

+31 85 0653 776

hello@cadbooster.com



How to use custom properties in the SOLIDWORKS API (part 9)

2023-07-22 / by Peter Brinkhuis / in API / macros

This is part nine of our [blog series](#) about the SOLIDWORKS API. There are links to all other parts at the [bottom](#) of this post.

Today, we are diving into custom properties.

In this blog post, you'll find

1. [What are custom properties?](#)
 1. [Read this first](#)
2. [Three places where you can add custom properties](#)
 1. [Custom properties for files](#)
 2. [Custom properties for configurations](#)
 3. [Custom properties for cut-list folders](#)
3. [What is the CustomPropertyManager?](#)
 1. [I'd like to speak to the manager](#)
 2. [CustomPropertyManager property accessors](#)
 3. [The most important methods and properties](#)
4. [How to read custom properties via the API](#)
5. [How to write custom properties via the API](#)
6. [How to check if a custom property exists](#)
7. [Data types for custom properties](#)
 1. [Available data types](#)
 2. [Numbers and dates, represented by strings](#)
8. [Resolved values vs normal property values](#)
 1. [What is the "resolved value"?](#)
 2. [How to enter an equation or variable from code](#)
 3. [Custom property values in equations](#)
9. [Custom properties for cut list folders](#)
 1. [Get all children of the "solid body folder"](#)
 2. [Find all cut list folders in the feature tree](#)
10. [A faster way: how to use the Document Manager](#)
 1. [What is the Document Manager?](#)
 2. [How fast is reading custom properties using the Document Manager?](#)
 3. [Algorithm for reading of writing custom properties](#)
 4. [Sample code for the Document Manager](#)
11. [A user-friendly way: using SolidDNA](#)
12. [How to write Summary Info](#)
13. [Products for batch processing custom properties](#)

1. What are custom properties?

A *custom property* is a piece of data with a name, type and value. Within SOLIDWORKS, we generally use it to store data like the description, mass and author of a file.

	Property Name	Type	Value / Text Expression	Evaluated Value
1	DrawnDate	Date	22-10-2019	22-10-2019
2	CheckedBy	Text	PB	PB
3	DrawnBy	Text	EM	EM
4	Weight	Text	"SW-Mass@Strongback.SLDPRT"	146458706.59
5	Number	Number	23,000000	23,000000
6	PartNo	Text		
7	<Type a new property>			

It's called *custom* because you can pick your own name and type; you get to create your own *property*.

But a custom property is not actually a SOLIDWORKS concept, it's used by Microsoft for all kinds of Windows files. See for example [Creating Custom Properties](#) on the Microsoft App Development website.

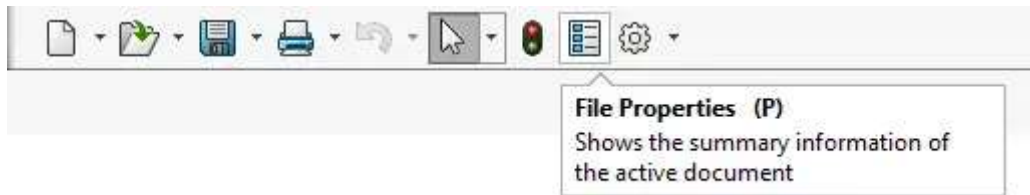
1.1. Read this first

In this blog post, we will explain how to work with custom properties in the SOLIDWORKS API. We assume that you know what custom properties are. If you don't, please read these two blog posts first:

1. [How to work with custom properties \(and cut lists\)](#)
2. [All available variables for custom properties \(and cut lists\)](#)

2. Three places where you can add custom properties

To access the custom properties, click the *File Properties* button in the top bar of SOLIDWORKS.



1. **Summary info:** contains file properties that are readable by Windows. These are not custom properties. See [How to write Summary Info](#).
2. **Custom:** custom properties for the entire model/file
3. **Configuration Specific:** custom properties for each configuration

The image shows the 'Summary Information' dialog box in SOLIDWORKS. It has three tabs: 'Summary', 'Custom', and 'Configuration Specific'. The 'Summary' tab is selected. At the top right, there is a 'BOM quantity:' dropdown menu set to '- None -' and an 'Edit List' button. Below this is a table with the following data:

	Property Name	Type	Value / Text Expression	Evaluated Value
1	DrawnDate	Date	22-10-2019	22-10-2019
2	CheckedBy	Text	PB	PB
3	DrawnBy	Text	EM	EM
4	Weight	Text	"SW-Mass@Strongback.SLDPRT"	146458706.59
5	Number	Number	23,000000	23,000000
6	PartNo	Text		
7	<Type a new property>			

At the bottom of the dialog are 'OK', 'Cancel', and 'Help' buttons.

