

Brain Dump: Property Data Enhancements

Scott Arvin
Autodesk, Inc
March 2003

Terminology

There are lots of terms containing the word 'property' in them, so first of all I will show a diagram that helps to relate them all.

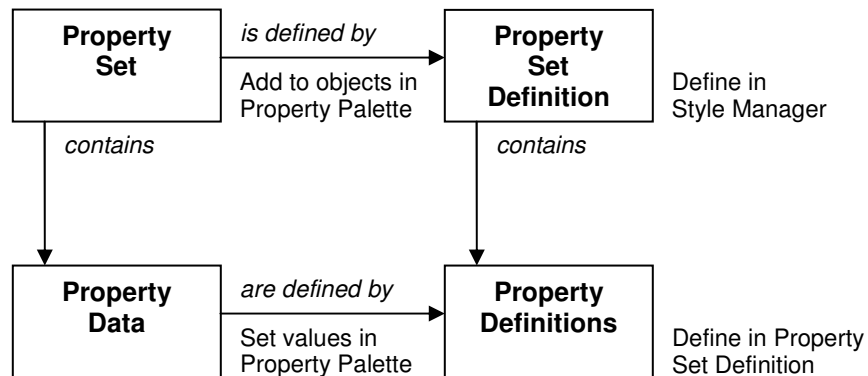









Figure 1 – Property Data Terminology

Property definitions

In ADT2004 we added five new types of property definitions. The two existing types are Manual and Automatic. The difference between these is that a Manual data value is set by the user, while an Automatic data value is a query of a hard coded object or style property. All of the new types of property definitions are also automatic; they acquire data from a specific place and cannot be directly set by the user. Table 1 lists all available property definitions with a quick description of each.

Table 1 - Property Definition Types

Property Definition	Description
 Manual	set by the user
 Automatic	get from object or style property
 Formula	get from other property data and process it somehow
 Location	get from Space, Area, or AEC Polygon
 Classification	get from classification property data, or get classification
 Material	get from material property data, or get material
 Project	get from project data

Automatic Property Definition

In the past you had to select Automatic Property Sources individually. This has been changed depending on how many objects or styles the Property Set Definition applies to. If it applies to only one object or style you can select multiple sources to create multiple Automatic Property Definitions at one time, as shown in Figure 2.

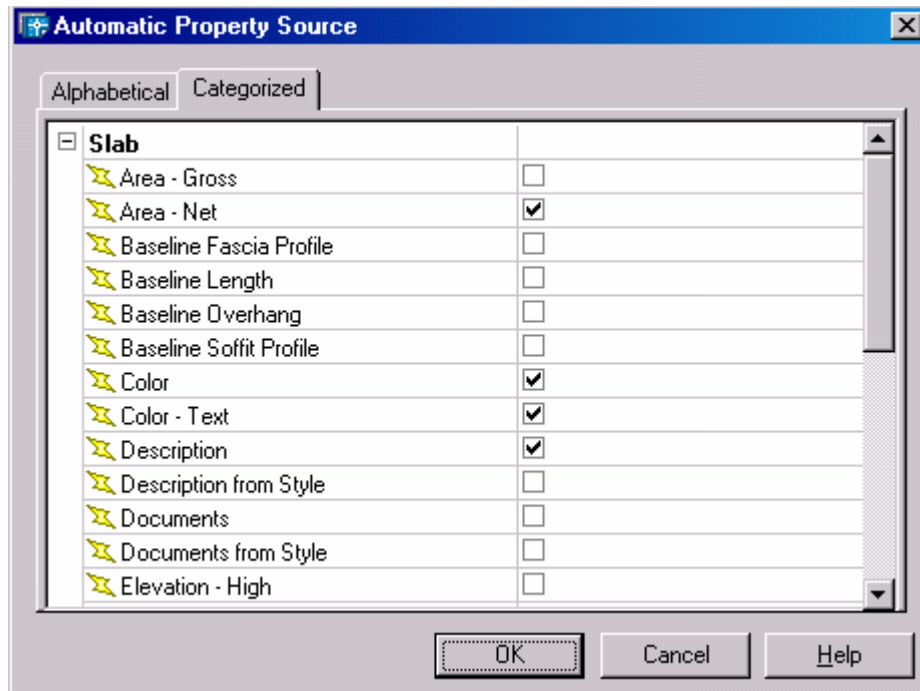


Figure 2 - Automatic Property Source dialog

If the Property Set Definition applies to more than one object or style, you can only create one Automatic Property Definition at a time, but selecting sources from the different objects is now easier. If you check/uncheck a source for one object, the source with the same name will be checked/unchecked for all other objects. This will help when needing a source that is the same for all objects, such as Color or Layer. If the Ctrl key is pressed before selecting, only the item selected will be checked/unchecked, leaving the source for all other objects unchanged. This will help when needing sources with different names for different objects.

