Developer Documentation

Developer Documentation

The PlanSwift API provides developers the documentation (much of which is "coming soon") on PlanSwift that will provide them the tools needed to develop and link applications to PlanSwift.

Before working with the API, a good understanding of the internal structure is vital and will require an Under-The-Hood (U-T-H) tab to be enabled.

CAUTION

By modifying or changing anything in the back end, you may adversely affect the operation of PlanSwift. Modifications should be done in a read-only mode. If any modifications are done to the back end, those modifications will be lost when the application is re-installed.

CAUTION

Unless you really must do this, we recommend that you do not proceed.

PlanSwift does not provide Technical Support for this.

You are on your own!

Use at your own risk!

Should you choose to proceed anyway, you will need a password to go "under the hood" that can be obtained from your PlanSwift representative or by sending an email to takeoff@constructconnect.com.

API Documentation

API Documentation

This API documentation provides the development information needed to connect PlanSwift to other applications. It includes a description of the internal structure of the API, the object structure of the API, and how to access PlanSwift's Under-The-Hood section; it describes the root properties and settings hierarchy of the application commands (including code examples in C#, Delphi, VB/VBA / OLE, Pascal Scripting, and Pascal Scripting OLE), settings hierarchy, sections on connecting with OLE and COM, the developer documents section, and the API Reference section, including coding examples for COM Object Models and Scripting.

PlanSwift does not provide technical support for using the API.

Section Contents

- > PlanSwift Structure
- Connecting to PlanSwiff
- > Developer Documents
- > API COM Reference

Related articles

Filter by label
There are no items with the selected labels at this time.

PlanSwift Structure

PlanSwift Structure

Going "Under-The-Hood" allows the programmer to link to PlanSwift software to develop applications that work with PlanSwift. Under-The-Hood provides a programmer the capability to link to PlanSwift software to develop applications that work with PlanSwift.

Overview of Under-The-Hood

Any development environment that supports COM, such as: C#, Visual Basic, Delphi, Java, etc., can utilize the PlanSwift API. Developers can use whatever Integrated Development Environment (IDE) they would like to. With managed code, you you don't have to worry about freeing up PlanSwift (removing it from memory). When programming in unmanaged code, such as Delphi, you will have to free up PlanSwift.

There are two ways to connect to PlanSwift: one is Early Binding, and the other is Late Binding. Early Binding allows you access to Types: everything is pre-loaded into the proper structure. Late Binding means that everything has already been executed within code; you just want to hook into that via Object Linking and Embedding (OLE) and listen for those events, so you do not get the same functionality as in COM with its pre-loaded Types and Items. Early Binding is preferable because it is a lot easier to use Early Binding with Types versus Late Binding and having to do a lot of guesswork.

Obtaining the Under-The-Hood password

To obtain the Under-The-Hood password, please contact your PlanSwift representative or send an email to takeoff@constructconnect.com. Section

Contents

- Accessing Under-The-Hood (U-T-H)
- Root Properties
- Settings Hierarchy
- Item Structure Overview
- Job
- Storages
- Plugins (2)
- _Units
- _Types
- Types
- Lists
- Reports
- Developers
- Hatches
- Estimating
- Textures

Related articles

Filter by label

There are no items with the selected labels at this time.

Accessing Under-The-Hood (U-T-H)

Accessing Under-The-Hood (U-T-H)

Before working with the API, a good understanding of the internal structure is recommended. To review the structure, the Under-The-Hood (U-T-H) tab needs to be enabled. This section describes how to enable the U-T-H tab in order to access the internal structure of PlanSwift.

By modifying or changing anything in the back end, you may adversely affect the operation of the application. Modifications should be done in a read-only mode. If any modifications are done to the back end, those modifications will be lost when the application is re-installed.

Follow these steps to enable the Under-The-Hood U-T-H tab.

- 1. Open PlanSwift.
- 2. Click on ${\bf Settings}$ along the top ribbon bar (see $\,\#{\bf 1}$ on Figure 1 below).
- 3. Select Interface from the options on the left (see #2 on Figure 1 below).
- 4. Click on **Show Under the Hood Screen**(see #3 on Figure 1 below).

	ew Estimating Lists Templates Settings Reports Help Plugins Search Undo
License Manager - Licensing Data Storage General - Company	Interface Settings
- Company - Keybooard Hotkeys - Interface - Zoom / Pan - Arnotations - Graphics - Takeoff Tools - Snapping - Notifications - Digitizer Tablet - Property Groups - Advanced	Color Scheme: Blue 3 Show Under The Hood Screen 3 Show Types Tab in Templates Screen Show Full Screen Cursor Draw Cross Hairs Cursor on Image Large Scroll Bars Show Flying Tool Hints Show Devr Inits Show Devr Tools on Plugins Tab Show Browse For Job
	 Disable Job Locking (requires Username when active). Show RepotDesigner Disable Estimage Grid Overtyping Show Auto Count Tool Enable Extended DPI Range for TIFs

Figure 1

- 5. For the password, please contact your PlanSwift representative or send an email to takeoff@constructconnect.com. Enter the password (see number #4 of Figure 1) and click on Ok.
- 6. An U-T-H (for "Under-the-Hood") tab now appears on the top ribbon bar (see the red arrow in Figure 2). Click on U-T-H.

														/	1			
The second	Home	Page	Tools	View	Estimatin	ng Lists	Ten	nplates	Setting	s Re	ports	Help	U-T-H	Plugins	Searc	h	Undo	
۲	10	2		4		Q	÷,	Ð,	Q	Ð	ν.			¢	Ŀr.	R	:-	*
New	Open	Print	Email Job	Back	Fwd	Zoom	Fit Page	Zoom	Zoom Out	Pan	Scale	Dim	ension	Area	Linear	Segment	Count	Auto Count
		lob		Nav	/igate		Z	oom / Pai	n			Measur	e				Takeoff	

Figure 2

7. Clicking on U-T-H tab displays the Under-The-Hood (U-T-H) hierarchy (Figure 3). PlanSwift is the root, or the parent object. Each of the folders beneath PlanSwift is a child of PlanSwift.

	Ног	ne Page Tools View Estimating Lists Templates Settings Reports Help U-T-H
Prope	P ertie	s Copy Paste Delete Refresh Bulk Update Browse to Properties PlanSwift Folder Developers
Name	_	
	Contraction of the local division of the loc	nSwift Settings
Đ	0	Job
Ŧ	0	Storages
	-	Plugins
Đ	á	Units
		_Types
	-	Types
ŧ	-	Lists
Đ		Reports
Ŧ	(A)	Developers
Ŧ	0	Hatches
(±	0	Estimating
1000	0	Textures

Figure 3

Root Properties

Root Properties

This section describes how to access the advanced Root Object Properties window.

Double-click on PlanSwift (see red arrow of Figure 1) to open the Advanced Properties window for the root object. Simple

Description

API Call: planswift

planswift.root

Use "\" (without the quotes) to access the root object.

opertie	es Copy Paste Delete	Refresh Bulk Update Browse to Properties PlanSwift Folder Custom 1	* • •	Doc Editor Developers	corruption/loss, and PlanSwift will not be		or inco	rrect data.
		Custom 1		pevelopeis		warning /	UISCIDII	ner
ne 😑 Pia	anSwift							
e Ø								
Ð	± + 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1	· - * / () () · 🗟 🛍	* * 5	Show: No	rmal 🛛 🖪 Form Lavout	Events -		
ŧ	Name	Formula	Input Units	Adjust	Result	Output Units		
	Item	50 				10.5	221 132	
Đ	Name						B (2
	Icon	\			٠		8	
	Other							
201	Folder	[!EXEPath]\Data			C:\Program Files (x86)			
-	The second se	The survey of the second		1		44	10	

Figure 1

Settings Hierarchy

Settings Hierarchy

Settings are the root setting for PlanSwift. They are the default configuration settings for PlanSwift and can be written, read, and modified. Custom settings can also be added. This section provides a list of these settings and coding examples in C#, Delph, VB / VBA OLE, and Scripting of how to access the settings.

To obtain the Advanced Properties of Settings, double click on Settings (red arrow in Figure 1).

) Ho	me	Page	Tools	View	Estimatin	ng L
They		6	-	×	5	
				~	4	
Properti	es Ci	ору	Paste	Delete	Refresh	Bulk
					(T	Prop
					Custor	n ı
Name						
	anSwift					
🖻 🤪 Se	ttings	-	_		_	
C) Open	Job Dia	alog			
D-C	Recen	nt Jobs				
	Bl	ake Re	sidence			
	Sa	ample P	lan			
	10	004				
-	Notes					
) Show	Me Ho	N			
	🐨 Er	mail Job)			
	T Ne	ew Job				
		pen Joł	5			
	Pr	int				
	T So	2.542.01				
	T Pr	opertie	5			
	-	onfirmD				
		atch Re	name			
	1.103	verlay				
		dd Page	es			
1. State 1.	Dockir					
13 81	0					
) Script					
	Repor					
	New J					
12	Prope	and the second states	alog			
) Estima					
F	Add P	age Dia	alog			
E Jo	D					

See Figure 2 for the Settings Advanced Properties window. Note that you may or may not have some of these property values in your version of the software. API calls for each of these are covered in alphabetical order.

	- * / () () - 🖓 🖺		Show: No	Lavout	Events -	· · ·		
	Formula	Input Units	Adjust	Result	Output Units			
em	ŵ			eta i		0	0	_
Icon	Q			0		8	¥	
Name	Settings			Settings		ð	Ŷ	
her			1				0	_
AngleSnapHotKey	119			119		ß	Ŷ	
AutoSelectFirstPage	X			True		ß	Ŷ	
AutoUpdate	V			True		₽	Ŷ	
Color Scheme	Blue			Blue		ß	Ş	
ContinueWithHotkey	0			0		B	9	
CreateBookmarkHotkey	66			66		ß	Q	
CustomerNum	dave.hansen@planswift.com			dave.hansen@planswif		ß	9	
Default Measurement Type	English			English		ß	Q	
DefaultAreaTransparency	100			100		ß	0	
DefaultCountTransparency	100			100		8	Ŷ	
DefaultCurrency	\$			\$		8	Ş	
	100			100		8	-	
DefaultLinearTransparency							Q	_
DefaultNoteTransparency	105			105		8		
DefaultSegmentTransparency	100			100		ß	9	
DigitizerSnap	Y			True		ß	Ş	
DigitizerSnapHotKey	114			114		₽	9	
EnhancedImage				True		ß	Q	
FlyingToolHints				True		₽	9	
FullScreenCursor	¥			True		ß	Ŷ	
Height	822			822		8	0	
Hide Types Tab				False		8	Ŷ	
InstallGUID	{ACD46CF9-B796-4BEA-9398-60733AA63EBC}			{ACD46CF9-B796-48E/		8	Ŷ	
JumpLastView			-	False		8	Ŷ	
							-	
Language	English			English		ß		
LastReportValidUntil	60 days			60 days		ß	9	
Left	-1905			-1905		₽	Ŷ	
MagnifierHotKey	77			77		ß	Q	
MeasurementEntry	English			English		S	Ŷ	
NewAreaHotKey	49			49		ß	9	
NewCountHotKey	52			52		ß	0	
NewFromTemplateHotKey	84			84		8	Ŷ	
NewLinearHotKey	50			50		8	0	
NewNoteHotKey	187			187		ß	Ŷ	
	78			78		6		
NewSectionHotKey							9	
NewSegmentHotKey	51			51		ß	Ŷ	
NoAskBreakInheritance				False		₽	Ŷ	
NoAskCopyPoints				False		₽	Ŷ	
NodeSize	4			4		ß	Q	
Ortho	×			True		ß	0	
PIN	PatioFive50			PatioFive50		8	Ŷ	
PageNameFormula	[Title][-][Pg#]			[Title][-][Pg#]		8	Ŷ	
PanHoverSpeed	44			44		8	Ŷ	
PanZoomHover				True		<u> </u>	▼	
							*	_
PanZoomHoverDelay	500			500		8	9	
PanZoomHoverTransparency	90			90		ß	9	
PanspeedKeyboard	125			125.00		ß	Ŷ	
PointSize	10			10		₽	Ŷ	
PropertiesWindowHeight	952			952		ß	Ŷ	
PropertiesWindowLeft	410			410		ß	Ŷ	
PropertiesWindowTop	49			49		ß	Ŷ	
PropertiesWindowWidth	883			883		8	0	
Property Groups	Item			Item		8	Ŷ	
REPORTDESIGNER				True		8	Ŷ	
							-	_
RecordHotKey	82			82		ß	9	
ReportValidUntil	30 Days			30 Days		ß	9	
SalesTax1	0	%		0.00	%	ß	Ŷ	
SalesTax2	0	%		0.00	%	ß	Ŷ	
ScrollDownHotkey	68			68		ß	Q	
ScrollLeftHotkey	83		1	83		8	0	

scrolikightHotkey	70	20	₫" ∀ ⊔]
ScrollUpHotkey	69	69	🗗 💡 🗅
SendErrorReport		False	
SendScreenshot		False	🗗 💛 🗅
Show Under The Hood Screen	True	True	
ShowWelcome		False	🗳 🔶 🗆
SmartOrtho	V	True	
FoggleImageEstimatingHotKey	123	123	🗳 😔 🗆
FoggleOrthoHotkey	72	72.00	♀ ⊡
Гор	8	8	🗗 💛 🗅
Width	1534	1534	
WindowState	1	1	
ZoomHotkey	121	121	🗳 😔 🗆
ZoomHoverSpeed	40	40	♀ □_
ZoomInHotKey	174	174	♀ ⊡
ZoomOutHotKey	109	109	♀ ⊡
ZoomSpeed	5	5.00	🗗 💡 🗅
ZoomSpeedKeyboard	50	50.00	🗗 💛 🗅
ZoomToFitHotkey	118	118	€
eritance path:	1		
	orm		Ok

Figure 2

API Calls

- Delphi
 C#
 VB/VBA (OLE)
 Scripting

Delphi

1	
2 P	procedure main;
3 V	
4	planswift: IPlanSwift;
5	settings: IItem;
6 b	
7	<pre>planswift := coPlanSwift Create();</pre>
8	<pre>settings := planswift.getItem('\Settings');</pre>

C#

Using Iltem Object Model							
1							
2 Pr	ivate void Main()						
3 {							
4	PlanSwift planswift = new PlanSwift();						
5	IItem settings = planswift.GetItem(@"\Settings");						
}							

VB/VBA (OLE)

Usin	ng litem Object Model
1	
2	Sub main()
3	Dim planswift = CreateObject("PlanSwift9.PlanCenter")
4	Dim settings = planswift.GetItem("\Settings")
-	End Sub



- settings := PlanSwift.getItem('\Settings'
- e

AllowExtenderDPI

AllowExtenderDPI

Boolean value that enables or disables extended DPI Range for TIFFs. Checked is true and enables it; unchecked is false and disables it. Figure 1 shows where this setting is controlled in the Settings / Interface / Interface Settings window.

Figure 1

API Calls

- Delphi
- C#
- VB/VBA (OLE)
- Pascal Scripting
- Pascal Scripting (OLE)

Delphi

Using Iltem Object Model						
1						
2	procedure main;					
3	var					
4	planswift: IPlanSwift;					
5	settings: IItem;					
6	property: IPropertyObject					
7	begin					
8	<pre>planswift := coPlanSwift Create();</pre>					
9	<pre>settings := planswift getItem('\Settings');</pre>					
10	<pre>property := planswift .GetProperty('AllowExtenderDPI');</pre>					
11	WriteLn (property ResultAsBoolean())					
	end					

Using PlanSwift Object Model

1

- 2 //or
- 3 procedure
- 4 van
- 5 planswift: IPlanSwift;
- 6 property:IPropertyObject;
- 7 begin planswift := coPlanSwift .Create(
- 8 property := planswift.GetProperty('\Settings','AllowExtenderDPI');
- 9 WriteLn (property.ResultAsBoolean()

end;

C#

Usin	g Iltem Object Model
1	
2	private void Main()
3	
4	PlanSwift planswift = new PlanSwift();
5	IItem settings = planswift.GetItem(@"\Settings");
6	IPropertyObject property = settings.GetProperty("AllowExtenderDPI");
7	console.WriteLn(property.ResultAsBoolean())

1 2

- 3
- 4 5
- 6
- 7

VB/VBA (OLE)

Using IItem Object Model	
1 2 Sub main()	
<pre>Dim planswift = CreateObject("PlanSwift9.PlanCenter")</pre>	
<pre>4 Dim settings = planswift.GetItem("\Settings")</pre>	
5 Dim property = settings.GetProperty("AllowExtenderDPI")	
<pre>6 Console.WriteLn(property.ResultAsBoolean()); End Sub</pre>	

Using PlanSwift Object Model

1	
2	Main()
3	<pre>Dim planswift = CreateObject("PlanSwift9.PlanCenter")</pre>
4	<pre>Dim nameProperty = planswift.GetProperty("\Settings", "AllowExtenderDPI")</pre>
5	Console.WriteLn(property.ResultAsBoolean)

Pascal Scripting

Using Item	n Object Model
1	
2 beg	gin
3	<pre>settings := PlanSwift.getItem('\Settings');</pre>
4	<pre>property := settings.GetProperty('AllowExtenderDPI');</pre>
5	ShowMessage (property.ResultAsBoolean);
end	a
5	ShowMessage(property.ResultAsBoolean);

Using the PlanSwift Object Model

1	
2	begin
3	<pre>property := PlanSwift.GetProperty('\Settings','AllowExtenderDPI');</pre>
4	ShowMessage(property.ResultAsBoolean);
	end

Pascal Scripting (OLE)

Using	g Item Object Model
1	
2	begin
3	<pre>settings := getItem('\Settings');</pre>
4	ShowMessage(GetResultAsBoolean(settings, 'AllowExtenderDPI'));
	end

Root Object Model	
<pre>1 2 begin 3 ShowMessage(GetResultAsBoolean('\Settings','AllowExtenderDPI')); end</pre>	
	i ,

AngleSnapHotKey

AngleSnapHotKey

Integer value that returns an ANSI key code (default code 72, key H). Figure 1 shows where the AngleSnapHotKey assignment is made. Ortho Snap (another name for Angle Snap) is also controlled in the PlanSwift window (Figure 2) and in the Settings Window (Figure 3).

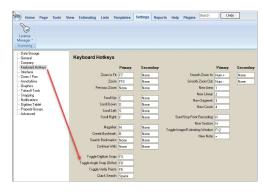


Figure 1



Figure 2

License Manager * Licensing	
Data Storage General Company Keyboard Holkeys Interface Annotations Graphics Takeolf Tools Sinapping Notications Dipitice Tablet Property Groups Advanced	Snapping Enable Snapping to points Snap to PlanSwift points Snap to CAD points Hover to Snap Avis Ottho - Snap to angles (hold Shift for temporary enable/disable Smart Ortho - Snap only when close to angles

Figure 3

API Calls

- Delphi
- C#
- VB/VBA (OLE)
- Pascal Scripting OLE
- Pascal Scripting

Delphi

Using	Iltem Object Model
1	
2	procedure main;
3	var
4	planswift: IPlanSwift;
5	settings: IItem;
6	property: IPropertyObject
7	begin
8	planswift := coPlanSwift .Create();
9	<pre>settings := planswift getItem('\Settings');</pre>
10	<pre>property := planswift.GetProperty('AngleSnapHotKey');</pre>
11	WriteLn (property ResultAsInteger())
	end

Using PlanSwift Object Model

1	
2	//or
3	procedure main;
4	var
5	planswift: IPlanSwift;
6	property:IPropertyObject;
7	begin
8	<pre>planswift := coPlanSwift Create();</pre>
9	<pre>property := planswift GetProperty('\Settings','AngleSnapHotKey');</pre>
10	WriteLn(property.ResultAsInteger())

C#

Using I	lltem Object Model
1	
2	private void Main()
3	
4	PlanSwift planswift = new PlanSwift();
5	<pre>IItem settings = planswift.GetItem(@ "\Settings");</pre>
6	IPropertyObject property = settings.GetProperty("AngleSnapHotKey");
7	console.WriteLn(property.ResultAsInteger())
	}

Using PlanSwift Object Model

1	pr	rivate void Main()
3 4 5		<pre>PlanSwift planswift = new PlanSwift(); IPropertyObject property = planswift.GetProperty(@"\Settings","AngleSnapHotKey")</pre>
6	}	console.WriteLn(property.ResultAsInteger)

VB/VBA (OLE)

1	n Object Model
2 Sub	main()
3	Dim planswift = CreateObject("PlanSwift9.PlanCenter")
4	<pre>Dim settings = planswift.GetItem("\Settings")</pre>
5	Dim property = settings.GetProperty("AngleSnapHotKey")
6	Console.WriteLn(property.ResultAsInteger());

1 2 Sub Main()

- 3
- 4 5

Pascal Scripting OLE

Item Object Model 1 2 3 4 Root Object Model 2

- 3

Pascal Scripting

odel
ttings := PlanSwift.getItem('\Settings');
<pre>operty := settings.GetProperty('AngleSnapHotKey');</pre>
owMessage(property.ResultAsInteger);

Using the PlanSwift Object Model

1		
2	begin	
3]	property := PlanSwift.GetProperty('\Settings','AngleSnapHotKey');
4		ShowMessage(property.ResultAsInteger);
	end	

AutoUpdate

AutoUpdate

Boolean value that toggles AutoUpdate on or off. Checked is true (on) and enables it. Unchecked is false (off) and disables it. If this box is checked, the Update Notifications settings in Figure 1 will be set to Notify me of all recommended updates. If the box is not checked, then the Update Notifications screen will be set to Do not notify me of updates. If the Update Notifications in the Settings / Notifications area is set to Notify me of all recommended updates, then the U-T-H AutoUpdate value will toggle to checked (true).

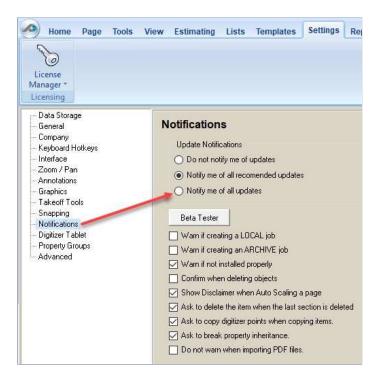


Figure 1

API Calls

- Delphi
- C#
- VB/VBA (OLE)
- Pascal Scripting OLE
- Pascal Scripting

Delphi





C#

	ft planswift = new Pla	ocuift().		
	it planswill = new Pla			
	ettings = planswift.Get			
J Durana a	tyObject property = set			
-	.WriteLn (property.Resul		(Macoopaace //	
}				

2 3

- 4
- 5 6

VB/VBA (OLE)

Sub	o main()
	Dim planswift = CreateObject("PlanSwift9.PlanCenter")
	Dim settings = planswift.GetItem("\Settings")
	Dim property = settings.GetProperty("AutoUpdate")
	Console.WriteLn(property.ResultAsBoolean());
End	

2	Main()
2	Dim planswift = CreateObject("PlanSwift9.PlanCenter")
4	<pre>Dim nameProperty = planswift.GetProperty("\Settings", "AutoUpdate")</pre>
5	Console.WriteLn(property.ResultAsBoolean)
5	

Pascal Scripting OLE

Ite	m Object Model
1	
2	
3	<pre>settings := getItem('\Settings');</pre>
4	ShowMessage(GetResultAsBoolean(settings, 'AutoUpdate'));
	end
RO	ot Object Model
1	
2	
3	ShowMessage(GetResultAsBoolean('\Settings','AutoUpdate'));

Pascal Scripting

Item Object Model

1	
2	begin
3	<pre>settings := PlanSwift.getItem('\Settings');</pre>
4	<pre>property := settings.GetProperty('AutoUpdate');</pre>
5	ShowMessage(property.ResultAsBoolean);
	end
Usin	g the PlanSwift Object Model

Using the PlanSwift Object Model

- property := PlanSwift.GetProperty('\Settings', 'AutoUpdate');
- 4 ShowMessage(property.ResultAsBoolean)
- €

AUTOCOUNTWIZARD

AUTOCOUNTWIZARD

Boolean value that enables or disables the display of Auto Count tool on the Main Menu. Checked (true) enables its display; unchecked (false) disables it. Figure 1 shows where this value is controlled in the Main Menu / Settings / Interface Settings window. Figure 2 shows where the Auto Count tool is displayed on the Main Menu when enabled.

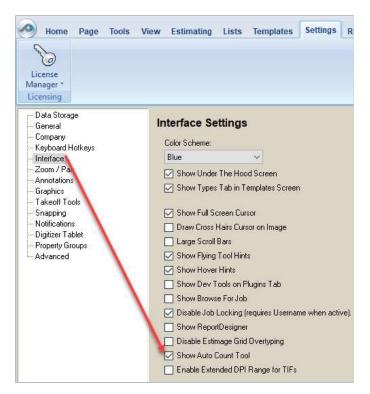


Figure 1

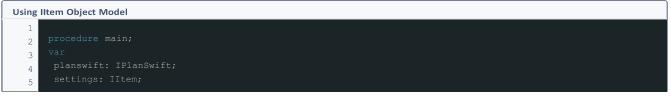


Figure 2

API Calls

- Delphi
- C#
- VB/VBA (OLE)
- Pascal Scripting (OLE)
- Pascal Scripting

Delphi



6 property: IPropertyObject

- 7 begin
- 8 planswift := coPlanSwift .Create
- 9 settings := planswift getItem('\Settings');
- 10 property := planswift GetProperty('AUTOCOUNTWIZARD');
- 11 WriteLn (property .RResultAsBoolean()

e

Using PlanSwift Object Model

1	
-	

- procedure mai
- 3 P 4 V
- 5 planswift: IPlanSwift;
- 6 property:IPropertyObject
- 7 begi
- 8 planswift := coPlanSwift .Create();
- g property := planswift.GetProperty('\Settings','AUTOCOUNTWIZARD')
- 10 WriteLn (property.ResultAsBoolean()

e

C#

Using Htem Object Model private void Main() { PlanSwift planswift = new PlanSwift(); IItem settings = planswift.GetItem(@"\Settings"); IPropertyObject property = settings.GetProperty("AUTOCOUNTWIZARD"); console.WriteLn(property.ResultAsBoolean()) }

Usin	g PlanSwift Object Model
1	
2	private void Main()
3	
4	<pre>PlanSwift planswift = new PlanSwift();</pre>
5	IPropertyObject property = planswift.GetProperty(@"\Settings","AUTOCOUNTWIZARD")
6	console.WriteLn(property.ResultAsBoolean)
Ŭ	

VB/VBA (OLE)

Using Ilter	n Object Model
1	
2 Sub	main()
3	Dim planswift = CreateObject("PlanSwift9.PlanCenter")
4	<pre>Dim settings = planswift.GetItem("\Settings")</pre>
5	<pre>Dim property = settings.GetProperty("AUTOCOUNTWIZARD")</pre>
6	Console.WriteLn(property.ResultAsBoolean());
End	

1	
2	Main()
3	Dim planswift = CreateObject("PlanSwift9.PlanCenter")
4	Dim nameProperty = planswift.GetProperty("\Settings","AUTOCOUNTWIZARD")
5	Console.WriteLn(property.ResultAsBoolean)

m Obj	ect Model
- L	adin
	settings := getItem('\Settings');
3	Sectings - gettem('Sectings'); ShowMessage(GetResultAsBoolean(settings, 'AUTOCOUNTWIZARD'));
4 ei	nd
ot Ob	ject Model
1 2 be	egin
	ShowMessage(GetResultAsBoolean('\Settings','AUTOCOUNTWIZARD'));
3	

Pascal Scripting

Item	Item Object Model		
1			
2	begin		
3	<pre>settings := PlanSwift.getItem('\Settings');</pre>		
4	<pre>property := settings.GetProperty('AUTOCOUNTWIZARD');</pre>		
5	ShowMessage(property.ResultAsBoolean);		
	end		

Using the PlanSwift Object Model

1 2 be

- 3 property := PlanSwift.GetProperty('\Settings',')
 4 ShowMessage(property.ResultAsBoolean);
 - end

AutoDimenOnScale

AutoDimenOnScale

Boolean value that controls the display of the **Disclaimer** when **Auto Scaling** a page if box is checked. Checked (true) displays the disclaimer; unchecked (false) disables display of the disclaimer. The PlanSwift **Home** / **Main Menu** ribbon bar **Scale Settings** control and the window it opens is shown in Figure 1. Click on **Auto** in the window to control scale automatically (Figure 2). After selecting a scale (from the drop-down shown in Figure 3) and clicking **OK**, the **Auto Scale Disclaimer** window (Figure 4) is visible (as long as the **AutoDimenOnscale** variable is set to true).



Figure 1

Standard F-I-S Metric	Auto
	~
INO DAITANC . Automatic anti-	of scale is only as
	awing MAKE SURE
accurate as the scanned in d YOU DOUBLE CHECK SEVERA	L DIMENSIONS ON
accurate as the scanned in d	L DIMENSIONS ON HE SCALE WAS SET
accurate as the scanned in d YOU DOUBLE CHECK SEVER THE PAGE TO MAKE SURE T ACCURATE	L DIMENSIONS ON HE SCALE WAS SET
accurate as the scanned in d YOU DOUBLE CHECK SEVERA THE PAGE TO MAKE SURE T	L DIMENSIONS ON HE SCALE WAS SET

Figure 2

Standard 🔨	F-I-S Metric Au		t
YOU DOUBL	1/16" = 1' 0" 3/32" = 1' 0" 1/8" = 1' 0" 3/16" = 1' 0"		only as TE SURE DNS ON AS SET
O'O'' Apply Scal Automatica	1-1/2" = 1'0"		lear Scale
		~	Cance

Figure 3

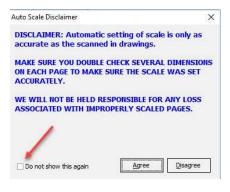


Figure 4

API Calls

- Delphi
- C#
- VB/VBA (OLE)
- Pascal Scripting OLE Pascal
- Scripting

Delphi

4	lltem Object Model
Ţ	procedure main;
2	var
3	planswift: IPlanSwift;
4 5	settings: IItem;
6	property: IPropertyObject
7	
8	planswift := coPlanSwift .Create();
9	<pre>settings := planswift.getItem('\Settings');</pre>
10	<pre>property := planswift.GetProperty('AutoDimenOnScale');</pre>
11	WriteLn (property.RResultAsBoolean())

0	
1	
2	
3	procedure main;
4	
5	planswift: IPlanSwift;
6	property:IPropertyObject;
7	
8	<pre>planswift := coPlanSwift .Create();</pre>
9	<pre>property := planswift.GetProperty('\Settings','AutoDimenOnScale');</pre>
10	WriteLn (property.ResultAsBoolean())

Using Iltem Object Model

2 private void Main()

3 [{]

1

5 6 7

- PlanSwift planswift = new PlanSwift();
- IItem settings = planswift.GetItem(@"\Settings");
- IPropertyObject property = settings.GetProperty("AutoDimenOnScale");
- console.WriteLn(property.ResultAsBoolean())

1	
2 P	private void Main()
3 {	
4	<pre>PlanSwift planswift = new PlanSwift();</pre>
5	IPropertyObject property = planswift.GetProperty(@"\Settings","AutoDimenOnScale")
6	console.WriteLn(property.ResultAsBoolean)

VB/VBA (OLE)

Sul	o main()
	Dim planswift = CreateObject("PlanSwift9.PlanCenter")
	Dim settings = planswift.GetItem("\Settings")
	Dim property = settings.GetProperty("AutoDimenOnScale")
	Console.WriteLn(property.ResultAsBoolean());
End	

USIII		
1		
2	Sub Main()	
3	Dim planswift = CreateObject("PlanSwift9.PlanCenter")	
4	Dim nameProperty = planswift.GetProperty("\Settings","AutoDimenOnScale")	
5	Console.WriteLn(property.ResultAsBoolean)	

Pascal Scripting OLE

Item Object Model			
<pre>1 2 begin 3 settings := getItem('\Settings'); 4 ShowMessage(GetResultAsBoolean(settings, 'AutoDimenOnScale')); end</pre>			
Root Object Model			

Pascal Scripting

beg:	
	<pre>settings := PlanSwift.getItem('\Settings');</pre>
	<pre>property := settings.GetProperty('AutoDimenOnScale');</pre>
	ShowMessage(property.ResultAsBoolean);
end	
end	
end	
	PlanSwift Object Model
	PlanSwift Object Model
g the I	PlanSwift Object Model
g the I	PlanSwift Object Model

AutoSelectFirstPage

AutoSelectFirstPage

Boolean value controlling whether the first page (in the Pages, Bookmarks window from the Home tab) is automatically selected (Figure 1). Checked (true) selects the first page automatically (Figure 2). Unchecked (false) brings up a blank PlanSwift screen (Figure 3).



Figure 1

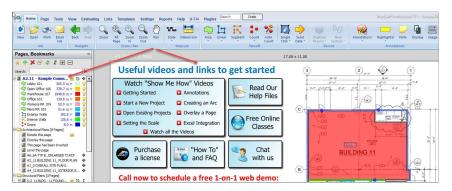


Figure 2

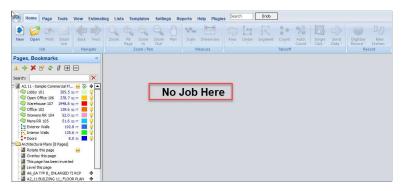


Figure 3

API Calls

- Delphi
- C#
- VB/VBA (OLE)
- Pascal Scripting (OLE)
- Pascal Scripting

Delphi

Using Iltem Object Model

-	
1	
2	procedure main;
3	
4	planswift: IPlanSwift;
5	settings: IItem;
6	property: IPropertyObject
7	
8	planswift := coPlanSwift .Create();
9	<pre>settings := planswift getItem('\Settings');</pre>
10	<pre>property := planswift.GetProperty('AutoSelectFirstPage');</pre>
11	WriteLn (property.RResultAsBoolean())

1	
2	
3	procedure main;
4	
5	planswift: IPlanSwift;
6	property:IPropertyObject;
7	
8	planswift := coPlanSwift .Create();
9	<pre>property := planswift GetProperty('\Settings','AutoSelectFirstPage');</pre>
10	WriteLn (property.ResultAsBoolean())

Using Iltem Object Model

Usi	ng PlanSwift Object Model
1	
2	private void Main()
3	{
4	<pre>PlanSwift planswift = new PlanSwift();</pre>
5	<pre>IPropertyObject property = planswift.GetProperty(@"\Setti</pre>
6	console.WriteLn(property.ResultAsBoolean)
-	

VB/VBA (OLE)

Using Iltem Object Model
2 Sub main()
<pre>3 Dim planswift = CreateObject("PlanSwift9.PlanCenter")</pre>
<pre>4 Dim settings = planswift.GetItem("\Settings")</pre>
5 Dim property = settings.GetProperty("AutoSelectFirstPage")
<pre>6 Console.WriteLn(property.ResultAsBoolean());</pre>
End Sub
Using PlanSwift Object Model

1		
2	Sub	Main()
3		Dim planswift = CreateObject("PlanSwift9.PlanCenter")
4		Dim nameProperty = planswift.GetProperty("\Settings","AutoSelectFirstPage")
5		Console.WriteLn(property.ResultAsBoolean)
	End	Sub

Pascal Scripting (OLE)

Item	Object Model
1 2	<pre>begin settings := getItem('\Settings');</pre>
3 4	Settings gettem(\Settings ;, ShowMessage(GetResultAsBoolean(settings, 'AutoSelectFirstPage')); end
Root	Object Model

2	, begin
3	<pre>ShowMessage(GetResultAsBoolean('\Settings','AutoSelectFirstPage'));</pre>
	end

Pascal Scripting

Item Object Model
1 2 begin
<pre>3 settings := PlanSwift.getItem('\Settings');</pre>
<pre>property := settings.GetProperty('AutoSelectFirstPage');</pre>
5 ShowMessage(property.ResultAsBoolean);
end
Using the PlanSwift Object Model
2 begin
<pre>3 property := PlanSwift.GetProperty('\Settings', 'AutoSelectFirstPage');</pre>
4 ShowMessage(property.ResultAsBoolean);
end

Color Scheme

Color Scheme

String value that controls the color scheme of PlanSwift's top window area. Choices are **Blue**, **Black**, or **Silver**. The default is **Blue**. See Figure 1. Selecting Black is shown in Figure 2.

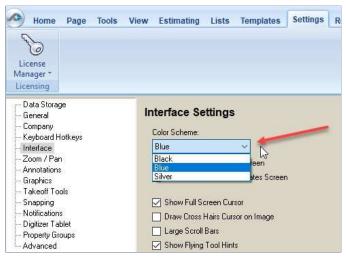


Figure 1





API Calls

- Delphi
- C#
- VB/VBA (OLE)
- Pascal Scripting (OLE)
- Pascal Scripting

Delphi

1	
2	procedure main;
3	
4	planswift: IPlanSwift;
5	settings: IItem;
6	property: IPropertyObject
7	
8	planswift := coPlanSwift Create();
9	<pre>settings := planswift .getItem('\Settings');</pre>
.0	<pre>property := planswift .GetProperty('Color Scheme');</pre>
1	WriteLn (property.ResultAsString())
-	



C#

#		
U	sing II	tem Object Model
	1	
	2 P	rivate void Main()
	3 {	
	4	PlanSwift planswift = new PlanSwift();
	5	IItem settings = planswift.GetItem(@"\Settings");
	6	IPropertyObject property = settings.GetProperty("Color Scheme");
	7	console.WriteLn(property.ResultAsString())
	}	

Using PlanSwift Object Model

1	
2	private void Main()
3	
4	PlanSwift planswift = new PlanSwift();
5	IPropertyObject property = planswift.GetProperty(@"\Settings","Color Scheme")
6	console.WriteLn(property.ResultAsString)

VB/VBA (OLE)

Using Iltem Object Model
2 Sub main()
<pre>Dim planswift = CreateObject("PlanSwift9.PlanCenter")</pre>
<pre>4 Dim settings = planswift.GetItem("\Settings")</pre>
5 Dim property = settings.GetProperty("Color Scheme")
<pre>6 Console.WriteLn(property.ResultAsString());</pre>
End Sub
Using PlanSwift Object Model
1

2 Sub Main() 3 Dim planswift = CreateObject("PlanSwift9.PlanCent

- Dim nameProperty = planswift.GetProperty("\Settings", "Color Scheme")
- 5 Console.WriteLn(property.ResultAsString
- End Sub

Pascal Scripting (OLE)

4

Item Object Model

1 2 begin 3 settings := getItem('\Settings');

ShowMessage(GetResultAsString(settings, 'Color Scheme'));

Root Object Model

1 2 3	begin ShowMessage(GetResultAsString('\Settings','Color Scheme')); end

Pascal Scripting

<pre>settings := PlanSwift.getItem('\Settings');</pre>
<pre>property := settings.GetProperty('Color Scheme');</pre>
ShowMessage(property.ResultAsString);

1	be	gin
3		<pre>property := PlanSwift.GetProperty('\Settings','Color Scheme'); ShowMessage(property.ResultAsString);</pre>
	en	a

ContinueWithHotKey

ContinueWithHotKey

Integer value returns an ANSI key code (default is "None") for the **Continue With** command. Figure 1 shows where the **Continue With** hotkey is assigned. Figure 2 shows the **Continue With** window when opened. Figure 3 shows the right-click menu where **Continue With** is normally activated (click on the takeoff item that is to be continued to select it, then right-click on it to see the drop-down menu, and then click on **Continue With** to open the **Continue With** window).

