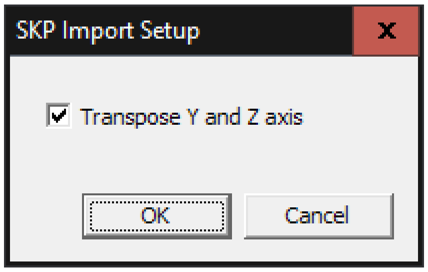
DesignCAD 3D Max 2016 Release Notes

July 26, 2016

Thanks for purchasing, or upgrading to, DesignCAD 3D Max 2016. Here's a detailed look at what's new in this version.

New Import/Export feature

SketchUp Import – DesignCAD 2016 can now import SketchUp drawings. SketchUp versions through SketchUp 2015 are currently supported. When importing SketchUp drawings, each separate solid object is imported as a distinct Solid Surface entity in DesignCAD.



The SketchUp Import Setup dialog includes a Transpose Y and Z option. When checked, this allows SketchUp objects imported into DesignCAD to appear "right side up". This option is provided because DesignCAD uses Y for the vertical axis, whereas SketchUp uses Z for the vertical axis. If this option is left unchecked, SketchUp imports will appear on their sides instead of upright.

Insert Manager

File/Insert Manager is a new command that allows the user to review and manage all blocks, symbols, and image files that are reference by or embedded in the drawing.

When the Insert Manager command is run, a dialog box appears with three tabs, one each for Blocks, Images, and Symbols.

Blocks

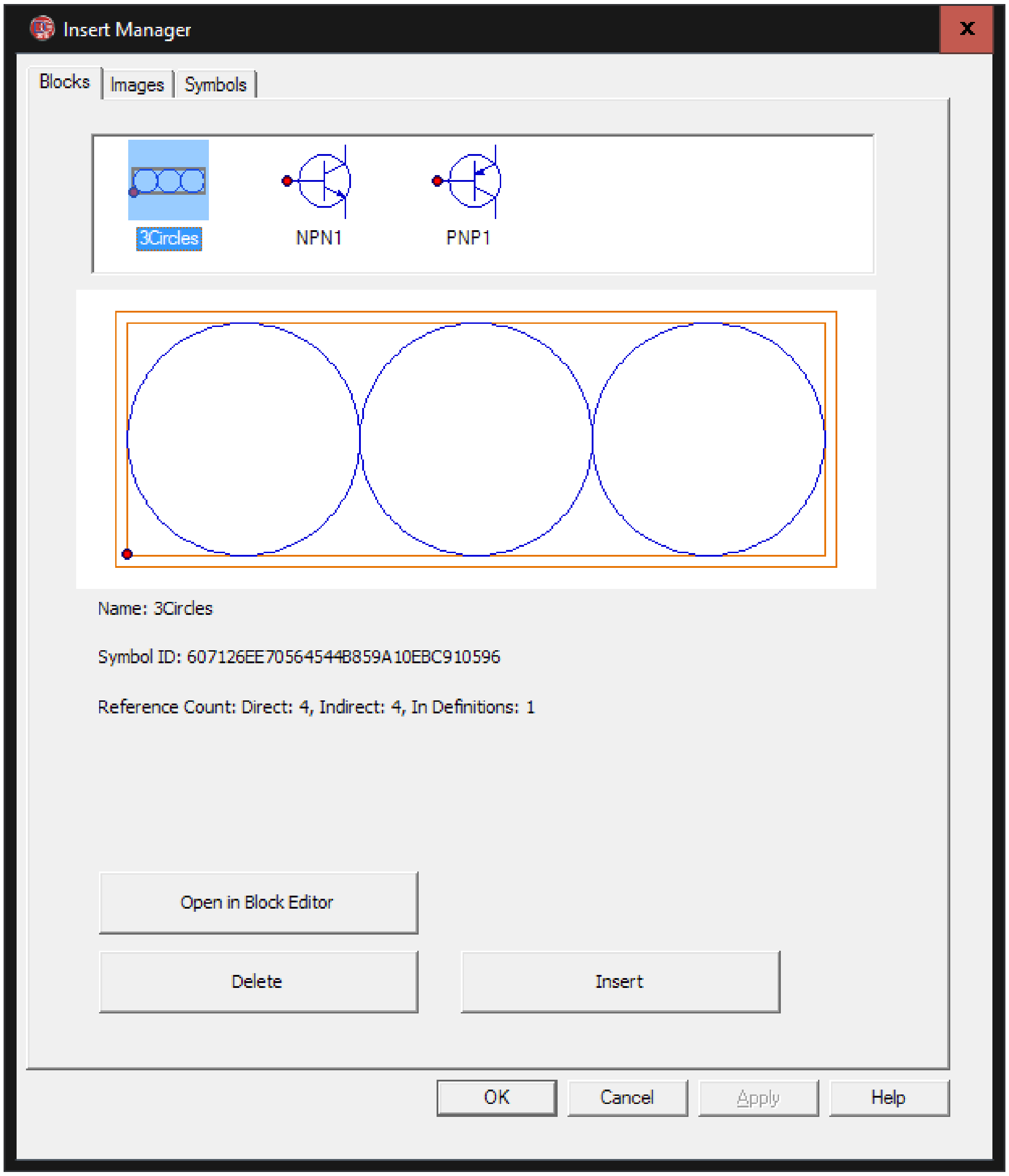
The **Blocks** tab shows a visual list of all blocks that are present in the drawing. When a single block is highlighted by clicking on it, a larger preview is displayed below the list. Beneath the larger view is more information about the block: its Name, ID, and a Reference Count.