Formula Property Definition

A Formula Property Definition was designed to perform computation on one or more property data values. For example, if you have automatic door Width and Height properties you can create a Perimeter property definition with the formula "2 * [Width] + 2 * [Height]". See how formulas are used to number doors based on room numbers in the example at the end of the Project Property Definition section.

A formula is computed using VBScript, so all functions available in VBScript are available in a formula property definition. Basically you have a complete programming language available to massage your property data however you want. In the formula dialog you define a string that is the formula itself. Into this string you can insert references to other property definitions either within the same property set definition or from other property set definitions that might be on the object. The values of these other property definitions are inserted into the formula string, and then the string is evaluated with VBScript. The result of that evaluation is returned as the value of the formula property.

Formulas are going to be the most difficult property definition to work with, because for all but the simplest tasks of basic mathematics and concatenating strings together, programming is required. Let me say that again – *creating formulas is often programming!!* So they are not for the faint of heart, and will likely take some effort to make them work correctly. You're basically programming with values you can't easily predict, and without the aid of a development environment. Yes, this may be difficult to use at times, but it is incredibly flexible once you get the hang of it (as with many AutoCad features) and was very easy to implement; the alternative was to not have the feature at all.

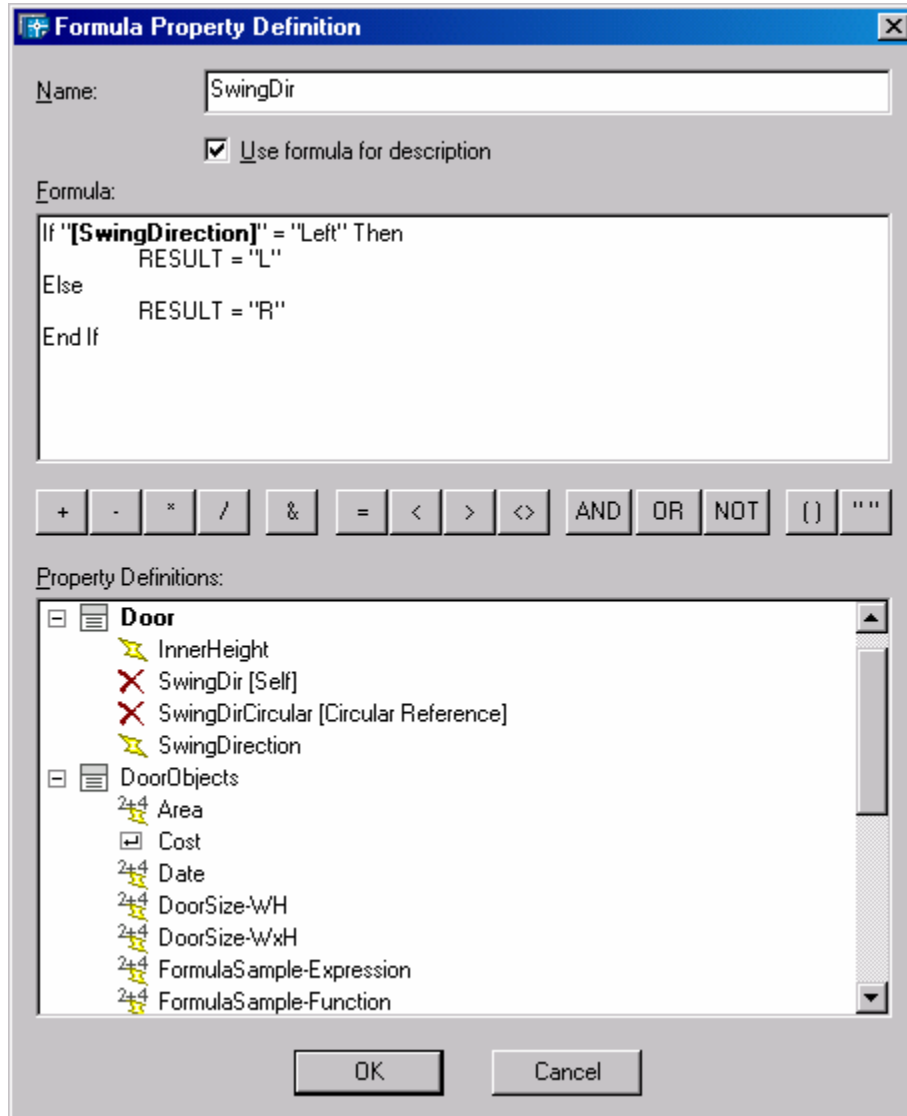


Figure 3 – Formula Property Definition dialog

Figure 3 shows the Formula Property Definition dialog. The Formula edit box contains the formula string itself. Below it are buttons used to paste common operations into the formula. The parenthesis and quotes buttons on the right end surround the current selection with parenthesis and quotes, respectively. All other buttons replace the current selection.

Property Definitions

The Property Definitions tree at the bottom contains all property definitions that can be applied to objects in the Applies To list of the property set definition containing the formula being edited.