2.1. Custom properties for files

These properties are bound to the file, to the entire model. To access these properties, get the [CustomPropertyManager](#) and pass an empty string for the name of the configuration.

An example in VBA:

```

1 Option Explicit
2
3 Sub main()
4     Dim swApp As SldWorks.SldWorks
5     Set swApp = Application.SldWorks
6
7     Dim swModel As ModelDoc2
8     Set swModel = swApp.ActiveDoc
9
10    Dim swExtension As ModelDocExtension
11    Set swExtension = swModel.Extension
12
13    Dim customPropManager As CustomPropertyManager
14    Set customPropManager = swExtension.CustomPropertyManager("")
15 End Sub

```

Now that you have access to the CustomPropertyManager, you can read, add or delete custom properties on the file level. See the next section [What is the CustomPropertyManager?](#)

2.2. Custom properties for configurations

Configurations are designed for slight variations of your model. If you only change the looks (or hide/show assembly components), you should use Display States. But if the geometry varies, you need configurations. We explain the speed differences in our ebook [Secrets to SOLIDWORKS Performance](#).

Every configuration can have its own list of custom properties.

But be aware: if you add the same property to both the file and the configuration, SOLIDWORKS will use the configuration-specific one. See [Don't add a property in both the Custom and Configuration Specific tabs](#) in our other post.

The code to access the configuration-specific properties is nearly identical to the code above. The only difference is that you pass the name of the configuration you want to access:

```

1 Option Explicit
2
3 Sub main()
4     Dim swApp As SldWorks.SldWorks
5     Set swApp = Application.SldWorks
6
7     Dim swModel As ModelDoc2
8     Set swModel = swApp.ActiveDoc
9
10    Dim swExtension As ModelDocExtension
11    Set swExtension = swModel.Extension
12
13    Dim customPropManager As CustomPropertyManager
14    Set customPropManager = swExtension.CustomPropertyManager("Configuration1")
15 End Sub

```

If you already have an IConfiguration object, you can access the custom property manager directly:

```

1 Option Explicit
2

