this case, the condition tests whether the Name of the Window's Level is equal to "Level 0", and the results are the values of the two global parameters.

If you now change the values of the global parameters, the heights of the windows on the two floors will change to match.



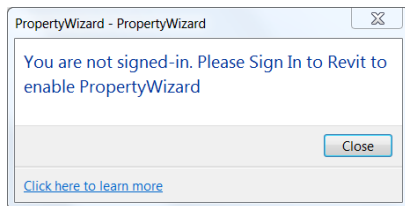
Dealing with errors

Licensing errors

PropertyWizard licensing should 'just work'. But occasionally you may encounter problems. This page lists the most likely problems and solutions, but if your problem is not listed here, please get in touch: support@davidwooddesign.com

Not Signed In

Your PropertyWizard license is linked to your Autodesk Account. To use PropertyWizard, you have to be signed in to your Autodesk Account in Revit. If you are not signed in, PropertyWizard will show this dialog:

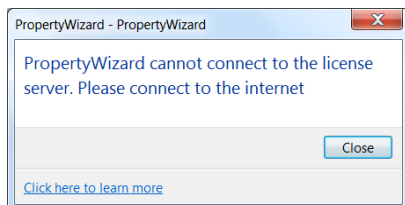


Fix this problem by signing in to Revit using the button at the top-right of the Revit window:



No Network

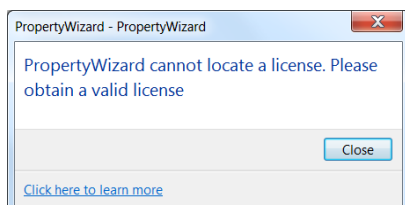
Your PropertyWizard license is stored on Autodesk's cloud server. If PropertyWizard cannot contact the server (for example, if you are not connected to the internet), PropertyWizard will not be able to access your license and will show this dialog:



Fix this problem by connecting to the internet.

No License

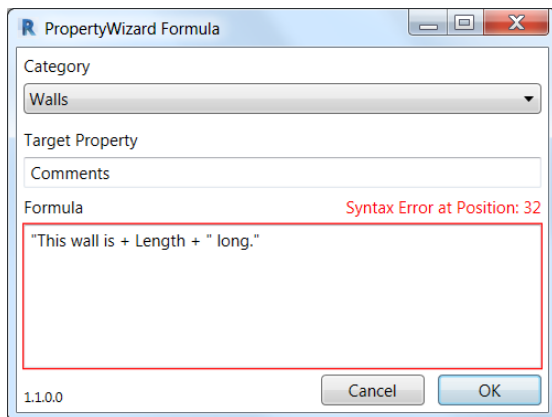
You need a license to use PropertyWizard. If the license server does not contain a license matching your Autodesk Account, PropertyWizard will show this dialog:



Fix this problem by assigning a valid PropertyWizard license to your Autodesk Account.

Syntax errors

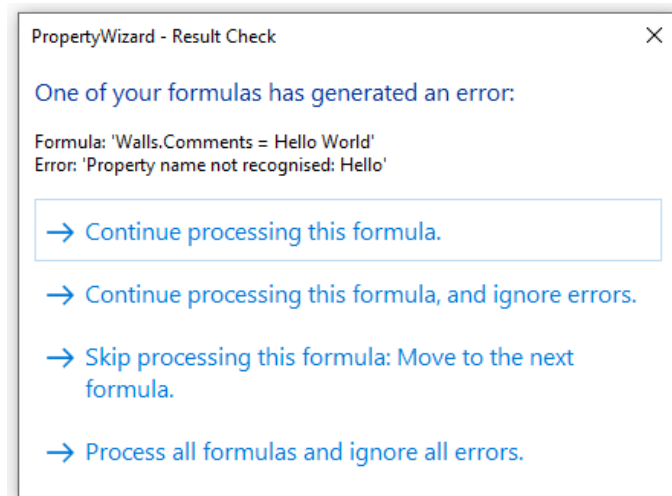
When you type in a formula, PropertyWizard checks that the syntax is correct, i.e. that the formula makes sense in the formula language. If your formula has a syntax error, PropertyWizard will highlight the location of the error like this:



In this case the syntax check found an error after the 32nd character in the formula. Sometimes it is easy to see what has gone wrong. Other times, you have to check your formula very carefully (and maybe break it down into pieces) to find where the fault lies.