Right-clicking on the list of symbol icons displays a context menu for sorting the blocks. Blocks may be sorted by Name, ID, or Reference Count.

Right-clicking on the **Name** line gives you the option of copying the name to the clipboard.

Right-clicking on the **Symbol ID** line gives you the option of copying the id to the clipboard (perhaps for use in a macro or COM program).

The **Reference Count** line shows three different values: -- **Direct**, **Indirect**, and **In Definitions**. The **Direct** count is the number of times the block is included in the drawing as a stand-alone block (i.e. not as part of another block or symbol). The **Indirect** count is how many times the block is included in other blocks or symbols that are included in the drawing. The **In Definitions** count is how many block or symbol definitions include the block as part of that definition.



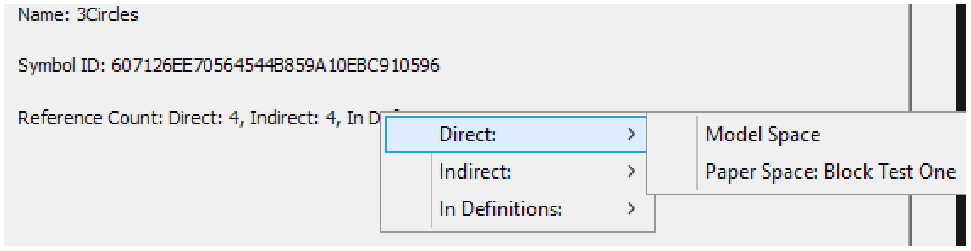
Name: 3Circles

Symbol ID: 607126EE70564544B859A10EBC910596

Reference Count: Direct: 4, Indirect: 4, In Definitions: 1

Right-click on this **References** line for a more detailed view of how the block is used; a context menu appears with the same categories: **Direct**, **Indirect**, and **In Definitions**.

Expanding the **Direct** category will indicate if the block is in use by the drawing's Model Space view and/or by one or more Paper Space layouts. Expanding the **Indirect** category will show which symbols and/or blocks are in active use that include this block. Expanding the **In Definitions** category will indicate which symbols and/or blocks include this block, whether or not there are any active references to those symbols/blocks.


Name: 3Circles

Symbol ID: 607126EE70564544B859A10EBC910596

Reference Count: Direct: 4, Indirect: 4, In D

The Blocks tab of the Insert Manager offers three editing operations: **Open in Block Editor**, **Delete**, and **Insert**.

Clicking on the **Insert** button will run the Block Insert command, and show the Block Insert dialog box with the highlighted block already selected.

Clicking on the **Delete** button will allow you to delete all instances of the highlighted block from the drawing, as well as the underlying block definition.

Clicking on the **Open in Block Editor** button will open a special view window with a limited menu and command list. While in the Block Editor, you can make almost any changes to the block definition that you desire, without the distraction of other drawing entities. See Block Editor below.

Images

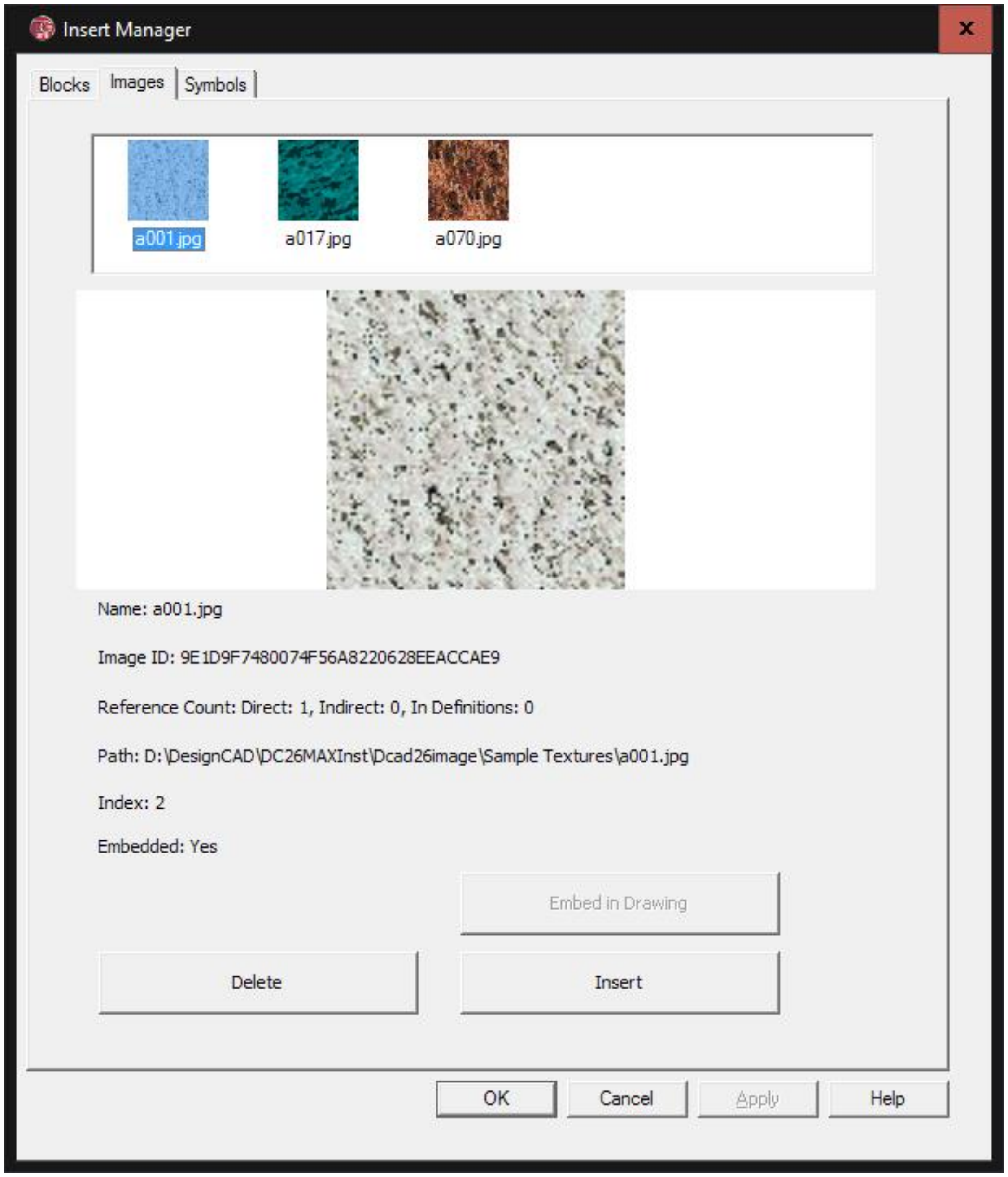
The **Images** tab of the Insert Manager is quite similar to the Blocks tab, showing all raster images that have been added to the drawing. Most of the options for Images are the same as for Blocks. Images also contain a **Path** field, an **Index** field, and an **Embedded** field, and these are additional sort options if you right-click on the thumbnail list at the top.