```

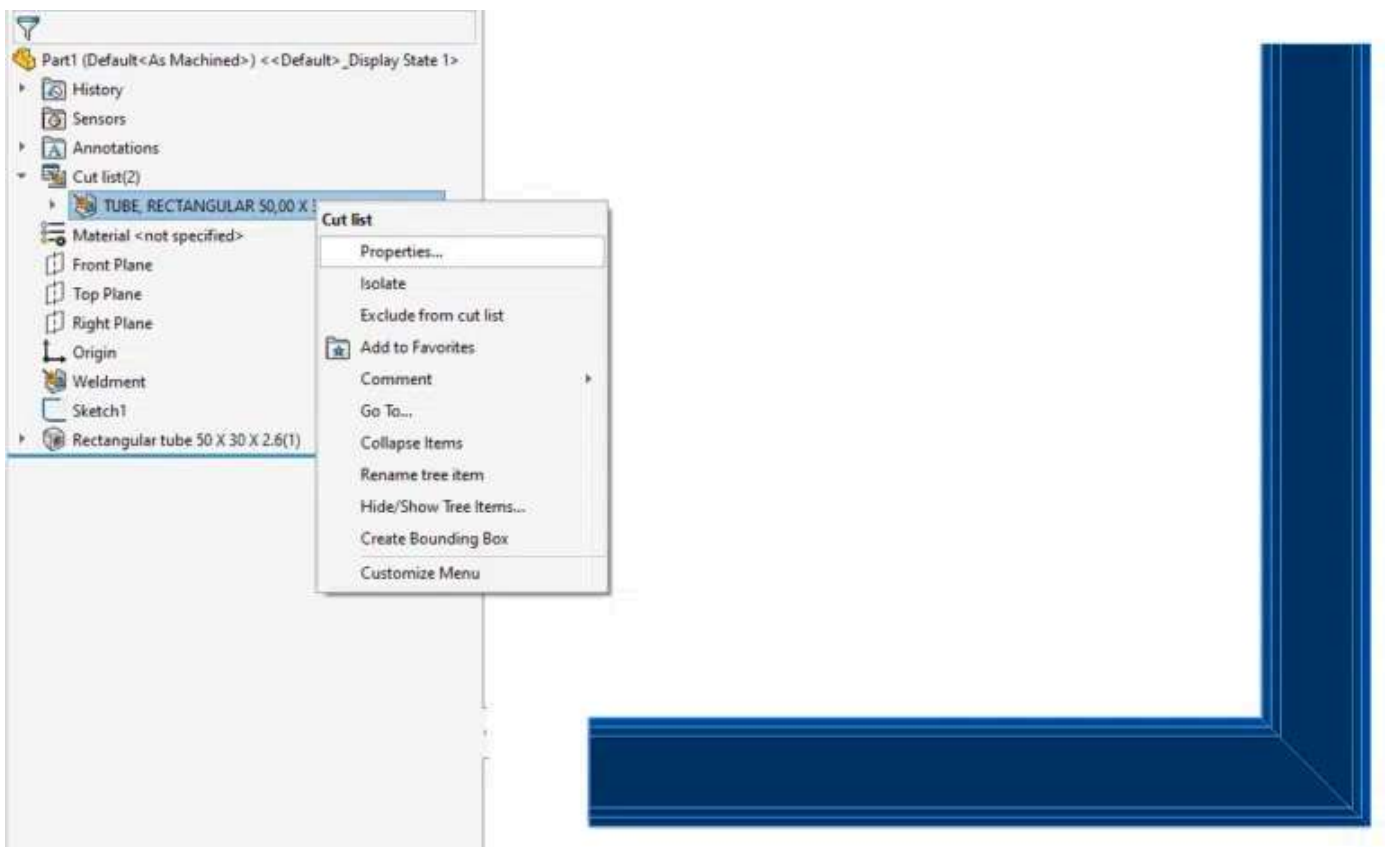
```

3 Sub main()
4   Dim swApp As SldWorks.SldWorks
5   Set swApp = Application.SldWorks
6
7   Dim swModel As ModelDoc2
8   Set swModel = swApp.ActiveDoc
9
10  Dim swConfiguration As Configuration
11  Set swConfiguration = swModel.GetActiveConfiguration
12
13  Dim customPropManager As CustomPropertyManager
14  Set customPropManager = swConfiguration.CustomPropertyManager
15 End Sub

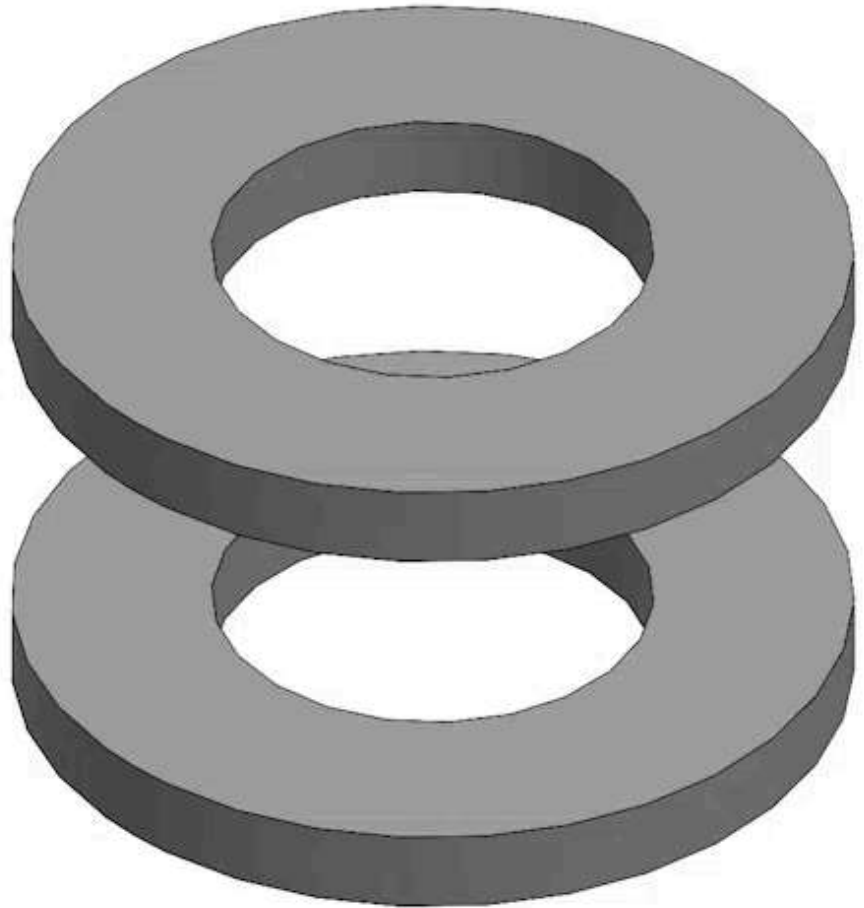
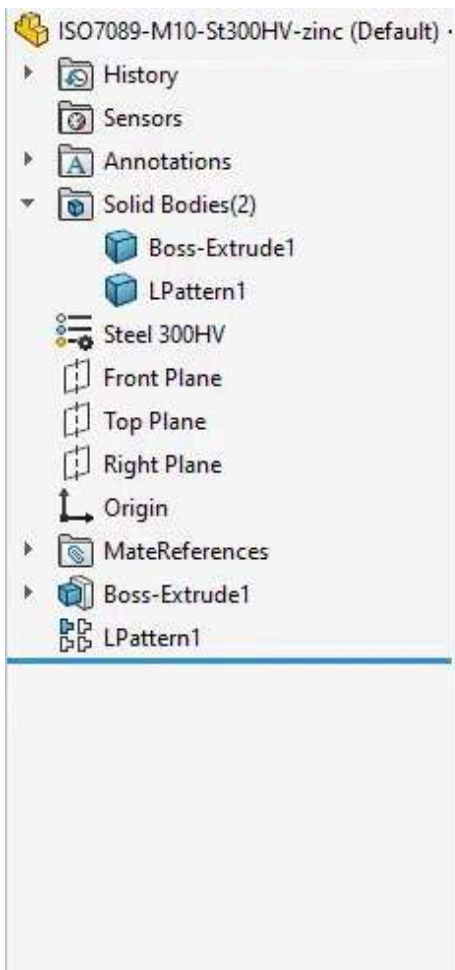
```