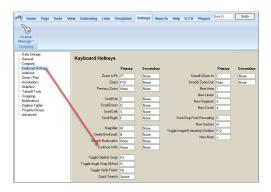


Figure 1

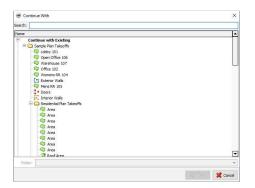


Figure 2

P	<u>P</u> roperties
Q	<u>H</u> ide
	Move
0	Lock
	Sho <u>w</u> Label
0	Add Mo <u>r</u> e Points
00	New Section
Q	Subtract From Section (P2P)
	Subtract from Section (BOX)
	Single Clic <u>k</u> Subtract Section
2	<u>С</u> ору
ß	P <u>a</u> ste
×	<u>D</u> elete
	Add Arc Po <u>i</u> nt
1t	<u>O</u> rder •
8	<u>E</u> dit Hyperlink
	Send Data
æ	<u>G</u> o To Item in Estimating
	Go To Item in Takeo <u>f</u> f Summary
800	Con <u>t</u> inue With
•)	Undo Report La <u>v</u> out Change

Figure 3

API Calls

- Delphi
- C#
- VB/VBA (OLE)
- Pascal Scripting (OLE)
- Pascal Scripting

Delphi

Using Iltem Object Model



1	
2	
3	procedure main;
4	
5	planswift: IPlanSwift;
6	property:IPropertyObject;
7	
8	planswift := coPlanSwift .Create();
9	<pre>property := planswift.GetProperty('\Settings','ContinueWithHotKey');</pre>
10	WriteLn (property.ResultAsInteger())

C#

Using Iltem Object Model
1 2 private void Main() 3 {
<pre>4 PlanSwift planswift = new PlanSwift();</pre>
<pre>5 IItem settings = planswift.GetItem(@"\Settings");</pre>
<pre>6 IPropertyObject property = settings.GetProperty("ContinueWithHotKey");</pre>
<pre>7 console.WriteLn(property.ResultAsInteger())</pre>
Using PlanSwift Object Model
1 2 private void Main() 3 {

- 4
- 5 6

VB/VBA (OLE)

Sub	main()
	Dim planswift = CreateObject("PlanSwift9.PlanCenter")
	<pre>Dim settings = planswift.GetItem("\Settings")</pre>
	Dim property = settings.GetProperty("ContinueWithHotKey")
	Console.WriteLn(property.ResultAsInteger());
End	
	Swift Object Model

2	Sub Main()
3	Dim planswift = CreateObject("PlanSwift9.PlanCenter")
4	<pre>Dim nameProperty = planswift.GetProperty("\Settings","ContinueWithHotKey")</pre>
5	Console.WriteIn(property.ResultAsInteger)
	End Sub

Pascal Scripting (OLE)

<pre>1 2 begin 3 settings := getItem('\Settings'); 4 ShowMessage(GetResultAsInteger(settings, 'ContinueWithHotKey')); end</pre>	Item Object	: Model
<pre>3 settings := getItem('\Settings'); 4 ShowMessage(GetResultAsInteger(settings, 'ContinueWithHotKey'));</pre>	1	
<pre>ShowMessage(GetResultAsInteger(settings, 'ContinueWithHotKey'));</pre>	2	
	5	
	-	Showmessage(GetResultAsinteget(Settings, Continuewithnotkey //,
	Root Object	t Model

<pre>1 2 begin 3 ShowMessage(GetResultAsInteger('\Settings','ContinueWithHotKey')); end</pre>	
---	--

Pascal Scripting

Item Ob	bject Model
1	
h	pegin
2 ~	
3	<pre>settings := PlanSwift.getItem('\Settings');</pre>
4	<pre>property := settings.GetProperty('ContinueWithHotKey');</pre>
5	ShowMessage(property.ResultAsInteger);
	and

Using the PlanSwift Object Model

- 1 2 beg
 - property := PlanSwift.GetProperty('\Settings', 'ContinueWithHotKey')
 - ShowMessage(property.ResultAsInteger);

e

3 4

CreateBookmarkHotKey

CreateBookmarkHotKey

Integer value returns an ANSI key code (default code 66, the letter B) for the Create Bookmark command. Figure 1 shows where the Create Bookmark hotkey is assigned.

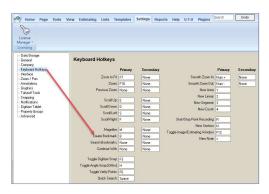


Figure 1

API Calls

- Delphi
- C#
- VB/VBA (OLE)
- Pascal Scripting (OLE)
- Pascal Scripting

Delphi

1	
2	procedure main;
3	
4	planswift: IPlanSwift;
5	settings: IItem;
5	property: IPropertyObject
7	
3	planswift := coPlanSwift Create();
)	<pre>settings := planswift .getItem('\Settings');</pre>
	<pre>property := planswift.GetProperty('CreateBookmarkHotkey');</pre>
	WriteLn (property.ResultAsInteger())
<u> </u>	

1	
2	//or
3	procedure main;
4	var
5	planswift: IPlanSwift;
6	<pre>property:IPropertyObject;</pre>
7	begin
8	planswift := coPlanSwift .Create();
9	<pre>property := planswift.GetProperty('\Settings','CreateBookmarkHotkey');</pre>
10	WriteLn (property.ResultAsInteger())
	end;
C#	

1	
2 P	rivate void Main()
3 {	
4	PlanSwift planswift = new PlanSwift();
5	IItem settings = planswift.GetItem(@"\Settings");
6	IPropertyObject property = settings.GetProperty("CreateBookmarkHotKey");
7	console.WriteLn(property.ResultAsInteger())
}	

1	
2	private void Main()
3	
4	PlanSwift planswift = new PlanSwift();
5	IPropertyObject property = planswift.GetProperty(@"\Settings","CreateBookmarkHotKey")
6	console.WriteLn(property.ResultAsInteger)

VB/VBA (OLE)

Suk	b main()
	Dim planswift = CreateObject("PlanSwift9.PlanCenter")
	Dim settings = planswift.GetItem("\Settings")
	Dim property = settings.GetProperty("CreateBookmarkHotKey")
	Console.WriteLn(property.ResultAsInteger());
Enc	

Using	PlanSwift	Object	Model
-------	-----------	--------	-------

1		
2	Sub	Main()
3		Dim planswift = CreateObject("PlanSwift9.PlanCenter")
4		Dim nameProperty = planswift.GetProperty("\Settings","CreateBookmarkHotKey")
5		Console.WriteLn(property.ResultAsInteger)
	End	

Pascal Scripting (OLE)

beg	
	<pre>settings := getItem('\Settings');</pre>
	ShowMessage(GetResultAsInteger(settings, 'CreateBookmarkHotKey'));
end	

Root Object Model

|--|

Pascal Scripting

begin		
3 se	ttings := PlanSwift.getItem('\Settings');	
4 pr	<pre>operty := settings.GetProperty('CreateBookmarkHotKey');</pre>	
5 Sh	owMessage(property.ResultAsInteger);	
end		

Using the PlanSwift Object Model

- 1 2 begi
 - property := PlanSwift.GetProperty('\Settings','CreateBookmarkHotKey')
 - ShowMessage(property.ResultAsInteger);

en

3 4

DefaultAreaTransparency

DefaultAreaTransparency

Integer value that sets the Area transparency default. Values range from 0 to 255, with 100 being the default. Figure 1 shows the DefaultAreaTransparency controls in both the Settings screen and the U-T-H Settings / Advanced Properties.

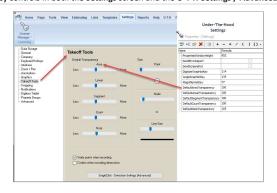


Figure 1

API Calls

- Delphi
- C#
- VB/VBA (OLE)
- Pascal Scripting (OLE)
- Pascal Scripting

Delphi

Using	Using IItem Object Model				
1					
2	procedure main;				
3	var				
4	planswift: IPlanSwift;				
5	settings: IItem;				
6	property: IPropertyObject				
7	begin				
8	planswift := coPlanSwift .Create();				
9	<pre>settings := planswift getItem('\Settings');</pre>				
10	<pre>property := planswift GetProperty('DefaultAreaTransparency');</pre>				
11	WriteLn (property ResultAsInteger())				
	end				

Using PlanSwift Object Model

1	
2	
3	procedure main;
4	
5	planswift: IPlan
6	property:IProper
7	
8	
~	WriteIn (property

- property := planswift.GetProperty('
- 9 WriteLn (property.ResultAsInteger()

C#

Using Iltem Object Model



- PlanSwift planswift = new PlanSwift();
- IPropertyObject property = planswift.GetProperty(@"\Settings","DefaultAreaTransparency")
- 5 IPropertyObject property = planswift.GetF 6 console.WriteLn(property.ResultAsInteger)

VB/VBA (OLE)

4

	Object Model
Sub	main()
3	Dim planswift = CreateObject("PlanSwift9.PlanCenter")
4	Dim settings = planswift.GetItem("\Settings")
5	Dim property = settings.GetProperty("DefaultAreaTransparency")
5	Console.WriteLn(property.ResultAsInteger());
End	

Using PlanSwift Object Model

1	
2	Sub Main()
3	Dim planswift = CreateObject("PlanSwift9.PlanCenter")
4	Dim nameProperty = planswift.GetProperty("\Settings","DefaultAreaTransparency")
5	Console.WriteLn(property.ResultAsInteger)
Ŭ	End Sub

Pascal Scripting (OLE)

Item Object N	Лodel		
1			
2 begin			
3 56	ettings := getItem('\Settings');		
4 St	howMessage(GetResultAsInteger(settings,	'DefaultAreaTransparency'));	
end			
Root Object N	Model		

1

```
2 begin
3 ShowMessage(GetResultAsInteger('\Settings','Default
```

Pascal Scripting

Item	Object Model
1	
2 3	<pre>begin settings := PlanSwift.getItem('\Settings');</pre>
4	<pre>property := settings.GetProperty('DefaultAreaTransparency');</pre>
5	<pre>ShowMessage(property.ResultAsInteger);</pre>
Usin	g the PlanSwift Object Model
1 2	begin
3	<pre>property := PlanSwift.GetProperty('\Settings','DefaultAreaTransparency');</pre>
4	ShowMessage(property.ResultAsInteger);

DefaultCountTransparency

DefaultCountTransparency

Integer value that sets the **Count** transparency default. Values range from 0 to 255, with 100 being the default. Figure 1 shows the **DefaultCountTransparency** controls in both the **Settings** screen and the **U-T-H Settings / Advanced Properties**.

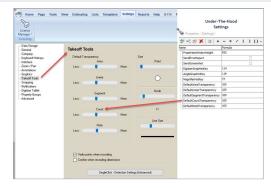


Figure 1

API Calls

- Delphi
- C#
- VB/VBA (OLE)
- Pascal Scripting (OLE)
- Pascal Scripting

Delphi

1	
2	procedure main;
3	
4	planswift: IPlanSwift;
5	settings: IItem;
6	property: IPropertyObject
7	
8	planswift := coPlanSwift .Create();
9	<pre>settings := planswift .getItem('\Settings');</pre>
10	<pre>property := planswift.GetProperty('DefaultCountTransparency');</pre>
11	WriteLn (property.ResultAsInteger())

Using PlanSwift Object Model

1	
2	//or
3	procedure main;
4	var
5	planswift: IPlanSwift;
6	property:IPropertyObject;
7	begin
8	planswift := coPlanSwift .Create();
9	<pre>property := planswift .GetProperty('\Settings','DefaultCountTransparency');</pre>
10	WriteLn (property ResultAsInteger())
20	end;

C#

Using Iltem Object Model

- 4 PlanSwift planswift = new PlanSwift();
 - IPropertyObject property = planswift.GetProperty(@"\Settings","DefaultCountTransparency")
- 5IPropertyObject property = planswift.Get6console.WriteIn(property.ResultAsInteger

VB/VBA (OLE)

Using IItem Object Model
1
2 Sub main()
<pre>3 Dim planswift = CreateObject("PlanSwift9.PlanCenter")</pre>
<pre>4 Dim settings = planswift.GetItem("\Settings")</pre>
5 Dim property = settings.GetProperty("DefaultCountTransparency")
6 Console.WriteLn(property.ResultAsInteger());
End Sub

Using PlanSwift Object Model

1		
2	Sub 1	Main()
3		Dim planswift = CreateObject("PlanSwift9.PlanCenter")
4		Dim nameProperty = planswift.GetProperty("\Settings","DefaultCountTransparency")
5		Console.WriteLn(property.ResultAsInteger)
	End	

Pascal Scripting (OLE)

beq	pegin	
	<pre>settings := getItem('\Settings');</pre>	
	<pre>ShowMessage(GetResultAsInteger(settings, 'DefaultCountTransparency'));</pre>	
end		

Root Object Model

1	
2	begin
3	<pre>ShowMessage(GetResultAsInteger('\Settings', 'DefaultCountTransparency'));</pre>

Pascal Scripting

	n Object Model
1	
2	<pre>begin settings := PlanSwift.getItem('\Settings');</pre>
3	<pre>property := settings.GetProperty('DefaultCountTransparency');</pre>
5	<pre>ShowMessage(property.ResultAsInteger);</pre>
	end
Usir	ng the PlanSwift Object Model
Usir 1	
Usir 1 2	ng the PlanSwift Object Model
1	
1 2	begin
1 2	begin

DefaultCurrency

DefaultCurrency

String value that displays and controls the default currency (Figure 1). Options are those on the **Default Currency** drop-down menu below (see Figure 1). In the **U-T-H Settings** (advanced properties) screen (Figure 2), click the **Formula** cell for **Default Currency** to open the **DefaultCurrency Formula Editor** window. The currency may be edited here. Clicking on **OK** saves it, and it will be displayed in the **Default Currency** field in the **Settings** / **General** window.







Figure 2

API Calls

- Delphi
- C#
- VB/VBA (OLE)
- Pascal Scripting (OLE)
- Pascal Scripting

Delphi

Using	Iltem Object Model
1	
2	procedure main;
3	var
4	planswift: IPlanSwift;
5	settings: IItem;
6	property: IPropertyObject
7	begin
8	<pre>planswift := coPlanSwift .Create();</pre>
9	<pre>settings := planswift.getItem('\Settings');</pre>
10	<pre>property := planswift.GetProperty('DefaultCurrency');</pre>
11	WriteLn (property ResultAsString())
	end

Using PlanSwift Object Model

1	
2	
3	procedure main;
4	
5	planswift: IPlanSwift;
6	property:IPropertyObject;
7	
8	<pre>planswift := coPlanSwift .Create();</pre>
9	<pre>property := planswift.GetProperty('\Settings','DefaultCurrency');</pre>
10	WriteLn (property.ResultAsString())

C#

Using Iltem Object Model

- 1 2 private void Main
- 3
- 4 PlanSwift planswift = new PlanSwift();
- 5 IItem settings = planswift.GetItem(@"\Settings");
- 6 IPropertyObject property = settings.GetProperty("DefaultCurrency");
- 7 console.WriteLn(property.ResultAsString())

Using PlanSwift Object Model private void Main() { PlanSwift planswift = new PlanSwift(); FropertyObject property = planswift.GetProperty(@"\Settings","DefaultCurrency") console.WriteLn(property.ResultAsString) }

VB/VBA (OLE)

Sub	o main()
	Dim planswift = CreateObject("PlanSwift9.PlanCenter")
	Dim settings = planswift.GetItem("\Settings")
	Dim property = settings.GetProperty("DefaultCurrency")
	Console.WriteLn(property.ResultAsString());
Enc	

1	
2 Sub	Main()
3	Dim planswift = CreateObject("PlanSwift9.PlanCenter")
4	<pre>Dim nameProperty = planswift.GetProperty("\Settings","DefaultCurrency")</pre>
5	Console.WriteLn(property.ResultAsString)
End	

Pascal Scripting (OLE)

1	iect Model
2 be	egin
3	<pre>settings := getItem('\Settings');</pre>
1	ShowMessage(GetResultAsString(settings, 'DefaultCurrency'));
-	
	nd
E1.	nd
	nd
ot Obj	ject Model
ot Obj 1 2 be	ject Model
oot Obj	ject Model

Pascal Scripting

Item Obje	rt Model
3	settings := PlanSwift.getItem('\Settings');
4	<pre>property := settings.GetProperty('DefaultCurrency');</pre>
5	ShowMessage(property.ResultAsString);
end	
Using the	PlanSwift Object Model

1 2 begi 3 4

- property := PlanSwift.GetProperty('\Settings', 'DefaultCurrency');
- ShowMessage(property.ResultAsString);

DefaultLinearTransparency

DefaultLinearTransparency

Integer value that sets the Linear transparency default. Values range from 0 to 255, with 100 being the default. Figure 1 shows the DefaultLinearTransparency controls in both the Settings screen and the U-T-H Settings / Advanced Properties.

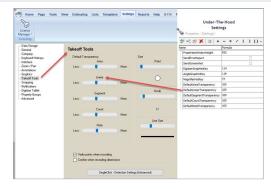


Figure 1

API Calls

- Delphi
- C#
- VB/VBA (OLE)
- Pascal Scripting (OLE)
- Pascal Scripting

Delphi

1	
2	procedure main;
3	
4	planswift: IPlanSwift;
5	settings: IItem;
6	property: IPropertyObject
7	
8	planswift := coPlanSwift Create();
9	<pre>settings := planswift .getItem('\Settings');</pre>
10	<pre>property := planswift.GetProperty('DefaultLinearTransparency');</pre>
11	WriteLn (property.ResultAsInteger())

Using PlanSwift Object Model

1	//or
2	procedure main;
3 4	var
5	planswift: IPlanSwift;
6	<pre>property:IPropertyObject;</pre>
7	begin
8	<pre>planswift := coPlanSwift .Create();</pre>
9	<pre>property := planswift.GetProperty('\Settings','DefaultLinearTransparency');</pre>
10	WriteLn (property .ResultAsInteger())
	end;

C#

Using IItem Object Model

private void Main() { PlanSwift planswift = new PlanSwift(); IItem settings = planswift.GetItem(@"\Settings"); IPropertyObject property = settings.GetProperty("DefaultLinearTransparency"); console.WriteLn(property.ResultAsInteger()) } Using PlanSwift Object Model private void Main() {

- 4 PlanSwift planswift = new PlanSwift();
- 5 IPropertyObject property = planswift.GetProperty(@"\Settings","DefaultLinearTransparency")
- 6 console.WriteLn(property.ResultAsInteger

VB/VBA (OLE)

Using Iltem Object Model
2 Sub main()
<pre>3 Dim planswift = CreateObject("PlanSwift9.PlanCenter")</pre>
<pre>4 Dim settings = planswift.GetItem("\Settings")</pre>
5 Dim property = settings.GetProperty("DefaultLinearTransparency")
<pre>6 Console.WriteLn(property.ResultAsInteger());</pre>
End Sub

Using PlanSwift Object Model

1		
2	Sub	Main()
3		Dim planswift = CreateObject("PlanSwift9.PlanCenter")
4		<pre>Dim nameProperty = planswift.GetProperty("\Settings","DefaultLinearTransparency")</pre>
5		Console.WriteLn(property.ResultAsInteger)
-	End	

Pascal Scripting (OLE)

Item	Object Model
1	
2	begin
3	
4	

Root Object Model

2	begin
3	<pre>ShowMessage(GetResultAsInteger('\Settings', 'DefaultLinearTransparency'));</pre>
	and

Pascal Scripting

Item	Object Model
1	
2	<pre>begin settings := PlanSwift.getItem('\Settings');</pre>
3	<pre>settings := rianswirt.getitem(\Settings); property := settings.GetProperty('DefaultLinearTransparency');</pre>
4	ShowMessage (property.ResultAsInteger);
5	
Usin	g the PlanSwift Object Model
Usin 1	g the PlanSwift Object Model
Usin 1 2	g the PlanSwift Object Model
1	
1	
1 2	begin
1 2 3	<pre>begin property := PlanSwift.GetProperty('\Settings','DefaultLinearTransparency');</pre>

DefaultMeasurementType

DefaultMeasurementType

String value that selects between Metric and English (Imperial) measurement types (see Figure 1). The U-T-H window allows the default measurement type to be changed there (Figure 2). Enter English or Metric and click on OK. Figure 3 shows where the Measurement Type is set for a New Job.

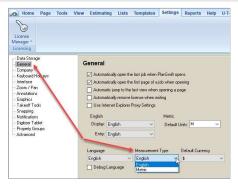


Figure 1

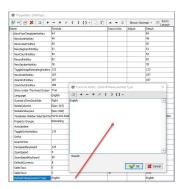


Figure 2

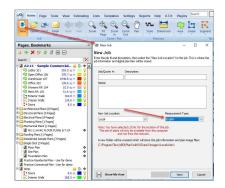


Figure 3

API Calls

- Delphi
- C#
- VB/VBA (OLE)
- Pascal Scripting (OLE)
- Pascal Scripting

Delphi

Using Iltem Object Model

1
2 procedure main;
3 var
4 planswift: IPlanSwift;
5 settings: IItem;
6 property: IPropertyObject
7 begin planswift := coPlanSwift.Create();
8 settings := planswift getItem('\Settings');
9 property := planswift GetProperty('DefaultMeasurementType');
10 WriteLn(property.ResultAsString())
end

Using PlanSwift Object Model

1	
2	
3	procedure main;
4	
5	planswift: IPlanSwift;
6	property:IPropertyObject;
7	
8	<pre>planswift := coPlanSwift Create();</pre>
9	<pre>property := planswift GetProperty('\Settings','DefaultMeasurementType');</pre>
10	WriteLn (property.ResultAsString())

C#

Using	lltem Object Model
1	
2	private void Main()
3	
4	<pre>PlanSwift planswift = new PlanSwift();</pre>
5	IItem settings = planswift.GetItem(@"\Settings");
6	IPropertyObject property = settings.GetProperty("DefaultMeasurementType");
7	console.WriteLn(property.ResultAsString())
	F

Using PlanSwift Object Model

1	private void Main()
3	{ PlanSwift planswift = new PlanSwift();
4 5	<pre>IPropertyObject property = planswift.GetProperty(@"\Settings","DefaultMeasurementType")</pre>
6	<pre>console.WriteLn(property.ResultAsString) }</pre>

VB/VBA (OLE)

Using Iltem Object Model	
1	
2	Sub main()
3	Dim planswift = CreateObject("PlanSwift9.PlanCenter")
4	Dim settings = planswift.GetItem("\Settings")
5	Dim property = settings.GetProperty("DefaultMeasurementType")
6	Console.WriteLn(property.ResultAsString());
Ŭ	

Using PlanSwift Object Model

1	
2 Sub	Main()
3	Dim planswift = CreateObject("PlanSwift9.PlanCenter")
4	Dim nameProperty = planswift.GetProperty("\Settings","DefaultMeasurementType")
5	Console.WriteLn(property.ResultAsString)
End	

Pascal Scripting (OLE)

Item Object	ct Model
1 2 beg	
2 Deg 3	settings := getItem('\Settings');
4	ShowMessage(GetResultAsString(settings, 'DefaultMeasurementType'));
end	
Root Obje	ct Model
1	
1 2 beg	
2	ShowMessage(GetResultAsString('\Settings','DefaultMeasurementType'));

Pascal Scripting

Item Object Model

1	
2	begin
3	<pre>settings := PlanSwift.getItem('\Settings');</pre>
4	<pre>property := settings.GetProperty('DefaultMeasurementType');</pre>
5	ShowMessage(property.ResultAsString);

Using the PlanSwift Object Model

- 1 2 beg
- 3 property := PlanSwift.GetProperty('\Settings', 'DefaultMeasurementType');
- 4 ShowMessage(property.ResultAsString);

DefaultNoteTransparency

DefaultNoteTransparency

Integer value that sets the **Note** transparency default. Values range from 0 to 255, with 100 being the default. Figure 1 shows the **DefaultNoteTransparency** controls in both the **Settings** screen and the **U-T-H Settings / Advanced Properties**.

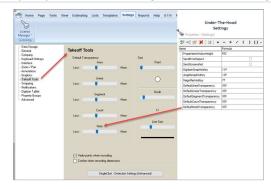


Figure 1

API Calls

- Delphi
- C#
- VB/VBA (OLE)
- Pascal Scripting (OLE)
- Pascal Scripting

Delphi

1	
2	procedure main;
3	
4	planswift: IPlanSwift;
5	settings: IItem;
6	property: IPropertyObject
7	
8	planswift := coPlanSwift .Create();
9	<pre>settings := planswift .getItem('\Settings');</pre>
10	<pre>property := planswift.GetProperty('DefaultNoteTransparency');</pre>
11	WriteLn (property.ResultAsInteger())

Using PlanSwift Object Model

1	
2	
3	procedure main;
4	
5	planswift: IPlanSwift;
6	property:IPropertyObject;
7	
8	planswift := coPlanSwift .Create();
9	<pre>property := planswift.GetProperty('\Settings','DefaultNoteTransparency');</pre>
10	WriteLn(property.ResultAsInteger())

C#

Using Iltem Object Model

- 4 PlanSwift planswift = new PlanSwift();
 - IPropertyObject property = planswift.GetProperty(@"\Settings","DefaultNoteTransparency")
- 5IPropertyObject property = planswift.Get6console.WriteIn(property.ResultAsInteger

VB/VBA (OLE)

sing Iltem Object Model
2 Sub main()
<pre>3 Dim planswift = CreateObject("PlanSwift9.PlanCenter")</pre>
<pre>4 Dim settings = planswift.GetItem("\Settings")</pre>
5 Dim property = settings.GetProperty("DefaultNoteTransparency")
6 Console.WriteIn(property.ResultAsInteger());
End Sub

Using PlanSwift Object Model

	1	
	2	Sub Main()
	3	Dim planswift = CreateObject("PlanSwift9.PlanCenter")
	4	Dim nameProperty = planswift.GetProperty("\Settings","DefaultNoteTransparency")
Console.WriteLn(property.ResultAsInteger)		Console.WriteLn(property.ResultAsInteger)
		End Sub

Pascal Scripting (OLE)

Item	Object Model
1 2 3 4	<pre>begin settings := getItem('\Settings'); ShowMessage(GetResultAsInteger(settings, 'DefaultNoteTransparency'));</pre>
	end
Deed	

Root Object Model

-	
2	begin

3 ShowMessage(GetResultAsInteger('\Settings','DefaultNoteTransparency'));

Pascal Scripting

Item Object Model	
<pre>begin settings := PlanSwift.getItem('\Settings');</pre>	
<pre>settings := rianswift.getftem(\Settings); property := settings.GetProperty('DefaultNoteTransparency');</pre>	
ShowMessage(property.ResultAsInteger);	
g the PlanSwift Object Model	
begin	
<pre>property := PlanSwift.GetProperty('\Settings','DefaultNoteTransparency');</pre>	
ShowMessage(property.ResultAsInteger);	

DefaultSegmentTransparency

DefaultSegmentTransparency

Integer value that sets the Segment transparency default. Values range from 0 to 255, with 100 being the default. Figure 1 shows the DefaultSegmentTransparency controls in both the Settings screen and the U-T-H Settings / Advanced Properties.

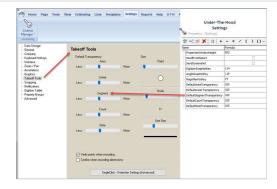


Figure 1

API Calls

- Delphi
- C#
- VB/VBA (OLE)
- Pascal Scripting (OLE)
- Pascal Scripting

Delphi

2	procedure main;
3	
1	planswift: IPlanSwift;
5	settings: IItem;
5	property: IPropertyObject
7	
3	planswift := coPlanSwift Create();
	<pre>settings := planswift getItem('\Settings');</pre>
	<pre>property := planswift.GetProperty('DefaultSegmentTransparency');</pre>
	WriteLn (property.ResultAsInteger())

Using PlanSwift Object Model

1	
2	
3	procedure main;
4	
5	planswift: IPlanSwift;
6	property:IPropertyObject;
7	
8	planswift := coPlanSwift .Create();
9	<pre>property := planswift.GetProperty('\Settings','DefaultSegmentTransparency');</pre>
10	WriteLn (property.ResultAsInteger())

C#

Using Iltem Object Model



- PlanSwift planswift = new PlanSwift();
- IPropertyObject property = planswift.GetProperty(@"\Settings","DefaultSegmentTransparency")
- 5 IPropertyObject property = planswift.GetF 6 console.WriteLn(property.ResultAsInteger)

VB/VBA (OLE)

4

	Object Model
2 Sub	main()
3	Dim planswift = CreateObject("PlanSwift9.PlanCenter")
4	Dim settings = planswift.GetItem("\Settings")
5	Dim property = settings.GetProperty("DefaultSegmentTransparency")
6	Console.WriteLn(property.ResultAsInteger());
End	

Using PlanSwift Object Model

Pascal Scripting (OLE)

ltem	Object Model
2 3	<pre>begin settings := getItem('\Settings');</pre>
4	ShowMessage(GetResultAsInteger(settings, 'DefaultSegmentTransparency')); end
Root	C Object Model

1

- 2 beg
- 3 ShowMessage(GetResultAsInteger('\Settings','DefaultSegmentTransparency'));

Pascal Scripting

Item	Object Model
1	
2	<pre>begin settings := PlanSwift.getItem('\Settings');</pre>
3	<pre>property := settings.GetProperty('DefaultSegmentTransparency');</pre>
4	ShowMessage (property.ResultAsInteger);
J	
Usin	g the PlanSwift Object Model
1	
2	begin
2	
3 4	<pre>property := PlanSwift.GetProperty('\Settings', 'DefaultSegmentTransparency');</pre>
4	ShowMessage(property.ResultAsInteger); end

DigitizerSnap

(false) it is not highlighted.

DigitizerSnap

 Home
 Page
 Tools
 View
 Estimating
 Lists
 Templates
 Settings
 Reports

 License
 Manager Interface
 Interface
 Interface
 Interface
 Snap to CAD points
 Snap to CAD points
 Snap to CAD points
 Interface
 Interface

Figure 1

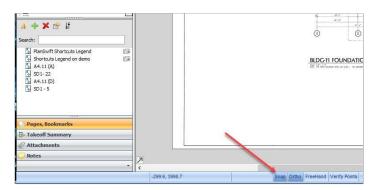


Figure 2

API Calls

- Delphi
- C#
- VB/VBA (OLE)
- Pascal Scripting (OLE)
- Pascal Scripting

Delphi

1	
2	procedure main;
3	
4	planswift: IPlanSwift;
5	settings: IItem;
6	property: IPropertyObject
7	
8	<pre>planswift := coPlanSwift Create();</pre>
9	<pre>settings := planswift .getItem('\Settings');</pre>
0	<pre>property := planswift.GetProperty('DigitizerSnap');</pre>
1	<pre>WriteLn (property.ResultAsBoolean())</pre>
-	

Boolean value that toggles digitizer Snap On and Off. Figures 1 and 2 show where Snap is controlled. When enabled (true), the Snap control is highlighted; when disabled

Using PlanSwift Object Model

1	
2	
3	procedure main;
4	
5	planswift: IPlanSwift;
6	property:IPropertyObject;
7	
8	<pre>planswift := coPlanSwift Create();</pre>
9	<pre>property := planswift GetProperty('\Settings','DigitizerSnap');</pre>
10	WriteLn (property.ResultAsBoolean())
	end;

C#

Using Iltem Object Model

1	
2	private void Main()
3	
4	PlanSwift planswift = new PlanSwift();
5	<pre>IItem settings = planswift.GetItem(@"\Settings");</pre>
6	IPropertyObject property = settings.GetProperty("DigitizerSnap");
7	console.WriteLn(property.ResultAsBoolean())
	3

Using PlanSwift Object Model

1 2	private void Main()
3 4	{ PlanSwift planswift = new PlanSwift();
5	
6	console.WriteLn(property.ResultAsBoolean) }

VB/VBA (OLE)

Sub ma	
D	<pre>im planswift = CreateObject("PlanSwift9.PlanCenter")</pre>
D	im settings = planswift.GetItem("\Settings")
D	<pre>im property = settings.GetProperty("DigitizerSnap")</pre>
C	onsole.WriteLn(property.ResultAsBoolean());
End Si	

2 Sub Main()	
3 Dim planswift = CreateObject("PlanSwift9.PlanCenter")	
<pre>Dim nameProperty = planswift.GetProperty("\Settings", "DigitizerSnap")</pre>	
5 Console.WriteLn(property.ResultAsBoolean)	
End Sub	

Pascal Scripting (OLE)

Item Objec	ct Model
1 2 beg 3 4 end	settings := getItem('\Settings'); ShowMessage(GetResultAsBoolean(settings, 'DigitizerSnap'));
Root Obje	ct Model
1 2 beg 3	<pre>in ShowMessage(GetResultAsBoolean('\Settings', 'DigitizerSnap'));</pre>
and	

Pascal Scripting

Item Object Model	
2 begin	
<pre>3 settings := PlanSwift.getItem('\Settings');</pre>	
<pre>4 property := settings.GetProperty('DigitizerSnap');</pre>	
<pre>5 ShowMessage(property.ResultAsBoolean);</pre>	
end	
Using the PlanSwift Object Model	

DigitizerSnapHotKey

DigitizerSnapHotKey

Snap is also controlled.

 Internet
 Page
 Total
 Verwer
 Extended
 Setting
 Respects
 Heigh
 Result
 Linds

 Understand
 Dark Strategy

Figure 1



Integer value that returns an ANSI key code (default code 114, key F3). Figure 1 shows where the DigitizerSnapHotKey assignment is made. Figures 2 and 3 show where

Figure 2



Figure 3

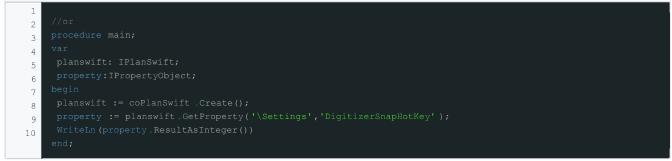
API Call

- Delphi
- C#
- VB/VBA (OLE)
- Pascal Scripting (OLE)
- Pascal Scripting

Delphi

Using IItem Object Model		
1		
2	procedure main;	
3	var	
4	planswift: IPlanSwift;	
5	settings: IItem;	
6	property: IPropertyObject	
7	begin	
8	planswift := coPlanSwift .Create();	
9	<pre>settings := planswift_getItem('\Settings');</pre>	
10	<pre>property := planswift GetProperty('DigitizerSnapHotKey');</pre>	
11	WriteLn (property ResultAsInteger())	
	end	

Using PlanSwift Object Model



C#

Using Iltem Object Model private void Main() { PlanSwift planswift = new PlanSwift(); Iltem settings = planswift.GetItem(@"\Settings"); IPropertyObject property = settings.GetProperty("DigitizerSnapHotKey"); console.WriteLn(property.ResultAsInteger()) } Using PlanSwift Object Model

private void Main() { PlanSwift planswift = new PlanSwift(); IPropertyObject property = planswift.GetProperty(@"\Settings","DigitizerSnapHotKey") console.WriteLn(property.ResultAsInteger) }

VB/VBA (OLE)

sing Iltem	Object Model
1	
2 Sub	main()
3	Dim planswift = CreateObject("PlanSwift9.PlanCenter")
4	<pre>Dim settings = planswift.GetItem("\Settings")</pre>
5	Dim property = settings.GetProperty("DigitizerSnapHotKey")
6	Console.WriteLn(property.ResultAsInteger());
End	
-:	
sing Plan	Swift Object Model
1	
2 Sub	Main()
3	Dim planswift = CreateObject("PlanSwift9.PlanCenter")
	Dim nameProperty = planswift.GetProperty("\Settings","DigitizerSnapHotKey")

5 Console.WriteLn(property.ResultAsInteger)

End Sul

Pascal Scripting (OLE)

Item Object Model 1 2 begin 3 settings := getItem('\Settings'); 4 ShowMessage(GetResultAsInteger(settings, 'DigitizerSnapHotKey')); end Root Object Model

ROOT	
1 2 3	<pre>begin ShowMessage(GetResultAsInteger('\Settings', 'DigitizerSnapHotKey')); end</pre>
З	ShowMessage(GetResultAsInteger('\Settings','DigitizerSnapHotKey'));

Pascal Scripting

begi	
	<pre>settings := PlanSwift.getItem('\Settings');</pre>
	property := settings.GetProperty('DigitizerSnapHotKey');
	ShowMessage(property.ResultAsInteger);
end	

Using the PlanSwift Object Model

1	
2	
3	<pre>property := PlanSwift.GetProperty('\Settings','DigitizerSnapHotKey');</pre>
4	<pre>ShowMessage(property.ResultAsInteger);</pre>

EnhancedImage

EnhancedImage

Boolean value that controls whether an image is enhanced. Checked (true) enhances the image. Unchecked (false) doesn't enhance it.

API Calls

- Delphi
- C#
- VB/VBA (OLE)
- Pascal Scripting (OLE)
- Pascal Scripting

Delphi

p	rocedure main;
v	
	planswift: IPlanSwift;
	settings: IItem;
; 1	property: IPropertyObject
b	
	planswift := coPlanSwift Create();
, .	<pre>settings := planswift .getItem('\Settings');</pre>
1	<pre>property := planswift .GetProperty('EnhancedImage');</pre>
	WriteLn (property .RResultAsBoolean())

Using PlanSwift Object Model

C#

Using Iltem Object Model

Using PlanSwift Object Model		
1		
2	private void Main() {	
3	PlanSwift planswift = new PlanSwift();	
4	IPropertyObject property = planswift.GetProperty(@"\Settings","EnhancedImage")	
5	console.WriteLn(property.ResultAsBoolean)	
6		

VB/VBA (OLE)

<pre>swift = CreateObject("PlanSwift9.Plan(</pre>			
<pre>ings = planswift.GetItem("\Settings")</pre>			
erty = settings.GetProperty("Enhanced	Image")		
<pre>%riteLn(property.ResultAsBoolean());</pre>			
		perty = settings.GetProperty("EnhancedImage")	<pre>perty = settings.GetProperty("EnhancedImage")</pre>

- Dim nameProperty = planswift.GetProperty("\Settings", "EnhancedImage")
- 5 Console.WriteLn(property.ResultAsBoolean
- End Sub

Pascal Scripting (OLE)

Item Object Model			
1			
2	begin		
3	<pre>settings := getItem('\Settings');</pre>		
4	ShowMessage(GetResultAsBoolean(settings, 'EnhancedImage'));		
-	end		

Root Object Model

1	
2	begin
3	ShowMessage(GetResultAsBoolean('\Settings','EnhancedImage'));
-	

Pascal Scripting

Item Object Model	
<pre>1 2 begin 3 settings := PlanSwift.getItem('\Settings'); 4 property := settings.GetProperty('EnhancedImage'); 5 ShowMessage(property.ResultAsBoolean); end</pre>	

Using the PlanSwift Object Model

1	
2	begin
3	<pre>property := PlanSwift.GetProperty('\Settings','EnhancedImage');</pre>
4	ShowMessage(property.ResultAsBoolean);
	end

FullScreenCursor

FullScreenCursor

Boolean value controlling whether the cursor crosshairs extend fully up and down and left and right across the screen (see Figure 1). Checked (true) displays extended crosshairs. Unchecked (false) displays shortened crosshairs.

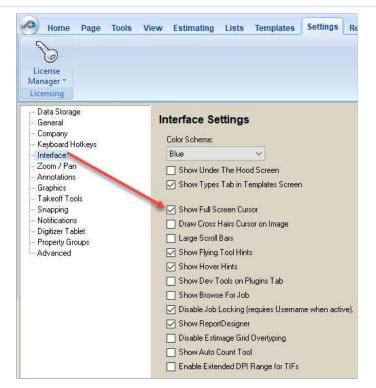
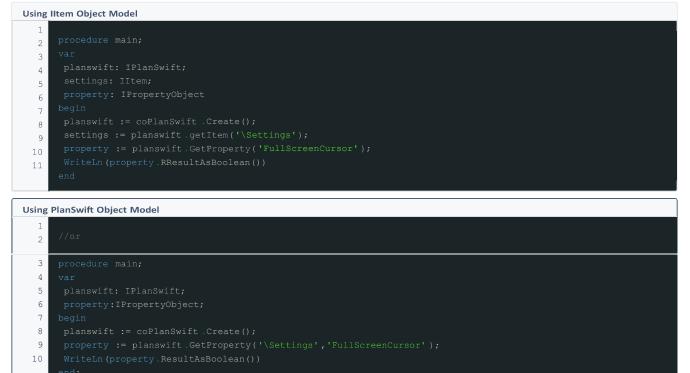


Figure 1

API Calls

- Delphi
- C#
- VB/VBA (OLE)
- Pascal Scripting (OLE)
- Pascal Scripting

Delphi



C#

<pre>hin() hin() hinswift = new PlanSwift(); ngs = planswift.GetItem(@"\Settings"); hject property = settings.GetProperty("Full hins in the settings.detProperty("Full hins in</pre>	ScreenCursor");
ngs = planswift.GetItem(@"\Settings"); ject property = settings.GetProperty("Full	ScreenCursor");
ngs = planswift.GetItem(@"\Settings"); ject property = settings.GetProperty("Full	ScreenCursor");
ject property = settings.GetProperty("Full	ScreenCursor");
	ScreenCursor");
teLn(property.ResultAsBoolean())	
a - J - 1	
10061	
N	Model

3				
4	PlanSwift	planswift		<pre>PlanSwift();</pre>

- 5 6

VB/VBA (OLE)

Sub	<pre>main()</pre>
	Dim planswift = CreateObject("PlanSwift9.PlanCenter")
	Dim settings = planswift.GetItem("\Settings")
	Dim property = settings.GetProperty("FullScreenCursor")
	Console.WriteLn(property.ResultAsBoolean());
End	

2	Sub Main()
3	Dim planswift = CreateObject("PlanSwift9.PlanCenter")
4	<pre>Dim nameProperty = planswift.GetProperty("\Settings","FullScreenCursor")</pre>
5	Console.WriteIn(property.ResultAsBoolean)

Pascal Scripting (OLE)

Item	Object Model
1	
2	begin
3	settings := getItem('\Settings'); ShowMessage(GetResultAsBoolean(settings, 'FullScreenCursor'));
4	end
Root	Object Model
1	

Pascal Scripting

2 3

Item Object Model	
1	
2 begin	
3 setting	s := PlanSwift.getItem('\Settings');
4 propert	y := settings.GetProperty('FullScreenCursor');
5 ShowMes	<pre>sage(property.ResultAsBoolean);</pre>
end	
Using the PlanSwift	Object Model
1	
2 begin	
3 propert	y := PlanSwift.GetProperty('\Settings','FullScreenCursor');
-	<pre>sage(property.ResultAsBoolean);</pre>

en

HideTypesTab

HideTypesTab

Boolean value that toggles the **Type Tab** in the **Templates** screen to show or hide. Checked (true) displays the **Types Tab** (Figure 1); unchecked (false) does not display it. Figure 2 shows the **Types** tab.