Notes about property definitions in formulas:

- ▶ If the property definition to be inserted in the formula comes from the same property set definition as the formula, only the property definition name is displayed, e.g. "[SwingDirection]". Otherwise the property set definition name is displayed as well, e.g. "[DoorObjects:Area]".
- ▶ The red cross ✗ means the property definition cannot be used in that formula, because:
 - it is the formula being edited, as indicated by "[Self]" after its name. You can't have a value in the formula be the formula itself.
 - it is a property definition that eventually refers to the formula being edited, as indicated by "[Circular]" after its name. Typically these are location or formula property definitions.
- ▶ In order to use the value of a property definition in a formula you **must** select it from the tree, so that it displays as bold, e.g. [SwingDirection], and in the Formula edit box you can only select the entire name and not just part of it. You cannot simply type the name of the property definition within brackets. If you do, it will display and behave as plain text like the rest of the formula. The reason for this is that a reference to the property definition is stored in the formula, not its name, so if you can change the name of the property definition the new name will be displayed in the formula. Another consequence of this is that when you copy and paste a formula string from one formula to another, you must reselect property definitions used in it.

Property Data Formats

Before the value of a property definition is inserted in the formula string, it is formatted with its property data format. If we didn't you couldn't get Door Size results like "3'-0" x 6'-8"", and instead would get results like "36 x 80". If you're going to do numerical calculations with a property that is normally formatted, then you should create another property definition that is unformatted. For example, create one automatic property called [Width] that uses an architectural format, and another called [WidthUnformatted] that is unformatted. Use [WidthUnformatted] in formulas, which then use the architectural format. This should be no big deal for automatic properties. To avoid having to duplicate manual property data, create an unformatted manual property to hold the actual data, and then create a formatted formula referencing the manual property.

Expressions vs. Functions

An *expression* is a simple single line statement using basic mathematical operations such as "+", "-", "*", and "/", or the string concatenation operator "&".

A *function* is a more complicated formula that uses multiple statements using VBScript syntax. If the formula contains the string "RESULT", in all caps, a function called RESULT is created and sent to VBScript for evaluation. For example, if this is the formula string:

```
If "[SwingDirection]" = "Right" Then
    RESULT = "R"
Else
    RESULT = "L"
End If
```

and the value of [SwingDirection] is "Left", then this is sent to VBScript for evaluation:

```
Function RESULT()
    If "Left" = "Right" Then
        RESULT = "R"
    Else
        RESULT = "L"
    End If
End Function
```

Visual Basic programmers will recognize that assigning a value to the name of the function is how you set the return value of the function.

How a Formula is evaluated

When the value of the formula property is needed, the process used to determine that value is:

1. For each property definition in the formula,
 - a. the value of the property using it is retrieved from the object or style,
 - b. it is formatted using the property data format defined for the property definition, and
 - c. the formatted value is inserted in the formula in place of the property definition name and its surrounding brackets.

Note: Make sure that all property sets used in the formula have been added to the object or style. Otherwise, “?”s will be inserted instead.

2. If the formula string contains the word “RESULT”, it is evaluated as a VBScript function.
3. If the formula string does not contain the word “RESULT”, it is evaluated as a mathematical expression.
4. If VBScript doesn’t know how to evaluate the resulting formula, the resulting formula becomes the value of the formula property definition. Depending on the formula, there will be times when it looks like the formula is ‘exploded,’ i.e. what is displayed is the formula instead of what you think should be the result of the formula. This is so you can ‘debug’ the problem formula and see what it will take to fix it. If this was not done, it would be difficult to see what might be wrong.

Strings

It is possible to concatenate two or more properties in a formula by simply putting them together, such as

[RoomNumber][RoomNumberSuffix].

But in some circumstances that might lead to unintended results, as described in the next section on numbers. If so, try using the string concatenation operator “&” and surrounding properties with quotes, such as

“[RoomNumber]” & “[RoomNumberSuffix]”

Imperial architectural formats are going to make things doubly difficult, because often you want to display them as strings, but they already contain both the single and double quotes. There is no clean way of dealing with these directly in a function. Try using unformatted values, as described above in Property Data Formats, and setting an appropriate format for the formula itself.

Remember

And just in case you missed the point earlier, ***creating formulas is often programming!!*** ☺ So don’t be surprised if it takes a bit of work to get what you want!

Sample Formulas

Property Data Format	Description
Standard	Nothing special. Used for all property definitions unless otherwise noted.
Architectural	Real numbers set to Architectural, e.g. 2'-8"
CostFormula	Undefined property is set to "Unknown" (with quotes) instead of "?" (without quotes), so that if a property set is not on an object it can still be used in a formula. Otherwise, VBScript will not know how to evaluate it, and it will display 'exploded.'
Cost	Prefix property set to "\$"