The **Path** field indicates the full path and filename of the original image. For embedded images, this original file may no longer exist in the referenced location. Right-clicking on the **Path** field allows you to copy the full path and filename to the clipboard for later reference.

The **Index** field indicates the index of the image in the list of images, i.e. the relative order in which the image was originally added to the drawing.

The **Embedded** field indicates whether the image is stored completely in the drawing (Yes) or is only a reference to an external file (No). *Blocks are always embedded, so this button is hidden on the Blocks page.*

If an Embedded image is highlighted in the visual list, the **Embed in Drawing** button becomes highlighted. Clicking on this button allows a referenced image to be embedded in the drawing, so that the external image file is no longer necessary. This also changes the **Embedded** field from Yes to No. Note that the original image file must be present in the specified path for this to work.



Name: a001.jpg

Image ID: 9E1D9F7480074F56A8220628EEACCAE9

Reference Count: Direct: 1, Indirect: 0, In Definitions: 0

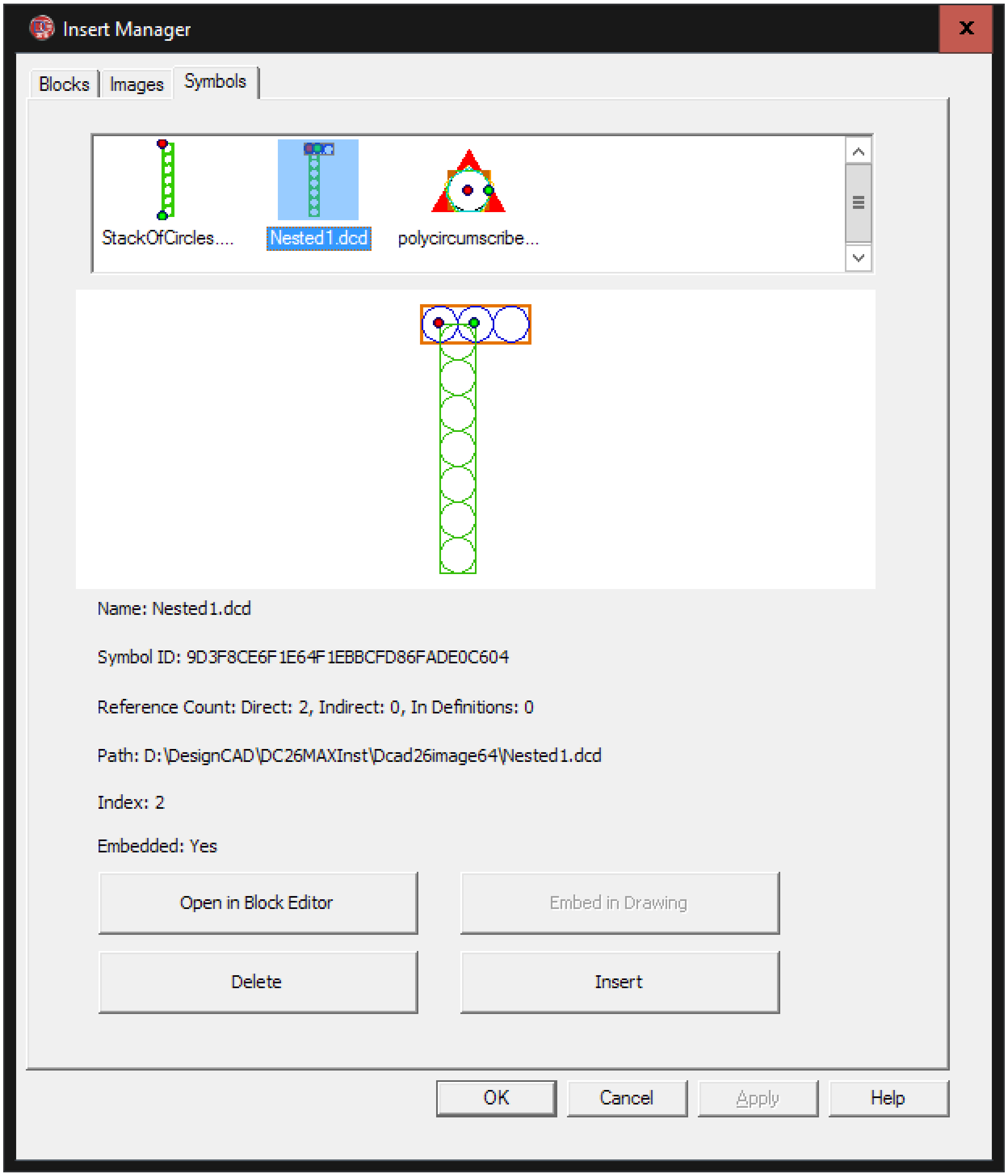
Path: D:\DesignCAD\DC26MAXInst\Dcad26image\Sample Textures\a001.jpg