Formula errors

When your formula syntax is correct, PropertyWizard will try to run the formula on all the relevant elements. If one of the elements generates an error, PropertyWizard will show a dialog box like this:



The dialog gives you four options:

If you choose the first option, *Continue processing this formula*, PropertyWizard will stop processing this element and move on to the next. If the next element also generates the error, PropertyWizard will show the dialog again. This can get tedious if you have a lot of elements to process.

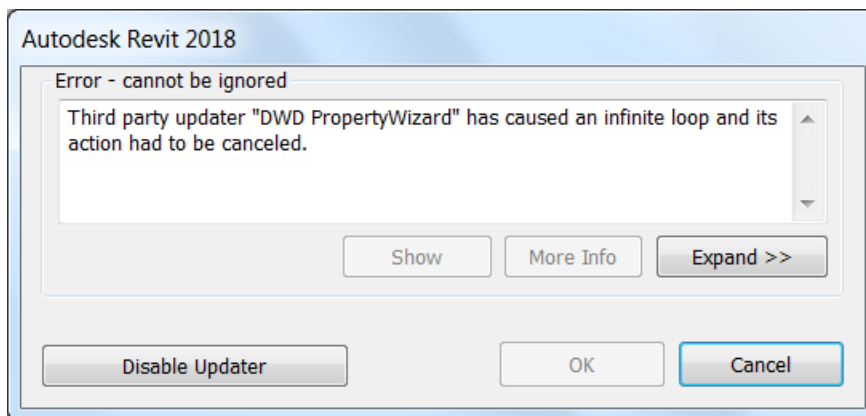
If you choose the second option, *Continue processing this formula, and ignore errors*, PropertyWizard will carry on processing the rest of the elements without reporting any errors. This is useful if the problem only affects some of the elements, and you want PropertyWizard to process the ones it can.

If you choose the third option, *Skip processing this formula...* PropertyWizard will stop processing this formula and move immediately to the next formula (if any). This is useful if you have made a mistake and want to fix your formula before running it any further.

If you choose the fourth option, *Process all formulas and ignore all errors*. PropertyWizard will carry on processing this formula, and process any other formulas, but will not report any more errors for any of the formulas. This is useful when you have a lot of formulas and want to run them all without stopping to look at individual errors. You will still get the final result dialog so you can see which formulas generated errors.

Infinite loop error

If you create two formulas that try to set the same Target Property, it's likely that you will get this Infinite Loop Error:

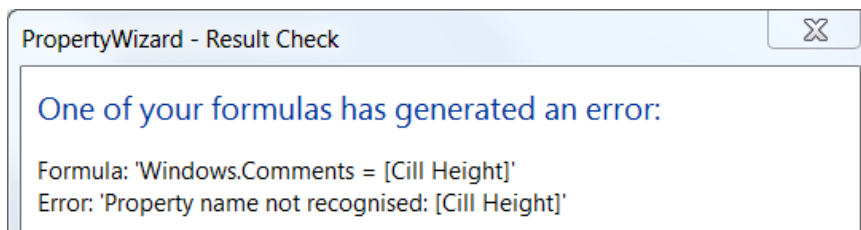


This is not the end of the world. Just click the Cancel button and Edit one of the conflicting formulas to fix the problem.

Property name not recognised

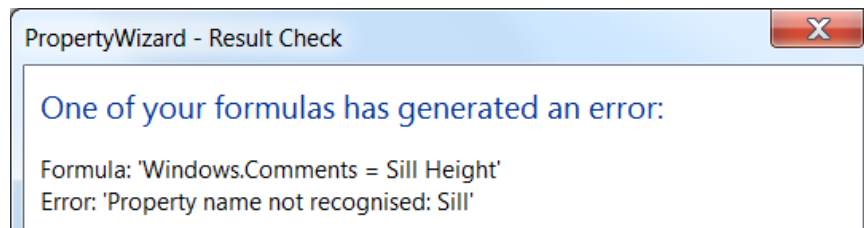
This must be the most common error I get in PropertyWizard. It means that there isn't a property called 'whatever' on the element. And the key to fixing the problem is to look carefully at the name PropertyWizard thinks it's looking for ('whatever'), and ask yourself if that's really the property name you want.