Property Set Definition DoorObjects , applied to door objects		
Property Def	Format	Data and notes
Manual		
Cost	CostFormula	Real <i>Used in CostObjects:Cost to demonstrate how one cost property displayed in a schedule table could get different kinds of data from different objects. In this case, a unit cost not based on a variable.</i>
Automatic		
Height	Architectural	Height <i>Formatted as architectural for display in tables and tags.</i>
HeightUnformatted		Height <i>Unformatted for use in formulas.</i>
SwingDirection		Swing Direction
Width	Architectural	Width
WidthUnformatted		Width
Formula		
Area		[WidthUnformatted] * [HeightUnformatted] <i>Example of a simple expression.</i>
DoorSize-WH		[WidthUnformatted][HeightUnformatted] <i>This will concatenate width and height, so you can display "3080", but will display undesired results for fractional inches. For example, if the width was 30.75 this would concatenate as "30.7580", which using the standard format would be rounded to "30.76".</i>

Property Set Definition DoorObjects , applied to door objects (cont.)		
Property Def	Format	Data and notes
DoorSize-WxH		<p>width = [WidthUnformatted] widthFeet = CInt(width / 12) widthInches = width Mod 12 height = [HeightUnformatted] heightFeet = CInt(height / 12) heightInches = height Mod 12</p> <p>RESULT = CStr(widthFeet) & "-" & CStr(widthInches) & Chr(34) & " x " & CStr(heightFeet) & "-" & CStr(heightInches) & Chr(34)</p> <hr/> <p><i>Here's a fun one. This formula basically replaces the automatic "DoorSize-WxH" that's been available since ADT3. You can modify it to display door/window sizes in whatever format you like.</i></p> <ul style="list-style-type: none"> ▶ <i>Since this formula contains the string "RESULT" in all caps it will be evaluated as a function.</i> ▶ <i>width, widthFeet, etc. are variables used elsewhere in the function.</i> ▶ <i>You could use [WidthUnformatted] in the expressions for widthFeet and widthInches, but that would require getting the same information twice, so this is faster.</i> ▶ <i>Notice the use of CInt() and CStr() functions to convert values.</i> ▶ <i>Notice the use of quotes, the "&" string concatenation operator, and Chr(34), which is used to add a double quote to the string.</i>
Perimeter		<p>(2 * [HeightUnformatted]) + (2 * [WidthUnformatted])</p> <hr/> <p><i>Use parentheses liberally, to make sure you get the result you're looking for.</i></p>
SmallMediumLarge		<p>dHeight = [HeightUnformatted]</p> <p>If dHeight < 72 Then RESULT = "Small" Elseif dHeight < 96 Then RESULT = "Medium" Else RESULT = "Large" End If</p>
SwingDir		<p>If "[SwingDirection]" = "Left" Then RESULT = "L" Else RESULT = "R" End If</p>
Date		RESULT = CStr(Date)
Time		RESULT = CStr(Time)
		<hr/> <i>Examples using other VBScript functions.</i>

Property Set Definition SpaceObjects , applied to space objects		
Property Def	Format	Data
Manual		
HeightOverride		Text
Automatic		
HeightUnformatted		Height
Formula		
Height	Architectural	<p>strHeightOverride = "[HeightOverride]"</p> <p>If strHeightOverride = "" Then RESULT= CDbI([HeightUnformatted]) Else RESULT = strHeightOverride End If</p> <hr/> <p><i>Here is a formula stolen from Rob Starz (thanks Rob!) that demonstrates how you can have a manual override to an automatic property. The classic example is spaces, whose height in ADT only returns a single real number. If you have a space whose height varies, you can use this formula to indicate that in tags and tables.</i></p>

Property Set Definition WallStyles , applied to wall styles		
Property Def	Format	Data
Manual		
CostPerLF	CostFormula	Real
		<hr/> <i>Cost per linear foot.</i>
Automatic		
Length		Length
Formula		
Cost		<p>CDbI([CostPerLF] * [Length])</p> <hr/> <p><i>Used in CostObjects:Cost to demonstrate how one cost property displayed in a schedule table could get different kinds of data from different objects. In this case, a variable cost based on length.</i></p>

Property Set Definition SlabStyles , applied to slab styles		
Property Def	Format	Data
Manual		
CostPerSF	CostFormula	Real
		<hr/> <i>Cost per square foot.</i>
Automatic		
Area-Net		Area – Net
Formula		
Cost		<p>CDbI([Area-Net] * [CostPerSF])</p> <hr/> <p><i>Used in CostObjects:Cost to demonstrate how one cost property displayed in a schedule table could get different kinds of data from different objects. In this case, a variable cost based on area.</i></p>

Property Set Definition CostObjects , applied to all objects		
Property Def	Format	Data
Automatic		
ObjectType		Object Type
Formula		
Cost	Cost	Select Case "[ObjectType]" Case "Wall" RESULT = [WallStyles:Cost] Case "Slab" RESULT = [SlabStyles:Cost] Case "Door" RESULT = [DoorObjects:Cost] Case Else RESULT = "Unknown" End Select <hr/> <i>Here is an example of how you can use formulas to display different kinds of data from different objects in the same schedule table column. For example, to do a cost estimate you need different cost formulas for different object types, e.g. wall costs are based on area or length, slab costs by area, door costs by unit, etc. Since table columns can only get one property, you can use a formula similar to this to get values based on object type.</i>



Location Property Definition

A Location Property Definition can be used to retrieve property data on one object based on its location relative to another object. The main reason this type was implemented was to allow door numbers to be based on room numbers. For example, a space object could have a RoomNumber property definition, and the door next to the space could retrieve that RoomNumber and use it as part of its door number. See another detailed example of how this is used at the end of the Project Property Definition section. In order to create a Location Property Definition, you must already have created a property set definition that is applied to spaces, areas, or AEC polygons.