Figure 1

	Home	Pag	e Tool	s View	Estimating	Lists	Templates	Settings	Reports
1)		In.	R				~
New		ew m *	New Area *	New Linear *	New Segment *	New Count *	New	New Part *	New from
					,			/	Ten
				1.		mblies 🧯	Tunor	a las	
😐 Sa	mple Ta	keof	f Templa	tes 🚺 P	arts and Asse	mblies	Types Li	inks	
	imple Ta	ikeof	f Templa		Parts and Assei	Price E		Color	
lame	Area Dro								
lame		opdow	vn						
lame t 🗀	Area Dro	opdow Tools	vn						
Name	Area Dro Scripted	opdow Tools ropdo	vn s wn						
lame	Area Dro Scripted Count D	opdow Tools ropdo ropdo	vn : : wn : : :						
	Area Dro Scripted Count D Linear D	opdow Tools ropdo ropdo t Drop	wn wwn wn wdown						
	Area Dro Scripted Count D Linear D Segmen	opdow Tools ropdo ropdo t Drop Stamp	wn wn wn odown os						

Figure 2

API Calls

- Delphi
- C#
- VB/VBA (OLE)
 Pascal Scripting
- Pascal Scripting (OLE)
- Pascal Scripting

Delphi

Using	Iltem Object Model
1	
2	procedure main;
3	
4	planswift: IPlanSwift;
5	settings: IItem;
6	property: IPropertyObject
7	
8	planswift := coPlanSwift .Create();
9	<pre>settings := planswift getItem('\Settings');</pre>
10	property := planswift GetProperty('Hide Types Tab');
11	WriteLn (property .RResultAsBoolean())
Using	PlanSwift Object Model

4 planswift: I 5 property:IPr

6 begin

1 2 3

- 7 planswift := coPlanSwift Create();
- 8 property := planswift.GetProperty('\Settings','Hide Types Tab');
- 9 WriteLn (property.ResultAsBoolean())

end

C#

pr	rivate void Main()
{	
	PlanSwift planswift = new PlanSwift();
	IItem settings = planswift.GetItem(@"\Settings");
	IPropertyObject property = settings.GetProperty("HideTypesTab");
	<pre>console.WriteLn(property.ResultAsBoolean())</pre>
}	

Usin	g PlanSwift	Object	Model	
1				

2	private void Main()
3	- t
4	PlanSwift planswift = new PlanSwift();
5	IPropertyObject property = planswift.GetProperty(@"\Settings","HideTypesTab")
6	console.WriteLn(property.ResultAsBoolean)

VB/VBA (OLE)

Using Ilter	Using IItem Object Model	
1		
2 Sub	main()	
3	Dim planswift = CreateObject("PlanSwift9.PlanCenter")	
4	Dim settings = planswift.GetItem("\Settings")	
5	Dim property = settings.GetProperty("HideTypesTab")	
6	Console.WriteLn(property.ResultAsBoolean());	
_		

Using PlanSwift Object Model

1	
2	Main()
3	<pre>Dim planswift = CreateObject("PlanSwift9.PlanCenter")</pre>
4	<pre>Dim nameProperty = planswift.GetProperty("\Settings", "HideTy</pre>
5	Console.WriteLn(property.ResultAsBoolean)

Pascal Scripting (OLE)

Item Obje	ct Model
1 2 bec	in l
2 beg 3	settings := getItem('\Settings');
4	ShowMessage(GetResultAsBoolean(settings, 'HideTypesTab'));
enc	
Root Obje	ct Model
1	
2 beg	
3	<pre>ShowMessage(GetResultAsBoolean('\Settings', 'HideTypesTab'));</pre>
enc	

Pascal Scripting

Item Object Model

1
2 begin
<pre>3 settings := PlanSwift.getItem('\Settings');</pre>
<pre>4 property := settings.GetProperty('HideTypesTab');</pre>
5 ShowMessage (property.ResultAsBoolean);
end
Using the PlanSwift Object Model

1					
2	begin				

- 3 property := PlanSwift.GetProperty('\Settings', 'HideTypesTab');
- 4 ShowMessage(property.ResultAsBoolean);
- e

Icon

lcon

String value for the PlanSwift icon.

API Calls

- Delphi
- C#
- VB/VBA (OLE)
- Pascal Scripting (OLE)
- Pascal Scripting

Delphi

1	
2	procedure main;
3	
4	planswift: IPlanSwift;
5	settings: IItem;
6	property: IPropertyObject
7	
8	planswift := coPlanSwift .Create();
9	<pre>settings := planswift .getItem('\Settings');</pre>
0	<pre>property := planswift.GetProperty('Icon');</pre>
1	WriteLn (property.ResultAsString())

Using PlanSwift Object Model

1	
2	//or
3	procedure main;
4	var
5	planswift: IPlanSwift;
6	property:IPropertyObject;
7	<pre>begin planswift := coPlanSwift .Create();</pre>
8	<pre>property := planswift GetProperty('\Settings','Icon');</pre>
9	WriteLn (property .ResultAsString())
	end;

C#



IPropertyObject property = planswift.GetProperty(@"\Settings","Icon")

VB/VBA (OLE)

ing Iltem (Object Model
Sub n	nain ()
I	<pre>Dim planswift = CreateObject("PlanSwift9.PlanCenter")</pre>
Ι	Dim settings = planswift.GetItem("\Settings")
Ι	Dim property = settings.GetProperty("Icon")
C	Console.WriteLn(property.ResultAsString());
End S	
ng PlanSv	wift Object Model
Sub N	4ain ()
I	Dim planswift = CreateObject("PlanSwift9.PlanCenter")

- Dim nameProperty = planswift.GetProperty("\Settings", "Icon")
- 5 Console.WriteLn(property.ResultAsString
- End Sub

Pascal Scripting (OLE)

Item Object Model 1 2 begin 3 4 ShowMessage(GetResultAsString(settings, 'Icon')); end

Root Object Model

1 2	begin
3	ShowMessage(GetResultAsString('\Settings','Icon'));
	end

Pascal Scripting

Item Object Model	
4 property :=	<pre>= PlanSwift.getItem('\Settings'); = settings.GetProperty('Icon'); e(property.ResultAsString);</pre>

Using the PlanSwift Object Model

1	begin
2	property of Disperift CotProperty (1) Cottingel ITagel).
3	ShowMessage (property.ResultAsString);
4	end

InstallGUID

InstallGUID

Read-only string value that is a unique identifier for installation.

API Calls

- Delphi
- C#
 VB/VBA (OLE)
- Pascal Scripting (OLE)
- Pascal Scripting

Delphi

1	
2	procedure main;
3	
4	planswift: IPlanSwift;
5	settings: IItem;
6	property: IPropertyObject
7	
8	<pre>planswift := coPlanSwift .Create();</pre>
9	<pre>settings := planswift.getItem('\Settings');</pre>
.0	<pre>property := planswift GetProperty('InstallGUID');</pre>
.1	WriteLn (property ResultAsString())

//or
procedure main;
var
planswift: IPlanSwift;
property:IPropertyObject;
begin
planswift := coPlanSwift .Create();
<pre>property := planswift GetProperty('\Settings','InstallGUID');</pre>
WriteLn (property .ResultAsString())
end;

C#		
	4	<pre>PlanSwift planswift = new PlanSwift();</pre>
	5	IPropertyObject property = planswift.GetProperty(@"\Settings","InstallGUID")
	6	console.WriteLn(property.ResultAsString)
	}	

VB/VBA (OLE)

pr:	ivate void Main()
{	
	<pre>PlanSwift planswift = new PlanSwift();</pre>
	IItem settings = planswift.GetItem(@"\Settings");
	IPropertyObject property = settings.GetProperty("InstallGUID");
	console.WriteLn(property.ResultAsString())
}	

2		Main()

ng Ilten	n Object Model
Sub	main()
040	Dim planswift = CreateObject("PlanSwift9.PlanCenter")
	Dim settings = planswift.GetItem("\Settings")
	Dim property = settings.GetProperty("InstallGUID")
	Console.WriteLn(property.ResultAsString());
End	
ng Plan	Swift Object Model
Sub	Main()
	Dim planswift = CreateObject("PlanSwift9.PlanCenter")

- Dim nameProperty = planswift.GetProperty("\Settings", "InstallGUID"
- 5 Console.WriteLn(property.ResultAsString
- End Sub

Pascal Scripting (OLE)

Item	Item Object Model		
1			
2	begin		
3	<pre>settings := getItem('\Settings');</pre>		
4	ShowMessage(GetResultAsString(settings, 'InstallGUID'));		
	end		

Root Object Model

1	
2	begin
3	<pre>ShowMessage(GetResultAsString('\Settings', 'InstallGUID'));</pre>
-	

Pascal Scripting

Item Object Model			
1			
2 begin			
<pre>3 settings := PlanSwift.getItem('\Settings');</pre>			
<pre>4 property := settings.GetProperty('InstallGUID');</pre>			
<pre>5 ShowMessage(property.ResultAsString);</pre>			
end			

Using the PlanSwift Object Model

1	
2	begin
3	<pre>property := PlanSwift.GetProperty('\Settings','InstallGUID');</pre>
4	ShowMessage(property.ResultAsString);

JumpLastView

JumpLastView

Boolean value controlling whether to jump to the last view of a page when reopening that page (Figure 1). For example, if a particular area of a page has been zoomed in on, and then a different page opened, then, if the **Automatic jump to the last view when opening a page** box is checked, then the zoomed-in view will be visible when reopening that page. If that box has not been checked, then the user will only see the default **Fit-to-Page** view when reopening that page. Checked (true) jumps to the last view. Unchecked (false) jumps to the default view.

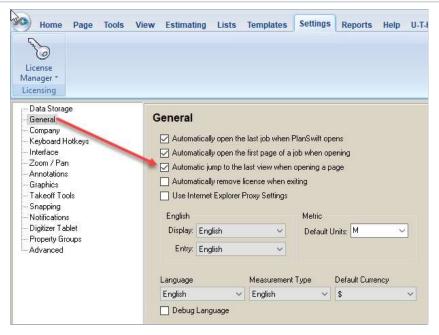


Figure 1

API Calls

- Delphi
- C#
- VB/VBA (OLE)
- Pascal Scripting (OLE)
- Pascal Scripting

Delphi



2

3 procedure main;

4 5

- planswift: IPlanSwift;
- 6 property:IPropertyObject
- 7 be
- 8 planswift := coPlanSwift Create()
- 9 property := planswift.GetProperty('\Settings','JumpLastView');
- 10 WriteLn (property.ResultAsBoolean

C#

Using IItem Object Model		
<pre>private void Main() ////////////////////////////////////</pre>		
<pre>3 { 4 PlanSwift planswift = new PlanSwift();</pre>		
<pre>5 IItem settings = planswift.GetItem(@"\Settings");</pre>		
<pre>6 IPropertyObject property = settings.GetProperty("JumpLastView");</pre>		
7 console.WriteLn(property.ResultAsBoolean())		

Using PlanSwift Object Model

1		
2	pri	vate void Main()
3	{	
4		PlanSwift planswift = new PlanSwift();
5		IPropertyObject property = planswift.GetProperty(@"\Settings","JumpLastView")
6		console.WriteLn(property.ResultAsBoolean)
-	}	

VB/VBA (OLE)

1	n Object Model
2 Sub	main()
3	Dim planswift = CreateObject("PlanSwift9.PlanCenter")
4	Dim settings = planswift.GetItem("\Settings")
5	Dim property = settings.GetProperty("JumpLastView")
6	Console.WriteLn(property.ResultAsBoolean());
-	

Using PlanSwift Object Model

Pascal Scripting (OLE)

1	
2	Sub Main()
3	Dim planswift = CreateObject("PlanSwift9.PlanCenter")
4	Dim nameProperty = planswift.GetProperty("\Settings","JumpLastView")
5	Console.WriteLn(property.ResultAsBoolean)
	End Sub

em Obj	
2 b	egin
3	<pre>settings := getItem('\Settings');</pre>
4	ShowMessage(GetResultAsBoolean(settings, 'JumpLastView'));
~	
e.	
е.	nd
	ject Model
pot Ob	

Pascal Scripting

em Object Model	
2 begin	
<pre>settings := PlanSwift.getItem('\Settings');</pre>	
<pre>property := settings.GetProperty('JumpLastView');</pre>	
5 ShowMessage(property.ResultAsBoolean);	
end	
sing the PlanSwift Object Model	
sing the PlanSwift Object Model	
sing the PlanSwift Object Model	

ShowMessage(property.ResultAsBoolean);

4 e

Language

Language

String value that displays or changes the region language: Deutsch, English, Español, Italiano, Français. It is also controlled from the Settings / General screen (see Figure 1). PlanSwift must be restarted for the change to take effect.

🔗 Home Page Tools Vi	ew Estimat	ing Lists	Templates	Settings	Reports	Help P
License Manager * Licensing						
Data Storage General Company Keyboard Holkeys Interface Zoom / Pan Annotations Graphics Takeoff Tools Snapping Notifications Digitizer Tablet Property Groups Advanced	Automa Automa Automa Automa Use Int English Display:	atically open th atic jump to the atically remove ernet Explorer	e last job when I e first page of a I last view when license when ex Proxy Settings	job when op opening a pa	ening age	v
	Language		Measurement	Туре	Default Curre	ncy
	English Deutsch English Español Italiano Français	v	English	~	\$	×

Figure 1

API Calls

- Delphi
- C#
- VB/VBA (OLE)
- Pascal Scripting (OLE)
- Pascal Scripting

Delphi

Using Iltem Object Model				
1				
2 procedure main;				
3 var				
4 planswift: IPlanSwift;				
5 settings: IItem;				
6 property: IPropertyObject				
7 begin				
<pre>8 planswift := coPlanSwift .Create();</pre>				
<pre>9 settings := planswift.getItem('\Settings');</pre>				
<pre>10 property := planswift GetProperty('Language');</pre>				
<pre>11 WriteLn (property ResultAsString())</pre>				
end				
Using PlanSwift Object Model				

2 //or 3 procedure main;

4	<pre>var planswift: IPlanSwift;</pre>
5	property.IPropertyObject:

C#

Using Iltem Object Model
<pre>private void Main() { FlanSwift planswift = new PlanSwift(); IItem settings = planswift.GetItem(@"\Settings"); IPropertyObject property = settings.GetProperty("Language"); console.WriteLn(property.ResultAsString()) } </pre>
Using PlanSwift Object Model
<pre>private void Main() 3 {</pre>

VB/VBA (OLE)

Usin	g lltem	Object Model
1	Cu-la	
2		main()
3		Dim planswift = CreateObject("PlanSwift9.PlanCenter")
4		Dim settings = planswift.GetItem("\Settings")
5		Dim property = settings.GetProperty("Language")
6		Console.WriteLn(property.ResultAsString());
	End	

Using PlanSwift Object Model

1		
2	Sub	Main()
3		Dim planswift = CreateObject("PlanSwift9.PlanCenter")
4		Dim nameProperty = planswift.GetProperty("\Settings","Language")
5		Console.WriteLn(property.ResultAsString)
0	End	

Item Object Model

1	havia
2	<pre>begin settings := getItem('\Settings');</pre>
2	ShowMessage(GetResultAsString(settings, 'Language'));
4	end
Root	Object Model
1	
2	begin
3	<pre>ShowMessage(GetResultAsString('\Settings','Language'));</pre>

Pascal Scripting (OLE) Pascal Scripting

Item Object Model

1	
2 begin	
<pre>3 settings := PlanSwift.getItem('\Settings');</pre>	
<pre>4 property := settings.GetProperty('Language');</pre>	
5 ShowMessage(property.ResultAsString);	
end	
Using the PlanSwift Object Model	

1 2 ^{be} 3 4

- property := PlanSwift.GetProperty('\Settings', 'Language');
- ShowMessage(property.ResultAsString);
- €

LastReportValidUntil

LastReportValidUntil

String value that the sets the number of days the last report is valid for.

API Calls

- Delphi
- C#
- VB/VBA (OLE)
- Pascal Scripting (OLE)
- Pascal Scripting

Delphi

Using Iltern Object Model procedure main; var planswift: IPlanSwift; settings: IItem; property: IPropertyObject begin planswift := coPlanSwift .Create(); settings := planswift .getItem('\Settings'); property := planswift .GetProperty('LastReportValidUntil'); WriteLn(property.ResultAsString()) end

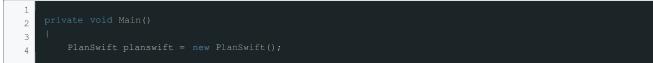
Using PlanSwift Object Model



C#

Using Iltem Object Model				
1 2 private void Main()				
<pre>3 { 4 PlanSwift planswift = new PlanSwift();</pre>				
<pre>5 IItem settings = planswift.GetItem(@"\Settings");</pre>				
<pre>IPropertyObject property = settings.GetProperty("LastReportValidUntil");</pre>				
<pre>7 console.WriteLn(property.ResultAsString())</pre>				

Using PlanSwift Object Model



VB/VBA (OLE)

ing Ilten	n Object Model
1	
2 Sub	main()
3	Dim planswift = CreateObject("PlanSwift9.PlanCenter")
	Dim settings = planswift.GetItem("\Settings")
	Dim property = settings.GetProperty("LastReportValidUntil")
	Console.WriteLn(property.ResultAsString());
ing Plan	Swift Object Model
1	
2 Sub	Main()
3	Dim planswift = CreateObject("PlanSwift9.PlanCenter")

- Dim nameProperty = planswift.GetProperty("\Settings", "LastReportValidUntil")
- 5 Console.WriteLn(property.ResultAsString)
- End Sub

Pascal Scripting (OLE)

Iten	n Object Model	
1 2	begin	
3	<pre>settings := getItem('\Settings');</pre>	
4	ShowMessage(GetResultAsString(settings,	<pre>/LastReportValidUntil'));</pre>
	end	

Root Object Model

1	
2	begin
3	ShowMessage(GetResultAsString('\Settings','LastReportValidUntil'));

Pascal Scripting

Item	Object Model
1	
2	begin
3	<pre>settings := PlanSwift.getItem('\Settings');</pre>
4	<pre>property := settings.GetProperty('LastReportValidUntil');</pre>
5	ShowMessage(property.ResultAsString);

Using the PlanSwift Object Model

1 2	begin
3	<pre>property := PlanSwift.GetProperty('\Settings', 'LastReportValidUntil'); ShowMessage(property.ResultAsString); end</pre>

Left

Left

Read-only integer value that displays the left position of the Main Window.

API Calls

- Delphi
- C#
- VB/VBA (OLE)
- Pascal Scripting (OLE)
- Pascal Scripting

IPropertyObject property = planswift.GetProperty(@"\Settings","Left" console.WriteLn(property.ResultAsInteger)

Delphi


```
grocedure main;
var
planswift: IPlanSwift;
property:IPropertyObject;
begin
planswift := coPlanSwift .Create();
property := planswift .GetProperty('\Settings','Left');
WriteLn (property.ResultAsInteger())
end;
```

C#

р	.vate void Main()
{	
	PlanSwift planswift = new PlanSwift();
	IItem settings = planswift.GetItem(@"\Settings");
	IPropertyObject property = settings.GetProperty("Left");
	console.WriteLn(property.ResultAsInteger())
}	

1 2 private void Main() 3 { 4 PlanSwift planswift = new PlanSwift();

VB/VBA (OLE)

Sub main()					
	answift = Create	bject("PlanSwift			
	tings = planswi	t.GetItem("\Sett			
Dim pr	operty = setting	.GetProperty("Le	ft")		
Consol	.WriteLn(proper	y.ResultAsInteger	c());		
g PlanSwift Ob	ect Model				

- Dim nameProperty = planswift.GetProperty("\Settings", "Left")
- 5 Console.WriteLn(property.ResultAsInteger)
- End Sub

Pascal Scripting (OLE)

Item Object Model

1	
2	begin
3	<pre>settings := getItem('\Settings');</pre>
4	ShowMessage(GetResultAsInteger(settings, 'Left'));
	end

Root Object Model

1	
2	begin
3	<pre>ShowMessage(GetResultAsInteger('\Settings','Left'));</pre>
	end

Pascal Scripting

Item Object Model	
<pre>1 2 begin 3 settings := PlanSwift.getItem('\Settings'); 4 property := settings.GetProperty('Left'); 5 ShowMessage(property.ResultAsInteger); end</pre>	

Using the PlanSwift Object Model	
1	
2 begin	
<pre>3 property := PlanSwift.GetProperty('\Settings', 'Left');</pre>	
4 ShowMessage (property.ResultAsInteger);	
end	

MagnifierHotKey

MagnifierHotKey

Integer value that returns an ANSI key code (default code 77). The (default hotkey) letter M invokes the Magnifier command in PlanSwift (Figure 1).

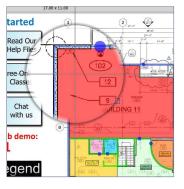


Figure 1

API Calls

- Delphi
- C#
- VB/VBA (OLE)
- Pascal Scripting (OLE)
- Pascal Scripting

Delphi

Using	lltem Object Model
1	
2	procedure main;
3	var
4	planswift: IPlanSwift;
5	settings: IItem;
6	property: IPropertyObject
7	begin
8	planswift := coPlanSwift .Create();
9	<pre>settings := planswift getItem('\Settings');</pre>
10	property := planswift GetProperty('MagnifierHotKey');
11	WriteLn (property.ResultAsInteger())
	end

Using PlanSwift Object Model



1 2 3 4 7 PlanSwift planswift = new PlanSwift(); 1Item settings = planswift.GetItem(@"\Settings"); IPropertyObject property = settings.GetProperty("MagnifierHotKey"); console.WriteLn(property.ResultAsInteger()) } Using PlanSwift Object Model 1 2 private void Main()

2	private void Main()
3	
4	PlanSwift planswift = new PlanSwift();
5	IPropertyObject property = planswift.GetProperty(@"\Settings","MagnifierHotKey")
6	console.WriteIn(property.ResultAsInteger)

VB/VBA (OLE)

ng Iltem Object N	
Sub main()	
Dim pla	answift = CreateObject("PlanSwift9.PlanCenter")
Dim set	ctings = planswift.GetItem("\Settings")
Dim pro	<pre>operty = settings.GetProperty("MagnifierHotKey")</pre>
Console	<pre>%.WriteLn(property.ResultAsInteger());</pre>
End Sub	

Using PlanSwift Object Model

1		
2	Sub	Main()
3		Dim planswift = CreateObject("PlanSwift9.PlanCenter")
4		Dim nameProperty = planswift.GetProperty("\Settings","MagnifierHotKey")
5		Console.WriteLn(property.ResultAsInteger)

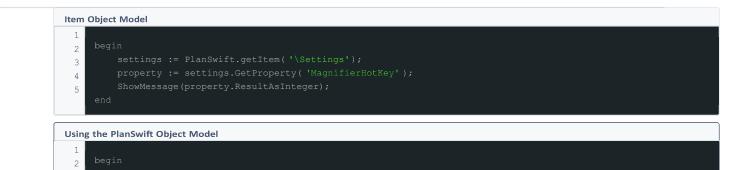
Pascal Scripting (OLE)

begi	in		
	<pre>settings := getItem('\Settings');</pre>		
	ShowMessage (GetResultAsInteger (settings, 'Ma	agnifierHotKey'));	
end			

Root Object Mode

-	
2	begin
3	ShowMessage(GetResultAsInteger('\Settings'

Pascal Scripting



4

- property := PlanSwift.GetProperty('\Settings', 'MagnifierHotKey'); ShowMessage(property.ResultAsInteger);

MeasurementEntry

MeasurementEntry

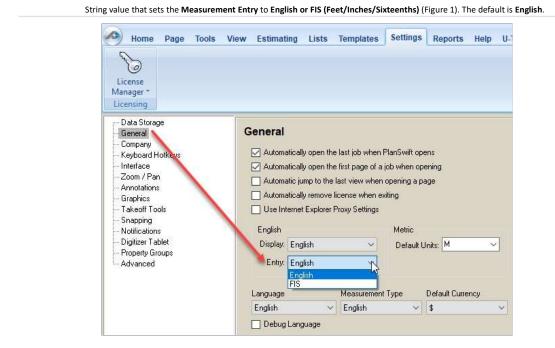


Figure 1

API Calls

Delphi

- C#
- VB/VBA (OLE)
- Pascal Scripting (OLE)
- Pascal Scripting

. Delphi

Using Iltem Object Model



Using PlanSwift Object Model

```
1

2 //or

3 procedure main;

4 var

5 planswift: IPlanSwift;

6 property:IPropertyObject
```

- 8 planswift := coPlanSwift .Create()
- 9 property := planswift.GetProperty('\Settings','MeasurementEntry');
- 10 WriteLn (property.ResultAsString())

e

C#

1	n Object Model
2 pri	vate void Main()
3 {	
4	<pre>PlanSwift planswift = new PlanSwift();</pre>
5	IItem settings = planswift.GetItem(@"\Settings");
6	IPropertyObject property = settings.GetProperty("MeasurementEntry");
7	console.WriteLn(property.ResultAsString())
<i>`</i> }	

1	
2	private void Main()
3	
4	PlanSwift planswift = new PlanSwift();
5	IPropertyObject property = planswift.GetProperty(@"\Settings","MeasurementEntry")
6	console.WriteIn(property.ResultAsString)

VB/VBA (OLE)

Usin	g Iltem Object Model
1	
2	Sub main()
3	Dim planswift = CreateObject("PlanSwift9.PlanCenter")
4	Dim settings = planswift.GetItem("\Settings")
5	Dim property = settings.GetProperty("MeasurementEntry")
6	Console.WriteLn(property.ResultAsString());
Ť	End Sub

Using PlanSwift Object Model

1		
2	Sub	Main()
3		Dim planswift = CreateObject("PlanSwift9.PlanCenter")
4		Dim nameProperty = planswift.GetProperty("\Settings","MeasurementEntry")
5		Console.WriteLn(property.ResultAsString)

Pascal Scripting (OLE)

em Object M 1 2 begin 3 s		
	howMessage(GetResultAsString(settings,	'MeasurementEntry'));

1	
2	begin
3	<pre>ShowMessage(GetResultAsString('\Settings', 'MeasurementEntry'));</pre>
	end



end

Name

Name

String value containing the property name.

API Calls

- Delphi
- C#
- VB/VBA (OLE)
- Pascal Scripting (OLE)
- Pascal Scripting

Delphi

Ising	lltem Object Model
1 2 3 4 5 6	procedure main; var planswift: IPlanSwift; settings: IItem; property: IPropertyObject
7 8 9 10 11	<pre>begin planswift := coPlanSwift .Create(); settings := planswift .getItem('\Settings'); property := planswift .GetProperty('Name'); WriteLn(property.ResultAsString()) end</pre>
Jsing	PlanSwift Object Model
2 3	//or procedure main;
4 5 6	var planswift: IPlanSwift; property:IPropertyObject;
7	<pre>begin planswift := coPlanSwift .Create(); property := planswift .GetProperty('\Settings','Name');</pre>

9

C#

1	ltem Object Model
2 1	orivate void Main()
3	
4	<pre>PlanSwift planswift = new PlanSwift();</pre>
5	<pre>IItem settings = planswift.GetItem(@"\Settings");</pre>
6	IPropertyObject property = settings.GetProperty("Name");
7	console.WriteLn(property.ResultAsString())
	1
Ising P	lanSwift Object Model
Ising P	PlanSwift Object Model
1	
1	/ PlanSwift Object Model oprivate void Main() {

VB/VBA (OLE)

ng Iltem Object Model
Sub main()
Dim planswift = CreateObject("PlanSwift9.PlanCenter")
Dim settings = planswift.GetItem("\Settings")
Dim property = settings.GetProperty("Name")
Console.WriteLn(property.ResultAsString());
End Sub
ng PlanSwift Object Model

1	
2	Sub Main()
3	Dim planswift = CreateObject("PlanSwift9.PlanCenter")
4	Dim nameProperty = planswift.GetProperty("\Settings","Name")
5	
	End Sub

Pascal Scripting (OLE)

Item Object Model	
<pre>1 2 begin 3 settings := getItem('\Settings'); 4 ShowMessage(GetResultAsString(settings, ' end</pre>	Name'));
Root Object Model	
1 2 begin	

ShowMessage(GetResultAsString('\Settings', 'Name'));

Pascal Scripting

3

Item	Object Model
1 2 3	begin settings := PlanSwift.getItem('\Settings');
4 5	<pre>property := settings.GetProperty('Name'); ShowMessage(property.ResultAsString); end</pre>

Using the PlanSwift Object Model

1	
2	begin
3	<pre>property := PlanSwift.GetProperty('\Settings','Name');</pre>
4	<pre>ShowMessage(property.ResultAsString);</pre>
-	

NewAreaHotKey

NewAreaHotKey

Integer value returns an ANSI key code (default code 49, the number 1) for the New Area hotkey. Figure 1 shows where the New Area hotkey is assigned. Figure 2 shows where New Area is invoked on the Main Ribbon Bar.

Item Object Model

Using the PlanSwift Object Model

icensing						
- Data Storage - General - Company	Keyboard Hotkeys					
Keyboard Hotkeys		Primary	Secondary		Prinary	Secondar
Interface Zoots / Pan	Cost in Fit	F7	None	Smooth Zoom In:	Num +	None
Annotations	Zoom	F10	None	Smooth Zoom Dut	Num -	None
- Graphics - Takeoff Tools	Previous Zoom	None	None	New Area	1	
- Takeon Loois - Snapoing				New Linear:	2	
Notifications	S cooll Up:		None	New Segment:	3	
Digitizer Tablet	Scroll Down:		None	New Count	4	
Property Groups Advanced	Scroll Left:		None			
	Scroll Right:	F	None	Start/Stop Point Recording:	R	
	Magnifier:		None	New Section:	N	
				Toggle Image/Estimating Window:	F12	
	Create Bookmark:		None	New Note:		
	Search Bookmarks:		None			
	Continue With:		None			

Figure 1



Figure 2

API Calls

Delphi

Using Iltem Object Model	
Using PlanSwift Object Model	

C#

Using Iltem Object Model	
Using PlanSwift Object Model	

VB/VBA (OLE)

Using Iltem Object Model	
Using PlanSwift Object Model	

Pascal Scripting (OLE)

Item Object Model	
Root Object Model	

Pascal Scripting

NewCountHotKey

NewCountHotKey

Integer value returns an ANSI key code (default code 52, the number 4) for the New Count hotkey. Figure 1 shows where the New Count hotkey is assigned. Figure 2 shows where New Count is invoked on the Main Ribbon Bar.

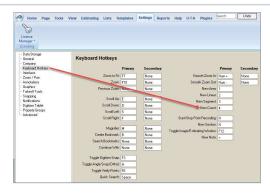


Figure 1

	Home	Page	Tools	View	Estimatir	ig List	s Ter	nplates	Settin	gs Re	ports	Help U-T-H	Plugin	s Searc	th _	Undo]
۲	10	2				Q	.	Ð,	Q	\bigcirc	17.			In.	R	-	*
New	Open	Print	Email Job	Back	Fwd	Zoom	Fit Page	Zoom In	Zoom Out	Pan	Scale	Dimension	Area	Linear	Segment	Count	Auto Count
	J	ob		Navi	igate		Z	oom / Pa	n		N	leasure				Takeoff	

Figure 2

API Calls

elphi	
Using Iltem Object Model	Expand source
Using PlanSwift Object Model	> Expand sourc
2	
Using Iltem Object Model	Expand source
Using PlanSwift Object Model	> Expand sourc
B/VBA (OLE)	
Using Iltem Object Model	Expand source
Using PlanSwift Object Model	> Expand sourc
ascal Scripting (OLE)	
	Expand source
Item Object Model	

Pascal Scripting

Using the PlanSwift Object Model

📜 Unknown macro: 'sp-tabs'

>

NewLinearHotKey

NewLinearHotKey

Integer value returns an ANSI key code (default code 50, the number 2) for a New Linear hotkey. Figure 1 shows where the New Linear hotkey is assigned. Figure 2 shows where New Linear is invoked on the Main Ribbon Bar.

Home Page Tools	View Estimating Lists Te	mplates	Settings Rep	orts Help U-T-H Plugins	Search	Undo
License anager * censing						
Data Storage General Company	Keyboard Hotkeys					
Keyboard Holkeys		Primary	Secondary		Primary	Secondary
Zoom / Pan	Zoom to Fit	F7	None	Smooth Zoom In:	Num +	None
Annotations	Zoom	F10	None	Smooth Zoom Out:	Nun -	None
Graphics Takeoff Tools	Previous Zoom	None	None	New Area		
Snapping	Sciol Up:	-	None	New Linear:		
Notifications Digitizer Tablet	Sciol Down		None	New Segment:		
Property Groups	Scallet		None	New Count	4	
Advanced	Scroll Right		None	Sale/Stop Point Recording	P	
			India	New Section		
	Megnitien		None	Toggie Image/Estimating Window.		
	Create Bookmark:		None	New Note:		
	Search Bookmarks:		None			
	Continue With	None	None			
	Toggle Digitizer Snap	F3				
	Toggle Angle Snap (Ortho)					
	Toode Veniv Points					
	Quick Search:	Casar				

Figure 1



Figure 2

API Calls

Delphi

Using Iltem Object Model	> Expand source
Using PlanSwift Object Model	Expand source
C#	
Using Iltem Object Model	> Expand source
Using PlanSwift Object Model	Expand source
VB/VBA (OLE)	
Using Iltem Object Model	> Expand source
Using PlanSwift Object Model	Expand source
Pascal Scripting (OLE)	
Item Object Model	Expand source
Root Object Model	> Expand source

Pascal Scripting Using the PlanSwift Object Model

NewNoteHotKey

NewNoteHotKey

Integer value returns an ANSI key code (default code 187, the character =) for the **New Note** hotkey. Figure 1 shows the **Note** tool on the **Main Menu** ribbon bar. Figure 2 shows how a **Note** is created: click on **Note** (or the hotkey for **Note**), then click and drag in the area of the plan where you want the note to be inserted.

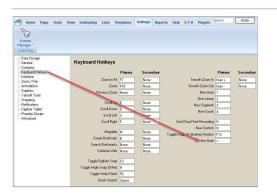


Figure 1

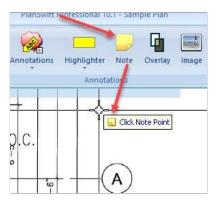


Figure 2

API Calls

Delphi

> Expand source
Expand source
> Expand source
> Expand source
Expand source
> Expand source
Expand source

Root Object Model

>

>	Ex	par	١d	soι	irce
---	----	-----	----	-----	------

Pascal Scripting

Item Object Model

Using the PlanSwift Object Model

Expand source

Expand source

NewSectionHotKey

NewSectionHotKey

Integer value returns an ANSI key code (default code 78, the letter **N**) for the **New Section** hotkey. Figure 1 shows where the **New Section** hotkey assignment is made. Figure 2 shows where the **New Section** command is invoked on the PlanSwift Main Menu ribbon bar. Figure 3 shows where **New Section** may be invoked by right-clicking on a selected selection.

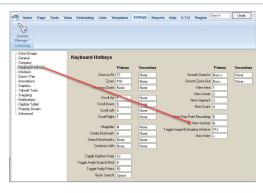


Figure 1







Figure 3

API Calls

Delphi	
Using IItem Object Model	> Expand source
Using PlanSwift Object Model	> Expand source
C#	

Using IItem Object Model	> Expand source
Using PlanSwift Object Model	Expand source

>

VB/VBA (OLE)

Using Iltem Object Model	> Expand source
Using PlanSwift Object Model	> Expand source

Pascal Scripting (OLE)

Item Object Model	> Expand source
Root Object Model	> Expand source
Pascal Scripting	
Item Object Model	> Expand source
Using the PlanSwift Object Model	> Expand source

NewSegmentHotKey

NewSegmentHotKey

Integer value returns an ANSI key code (default code 51, the number 3) for the New Segment hotkey. Figure 1 shows where the New Segment hotkey is assigned. Figure 2 shows where New Segment is invoked on the Main Ribbon Bar.

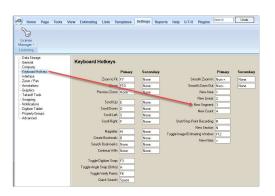


Figure 1



Figure 2

API Calls

Delphi

Using IItem Object Model	Expand source
Using PlanSwift Object Model	Expand source
C#	
Using Iltem Object Model	Expand source
Using PlanSwift Object Model	> Expand source
VB/VBA (OLE)	
Using Iltem Object Model	> Expand source
Using PlanSwift Object Model	> Expand source
Pascal Scripting (OLE)	
Item Object Model	> Expand source
Root Object Model	> Expand source
Pascal Scripting	
Item Object Model	> Expand source
Using the PlanSwift Object Model	> Expand source

NoAskBreakInheritance

NoAskBreakInheritance

Boolean value that allows selecting whether a confirmation box should be displayed asking the user to confirm an **Inheritance Break**. Checking the box (true) will cause the confirmation box not to appear. Unchecking the box (false) causes the confirmation box to appear. This boolean value is also controlled when a template's advanced properties are edited in the **Settings / Notifications** window (Figure 1).

To see a confirmation box:

Click on the Estimating tab on the Main Ribbon menu.

Double click on an Estimating template (Figure 2) to show the Template's Properties window

Click on Advanced.

In the Advanced window (Figure 3), click on the formula cell for the Linear Total takeoff and wait a second

Click on the same cell again. This opens the window asking Editing this property will cause it to no longer be inherited. Do you want to continue?

Click on the **Do not ask again** box in order not to be asked the question again: Clicking on the **Do Not ask again** box sets the **NoAskBreakInheritance** boolean value to true.



Figure 1

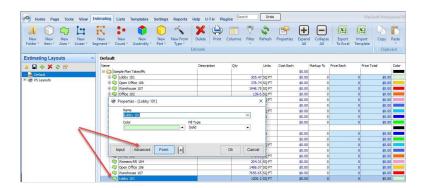


Figure 2

Home Page Tools View New New New New New Folder * Item * Ares * Lines *	New New New Ne	New From Delete	rint Columns Filter	Refresh Prop		Collapse Export Import All To Excel Templat	e Cipboard		djust	
Estimating Layouts	« Default									
A B + X & C Default © Ø PS Layouts	Name Sample Plan Takeoffs () Lobby 101	 Ø Properties - [Lobby 101] 	- * / ()	0 • २ r		Show: Normal + 🛅 Form	Events +			>
0100000	Open Office 106	Name	Formula			Adjust Result	Cutput Units	TT		1 0
	Warehouse 107 Office 102	Takeoff Data			1					1
	Womens RR 104	Depth	T		FT	0.00	FT	8		
	Im Exterior Walls	Volume	[Area] * [Depth]	4	CUFT	0.00	CUFT	8		
	Hens RR 105	Wal Height	freed freedom		ET.	0.00	FT	8		
	Doors	Wal Area	[Unear Total] * (Wall H		SO FT	0.00	SO FT	8		
	Interior Walls	Takeoff	[Area]		SO FT	1,200,20	SOFT	8		
	🕀 🧰 Residential Plan Takeoffs							8	10	
	Doors	Area	[!Area]		SQ FT	1,200.20	SQ FT	6	10	
	Triterior Walls	Linear Total	[!Linear Total]	PlanSwift.exe	FT	107.77	FT	8	20	
	Mens RR 105	Segment Count	[ISegment Count]	Planswittleve	•			8		
	Exterior Walls	Point Count	[IPoint Count]	A Editing	this property will a	ause it to no longer be	EA	8		
	Office 102	Default	[Takeoff]	inherite	ed.	ause is to no longer be		8		
	Womens RR 104	Section Count	Rection Countil	Do you	want to continue			8	in	
	Open Office 106	Scale Units	FT	-				8	án	
	Warehouse 107	Work Breakdown Struct		Do not asl	a nomin	Yes No		19.14	F 1-1	
	Lobby 101	WORK Breakdown Struct	ure		c uguni			e (

Figure 3

API Calls

Delphi

Using Iltem Object Model	> Expand source
Using PlanSwift Object Model	> Expand source

C#

Using IItem Object Model	> Expand source
Using PlanSwift Object Model	> Expand source

VB/VBA (OLE)

Using Iltem Object Model	Expand source
Using PlanSwift Object Model	> Expand source

Pascal Scripting (OLE)

Item Object Model	Expand source
Root Object Model	> Expand source

Pascal Scripting

Item Object Model	> Expand source
Using the PlanSwift Object Model	> Expand source

NoAskCopyPoints

NoAskCopyPoints

Boolean value that allows selecting whether a confirmation box should be displayed asking the user whether the digitizer points should be copied when copying a takeoff. Checking the box (true) will cause the confirmation box not to appear. Unchecking the box (false) causes the confirmation box to appear. This is also controlled in the Settings / Notifications window (Figure 1). The confirmation box is shown in Figure 2. Selecting "Do Not Ask Again" from this confirmation window to confirm that the digitizer points should be copied when copying a Takeoff also sets the NoAskCopyPoints boolean variable to true.