Index: 2

Embedded: Yes

Symbols

The **Symbols** tab of the Insert Manager has the same features as the **Images** tab, and also has the **Open in Block Editor** button just like the **Blocks** tab.



Name: Nested1.dcd

Symbol ID: 9D3F8CE6F1E64F1EBBCFD86FADE0C604

Reference Count: Direct: 2, Indirect: 0, In Definitions: 0

Path: D:\DesignCAD\DC26MAXInst\Dcad26image64\Nested1.dcd

Index: 2

Embedded: Yes

Block Editor

When you highlight a symbol or block in the Insert Manager and click **Open in Block Editor**, the contents of the block or symbol are displayed in a separate window, with no other entities in view. This Block Editor mode has a somewhat limited set of drawing and editing commands; there is no access to loading symbols or blocks via the File menu, no print command, no options to save the symbol or block as an external file, and no access to special modes like Paper space Mode or Animation and Walkthrough Modes. There is also no access to the Help or Window menu commands, and the Options command has no effect. Aside from these exceptions, most other drawing and editing commands are available to you in the Block Editor.

If you need to add a symbol or block to one that you plan to edit, this cannot be done through the Block Editor's File menu. Instead, first select the block or symbol to be added in the main drawing window and copy it to the clipboard. Then you can open the symbol or block in the Block Editor and paste in the additional content. ***Care should be taken, however, not to paste a copy of a block or symbol into itself while it is being edited – this causes a circular reference, and may result in any number of difficult problems.***

To discard the changes made in the Block Editor, choose **File/Cancel Changes and Exit**. To save the changes made to a block or symbol in the Block Editor, choose **File/Save Changes and Exit**.

The Insert Manager and the Block Editor are available in both Model Space and Paper Space. However, you may find some blocks and symbols difficult to edit while in Paper Space Mode, due to the limited drawing area that is available in that mode. If you have difficulties editing a particular block or symbol in Paper Space Mode, try exiting to Model Space and editing it there.

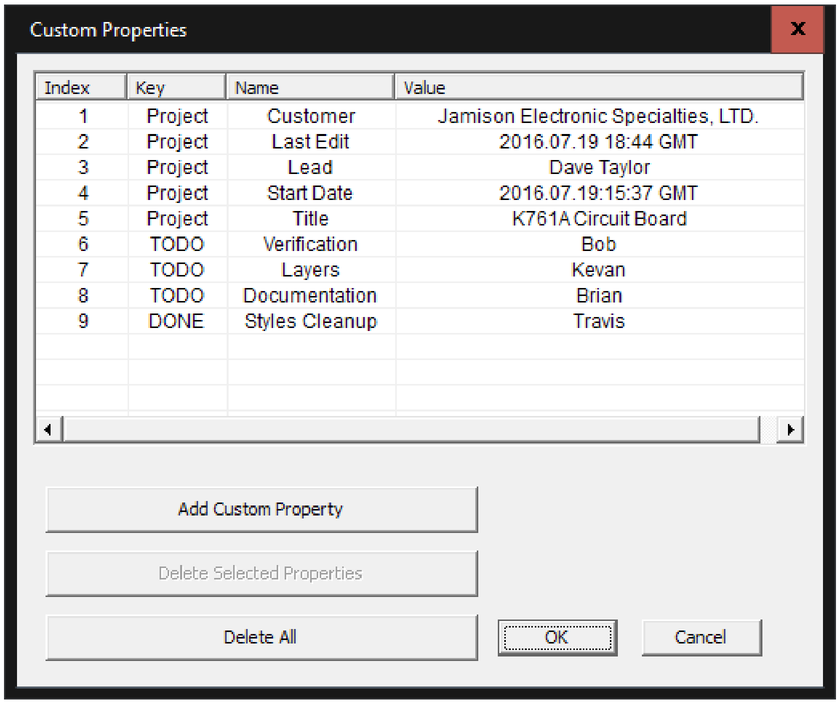
Custom Properties

DesignCAD now supports the addition of custom properties to drawings and to drawing entities.

Custom Properties are bits of text that are identified by a combination of a Key and a Name field. This is a means of adding "invisible" text information to a drawing or an object. These custom properties can be viewed and edited manually by means of a Custom Properties dialog, or processed by code via BasicCAD macros or OLE Automation programs.

Drawing Custom Properties are accessed via the **File/Document Custom Properties** command.