2.3. Custom properties for cut-list folders

If a part has **Weldment** or **Sheet Metal bodies**, SOLIDWORKS will group identical bodies in *cut list folders*.



Other body types are not grouped, even when they are identical.



Cut lists are tricky to work with and there are two methods for finding all cut list folders, so I wrote a separate section: [Custom properties for cut list folders](#).

3. What is the CustomPropertyManager?

3.1. I'd like to speak to the manager

Every big feature in SOLIDWORKS is a *manager*. The tabs above the feature tree are all managers:

1. FeatureManager
2. PropertyManager
3. ConfigurationManager
4. DimXpertManager
5. DisplayManager



3.2. CustomPropertyManager property accessors

We access custom properties in the API via the CustomPropertyManager. As you saw in Section 2, three SOLIDWORKS API objects have a CustomPropertyManager property:

- **ModelDoc2.Extension.CustomPropertyManager** (pass an empty string to access the file properties, pass a configuration name to access the configuration properties)
- **IConfiguration.CustomPropertyManager** (to access the configuration properties directly)
- **IFeature.CustomPropertyManager**

3.3. The most important methods and properties

These are the methods (+ the count property) you need most often:

1. Add3:

- Add a custom property with a certain type.
- When in doubt, create a Text property.
- Can overwrite an existing property.

2. Get6:

- Get the raw value ("SW-Mass@Part1.SLDPRT", including quotes) and *evaluated* value (123.12).
- Set UseCached to *false* to always get the latest data.

3. GetAll3:

- Get five arrays with the names, types, values, resolved values and whether they are **linked**.

4. GetNames:

- Get an array of all custom property names.
- Use a *variant* in VBA. If you are using C#, cast the return value to a string array.