Sample process for getting space data in doors:

- ▶ Create a property set definition applied to spaces, with a number of property definitions.
- ▶ Create another property set definition applied to doors.
- ▶ In it create a location property definition, and select a property in the tree, as shown in Figure 4.
- ▶ Create a number of spaces.
- ▶ Add the space property set to the spaces.
- ▶ Create a door.
- ▶ Select the door, and notice its grips.
- ▶ Add the door property set to the door.
- ▶ Select the door again, and notice that it has a new 'Star' shaped grip, as shown in Figure 5. This grip is called the Data Location grip, and is used to define the data location point and find spaces, areas, and AEC polygons for location property definitions.
- ▶ Select the data location grip and move it over a space.
- ▶ In the extended data tab of the property palette, the space's data should be displayed in the door property set.

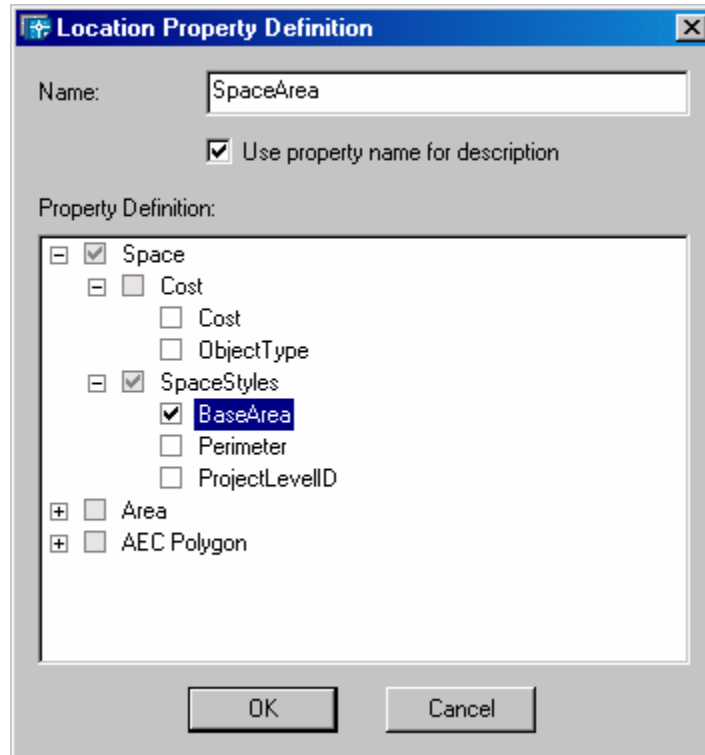


Figure 4 – Location Property Definition dialog

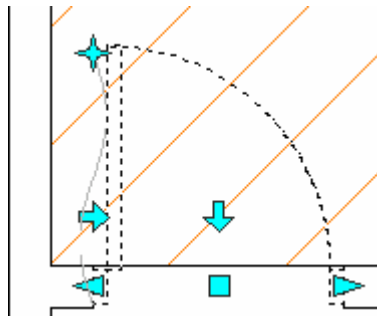


Figure 5 – Data Location Grip

Notes about location property definitions:

- ▶ The data location grip will display on an object only after the first time a property set definition containing a location property definition is attached to it.
- ▶ There is only one data location point for an object, no matter how many location property definitions it may use, and no matter how many location objects, i.e. spaces, areas, or AEC polygons, they use.
- ▶ The data location point for doors is initially offset to one side, the swing side of a single swing door. For all other object types, the data location point is initially at the origin of the entity's coordinate system.
- ▶ A location property will search for the closest object under the data location grip in the world coordinate system, i.e. in the negative Z direction.
- ▶ If one object has one location property referring to a space property, and another referring to an area property, the first will find the closest space and the second will find the closest area. So the same data location point is used to find different kinds of objects.
- ▶ The search for spaces, areas, or AEC polygons will extend across all xrefs.

- ▶ In ADT2004 the process for finding the location is *very* slow, as there wasn't enough time to optimize the search process. Performance should hopefully be greatly improved in the next release.

As noted, the reason this property definition was implemented was to allow doors to get room information. Another similar use case, implemented by Paul Aubin, is to number one room based on another room. For example, in a hotel plan where a guest room is numbered 206, the room's bathroom and closet could be numbered 206A and 206B, respectively. Another use case is in tracking furniture and equipment for facility management. For example, a room has property data for the occupant's name, and multi-view blocks for furniture or computers can use a location property definition to retrieve whose office they are in.