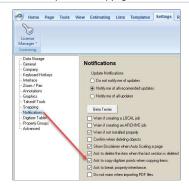


Figure 1

Using IItem Object Model	> Expand source
Using PlanSwift Object Model	> Expand source

		Estimate			AI
Estimating Layouts **	Default				
1 □ ÷ X ≎ B	Name	Description	QRY	Units	Cost Each
Defailt	(f) (2) Lobby 301		305.47	SOFT	\$0.0
@ PS Layouts	(i) Copen Office 106		378.74	SQFT	\$0.0
	iii 😳 Warehouse 107		1948.75	SQFT	\$0.0
	8 (g) Office 302		179.6	SQFT	\$0.0
	(i) 😳 Womens RR 104		52	SQIT	\$0.0
	(i) In Exterior Wals		192.77	FT	\$0.0
	(i) 😳 Mens RR 105		51.50	SQIFT	\$0.0
	(i) . Doors		8	EA	\$0.0
	(i) K Interior Walls		135.55	FT	\$0.0
	(i) C Residential Plan Takeoffs				
	Deers		8	EA	\$0.0
	Interior Walls		268.5	FT	\$0.0
	Mens RR 105		202.64	SQ FT	\$0.0
	In Exterior Wals		321.95	FT	\$0.0
	Office 102		548.51	SQ FT	\$0.0
	Womens RR 104		204.31	SQ FT	\$0.0
	- Open Office 106		1488.07	SQ FT	\$0.0
	Warehouse 107		7656.65	SQ FT	\$0.0
	🖓 (stby 9)1 🥒		1200.2	SOFT	\$0.0

Figure 2

API Calls

Delphi

Using Iltem Object Model	> Expand source
Using PlanSwift Object Model	> Expand source
C#	
Using Iltem Object Model	> Expand source
Using PlanSwift Object Model	> Expand source

VB/VBA (OLE)

Pascal Scripting	(OLE)
------------------	-------

Item Object Model	> Expand source
Pascal Scripting	
Item Object Model	Expand source
Using the PlanSwift Object Model	> Expand source

> Expand source

NoAskDeleteItemOnLastObject

NoAskDeleteItemOnLastObject

Boolean value that allows or disallows the display of a **Delete Items** confirmation popup screen to ask whether to delete the last section of a multi-section item. When the box is checked (true) in the **Notifications** area (Figure 1), the display of the **Delete Items** confirmation window is enabled. When the box is unchecked (false), it is disabled. Clicking the **Do not ask again** box in the **Delete Items** window disables the display of the **Delete Items** window. Leaving it unchecked, allows it to be displayed (Figure 2).





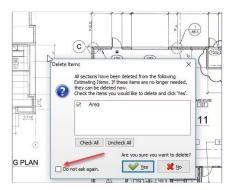


Figure 2

API Calls

C	elphi	
	Using IItem Object Model	> Expand source
	Using PlanSwift Object Model	> Expand source

C#

Using Iltem Object Model	Expand source
Using PlanSwift Object Model	Expand source
VB/VBA (OLE)	
Using Iltem Object Model	Expand source

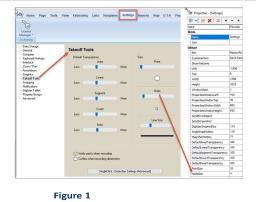
Pascal Scripting (OLE) Pascal Scripting

Item Object Model

Item Object Model	> Expand source
Using the PlanSwift Object Model	> Expand source

NodeSize

NodeSize



Integer value that sets the **Node Size**. Values range from 0 to 20, with 10 being the default. Figure 1 shows the corresponding screen areas of the **Settings** screen and the **U-T-H** area.

API Calls

Delphi

Using Iltem Object Model > Expand source Using PlanSwift Object Model > Expand source VB/VBA (OLE) > Expand source Using Iltem Object Model > Expand source Using PlanSwift Object Model > Expand source Pascal Scripting (OLE) > Expand source Item Object Model > Expand source Root Object Model > Expand source		
C# Using Iltem Object Model Expand source Using PlanSwift Object Model Expand source Using Iltem Object Model Using Iltem Object Model Expand source Using PlanSwift Object Model Expand source Pascal Scripting (OLE) Item Object Model Expand source Root Object Model Expand source Pascal Scripting Expand source Pascal Scripting Expand source Expand sour	Using IItem Object Model	Expand source
Using Iltem Object Model > Expand source Using PlanSwift Object Model > Expand source VB/VBA (OLE)	Using PlanSwift Object Model	> Expand source
Using Iltem Object Model > Expand source Using PlanSwift Object Model > Expand source VB/VBA (OLE)		
Using PlanSwift Object Model > Expand source VB/VBA (OLE) > Expand source Using Iltem Object Model > Expand source Pascal Scripting (OLE) > Expand source Item Object Model > Expand source Pascal Scripting > Expand source	C#	
VB/VBA (OLE) Using Iltem Object Model * Expand source Using PlanSwift Object Model * Expand source Pascal Scripting (OLE) Item Object Model * Expand source Pascal Scripting Item Object Model * Expand source	Using IItem Object Model	Expand source
Using Iltem Object Model > Expand source Using PlanSwift Object Model > Expand source Pascal Scripting (OLE) > Expand source Root Object Model > Expand source Pascal Scripting > Expand source Pascal Scripting > Expand source	Using PlanSwift Object Model	Expand source
Using Iltem Object Model > Expand source Using PlanSwift Object Model > Expand source Pascal Scripting (OLE) > Expand source Root Object Model > Expand source Pascal Scripting > Expand source Pascal Scripting > Expand source		
Using PlanSwift Object Model > Expand source Pascal Scripting (OLE) Item Object Model > Expand source Root Object Model > Expand source Pascal Scripting Item Object Model > Expand source	VB/VBA (OLE)	
Pascal Scripting (OLE) Item Object Model > Expand source Root Object Model > Expand source Pascal Scripting	Using IItem Object Model	Expand source
Item Object Model > Expand source Root Object Model > Expand source Pascal Scripting > Expand source Item Object Model > Expand source	Using PlanSwift Object Model	Expand source
Root Object Model > Expand source Pascal Scripting > Expand source Item Object Model > Expand source	Pascal Scripting (OLE)	
Pascal Scripting Item Object Model Expand source	Item Object Model	Expand source
Item Object Model > Expand source	Root Object Model	> Expand source
	Pascal Scripting	
Using the PlanSwift Object Model > Expand source	Item Object Model	> Expand source
Using the PlanSwift Object Model Capacity Sector Se		
	Using the PlanSwift Object Model	> Expand source

Ortho

Ortho

Boolean value that toggles **Ortho** on or off. Checked is true (on) and enables it (highlighting) the **Ortho** toggle control. Unchecked is false (off) and disables (unhighlights) **Ortho**. **Ortho** is also controlled in the **Main / Settings** window (Figure 1). If **Ortho** is disabled, then **Smart Ortho** (right below it) is also disabled, although in the **U-T-H Settings Advanced Properties, Smart Ortho** will show as enabled even though **Ortho** is disabled. **Ortho** may also be toggled on (highlighted) and off (un-highlighted) at the bottom of the PlanSwift main window (Figure 2).

Home	Page	Tools	View	Estimating	Lists	Templates	Settings	Reports
License Manager * Licensing								
Data Storag General Company Keyboard Hi Interface Zoom / Pan Annotations Graphics Takeoff Toc Snapping Notifications Digitzer Tab Property Gric Advanced	otkeys Is		s	_	o PlanSwil o CAD poi to Snap A up to angle	t points nts		able/disable

Figure 1



Figure 2

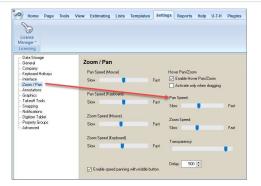
API Calls

Delphi	
Using Iltem Object Model	> Expand source
Using PlanSwift Object Model	> Expand source
C#	
Using Iltem Object Model	Expand source
Using PlanSwift Object Model	> Expand source
VB/VBA (OLE)	
Using Iltem Object Model	> Expand source
Using PlanSwift Object Model	> Expand source
Pascal Scripting (OLE)	
Item Object Model	> Expand source
Root Object Model	> Expand source
Pascal Scripting	
Item Object Model	Expand source
Using the PlanSwift Object Model	> Expand source

PanHoverSpeed

PanHoverSpeed

Integer value that controls the **Hover Pan Speed** (Figure 1). Value ranges from 5 to 100. Figure 2 shows the light blue transparencies (and darker blue arrows) at the edges of the window and the darker triangular transparencies (arrows) in the corners. Hovering (not clicking) your mouse in any of those blue areas makes the plan scroll quickly in the direction of the arrow. Pressing the keyboard space bar reverses the scrolling direction.





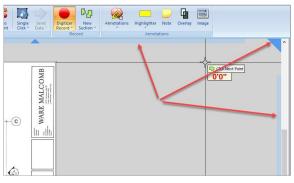


Figure 2

API Calls

Delphi	
Using IItem Object Model	> Expand source
Using PlanSwift Object Model	> Expand source

C#

Using IItem Object Model	> Expand source
Using PlanSwift Object Model	> Expand source

VB/VBA (OLE)

Using IItem Object Model	> Expand source
Using PlanSwift Object Model	> Expand source

Pascal Scripting (OLE)

Pascal	Scri	nting

Item Object Model	Expand source
Using the PlanSwift Object Model	Expand source
Item Object Model	> Expand source
Root Object Model	> Expand source

PanSpeed

PanSpeed

Integer value that controls the mouse Pan Speed. Value ranges from 1 to 9 (see Figure 1).

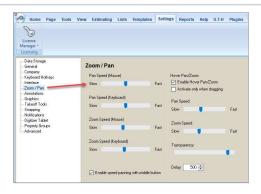


Figure 1

API Calls

Delphi

Using Iltem Object Model	Expand source
Using PlanSwift Object Model	> Expand source
C#	
Using Iltem Object Model	Expand source
Using PlanSwift Object Model	> Expand source
VB/VBA (OLE)	
Using Iltem Object Model	> Expand source
Using PlanSwift Object Model	> Expand source
Pascal Scripting (OLE)	
Item Object Model	Expand source
Root Object Model	> Expand source
Pascal Scripting	
Item Object Model	> Expand source
Using the PlanSwift Object Model	> Expand source
PanSpeedKeyboard	

PanSpeedkeyboard

PanSpeedKeyboard

Integer value that controls the keyboard Pan Speed. Value ranges from 1 to 250 (see Figure 1). Keyboard keys E, S, D, and F pan the image Up, Left, Down, and Right, respectively.

0					
License					
Manager * Licensing					
	1				
- Data Storage	Zoom / Pa	in			
Company				Hover Pan/Zoom	
- Keyboard Holkeys	Pan Speed (Mouse)			
- Interface - Zoom / Pan	Slow		Fast	Enable Hover Pan/Z	
- Accolations				Activate only when d	kagging
- Graphics	Pan Speed (Keyboard)			
- Takeoff Tools	Slow		East	Pan Speed	
- Snapping				Slow	Fast
 Notifications Digitizer Tablet 	Zoom Speed	[Mercen]			
- Digitizer Laber - Property Groups		((((0000))		Zoom Speed	
Advanced	Slow		Fast	Slow	Fast
	Zoom Speed	(Keyboard)		Transparency.	
	Slow		Fast	indropalency.	

Figure 1

API Calls

> Expand source
Expand source
> Expand source
> Expand source
> Expand source
Expand source

Pascal Scripting (OLE)

Pascal Scripting

Item Object Model	> Expand source
Root Object Model	> Expand source

Item Object Model	> Expand source
Using the PlanSwift Object Model	Expand source

PanZoomHover

PanZoomHover

Boolean value enabling or disabling **Hover Pan/Zoom** (see Figure 1). Checked (true) enables **Hover Pan/Zoom**. Unchecked (false) disables it. Figure 2 shows the light blue transparencies (and darker blue arrows) at the edges of the window and the darker triangular transparencies (arrows) in the corners. Hovering (not clicking) your mouse in any of those blue areas makes the plan scroll quickly in the direction of the arrow. Pressing the keyboard space bar reverses the scrolling direction.

8					
License					
Manager * Licensing					
- Data Storage					
- General	Zoom / Pan				
 Company Keyboard Holkeys 	Pan Speed (Mouse)		Hover Pan/Zoom		
- Interface	Slow		Enable Hover I	Pan/Zoom	
Zoom / Pan		1.011	Activate only w		
- Annotations	Pan Speed (Keyboard)				
- Graphics - Takeoff Tools			Pan Speed:		
- Snapping	Slow	Fast	Slow		Faat
- Notifications					
- Digitizer Tablet	Zoom Speed (Mouse)		Zoom Speed		
- Property Groups - Advanced	Slow	Fast	Skar		Fast
Advanced			51014		Fast
	Zoom Speed (Keyboard)				
	Slow	Fast	Transparency.		

Figure 1

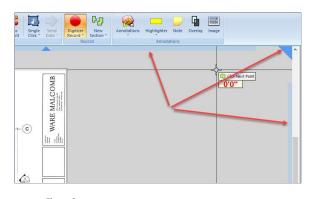


Figure 2

API Calls

Delphi

Using Iltem Object Model	> Expand source
Using PlanSwift Object Model	> Expand source

C#

Using Iltem Object Model	Expand source
Using PlanSwift Object Model	Expand source
VB/VBA (OLE)	
Using Iltem Object Model	> Expand source
Using PlanSwift Object Model	Expand source
Pascal Scripting (OLE)	
Item Object Model	Expand source
Root Object Model	> Expand source

Pascal Scripting

Item Object Model

> Expand source

Using the PlanSwift Object Model

> Expand source

PanZoomHoverDelay

PanZoomHoverDelay

Integer value that controls the **Delay** for the **Hover Pan/Zoom**. Value is in milliseconds (see figure 1). Figure 2 shows the light blue transparencies (and darker blue arrows) at the edges of the window and the darker triangular transparencies (arrows) in the corners. Hovering (not clicking) your mouse in any of those blue areas makes the plan scroll quickly in the direction of the arrow. Pressing the keyboard space bar reverses the scrolling direction.

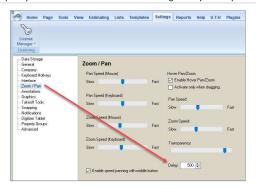


Figure 1

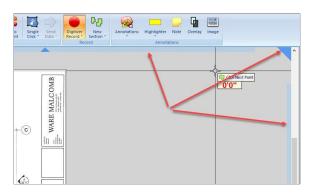


Figure 2

API Calls

Delphi	
Using Iltem Object Model	> Expand source
Using PlanSwift Object Model	Expand source

C#

Using IItem Object Model	Expand source
Using PlanSwift Object Model	Expand source

VB/VBA (OLE)

Using Iltem Object Model	> Expand source
Using PlanSwift Object Model	> Expand source

Pascal Scripting (OLE)

Item Object Model Pascal Scripting

Item Object Model	Expand source
Using the PlanSwift Object Model	> Expand source

PanZoomHoverTransparency

PanZoomHoverTransparency

Integer value that controls the **Hover Zoom/Pan Transparency**. Value ranges from 0 to 100 (see Figure 1). Figure 2 shows the light blue transparencies (and darker blue arrows) at the edges of the window and the darker triangular transparencies (arrows) in the corners. Hovering (not clicking) your mouse in any of those blue areas makes the plan scroll quickly in the direction of the arrow. Pressing the keyboard space bar reverses the scrolling direction.

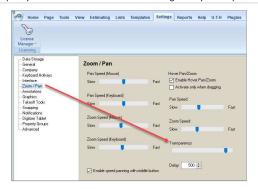


Figure 1

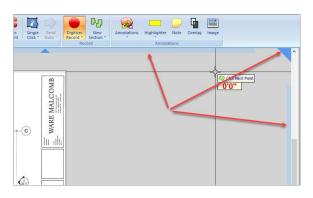


Figure 2

API Calls

Delphi			

Using Iltem Object Model	* Expand source
Using PlanSwift Object Model	Expand source
C#	
Using Iltem Object Model	> Expand source
Using PlanSwift Object Model	> Expand source
VB/VBA (OLE)	
Using Iltem Object Model	> Expand source
Using PlanSwift Object Model	> Expand source

Pascal Scripting (OLE)

Item Object Model Pascal Scripting

Item Object Model	Expand source
Using the PlanSwift Object Model	> Expand source

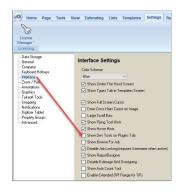
PLUGINDEVTOOLS

PLUGINDEVTOOLS

Boolean value that toggles whether **Plugin Developer Tools** are displayed on the **Plugin** ribbon bar (Figure 1). This variable is also set in the **Main / Settings / Interface** window (Figure 2). Checked is true and displays the **Plugin Developer Tools**; unchecked (the default) is false and does not show them.



Figure 1





API Cals

Using PlanSwift Object Model Expand sour Using PlanSwift Object Model Expand sour Using PlanSwift Object Model Expand sour B/VBA (OLE) Using PlanSwift Object Model Expand sour Using PlanSwift Object Model Expand sour ascal Scripting (OLE) Item Object Model Expand sour ascal Scripting (OLE) Item Object Model Expand sour ascal Scripting Expand sour Expand	Delphi	
# Using Iltem Object Model > Expand sour Using PlanSwift Object Model > Expand sour B/VBA (OLE)	Using IItem Object Model	> Expand source
Using Iltem Object Model > Expand sour Using PlanSwift Object Model > Expand sour B/VBA (OLE) > Using Iltem Object Model > Expand sour Using PlanSwift Object Model > Expand sour ascal Scripting (OLE) > Item Object Model > Expand sour ascal Scripting (OLE) > Item Object Model > Expand sour ascal Scripting > Expand Sour > Expand sour	Using PlanSwift Object Model	> Expand source
Using PlanSwift Object Model > Expand sour B/VBA (OLE) > Expand sour Using PlanSwift Object Model > Expand sour ascal Scripting (OLE) > Expand sour Item Object Model > Expand sour ascal Scripting (OLE) > Expand sour Item Object Model > Expand sour ascal Scripting > Expand sour ascal Scripting > Expand sour ascal Scripting > Expand sour	C#	
B/VBA (OLE) Using Iltem Object Model * Expand sour Using PlanSwift Object Model * Expand sour ascal Scripting (OLE) Item Object Model * Expand sour ascal Scripting Item Object Model * Expand sour	Using Iltem Object Model	Expand source
Using Iltem Object Model > Expand sour Using PlanSwift Object Model > Expand sour ascal Scripting (OLE) > Expand sour Item Object Model > Expand sour ascal Scripting > Expand sour Root Object Model > Expand sour ascal Scripting > Expand sour Item Object Model > Expand sour ascal Scripting > Expand sour	Using PlanSwift Object Model	> Expand source
Using PlanSwift Object Model > Expand sour ascal Scripting (OLE) Item Object Model Root Object Model > Expand sour ascal Scripting > Expand sour Item Object Model > Expand sour ascal Scripting > Expand sour ascal Scripting > Expand sour	VB/VBA (OLE)	
ascal Scripting (OLE) Item Object Model Root Object Model ascal Scripting Item Object Model Expand sour Ascal Scripting Item Object Model Expand sour	Using Iltem Object Model	Expand source
Item Object Model > Expand sour Root Object Model > Expand sour ascal Scripting > Item Object Model > Expand sour	Using PlanSwift Object Model	> Expand source
Root Object Model > Expand sour ascal Scripting > Expand sour Item Object Model > Expand sour	Pascal Scripting (OLE)	
ascal Scripting Item Object Model * Expand sour	Item Object Model	> Expand source
Item Object Model Capacity Second Sec	Root Object Model	Expand source
	Pascal Scripting	
> Expand source	Item Object Model	> Expand source
		> Expand source

Using the PlanSwift Object Model

PointSize

PointSize

Integer value that sets the **Point Size**. Values range from 0 to 20, with 10 being the default. Figure 1 shows the corresponding screen areas of the **Settings** screen and the **U-T-H** area.

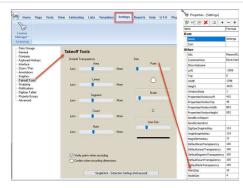


Figure 1

API Calls

Delphi	
Using IItem Object Model	Expand source
Using PlanSwift Object Model	> Expand source
C#	
Using Iltem Object Model	> Expand source
Using PlanSwift Object Model	> Expand source
VB/VBA (OLE)	
Using Iltem Object Model	> Expand source
Using PlanSwift Object Model	> Expand source
Pascal Scripting (OLE)	
Item Object Model	> Expand source
Root Object Model	> Expand source
Pascal Scripting	
Item Object Model	> Expand source
Using the PlanSwift Object Model	> Expand source

Property Groups

Property Groups

String value that controls the grouping order of properties in **Property Groups** (Figure 1 shows an example of properties indicated by red arrows). Figure 2 shows where to click to select the cell (left arrow). After clicking in the cell, click on the **Up** and **Down** arrows in the cell to move the selected property in priority. Clicking on the Elipsis to the right of the arrows opens the **U-T-H Formula Editor - Property Groups** window (Figure 3). The **U-T-H Property Groups** property is shown in Figure 2 (note that the **elipsis** disappears when the **Formula Editor** window opens). The **Property Group** items may be added, deleted, reordered, or edited in this window. After making any

changes, click on **OK** in the **Formula Editor** window, and then on **OK** in the **Properties** window, and the changes will be reflected in the PlanSwift **Settings** tab / **Property Groups** window (Figure 3) and in the PlanSwift **Property Groups** window shown in Figure 1.

췙 📲 🔗 🗶 💷 🕇 +	- * / (
Name	Formula
Item	1.979/1007/07162
Name	Office 102
Item #	
Estimating	
Qty	[Takeoff]
Cost Type	
Cost Total	[Qty] * [Cost Each]
Markup Each	[Cost Each] * ([Mai
Markup Total	[Qty] * [Markup Ea
Price Each	[Cost Each] + [Mar
Price Total	[Qty] * [Price Each
Fill	
Transparency	T 0
Texture Height	1
Takeoff Data	
Volume	[Area] * [Depth]
Wall Area	[Linear Total] * [Wi
Takeoff	[Area]
Area	[!Area]
Linear Total	[!Linear Total]
Segment Count	[!Segment Count]
Point Count	[!Point Count]
Default	[Takeoff]
Section Count	[!Section Count]
Work Breakdown Structu	15 contraction of the
Zone	
Folder Path	[!TakeoffPath]
Folder	[!TakeoffFolder]
Other	[runcom order]
Icon	
IsItem	-
DoubleClickAction	Record
	SubItem
DragDropAction	Record
IsArea	Record
	Accombly
Subitem Type	Assembly
TakeoffSummary_Expanded	
Audit Trail	1/22/2012 2:05 10
Time Stamp	1/23/2013 2:05:16
Videos	1918
SwiftTube VideoID	112
Inheritance path: _All_Item_1	「akeoff Item_Area∖
Input Advanced	Form

Figure 1

변 =은 😁 🗶 💷 🛛 +	- * / () () • 🖹 🖺	* * 5	Show: N	Iormal • 📰 Form
sme	Formula	Input Units	Adjust	Result
NewFromTemplateHotKey	84			84
NewAreaHotKey	49			49
NewLinearHotKey	50			50
NewSegmentHotKey	51			51
NewCountHotKey	52			52
RecordHotKey	82			82
NewSectionHotKey	78			78
ToggleImageEstimatingHotKey	123			123
NewNoteHotKey	187			187
ZoomIn2HotKey	187			187
ZoomOut2HotKey	189			189
Show Under The Hood Screen	True			True
Language	English			English
SummaryFormDockSide	Right			Right
NotesColumns	(Size: 307)			
NotesGridLayout	(Size: 608)			
Templates Sidebar Selected Ite	Parts and Assembles			Parts and Assemblies
Property Groups	Fil 🗧		••••••	Fil

Figure 2

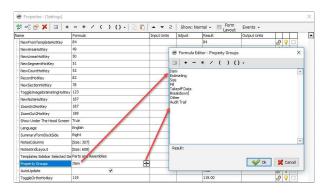


Figure 3

API Calls

Delphi	
Using Iltem Object Model	> Expand source
Using PlanSwift Object Model	Expand source
#	
Using IItem Object Model	> Expand source
Using PlanSwift Object Model	Expand source
B/VBA (OLE)	
Using IItem Object Model	> Expand source
Using PlanSwift Object Model	Expand source
ascal Scripting (OLE)	
the set Oh is set Mandal	> manual second

Item Object Model	Expand source
Root Object Model	> Expand source

Pascal Scripting	
Item Object Model	> Expand source
Using the PlanSwift Object Model	> Expand source

PropertiesWindowHeight

PropertiesWindowHeight

Read-only integer that displays the height of the Properties window.

API	Calls

D	elphi	
	Using IItem Object Model	> Expand source
	Using PlanSwift Object Model	> Expand source

C#

Using Iltem Object Model	Expand source
Using PlanSwift Object Model	> Expand source

VB/VBA (OLE)

Using IItem Object Model	> Expand source
Using PlanSwift Object Model	> Expand source

Pascal Scripting (OLE)

Item Object Model	Expand source
Root Object Model	> Expand source
Provide a state of the state of	
Pascal Scripting	

Item Object Model	> Expand source
Using the PlanSwift Object Model	> Expand source

PropertiesWindowLeft

PropertiesWindowLeft

Read-only integer value that displays the left position of the Properties window.

API Calls

Using Iltem Object Model

Using PlanSwift Object Model

C#

Delphi

Using IItem Object Model

Using PlanSwift Object Model

VB/VBA (OLE)

Using Iltem Object Model

Using PlanSwift Object Model

Pascal Scripting (OLE)

Item Object Model	> Expand source
Using the PlanSwift Object Model	> Expand source

 Item Object Model
 > Expand source

 Root Object Model
 > Expand source

Pascal Scripting Properties Window Top

PropertiesWindowTop

Read-only integer value that displays the top position of the Properties window.

API Calls

Delphi	
Using Iltem Object Model	> Expand source
Using PlanSwift Object Model	> Expand source

C#

Using Iltem Object Model	Expand source
Using PlanSwift Object Model	> Expand source

VB/VBA (OLE)

Item Object Model	> Expand source
Using the PlanSwift Object Model	> Expand source

Using Iltem Object Model

Using PlanSwift Object Model

Pascal Scripting (OLE)

Item Object Model

Root Object Model

Pascal Scripting PropertiesWindowWidth

PropertiesWindowWidth

Read-only integer value that displays the width of the Properties window.

API Calls

Delphi
Using Iltem Object Model

Using PlanSwift Object Model

C#

Using Iltem Object Model	Expand source
Using PlanSwift Object Model	> Expand source
B/VBA (OLE)	
Using IItem Object Model	> Expand source
Using PlanSwift Object Model	> Expand source
ascal Scripting (OLE)	
Item Object Model	> Expand source
Root Object Model	Expand source
	Expand source
Root Object Model ascal Scripting Item Object Model	 Expand source Expand source

RecordHotKey

RecordHotKey

Integer value returns an ANSI key code (default code 82, the letter **R**) for the **Stop/Start Point Recording** hotkey. Figure 1 shows where **Stop/Start Point Recording** is invoked on the **Main Ribbon Bar.**

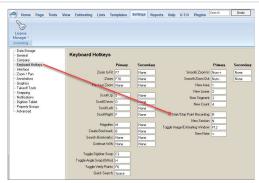


Figure 1

Help U-T-H	Plugin	s Searc	h	Undo]		-		
	$\hat{\mathbf{Q}}$	In.	R	:.	*	*			Ъŋ
Dimension	Area	Linear	Segment	Count	Auto Count	Single Click *	Send Data *	Digitizer Record *	New Section *
Measure				Takeoff				Rec	ord

Figure 2

API Calls

Item Object Model

Del	p	hi

Using IItem Object Model	Expand source
Using PlanSwift Object Model	> Expand source
C#	
Using Iltem Object Model	> Expand source
Using PlanSwift Object Model	> Expand source
/B/VBA (OLE)	
Using Iltem Object Model	> Expand source
Using PlanSwift Object Model	> Expand source
Pascal Scripting (OLE)	
Item Object Model	> Expand source
Root Object Model	> Expand source
Pascal Scripting	

\	
² Expand sour	ce

ReportValidUntil

ReportValidUntil

String value that sets the number of days report is valid for. This value is changed in the **Reports / Settings / Company Information** area, shown in Figure 1. Click on **Reports** (red arrow #1), and then click on **Settings** (red arrow #2): this opens the **Company Information** screen where the **Valid Until** value may be set. Figure 2 shows an example of where this value is implemented in a report.

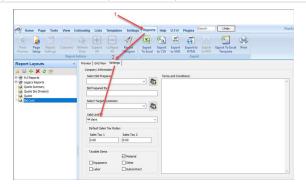


Figure 1

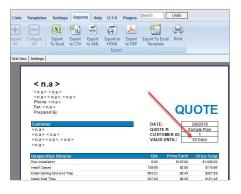


Figure 2

API Calls

Delphi

Using IItem Object Model	> Expand source
Using PlanSwift Object Model	> Expand source

C#

Using Iltem Object Model	Expand source
Using PlanSwift Object Model	> Expand source
/B/VBA (OLE)	
Using Iltem Object Model	Expand source
Using PlanSwift Object Model	> Expand source

Pascal Scripting (OLE) Pascal Scripting

Item Object Model

Item Object Model

> Expand source

Using the PlanSwift Object Model

> Expand source

SalesTax1

SalesTax1

affolding (priced per SQ FT) ental per month) Plank and Se Templates icaffolding (p ntal pe Parts and A Cost Type Qty Formule Qty Formule Qty Cost Total Markup Each Markup Tota Price Each Markup Tota Cost Each Markup Si Sales Tax 1 Sales Tax 2 Taxable Tax1 Tax2 Price Total B 🖨 SQ FT lp([Qty Formula] * ([Waste %]/100+1] SQ FT (coundup([Qty Form [Qty] * [Cost Each] [Cost Each] * ([Mark [Qty] * [Markup Each 0.00 tup %6] / 100) ch] + (M up Each] × 1.29 0.00 0.00 False 0.00 0.00 ob/Sales Tax 2] * [Taxable [Qty] "[Price Each]) * ([Sales Tax [Qty] "[Price Each]) * ([Sales Tax 2ty] * [Price Each]

Integer value that allows for a sales tax rate to be entered. These rates are utilized in the properties windows for **Parts and Assemblies Templates** (Figure 1 is an example).

Figure 1

API Calls

Delphi	
Using Iltem Object Model	> Expand source
Using PlanSwift Object Model	> Expand source

C#

Using IItem Object Model	> Expand source
Using PlanSwift Object Model	> Expand source

VB/VBA (OLE)

Using IItem Object Model	> Expand source
Using PlanSwift Object Model	> Expand source

Pascal Scripting (OLE)

Item Object Model	> Expand source
Root Object Model	Expand source

Pascal Scripting

Item Object Model	Expand source
Using the PlanSwift Object Model	> Expand source

SalesTax2

SalesTax2

Integer value that allows for a second sales tax rate to be entered. These rates are utilized in the properties windows for **Parts and Assemblies Templates** (Figure 1 is an example).

k 🕫 😤 🗶 🖄	+-*/()()+ 2 🖒	* * 5	Show: N	iormal • 📰 Form Lavout	Events *							
ame	Formula	Input Units	Adjust	Result	Output Units		Г		P	•		
em							_	_		Т		
Type	Equipment			Equipment		8	0			÷		
Name	Scaffolding (priced per SQ FT)			Scaffolding (priced per		8	9				 Templates 	
Description	(rental per month) Plank and Scaffold only			(rental per month) Plan		8	Q			E		
Item #						8	Ş			Т		
stimating						-				T	Parts and Assemblies	
Cost Type	Equipment			Equipment		8	9			Т	🕀 🧰 Parts	
Qty Formula	[Area]	SQ FT		0.00	SQ FT	8	Q			T	🖯 🗀 Equipment Parts (Drag 🖯 🧰 Rental Equip. (Are	-Drop
Qty	(RoundUp([Qty Formula] * ([Waste %]/100+1	SQ FT		0.00	SQ FT	8	0			Т	Scaffolding (pr	
Cost Total	[Qty] * [Cost Each]	\$		0.00	s	8	Q		_	+	🖲 🗀 Rental Equip. (Cou	
Markup Each	(Cost Each) * ([Markup %] / 100)			0.00		103	Q			Т	🗄 🧰 Rental Equip. (Line	
Markup Total	[Qty] * [Markup Each]	\$		0.00	\$	8	9			T	🕀 🧰 Rental Equip. Parts 🕀 🧰 Labor Parts (Drag-Droj	
Price Each	[Cost Each] + [Markup Each]	e		1.29	\$	8	Q			Т	Labor Parts (Drag-Dro Drag-Dro Material Parts (Drag-D	
Waste %	0	%		0	%	8	0			Т	🗄 🤤 Subcontract Parts (Dra	
Cost Each	1.29	\$		1.29	s	8	S			Т	🕀 🧰 Assembles	
Markup %	0	%		0.00	55	8		×		T		
Sales Tax 1	[Uob\Sales Tax 1] [Taxable]	%		0.00	96	8	Q			Т		
Sales Tax 2	[Uob/Sales Tax 2] * [Taxable]	%		0.00	96	8	Q			Т		
Taxable			1	False		8	Q			Т		
Taxi	([Qty]*[Price Each]) * ([Sales Tax 1] / 100)			0.00		8	Q			Т		
Tax2	([Qty]*[Price Each]) * ([Sales Tax 2] / 100)			0.00		8	9			Т		
Price Total	[Qty] * [Price Each]	\$		0.00	s	B				1		

Figure 1

API Calls

Do	Inhi
De	ihiii

Using IItem Object Model	Expand source
Using PlanSwift Object Model	> Expand source
C#	
Using Iltem Object Model	> Expand source
Using PlanSwift Object Model	> Expand source
VB/VBA (OLE)	
Using Iltem Object Model	Expand source
Using PlanSwift Object Model	> Expand source
Pascal Scripting (OLE)	
Item Object Model	Expand source
Root Object Model	> Expand source

Pascal Scripting	
Item Object Model	> Expand source
Using the PlanSwift Object Model	Expand source

ScrollDownHotkey

ScrollDownHotkey

Integer value returns an ANSI key code (default code 68, the letter D) for the Scroll Down command. Figure 1 shows where the Scroll Down hotkey is assigned.

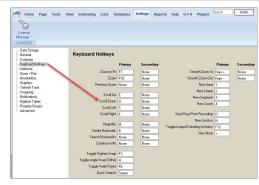


Figure 1

API Calls

Delphi

Using Iltem Object Model	> Expand source
Using PlanSwift Object Model	> Expand source

C#

Using IItem Object Model	Expand source
Using PlanSwift Object Model	> Expand source

VB/VBA (OLE)

Using Iltem Object Model	> Expand source
Using PlanSwift Object Model	> Expand source

Pascal Scripting (OLE)

Item Object Model	> Expand source
Root Object Model	> Expand source
Pascal Scripting	
Item Object Model	Expand source
Using the PlanSwift Object Model	• Expand source

ScrollLeftHotKey

ScrollLeftHotKey

Integer value returns an ANSI key code (default code 83, the letter S) for the Scroll Left command. Figure 1 shows where the Scroll Left hotkey is assigned.

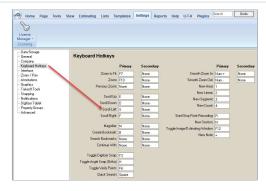


Figure 1

API Calls

Delphi	
Using Iltem Object Model	> Expand source
Using PlanSwift Object Model	Expand source
C#	
Using Iltem Object Model	Expand source

VB/VBA (OLE)

Using litem Object Model	> Expand source
Using PlanSwift Object Model	Expand source

Pascal Scripting (OLE)	
Item Object Model	> Expand source
Root Object Model	> Expand source
Pascal Scripting	
Item Object Model	Expand source

	Using the PlanSwift Object Model	> Expand source
-		

ScrollRightHotKey

ScrollRightHotKey

Integer value returns an ANSI key code (default code 70, the letter F) for the Scroll Right command. Figure 1 shows where the Scroll Right hotkey is assigned.

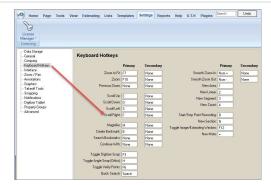


Figure 1

API Calls

Delphi

Using IItem Object Model	Expand source
Using PlanSwift Object Model	> Expand source
C#	

Using Iltem Object Model	Expand source
Using PlanSwift Object Model	Expand source

VB/VBA (OLE)

Using Iltem Object Model	> Expand source
Using PlanSwift Object Model	> Expand source
Pascal Scripting (OLE)	
Item Object Model	> Expand source
Root Object Model	> Expand source
Pascal Scripting	
Item Object Model	> Expand source
Using the PlanSwift Object Model	> Expand source

ScrollUpHotKey

ScrollUpHotKey

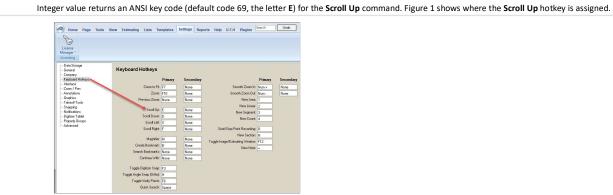


Figure 1

API Call

Delphi	
Using litem Object Model	> Expand source
Using PlanSwift Object Model	> Expand source

C#

Using Iltem Object Model	> Expand source
Using PlanSwift Object Model	Expand source
/B/VBA (OLE)	
Using Iltem Object Model	> Expand source
Using PlanSwift Object Model	Expand source
ascal Scripting (OLE)	
Item Object Model	> Expand source
Root Object Model	Expand source
Pascal Scripting	
Item Object Model	Expand source
Using the PlanSwift Object Model	> Expand source
SendErrorReport	

SendErrorReport

Boolean value controlling whether error reports are sent to PlanSwift. Checked is true and sends error reports. Unchecked is false and does not send them.

API Calls	
Delphi	
Using IItem Object Model	Expand source
Using PlanSwift Object Model	> Expand source
C#	
Using IItem Object Model	Expand source
Using PlanSwift Object Model	> Expand source

VB/VBA (OLE)

Using Iltem Object Model	Expand source
Using PlanSwift Object Model	Expand source
ascal Scripting (OLE)	
Item Object Model	> Expand source
Root Object Model	> Expand source
ascal Scripting	
Item Object Model	> Expand source
Using the PlanSwift Object Model	> Expand source

SendScreenShot

SendScreenShot

Boolean value controlling whether screenshots of errors are sent to PlanSwift. Checked is true and sends screenshots. Unchecked is false and does not send them.