5. Set2:

- Change the value of an existing property.
- Make sure it exists first, or check the return value to learn if it didn't exist.

6. Delete2:

- Delete a custom property by its name.

7. Count:

- Get the number of custom properties for this model, configuration or feature.

If the method has a number behind it, that means it is the nth version of that API. Older versions are not removed but have either a bug or fewer parameters. It's up to you to figure out which one it is, unfortunately.

4. How to read custom properties via the API

There are two methods for getting the value of a custom property:

1. To get a single custom property, call `Get6`
2. To get all custom properties, call `GetAll3`

I usually wrap both of these methods because I don't like them. `Get6` uses a bunch of *out* parameters for the type and the resolved value, but I prefer to have two explicit methods for getting the raw value and getting the resolved value. Just to avoid future mistakes.

`GetAll3` returns five arrays, hidden in objects, hidden in *out* parameters. Some of them are string arrays and the last one is a boolean array, hidden in an integer array, hidden in an object. It's pretty annoying.

Luckily, the open-source framework `SolidDNA` (that we currently maintain) has most of these methods wrapped by now. Read more below in [A user-friendly way: using SolidDNA](#).

5. How to write custom properties via the API

There are two ways for writing the value of a custom property:

1. `Set2`:
 - Is the default way of writing to an existing custom property.
 - Fails when the property does not exist yet.
 - Fails when the *type* is incorrect.
 - May fail when the property is linked.
2. `Add3`
 - Lets you add a custom property or overwrite an existing property.
 - The fourth parameter lets you choose whether to delete or overwrite an existing property. If you delete it, it ends up at the bottom of the list. If you overwrite it, it fails when the type is different. So choose this option carefully.

6. How to check if a custom property exists

There is no API to check if a custom property exists, unfortunately. So, there are two ways for figuring that out:

1. `Get6`:
 - Calling `Get6` will return an empty string when the property does not exist. But don't use that, because an empty existing property will also return an empty

string.

- Instead, use the return value, which will be 1 / `swCustomInfoGetResult_NotPresent` when the property does not exist.

2. `GetNames`

- Calling `GetNames` will return an array of custom property names. If your custom property name is in the list, the custom property exists.

You could also use `GetAll3`, but that just returns more data that you don't need when you only want to know if a property exists.

7. Data types for custom properties

7.1. Available data types

1. Text

- The most basic type, the most useful type, and the default. The maximum length is 1023 characters.

2. Date:

- Warning: dates are localized. That means they appear different and they depend on the system language of the PC that **created** the model. They're not ideal.

3. Number:

- These can be whole numbers (integers) or comma values (doubles). SOLIDWORKS seems to round and evaluate these value values and shows you the result.
- 24 becomes 24
- 24.0 becomes 24.0
- 24.000000000000001 becomes 24.000000

4. Yes / No

- You can pass "yes" or "Yes" as a string value, it's not case-sensitive. But strangely enough, it does display the case-sensitive actual value in the *Evaluated Value* column.
- Passing "1" or "true" for yes does not work. But for some reason, "false" or "0" (that's a zero) does work for no. I have just reported this as a bug.

Properties

Summary

Custom

Configuration Properties

Properties Summary

Delete

	Property Name	Type	Value / Text Expression	Evaluated Value
1	YesN	Yes or no	Yes v	yes
2	<Type a new property>			

7.2. Numbers and dates, represented by strings

I find it strange that you can only pass strings as values, even when the type is a number. What makes it even worse is that SOLIDWORKS *localizes* your numbers (and dates, for that matter), so it may use a comma or a dot as a separator. Windows, SOLIDWORKS and Excel all have their own separator setting, so this can go very wrong very easily.

This has created problems for me in the past. In the code that creates our [Fastener Models library](#), I diligently replace commas with dots before writing a number (coming from Excel, user input or somewhere else) to a custom property. And after creating the files, we have a separate tool that checks the separator in all custom properties.

Number and date properties were so annoying to work with that I now **only use text properties**.

8. Resolved values vs normal property values

8.1. What is the “resolved value”?

SOLIDWORKS supports a list of variables (see [All available variables for custom properties \(and cut lists\)](#) for the complete list) to be used in custom properties. You can enter these values manually, but some of them are also available from a dropdown. One example is the mass:

```
"SW-Mass@Part1.SLDPR1"
```

When the part names changes, the filename gets updated automatically. And on every rebuild (I assume), this value gets recalculated. The result is called the *resolved value* in code and the *evaluated value* in the user interface.