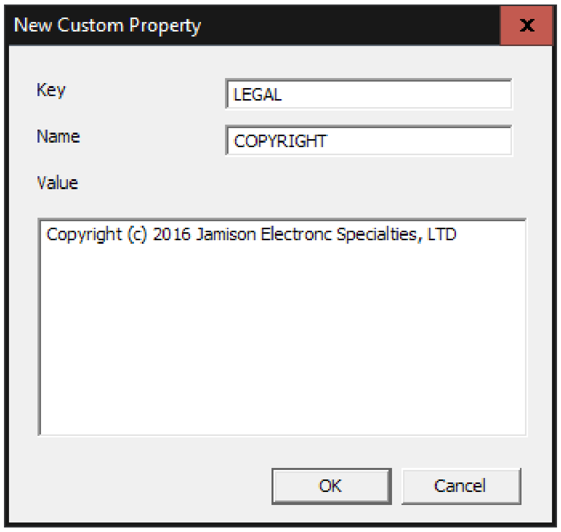
Entity Custom Properties are accessed via the **Custom Properties** button in the Info Box. Either method opens a Custom Properties editor:



Index	Key	Name	Value
1	Project	Customer	Jamison Electronic Specialties, LTD.
2	Project	Last Edit	2016.07.19 18:44 GMT
3	Project	Lead	Dave Taylor
4	Project	Start Date	2016.07.19:15:37 GMT
5	Project	Title	K761A Circuit Board
6	TODO	Verification	Bob
7	TODO	Layers	Kevan
8	TODO	Documentation	Brian
9	DONE	Styles Cleanup	Travis

The dialog shows four fields: Index, Key, Name, and Value. The Index is simply the order in which each custom property was added. The Key is one of two fields that uniquely identify each custom property. The Name field is the second field used to identify a custom property. The Value field is the "contents" of the custom property.

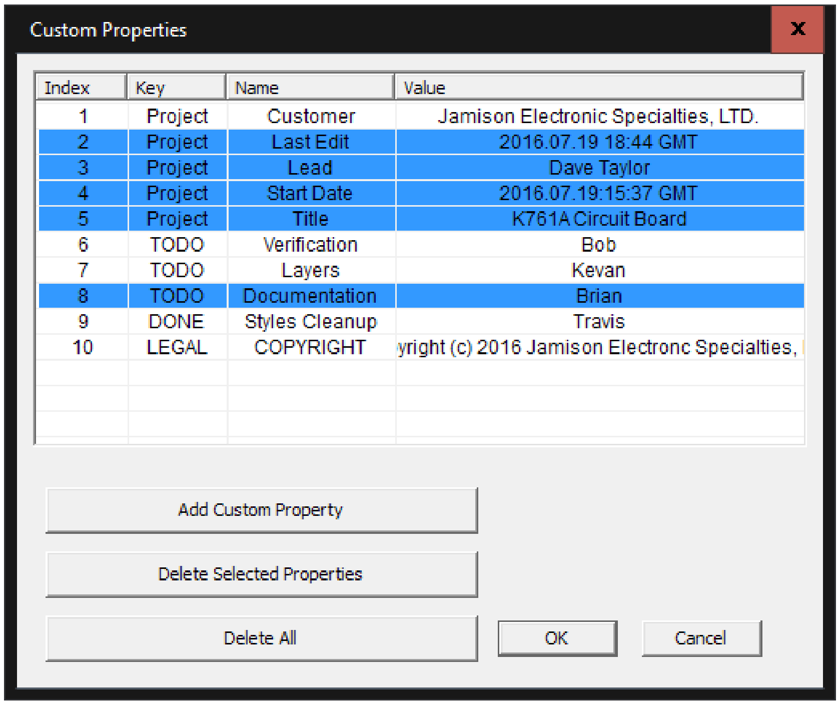
A new Custom Property may be added using the **Add Custom Property** button.



Key: LEGAL
Name: COPYRIGHT
Value: Copyright (c) 2016 Jamison Electronc Specialties, LTD

No two custom properties for a given drawing or drawing entity may have the same Key/Name combination. Two different drawings may have identical Key/Name pairs, and a drawing and one or more entities contained within it may have the same Key/Name pairs; the restriction only applies for the set of custom properties in use by a particular drawing or a particular drawing entity.

A single custom property can be selected by left-clicking on it. More can be selected using Ctrl-left-click or by Shift-left-click or Shift-down-arrow. Once one or more custom properties are selected in the Custom Properties editor, they can be deleted using the **Delete Selected Properties** button. The **Delete All** button will delete all custom properties that have been added via this editor.



Index	Key	Name	Value
1	Project	Customer	Jamison Electronic Specialties, LTD.
2	Project	Last Edit	2016.07.19 18:44 GMT
3	Project	Lead	Dave Taylor
4	Project	Start Date	2016.07.19:15:37 GMT
5	Project	Title	K761A Circuit Board
6	TODO	Verification	Bob
7	TODO	Layers	Kevan
8	TODO	Documentation	Brian
9	DONE	Styles Cleanup	Travis
10	LEGAL	COPYRIGHT	yright (c) 2016 Jamison Electronc Specialties,

Note, however, that some custom properties that were added via BasicCAD or OLE Automation may not be editable. Any such non-editable custom properties will have a gray background, and can only be edited or deleted using BasicCAD or OLE Automation methods.

Accessing custom properties via BasicCAD and OLE Automation is described below.

Custom Properties of drawing objects are duplicated when these objects are copied and pasted, duplicated, arrayed, or trimmed. (Note: any non-editable properties are NOT duplicated in this way.)