Often, I find that I've mis-spelled the property name – In this case, I've typed 'Cill' instead of 'Sill':



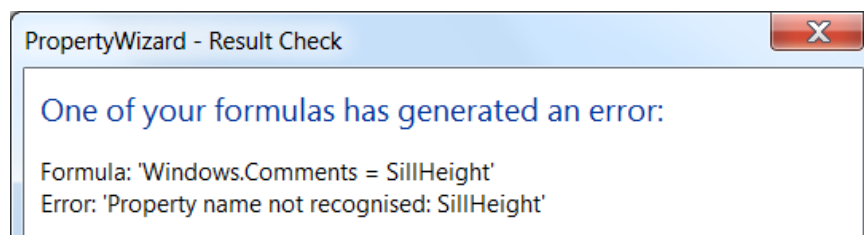
Double-check the name of the property in the Properties box or in RevitLookup.

Other times, I've forgotten that the property name contains a space – property names that contain spaces or special characters have to be enclosed in square brackets:



You can see that PropertyWizard has only found the first part of the name, stopping at the space, and of course there isn't a property called 'Sill' on the element.

Still other times, I've accidentally missed out the space altogether, and typed 'SillHeight'. Again, there's no such property:

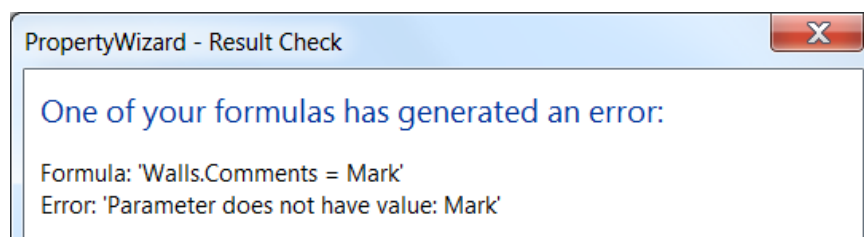


And finally, I sometimes forget that a property is a type property, and so don't prefix the property name with 'Type.'

So, there are plenty of ways you can type a property name wrong in your formulas, and only one way to type it right. But PropertyWizard's error message tells you how it has interpreted your typing, and from that you can usually work out what to fix.

Parameter does not have value

Revit optimises its memory usage by only generating element properties when they are needed. So occasionally you will come across properties that do exist, but don't have a value:



Very often, the property will have a value for some of your elements, but not for all. So you want to run the formula on the valid elements and highlight the elements that don't have values. The try() function lets you do that.

Please note that most formulas do not need this level of control, because most properties do have values. I would only implement the steps in this section if you find properties that don't return values. And be careful, because try() will catch all errors, not just *Parameter does not have value*. So if you use try() indiscriminately, it will catch and hide formula errors that you should really fix.

The try() function has two inputs: The expression to try, and the value to return if the expression returns an error. For example:

```
try(Mark, "NO_MARK")
```

In this case, the try() expression is Mark, and the formula will return the text NO_MARK if the Mark property does not have a value. If you just want to return an empty string instead, use two double-quotes: "": you can use any value you like as the default.

Duplicate Properties

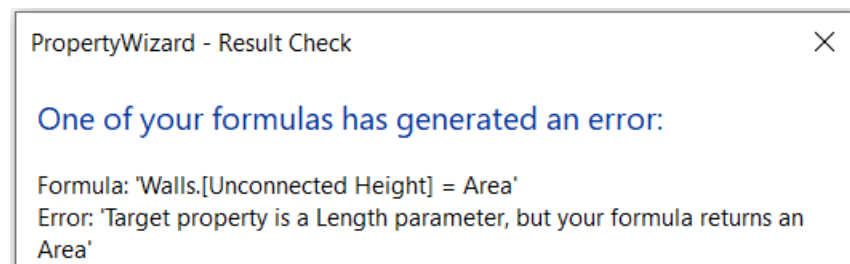
It is possible to have several Revit properties with the same name on the same category of elements. PropertyWizard can only access the first of these duplicate properties on each element, and it always searches for parameter properties first and API properties second.