Properties

Summary Custom Configuration Properties Properties Summary

Delete

	Property Name	Type	Value / Text Expression	Evaluated Value		
1	Weight	Text	"SW-Mass@Part1.SLDPRT"	148.62	☐	
2					☐	

8.2. How to enter an equation or variable from code

Every link to a variable needs to be surrounded by double quotes " for it to work. But in code, those double quotes are already used for strings. So, we have to either *escape* these characters or enter them by their character number.

In VBA, you use chr(34) for the double quote character and you concatenate three strings into a single value:

```
1 customPropManager.Set2 "test", Chr(34) + "SW-Mass@Part1.SLDPRT" + Chr(34)
```

In C#, you escape the double quote character inside a string by adding a backslash before each one:

```
1 customPropManager.Set2("test", "\"SW-Mass@Part1.SLDPRT\"");
```

8.3. Custom property values in equations

You can use custom property values in equations and you can add the result of an equation to a custom property. But that has little to do with the API, so I added it to [section 13 of How to use custom properties in equations](#).

Properties

Summary Custom Configuration Properties Properties Summary

Delete

BOM quantity:

	Property Name	Type	Value / Text Expression	Evaluated Value
1	ten	Text	10	10
2	twenty	Text	"D1@Sketch1@Part1.SLDPRT"	20,00
3	<Type a new property			

Equations, Global Variables, and Dimensions





Filter All Fields



Default<As Machined>

Name	Value / Equation	Evaluates to	Comments
Global Variables			
Add global variable			
Features			
Add feature suppression			
Equations			
"D1@Sketch1"	= 2 * "ten"	20mm	
Add equation			

☒ Automatically rebuild
 Angular equation units: Degrees
 ☒ Automatic solve order

☐ Link to external file:

OK
 Cancel
 Import...
 Export...
 Help

9. Custom properties for cut list folders

In theory, every feature can contain custom properties. But in practice, only cut list folder features can have them.

There are two methods for finding all cut list folders: get all children of the *solid body folder* or traverse the entire feature tree and collect all cut list folders.

You only need the second method if you are working with *sub-weldments* because these features mess up your feature tree. I will always advise to not use sub-weldments because:

- They mess up the grouping of identical bodies in cut list folders. This makes ordering identical parts problematic.
- They create sub-folders inside cut list folders in the solid body folder. This makes working with the API harder.

9.1. Get all children of the “solid body folder”

We explain how to work with the FeatureManager / feature tree in part 3 of this blog series: [How to work with Features in the SOLIDWORKS API](#). To find the solid body folder (there is always only one) feature, we traverse the feature tree until we find it.

You might also be able to get the solid body folder by its name, but I suspect the feature to have a localized name in other *locales* so it's better to use the type name.

The next step is getting all children/subfeatures of the solid body folder. All of its children are cut list folders. Here's a full example to get all cut list folders (and their custom property count) in VBA:

```

1 Option Explicit
2
3 Sub main()
4     Dim swApp As SldWorks.SldWorks, swModel As ModelDoc2
5     Set swApp = Application.SldWorks
6     Set swModel = swApp.ActiveDoc
7
8     Dim solidBodyFolder As Feature
9     Set solidBodyFolder = GetSolidBodyFolder(swModel)
10
11     Dim subFeature As Feature
12     Set subFeature = solidBodyFolder.GetFirstSubFeature
13
14     While Not subFeature Is Nothing
15         Dim customPropManager As CustomPropertyManager
16         Set customPropManager = subFeature.CustomPropertyManager
17
18         Debug.Print customPropManager.Count
19
20         Set subFeature = subFeature.GetNextSubFeature
21     Wend
22 End Sub
23
24 Private Function GetSolidBodyFolder(swModel As ModelDoc2) As Feature
25     Dim swFeat As Feature
26     Set swFeat = swModel.IFirstFeature()
27     While Not swFeat Is Nothing
28         If swFeat.GetTypeName2 = "SolidBodyFolder" Then
29             Set GetSolidBodyFolder = swFeat
30             Exit Function
31         End If
32         Set swFeat = swFeat.IGetNextFeature
33     Wend
34 End Function