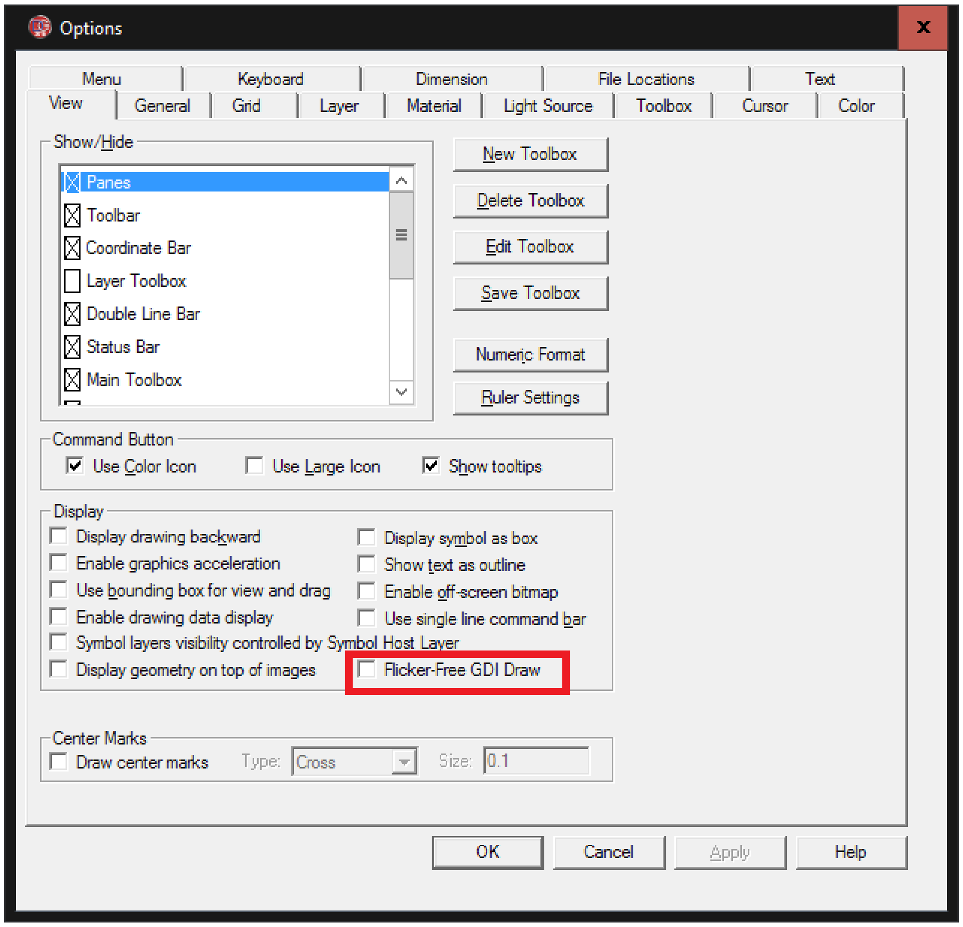
When you **Explode** a compound drawing object (such as a dimension or grid) that contains custom properties, the exploded pieces will no longer contain those properties. Similarly, **Combine Lines** causes the resulting object to lose any custom properties that may have been in its component pieces.

Custom Properties of drawings are saved with the drawing when you use **Save**, **Save As**, **Save a Copy**, or **Save as Symbol**; however, they are not transferred into a host drawing if you use **Symbol Load**.

Other Changes in DesignCAD 2016

Flicker-Free GDI Draw

A new option has been added to the View Options panel – Flicker-Free GDI Draw. When this option is selected, DesignCAD uses special optimizations to speed up the draw of wide lines in GDI wireframe views. This option has no effect when RedSDK Mode is active. To access this setting press the '**Q**' key to open the Options panel, and activate the 'View' tab. The checkbox is on the bottom right of the Display options group. The difference is particularly notable under Windows 7 if you have transparency enabled.



Options

Menu | Keyboard | Dimension | File Locations | Text
View | General | Grid | Layer | Material | Light Source | Toolbox | Cursor | Color

Show/Hide
- [✓] Panes
- [✓] Toolbar
- [✓] Coordinate Bar
- [ ] Layer Toolbox
- [✓] Double Line Bar
- [✓] Status Bar
- [✓] Main Toolbox

New Toolbox | Delete Toolbox | Edit Toolbox | Save Toolbox | Numeric Format | Ruler Settings

Command Button
- [✓] Use Color Icon
- [ ] Use Large Icon
- [✓] Show tooltips

Display
- [ ] Display drawing backward
- [ ] Enable graphics acceleration
- [ ] Use bounding box for view and drag
- [ ] Enable drawing data display
- [ ] Symbol layers visibility controlled by Symbol Host Layer
- [ ] Display geometry on top of images
- [ ] Display symbol as box
- [ ] Show text as outline
- [ ] Enable off-screen bitmap
- [ ] Use single line command bar
- [ ] Flicker-Free GDI Draw

Center Marks
- [ ] Draw center marks   Type: Cross   Size: 0.1

OK | Cancel | Apply | Help

General Fixes and Improvements:

- When exporting files to DWG or DXF format, sometimes angled text would be converted to horizontal text. This has been fixed.
- The Scan Image command sometimes was grayed out because DesignCAD couldn't find the necessary program. This has been fixed.
- When applying a material (such as Oak, Cherry, Pearl, etc.) to a Solid Surface entity, the material was not saved with the object when the drawing was saved.
- Save as 2D Projection was not working with Solid Surface entities.
- Working with multi-lines could sometimes crash DesignCAD.