API Calls

Delphi	
Using IItem Object Model	> Expand source
Using PlanSwift Object Model	> Expand source
C#	
Using IItem Object Model	Expand source
Using PlanSwift Object Model	> Expand source
VB/VBA (OLE)	
Using IItem Object Model	> Expand source
Using PlanSwift Object Model	> Expand source

Pascal Scripting (OLE)

Item Object Model	> Expand source
Root Object Model	Expand source
ascal Scripting	
	> Expand source
Item Object Model	Expand source

Show Overview

Show Overview

Boolean value that controls the **Show Overview** function. Checked is true and enables it; unchecked is false and disables it. Show overview is activated in PlanSwift software by clicking on the small arrow, which opens the overview box (shaded blue); see Figure 1.



Figure 1

API Calls

Delphi

Using Iltem Object Model	Expand source
Using PlanSwift Object Model	> Expand source
C#	
Using Iltem Object Model	> Expand source

> Expand source

Using Blan Swift Object Medal	
Using PlanSwift Object Model	

VB/VBA (OLE)

Using Iltem Object Model	Expand source
Using PlanSwift Object Model	Expand source
ascal Scripting (OLE)	
Item Object Model	Expand source
Root Object Model	Expand source
ascal Scripting	
Item Object Model	> Expand source
Using the PlanSwift Object Model	Expand source

Show Under The Hood Screen

Show Under The Hood Screen

Boolean value controlling whether the **U-T-H** (Under-The-Hood) **Main Menu** tab is visible. Figure 1 shows that it is not visible. To make it visible, click on **Settings**, then on **Interface**, and then on Show Under The Hood Screen. Figure 2 shows where the password is entered. Figure 3 shows the U-T-H tab after the password is successfully entered. Figure 3 also shows how clicking the **U-T-H** tab, then the **Settings** advanced properties, displays the **Show Under-The-Hood Screen**. When enabled, the value in the **Formula** field shows as True displaying the tab. If False is entered and the screen closed by clicking on OK, then the **UT-C** tab is no longer visible. To obtain the **U-T-H** password, contact your PlanSwift representative or send an email to takeoff@constructconnect.com.

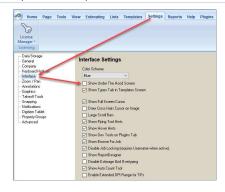


Figure 1

Using IItem Object Model	> Expand source
Using PlanSwift Object Model	Expand source

Under The	Hood				×
Enter Passi	word Fo	r Under	The Hoo	d	
			🏈 Ok		Cancel

Figure 2

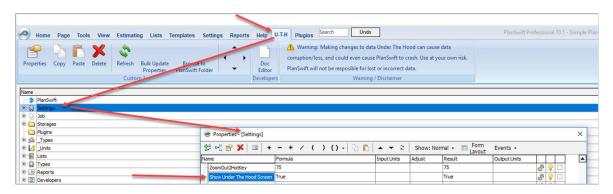


Figure 3

API Calls

Delphi

Using IItem Object Model	Expand source
Using PlanSwift Object Model	> Expand source
VB/VBA (OLE)	
Using IItem Object Model	Expand source
Using PlanSwift Object Model	Expand source
Pascal Scripting (OLE)	
Item Object Model	Expand source
Root Object Model	> Expand source
Pascal Scripting	
Item Object Model	Expand source
Using the PlanSwift Object Model	> Expand source

ShowWelcome

ShowWelcome

Boolean value determining whether the Welcome to PlanSwift screen (Figure 1) is displayed. The Welcome screen requires the user to enter a Customer # and a PIN # in order to log in to the PlanSwift Professional mode (Figure 2) and not be in Viewer-only mode (Figure 3).

When checked (true), the Welcome screen will be displayed when the software starts up, whether the software has been activated or not.

If it is not checked and the user has activated PlanSwift, then the Welcome screen will not appear at startup.

If it is not checked and PlanSwift has not been activated, then PlanSwift will load in the **Viewer** mode (Figure 3). Any attempts to command the software will cause the **Activate PlanSwift Professional** window (Figure 4) to appear, requiring the entry of the **Customer #** and **Pin #** before PlanSwift can be put into its **Professional** mode.

5	Velcome to PlanSwift		3
	Welcome to	PlanSwift	
		vift Professional, enter the in # that was emailed to you, or eb site:	
	Customer #:	c#######	
	Pin #:	****	
		Activate	
		ouble unlocking, feel free to call us: 888-PLANSWIFT	
	To continue as	the view only version of PlanSwift, Click Here	

Figure 1

Templates	Settings	Reports	Help	Plugins	Search Undo	PlanSwift Professional 10.1 - Sample Plan
	88 temote sistance	<u>f</u> About	Report a Bug	Feature Request	Your Customer ≠ : dave.hansen@planswift.com	
ensing		About	Beta	Tools	Support	

Figure 2

Templa	tes Settings	Reports	Help	Plugins	Search Undo	PlanSwift Professiona Viewer 0.1 - Sample Plan
Check for Updates	Remote Assistance	<u>ඩ</u> About	Report a Bug		Your Customer # : dave.hansen@planswift.com	
ensing		About	Bet	a Tools	Support	

Figure 3

	are is not currently a	ctivated.	
			ble in the Viewer Only in codes, please ente
Customer #			Activate Now
Pin #	t:		y
	Forgot Codes?		
			to download a free , please select one o
	Free 14 Day Tria		urchase

Figure 4

API Calls

Delphi

Using PlanSwift Object Model

Expand source

C#

Using	lltem	Object	Model
-------	-------	--------	-------

> Expand source

Using Iltem Object Model	Expand source
Using PlanSwift Object Model	> Expand source
VB/VBA (OLE)	
Using Iltem Object Model	> Expand source
Using PlanSwift Object Model	> Expand source
Pascal Scripting (OLE)	
Item Object Model	> Expand source
Root Object Model	> Expand source
Pascal Scripting	
Item Object Model	> Expand source
Using the PlanSwift Object Model	> Expand source

SmartOrtho

SmartOrtho

Boolean value that toggles **Smart Ortho** on or off. When enabled (True), **Smart Ortho** operates automatically when close to angles. When disabled (False) it does not operate. Figure 1 shows where **Smart Ortho** is controlled in the **Main Menu Settings** / **Snapping** window. **Smart Ortho** cannot be enabled if **Ortho** is disabled.

🧐 Home Page To	View Estimating Lists Templates Settings Reports
License Manager * Licensing	
Data stologie General – Company – Ksyboard Hotkeys – Interface – Zoom / Pan – Annotations – Graphics – Graphics – Takeoff Tools – Snapping – Notifications – Digitizer Tablet – Propety Groups – Advanced	Snapping Chable Snapping to points Snap to PlanSwitt points Snap to CAD points Hover to Snap Axis Ortho - Snap to angles (hold Shitt for temporary enable/disable Smart Ortho - Snap only when close to angles

Figure 1

API Call

Delphi

Delphi	
Using IItem Object Model	> Expand source
Using PlanSwift Object Model	> Expand source

C#

> Expand source
> Expand source
> Expand source
> Expand source
Expand source
> Expand source
> Expand source
> Expand source

SuppressAutoScaleDisclaimer

SuppressAutoScaleDisclaimer

Boolean value that suppresses or allows the "Do not show this again" Auto Scale Disclaimer. This is controlled in the Settings/Notification screen (Figure 1) and can also be turned off on the Auto Scale Disclaimer window (Figure 2). A check in the Settings/Notification screen enables the Auto Scale Disclaimer; a check in the Auto Scale Disclaimer window disables the disclaimer.

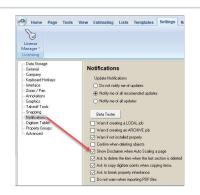


Figure 1

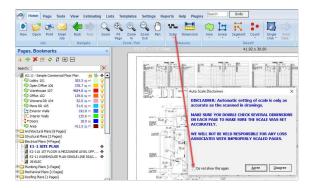


Figure 2

API Calls

Delphi

Using litem Object Model	Expand source
Using PlanSwift Object Model	> Expand source
C#	
Using Iltem Object Model	Expand source
Using PlanSwift Object Model	> Expand source
/B/VBA (OLE)	

Using litem Object Model	> Expand source
	> Expand source

> Expand source

> Expand source

Pascal Scripting (OLE)

Root Object Model

Pascal Scripting

Item Object Model

Item Object Model

Using the PlanSwift Object Model

> Expand source

> Expand source

ToggleImageEstimatingHotKey

ToggleImageEstimatingHotKey

Integer value returns an ANSI key code (default code 123, the character F12) for the Toggle Image Estimating hotkey (which selects the Estimating tab). Figure 1 shows the Main Menu / Settings / Keyboard Hotkeys window, which allows for the hotkey to be selected. Figure 2 shows the Estimating tab.

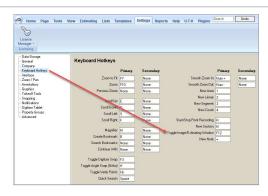


Figure 1

A Home Page Tools View	stimating Lists Templates Settings Reports	Help U-T-H Plugins	Search	Undo			Pla	nSwift Professiona
New New New New New Folder * Item * Area * Linear *	New New New New New From Segment Count Assembly Part Type -	Delete Print Colu	umns Filter R	ر Lefresh	Properties Expans		Export Import To Excel Template	Copy Paste
		Estimate						Clipboard
Estimating Layouts	Default							
3 🖬 🕂 🗙 🌣 😁	Name	Description	Qty	Units	Cost Each	Markup % Pric	e Each Price To	tal Color
Default	😑 🚞 Sample Plan Takeoffs				\$0.00			_
PS Layouts	🕀 🏟 Lobby 101		305.47	SQ FT	\$0.00	0	0	\$0.00
e estavous	🕀 🏟 Open Office 106		378.74	SQ FT	\$0.00	0	0	\$0.00
	🕀 😳 Warehouse 107		1948.75	SQ FT	\$0.00	0	0	\$0.00
	Office 102		139.6	SQ FT	\$0.00	0	0	\$0.00
	🕀 🏟 Womens RR 104		52	SQ FT	\$0.00	0	0	\$0.00
	Exterior Walls		192.77	FT	\$0.00	0	0	\$0.00
	Mens RR 105		51.58	SQ FT	\$0.00	0	0	\$0.00
	Doors		8	EA	\$0.00	0	0	\$0.00
	Interior Walls		135.55	FT	\$0.00	0	0	\$0.00
	Residential Plan Takeoffs							

Figure 2

API Calls

Delphi Using Iltem Object Model Expand source Using PlanSwift Object Model Expand source

C#

Using IItem Object Model	> Expand source
Using PlanSwift Object Model	> Expand source

VB/VBA (OLE)

Using litem Object Model	> Expand source
Using PlanSwift Object Model	> Expand source

Pascal Scripting (OLE)

Item Object Model	> Expand source
Root Object Model	> Expand source
	Expand Source

Pascal Scripting	
Item Object Model	> Expand source
Using the PlanSwift Object Model	> Expand source

ToggleOrthoHotKey

ToggleOrthoHotKey

Integer value returns an ANSI key code (default code 115, the key F4 in hotkey window) for the **Toggle Angle Snap (Ortho)**, shown in Figure 1). This is controlled in the Main Menu / Settings / Snapping window (**Ortho Snap to Angles**, Figure 2) and at the bottom of the PlanSwift window (Figure 3).



Figure 1



Figure 2

Practice Residential Plan - Use for demo	⊕ ▼	
1 + 🗙 🕾 I		a sing
	-	select Stop
Search:	_	www.
	9 🗄	yboard shortcuts
Pages, Bookmarks		
🔆 Takeoff Summary		
🚽 Notes		rea and Linear tool
	•	<
		1820.5, 2811.0 Snap Ortho FreeHand Verify Points

Figure 3

API Calls

Delphi	
Using Iltem Object Model	> Expand source
Using PlanSwift Object Model	> Expand source
#	
Using Iltem Object Model	> Expand source

VB/VBA (OLE)

Using IItem Object Model	Expand source
Using PlanSwift Object Model	Expand source
Pascal Scripting (OLE)	
Item Object Model	> Expand source
Root Object Model	> Expand source
Pascal Scripting	
Item Object Model	Expand source
Using the PlanSwift Object Model	> Expand source

Тор

Тор

Read-only integer value that displays the top position of the Main Window.

API Calls

Delphi

Using Iltem Object Model	
Using PlanSwift Object Model	

C#

Using IItem Object Model	
Using PlanSwift Object Model	

VB/VBA (OLE)

Using Iltem Object Model

Using PlanSwift Object Model

Pascal Scripting (OLE)

Item Object Model	
Root Object Model	

Pascal Scripting

Item Object Model

Using the PlanSwift Object Model

Verify

Verify

Boolean value that toggles Verify Points On and Off. Figure 1 shows where Verify is controlled in the PlanSwift Main Menu / Takeoff Tools screen. Figures 2 and 3 show Verify Points in its highlighted and un-highlighted state in the PlanSwift window. When highlighted, Verify Points is True (on). When un-highlighted, Verify Points is False (off). When it is on, the Verify Entry popup window (Figure 4) is displayed after each takeoff is entered.

Data Storage					
General	Takeoff To	ools			
Company Kepboard Hofkeys	Default Tra	MONPOCE		Size	
Interface		Area			Point
Zoon / Pan	Less		More		
Annotations		CONTRACTOR OF CONTRACTOR		100	
Graphics Tekeoti Tools		Linear			0
Lakwolt Loolz Snepping	Less	Come I	Mran		0
Notifications	Less		PECHO		
Digitizer Tablet					Node
Property Groupe		Segment			
Advanced	Lenn	•	More	1.000	
1					
1		Count			
1	Less		More		
	1.1				Line Size
	\	Note			
	Less		More		Contraction and the state
	100				
	\				

Figure 1

		Bookmark	Tap B once on the keyboard to create a	Bookmark of th	e current view of t	he page
Pages, Bookmarks		New Section	Double Click to create a New Section who	en digitizing wi	th the Area and Lin	ear tools
G: Takeoff Summary						
Attachments						e Measure
Notes	7					
	- <				1	
		-152.1, 3113.7	Snap	Ortho Freeh	and Verify Points	

Figure 2

And a second		to Stop the	digitizer or Right	Click an	id selec	t Stop	
🔂 Takeoff Summary		oard to create an Arc or a Curved Line when digitizing					
					ect the Keyboard Hotkeys to see the keyboard shortcuts		
		/		1 to creat	te a Bookmark of t	the curr	rent vie
		<					
	1013.4. 2282.5			Snan	Ortho	FreeHand	Verify Point

Figure 3

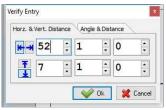


Figure 4

API Calls

Using IItem Object Model	Expand source
Using PlanSwift Object Model	> Expand source

C#

Using Iltem Object Model	Expand source
Using PlanSwift Object Model	> Expand source
VB/VBA (OLE)	
Using Iltem Object Model	Expand source
Using PlanSwift Object Model	> Expand source

Expand source

Item Object Model	> Expand source
Using the PlanSwift Object Model	> Expand source

Width

Width

Read-only integer value that displays the width of the Main Window.

API Calls	
Delphi	
Using Iltem Object Model	> Expand source
Using PlanSwift Object Model	> Expand source
C#	
Using Iltem Object Model	> Expand source
Using PlanSwift Object Model	> Expand source
/B/VBA (OLE)	
Using Iltem Object Model	Expand source
Using PlanSwift Object Model	> Expand source
Pascal Scripting (OLE)	
Item Object Model	Expand source
Root Object Model	> Expand source
Pascal Scripting	
Item Object Model	Expand source
Using the PlanSwift Object Model	> Expand source
WindowState	

WindowState

Read-only integer value that shows whether PlanSwift program window is minimized or maximized. A value of 0 is minimized and 1 is maximized.

API Calls

Delphi	
Using Iltem Object Model	> Expand source
Using PlanSwift Object Model	> Expand source
C#	

Using IItem Object Model	> Expand source
Using PlanSwift Object Model	> Expand source

VB/VBA (OLE)

Using Iltem Object Model	> Expand source
Using PlanSwift Object Model	> Expand source
ascal Scripting (OLE)	
Item Object Model	> Expand source
Root Object Model	> Expand source
Pascal Scripting	
Item Object Model	Expand source
Using the PlanSwift Object Model	> Expand source

ZoomHotKey

ZoomHotKey

Integer value that returns an ANSI key code (default code 121, Function key F10). Figure 1 shows where the **Zoom** hotkey assignment is made. Figure 2 shows where the **Zoom** command is invoked.

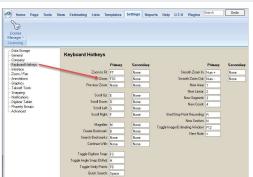


Figure 1



Figure 2

API Calls

Delphi

Using Iltem Object Model	> Expand source
Using PlanSwift Object Model	> Expand source

C#

Using Iltem Object Model	> Expand source
Using PlanSwift Object Model	> Expand source

VB/VBA (OLE)

Item	Object	Model

Using Iltem Object Model	Expand source
Using PlanSwift Object Model	> Expand source

Pascal Scripting (OLE)

Item Object Model	Expand source
Root Object Model	> Expand source

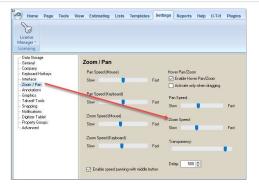
Pascal Scripting

Using the PlanSwift Object Model	Expand source
----------------------------------	---------------

ZoomHoverSpeed

ZoomHoverSpeed

Integer value that controls the Hover Zoom Speed (Figure 1). Value ranges from 5 to 100. Figure 2 shows the light blue transparencies (and darker blue arrows) at the edges of the window and the darker triangular transparencies (arrows) in the corners. Hovering (not clicking) your mouse in any of those blue areas makes the plan scroll quickly in the direction of the arrow. Pressing the keyboard space bar reverses the scrolling direction.





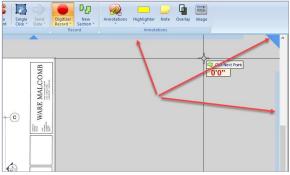


Figure 2

API Calls

Delphi	
Using Iltem Object Model	> Expand source
Using PlanSwift Object Model	> Expand source
C#	

Using Iltem Object Model	> Expand source
Using PlanSwift Object Model	> Expand source

VB/VBA (OLE)

Using IItem Object Model	> Expand source
Using PlanSwift Object Model	> Expand source

Pascal Scripting (OLE)

Item Object Model	> Expand source
Root Object Model	> Expand source
Pascal Scripting	
Item Object Model	> Expand source
Using the PlanSwift Object Model	> Expand source

ZoomInHotKey

ZoomInHotKey

Integer value returns an ANSI key code (default code 107, the number pad "+" sign). Figure 1 shows where the **Smooth Zoom In** hotkey is assigned. Figure 2 shows where the **Zoom In** command is located along the PlanSwift **Main Menu bar**.

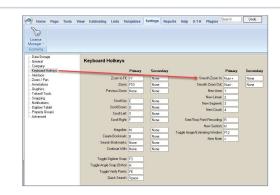


Figure 1



Figure 2

API Call

Delphi

Using IItem Object Model	> Expand source
Using PlanSwift Object Model	Expand source
2#	
Using Iltem Object Model	> Expand source
Using PlanSwift Object Model	> Expand source
/B/VBA (OLE)	
Using Iltem Object Model	> Expand source
Using PlanSwift Object Model	Expand source
Pascal Scripting (OLE)	
Item Object Model	> Expand source
Root Object Model	> Expand source
Pascal Scripting	
Using the PlanSwift Object Model	> Expand source

Item Object Model

ZoomOutHotKey

ZoomOutHotKey

Integer value returns an ANSI key code (default code 109, the number pad "-" sign). Figure 1 shows where the **Smooth Zoom Out** hotkey is assigned. Figure 2 shows where the **Zoom Out** command is located along the PlanSwift **Main Menu bar**.

Manager * Licensing						
– Data Storage General – Company	Keyboard Hotkeys					
Keyboard Hotkeys	and the second sec	Primary	Secondary		Primary	Seconda
- Interface - Zoom / Pan	Zoom to Pac	11	None	Smooth Zoom In:	Num+	None
Annotations	Zoom	F10	None	Smooth Zoom Dut	Num -	None
- Graphics - Takenti Tonk	Previous Zoom	None	None	New Asea	1	
- Snepping				New Linear	2	
Notifications	Scroll Up:		None	New Segment	3	
Digitizer Tablet Property Groups	Sciol Down		None	New Count	4	
- Advanced	Scroll Left:		None			
	S croll Right:	F	None	Start/Stop Point Recording		
	Magnifier:	м	None	New Section		
	Create Bookmark:	B	None	Toggle Irrage/Estimating Window		
	Search Bookmarks:	None	None	New Note	-	
	Continue With:	None	None			
	Tocale Diatizer Snap	F1				
	Togole Angle Snap (Ortho):					
	Topple Verily Points:					
	Quick Search:	Cana				

Figure 1



Figure 2

API Call

Del	: ما مرا
De	ipni

Using Iltem Object Model	Expand source
Using PlanSwift Object Model	Expand source
Using Iltem Object Model	> Expand source
Using PlanSwift Object Model	> Expand source
/VBA (OLE)	
Using Iltem Object Model	Expand source
Using PlanSwift Object Model	> Expand source
scal Scripting (OLE)	
Item Object Model	Expand source
Root Object Model	Expand source
iscal Scripting	
Using the PlanSwift Object Model	Expand source

ZoomSpeed

ZoomSpeed

Integer value that controls the mouse Zoom Speed. Value ranges from 0 to 10. The Main Menu / Settings / Zoom/Pan window (Figure 1) also controls Zoom Speed. Zoom is controlled by the mouse's wheel.



Figure 1

API Calls

Delphi

Using Iltem Object Model	> Expand source
Using PlanSwift Object Model	> Expand source
C#	
Using Iltem Object Model	> Expand source
Using PlanSwift Object Model	> Expand source
VB/VBA (OLE)	
Using Iltem Object Model	> Expand source
Using PlanSwift Object Model	> Expand source
Pascal Scripting (OLE)	
Item Object Model	> Expand source
Root Object Model	> Expand source
Pascal Scripting	
Item Object Model	> Expand source
Using the PlanSwift Object Model	> Expand source

ZoomSpeedKeyboard

ZoomSpeedKeyboard

Integer value that controls the keyboard **Zoom Speed**. Value ranges from 5 to 100. Figure 1 shows where is value is controlled in the **Main Menu / Zoom/Pan** window. Controlled by +/- on keyboard's ten-key keypad.

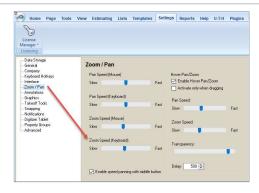


Figure 1

API Calls

Delphi

Using IItem Object Model	> Expand source
Using PlanSwift Object Model	> Expand source
C#	
Using IItem Object Model	Expand source
Using PlanSwift Object Model	> Expand source
VB/VBA (OLE)	
Using IItem Object Model	Expand source
Using PlanSwift Object Model	> Expand source
Pascal Scripting (OLE)	
Item Object Model	Expand source
Root Object Model	> Expand source
Pascal Scripting	
Item Object Model	> Expand source
Using the PlanSwift Object Model	> Expand source

ZoomToFitHotKey

ZoomToFitHotKey

Integer value that returns an ANSI key code (default code 118 or function key F7). Figure 1 shows where the **Zoom To Fit** hotkey assignment is made. Figure 2 shows where **Zoom To Fit** is invoked on the **Main Menu Ribbon** bar.

Keyboard Hotkeys					
Keyboard Hotkeys					
	Primary	Secondary		Primary	Secondar
Zoom to Fit:	F7	None	Smooth Zoom In:	Num +	None
Zoom	F10	None	Smooth Zoom Dut:	Num -	None
Previous Zoom	None	None	New Area:	1	
			New Linear:	2	
			New Segment:	3	
			New Count:	4	
			0.00	-	
Scroll Highl:	F	None			
Magniller:	м	None			
Create Bookmark:	B	None			
Search Bookmarka:	None	None	New Note:	-	
Continue Wilh:	None	None			
Tamila Distant Same					
	Previous Zoom Scroll Up: Scroll Law Scroll Law Scroll Right Center Bookmark: Search Bookmark: Cartinee Welt Toggle Angle Snap (Dirtho) Toggle Vage (Darks)	2001 10 10 10 10 10 10 10 10 10 10 10 10	Person 2001 Tore None Soci Up E None Soci Up E None Soci Up E None Soci Up F None Soci Up E None Soci Up E None Contra WM None None Togle Opter Soci IP Togle Opter None I Togle Opter None I I	Person 2001 Non Base Non / An Soci Up Taran Non / An Non / An Soci Up Taran Non / An Non / An Soci Up Taran Non / An Non / An Soci Up Taran Non / An Non / An Soci Up Taran Non / An Non / An Contra Wei Taran Taran Non Non / An Tageh Opter Non / Taran Non Non / An Tageh Opter Non / Taran Taran Non Non	Person Zoon Noon I non Networks I Social by E I name Networks I Social by E I name Networks I Social by E I name Networks I Social by F I name Networks I Name Network I Control bords I Dear Boltowick Noon I Control bords I Dear Boltowick Noon I Taggie Nogles Tool I Taggie Nogles Tool I Taggie Nogles Tool I Taggie Nogles Tool I

Figure 1



Figure 2

API Calls

Delphi

Using IItem Object Model	> Expand source
Using PlanSwift Object Model	Expand source
C#	
Using IItem Object Model	Expand source
Using PlanSwift Object Model	> Expand source
VB/VBA (OLE)	
Using Iltem Object Model	> Expand source
Using PlanSwift Object Model	> Expand source
Pascal Scripting (OLE)	
Item Object Model	Expand source
Root Object Model	> Expand source
Pascal Scripting	
Using the PlanSwift Object Model	Expand source

Item Structure Overview

Item Structure Overview

This section describes the internal structure of the API and how to enable the Under-The-Hood tab.

Before working with the API, a good understanding of the internal structure is recommended. To review the structure, the under-the-hood (U-T-H) tab needs to be enabled.

CAUTION

By modifying or changing anything in the back end, you may adversely affect the operation of the application. Modifications should be done in a read-only mode. If any modifications are done to the back end, those modifications will be lost when the application is re-installed.

Follow these steps to enable the Under-The-Hood tab.

- 1. Open PlanSwift.
- 2. Click on Settings along the top ribbon bar (see #1 on Figure 1 below).
- 3. Select *Interface* from the options on the left (see #2 on Figure 1 below).
- 4. Click on Show Under the Hood Screen(see #3 on Figure 1 below).

Data Storage General	Sectors and a sector of			
Company Republic Holings Heritica Can / Pan Can / Pan Ca	Interface Settings Cata States Back States	3	4	Mader The House The Parameter Vacion The House The Parameter Vacion The House With Company of the House With Company With

Figure 1

- 5. For the password, please contact your PlanSwift representative. Enter the password (see number #4 of Figure 1) and click on Ok.
- 6. An U-T-H (for "Under the Hood") now appears on the top ribbon bar (see number #1 of Figure 2). Click on U-T-H to display the screen resembling Figure 2.

December Com Bute Debte Referit Buck Hoders Provesto	ing changes to data Under The Hood can cause data d could even cause PlanSwift to crash. Use at your own risk. e resposible for lost or incorrect data. Warning / Disclaimer.
Name	Loaded Child List Lo
2 Harovit	True True
8 g Settings	True True
- 0 Job	True True
+ 🔁 Storages	Palse True
D Plugins	True True
8 🙉 _Types	True True
Units	False True
9 🗐 Lina	False True
C Types	True True
E in Reports	False False
8 2 Developers	Palse Palse
8 😋 Estimating	False False
8 🦳 Hatches	False False
8 🤤 Textures	False False
Same Toelkert ann Dogener State 4+440+441(21)2/SAS(2) saket Train Saket Train State Sta	Ohrgescheti Fele Neusgescheti Fale
Address: MAR 2700 Order Index: - 99999 Figure 1.1-2	



7. This window shows the complete back end of the PlanSwift structure, which shows everything from your jobs, your storages, your plug-ins, and your types. These are the most commonly utilized items when you're developing against anything within PlanSwift. The structure always starts at PlanSwift, which is the parent, or the root object, for everything that you want to access (see 1 of Figure 3). Each of the folders beneath PlanSwift is a child of PlanSwift. Click on the + to the left of Job folder (number #2 of Figure 3) to open the Job folder, which will display the Pages, Takeoff, and Bookmarks (see number #3 of Figure 3). This Job folder contains the current active Job that is loaded into PlanSwift. When programming, in order to access job information in the Job folder, use the relative path of \Job to access the Job folder. Under the \Job folder, you will see Pages, Takeoff, and Bookmarks. If no job is loaded, then the job's value will be Null. Pages items reside in the \Job\Pages; Bookmarks items reside in the U-T-H \Job\Pages folder. The PlanSwift Pages information resides in the U-T-H \Job\Pagef folder.



8. Figure 4 shows an example of the Estimating screen.

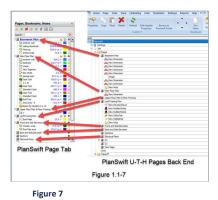
		Estimate						Cipto	ed, Afort			Templates, Inputs
Estimating Layouts	Default	97	Description	LININ	CetEah 1	Section 75	Dirfab f	Total	Duration	#-h-	and result.	
I 🖬 + ¥ < 🕾	C C lasenert	41	Constraint.	010	CHAILEREN		THE CAU		UNVIRON	-		8 \$ H O B B
topial private a	8 [1] exterior wall	\$29.68		FT	60.00	0		#2.02	Challing 1			Phy Tab
Certault	(2x+bd) litered Stud (priced EA)		161.00	10	\$1.00	0			d Rived and Planks		12	
er PS Layouts	2x4x6 Wood Stud (priced EA)	\$12	16° OC	14	\$4.00	0		\$1,248,00	6 Wood and Plantics		12	- Centerior walls
	2x4 Vised Plate Spriced EX)			EA	\$5.00	0			4 Wood and Plantics		10	Concrete T Concrete
	(\$10° Drynal (primit #A)	117	f a 10 Seets	FA	\$11.25	20	13.5	\$1,579,52	1 Friday		8	
	R-30 (maulation (priced per Rolf)		100 SQ FT per Rol		\$24.00	20			7 Thermal and Nost		8	
	Labor per FT (priced per Hour)	52,97		Pava	\$155.00	15		120.051.05			12	
	C cellos devizad	2245.62		50.FT	80.00	0		80.00				
	C Party	2245.61		SQ FT	\$2.00	0	0	\$2.00			0.00	
	B C Ner Floor										0.00	
	C bedroon	232.7		50.FT	\$0.00	0		\$0.00			0.00	
	In Linear	13.72		FT	\$0.00	0	0	\$2.02			0.00	
	They Septent	10.00		FT	\$5.00	0	0	\$0.00			0.00	
	ber stock	2		EA	\$0.00	0	9	\$0.00				
	ograge ped	94.34		50 YD	\$0.00	0	0	\$0.00		-	0	
	8- mail Asist Taxi	708.83		5Q.FT	80.00	0	0	80.02			0.00	
	8 🗰 Jost Teal	907.74		50.97	\$0.00	0	0	\$0.00			0.00	
	8 🗰 Joset Taxi	291.95		50.17	\$0.00	0	0	\$0.00			0.00	
	30 m Grid Tool	893.01		50.77	\$0.00	0	0	90.00			0.00	
	3 in Nean for headers 2 x 12	156		FT	80.00	0	0	\$0.00		-	0.00	
	- Reof mps	25.5		PT	\$2.00		0	\$0.00			0.00	
	40 Area	0		50.97	\$0.00	0	0	90.00		-	0.00	
	Window area				\$2.00	0	2	\$0.00			6.00	
	Co Roof hp area	0		SQ FT	80.00	0	0	\$0.00			0.00	
	Roof no area	734.65		50.FT	\$5.00	0	0	\$0.00			0.00	
	fasic Count	0		EA	\$0.00	0		\$0.00				
	12 1	1853.53		FT	\$0.00		0	\$0.00			0.00	
	A deetack celling	0		5Q.FT	\$5.00	0	. 0	80.00				
	- DiterorWal	77.26		97	\$0.00	0	0	\$0.00			0.00	
	C Interior Vials				\$0.00							
	Second foor									-		

Figure 4

 Note the similarities in the red rectangular areas in Figure 5 (U-T-H screen) and Figure 1.1-6 (Estimating screen). When you are developing, you will not be writing to an estimating screen; you will be writing into the U-T-H \Job\Takeoff folder, which is the back-end structure. As you update this back-end structure, it updates the rest of the screens.

Figure 5	
Figure 6	

^{10.} The Pages tab works similarly (see figure 7). If a page is added on the back end, then it will reflect in the screens.



11. If you need to access templates, they can be accessed via the Storages (\Job\Storages). The example in Figure 8 shows only a Local storage being set up. If you have a LAN, that would show up as well. Templates can live anywhere on your network. Right-click on the Local to open the Properties box for Local.



12. Figure 9 shows the properties box for Local: Click on Advanced.

Name	Value	Units
Local		
Color	536,870,911.00	
Input Advance	d Form	Ok Cancel
	Figure	1.1-9
Figure 9		

13. Clicking on Advanced opens the Advanced Properties window (Figure 10), which will tell you exactly where your storage is physically located on the computer. If you have a network set up, then a property will be set up called **network.path**. Use the **item.fullpathproperty** within the API. Use the full path if you need to copy files to that network location.

Formula	Input Units	Adjust	Result	Output Units			-	
Local		T	have a					
Cools					a		52	
		-	LUCO		8	V		
					0.	V	100	
Storage	1	1	Storage	Ť.	æ	0		
						ò	Ť	
					8	ó		
-			-				-	
[\Settings\User]				1	8	0		
[!Date] [/Time]			1/2/2018 12:41:41 PM		0	0		
oge\ Form								Ok Cano
	[GettropUser] [Gata] (Tine)	[SettingsTate] [Cate] [Time]	[Settorp Xwe] [Deve] (Tree]	[Settrop Use] [Deve] (Tree]] 1/2/2018 12-41-41.94 99/	SettingsTate) (Cane) (Time) UCane) (Time) UCane) (Time) UCane) (Time)	Centrop: Low] 0 [Dentrop: Low] 0	CentropUsed Image: Centrop	Centrop:Used I/20018 I2-41-41 PM I/20018 I2-41-41 PM Upper I/20018 I2-41-41 PM II/20018 I2-41-41 PM

Figure 10

14. This completes the overview coverage of the COM and Scripting capabilities and how data is stored.

1244

Property Object Structure Overview

Property Object Structure Overview

This section describes the object structure of Items and Types and how a _Type can be modified into a new Type.

Understanding the structure of Items and Types is important to developers. **Type** is a very specific word that PlanSwift uses. A folder (in the Under-The-Hood window) is a physical Item of the Type "Folder." Everything in a Folder is considered to be an Item. Items are the "building blocks" of PlanSwift.

There are two types of **Items** in a **Folder**: one is "**Types**" and the other is "**_Types**." These are the two identifiers for what that **Item** is. The "**_Types**" is the base class (a master template containing the default properties) of all **Items** in PlanSwift. The **Type** allows the user to inherit a "**_Type**" and customize it into a new custom **Type**.

For example, the **Area Item Type** in PlanSwift can be modified to produce new measurement types, such as **Roof Area**, **Joist Tool**, **Grid Tool**, and more. Each of these new measurement types are built upon the base type for an Area Item. Figure 1 shows these custom area item types added to the Area drop-down menu on the takeoff ribbon group on the home tab ribbon-bar. The drop-down menu contains only some of possible modified Area Items that can be created. Users can create their own custom types and even add them to the drop-down menu if desired. Some modified types, like the **Roof Area**, have only simple modifications to their properties. Other modified types, like the **Joist Tool** and **Grid Tool**, have more complex modifications to their properties, such as scripted properties and custom sub-item section types. Regardless of the complexity, each of these custom types is built on the foundation of the original area item type. The Area item type along with the other takeoff types (Linear, Segment, and Count) are some of the essential "building blocks" of PlanSwift.

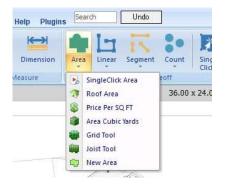


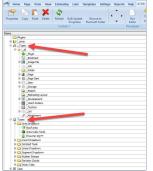
Figure 1

_Types (Property Object)

_Types

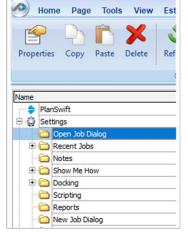
This section describes the object structure of **_Types**, how to access and view the properties of **_Types**, and how to configure the default setup configuration of **Items** within PlanSwift.

If you do not have PlanSwift open, follow steps 1-6 in the previous section (Item Structure Overview) to open PlanSwift and display the U-T-H tab; then click on the U-T-H tab to open the U-T-H window (see Figure 1 below). The two red arrows point at _Types and Types. Any new Types that you create will be visible here. Click on the Settings tab on the top ribbon menu.





2. Right-click on the **Open Job Dialog** folder in Figure 2.





3. Select Properties (Advanced) from the drop-down menu in Figure 3.

ame		
PlanSwift		
🖻 🥥 Settings		
Open Jo Recent Show Me	Add Delete	>
Notes Docking Scripting	Copy Paste	
Reports	Load Tree Unload	
Estimati	Break Link	
	Properties (Advanced)	
E Corages	Test	
Plugins Units	Rebuild	
± 🔐 _Types		
🛨 💭 Types		

Figure 3

4. Click on Advanced in Figure 4.

Nar	ne en Job Dialog	

Figure 4

5. This opens the Advanced **Properties** screen in Figure 5. Note that the **Type** of the Open Job Dialog is "**Folder**" (see **#1** in Figure 1.2-5). Note that the "Inheritance path is shown by arrow **#2**. Click in the **Folder** field to display a down arrow, then click on the down arrow.

lame	Formula	Input Units Adjust	Result	Output Units			
tem		March March					
Color				6	@		
Name	Open Job Dialog	4	Open Job Dialog	8	9	N	
other							
Type	Folder		Folder	8	9		
Icon	6			6	9		
Width	747		747	ď	0		
Height	660		660	ď	9		
Left	-1334		-1334	ď	9		
Тор	190		190	8	9		
DragDropAction	SubItem		SubItem	8	9		
Location	Local		Local	8 8 8	9		
Columns	(Size: 470)			8	9		
GridLayout	(Size: 835)	1		8	0		
Audit Trail							
Created By	[\Settings\User]				9		
Time Stamp	[IDate] [ITime]		1/3/2018 3:10:38 PM	đ	9		

Figure 5

6. This displays a drop-down menu that allows you to select the Type (see Figure 6). Select Item from the drop-down menu.

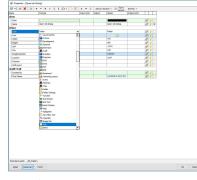


Figure 6

7. This opens the screen shown in Figure 7. Selecting Item changed the Folder Item type to a generic PlanSwift Item type. Note that the properties of a generic Item are significantly different than the properties of a Folder Type. Note also that the Inheritance path at the bottom of the screen now displays as

"_All_Ite	em". The Type controls the functionality of an Item and what that Item does. Click on Cancel at the bottom of the screen, and select No from the warning window
	that asks if you want to save any changes.

Figure 7

8. If you aren't at the U-T-H window, click on the U-T-H tab on the top ribbon menu. Now click on the + box next to _Types, then open the _All box the same way, and then click on _Item. You should see the same information as displayed in Figure 8. Types, then, are the general types of Items that we have. When we specify a Type with an underscore (_), we are specifying a base class of an Item or Item _Type. At the arrow 1 in Figure 8 you will see that _Item is broken down into three different _Items: _Takeoff Item, _Part, and _Assembly. Each one of these has a different configuration. A _Takeoff Item is a specific digitizer Item, meaning you're going to have an area, a line object (linear or segment), or a count object. Their primary function is to perform very specific actions to record data onto images. They are completely separate from any other Item type in PlanSwift and are the only items that can be used to record digitizing information. Click on the + next to _Takeoff Item.

Figure 8



Figure 9

10. Click on Properties (Advanced) in Figure 10.

Takeoff Item		
Area		
E-lat Line		
Linear Segment	Add >	
Count	Сору	
Material	Paste	
_Labor	Load Tree	
	Unload	
Other	Break Link	
Assembly		
_Storage	Properties (Advanced)	
_Report	Test	
_Estimating Layout	Rebuild	
_Development		
_Hatch Pattern		

Figure 10

11. Click on **Advanced** in Figure 11.

Nan	ne			
Li	ne			
Cold	or		Fill Type	
[!R	andomColor]	*	Solid	•
			4 L	

Figure 11

12. The Properties for the _Line are shown in Figure 12. The properties' rows may be shaded in blue, green, white, orange, yellow, and gray (not all shading colors are shown in this particular figure).

Blue Fill: The blue fill on the row indicates that the property on that row was inherited from the system (somewhere along the inheritance path)—as both a result and a formula (every value is available).