Classification Property Definition

In ADT2004 you can *classify* ADT styles by creating a Classification Definition that applies to one or more styles, creating a number of classifications, then assigning one of those classifications to each of your styles. For example, you can create a Classification Definition called "DoorTypes" that applies to Door Styles, with classifications Interior, Exterior, Toilet, etc. After classifying your door styles, you can create property set definitions and schedule table styles that apply only to specified classifications. So now you can create a door schedule that does not include toilet doors!

A Classification Property Definition can be used to retrieve an object's classification data. There are two different kinds of data you can retrieve, the name of the classification on a specified component, or a property of that classification. Figure 6 shows the Classification Property Definition dialog for a property set definition that applies to walls. First, on the left side you check the classification definition you are interested in. If you do not check the Classification Property check box, this property definition will return the name of the classification specified for the selected classification definition.

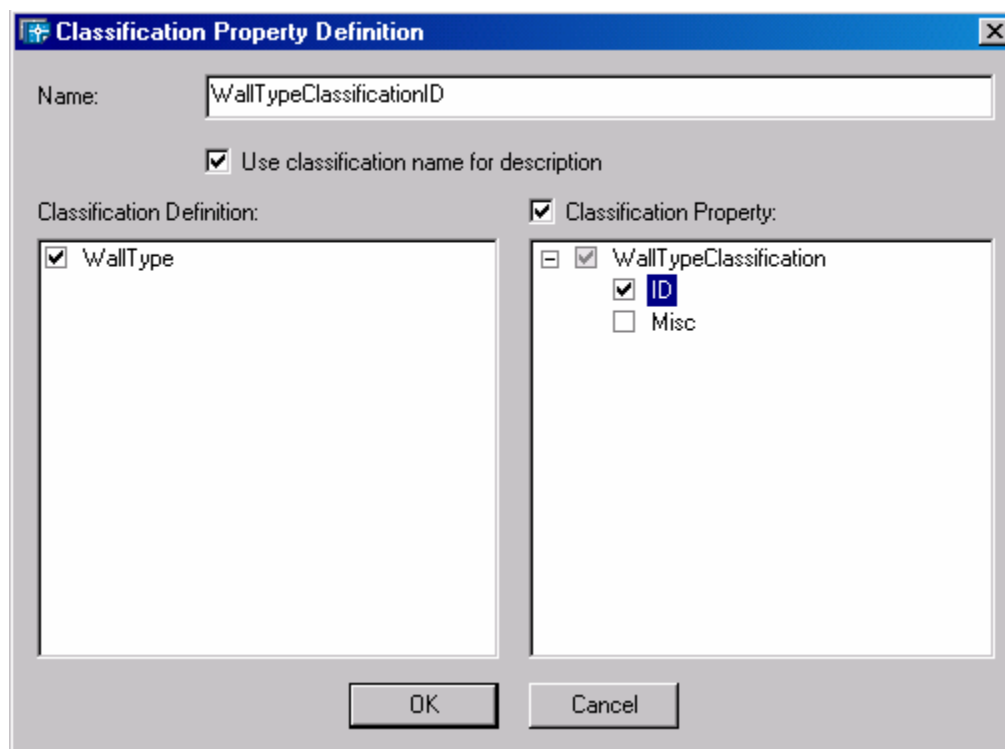


Figure 6 – Classification Property Definition dialog

In ADT2004 you can create a property set definition that applies to Classifications. If the Classification Property box is checked, you select one of these property definitions. Then this property definition will

return the value of that property for the classification. There was not a specific design reason for enabling property data on classifications, but it made sense to implement since classifications are a very generic feature and it was extremely easy to do. One use that comes to mind is manufacturer's information, e.g. you can create a classification definition of manufacturers or suppliers, and a property set definition with associated information that is available to your styles.

Material Property Definition

A Material Property Definition can be used to retrieve an object's material data. There are two different kinds of data you can retrieve, the name of the material on a specified component, or a property of that material. Figure 7 shows the Material Property Definition dialog for a property set definition that applies to doors. First, on the left side you check the component you are interested in. If you do not check the Material Property check box, this property definition will return the name of the property specified for the selected component.

In ADT2004 you can create a property set definition that applies to Material Definitions. You could then add properties to your materials, such as a display name, or cost per unit area or volume. If the Material Property box is checked, you select one of these property definitions. Then this property definition will return the value of that property for the material specified for the selected component. Since the material definition names in ADT content are fairly long and useless in schedule tables, e.g. "Doors & Windows.Wood Doors.Ash", you'll probably want to use a property set definition applied to materials to define a display name such as "Wood", and retrieve that name via a material property definition.

Note: Unfortunately, you cannot retrieve material data for objects that have a variable number of components, such as walls.