BasicCAD Additions for Custom Properties

STATEMENT: ADDPROPERTY entid, key$, name$, value$, nocopy

Arguments:

- **entid** – a number. If entid is greater than zero it represents the index of the entity to which the new property is to be added. If entid is 0, the property will be added to the drawing instead.
- **key$ --** a string representing the key to be used in identifying this custom property.
- **name$** – a string representing the name to be used in identifying this custom property.
- **value$** – a string representing the contents of the property.
- **nocopy** – a flag that indicates whether this custom property should be propagated from the original entity via Duplicate, Copy/Paste, Array, or other editing operations. Also, if set, does not allow editing or deletion of this property via the Custom Property editor.

The ADDPROPERTY statement adds a new custom property to the specified entity if entid > 0, or to the drawing if entid = 0.

Errors:

- If the key/name pair clashes with an already-defined custom property, the statement triggers Error 114.
- If entid is non-zero and does not specify a legal entity id, the statement triggers Error 60.

Example 1:

```
' add a custom property to the first selected entity
PRECISION 0
ON ERROR GOTO oops
entid = GETSELECT 1, j
key$ = "PART"
name$ = "NUMBER"
value$ = "ZX_500_A12"
nocopy = 0 'false - this custom property can be edited by the drawing's user.
ADDPROPERTY entid, key$, name$, value$, nocopy
END

oops:
e = Error(q) ' get error number
MESSAGE "Error", e, "; ", Err$(e)
RETURN
```

Example 2:

```
' add a custom property to the drawing
entid = 0 ' specifies the drawing instead of an entity
key$ = "DRAWING"
name$ = "START DATE"
value$ = "06/25/2016"
nocopy = 0 'false - this custom property can be edited by the drawing's user.
ADDPROPERTY entid, key$, name$, value$, nocopy
```

STATEMENT: GETPROPERTY entid, propid, key$, name$, value$, nocopy

Arguments:

- **entid** – a number. If entid is greater than zero it represents the index of the entity from which the property is to be retrieved. If entid is 0, the property will be retrieved from the drawing instead.
- **propid** – a number specifying the index of the property to be retrieved. Indexing begins at 1, not 0.
- **key$ --** a string that will contain the key of this custom property.
- **name$** – a string representing the name of this custom property.
- **value$** – a string representing the contents of this custom property.
- **nocopy** – a flag that indicates whether this custom property can be propagated from the original entity via Duplicate, Copy/Paste, Array, or other editing operations. Also, if set, does not allow editing or deletion of this property via the Custom Property editor.

The GETPROPERTY statement retrieves a custom property from the specified entity if entid > 0, or from the drawing if entid = 0.

Errors:

- If entid is non-zero and does not specify a legal entity id, the statement triggers Error 60.
- If propid is not the index of an existing property, the statement triggers Error 25.

Example 1:

```
' get the third custom property from the first selected entity
e = 0
PRECISION 0
ON ERROR GOTO oops
entid = GETSELECT 1, j
propid = 3
key$ = "aaaa"
name$ = "bbbb"
value$ = "cccc"
nocopy = -1
GETPROPERTY entid, propid, key$, name$, value$, nocopy
IF e = 0 THEN
      MESSAGE "Property", propid, "   KEY:", key$, "; NAME:", name$, ";
VALUE: ", values$, "; nc: ", nocopy
END IF
END

oops:
e = Error(q) ' get error number
MESSAGE "Error", e, "; ", Err$(e)
RETURN
```

STATEMENT: SETPROPERTY entid, propid, key$, name$, value$, nocopy

Arguments:

- **entid** – a number. If entid is greater than zero it represents the index of the entity for which the property is to be changed. If entid is 0, the indexed drawing property will be changed instead.
- **propid** – a number specifying the index of the property to be changed. Indexing begins at 1, not 0.
- **key$ --** a string that will contain the new key of this custom property.
- **name$** – a string representing the new name of this custom property.
- **value$** – a string representing the new contents of this custom property.
- **nocopy** – a flag that indicates whether this custom property can be propagated from the original entity via Duplicate, Copy/Paste, Array, or other editing operations. Also, if set, does not allow editing or deletion of this property via the Custom Property editor.

The SETPROPERTY statement modifies a custom property from the specified entity if entid > 0, or from the drawing if entid = 0.

Errors:

- If entid is non-zero and does not specify a legal entity id, the statement triggers Error 60.
- If propid is not the index of an existing property, the statement triggers Error 25.
- If key$ and name$ would clash with an already-existing property, the statement triggers Error 114.

Example 1:

```
' Set the third custom property in the drawing
e = 0
PRECISION 0
ON ERROR GOTO oops
entid = 0 'we're targeting a drawing property instead of an entity property
propid = 3
key$ = "NewKey"
name$ = "NewName"
value$ = "I have changed."
nocopy = 1
SETPROPERTY entid, propid, key$, name$, value$, nocopy
IF e = 0 THEN
    GETPROPERTY entid, propid, nkey$, nname$, nvalue$, nnc
    MESSAGE "Updated Property", propid, "  KEY:", nkey$, "; NAME:", nname$, _
"; VALUE: ", nvalues$, "; nc: ", nnc
END IF
END

oops:
e = Error(q) ' get error number
MESSAGE "Error", e, "; ", Err$(e)
RETURN
```

STATEMENT: GETPROPERTYINDEX entid, name$, startindex, foundindex

Arguments:

- **entid** – a number. If entid is greater than zero it represents the index of the entity to be searched. If entid is 0, the drawing properties will be searched instead.
- **name$** – a string representing the custom property name to be searched for.
- **startindex** – a number representing which property index to start at.
- **foundindex** – a number representing the index of the next property that matches the specified name. Set to -1 if no matching property is found.