Green Fill: The green fill on the row indicates that the property on that row was inherited--as a result only--from the system.

White Fill: The white fill on the row indicates that the property on that row was created by the user or was green (no longer inherited) and was modified and turned white.

Orange Fill: The orange fill on the row indicates that you modified an inherited property but that you want to maintain the inheritance.

Yellow Fill: The yellow fill that you have locked the property.

Gray Fill: Gray fill indicates that the property is hidden.

To display any rows that are grayed (hidden), click on the Show:Normal selection indicated by the red arrow in Figure 12.

	• - • / () () · [] ()	A 7 2	Store	form Form	Erents -		
974	Foma	Doort new	Abut	Loved.	lassure		
han			[1.000	Terrar and a second		- 1
here	Line -			Life		LOIDI	7
Cruckler .						20	81
Dan #						20	
Car	Selection .			Map# Inm		- Giĝi	
stinatino							
lices	Ciden#1	17		6.00	10	40 01	
(otta)		5		6.00	4	20	
Arker N		N		6.00		60	
Most Taxe						00	
Cost Table	Kn/ * Ketten	1		6.00	1	60	
Netwine	Contract * (Provide NU/ 1993			5.00		60	
Natur Total	End * Personal			6.00		00	81
Postal	Contact + Ineito (etc)	5		6.00	6	60	8
P102 700/	Ight * Printent			6-00		60	
			-	1000	14	Ded 1 V 1	
links.	Transmost 1					000	2
ALL INC.	Delt.			Dela I		60	× 1
Paint.						6 6 6	<u> </u>
Parish Parlame Grain	1			1.00			<u> </u>
Parish Patient Color		-					Эł.
Carbon .		-		-			÷.
Tendure Dealer				ere bare o b		- 6 -	8
Column Texture				False.			
00041000	1	77		100	-	00	
Seturatings		77		1.00	6		
akeoff Euta						107 I V	
tial inte	Saw bad " tot was " to shod be	10.147		6.00	190.FT	10 2	
Indianet/	provide the second second	17		6.00	-	21	
and Delivery	1			100		60	
Didda				5.00	10	20	
Dealth				6.00	10	20	
Line 1000	Deets)	1		6.00	G	60	
104.707	447			Lut.		00	
Tukeo Y	direw Istal	PT.		6.00	-	8.4	H
think .	Sheed * Devid	0.77		6.00	QUET		4
dan.	Tarent .	80.77		6.00	BOFT	20	1
izer Tabl	Clarge Total	17		6.00	- C	60	H.
General Court	a based Cavel			6.00		60	8
	(removed	-		200		0 0	8
		-		6.00	F	60	H
Paint Court							
	(head)					20	

Figure 12

13. Note that the Show mode you are currently in is "Normal." Click on the All selection from the Show: drop-down menu (Figure 13).

Properties - [_Line]											
🛤 📲 😤 🗶 💷	+ - * / () () - 🖓 🖡	5 × ×	2	Show: Norma	-	Form Lavout	Events -				
ame	Formula	Input Unit	s	Input		1	Output Units				Γ
tem				✓ Normal							1
Name	_Line			Hidden				ß	0	•	1
Description	******			All	_			ß	9		1
Item #								ß	9		1
Туре	Takeoff Item			Ta	akeoff Iter	m		0	9		
Default Properties View	Form			Fo	orm			R	0		

Figure 13

14. Scroll down to the bottom of the window. A grayed row is now visible (see the red arrow in Figure 14); this row is gray because the item is hidden.

Padlocks: The column of padlock icons on the right allows you to lock or unlock whether the property Name, Formula, Input Units, Adjust, Result, and Output Units values (listed in the column headers) can be edited. Click on the padlock to toggle between locked and unlocked.

Lightbulbs: The row of lightbulbs to the right of the padlocks allows you to select whether the property for each row is visible in the Show: Normal mode. Clicking on a yellow bulb when in Show: Normal view causes the property on that row to disappear and be hidden. Clicking on a yellow bulb when in Show: All turns the bulb blue; a property row with a blue bulb will not be visible in the Show: Normal mode. Clicking on a blue bulb when in Show: All mode, turns the bulb yellow and allows the property to be shown again when in Show: Normal mode.

GUID	{65FEA8FC-4963-4FC5-9C31-3C4E857DEFF5}	{65FEA8FC-4963-4FC5	A 9 🗆
Icon	la la	In	8 9 0
IsItem	V 🔺	True	l 🖓 🖸
OrderIndex	1	1.00	l 🖓 🔍 🗆
DoubleClickAction	Record	Record	a 🖓 🗆
ShowChildInputs	2	True	l 🖓 🖸
DragDropAction	SubItem	SubItem	8 9 🗆
LaunchAction	Record	Record	a 🖓 🗆
Form Layout	xml version="1.0" encoding="iso-8859-1"?	xml version="1.0" er</td <td>an 🖓 🗆</td>	an 🖓 🗆
Isline	 Image: A start of the start of	True	8 9 0
AllowDragDropOnPage	×	True	890
AllowDragDropOnTakeoffItem		True	8 9 🗆
IsTakeoffItem		True	890
Subitem Type	Assembly	Assembly	8 9 🗆
udit Trail			
Created By	[\Settings\User]		8 9 🗆
Time Stamp	[!Date] [!Time]	1/5/2018 10:27:33 AM	8 9 🗆

Figure 14

15. Click on Show: All and select Show: Normal for the mode (see step 13, but select Normal from the drop-down menu) (Figure 15).

lame	Formula	Input Units	Adjust	Result	Output Units			Γ
tem					A			
Name	Line			_Line		S	Q	V
Description						S	0	C
Туре	Takeoff Item			Takeoff Item		B	0	E
Item #				2		B	9	
Default Properties View	Form			Form		B	9	
stimating								
Qty	[Takeoff]	FT		0.00	FT	æ	0	
Cost Each		s		0.00	\$	8	0	C
Markup %		%6		0.00	96	S	Ŷ	C
Cost Type						ð	0	E
Cost Total	[Qty] * [Cost Each]	\$		0.00	\$		0	
Markup Each	[Cost Each] * ([Markup %] / 100)			0.00		0	9	C
Markup Total	[Qty] * [Markup Each]	s		0.00	s	0	9	C
Price Each	[Cost Each] + [Markup Each]	s		0.00	s	0	0	C
Price Total	[Qty] * [Price Each]	s		0.00	\$	0	0	
ill								
Color	[IRandomColor]					B	Ŷ	V
Fil Type	Solid			Solid	-	S	0	V
								100

Figure 15

16. Click on Cancel to cancel out of the Properties - [_Line] window so that you do not save any changes. Now double-click _Area under _Item in Figure 16.

	Warning: Making changes to data Under The Hood can cause data muption/loss, and could even cause PlansWith to crash. Use at your own risk: and/with will not be respeciable for loss to incorrect data. Warning / Disclaimer	
TN		
PlanSwift		
Q Settings		
200		
Ca Storages		
Ca Plugins		
jū jīpes		
8 👷 🔊		
월 _ Tree 5 응 같		
🖲 🔚 _Jwage File		
dot dot dot /		
- 🙆 _ Folder		
E 📓 Jaga		
🗄 👩 Page Dem		
B () Jan	Properties - [_Area]	×
E S- Takeoff Item	Properties - ["Aces]	×
C Jaco	Name	
® [t] _Une	Area	
In Jinear Segment	Color Fill Type	
	[RandonColor] Bold	
D. Sogara		
Count		
: _Count		
t Count B Part ∭ Material		
Count	Irput Advanced Form II+II OX Ca	ncel
t _ Count B ♥ Part ♥ _ Lateral ♥ _ Lateral	Irout Advanced Form	ncel
Source	Input Advanced Form	ncel
Source Section Sectio	Ingut Adanced Form	ncel

Figure 16

17. This opens the **Properties -** [_Area] window. The _Area item is derived from the _Takeoff Item and has had additional properties added for that Item. All of the blue shaded rows of properties have been inherited from _Takeoff Item. Click on Cancel (Figure 17).



Figure 17

18. Double-click on the _Part Item, then click on Advanced (Figure 18).

Properties Co	Page Tools View	Refresh Bulk Up Propert Custom 1	date Browset	 Doc Editor	Warning: corruption/lo PlanSwift will	ss, and could e not be resposit					
Name											
B DiarSwift											
· Q Settro											
(i) (i) (i)											
* Co Stored	065										
D Plugine	8										
8 d Junits	5										
8-ga _Type	es										
8 A A	AJ.										
	Plugin										
- 53	Bookmark										
۰ 🖬	Smage File										
	_Job										
0	Folder										
	Folder Page					-					
* 3	Folder Page Page Item					@ Prope	ties - [_Part]				
	Folder Page Page Iten Iten					@ Prope	ties - [_Part]	Value	Us	its	
	Folder Page Page Iten Jten Control Takeoff Iten						ties - [_Part]	Value	Un	its	
	Folder Page Page Iten Jten Contractor Iten					Name _Part		Value	Us	łð	
	Jolder Page Page Iten Jten So Takcoff Iten					Name _Part _Materia		Value	Us	its	
	Jolder Page Page Iten Jten So Takcoff Iten					Name _Part _Materia _Labor	1	Value	Un	its	
	Folder Page Page Iten Jien Jien Takcoff Iten Ing Area Ing Area Ing Area Ing Area Ing Area Ing Area					Name _Part _Materia _Labor _Equipme	nt .	Value	Us	its	
- <mark>1</mark> 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3	Folder Page Page Jien Jien Calcolf Iten Calcolf Iten					Nane _Part _Materia _Labor _Equipme _Subcont	nt .	Value	Un	its	
	Folder Page Page Page Page Takeoff Item Takeo					Name _Part _Materia _Labor _Equipme	nt .	Value	Us	its	
- 1 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3	Folder Page					Nane _Part _Materia _Labor _Equipme _Subcont _Other	nt ract		Us		
10 8 8 8 9 9 8 9 9 9	Page Page Inn Jam Br Akcoff Item Count Count Page Inn Acco Inn Count Count Page Inn Acco Count Count Count Count Page Inn Jam Count					Nane _Part _Materia _Labor _Equipme _Subcont	nt .	Value Form	Un	ets Ok	
10 8 8 8 9 9 8 9 9 9	Polder Page Page Page Page Inn Jann Takooff Tann Takooff Tann Takooff Tann Takooff Tann Takooff Tann Takooff Tann Page Takooff Tann Ta					Nane _Part _Materia _Labor _Equipme _Subcont _Other	nt ract		Un		
- 1 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3	Fider Page Iten Jam Jam Jam Jam Jam Takeoff Iten Jam Takeoff Iten					Nane _Part _Materia _Labor _Equipme _Subcont _Other	nt ract		Us		
- 1 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3	Polder Page Page Page Page Inn Jann Takooff Tann Takooff Tann Takooff Tann Takooff Tann Takooff Tann Takooff Tann Page Takooff Tann Ta					Nane _Part _Materia _Labor _Equipme _Subcont _Other	nt ract		Un		Cance

19. The _Part Item is the base class of _Material, _Labor, _Equipment, _Subcontract, and _Other. All of these inherit from _Part. Now double-click on the _Assembly Item (Figure 19).

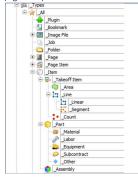


Figure 19

Figure 18

20. The _Assembly Item (see Figure 20) doesn't have any child items, because _Assembly is basically a container of parts. It only inherits from _Item in the same way as _Part inherits from _Item.

\$P +C (* ¥ 3	* = * / () () - 2	0 + + 5	Show: Normal - 10 100	Events +		
rane	Pomula	2007/01/2	Alut Salt	OVER CARLON BE		т
Dom						
bare	Asardy		Jaanday		2 9 9	6
7.94	Tan .		(3m)		20	
Description					30	
Dan P					00	
t-ctinating						
CO:			6.00		000	
Contech		5	6.00	6	2 2 2 3 3 4 2 2 2 3 4 3 4 3 4 3 4 3 4 3 4 3 4 3 4 3 4 3 4	
Neko N		N	6.00	*	10 0 0	
Cost Tape					000	
Cent Turisi	(Ord * [Continue]	8	6.00	4	- A V .	
Nekelish	(Caritan) * (Nexus N/ 2008		6.00		890	
Nether Tried	(cn) * here each		6.00	1	000	51
Prior Each	(contact) + (Markup Each)		6.00	1	90	51
Prior Table	Kind * Prosident		6.00	1	50	
Takeoff Epita						
A10	64	59.07	16:00	5077	000	
Linear Tobal	64	PT	6.00	12	000	51
Paint Court	64	44	6.00	KA.	00	51
Segnent Count	64	-	6.00	84.	222	51
Sector Count	61	EA	6.00	RA.	00	51
Nork Treakdown Gre	eten					п.
Draken					00	
- Subtraint					- Digit	
Place.					- Giêlî	
CORTCOM					- Gi Qi C	
Locetor					000000	
Load					000	
2mg					60	
Public 2uth	[Dev/Tell				66	
faller	Taxed Tube 1				65	8
Other				-		٩.
law					00	1
DealerChilde	Subtrant.		Datan.			1
1000	~ ~		2.4			1
DISDUATE	Soldian		Sellam			1
Shour Matrice IN	2		24		Laler	5
Inchatter	have		hire		80	
Interentia			24			1
Subtem Tune	-				60	÷
Ande Trail					N Y	4
Created Br	(petroaspect				29	

Figure 20

21. In summary, PlanSwift uses the _Types Items to configure the default setup configuration of Items within PlanSwift.

Types (Non-Underscore "Custom" Types)

Types (Non-Underscore "Custom" Types)

This section describes the object structure of Types, how to access the advanced properties, and how those advanced properties may be changed.

 The next steps will cover the Types (without the underscore) Items. These Types are custom types that are added by customers or developers who needs to add additional functionality to the default class Items without having the need to go in and change the default properties. When working with Types, developers would not use the Under-the-Hood (U-T-H) tab, but would instead go to the Templates tab, then the Types tab. Types may not be visible from the Templates tab ribbon menu (Figure 1); if it is not, then first click on Settings on the Main Menu ribbon bar.

Home Page Tools	View Estim	ating Lists	Templates	Settings	Reports	Help	U-T-H Plugins
New New New Folder * Item * Area *	New New Linear * Segme	w New	New Assembly	New Part *	New from Type *	Delet	e Columns
						CONTRACTOR OF THE OWNER.	
					Tem	plate To	015
Sample Takeoff Templat	es 👔 Parts and	Assemblies	Custom Lini	s My Ta			ois
Sample Takeoff Templat Name	es 👔 Parts and	Assemblies Price Each	Custom Lini	s My Ta			
				us My Ta	b]		
Name				s My Ta	b]		

Figure 1

2. After clicking on Settings (#1 arrow of Figure 2), click on Interface (#2 arrow), then click the checkbox for Show Types Tab in Templates Screen (#3 arrow).



Figure 2

3. Now click on **Templates** tab (Figure 3) and you will see the yellow **Types** tab, which is orange in color and has a cog to the left of it. Click on **Types**: yours will look similar to Figure 3, except that the window shown below has most of the + boxes clicked on to display the sub-Items. PlanSwift comes with several custom Items. These items are categorized according to the base digitizer class type. Open Scripted Tools, then Items, then double-click Joist Tool to see its properties.



4. The Joist Tool properties are shown in Figure 4. The Joist Tool template has been configured to completely handle a joist tool layout of various sizes. Click on Advanced to see the advanced Joist Tool Properties.

Name	
Joist Tool	
Joist Type	
	•••
Length Calculation	Joist Direction
None 👻	Select parallel side 👻
O.C. Spacing	Pitch
	0 •
Min Length	New Length
Color	Fill Type
[IRandomColor]	Solid 👻
 ☑ Group Sections ☑ Generate Parts 	Automatically Generate
iput Advanced Form	Ok Canc

Figure 4

5. Every Item has a Property Structure, which remains the same for all Items, because all Items inherit from their parent-class Item. Based on the Type you have selected, various properties may be added. Each property for an Item has columns (see Figure 5) specifying the property's Name, Formula, Input Units, Adjust, Result, and Output Units.

	* - * / (10 - 2			mai + III Farm	Events -		
ane .	Portula	TRUE DISK	ASIE	Read	OLGUE LINES		T
bom							
7.04	Ave			down.		1.0	
Seargion						1	
dam a						4	
Time	Jul 100			DOUT THEY		d	9 2
Netwyk				9.88		- 141	V 2
Amu Langth				0.88		141	9 2
stinating							
jobi	Cautt	SOFT		0.88	92.61	141	
Contact		16		0.88	5	8	
Markup N				0.88	~	1	
Kost Tipe						10	
Kost Telai	Stat * Keel Read	4		0.88	5	e	
Nake-fadt	(Continue) * (Markey %) / 200	4		0.88	1	1000	
Planker Tabel	POVI * NetwiseN			0.00	1	A.	
Postab	Contract + Mahartachi			0.00		0	
Prior Table	(gy) * Postabl	-		0.00	1	e	
	Net Present				1.	[w]	×11-
low	(Automote)			_		1.07	
of the	Selec			Self		- 61	å b
AND						10	àle
Hest Patent Sole				1.00		4	312
Capital Capita	0			100.00		1	
and being Color		_			_	1	3 5
John .		_				3	310
Sofur Inge				war South Cr 2		3	315
Coloria Trabar	0			The		4	315
Contra locare				1.00			215
Columnia and	1			1.0	ax ax	1	
	1	17		100	jox.	- a -	10
about Cuta		100			100		
Depth				9.88		_ a	
Culume .	(Avoi * Draft)	0.077		0.00	C1 FT	1	
nativeget						_ d 1	
COLUMN AND A	Ener Trial 10ral regio	5077		0.88	52.07	8	
fuiao#	Deed	50/T		0.88	92.07	6	
Assist Path	(Seed 10(ext(2)(ext(2))) = 246/123	SQ/PT		9.81	62.07	10	
A10	(See)	SQ/FT		0.00	62,87	8	
Linear Total	(Green Tarial)	PT .		0.88	PT	8	
Segnent Court	(Depart Court)			0.88		1	
Paint County	(Percount)	EA.		0.88	6A	a l	
Celsali	(Tenel)			0.88			
				0.88		100	



Name: The Name column identifies the name of the property. When coding, a developer will access this property either via the name or via an index.

Formula: The next column, Formula, allows the developer to input a formula and/or a numerical value. Variables are placed in brackets; operators, such as +, -, /, and * (and others, such as sin, cos, tan, etc.) may be used to operate on any variables or numerical values. The operators that are available for use in formulas are the same ones that would be available in a calculator or in the scripting language being used.

Input Units: The Input Units column allows the developer to select the input units. This can be inches (IN), feet (FT), yards (YD), miles (MI), millimeters (MM), centimeters (CM), meters (M), and kilometers (KM), each (EA), square inches (SQ IN), square feet (SQ FT), square yards (SQ YD), square miles (SQ MI), square millimeters (SQ MM), square centimeters (SQ CM), square meters (SQ M), square kilometers (SQ KM), cubic inches (CU IN) through cubic miles (CU MI), cubic millimeters (CU MM) through cubic kilometers (CU KM), and dollars represented with the dollar sign (\$). The Input Units calculation operates on the value developed by the Formula column.

Adjust: The Adjust column allows the developer to enter an adjustment, such as a waste percentage or a numerical value, to the number developed in the Formula column. Percentage numbers are followed by a % sign.

Result: The Result column takes the adjusted value and displays it as the Result in the units that are specified in the Output Units column.

Output Units: The Output Units specifies the units that the Results column displays in the same units listed in the Input Units column. Changing this to a different unit, such as inches, will convert the result to the selected unit. Of course, if you attempt to convert yards into cubic yards, you will get a conversion error since it is not an "apples to apples" conversion.

- 6. To see this in action, enter 10 in as the value of the New Length in the Formula column (red arrow #1 in Figure 6.
- 7. Enter [New Length]/2 in as the value of Min Length in the Formula column (red arrow #2 in Figure 6).
- 8. Click on the down arrow next to the parentheses pointed to by arrow #3 in Figure 6), and select RoundUp() (arrow #4).

Properties - [Joist Iool]									
8 =8 😭 🗶 💷 +	- * / ()	0 - 🕤 🛍	▲ → 2	Show: N	ormal + 🛅 For	m Events •			
me	Formula	()	E	Adjust	Result	Output Units			
em		RoundUp()	4 7						
Туре	Area	RoundDown()	1	Area		8	8	
Description				5			8	0	
Item #							8	0	
Name	Joist Tool	1			Joist Tool		8	0	
Min Length	[New Length]/2	2			5.00		8	8	
New Length	10	-			10.00		8		2

Figure 6

9. The equation Min Length/Formula equation will now be displayed as RoundUp([New Length]/2). The result will appear as 5.00 in the Result column. Click on the Min Length value in the Input Units column and select FT for feet (Figure 7). Note that the OutPut Units columns in the Min Length row changes to FT as well. If you want Output Units displayed in yards, click in the field where Min Length and Output Units meet, and select YD. For this exercise, however, keep the output set to FT.

Properties - [Joist Tool]							2
ቱ 🗠 📽 🗙 🗀 🖡	+ - * / () () - 🖓 🖡	5 ▲ ▼ 2 :	Show: Normal 👻 🛅 For	m Events -			
Vame	Formula		djust Result	Output Units			
Item							
Type	Area		Area		S	9 C	5
Description					S	90	
Item #					ß	90	
Name	Joist Tool		Joist Tool		ß	9	
Min Length	RoundUp([New Length]/2)	- 15	5.00	FT	ß	9	
New Length	10		10.00		8	9	2
Estimating		YD					
Qty	[Takeoff]	MI	0.00	SQ FT	ß	9 C	5
Cost Each		CM	0.00	\$	ß	9 C	5
Markup %		M KM T	0.00	%	ß	9 C	5
Cost Type					P.		5

Figure 7

10. Now enter **10%** in the field intersecting the **Min Length** row and the **Adjust** column to add a waste factor of 10% (Figure 8). Note that the **Result** now reads **5.50** since the 10% waste factor has been added. If you want a non-percentage value added, simply enter the numerical value without the percent sign.

# 📲 😭 🗶 💷 +	- * / () () - 🖓 🛍	★ ₹ 3	Show: N	Iormal 👻 🛅 Forr	n Events - ut			
me	Formula	Input Units	Adjust	Result	Output Units			
em								
Туре	Area			Area		8	-	
Description						8	9	
Item #						ð	9	
Name	Joist Tool			Joist Tool		8	9	V
Min Length	RoundUp([New Length]/2)	FT	10%	5.50	FT	8	0	
New Length	10			10.00		8	9	•

- 11. Selecting the **padlock** allows you to either lock or unlock the property.
- 12. Clicking on the light bulb either hides (blue) or unhides (yellow) the property.
- 13. Selecting the **box** (a check in it) will cause the property to be shown on the **Form** when the application is started. Note that **Name**, **Min Length**, and **New Length** have check marks in the check boxes. Click on **Form** at the bottom of the **Joist Tool** window (Figure 9).



- 14. Now note that the Name, Min Length, and New Length fields, along with their values, are displayed in this form, because the boxes for the same fields in the Advanced properties window are checked.
- 15. All of these properties are also available in the COM object.

Object Property Model

Object Property Model

This section describes how to create new items and how to set up and modify attributes to properties of Items.

 If you are not still on the Advanced Joist Tool Properties window, then click Templates tab, click Types tab, open Scripted Tools folder, open Items folder, double-click Joist Tool, then click on Advanced. This window is divided into ten different groups: Item, Estimating, Fill, Takeoff Data, Work Breakdown Structure, Other, Audit Trail, Joist Properties, Videos, and Events (Figure 1).

e 😪 😭 🗶 🔟	+ - * / () () - 🕤	🛍 🔺 👻 🕄	Show: N	Iormal + 📰 Fo	rm Events •			
arre	Formula	Input Units	Adjust	Result	Output Units			
tem								
Туре	Area			Area		8	9	
Description						8	9	
Name	Joist Tool			Joist Tool		8	0	
Item #						S	9	
Min Length				0.00		8	9	
New Length				0.00		S	9	•
stimating								
Qty	[Takeoff]	SQ FT		0.00	SQ FT	di	9	
Cost Each		\$		0.00	\$	8	9	
Markup %		5		0.00	%	8	9	
Cost Type			-			8	9	
Cost Total	[Qty] * [Cost Each]	\$		0.00	\$	2	9	
Markup Each	[Cost Each] * ([Markup %] / 100)	\$		0.00	\$	0	9	
Markup Total	[Qty] * [Markup Each]	\$		0.00	\$	0		
Price Each	[Cost Each] + [Markup Each]	\$		0.00	s	2	0	
Price Total	[Qty] * [Price Each]	\$		0.00	\$	0	0	
test mine				0.00		ð	Ŷ	
Color	[IRandomColor]			and the second sec		6	Ŷ	
Fil Type	Solid			Solid		8	9	
Hatch						8	9	
Hatch Pattern Scale	1			1.00		8	9	
Transparency	0			100.00		S	0	

Figure 1

2. Double-click on the Cost Total field under Estimating. This opens the Edit Property window (Figure 2) for the Joist Tool's Cost Total property, which is grouped under Estimating. Developers can use this window to modify the physical property settings (or attributes) of the property model. Everything that is in the Edit Property window is available through API. This is where a developer will spend a lot of time setting up attributes to properties of Items, so it's very important to understand what the functions of these properties do. Note that the descriptions that follow are modeled for COM rather than for scripting. Click on Cancel to close the Edit Property window, then click on any item in the first column of the Estimating group.



Figure 2

3. Now click on the **Add Property** icon as shown in Figure 3.

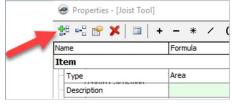


Figure 3

4. This opens a new Edit Property window (Figure 4). Notice that the Name: field is blank in this window, allowing you to give it a name of your choosing. The Type: field's down-arrow opens a drop-down menu that allows you to specify the type of value assigned to the

field: Number, Color, Text, Memo, CheckBox, Path, Image, Large Image, Type, Script, File, Large File, File Name, Folder, Font Name, Connection String, Slider, and Dimension. The Type: field's default is Number. The attributes a developer would most commonly use are Text, Memo, or Number. A few others, including CheckBox and Slider could also be useful. Selecting CheckBox would display a checkbox, which represents a boolean value (true or false): if it is checked, it is true; if not checked, then it is false. Enter the name Test Property in the Name: field; click on the Type: field's down-arrow and select

CheckBox from the menu, and then click on OK.

Name:				List Type:	
Type:	Number		Expression	Simple List	
Group:	Estimating	•		Only allow choosing from the list	
ool Hint:				Parse formulas	
	Remember Value	arse Por	mula		
put Opt	ions				
C Input	Condition:				
ompile	d Options				
Deny	Read Deny Write D	eny OL	E/Script Access		
Input	Units:	- [Hidden		
1	Units:	•	Locked		
Decimal P	laces: 2				
When cre	ating new items of this type:				
Norm	al 🔿 Inherit 🔿 Ignore				
Form					
() Resu	t Calculate before	inherit			
Pull	From:				
				1	

Figure 4

5. You will now see "Test Property" at the bottom of the first column in the Estimating group (Figure 5). The next column to the right is the checkbox. Since the checkbox is not checked, its value is False, as seen in the 5th column (Result) to the right. Once checked, its value shows as True.

Qty	[Takeoff]	SQ FT	0.00	SQ FT	<u></u>	
Cost Each		\$	0.00	\$	8 💡	
Markup %		%	0.00	%	8 😔	
Cost Type					8 9	
Cost Total	[Qty] * [Cost Each]	\$	0.00	\$	0 9	
Markup Each	[Cost Each] * ([Markup %] / 100)	\$	0.00	\$	0 9	
Markup Total	[Qty] * [Markup Each]	\$	0.00	\$	0 9	
Price Each	[Cost Each] + [Markup Each]	\$	0.00	\$	0 9	
Price Total	[Qty] * [Price Each]	\$	0.00	\$	0 9	
test mine			0.00		<u> 8</u>	
Test Property			False		<u> </u>	

Figure 5

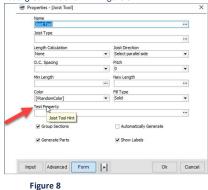
6. Double-click on Test Property to open its Edit Property window again (Figure 6). This time select the Slider tool from the Type: drop-down menu. This opens the Slider Options (see red arrow). The Slider function is very useful and has its own properties, which is the minimum value and the maximum value. Enter "1" in as the minimum value and "100" in as the maximum value; set the Tick Frequency: field to 10 so that the ticks will show up every tenth time; and check the box Show Ticks. Now click on OK.

Figure 6

- 7. The **Test Property** property will now show a slider bar with eleven tick marks. Click and hold on the slider on the bar, and drag it to anywhere on the bar; as you drag it, the value will be displayed in the 5th column (**Result**). The **Slider** function can be valuable in cases where you might have an image transparency function, or you need a finite number based off a value, or you need any type of minimal adjustment. Click on **Type:** again and select **Number**.
- 8. The other Type: field values may be useful but will not be covered at this time.
- 9. The Group: field shows that Test Property is assigned to the Estimating group. Clicking on the down-arrow allows you to assign it to one of the ten available groups in the Joist Tools Properties window: Item, Estimating, Fill, Takeoff Data, Work Breakdown Structure, Other, Audit Trail, Joist Properties, Videos, or Events.
- 10. The **Tool Hint:** field allows you to enter a short description of the tool, which will be visible in the **Form** window when the cursor is hovered over the property. Such a hint can be helpful to explain the functionality of a property to a user. To see this in action, type "*Joist Tool Hint*" in the **Hint:** field; then click on **OK** to close the **Edit Property** window.
- 11. At the Joist Tool Properties window, click on the box for the Test Property property you created (see arrow in Figure 7) so that it can be displayed in the Input and Form windows.

Figure 7

12. Scroll to the bottom of the Joist Tool Properties window and click on Form, then hover over the Test Property text until the "Joist Tool Hint" appears with a yellow background as shown in Figure 8.



13. Click on Advanced again to return to the Joist Tools Properties window, double-click on the Test Property property you created previously, and delete the Tool Hint: text. The Edit Property window for the Joist Tool should look similar to Figure 9.

Name:	Test Property		List Type:	
Type:	Number	Expression	Simple List	
Group:	Estimating	•	Only allow choosing from the list	
Tool Hint:			Parse formulas	
	Remember Value Par	se Pormula		
input Opti				
🗹 Input	Condition:			
Compiled	Options			
Deny	Read Deny Write Den	vy OLE/Script Access		
Input L	inits:	Hidden		
	inits: •	Locked		
Decimal Pla	eces: 2			
	eces: 2			
When cre	ating new items of this type:			
When cre	ating new items of this type:			
When cre Norma Formu	ating new items of this type:	herit		
When cre Norma Formu Result	ating new items of this type: al O Inherit O Ignore de t Calculate before in	herit		
When cre Norma Formu Result	ating new items of this type: al () Inhent () Ignore	herit		
When cre Norma Formu Result	ating new items of this type: al O Inherit O Ignore de t Calculate before in		Cancel	

- 14. The Remember Value: checkbox has no functionality at this time.
- 15. The Parse Formula: checkbox, when checked, causes anything within brackets to be read as a variable. If this box is not checked, then the text inside the brackets is read as a text string, not as a variable. By default in COM, this box is checked automatically.
- 16. In the Input Options area of the Edit Properties window, there is an Input checkbox and a Condition: field. When the Input checkbox is checked and the condition in the Condition: field is satisfied, then the property will be displayed in the Form window. If the Condition: field is not satisfied, then the property will not be displayed in the Form window. If the Input box is checked but the Condition: field is blank, then the property will show up in the Form window. If the Input box is not checked, then the property will not be displayed in the Form window. If the Input box is checked but the Condition: field is blank, then the property will show up in the Form window. If the Input box is not checked, then the property will not be displayed in the Form window. As an example, enter [New Length] = 10 into the Condition: field. Also, make sure the Input box is checked. Click on OK to close the Edit Property window. Click on Form at the bottom of the Joist Tools Properties window. You'll see in Figure 10 that there is no Test Property field. You will also notice that the New Length field is blank. Now put the value of 10 into the New Length field and press the Tab key to invoke the changed value.

Name	
Joist Tool	
Joist Type	
Length Calculation	Joist Direction
None 👻	Select parallel side 👻
O.C. Spacing	Pitch
-	0 -
Min Length	New Length
Color	Fill Type
[!RandomColor]	Solid 👻
Group Sections	Automatically Generate
Generate Parts	Show Labels
put Advanced Form	Ok Car

Figure 10

17. The Test Property field now appears (Figure 11). If you change and invoke the value to anything but 10, then the Test Property field will disappear.



- 18. Click on Advanced to return to the Joist Tool Properties window (Figure 11), and double-click on Test Property. The Compiled Options are shown in Figure 12 and will not be discussed at this time.

Edit Prop	perty				
	Test Property			List Type:	
Type:	Number	-	Expression	Simple List	
	Estimating	•		Only allow choosing from the list	
Tool Hint:				Parse formulas	_
	Remember Value	Parse For	nula		
Input Optio					
[∐ Input	Condition: [New Le	engithij = 10	_		
Compiled	Options				
Deny R	beed 🗌 Deny Wri	te Deny OLE	/Script Access		
Input U	nits:	•	Hidden		
U	nits:		Locked		
Decimal Pla	case: 2				
When crea	ding new items of thi	s type:			
Normal	O Inherit O Ign	ore			
Formula	•				
O Result	Calculate	e before inherit			
Pull Fr	iner-				
				1	
			A		
			/ Ok	Cancel	

- Figure 12
- 19. The **Input Units** and **Units** (Output Units) area allow you to specify the **Input Units** and **Output Units** columns in the **Joist Tool Properties** window (see Figure 13). You may specify whether they are to be hidden or locked by clicking the check boxes for **Hidden** or **Locked**. You may also specify the decimal places by entering a decimal value in the **Decimal Places**: field.

e carri	operty				
Name	Test Property			List Type:	
Type	Number	•	Dipression	Simple List	
Group	Estimating			Only allow choosing from the list	
Tool Hint	Joist Tool Hint			Parse formulas	
	Remember Value [Parse For	mula		
Input Op					
🗹 Inpu	t Condition:				
Compile	d Options				
Den	y Read Deny Write [Deny OLE	E/Script Access		
Input	Units:	-	Hidden		
-	Units:	•	Locked		
Decimal	Naces: 2				
	eating new items of this typ	pe:			
	nal O Inherit O Ignore				
Forr Rea		fore inherit			
Pul	From:			1	
			Ø Øk	Cancel	

Figure 13

20. The When creating new items of this type: area can be set to Normal, Inherit, or Ignore (Figure 14). The Normal setting allows for inherited properties to be editable. The Inherit setting allows properties to be modifiable but only by permission. The Ignore means that "anytime I inherit a property, I specifically do not

want this property to be on that inheritance of that derived item.

Name:	Test Property	Expression	List Type:	
Type:	Number	▼ Cxpression	Simple List	
Group:	Estimating	-	Only allow choosing from the list	
Tool Hint:	Joist Tool Hint		Parse formulas	
	Remember Value Part	se Formula		
Input Opt	ions			
🗹 Input	Condition:			
Compileo	d Options			
Deny	Read Deny Write Der	vy OLE/Script Access		
Input	Units:	Hidden		
31	Units:	Locked	I	
Decimal Pl	aces: 2			
When cre	ating new items of this type:			
Norm	al 🔿 Inherit 🔿 Ignore			
Form	da			
O Resul	t Calculate before in	herit		
Pull	From:			
			1	
		e ok	Cancel	

Figure 14

21. Checking the Formula checkbox allows the formula and the result to be inherited. Checking the Result checkbox allows only the result to be inherited. The Pull From: field is not commonly used. It allows you to inherit the actual result from a completely different item from either the same estimate or somewhere else by providing the relative path of that item and the property. The List Type: field allows the developer to provide a list that acts as a drop-down list to the property (see Figure 15) but will not be discussed at this time.

Lat Type: Deficient in the second se
n Simple List Simple List List Tree List Execute Plugin
Single List Single List List Tree List Execute Plugin
List Tree List Execute Plugin
Execute Plugin
-
-
-
-

Figure 15

22. This completes the coverage of what you need to know to get started using the API.

Job

Job

Accessing the Active Job

Job is the current active job in PlanSwift. If there is no active job open, then Job will be null (empty). Once a job has been opened, this job property will update to the current job's pages, links, takeoffs, remembered values, autolist, bookmarks, and notes.

\Job is the relative path to access the Job folder. Under the \Job folder are folders for Pages, Takeoff, and Bookmarks. Pages items reside in \Job\Pages, Takeoff items in \Job\Takeoffs, and Bookmarks items in \Job\Bookmarks.

Ø	Home	Page	Tools	s Viev
NS I	2	D	R	X
Prop	erties	Сору	Paste	Delete
Name	PlanSv	.: 0		
ΦØ	Setting	all and a second second		
P.D	Job	40		
	-	ges		
	Lin			
÷	Ta	keoff		
	Re	member\	/alues	
	AL	ItoLists		
Đ	Bo	okmarks		
÷	O No	tes		
D	Storag	jes		

API Calls

Delphi

Using IItem Object Model	> Expand source
Using PlanSwift Object Model	> Expand source

C#

Using IItem Object Model	> Expand source
Using PlanSwift Object Model	> Expand source
VB/VBA (OLE)	
Using Iltem Object Model	> Expand source

> Expand source

Using	PlanSwift	Object	Model

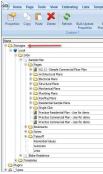
Storages

Storages

Accessing storage locations

Storages are the job storage locations. Storages are PlanSwift's job storage locations. They are unloaded by default and only accessed when opening a job.

Through the API, the parent's folder of each hierarchy must be loaded before accessing the children. Templates are accessed via the Storages (\Job\Storages).



API Calls

Delphi

Usin	g Iltem Object Model
1	
	Coming soon
Usin	g PlanSwift Object Model
Usin; 1	g PlanSwift Object Model

C#

Usin	ng Iltem Object Model
1	Coming soon
Usin	ng PlanSwift Object Model
1	Coming soon

VB/VBA (OLE)

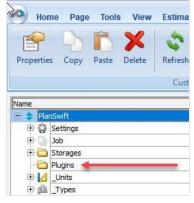
Using Iltem Object Model	
1	Coming soon
Usir	ng PlanSwift Object Model
1	Coming soon
	1262

Plugins (2)

Plugins

Accessing Plugins

This directory houses all installed plugins. Plugins can be created, modified, updated, and deleted.



API Calls

Delphi Using Iltem Object Model Coming soon Using PlanSwift Object Model 1 Coming soon 1

C#			
Using Iltem Object Mo	odel		
1 Coming soon			
Using PlanSwift Object	ct Model		
1 Coming soon			

Using IItem Object Model			
1			
	Coming soon		
Usin	Using PlanSwift Object Model		

_Units

_Units

Accessing _Units

_Units are conversion units. _Units should not be modified.

Hon 📀	ne Page	Tools	View	Estimat
P	D	ĥ	×	3
Properties	Сору	Paste	Delete	Refresh
				Custo
Name				
🗉 🔶 Plan	Swift			
E ()	Settings			
÷ 🗋	Job			
E 🗀	Storages			
6	Plugins			
E M	Units 👍	_	_	0
20	IN			
	FT			
1 1 Sec.	YD			
38 33	MI			
	MM			
	CM			
	M			
	KM			
	EA			
±-@	Types			

API Call:

Delphi Using Iltem Object Model 1 Coming soon Using PlanSwift Object Model 1 Coming soon

C#
Using IItem Object Model

Coming soon
Using PlanSwift Object Model

Coming soon

1	g PlanSwift Object Model Coming soon	
Using Iltem Object Model		
1	Coming soon	

_Types

_Types

Accessing _Types

_Types are the default types of all PlanSwift items. _Types should not be modified. Modifying any _Type will have an adverse effect on PlanSwift. Any modifications will be overwritten when PlanSwift is reinstalled or updated.

Properties Copy Paste Delete Refresh Bulk Prc Custom 1
Pro
Custom 1
Name
😑 🐤 PlanSwift
🗄 🤪 Settings
🕒 🕀 Job
🗉 🧰 Storages
Plugins
🕀 🛃 _Units
🕀 🙀 _Types 🔸
🖻 🙀 📶
Plugin
Bookmark
E Image File
Job
Folder
⊕ I Page
E 🛱 Page Item
1 Item
E Storage
Report
Estimating Layout
Development
Hatch Pattern
Texture
E List
@_Attachment
± © Types

API Calls

Delphi

Usin	Using Iltem Object Model		
1	Coming soon		
Usin	Using PlanSwift Object Model		
1	Coming soon		

C#

Usin	Using Iltem Object Model		
1	Coming soon		
Usir	Using PlanSwift Object Model		
1	Coming soon		

Using IItem Object Model		
1	Coming soon	
Usin	g PlanSwift Object Model	

Types

Types

Accessing Types

Types are derived from _Types. Special caution needs to be taken when modifying Types because modifying Types can cause an adverse effect on PlanSwift's operation. The Types are listed below.