So in the Element Id example above, if you created a new project parameter called Id, and referenced Id both in your formula, and as the Target Property, both those references would be pointing at your new project parameter and not at the underlying API property.

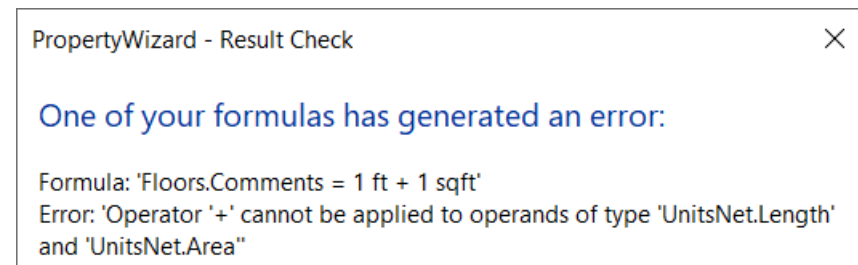
This can be the cause of some perplexing behaviour, so if your formula is not doing what you expect it is worth checking which properties you're really referencing.

Type mismatch errors (New in V1.4.0)

V1.4.0 has tightened up the code that handles measurement units (Lengths, Areas, etc), and now shows an error if the formula units do not match the target property units. So if you are setting a length parameter, your formula has to generate a length value:



You will also get type mismatch errors if you try to perform arithmetic on incompatible values. For example, trying to add a Length to an Area will result in an error like this:

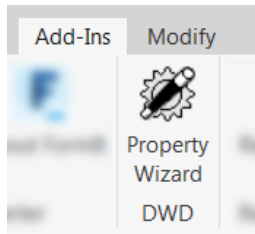


Reference

This section provides reference material.

Running PropertyWizard

PropertyWizard adds one button to a panel on the Add-Ins toolbar:

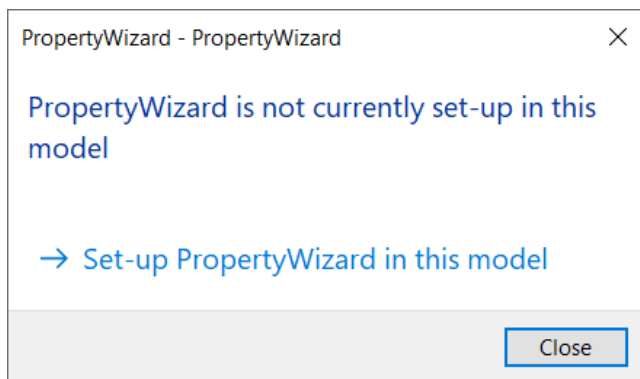


The button allows you to activate PropertyWizard and create formulas.

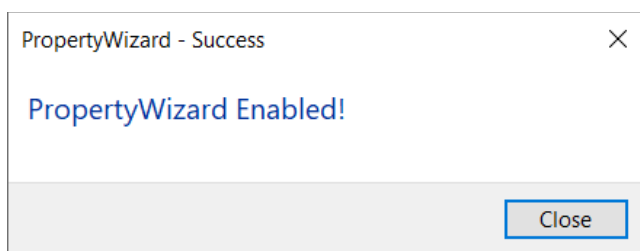
Activating PropertyWizard in a project

PropertyWizard will do nothing until you activate it.