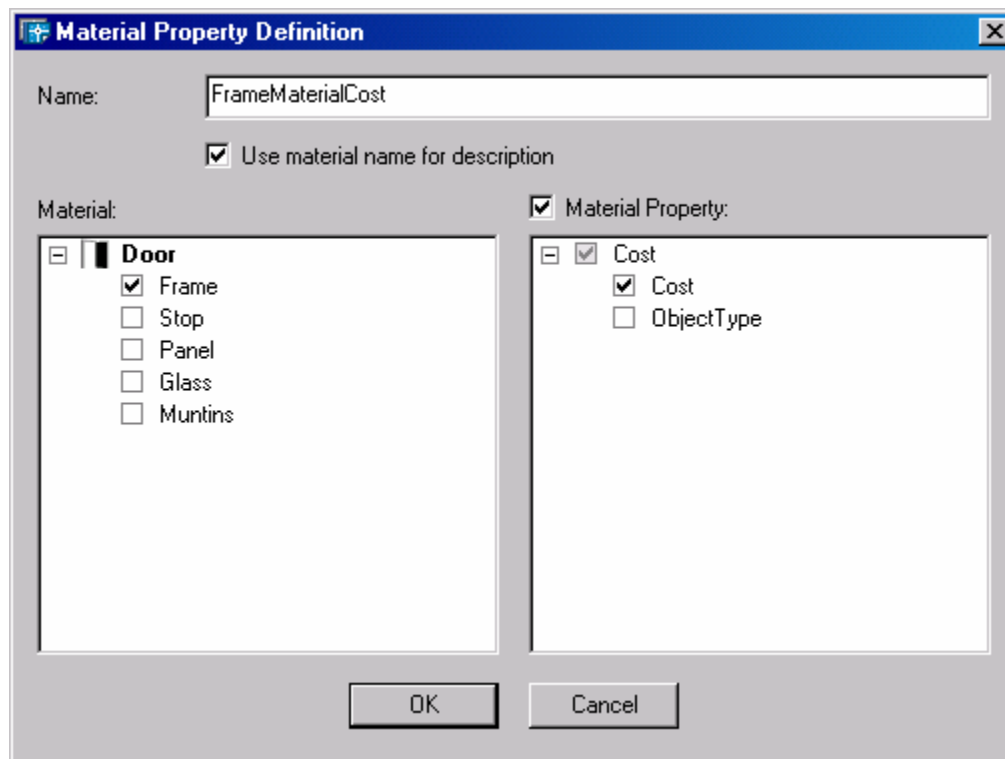


Figure 7 – Material Property Definition dialog



Project Property Definition

A Project Property Definition is used to get project information into property data. The main reason this type was implemented was to allow room numbers to be based on project level numbers. So a Space object could have one project property definition to retrieve the floor number the space is on, via the Level Id, e.g. "2", another manual property definition for the space's base number, e.g. "01", and a third formula property definition to tie the two together, resulting in a room number of e.g. "201". A detailed example follows below. Table 2 lists all the project property data that are available and Figure 8 shows the Project Property Definition dialog, where they are specified.

Table 2 - Project Property Data

Property	Description	Document Type
Name	Name of the project	All
Description	Description of the project	All
Number	Number of the project	All
Location	File location of the project on the hard disk or network drive	All
Number of Levels	Total number of levels in your project	All
Level ID	ID of the level the object is on. For example if you are creating a wall property set definition with this property, and attach it to a wall on the first floor, the extended data will display "Level 1". If you attach the property set definition to a wall on the second floor, this property will be changed accordingly.	Construct
Level Elevation	Elevation of the level the object is on.	Construct
Level Height	Height of the level the object is on.	Construct
Number of Divisions	Total number of divisions in your project	All
Division ID	ID of the division the object is in	Construct
Sheet Name	Name of the sheet drawing that contains the object.	Sheet
Sheet Description	Description of the sheet drawing that contains the objects.	Sheet
Details	In addition, you have properties for all project details you have created.	All

Note the Details project property! With this property, objects now have access to *any* kind of project data you desire, such as the owner's favorite color, or the average temperature on the building site in September. ☺