Types
Area Dropdown
Roof Area
Area Cubic Yards
Price Per SQ FT
Count Dropdown
Labeled Count
Scaled Count
Circle
Square
Diamond
Plus
Triangle
Scripted Tools
Sections
Beam Section
Grid Section
Joist Section
Joist Line
Scripted Tools
Items Beam
Tool
Grid Tool
Joist Segment
Joist Tool
Joist Material
Beam Material
Linear Dropdown
Wall Area
Linear Cubic Yards
Segment Dropdown
Segment Cubic Yards
Hip-Valley Tool
Wall Area
Rubber Stamps
Approved
As-Builts
Bid Set
Canceled
City Approved
Confidential

Construction Set

	Draft
	Field Set
	Final
	Not Approved
	Not for Construction
	Pending
	Preliminary
	Received
	Revised
	Priority
	Update
	Mine
Revi	sion Clouds
	Revision Cloud
Stick	ky Tabs
	URGENT
API (Calls

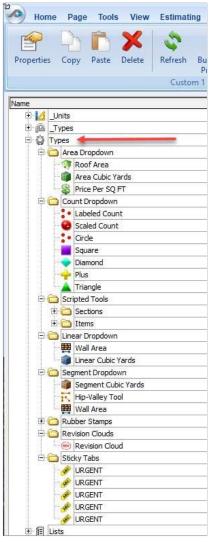
Delphi

Usin	Using Iltem Object Model		
1	Coming soon		
Usin	Using PlanSwift Object Model		
1	Coming soon		

C#

Usir	Using Iltem Object Model		
1			
	Coming soon		
Usir	ng PlanSwift Object Model		
1			

Using	Using Iltem Object Model		
1	Coming soon		
Usin	Using PlanSwift Object Model		
1	Coming soon		



Lists



Accessing Lists

Coming soon.

	ome Page Tools View
They	
Proper	ties Copy Paste Delete
Prope	iles Copy Paste Delete
Name	
the second se	'lanSwift
DALL NO. OF CO.) Settings
±.	Job
Đ.	🗋 Storages
-	D Plugins
+	🖞 _Units
÷.	L _Types
+	Types
Ē-	🛛 Lists 🚽
	PlanSwift
1.0	Reports

Coming soon

API Calls

1

Delphi	Delphi		
Usin	g Iltem Object Model		
1	Coming soon		
Usin	Using PlanSwift Object Model		
1	Coming soon		

C#

Using Iltem Object Model			
1 Coming soon			
Using PlanSwift Object Mo	lel		
1 Coming soon			

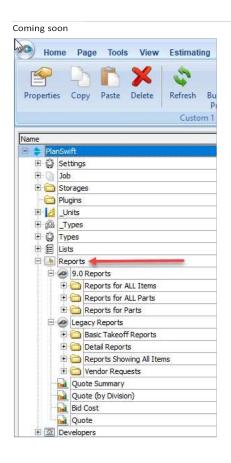
Using	Using Iltem Object Model	
1	Coming soon	
Using	g PlanSwift Object Model	

oming soon

Reports

Reports

Accessing Reports



API Calls

Delphi

Usin	Using Iltem Object Model		
1	Coming soon		
Usin	Using PlanSwift Object Model		
1	Coming soon		

C#

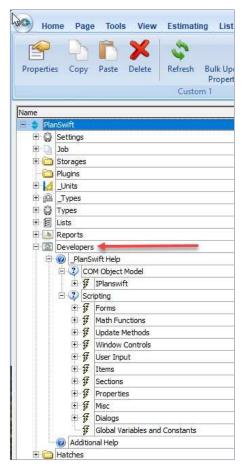
Usir	Using Iltem Object Model		
1	Coming soon		
Usir	ng PlanSwift Object Model		

Usin	g Iltem Object Model		
1	Coming soon		
Usin	Using PlanSwift Object Model		
1	Coming soon		

Developers

Developers

Accessing Developers



Coming soon.

API Calls

Delphi

Usin	Using Iltem Object Model		
1	Coming soon		
Usin	Using PlanSwift Object Model		
1	Coming soon		

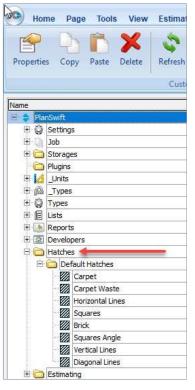
	C#
ŀ	1 Coming soon
	Using PlanSwift Object Model
	1 Coming soon

Usin	Using Iltem Object Model		
1	Coming soon		
Usin	Using PlanSwift Object Model		

Hatches

Hatches

Accessing Hatches



Coming soon.

AF Delphi	API Calls Delphi				
Usin	g PlanSwift Object Model				
1	Coming soon				
Licin	g litem Object Model				
USIN	g men object woder				
1	Coming soon				

C# Using Iltem Object Model

ming soon

1	Coming soon		
Ising P	PlanSwift Object Model		



Estimating

Estimating

Accessing Estimating

Hon	ne Page Tools View	Esti	
The			
E		9	
Properties	s Copy Paste Delete	Refr	
		ç	
Name			
	Swift		
E Q	Settings		
Ð D	Job		
Ð 🔂	Storages		
0	Plugins		
E 🚺	_Units		
± 🙉	_Types		
🕀 🥥 Types			
E E Lists Reports			
			1.1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1	Hatches	-	
	Estimating		
	training layout		
	Default		
	PS Layouts		
	Takeoff Data		
	Takeoff View		
	Wall Area		
	Round Qtys		
	Multiply and Location		
	Qty and Qty 2		
+ C	Textures		

Coming soon.

API Calls

Delphi	
Using	g Iltem Object Model
1	Coming soon
Usin	g PlanSwift Object Model
1	Coming soon

C#

Using Iltem Object Model		
1 Coming soon		
Using PlanSwift Object Model		

1	g Iltem Object Model Coming soon
Usin	g PlanSwift Object Model
1	

Textures

Textures

Accessing Textures

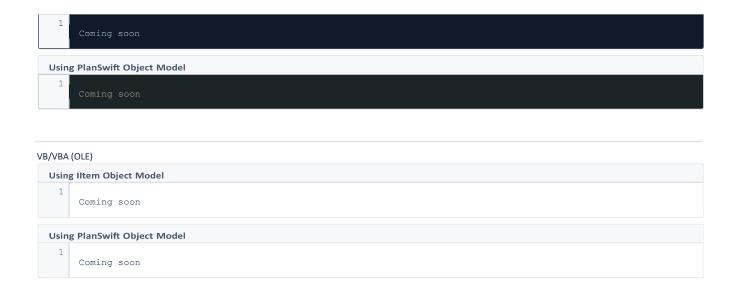
Ho	me Page	Tools	View	E
They	Do	Ph.	×	
			~	
Propertie	es Copy	Paste I	Delete	R
Name				_
	nSwift			
	Settings			
÷ 🕦	Job			
± 🗀	Storages			
	Plugins			
± 🛃	_Units			
÷-10	_Types			
E ()	Types			
E 🗐	Lists			
	Reports			
100	Developers			
	Hatches			
1. E. W. C.	Estimating			_
	Textures			-
Ð	Defau		3	
	🖃 🧰 Brid			
	LTT.	brick005		
		brick011		
		brick013		
	131 01-12	brick015		
		brick021		
	Flo	-		
	13 di 12.22	Carpet		
		wood002		
	LTT.B	wood007		
		wood018		
	E 🔂 Gro	natfl064		
	Contraction of the second	natfl217		
0	Contacts	natii21/		_

Coming soon.

API Calls

Delphi	Jelphi			
Usin	Using Iltem Object Model			
1	Coming soon			
Usin	g PlanSwift Object Model			
1	Coming soon			

C#



Connecting to PlanSwift

Connecting to PlanSwift

Connecting to PlanSwift may be done with OLE and COM. This section describes methods of connecting to PlanSwift and how to hook into active running processes.

PlanSwift does not provide technical support for this function.

Connecting with OL E

Connecting with CO M

Connecting with OLE

Connecting with OLE

Code examples in VB / VBA OLE and Pascal Scripting OLE below show examples of OLE access to the API.

API Calls

VB/VBA (OLE)

Using IItem Object Model	Expand source
Using PlanSwift Object Model	> Expand source

Pascal Scripting (OLE)

Item Object Model	Expand source
Root Object Model	> Expand source

Connecting with COM

Connecting with COM

C# and Delphi code examples below show how API is accessed via COM.

API Calls

Delphi

Using Iltem Object Model	> Expand source
Using meni Object Model	
Using PlanSwift Object Model	Expand source
ŧ	
Using Iltem Object Model	Expand sour
Using Iltem Object Model	Expand source

Developer Documents

Developer Documents

This section describes Page, Section, and Item creation, and adding a new Property.

PlanSwift does not provide technical support for these functions.

Page Creatio n

Section Creatio n

Adding New Propertie s

Item Creatio n

Page Creation

Page Creation

This allows the ability to create Page objects, such as notes, annotations, etc., through the API.

Annotations

Annotations

This allows the creation of annotation objects.

Syntax:

Procedure: Coming soon

API Calls

Delphi

Usin	g Iltem Object Model
1	Coming soon
Usin	g PlanSwift Object Model
1	Coming soon

C#

Using Iltem Object Model		
1	Coming soon	
Using PlanSwift Object Model		
1	Coming soon	

Using Iltem Object Model		
1	Coming soon	
Using PlanSwift Object Model		
1	Coming soon	

Item Object Model		
1	Coming soon	
Root Object Model		

1

Pascal Scripting (OLE) Pascal Scripting

1

Pascal scripting		
Item Object Model		
1	Coming soon	
Using the PlanSwift Object Model		

Coming soon

216

Section Creation

Section Creation

This allows the creation of new sections on a parent item. Sections are children of a parent item. Each time an item is digitized, a new section is created as a child of the item.

Syntax:

Procedure: Coming soon

API Calls

Delphi

Usin	g Iltem Object Model			
1	Coming soon			
Usin	Using PlanSwift Object Model			
1	Coming soon			

C#

Using Iltem Object Model		
1		
Coming soon		
Using PlanSwift Object Model		
1 Coming soon		
coming boon		

VB/VBA (OLE)

Usin	g Iltem Object Model
1	Coming soon
Usin	g PlanSwift Object Model
1	Coming soon

Adding New Properties

Adding New Properties

This allows a new IPropertyObject to be added to an item. Items are composed of properties. Properties are details that describe an item or a manipulation of that item.

Syntax:

Procedure: Coming soon

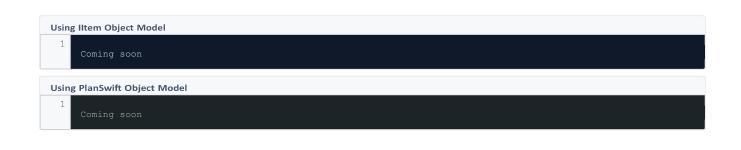
API Calls

Delphi	
Usin	g Iltem Object Model
1	Coming soon
Usin	g PlanSwift Object Model
1	Coming soon

C#

Usin	g Iltem Object Model
1	Coming soon

Usin	g PlanSwift Object Model
1	Coming soon
	Comming Soon



VB/VBA (OLE) Item Creation

Item Creation

The Item Creation options allow the creation of new IItems, which includes Jobs, Estimating items (Parts, Assemblies, or Takeoffs), Types, Reports, and Estimating Layouts.

Jobs Jobs

This creates a job object.

Syntax:

Procedure: Coming soon

API Calls

Delphi

Usin	g Iltem Object Model
1	Coming soon
Usin	g PlanSwift Object Model
1	Coming soon

C#

Using	g Iltem Object Model
1	Coming soon
Usin	g PlanSwift Object Model
Using 1	g PlanSwift Object Model

VB/VBA (OLE)

Usin	g Iltem Object Model				
1	Coming soon				
Usin	Using PlanSwift Object Model				
1	Coming soon				

Estimating Item

Estimating Item

An Estimating item is a part, assembly, or takeoff.

Assembly

Assembly

An assembly is an item that is composed of multiple parts; parts may also have sub-assemblies.

Syntax:

Procedure: Coming soon

API Calls

Delphi

Usin	g Iltem Object Model			
1	Coming soon			
Usin	Using PlanSwift Object Model			
1				
	Coming soon			

C#

Usin	g litem Object Model			
1	Coming soon			
Usin	Using PlanSwift Object Model			
1	Coming soon			

Using Iltem Object Model			
Coming soon			
Using PlanSwift Object Mod	del		
1 Coming soon			
VB/VBA (OLE)			
Part			
Part			
A part is an item with the par	<u>t</u> as the type.		
Syntax:			
Procedure: Coming soon			
API Calls			
Delphi			

Usin	Using IItem Object Model		
1	Coming soon		
Usin	Using PlanSwift Object Model		
1	Coming soon		

Usin	Using IItem Object Model		
1	Coming soon		
Usin	Using PlanSwift Object Model		
1	Coming soon		

C#

Usin	Using Iltem Object Model			
1				
	Coming soon			
Usin	Using PlanSwift Object Model			
1				
	Coming soon			

VB/VBA (OLE)

Takeoff Item

Takeoff Item

A takeoff item is an item of the types: area, linear, segment, or count.

Syntax:

Procedure: Coming soon

API Calls

Delphi			
Usin	g Iltem Object Model		
1	Coming soon		
Usin	g PlanSwift Object Model		
1	Coming soon		

C#

Using Iltem Object Model		
1	Coming soon	
Usin	g PlanSwift Object Model	

Using III	tem Object Model		
1 C	Coming soon		
Using P	lanSwift Object Model		
1 C	Coming soon		

VB/VBA (OLE)

Тур	e
-----	---

Туре

The type is a property of an item.

Syntax:

Procedure: Coming soon

API Calls

Delphi

Usin	Using Iltem Object Model			
1	Coming soon			
Usin	Using PlanSwift Object Model			
1	Coming soon			

Usin	g litem Object Model
1	Coming soon
Usin	g PlanSwift Object Model
1	Coming soon
VB/VBA	(QLE)
	g litem Object Model
1	Coming soon
Usin	g PlanSwift Object Model
1	Coming soon
Rep	orts (1) Reports
A	dds an item of the report type.
Syı	ntax:
	Procedure: Coming soon
٨١	PI Calls
Delphi	

Using Iltem Object Model		
1	Coming soon	
Using PlanSwift Object Model		
Usin	g PlanSwift Object Model	

Usin	Using Iltem Object Model		
1	Coming soon		
Usin	Using PlanSwift Object Model		
1	Coming soon		

C#

Using Iltem Object Model				
1 C	oming soon			
Using Pl	anSwift Object Model			
1 C	oming soon			

VB/VBA (OLE) Estimating Layouts

Estimating Layouts

An estimating layout is a view that allows modification of columns in a view.

Syntax:

Procedure: Coming soon

API Calls

Delphi

Usin	Using Iltem Object Model			
1	Coming soon			
Using PlanSwift Object Model				
1	Coming soon			

C#

	Usin	g Iltem Object Model		
	1	Coming soon		
2				
	Using PlanSwift Object Model			
	1			
		Coming soon		

VB/VBA (OLE)

Using Iltem Object Model			
1	Coming soon		
Using PlanSwift Object Model			

1 Coming soon

API COM Reference

API COM Reference

IPlanSwift

- About
- BeginUpdate
- BeginFormulaUpdate
- CancelTool
- CloseJob
- CompareVersion
- CopyItem
- CurrentVersion
- CurrentViewport
- DeleteItem (1)
- DeleteProperty (1)
- DrawOneWayLayout
- DrawTwoWayLayout
- Edition
- EndFormulaUpdate
- EndUpdate
- GetItem (1)
- GetLine
- GetOneWayLayout (2)
- GetProperty (1)
- GetPropertyFormula (1)
- GetPropertyResult (1)
- GetPropertyResultAsBoolean (1)
- GetPropertyResultAsFloat (1)
- GetPropertyResultAsInteger (1) • GetPropertyResultAsString (1)
- GetRect
- GetTwoWayLayout GetJobTotal
- GetZoom
- Handle
- Iltem
 - CanRecord
 - ChildCount
 - ChildItem
 - Delete
 - Deleteltem (2)
 - DeleteProperty (2)
 - DoRecord
 - Edit
 - FullPath
 - GetItem
 - GetItemByGUID
 - GetPoint
 - GetProperty (2)
 - GetPropertyFormula (2)
 - GetPropertyResult (2)
 - GetPropertyResultAsBoolean (2)
 - GetPropertyResultAsFloat (2)
 - GetPropertyResultAsInteger (2)
 - GetPropertyResultAsString (2)
 - GUID
 - ItemType
 - IPoint • X

 - Y
 - Name (2) NewItem

NewItemEx

NewPoint NewProperty

NewSection ParentItem PointCount PropertyCount PropertyItem SetPoint SetPropertyFormula

IPropertyObject Adjust

- •
- •

•

- •
- •
- •
- •
- •
- •
- •
- •
- •
- •
- •
- •
- •
- •
- •
- •
- •
- •
- •
- •
- •
- •
- •
- •
- .
- •
- •
- •
- •
- •
- •
- •
- •
- •
- •
- •
- •

•

•

CalculateBeforeInherit CompileDenyOLE CompileDenyRead CompileDenyWrite DecimalPlaces EditScript ExecuteScript Expression Formula Group ImageTransparent InheritAction InheritPullFrom InputCondition InputType InputUnits IsInherited IsInput List ListColumnAutoWidth ListFromProperty ListPropertiesToSet ListResultColumn ListReturnFullPath ListShow1Level ListShowOnlyTypes ListShowSearch ListType ListVisibleColumnsInDropdown MeetsInputCondition Name (3) PlugInToExecute PlugInToExecuteButtonCaption PropertyType ResultAsString ResultAsInteger ResultAsFloat ResultAsVariant ScriptType ScriptLanguage ScriptParameters SimpleList SliderMax SliderMin SliderShowTicks SliderTickFrequency SystemHidden SystemLocked TreeList ExecuteScript (2) Units UserHidden UserLocked IsBeta IsJobOpen IsUnlocked NewBlankPage NewChangeGroup NewItem (2)

- NewItemEx (2)
- NewJobEx
- NewPoint (2)
- NewSection (2)
- OnClose
- OnDoneRecordingDigitize
- OnCopyItem
- OnDigitizerSectionChanged
- OnDoneRecording
- OnltemChange
- OnltemDelete
- OnJobClo
- OnJobOpen
- OnNewItem
- OnNewJob
- OnSelectedPageChange
- OnSelectedSelectionChange
- OnSelectionChanged
- OpenJob
- OpenJobEx
- PointCount (2)
- PostChanges
- Root
- SaveScreenShct
- SelectedItem
- SelectedPage
- SelectItemDialg
- SelectionList
- SetPoint (2)
- SetPropertyFormula (2
- SetSelected
- SetZoom
- ISelectionList
 - Count
 - Items (1)
- SetActiveTab (TabName: String
- IsLoaded

IPlanSwift

IPlanSwift

This represents the root object of the Document Object Model.

This interface can be accessed from most development IDEs. It can also be used in the PlanSwift script IDE by using the always available "PlanSwift" object. You should never attempt to create or free the PlanSwift object from script.

While most IDE packages will include the PlanSwift DOM for early binding use, the following shows a late binding example.

API Call

> Expand source
Expand source

C#

Using Iltem Object Model	Expand source
Using PlanSwift Object Model	Expand source

> Expand source

About

About

Shows the About PlanSwift Dialog.

Syntax:

Procedure: IPlanswift.About; Code

Reference:

- 1. Create a New Forms Application
- 2. Add a PlanSwift to the References (Planswift_Tlb)
- 3. Add a button to the form
- 4. Copy code below to the onclick event of the button
- 5. Compile and run

API Calls

Delphi

Using PlanSwift Object Model	> Expand source
C#	
Using PlanSwift Object Model	> Expand source
VB/VBA (OLE)	
Using IItem Object Model	Expand source
Using PlanSwift Object Model	> Expand source

BeginUpdate

BeginUpdate

Signals the beginning of a formula change operation.

Syntax:

Procedure: BeginUpdate; Code

Reference:

- 1. Create a New Forms Application
- 2. Add a PlanSwift to the References (Planswift_Tlb)
- 3. Add a button to the form
- 4. Copy code below to the onclick event of the button
- 5. Compile and run

API Calls

Delphi

Using PlanSwift Object Model

C#

Using Iltem Object Model

VB/VBA (OLE)

Using Iltem Object Model

Using PlanSwift Object Model

BeginFormulaUpdate

BeginFormulaUpdate

Signals the beginning of a formula change operation.

Syntax:

Procedure: BeginFormulaUpdate; Code

Reference:

- 1. Create a New Forms Application
- 2. Add a PlanSwift to the References (Planswift_Tlb)
- 3. Add a button to the form
- 4. Copy code below to the onclick event of the button
- 5. Compile and run

API Calls

Delphi

Using PlanSwift Object Model	> Expand source
------------------------------	-----------------

C#	
Using IItem Object Model	> Expand source
Using PlanSwift Object Model	
1 Coming soon	

VB/VBA (OLE)

Using Iltem Object Model				
1				
	Coming soon			
Using	Using PlanSwift Object Model			
1				
	Coming soon			

CancelTool

CancelTool

Cancels the currently active tool in PlanSwift.

Syntax:

Procedure: CancelTool; Code

Reference:

- 1. Create a New Forms Application
- 2. Add a PlanSwift to the References (Planswift_Tlb)
- 3. Add a button to the form
- 4. Copy code below to the onclick event of the button
- 5. Open PlanSwift and select a digitizer object
- 6. Compile and run

API Calls

Delphi

Using PlanSwift Object Model

C#

Using Iltem Object Model

Using PlanSwift Object Model

 Using Iltem Object Model

 1

 Coming soon

 Using PlanSwift Object Model

 1

 Coming soon

VB/VBA (OLE)

CloseJob

CloseJob

Closes the currently opened job.

Syntax:

Procedure: CloseJob;

Code Reference:

- 1. Create a New Forms Application
- 2. Add a PlanSwift to the References (Planswift_Tlb)
- 3. Add a button to the form
- 4. Copy code below to the onclick event of the button
- 5. Compile and run

API Calls

Delphi

C#

Using PlanSwift Object Model	Expand source

Using Iltem Object Model

> Expand source

Using PlanSwift Object Model

Coming soon

VB/VBA (OLE)

1

Using Iltem Object Model

1 Coming soon

Using PlanSwift Object Model

Coming soon

CompareVersion

CompareVersion

Compares two different versions of PlanSwift.

Syntax:

Procedure: CompareVersion; Code

Reference:

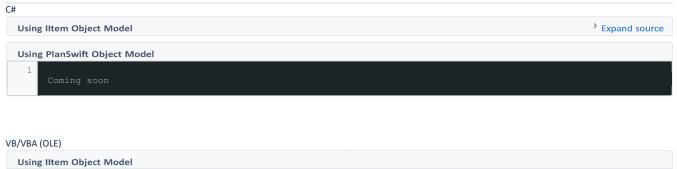
1. Create a New Form application

- 2. Add a button to the form
- 3. Add PlanSwift to reference (Planswift9_tlb in the uses)
- 4. Copy code to button onclick event

API Calls

Delphi

Using PlanSwift Object Model



1	Coming soon			
Usin	Using PlanSwift Object Model			
1	Coming soon			

CopyItem

Copyltem

Creates a copy of Item under Parent and returns the ID of the new item.

If IncludeChildren is true, child items will be copied also.

If SkipSections is true, digitized sections will be duplicated also.

Syntax:

Procedure: CopyItem(Item: String; Parent: String; IncludeChildren: boolean; SkipSections: boolean): String;

Code Reference:

- 1. Create a New Form application
- 2. Add a button to the form
- 3. Add PlanSwift to reference (Planswift9_tlb in the uses)
- 4. Copy code to button onclick event

API Calls

Delphi

Using PlanSwift Object Model

C#

Using Iltem Object Model		
Using PlanSwift Object Model		
1		

VB/VBA (OLE)

Usin	Using Iltem Object Model	
1	Coming soon	
Usin	g PlanSwift Object Model	
1	Coming soon	

CurrentVersion

CurrentVersion

Returns the current version of the active PlanSwift application.

Syntax:

Procedure: CurrentVersion; Code

Reference:

- 1. Create a New Form application
- 2. Add a button to the form
- 3. Add PlanSwift to reference (Planswift9_tlb in the uses)
- 4. Copy code to button onclick event

API Calls

Delphi

Using PlanSwift Object Model

1	
2	//PlanSwift code:
3	var
4	sVersion: string;
5	begin
6	<pre>sVersion := PlanSwift.CurrentVersion;</pre>
7	end;
8	
9	//FreshDesk code:
10	<pre>procedure TForm1.psCurrentVersion(sender: TObject);</pre>
11	var
12	<pre>ps: IPlanSwift;</pre>
13	begin
14	//Create Planswift Interface
15	<pre>ps := coPlanswift.Create;</pre>
16	//Show Current Version of Planswift
17	ShowMessage(ps.CurrentVersion);
18	//Free the Planswift Interface
19	ps := nil ;
	end;

C#



VB/VBA (OLE)

Using Iltem Object Model		
1	Coming soon	
Using	g PlanSwift Object Model	

CurrentViewport

CurrentViewport

Gets the Upper right and lower left points of the viewport.

Syntax:

Procedure: CurrentViewport; Code

Reference:

- 1. Create a New Form application
- 2. Add a button to the form
- 3. Add PlanSwift to reference (Planswift9_tlb in the uses)
- 4. Copy code to button onclick event

API Calls

Delphi

Using PlanSwift Object Model	Expand source

C# Using Iltem Object Model Using PlanSwift Object Model 1 Coming soon

VB/VBA (OLE)

Usin	Using Iltem Object Model		
1	1		
	Coming soon		
Usin	Using PlanSwift Object Model		
1			
	Coming soon		

DeleteItem (1)

DeleteItem

Deletes the item specified by *ItemPath* from the system.

Syntax:

Procedure: DeleteItem(ItemPath: String): Boolean;

Code Reference:

1. Navigate to Plugin Store->Tool Manager and create a new Plugin

- 2. Set the plugin type to Script Code and open the Editor
- 3. Copy Code into the editor

4. Press run

API Calls

Delphi

Using PlanSwift Object Model	> Expand source

C#

Using IItem Object Model	> Expand source
Using PlanSwift Object Model	
1 Coming soon	

VB/VBA (OLE)

Usin	Using Iltem Object Model		
1	Coming soon		
Usin	Using PlanSwift Object Model		
1	1 Coming soon		

DeleteProperty (1)

DeleteProperty

Deletes PropertyName from ItemPath.

Syntax:

Procedure: DeleteProperty(ItemPath, PropertyName: String): Boolean; Code

Reference:

- 1. Navigate to Plugin Store->Tool Manager and create a new Plugin
- 2. Set the plugin type to Script Code and open the Editor
- 3. Copy Code into the editor
- 4. Press run

API Calls

Delphi

Using PlanSwift Object Model	> Expand source

C#			
Using Iltem Object Model	> Expand source		
Using PlanSwift Object Model			
1 Coming soon			

VB/VBA (OLE)

Usin	Using Iltem Object Model		
1			
	Coming soon		
Usin	Using PlanSwift Object Model		
1			
	Coming soon		

DrawOneWayLayout

DrawOneWayLayout

Function used to perform segment layouts at a specified span, horizontal run, as well as spacing.

Syntax:

Procedure: DrawOneWayLayout(const Altem: WideString; const SpanLine: ILine; const RunLine: ILine; blncludeFirst:

Arguments:

Altem: WideString Specifies the area section to assign the layout segments to.

SpanLine: ILine Direction span start and endpoint.

RunLine: ILine

Horizontal (side to side) run direction of area to populate. Requires a start and endpoint;

bIncludeFirst: WordBool

Specifies whether to include a segment at the "start" run point. Even if it does not fall within the spacing range.

bincludeLast: WordBool

Specifies whether to include a segment at the "last" run point. Even if it does not fall within the spacing range.

nSpacing: Double

Specifies the "run" spacing used when laying out segment objects.

AArea: WideString (optional parameter)

Specifies a defined "Area Segment" to trim/extend laid segments to. Supply either the path or GUID (globally unique identifier) to the area section. Or, empty double-quotes for no trim/extending required.

Code Reference:

- 1. Create a New Form application
- 2. Add a button to the form
- 3. Add PlanSwift to reference (Planswift9_tlb in the uses)
- 4. Copy code to button onclick event

API Calls

Delphi

Using PlanSwift Object Model

C#	C#		
Using IItem Object Model			
Using PlanSwift Object Model			
	Coming soon		

VB/VBA (OLE)

Coming soon

Using PlanSwift Object Model

Coming soon

DrawTwoWayLayout

DrawTwoWayLayout

Function used to perform segment layouts (in 2 directions) at a specified span, horizontal run, as well as spacing.

Syntax:

Procedure: DrawTwoWayLayout(const Altem: WideString; const SpanLine: ILine; const RunLine: ILine; blncludeFirst:

Arguments:

Altem: WideString Specifies the area section to assign the layout segments to.

SpanLine: ILine Direction span start and endpoint.

RunLine: ILine

Horizontal (side to side) run direction of area to populate. Requires a start and endpoint;

bIncludeFirst: WordBool

Specifies whether to include a segment at the "start" run point. Even if it does not fall within the spacing range.

bincludeLast: WordBool

Specifies whether to include a segment at the "last" run point. Even if it does not fall within the spacing range.

nSpacing: Double

Specifies the "run" spacing used when laying out segment objects.

AArea: WideString (optional parameter)

Specifies a defined "Area Segment" to trim/extend laid segments to. Supply either the path or GUID (globally unique identifier) to the area section. Or, empty double-quotes for no trim/extending required.

Code Reference:

- 1. Create a New Form application
- 2. Add a button to the form
- 3. Add PlanSwift to reference (Planswift9_tlb in the uses)
- 4. Copy code to button onclick event

API Calls

Delphi

Using PlanSwift Object Model

Expand source

C#

2 puk	olic class PlanswiftApi
3 {	
1	private PlanSwift Planswift { get; }
5	public PlanSwiftApi()
5	
7	Planswift = new PlanSwift();
3	
}	

Coming soon

Using Iltem Object Model		
1	Coming soon	
Usin	Using PlanSwift Object Model	
1	Coming soon	

Edition

Edition

Returns the current PlanSwift Edition.

Syntax:

Procedure: Edition;

Code Reference:

1. Navigate to Plugin Store->Tool Manager and create a new Plugin

2. Set the plugin type to Script Code and open the Editor

3. Copy Code into the editor

4. Press run

API Calls

Delphi

Using PlanSwift Object Model

C#

1

Using Iltem Object Model

Using PlanSwift Object Model

Coming so



EndFormulaUpdate

EndFormulaUpdate

Signals an end to the formula update operation.

Syntax:

Procedure: EndFormulaUpdate; Code

Reference:

- 1. Create a New Forms Application
- 2. Add a PlanSwift to the References (Planswift_Tlb)
- 3. Add a button to the form
- 4. Copy code below to the onclick event of the button
- 5. Compile and run

Usin	Using Iltem Object Model		
1	Coming soon		
Usin	Using PlanSwift Object Model		
1	Coming soon		

API Calls

Delphi

Using PlanSwift Object Model

C#

Using Iltem Object Model

Using PlanSwift Object Model

1 Coming s

VB/VBA (OLE) EndUpdate

EndUpdate

Signals the end of update operations.

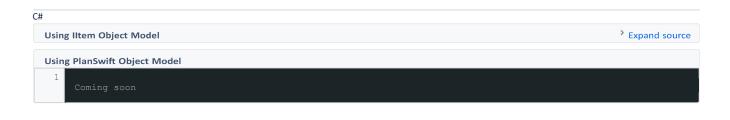
Syntax: Procedure: EndUpdate; Code Reference: 1. Create a New Project

2. Add PlanSwift Reference Usage

API Calls

Delphi

Using PlanSwift Object Model



Usin	Using Iltem Object Model		
1			
	Coming soon		
Usin	Using PlanSwift Object Model		
1			
	Coming soon		

GetItem (1)

GetItem

Returns the item given by FullPath. Returns Nil if the object is not found.

Syntax:

Procedure: GetItem(FullPath: String): IItem;

Code Reference:

- 1. Navigate to Plugin Store->Tool Manager and create a new Plugin
- 2. Set the plugin type to Script Code and open the Editor
- 3. Copy Code into the editor
- 4. Press run

API Calls

Delphi

Using PlanSwift Object Model

C#

Using Iltem Object Model

Using PlanSwift Object Model

1 Coming s

VB/VBA (OLE)

Usin	g Iltem Object Model
1	Coming soon
Usin	g PlanSwift Object Model
1	Coming soon

GetLine

GetLine

Prompts the user to click 2 points on the active plan to define a line then returns the coordinates in p1 and p2.

Returns 1 if the function is successful or 0 if the user cancels.

Syntax:

Procedure: GetLine(const ToolHint: WideString): ILine;

Code Reference:

- 1. Create a New Form application
- 2. Add a button to the form
- 3. Add PlanSwift to reference (Planswift9_tlb in the uses)
- 4. Copy code to button onclick event

API Calls

Delphi

Using PlanSwift Object Model	Expand source
------------------------------	---------------

C#	
Using IItem Object Model	> Expand source
Using PlanSwift Object Model	
1 Coming soon	

VB/VBA (OLE)

Using	Using Iltem Object Model		
1	Coming soon		
Using	Using PlanSwift Object Model		
1	Coming soon		

GetOneWayLayout (2)

GetOneWayLayout

Function used to perform segment layouts at a specified span, horizontal run, as well as spacing.

1

Procedure: GetOneWayLayout(const Altem: WideString; const sSpanHint: WideString; const sRunHint: WideString; blncludeFirst: WordBool; blncludeLast: WordBool; nSpacing: Double; const AArea: WideString): WordBool;

Arguments:

Altem: WideString

Specifies the area section to assign the layout segments to.

sSpanHint: WideString

Hint to user on mouse cursor specifying to select the span line.

sRunHint: WideString Hint to user on mouse cursor specifying to select the run line.

bIncludeFirst: WordBool

Specifies whether to include a segment at the "start" run point. Even if it does not fall within the spacing range.

bincludeLast: WordBool

Specifies whether to include a segment at the "last" run point. Even if it does not fall within the spacing range.

nSpacing: Double

Specifies the "run" spacing used when laying out segment objects.

AArea: WideString (optional parameter)

Specifies a defined "Area Segment" to trim/extend laid segments to. Supply either the path or GUID to the area section. Or, empty double-quotes for no trim/extending required.

Code Reference:

- 1. Create a New Form application
- 2. Add a button to the form

3. Add PlanSwift to reference (Planswift9_tlb in the uses)

4. Copy code to button onclick event

API Calls

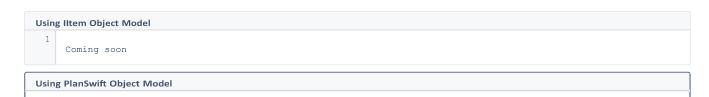
Delphi

Using PlanSwift Object Model

Expand source

C#	
Using Iltem Object Model	> Expand source
Using PlanSwift Object Model	
1 Coming soon	

VB/VBA (OLE)



GetProperty (1)

GetProperty

Returns the IPropertyObjectspecified by *ItemPath* and *PropertyName*. Returns *Nil* if the Item or Property is not found.

Syntax:

Procedure: GetProperty(ItemPath, PropertyName: String): IPropertyObject; Code

Reference:

- 1. Navigate to Plugin Store->Tool Manager and create a new Plugin
- 2. Set the plugin type to Script Code and open the Editor
- 3. Copy Code into the editor
- 4. Press run

API Calls

Delphi

Using PlanSwift Object Model

> Expand source

#		
Using Iltem Object Model	> Expand source	
Using PlanSwift Object Model		
Coming soon		

VB/VBA (OLE)

Using	Using Iltem Object Model		
1	Coming soon		
Using	Using PlanSwift Object Model		
1	Coming soon		

GetPropertyFormula (1)

GetPropertyFormula

Returns the formula string for the property specified by *ItemPath* and *PropertyName*. Returns an empty string (") if the item or property is not found.

Procedure: GetPropertyFormula(ItemPath, PropertyName: String): String; Code

Reference:

- 1. Navigate to Plugin Store->Tool Manager and create a new Plugin
- 2. Set the plugin type to Script Code and open the Editor
- 3. Copy Code into the editor
- 4. Press run

API Calls

Delphi

Using PlanSwift Object Model

1	
2	// PlanSwift code:
3	<pre>Result := PlanSwift.GetPropertyFormula(ItemPath, 'Name');</pre>
4	
5	//FreshDesk code:
6	<pre>procedure psgetPropertyFormula;</pre>
7	var
8	ps: IPlanSwift;
9	itm: IItem;
10	<pre>propvalue: WideString;</pre>
11	begin
12	//Create the Planswift Interfacev
13	<pre>ps := coplanswift.Create;</pre>
14	//Get the selected Item
15	<pre>itm := ps.SelectedItem;</pre>
16	//Set the property value
17	<pre>propvalue := ps.GetPropertyFormula(itm.GUID, 'Name');</pre>
18	//Chece if Property value is empty
19	<pre>if propvalue <> '' then</pre>
20	ShowMessage(propvalue);
21	//Free Planswift Interface
22	ps := nil;
	end;

C#

Using Iltem (Using Iltem Object Model	
1 nubli	ic class PlanswiftApi	
2 publi 3 {	ic class rianswitchpi	
	private PlanSwift Planswift { get; }	
5 1	public PlanSwiftApi()	
6		
7	Planswift = new PlanSwift();	
8)		
}		

Usin	g PlanSwift Object Model
1	Coming soon
VB/VBA	(OLE)
Using	; Iltem Object Model
1	Coming soon
Usin	g PlanSwift Object Model
1	Coming soon

GetPropertyResult (1)

GetPropertyResult

Returns the calculated result from the given property.

Syntax:

Procedure: GetPropertyResult(ItemPath, PropertyName: String): Variant; Code

Reference:

- 1. Navigate to Plugin Store->Tool Manager and create a new Plugin
- 2. Set the plugin type to Script Code and open the Editor
- 3. Copy Code into the editor
- 4. Press run

API Call

Delphi

Using PlanSwift Object Model	Expand source
------------------------------	---------------

C# Using Iltem Object Model Using PlanSwift Object Model 1

VB/VBA	(OLE)
Usin	g Iltem Object Model
1	Coming soon
Usin	g PlanSwift Object Model
1	Coming soon

GetPropertyResultAsBoolean (1)

GetPropertyResultAsBoolean

Attempt to return the result of the given property as a boolean value. If the calculated result cannot be converted to a boolean value, the default value is returned.

Procedure: GetPropertyResultAsBoolean(ItemPath, PropertyName: String; Default: Boolean = False): Boolean;

Code Reference:

- 1. Navigate to Plugin Store->Tool Manager and create a new Plugin
- 2. Set the plugin type to Script Code and open the Editor
- 3. Copy Code into the editor
- 4. Press run

API Calls

Delphi

Using PlanSwift Object Model

Expand source

C#



Using PlanSwift Object Model

Coming soor

VB/VBA (OLE)

Using	g Iltem Object Model
1	Coming soon
Using	g PlanSwift Object Model
1	Coming soon

GetPropertyResultAsFloat (1)

GetPropertyResultAsFloat

Attempts to return the given property value as a floating point value. If the calculated property value cannot be converted, the value supplied by Default is returned.

Procedure: GetPropertyResultAsFloat(ItemPath, PropertyName: String; Default: Double = 0): Double;

Code Reference:

1. Navigate to Plugin Store->Tool Manager and create a new Plugin

2. Set the plugin type to Script Code and open the Editor

3. Copy Code into the editor

4. Press run

API Calls

Delphi	
Using PlanSwift Object Model	> Expand source

C# Using Iltem Object Model Using PlanSwift Object Model Coming soon

VB/VBA (OLE)

Usin	g Iltem Object Model
1	
	Coming soon
Usin	g PlanSwift Object Model
1	
	Coming soon
	g PlanSwift Object Model Coming soon

GetPropertyResultAsInteger (1)

GetPropertyResultAsInteger

Attempts to return the property value as an Integer. If the calculated value cannot be converted to an integer, the value given in Default is returned.