```

9.2. Find all cut list folders in the feature tree (use this if the tree contains sub-weldments)

If you have a need to go through the entire feature tree in a search for cut list folders, here's how to do that:

```

1 Sub GetCutListFeaturesFromTree()
2     Dim swApp As SldWorks.SldWorks, swModel As ModelDoc2
3     Set swApp = Application.SldWorks
4     Set swModel = swApp.ActiveDoc
5
6     Dim swFeat As Feature
7     Set swFeat = swModel.FirstFeature
8
9     While Not swFeat Is Nothing
10         If swFeat.GetTypeName2 = "CutListFolder" Then
11             Dim folder As BodyFolder
12             Set folder = swFeat.GetSpecificFeature2
13             Debug.Print "Folder type: " & folder.GetCutListType
14         End If
15
16         Set swFeat = swFeat.GetNextFeature
17     Wend
18 End Sub

```

10. A faster way: how to use the Document Manager

10.1. What is the Document Manager?

The Document Manager API lets you view and edit some SOLIDWORKS file properties without having to open the model. It even works when you don't have SOLIDWORKS installed.

You cannot use the Document Manager in macros though, so I use it in add-ins (to avoid opening the files) or for tools for users without SOLIDWORKS.

10.2. How fast is reading custom properties using the Document Manager?

It's hard to understate how much faster the Document Manager is compared to opening the models.

In our [fastener add-in Lightning](#), we read custom properties at a rate of around 100 files per second. I still need to investigate if we can spin up more threads to speed this up even further.

10.3. Algorithm for reading of writing custom properties

Follow these steps in a C# project to read a custom property from a file:

1. Request a Document Manager key:

1. Log into the [Customer Portal](#) > API Support > [Document Manager Key Request](#).
2. Fill in the form and.
3. Wait a few days for the key to arrive via email.
4. Add this key string to your code. It cannot be publicly readable, but to have it in a DLL is allowed.
5. If you want to support the latest SOLIDWORKS version, you need to request a new key every year.

2. Add an assembly reference to *SolidWorks.Interop.swdocumentmgr.dll*. Set *Embed Interop Types* to False.

3. Create an instance of the Document Manager application by creating a new factory, then [getting the application](#). Pass your key as a parameter.

4. Call [GetDocument](#) and cast the return value to [ISwDMDocument29](#) (or an older version if you are not using SOLIDWORKS 2023 DLLs yet) to access the file. Version 29 is currently the latest version of this object type, but I'm sure there will be more...

5. Get a custom property value by calling [GetCustomProperty2](#).

6. Set a property with **SetCustomProperty** (pre SW2023) or **SetCustomProperty2** (SW2023 and newer).
7. Call **Save** if you changed something. Changes are not permanent if you don't save.
8. Call **CloseDoc** to close the file and free up memory.

10.4. Sample code for the Document Manager

Here's how to read or write a custom property using the Document Manager API:

```
1 public static string GetCustomPropertyValue(string customPropertyName)
2 {
3     var classFactory = new SwDMClassFactory();
4     var application = classFactory.GetApplication("your document manager key here");
5
6     var document = (ISwDMDocument29) application.GetDocument("C:\\test\\part.sldprt", SwDmDoc
7     var currentValue = document.GetCustomProperty(customPropertyName, out var customProperty
8
9     document.CloseDoc();
10    return currentValue;
11 }
12
13 public static void SetCustomPropertyValue(string customPropertyName, string newValue)
14 {
15     var classFactory = new SwDMClassFactory();
16     var application = classFactory.GetApplication("your document manager key here");
17
18     var document = (ISwDMDocument29) application.GetDocument("C:\\test\\part.sldprt", SwDmDoc
19     document.SetCustomProperty(customPropertyName, newValue);
20
21     document.Save();
22     document.CloseDoc();
23 }
```

11. A user-friendly way: using SolidDNA

SolidDNA is an open-source framework to make the SOLIDWORKS API more modern and easier to work with. It was originally built by SOLIDWORKS legend Luke Malpass, but we took it over when he no longer had the time for it.