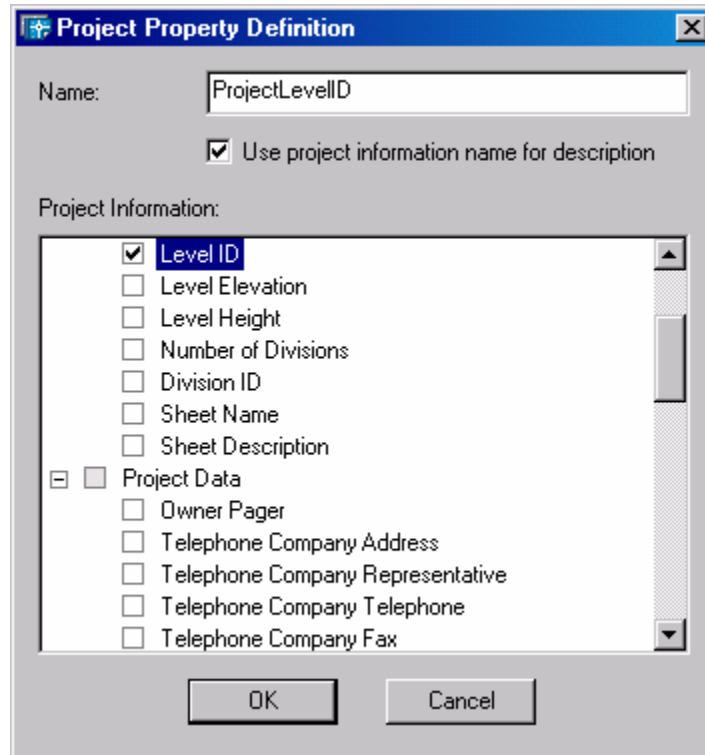


Figure 8 – Project Property Definition dialog

Project data and the currently active project

Project data is only available to an object if the drawing file that contains the object is in the currently active project. Most of the project data listed above is available for all types of project drawings, i.e. Sheets, Views, Constructs, and Elements. Some are only available if the object is in a Construct or a Sheet, as noted in Table 2. If the drawing is not of the correct document type, the property definition will display the Not Applicable setting of the current Property Data Format.

So what happens when the object's drawing is not contained in the currently active project? Property Data is typically displayed in three different places, Schedule Tags, Schedule Tables, and in dialog boxes such as the Extended Data tab of the Property Palette or the Property Data Browse dialog. Schedule Tags and Schedule Tables will display the value retrieved the last time the drawing was open and its project was active. In other words, *tags and tables displaying project data will not update if their project isn't active!* This means that it is possible that their data is wrong! For example, open a drawing with a tag that displays the current Level ID, close it, change the project's Level ID, close the project, then open the drawing. The tag will display the old ID!

Tags and tables also differ in what data gets updated in this situation. For tags, only project data will not be updated; any not project data will be updated. For tables, if only once cell in the entire table displays project data, the entire table will not be updated! This difference in behavior was due to differences in how old data is stored for tags vs. tables.

The third place mentioned above where property data is displayed is dialog boxes. If the drawing isn't in the currently active project, project data in the property palette and other dialog boxes will display as ****No Project****.

Putting it all together: Door numbers based on Room numbers based on project level numbers

Here is a process for creating door number properties that are based on room numbers, which in turn are based on project level numbers.

- ▶ Use one of your Model Management projects that has some floors and put some spaces and doors on different floors.
- ▶ Create the following Property Set Definitions (PSD) and Property Definitions (PD):
- ▶ SpaceObjects
 - LevelNumber - a project property with source "Level ID"
 - BaseNumber - a manual integer property
 - RoomNumber - a formula property whose formula should read [LevelNumber][BaseNumber]
- ▶ DoorObjects
 - RoomNumber - a location property that refers to the SpaceObjects:RoomNumber property of Space entities.
 - DoorNumberSuffix - a manual text property that should be blank, but that might later store 'A', 'B', etc.
 - DoorNumber - a formula property whose formula should read [RoomNumber][DoorNumberSuffix]
- ▶ Add the SpaceObjects PSD to your spaces, and set appropriate manual values.
- ▶ Add the DoorObjects PSD to your doors, and set appropriate manual values.
- ▶ Select a door, and notice the Data Location grip.
- ▶ Select the Data Location grip and move it over a space.
- ▶ Bring up the Extended Data tab on the door's property palette, and notice the DoorNumber.
- ▶ Move the grip to another space.
- ▶ Notice the DoorNumber property should reflect data from the other space.

Table 3 shows how all of this works together. The resulting door number is in the bottom row, and all the data making up parts of the door number come from other places and 'fall through' to the resulting value.

Table 3 – Project-based door numbers

Value	Property Set Definition	Property Definition	Description
2	SpaceObjects	LevelNumber	A Project Property Definition, getting the Level ID for the object for the current project.
01	SpaceObjects	BaseNumber	A Manual Auto Incrementing Property Definition
201	SpaceObjects	RoomNumber	A Formula Property Definition, with the Formula "[LevelNumber][BaseNumber]"
201	DoorObjects	RoomNumber	A Location Property Definition, getting the Space: RoomNumber property value for the space under the door's Data Location grip.
A	DoorObjects	DoorNumberSuffix	A Manual Text Property Definition
201A	DoorObjects	DoorNumber	A Formula Property Definition, with the formula "[RoomNumber][DoorNumberSuffix]"

Property Data Formats

(Formerly Schedule Data Formats)

Not Applicable

Some Automatic property values do not apply to certain objects for one reason or another. For example, the Swing Direction property does not apply to door styles that do not swing, or the Rise property does not apply to window shapes that do not have a rise. You can control how these values display with the Not Applicable setting in the Property Data Format, which is set to "NA" by default.

Round Off

You now have better control over how a real number is rounded – Up, Down, or to the Nearest specified value.

Scale

You can now scale real number values before they are displayed. To convert the current drawing units to other units set the scale to the appropriate conversion value, for example to convert inches to millimeters set scale to 25.4.