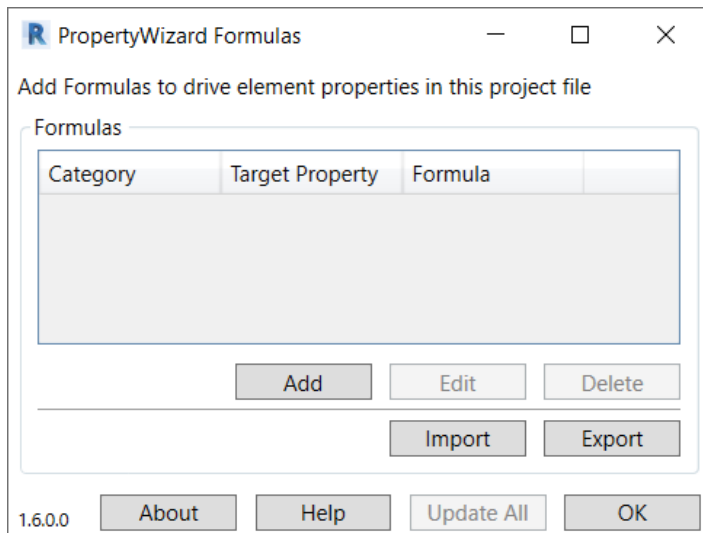
You activate it for each Revit .rvt file individually: just click the PropertyWizard button. In a new project, you will get this dialog:



Click *Set-up PropertyWizard in this model* to set up PropertyWizard in the project file. This will add PropertyWizard's shared parameter to the model.

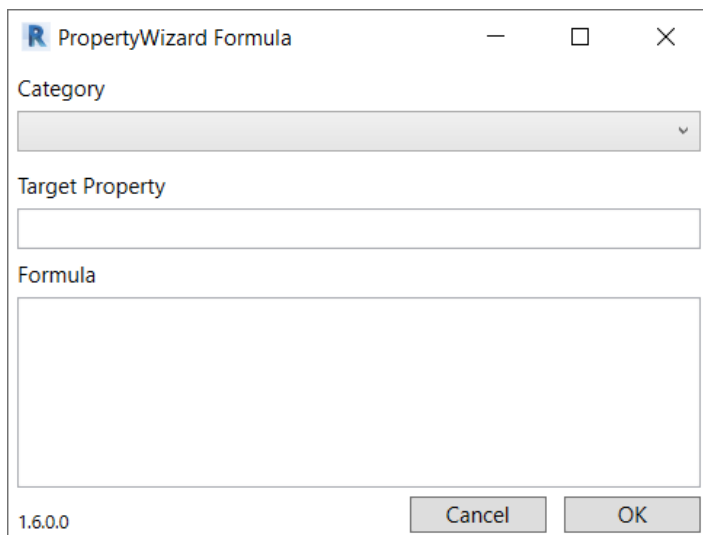


Click the Close button to launch the PropertyWizard main window, which will list the formulas you add to the project.



From here you can add new formulas, edit and delete your existing ones, and import and export your formulas.

The Add and Edit buttons both open the Formula Edit window:



Creating a formula

Each formula has three parts:

1. Select the Category – the formula will apply to all elements of this category.
2. Type in the name of the Target Property – the parameter you want to set.
3. Type in the Formula.

Target Properties

The Target Property can be a Type or an Instance property. For Instance properties, just enter the property name. For Type properties, enter 'Type', followed by a dot and then the property name.

Formulas

In the formula, you can type in:

- Operators and Functions, like +, sin() and if().

- Numbers and Integers. Numbers have a decimal point, Integers are whole numbers.
- Lengths (using the suffixes *mm*, *cm*, *m*, *inch* or *ft*).
- Areas (using the suffixes *sq m* or *sq ft*).
- Volumes (using the suffixes *cu m* or *cu ft*)
- Text strings, delimited by double-quotes.
- The literal values *true* and *false*.
- Names of source properties. The formula will use the value of the property.
- For built-in parameters, their BuiltInParameter enum name
- For shared parameters, their Guid (unique Id) value

Source Properties

You can refer to:

- Properties of the target element.
- Nested properties of the target element (e.g. properties of a Door's Level or a Wall's Type).
- API properties of the target element or nested elements (i.e. properties from the Revit API).
- Project Information parameters, by using *ProjectInfo.parametername*
- Global Parameters, by using *GlobalParameter.parametername*

Properties of the target element are just referred to by their name as seen in the Revit Properties window, for example, *Level*. If the name includes spaces or special characters, then it should be enclosed with square brackets like this: *[Sheet Name]*.