SolidDNA acts as a wrapper around the core SOLIDWORKS API, yet it still gives you access to the raw objects if there is no modern version of an API yet. If a SOLIDWORKS API topic is hard to understand or otherwise annoying, we create a more user-friendly version for it.

We have a few simple methods for custom properties:

```
1 var model = SolidWorksEnvironment.Application.ActiveModel;
2 model.GetCustomProperty("property name"); // Get a model property
3 model.GetCustomProperty("property name", "config name"); // Get a configuration property
4 model.GetCustomProperty("property name", "config name", true); // Get a resolved configuration property
5
6 model.SetCustomProperty("property name", "value"); // Set a model property
7 model.SetCustomProperty("property name", "value", "config name"); // Set a configuration property
8
```

```

9 var customPropertyEditor = model.Extension.CustomPropertyEditor("config name");
10 customPropertyEditor.CustomPropertyExists("property name"); // use the custom property editor

```

The **Model** and **ModelFeature** classes have a CustomPropertyEditor property, plus methods to get, set and delete custom properties. Use the optional parameters for the configuration name and whether to get the resolved value.

If you miss a useful method, feel free to send in a pull request on GitHub. All our products are based on SolidDNA and we are actively working on it.

12. How to write Summary Info

The first tab of the *file properties* window shows the Summary Info. Every Windows file can have these properties like the Author, Title and Subject.

The screenshot shows the 'Properties' dialog box with the 'Summary' tab selected. The fields are as follows:

- Author:** CAD Booster
- Keywords:** (empty)
- Comments:** Powered by CAD Booster - SOLIDWORKS automation
- Title:** ISO 7089 - M10 - St300HV - zinc
- Subject:** (empty)
- Statistics:**
 - Created: Wednesday, 7 December 2022 16:24:34
 - Last Saved: Wednesday, 7 December 2022 16:25:31
 - Last Saved By: peter
 - Last Saved With: SOLIDWORKS 2023

Buttons at the bottom: OK, Cancel, Help.

But you do not access them as you do with custom properties, you also don't get to choose the name of the field.

Instead, you use the **ModelDoc2.SummaryInfo** property and you pass the index of the field you want to get or set:

```

1 Option Explicit
2
3 Sub main()
4     Dim swApp As SldWorks.SldWorks, swModel As ModelDoc2
5     Set swApp = Application.SldWorks
6     Set swModel = swApp.ActiveDoc
7
8     swModel.SummaryInfo(swSummInfoField_e.swSumInfoTitle) = "Rocket"

```

```
9
10     Debug.Print swModel.SummaryInfo(swSummInfoField_e.swSumInfoAuthor)
11 End Sub
```

13. Products for batch processing custom properties

Now you know how to add, update custom properties via the SOLIDWORKS API.

But you don't have to reinvent the wheel, so here are some products that may already do what you are looking for:

- **CUSTOMTOOLS** by the Finnish software company ATR Soft. The product started as a batch custom property tool (hence the name), then grew much larger.
- **#TASK** by Australian reseller Central Innovation. #TASK (pronounced "sharp task") is a batch tool that lets you perform a list of scripts on a list of files.
- Our add-in **Lightning** makes your fastener library searchable by extracting fastener dimensions from custom properties.
- Our add-in **Drew** lets you batch-export models and drawings to STEP, PDF and more and you can use custom properties like the revision to build filenames.

All blog posts in this series

1. [The SOLIDWORKS Object Model + API explained](#)
2. [SOLIDWORKS API: the basics – SldWorks, ModelDoc2](#)
3. [How to work with Features](#)
4. [Persistent ID and sketch segment ID and more](#)
5. [All identifiers in the SOLIDWORKS API](#)
6. [About return values](#)
7. [Entities and GetCorresponding](#)
8. [How to work with selections](#)
9. [How to use custom properties](#)
10. [Understanding math, MathVector and MathTransform](#)
11. [Toolbars, menus and the Command Manager](#)
12. [How to create task panes and Property Manager Pages](#)
13. [How to create your own SOLIDWORKS add-in](#)
14. [Creating add-ins with SolidDNA: the basics](#)

CONTACT US

 +31 85 0653 776

 hello@cadbooster.com

PRODUCTS

Drew

Lightning

Fastener Models

Secrets to SW Performance

SolidDNA