The GETPROPERTYINDEX statement searches the custom properties of the drawing (if entid is 0), or of the specified entity (if entid > 0) for a name field matching name$. Set startindex to 1 to begin searching at the first custom property. If searching for multiple matches (which is possible, since there may be two or more properties with different key fields but the same name field), search for the second and subsequent matches by adding 1 to the last foundindex.

Errors:

- If entid is non-zero and does not specify a legal entity id, the statement triggers Error 60.
- If startindex is not the index of an existing property, the statement triggers Error 25.

Example 1:

```
' Search the drawing for all custom properties having the name "PARTNO"
entid = 0
name$ = "PARTNO"
startindex = 1
foundindex = 0
PRECISION 0
ON ERROR GOTO oops
DO WHILE foundindex >= 0
    GETPROPERTYINDEX entid, name$, startindex, foundindex
    IF e <> 0 THEN GOTO finish
    IF foundindex <> -1 Then
        MESSAGE  "Match at index", foundindex
        Startindex = foundindex + 1
    END IF
LOOP
finish:
END

oops:
e = ERR(q)
RESUME NEXT
```

STATEMENT: DELETEPROPERTY entid, propid

Arguments:

- **entid** – a number. If entid is greater than zero it represents the index of the entity from which a custom property is to be deleted. If entid is 0, a custom property will be deleted from the drawing properties instead.
- **propid** – the index of the custom property to be deleted.

The DELETEPROPERTY statement deletes the custom property specified by propid from the entity specified by entid, or from the drawing if entid is zero.

Errors:

- If entid is non-zero and does not specify a legal entity id, the statement triggers Error 60.
- If propid is not the index of an existing property, the statement triggers Error 25.

Example 1:

```
' Delete the fourth custom property from entity 2
entid = 2
propid = 4
DELETEPROPERTY entid, propid
```

New Sys() functions relating to Custom Properties

Sys(689) – the number of custom properties that the current drawing contains.

Sys(690) – the number of custom properties contained in the entity last loaded by the ENTITY statement.

New BasicCAD Commands and Parameters For Import/Export

STL SUPPORT

>StlOut

```
{
 <AsBinary [a]  '[0 = text, 1 = binary]
 <Oriented [o]  '[0=no, 1 = yes]
 <Filename [f$]  ' name and path of file to be exported
 <NULL       ' optional -- if used, only updates Setup options for STL export to match the parameters
             ' supplied in the macro
}
```

>StlIn

```
{
  <Filename [f$] ' name and path of file to be imported'
}
```

SKETCHUP SUPPORT

>SkpOut

```
{
  <Type [t] '3 = Sketchup 3.0, 4 = 4.0,  5= 5.0, 6 = 6.0, 7 = SketchUp 7.0, ...
        '8 = SketchUp 8, 9 = 2013, 10 = 2014, 11 = 2015.
        ' Numbers less than 3 default to SketchUp 3, and values greater
        ' than 11 match SketchUP 2015. (This upper end will 'slide' as newer
        ' version of SketchUp are supported)
  <TransposeYZ [tx] '0 = no, 1 = yes
  <Filename [f$]
  <NULL   'optional -- if used only updates Setup options for SketchUp export
}
```

>SkpIn

```
{
  <TransposeYZ [tx] ‘[0=no, 1=yes]
  <Filename [f$]
  <NULL   'optional -- if used, only updates Setup options for SketchUp import
}
```

OBJ SUPPORT

>ObjOut

```
{
  <Triangulate [tr] '[0 = no, 1 = yes] for triangulation of Solid Surfaces
  <Filename [f$]
  <NULL   'optional -- if used, only updates Obj export Setup settings
}
```

>ObjIn

```
{
  Filename [f$]
}
```

OLE Automation Classes, Methods, and Properties for Custom Properties

CustomProperties Class – a collection of Custom Property objects

Methods:

AddProperty(Key$, Name$, Value$, bNoCopy) as Integer
Returns the index of the newly-added property if successful, or -1 in case of failure.

DeleteProperty(iIndex) as Boolean
Returns True if successful or False if the property could not be deleted for any reason.

Item(iIndex) as CustomProperty
Returns the CustomProperty specified by the index.

Properties:

Count as Long
Returns the number of custom properties in the collection

CustomProperty Class – individual Custom Property objects

Methods:

SetKey(Key$) as Boolean
Sets the Custom Property's Key to the value specified in Key$. Returns False if the key could not be updated to the specified value (typically due to a key/name clash that would result).

SetName(Name$) as Boolean
Sets the Custom Property's Name to the value specified in Name$. Returns False if the Name could not be updated to the specified value (typically due to a key/name clash that would result).

Properties:

Key as String (read-only)
Returns the Key for the Custom Property. Read-only. To change the contents of the Key property, use the SetKey method.

Name as String (read-only)
Returns the Name for the Custom Property. Read-only. To change the contents of the Name property, use the SetName method.

Data as String (read/write)
Returns the Data contents for the Custom Property. Can be directly read and written.

NoCopy as Boolean
Returns the NoCopy field of the Custom Property. Can be set directly read and written.