Nested properties are referenced by 'dot notation' as used in Dynamo. So the name of a Door's Level can be referred to as *Level.Name*. There is no limit to the depth of nesting, but all the properties except the last must be Elements or ElementIDs in this version.

API properties are the 'behind the scenes' properties exposed in the Revit API. For example, the *Location* property of a Structural Pile. To find the correct names, I use a combination of the Revit Lookup add-in, and the official Revit API Help (downloadable from Autodesk, or online at Gui Talarico's <http://www.revitapidocs.com/>).

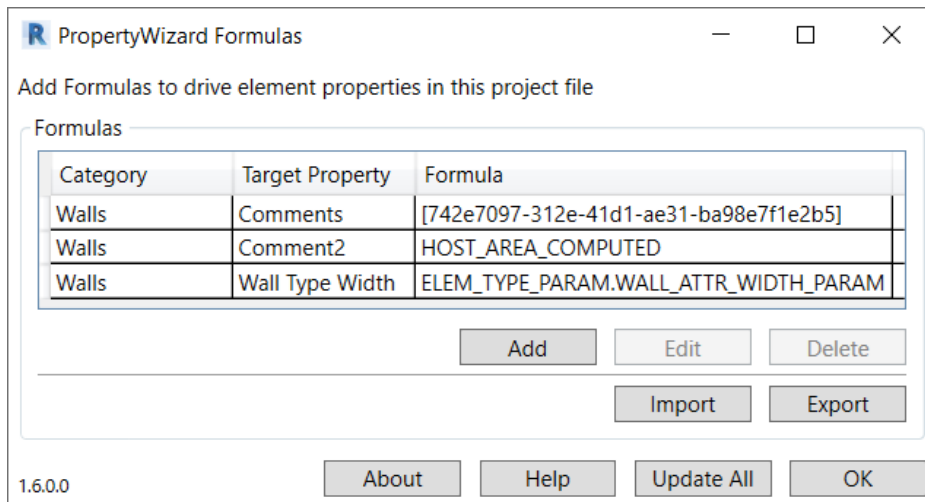
For more details on installing and using RevitLookup:

<http://www.davidwooddesign.co.uk/2018/08/19/how-to-install-revitlookup/>

<http://www.davidwooddesign.com/2021/03/31/how-to-use-revitlookup-getting-started/>

<http://www.davidwooddesign.co.uk/2018/08/27/what-properties-can-you-use-in-propertywizard-formulas/>

Sometimes, elements can have multiple properties with the same name. But there are ways to refer to a specific property. Each parameter that is 'built in' to Revit has a special 'BuiltInParameter' value, which you can find using RevitLookup, and you can use those values in PropertyWizard in place of the property name. Similarly, each Shared parameter has a special 'Guid' value, and you can use those values as well. The Guid is a long string of numbers and letters broken up by hyphens, so you need to enclose it in square brackets:



Parameter Types

In this version, Source and Target Properties can be any of these data types:

Common

- Text
- Integer
- Angle
- Area
- Length
- Number
- Volume
- Currency
- Yes/No

Structural

- BarDiameter
- CrackWidth
- ReinforcementCover
- ReinforcementLength
- ReinforcementSpacing
- SectionDimension.

HVAC

- DuctInsulationThickness
- DuctLiningThickness
- DuctSize

Electrical

- ElectricalCableTraySize
- ElectricalConduitSize

Piping

- PipeDimension
- PipeInsulationThickness
- PipeSize

Operators and Functions

| | Format | Notes |
|-----------------------------------|------------------|---------------------------------------|
| Number operators | | |
| Multiply | $a * b$ | |
| Divide | a / b | |
| Add | $a + b$ | + also provides string concatenation. |
| Subtract | $a - b$ | |
| Exponentiation | $a ^ b$ | a to the power b. |
| Number functions | | |
| Square Root | $\text{sqrt}(a)$ | $\text{sqrt}(4) == 2$ |
| Logarithm base 10 | $\log(a)$ | |

| | Format | Notes |
|----------------------------------------|----------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Exponentiation base 10 | exp(a) | |
| Absolute value | abs(a) | |
| Sine | sin(a) | |
| Cosine | cos(a) | |
| Tangent | tan(a) | |
| Arcsine | asin(a) | |
| Arccosine | acos(a) | |
| Arctangent | atan(a) | |
| Arctangent 2 | atan2(y, x) | Returns the arctan, given the y and x offsets. |
| PI | pi() | |
| Round | round(a) | Rounds a to an integer |
| Roundup | round(a, b) | Rounds a to b decimal places |
| Rounddown | roundup(a) | |
| Rounddown | rounddown(a) | |
| Logical functions | | |
| And | and(a, b...) | True only if both a and b are true. |
| Or | or(a, b...) | True if either a or b is true. |
| If | if(a, b, c) | If a is true, then b, else c. |
| Flow control functions | | |
| Try | try(a, b) | Tries a. If a throws an error, returns b |
| Logical operators | | |
| And | a && b | True only if both a and b are true. |
| Or | a b | True if either a or b is true. |
| Not | !a | True if a is false. |
| Equal | a == b | True if a is equal to b. |
| Not equal | a != b | True if a is not equal to b. |
| Greater than | a > b | True if a is greater than b. |
| Less than | a < b | True if a is less than b. |
| Greater than or equal to | a >= b | True if a is greater than, or equal to, b. |
| Less than or equal to | a <= b | True if a is less than, or equal to, b. |
| String Functions | | |
| String concatenation | strcat(a, b,...) | strcat("bow", "tie") == "bowtie" |
| String length | strlen(a) | |
| Sub-string | substr(<text>, <start>, <count>) | Returns part of <text> starting at character number <start> and of length <count>. Character numbers start at zero for the first character. substr("tree", 1, 2) == "re" |

| | Format | Notes |
|-----------------------|---------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| | substr(<text> ,<start>) | Returns part of <text> from character number <start> to the end of the text. substr("tree",1) == "ree" |
| Left-hand sub-string | leftstr(<text>, <count>) | Returns <count> characters from the start of <text>. |
| Right-hand sub-string | rightstr(<text>, <count>) | Returns <count> characters from the end of <text>. |
| String find | instr(<text1>, <text2>) | Finds <text2> in <text1>. Returns the number of the start character (from the first character = 0). Returns -1 if <text2> not found. instr("tree","re") == 1 |
| String format | format(a, b...) | Implements .Net String.Format |
| To lowercase | toLowerCase(<text>) | Converts the text to lowercase |
| To uppercase | toUpperCase(<text>) | Converts the text to uppercase |
| To title case | toTitlecase(<text>) | Capitalises each word in the text |
| Units | | |
| Millimetres | a mm | |
| Centimetres | a cm | |
| Metres | a m | |
| Inches | a inch | |
| Feet | a ft | |
| Square metres | a sq m | |
| Square feet | a sq ft | |
| Cubic metres | a cu m | |
| Cubic feet | a cu ft | |
| Coordinates | | |
| Internal -> Shared | internalToShared(a) | Converts a Revit database XYZ value into shared coordinates |
| Shared -> Internal | sharedToInternal(a) | |

Import/Export

Export your current formulas to a file on disk using the Export button on the Main window.

Import those formulas into another model using the Import button.

Avoid hand-editing the formula file.

Release Notes (What's New)

Compatibility

If you are sharing a file between users, I recommend that all users who are working on the same Revit model use the same version of PropertyWizard.

Each new version of PropertyWizard is designed to work properly with Revit files that contain formulas from previous versions. Also, the current version is file-compatible with previous versions of PropertyWizard, meaning the PropertyWizard data stored in the Revit model file has not changed. Of course, if you write formulas using features from new versions of PropertyWizard, those formulas will not work in older versions.

V1-9-0 New Features

- Revit 2023 version added.
- Extra Source and Target ParameterTypes added: PipeSize, DuctSize, PipeInsulationThickness, DuctInsulationThickness. Source and Target ParameterTypes added: PipeDimension, DuctLiningThickness, BarDiameter, ElectricalCableTraySize, ElectricalConduitSize, CrackWidth, ReinforcementCover, ReinforcementLength, ReinforcementSpacing, SectionDimension.
- Bugfixes and stability improvements.

V1-8-5 New Features

- Source ParameterTypes added: PipeSize, DuctSize, PipeInsulationThickness, DuctInsulationThickness.
- Bugfixes.

V1-8-4 New Features

- By default, does not 'Update All' on file Open. Added Checkbox in About dialog to switch on 'Update All' on Open and Synchronize.
- Bugfixes.

V1-8-3 New Features

- cm and inch units added.
- WorksharingInfo.Creator and WorksharingInfo.LastChangedBy removed.
- Bugfixes and stability improvements.

V1-8-2 New Features

- atan2() added.
- substr(<text>, <index>), leftstr(<text>, <count>) and rightstr(<text>, <count>) added.
- toLowercase(), toUppercase() and toTitlecase() added.
- WorksharingInfo.Creator and WorksharingInfo.LastChangedBy added.
- Function names changed to pascal case: localToShared(), sharedToLocal(). Old, all-lowercase versions will still work.
- Bugfixes and stability improvements.

V1-7-5 New Features

- Code Signing.
- Bugfix – CheckoutElementsRequestTooLarge exception with cloud models.

V1-7-1 New Features

- Revit 2022 version added.
- Revit 2016 and 2017 versions removed.
- Assemblies added to Category list.
- Bugfixes and stability improvements.

V1-6-0 New Features

- Export and Import your formulas.
- Round(a, b) function added – to round to a number of decimal places.
- Round functions handle midpoint values predictably.
- Trig functions now return radians.
- Bugfixes and stability improvements.

V1-5-0 New Features

- Access built-in parameters using their BuiltInParameter Enum name.
- Access shared parameter using their Guid (unique Id) value.
- 'Process all formulas and ignore all errors' option added to Result Check dialog.
- Title Blocks added to Category list.
- Bugfixes and stability improvements.

V1-4-3 New Features

- Bugfix – Hyphens in parameter names.
- Bugfixes and stability improvements.

V1-4-2 New Features

- Bugfix – CheckoutElementsRequestTooLarge exception in Revit 2021 with BIM 360.
- Bugfixes and stability improvements.
- Versions for Revit 2016 and 2017 not updated (will be removed in a later version)

V1-4-0 New Features

- Formulas can now target Type parameters by using *Type.parametername*.
- Formulas can now read Global Parameters by using *GlobalParameter.parametername*.
- Added more checks around mismatched units.
- Bugfixes and stability improvements.

V1-3-0 New Features

- Revit 2021 version added.
- Privacy Policy links added.

V1-2-5 New Features

- Bugfixes and stability improvements.

V1-2-4 New Features

- Bugfixes and stability improvements.

V1-2-3 New Features

- Bugfixes and stability improvements.

V1-2-2 New Features

- Synchronisation updates.
- Bugfixes and stability improvements.

V1-2-1 New Features

- Quickfix to synchronisation speed.
- Bugfixes and stability improvements.

V1-2-0 New Features

- Volume units added.
- Installation size reduced.
- Bugfixes and stability improvements.

V1-1-0 New Features

- Revit 2020 version added.
- Sqft unit added.
- Length and Area string formatting matches Revit Project Units setting.
- Help button added to main window.
- Bugfixes and stability improvements.

V1-0-5 New Features

- Bugfixes and stability improvements.

V1-0-4 New Features

- Feedback window clarified.
- About window updated.
- Bugfixes and stability improvements.

V1-0-3 New Features

- Improved handling of incompatible formulas.
- Improved UX for licensing.
- Improved UX for logging.
- Bugfixes and stability improvements.

V1-0-2 New Features:

- Improved reporting of formula errors.
- Bugfixes and stability improvements.

V1-0-1 New Features:

- Bugfixes and stability improvements.
- Preview version for Autodesk App Store.

V1-0-0 New Features:

- Preview version for Autodesk App Store.