Syntax:

Procedure: GetPropertyResultAsInteger(ItemPath, PropertyName: String; Default: Integer = 0): Integer;

Code Reference:

- 1. Navigate to Plugin Store->Tool Manager and create a new Plugin
- 2. Set the plugin type to Script Code and open the Editor
- 3. Copy Code into the editor
- 4. Press run

API Calls

Using Iltem Object Model

Delphi

C#

`	_ ·	
1	Expand	source

Using PlanSwift Object Model		
1 Coming soon		

VB/VBA (OLE)

Usin	g Iltem Object Model
1	
	Coming soon
Usin	g PlanSwift Object Model
1	
1	
	g PlanSwift Object Model

GetPropertyResultAsString (1)

GetPropertyResultAsString

Returns the result value of the given property. Returns Default if the property is not found.

Syntax:

Procedure: GetPropertyResultAsString(ItemPath, PropertyName: String; Default String = ''): String;

Code Reference:

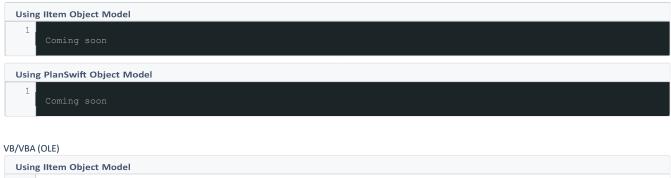
- 1. Create a New Form application
- 2. Add a button to the form
- 3. Add PlanSwift to reference (Planswift9_tlb in the uses)
- 4. Copy code to button onclick event

API Calls

Delphi

Using PlanSwift Object Model

> Expand source



0.5111	g litem Object Model
1	
+	Coming soon
	contrig soon
Usin	g PlanSwift Object Model
Usin	g PlanSwift Object Model
Usin 1	g PlanSwift Object Model
Usin 1	g PlanSwift Object Model

GetRect

GetRect

Prompts the user to click 2 points on the active plan to define a rectangle and returns the coordinates in *p1* and *p2*.

Returns 1 if the function is successful or 0 if the user cancels.

Syntax:

Procedure: GetRect(Var p1x: double; Var p1y: double; Var p2x: double; Var p2y: double; Hint: String): Integer;

Code Reference:

- 1. Create a New Form application
- 2. Add a button to the form
- 3. Add PlanSwift to reference (Planswift9_tlb in the uses)
- 4. Copy code to button onclick event

API Calls

Delphi

Using PlanSwift Object Model	> Expand source

C#

Using Iltem Object Model

> Expand source

Using PlanSwift Object Model

Coming soo

Usin	Using Iltem Object Model			
1	Coming soon			
Usin	Using PlanSwift Object Model			
1				
	Coming soon			

GetTwoWayLayout

GetTwoWayLayout

Function used to perform segment layouts (in 2 directions) at a specified span, horizontal run, as well as spacing.

Syntax:

Procedure: GetTwoWayLayout(const Altem: WideString; const sSpanHint: WideString; const sRunHint: WideString; blncludeFirst: WordBool; blncludeLast: WordBool; nSpacing: Double; const AArea: WideString): WordBool;

Arguments:

Altem: WideString Specifies the area section to assign the layout segments to.

sSpanHint: WideString

Hint to user on mouse cursor specifying to select the span line.

sRunHint: WideString

Hint to user on mouse cursor specifying to select the run line.

bIncludeFirst: WordBool

Delphi

Using PlanSwift Object Model	> Expand source
C#	

Using PlanSwift Object Model > Expand source

VB/VBA (OLE)

Usin	Using IItem Object Model		
1	Coming soon		
Usin	Using PlanSwift Object Model		
1			
	Coming soon		

Specifies whether to include a segment at the "start" run point. Even if it does not fall within the spacing range.

bincludeLast: WordBool

Specifies whether to include a segment at the "last" run point. Even if it does not fall within the spacing range.

nSpacing: Double

Specifies the "run" spacing used when laying out segment objects.

AArea: WideString (optional parameter)

Specifies a defined "Area Segment" to trim/extend laid segments to. Supply either the path or GUID to the area section. Or, empty double-quotes for no trim/extending required.

Code Reference:

- 1. Create a New Form application
- 2. Add a button to the form
- 3. Add PlanSwift to reference (Planswift9_tlb in the uses)
- 4. Copy code to button onclick event

API Calls

GetJobTotal

GetJobTotal

Retrieves the total number of items of a certain type in the entire opened job.

Syntax:

Procedure: GetJobTotal(const Propertyname: WideString; const ItemType: WideString = "): Double;

Code Reference:

- 1. Create a New Form application
- 2. Add a button to the form
- 3. Add PlanSwift to reference (Planswift9_tlb in the uses)
- 4. Copy code to button onclick event

API Call

Delphi	
Using PlanSwift Object Model	> Expand source

C#	
Using Iltem Object Model	Expand source
Using PlanSwift Object Model	
1 Coming soon	

VB/VBA (OLE)

Usin	Using Iltem Object Model		
1	Coming soon		
	-		
Usin	g PlanSwift Object Model		
Usin, 1	g PlanSwift Object Model		
1	g PlanSwift Object Model Coming soon		

GetZoom

GetZoom

Returns the current "zoom" scale factor for the active page.

Syntax:

Procedure: Get_Zoom: Double; Code

Reference:

1. Create a New Form application

2. Add a button to the form

3. Add PlanSwift to reference (Planswift9_tlb in the uses)

4. Copy code to button onclick event

API Call

Delphi Using PlanSwift Object Model Expand source

C#
Using Iltem Object Model
Using PlanSwift Object Model
Coming soon

VB/VBA (OLE)

Usin	Using IItem Object Model		
1	Coming soon		
Usin	Using PlanSwift Object Model		
1	Coming soon		

Handle

Handle

Gets the handle of the current PlanSwift application.

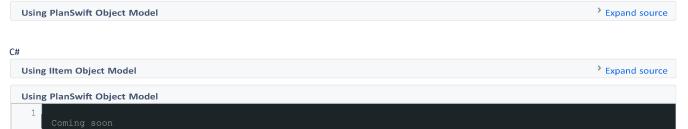
Procedure: Handle: HResult; Code

Reference:

- 1. Create a New Form application
- 2. Add a button to the form
- 3. Add PlanSwift to reference (Planswift9_tlb in the uses)
- 4. Copy code to button onclick event

API Calls

Delphi



VB/VBA (OLE)

Usin	Using Iltem Object Model		
1	Coming soon		
Usin	Using PlanSwift Object Model		
1	Coming soon		

lltem

lltem

This is the interface object for a PlanSwift Item.

In PlanSwift script, since each script is the property of an item, you can use the default *Item* and *Property* objects to access the IItem and IPropertyObject that the script belongs to.

Syntax:

Procedure: Coming soon

API Calls

Delphi

Using IItem Object Model	> Expand source
Using PlanSwift Object Model	
1 Coming soon	

C#

Using Iltem Object Model	Expand source
Using PlanSwift Object Model	
1 Coming soon	

	Using Iltem Object Model			
Coming soon				
Using PlanSwift Object Model				
1 Coming soon				

CanRecord

CanRecord

Returns true if the item is a recordable item.

Syntax:

Procedure: CanRecord: Boolean;

API Calls

Delphi

Using PlanSwift Object Model	> Expand source

C#

Using IItem Object Model

Expand source

Using PlanSwift Object Model

1 Coming soon

VB/VBA	. (OLE)
Usin	g Iltem Object Model
1	Coming soon
Usin	g PlanSwift Object Model
1	Coming soon

ChildCount

ChildCount

Returns the number of child items for the item.

Syntax:

Procedure: ChildCount: Integer;

API Calls

Delphi

Using PlanSwift Object Model

C#

1

Using Iltem Object Model

Using PlanSwift Object Model

Coming soon

VB/VBA	(OLE)
Usin	g Iltem Object Model
1	Coming soon
Usin	g PlanSwift Object Model
1	Coming soon
0.1	10.

ChildItem

ChildItem

Returns the child item at the given index position.

Syntax:

Procedure: ChildItem(Index: Integer): IItem;

API Calls

Delphi

Using PlanSwift Object Model

> Expand source

C#	
Using Iltem Object Model	
Using PlanSwift Object Model	
1 Coming soon	

Using	g Iltem Object Model
1	Coming soon
Lising	g PlanSwift Object Model
Usinį	
1	Coming soon
Dele	ete

Delete

Deletes the Item and its children from the system.

Syntax:

Procedure: Delete;

API Calls	
Delphi	
Using PlanSwift Object Model	> Expand source
C#	
Using IItem Object Model	> Expand source
Using DispSwift Object Madel	
Using PlanSwift Object Model	
1 Coming soon	

Usin	Using Iltem Object Model	
1	Coming soon	
Usin	g PlanSwift Object Model	
1	Coming soon	

Deleteltem (2)

DeleteItem

Deletes the given item if it exists.

Procedure: PlanSwift.DeleteItem(const ItemPath: WideString): WordBool;

API Calls

Delphi	
Using PlanSwift Object Model	
C#	
Using Iltem Object Model	
Using PlanSwift Object Model	
1 Coming soon	
VB/VBA (OLE)	
Using Iltem Object Model	
Using PlanSwift Object Model	
1	

Coming soon

DeleteProperty (2)

DeleteProperty

Deletes the given property if it exists.

Procedure: DeleteProperty(PropertyName: String);

API Calls

Delphi

Using PlanSwift Object Model	> Expand source

C#

Using Iltem Object Model
Cxpand source

Using PlanSwift Object Model

Coming soon

VB/VBA (OLE)

Usin	Using Iltem Object Model	
1	Coming soon	
Usin	g PlanSwift Object Model	
1	Coming soon	

DoRecord

DoRecord

Begins recording digitizer points for the Item. Returns False if no points are recorded.

Syntax:

Procedure: DoRecord: Boolean;

API Calls

Delphi	
Using PlanSwift Object Model	
C#	
Using Iltem Object Model	

Using PlanSwift Object Model

Coming soon

Usin	Using Iltem Object Model	
1	Coming soon	
Usin	Using PlanSwift Object Model	
1		

1

Edit

Edit

Displays the Item in the Editor Dialog.

Syntax:

Procedure: Edit(ShowAdvanced: Boolean = True): Boolean;

API Calls

Delphi

Using PlanSwift Object Model

> Expand source

Using Iltem Object Model			
1	Coming soon		
Using PlanSwift Object Model			
1	Coming soon		

C	2#
	Using Iltem Object Model
	Using PlanSwift Object Model
	1 Coming soon

FullPath

FullPath

Returns the full path to the Item.

Syntax:

Procedure: FullPath: String;

API Calls

Delphi

Using PlanSwift Object Model

C#

Using Iltem Object Model

Using PlanSwift Object Model

Coming soon

VB/VBA (OLE)

Using Iltem Object Model
Coming soon
Using PlanSwift Object Model
Coming soon

GetItem

GetItem

Returns the given child item of the item.

```
Syntax:
```

Procedure: GetItem(ItemPath: String): IItem;

Delphi

Using PlanSwift Object Model

C#

Using IItem Object Model

Using PlanSwift O	bject Model
-------------------	-------------

Using PlanSwift Object Model					
1 Coming					

VB/VBA (OLE)

Using IItem Object Model		
1	Coming soon	
Using PlanSwift Object Model		
00111		
1	Coming soon	
GetItemByGUID		

GetItemByGUID

Returns the child item specified by aGUID.

Syntax:

Procedure: GetItemByGUID(aGUID: String): IItem;

API Calls

Delphi

Using PlanSwift Object Model	Expand source

C#

Using Iltem Object Model

Expand source

Usin	Using PlanSwift Object Model		
1			
	Coming soon		

VB/VBA (OLE)

Usin	g Iltem Object Model
1	Coming soon
Usin	g PlanSwift Object Model
1	Coming soon

GetPoint

GetPointD

Returns the IPoint object from the given index position.

Syntax:

Procedure: GetPoint(PointIndex: Integer): IPoint

Delphi

Using PlanSwift Object Model

C#

Using Iltem Object Model

Using PlanSwift Object Model



VB/VBA (OLE)

1

Usin	g Iltem Object Model
1	Coming soon
Usin	ng PlanSwift Object Model
1	Coming soon

GetProperty (2)

GetProperty

Returns the given IPropertyObjector Nil if the property does not exist.

Syntax:

Procedure: GetProperty(PropertyName: String): IPropertyObject;

Delphi

Using PlanSwift Object Model

Expand source

C#

Using Iltem Object Model

Using PlanSwift Object Model

Coming soon

VB/VBA (OLE)

Usin	g Iltem Object Model
1	Coming soon
Usin	g PlanSwift Object Model
1	2
	Coming soon

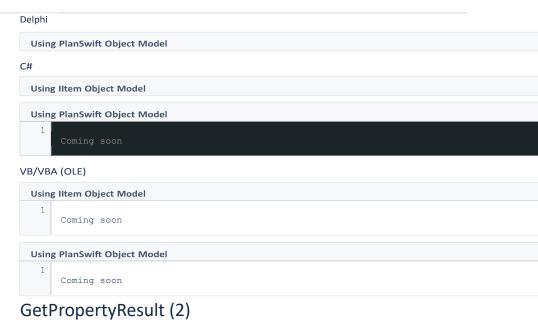
GetPropertyFormula (2)

GetPropertyFormula

Returns the formula from the given property.

Syntax:

Procedure: GetPropertyFormula(PropertyName: String): String;



GetPropertyResult

Returns the property result as a variant;

Syntax:

Procedure: GetPropertyResult(PropertyName: String): Variant;

Delphi

Using PlanSwift Object Model > Expand source

Expand source

C#

Using Iltem Object Model

Using PlanSwift Object Model

Coming soon

VB/VBA (OLE)

Usin	g Iltem Object Model
1	Coming soon
Usin	g PlanSwift Object Model
1	Coming soon

GetPropertyResultAsBoolean (2)

GetPropertyResultAsBoolean

Returns the given property result as a boolean value.

Syntax:

Procedure: GetPropertyResultAsBoolean(PropertyName: String; Default: Boolean = False): Boolean;

Delphi

Using PlanSwift Object Model

C#

Using Iltem Object Model
Using PlanSwift Object Model
Coming soon
VB/VBA (OLE)
Using Iltem Object Model

Usin	Using IItem Object Model		
1	Coming soon		
Using PlanSwift Object Model			
1	Coming soon		

GetPropertyResultAsFloat (2)

GetPropertyResultAsFloat

Returns the given property result as type double.

Syntax:

Procedure: GetPropertyResultAsFloat(PropertyName: String; Default: Double = 0): Double;

Delphi

Using PlanSwift Object Model	> Expand sourc
Using IItem Object Model	Expand source
Using PlanSwift Object Model	
1 Coming soon	

VB/VBA (OLE)

Using	g IItem Object Model
1	Coming soon
Using	g PlanSwift Object Model
1	Coming soon

GetPropertyResultAsInteger (2)

GetPropertyResultAsInteger

Returns the given property result value as an integer.

Syntax:

Procedure: GetPropertyResultAsInteger(PropertyName: String; Default: Integer = 0): Integer;

Delphi

Using PlanSwift Object Model

C#

Using Iltem Object Model

Using PlanSwift Object Model

VB/VBA (OLE)

1

	Usin	g Iltem Object Model
	1	Coming soon
	Usin	g PlanSwift Object Model
	1	Coming soon

GetPropertyResultAsString (2)

GetPropertyResultAsString

Returns the given property result value as a string.

Syntax:

Procedure: GetPropertyResultAsString(PropertyName: String; Default: String = ''): String;

Delphi

Using PlanSwift Object Model

Expand source

C#

Using Iltem Object Model > Expand source

Coming soon

VB/VBA (OLE)

Usin	g Iltem Object Model	
1	Coming soon	
Using PlanSwift Object Model		
1	Coming soon	

GUID

GUID

Returns the GUID (globally unique identifier) for the Item.

Syntax:

Procedure: GUID: String;

Delphi

Using PlanSwift Object Model

C#

Using Iltem Object Model

Using PlanSwift Object Model

1 Coming soon

VB/VBA (OLE)

	Usin	g Iltem Object Model
	1	Coming soon
	Usin	g PlanSwift Object Model
	1	Coming soon

ItemType

ItemType

Gets or Sets the Type property for the Item.

Syntax:

Procedure: ItemType: String;

Delphi

Using Iltem Object Model * Expand s Using PlanSwift Object Model C# Using Iltem Object Model * Expand s	Jource
C#	
Coming soon	
Lising litem Object Model	
Using item Object Model	source
Using PlanSwift Object Model	
1 Coming soon	
VB/VBA (OLE)	
IPoint	
IPoint	
The IPoint Interface	
Syntax:	

Usin	Using IItem Object Model	
1	Coming soon	
Usin	Using PlanSwift Object Model	
1	Coming soon	

Procedure: Coming soon.

API Calls

Delphi

Using Iltem Object Model
Using PlanSwift Object Model
1 Coming soon
C#
Using Iltem Object Model
Using PlanSwift Object Model
1 Coming soon
VB/VBA (OLE)
X
x
Gets or Sets the X coordinate for the IPoint.
Syntax:

Usin	Using Iltem Object Model		
1	Coming soon		
Usin	Using PlanSwift Object Model		
1	Coming soon		

Procedure: X: Double;

API Calls

Delphi

Using PlanSwift Object Model	Expand source
------------------------------	---------------

C#	
Using IItem Object Model	> Expand source
Using PlanSwift Object Model	

VB/VBA (OLE)

Usin	Using Iltem Object Model	
1		
	Coming soon	
Usin	Using PlanSwift Object Model	
1		
	Coming soon	
V		

Y

Υ

Gets or Sets the Y coordinate for the IPoint.

Syntax:

Procedure: Y: Double;

Delphi

Using PlanSwift Object Model

C#

 Using Iltem Object Model

 Using PlanSwift Object Model

 1
 Coming soon

 VB/VBA (OLE)

 Using Iltem Object Model

 1
 Coming soon

 Using PlanSwift Object Model

 1
 Coming soon

 Using PlanSwift Object Model

 1
 Coming soon

Name (2)

Name

Gets or Sets the Name property for the item.

Syntax:

Procedure: Name: String;

Delphi



Using IItem Object Model	Expand source
Using PlanSwift Object Model	
Coming soon	

VB/VBA (OLE)

NewItem

NewItem

Creates a new child item and returns the new item.

Syntax:

Usin	Using Iltem Object Model	
1	Coming soon	
Using PlanSwift Object Model		
1	Coming soon	

Procedure: NewItem(ItemType: String; AName: String = "): IItem;

API Calls

Delphi

Using PlanSwift Object Model
C#
Using Iltem Object Model
Using PlanSwift Object Model
1 Coming soon
VB/VBA (OLE)
Using Iltem Object Model
1 Coming soon
Using PlanSwift Object Model
1 Coming soon

NewItemEx

NewItemEx

Creates a new child item and returns the new item. If EditProperties is true then the property editor will be displayed when the item is created.

Syntax:

Procedure: NewItemEx(ItemType, AName: String; EditProperties: Boolean): IItem;

API Calls

Delphi

Using PlanSwift Object Model

Expand source

C#

Using IItem Object Model	> Expand source
Using PlanSwift Object Model	
1	

VB/VBA (OLE)

Usin	Using Iltem Object Model		
1	Coming soon		
Usin	Using PlanSwift Object Model		
1			
	Coming soon		

NewPoint

NewPoint

Creates a new digitizer point at the X, Y coordinates.

Syntax:

Procedure: NewPoint(X, Y: Double);

API Calls

Delphi

Using PlanSwift Object Model

Expand source

C#

Using Iltem Object Model	> Expand source
Using PlanSwift Object Model	
1 Coming soon	

VB/VBA (OLE)

Usin	Using Iltem Object Model		
1	Coming soon		
Usin	Using PlanSwift Object Model		
1	Coming soon		

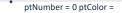
NewProperty

NewProperty

Creates a new property as specified and returns the new IPropertyObject.

Syntax:

Procedure: NewProperty(PropertyName: String; AFormula: String = "; PropertyType: PropertyTypes = ptNumber): IPropertyObject;



- 1 ptText = 2 ptMemo = 3
- ptCheckBox = 4 ptPath =
- 5 ptImage = 6
- ptLargeImage = 7
- ptType = 8 ptScript = 9
- ptFile = 10 ptLargeFile =
- 11 ptFileName = 12
- ptConnectionString = 13
- ptSlider = 14
- ptDimension = 15
- •
- •

•

API Calls

Delphi

Using Iltem Object Model

Usin	Using PlanSwift Object Model		
1	Coming soon		

C#

Cri	
Using Iltem Object Model	> Expand source
Using PlanSwift Object Model	
1	
Coming soon	

VB/VBA (OLE)

Using Iltem Object Model	
1	Coming soon
Usin	g PlanSwift Object Model
1	Coming soon

NewSection

NewSection

Createsa new section for the Item. If the Item is not a draw object, this function returns *Nil*.

Syntax:

Procedure: NewSection(AName: String = "): IItem;

API Calls

Delphi

Using PlanSwift Object Model

C#

Using Iltem Object Model

Using PlanSwift Object Model

1

VB/VBA	(OLE)
Using	g Iltem Object Model
1	Coming soon
Usin	g PlanSwift Object Model
1	Coming soon
Darc	ntltom

ParentItem

ParentItem

Returns the parent to the Item.

Syntax:

Procedure: ParentItem: IItem;

API Calls

Delphi

Using PlanSwift Object Model

> Expand source

C#

Using Iltem Object Model

Using PlanSwift Object Model

Coming soo

VB/VBA (OLE)

1

Usin	g litem Object Model
1	Coming soon
Usin	g PlanSwift Object Model
1	Coming soon
. .	

PointCount

PointCount

Returns the number of digitizer points for the item.

Syntax:

Procedure: PointCount: Integer;

API Calls	
elphi	
Using PlanSwift Object Model	> Expand source
#	
Using Iltem Object Model	> Expand source
Using PlanSwift Object Model	

VB/VBA (OLE)

1

Usin	Using IItem Object Model	
1	Coming soon	
Using PlanSwift Object Model		
Usin	g PlanSwift Object Model	

PropertyCount

PropertyCount

Returns the number of properties for this item.

Syntax:

Procedure: PropertyCount: Integer;

API Calls

Delphi Using PlanSwift Object Model C# Using Iltem Object Model Using PlanSwift Object Model 1 Coming soon

VB/VBA (OLE)

Using	; lltem Object Model
1	Coming soon
Using	g PlanSwift Object Model
1	

Propertyltem

Propertyltem

Returns the PropertyObject at the given index.

Syntax:

Procedure: PropertyItem(Index: Integer);

API Calls

Delphi Using Iltem Object Model Using PlanSwift Object Model Coming soon

C#

Using Iltem Object Model

Expand source

Using PlanSwift Object Model

1 Coming soon

VB/VBA (OLE)

SetPoint

Using Iltem Object Model	

SetPoint

Sets the digitizer point specified by *PointIndex* to the given *X*, *Y* coordinates.

Syntax:

Procedure: SetPoint(PointIndex: Integer; X, Y: Double);

API Calls

Delphi

Using PlanSwift Object Model

C#

Using Iltem Object Model

Using PlanSwift Object Model

1 Coming so

VB/VBA (OLE)

Usin	g Iltem Object Model
1	Coming soon
Usin	g PlanSwift Object Model
1	Coming soon

SetPropertyFormula

SetPropertyFormula

Sets the given property formula to value.

Syntax:

Procedure: SetPropertyFormula(PropertyName, value: String);

API Calls

Delphi

Using PlanSwift Object Model

> Expand source

C#

Using IItem Object Model

Using PlanSwift Object Model

Coming soo

VB/VBA (OLE)

1

Using Iltem Object Model 1 Coming soon Using PlanSwift Object Model 1 Coming soon

IPropertyObject

IPropertyObject

This is the interface object for each property on a PlanSwift Iltem. In PlanSwift script, since each script is the property of an item, you can use the default *Item* and *Property* objects to access the Iltem and IPropertyObject that the script belongs to.

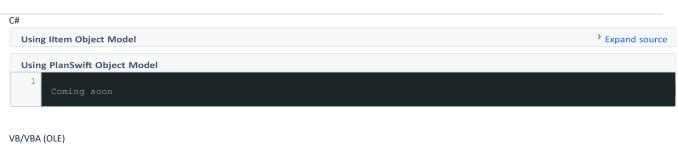
Syntax:

Procedure: Handle: HResult;

API Calls

Delphi		
Usin	g Iltem Object Model	Expand source
Using PlanSwift Object Model		
1	Coming soon	

Using Iltem Object Model	
1	Coming soon
Using PlanSwift Object Model	



Adjust

Adjust

Gets or Sets the Adjust attribute for the property.

Syntax:

Procedure: Adjust: String;

API Calls

Delphi

Using PlanSwift Object Model	Expand source
Using Planswitt Object Wodel	Expand source

C#

Using Iltem Object Model	> Expand source
Using PlanSwift Object Model	
1 Coming soon	

VB/VBA	(OLE)
Usin	g Iltem Object Model
1	Coming soon
Usin	g PlanSwift Object Model
1	Coming soon
Calc	ulateBeforeInherit

CalculateBeforeInherit

Gets or Sets the CalculateBeforeInherit attribute for the property.

Syntax:

Procedure: CalculateBeforeInherit: Boolean;

API Calls

Delphi

Using PlanSwift Object Model

> Expand source

C#

Using Iltem Object Model > Expand source

Using PlanSwift Object Model

Coming soo

VB/VBA (OLE)	
Using Iltem Object Model	
1 Coming soon	
Using PlanSwift Object Model	
1 Coming soon	
CompileDenyOLE	

CompileDenyOLE

Gets or Sets the CompileDenyOLE attribute for this property.

Syntax:

Procedure: CompileDenyOLE: Boolean;

API Calls

Delphi

Using PlanSwift Object Model

> Expand source

C#

Using Iltem Object Model	Expand source
Using PlanSwift Object Model	
1 Coming soon	



CompileDenyRead

CompileDenyRead

Gets or Sets the CompileDenyRead attribute for this property.

Syntax:

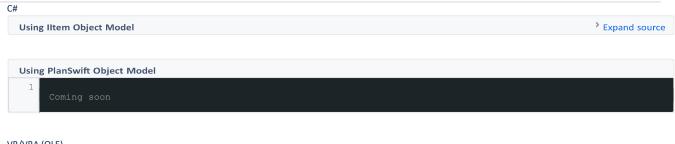
Procedure: CompileDenyRead: Boolean;

API Calls

Delphi

Using PlanSwift Object Model

> Expand source



VB/VBA (OLE)

Using Iltem Object Model		
1	Coming soon	
Using PlanSwift Object Model		
1	Coming soon	

CompileDenyWrite

CompileDenyWrite

Gets or Sets the CompileDenyWrite attribute for this property.

Syntax:

Procedure: CompileDenyWrite: Boolean;

Delphi		
Usin	g Iltem Object Model	> Expand source
Usin	ng PlanSwift Object Model	
1	Coming soon	

C#

Using Iltem Object Model	Expand source
Using PlanSwift Object Model	
1 Coming soon	

VB/VBA (OLE)

Usin	Using Iltem Object Model	
1	Coming soon	
Using PlanSwift Object Model		
1	Coming soon	

DecimalPlaces

DecimalPlaces

Gets or Sets the DecimalPlaces attribute for the property.

Syntax:

Procedure: DecimalPlaces: Integer;

API Calls

Delphi		
Usin	g Iltem Object Model	Expand source
Usin	g PlanSwift Object Model	
1	Coming soon	

C#		
	Using Iltem Object Model	> Expand source
	Using PlanSwift Object Model	
	1 Coming soon	

VB/VBA (OLE)

Using Iltem Object Model		
1	Coming soon	
Using PlanSwift Object Model		
1	Coming soon	

EditScript

EditScript

Opens the script property in the script editor. If the property is not of type ptScript, this method is ignored.

Syntax:

Procedure: EditScript;

API Calls

Delphi

Using PlanSwift Object Model

> Expand source

Using IItem Object Model		
1	Coming soon	
Using PlanSwift Object Model		
1	Coming soon	

C#	
Using Iltem Object Model	>
Using PlanSwift Object Model	
1 Coming soon	

ExecuteScript

ExecuteScript

Executes the script property, passing a CRLF delimited list of parameters. Returns the value assigned to Result in the script.

Syntax:

Procedure: ExecuteScript(ParamList: String = ''): Variant;

API Calls

Delphi

Using PlanSwift Object Model

> Expand source

Using Iltem Object Model		
1	Coming soon	
Using PlanSwift Object Model		
1	Coming soon	

C#	
Using Iltem Object Model	>
Using PlanSwift Object Model	
1 Coming soon	

Expression

Expression

Gets or Sets the Expression attribute for the property.

Syntax:

Procedure: Expression: Boolean;

API Calls

Delphi

Using PlanSwift Object Model

> Expand source

Using Iltem Object Model			
1	Coming soon		
Usin	Using PlanSwift Object Model		
1	Coming soon		

C#	
Using Iltem Object Model	>
Using PlanSwift Object Model	
1 Coming soon	

Formula

Formula

Gets or Sets the Formula attribute for the property.

Syntax:

Procedure: Formula: String;

API Calls

Delphi

Using PlanSwift Object Model	> Expand source
------------------------------	-----------------

C#

Using IItem Object Model	Expand source
Using PlanSwift Object Model	
1 Coming soon	

Using Iltem Object Model		
1		
	Coming soon	
Using PlanSwift Object Model		
1		

Group

Group

Gets or Sets the Group attribute for the property.

Syntax:

Procedure: Group: String;

API Calls

Delphi

Using PlanSwift Object Model

C#

Using IItem Object Model

Using PlanSwift Object Model

1 Coming soon

Usin	Using IItem Object Model		
1	Coming soon		
Usin	g PlanSwift Object Model		
1			
1	Coming soon		

ImageTransparent

ImageTransparent

Gets or Sets the ImageTransparent attribute for this property.

Syntax:

Procedure: ImageTransparent: Boolean;

API Calls

Delphi			
Usin	Using Iltem Object Model		
1	Coming soon		
Using PlanSwift Object Model			
1	Coming soon		

C#

Using Iltem Object Model	> Expand source
Using PlanSwift Object Model	

_

VB/VBA (OLE)

1

Using Iltem Object Model

1 Coming soon

Using PlanSwift Object Model

Coming soon

InheritAction

InheritAction

Gets or Sets the InheritAction attribute for this property.

Syntax:

Procedure: InheritAction: Inheritactions; iaNormal = 0 iaIgnore = 1 iaInheritFormula = 2 iaInheritResult = 3 iaFlatten = 4

API Calls

Delphi

Coming soon

Using Iltem O

Using Iltem Object Model	
Using PlanSwift Object Model	

> Expand source

C#

1

Using IItem Object Model > Expand source

Usi	g PlanSwift Object Model
1	Coming soon

VB/VBA (OLE) InheritPullFrom

InheritPullFrom

Gets or Sets the InheritPullFrom attribute for this property.

Using Iltem Object Model				
1				
	Coming soon			
Usin	Using PlanSwift Object Model			
1				
	Coming soon			

Syntax:

Procedure: InheritPullFrom: String;

API Calls

Delphi

Using PlanSwift Object Model

C#

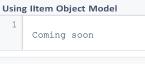
Using Iltem Object Model

Using PlanSwift Object Model

1 Coming s

VB/VBA (OLE)

1



Using PlanSwift Object Model

Coming soon

InputCondition

InputCondition

Gets or Sets the InputCondition attribute for the property.

Syntax:

Procedure: InputCondition: String;

API Calls

Delphi

Using PlanSwift Object Model

Expand source

C#

Using IItem Object Model	> Expand source
Using PlanSwift Object Model	

Coming soon

VB/VBA (OLE)

1

Usin	Using Iltem Object Model			
1	Coming soon			
Usin	Using PlanSwift Object Model			
1	Coming soon			

InputType

InputType

Gets or Sets the InputType attribute for the property.

Syntax:

Procedure: InputType: InputTypes;

- inpStoreLocal = 0 inpStoreParent
- = 1

API Calls

Delphi

Using Iltem Object Model

Using PlanSwift Object Model

Coming soon

C#

Using Iltem Object Model

Using PlanSwift Object Model

VB/VBA (OLE)

Using Iltem Object Model

 1
 Coming soon

 Using PlanSwift Object Model

Coming soon

InputUnits

InputUnits

Gets or Sets the InputUnits attribute for the property.

Syntax:

Procedure: InputUnits: String;

API Calls

Delphi

Using PlanSwift Object Model

C#

Using Iltem Object Model

Using PlanSwift Object Model

Coming soc

VB/VBA (OLE)

Using Iltem Object Model					
1	Coming soon				
Usin	Using PlanSwift Object Model				
1					

> Expand source

> Expand source

IsInherited

IsInherited

Gets or Sets the IsInherited attribute for this property.

Syntax:

Procedure: IsInherited: Boolean;

API Calls

Delphi

Using PlanSwift Object Model

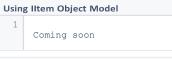
C#

Using Iltem Object Model

Using PlanSwift Object Model

1 Coming

VB/VBA (OLE)



Using PlanSwift Object Model

Coming soon

IsInput

1

IsInput

Gets or Sets the IsInput attribute for the property.

Syntax:

Procedure: IsInput: Boolean;

API Calls

Using IItem Object Model

Delphi

Using PlanSwift Object Model
C Expand source

C#

Expand source

Using PlanSwift Object Model							
1	Coming	soon					

Using Iltem Object Model					
1	Coming soon				
Usin	Using PlanSwift Object Model				
1	Coming soon				

List

List

Gets or Sets the List attribute for the property. If ListType = ItListthen this string will be the full path to the PlanSwift List Object as defined on the List tab on the main ribbon bar.

Syntax:

Procedure: List: String;

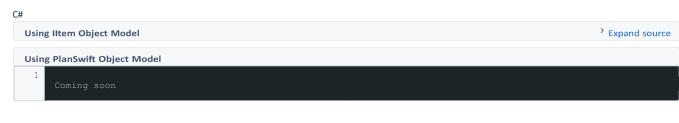
API Calls

Delphi

Using PlanSwift Object Model

> Expand source

Using Iltem Object Model		
1	Coming soon	
Usin	g PlanSwift Object Model	
50111	g Flanswirt Object Model	



vb/vba (OLE) ListColumnAutoWidth

ListColumnAutoWidth

Gets or Sets the ListColumnAutoWidth attribute for the property.

Syntax:

Procedure: ListColumnAutoWidth: Boolean;

API Calls

Delphi

Using PlanSwift Object Model

Expand source



Using Iltern Object Model 1 Coming soon Using PlanSwift Object Model 1 Coming soon

ListFromProperty

ListFromProperty

Gets or Sets the ListFromProperty attribute for the property;

Syntax:

Procedure: ListFromProperty: Boolean;

API Calls Delphi	
Using PlanSwift Object Model	> Expand source
C#	
Using Iltem Object Model	> Expand source
Using PlanSwift Object Model	
1 Coming soon	
VB/VBA (OLE)	
Using Iltem Object Model	
1 Coming soon	
Using PlanSwift Object Model	
1 Coming soon	

ListPropertiesToSet

ListPropertiesToSet

Gets or Sets the ListPropertiesToSet attribute for this property.

Syntax:

Procedure: ListPropertiesToSet: String;

API Calls Delphi Using PlanSwift Object Model C# Using Iltem Object Model Stepand source Calls C

vb/vba (OLE) ListResultColumn

ListResultColumn

Usin	g Iltem Object Model
1	Coming soon
Usin	g PlanSwift Object Model
1	Coming soon

Gets or Sets the ListResultColumn attribute for the property. If the ListType =ltList, this attribute specifies which column to return for the result.

Syntax:

Procedure: ListResultColumn: String;

API Calls

Delphi

Using PlanSwift Object Model

C#

1

Using Iltem Object Model

> Expand source

> Expand source

Using PlanSwift Object Model

Coming soon



ListReturnFullPath

Gets or Sets the ListReturnFullPath for this property.

Syntax:

Procedure: ListReturnFullPath: Boolean;

API Calls

Del	р	hi
-----	---	----

Using PlanSwift Object Model	> Expand source
C#	
Using IItem Object Model	Expand source
Using PlanSwift Object Model	

Using	g litem Object Model
1	Coming soon
Using	g PlanSwift Object Model
1	Coming soon

ListShow1Level

ListShow1Level

Gets or Sets the ListShow1Level attribute for this property.

Syntax:

Procedure: ListShow1Level: Boolean;

API Calls

Delphi

1

Using PlanSwift Object Model

C#

Using IItem Object Model

Using PlanSwift Object Model

. Comina s



ListShowOnlyTypes

ListShowOnlyTypes

Gets or Sets the ListShowOnlyTypes attribute for this property.

Syntax:

Procedure: ListShowOnlyTypes: String;

API Calls

Delphi

Using PlanSwift Object Model

> Expand source

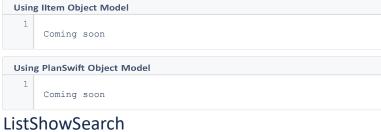
C#

Using IItem Object Model

Using PlanSwift Object Model

VB/VBA (OLE)

1



ListShowSearch

Gets or Sets the ListShowSearch attribute for the property.

Syntax:

Procedure: ListShowSearch: Boolean;

> Expand source
> Expand source

Usin	Using PlanSwift Object Model		
1			
	Coming soon		

VB/VBA (OLE)

Usin	Using Iltem Object Model		
1	Coming soon		
Usin	Using PlanSwift Object Model		
1	Coming soon		

ListType

ListType

Gets or Sets the ListType attribute for the property.

Syntax:

Procedure: <i>ListType: ListTypes;</i> ItSimpleList = 0 ItList = 1 ItTreeList = 2 ItExecutePlugin = 3
API Calls
Delphi
Using PlanSwift Object Model
C#
Using Iltem Object Model
Using PlanSwift Object Model
l Coming soon
VB/VBA (OLF)

vb/vba (ole) ListVisibleColumnsInDropdown

ListVisibleColumnsInDropdown

Usin	g Iltem Object Model
1	Coming soon
Usin	g PlanSwift Object Model
1	Coming soon

•

Gets or Sets the ListVisibleColumnsInDropdown attribute for this property.

Syntax:

Procedure: ListVisibleColumnsInDropdown: String;

API Calls

Delphi

Using PlanSwift Object Model

C#

Using Iltem Object Model

> Expand source

Using PlanSwift Object Model

Coming soo

VB/VBA (OLE)

_ ·		
Usin	Using Iltem Object Model	
1		
-	Coming soon	
Usin	Using PlanSwift Object Model	
1		
	Coming soon	

MeetsInputCondition

MeetsInputCondition

Returns true if theInputCondition has been met.

Syntax:

Procedure: MeetsInputCondition

API Calls

Delphi

Using PlanSwift Object Model

C#

Using Iltem Object Model

Using PlanSwift Object Model

Coming	

VB/VBA (OLE)

1

Usir	g lltem Object Model
1	Coming soon
Usir	ng PlanSwift Object Model
1	Coming soon

Name (3)

Name

Gets or Sets the Name attribute for the Property.

Syntax:

Procedure: Name: String;

API Calls

Delphi

Using PlanSwift Object Model

C#

1

> Expand source

> Expand source

Using PlanSwift Object Model

Coming soor

Using Iltem Object Model

VB/VBA (OLE)

Using Iltem Object Model
1
Coming soon

Using PlanSwift Object Model

1 Coming soon

PlugInToExecute

PlugInToExecute

Gets or Sets the PlugInToExecute attribute for this property.

Syntax:

Procedure: PlugInToExecute: String;

API Calls

Delphi

Using PlanSwift Object Model



VB/VBA (OLE)

Usin	Using Iltem Object Model		
1	Coming soon		
Usin	Using PlanSwift Object Model		
1	Coming soon		

PlugInToExecuteButtonCaption

PlugInToExecuteButtonCaption

Gets or Sets the PlugInToExecuteButtonCaption attribute for this property.

Syntax:

Procedure: PlugInToExecuteButtonCaption: String;

API Calls

Delphi

Using PlanSwift Object Model

C#

Using Iltem Object Model

Using PlanSwift Object Model

1 Coming soc

VB/VBA (OLE)

Using Iltem Object Model	
1	Coming soon
Using PlanSwift Object Model	
1	Coming soon

PropertyType

PropertyType

Returns the Type attribute for the property.

Syntax:

Procedure: PropertyType: String;

API Calls

Using IItem Object Model

Delphi

Using PlanSwift Object Model

C#

Usin	Using PlanSwift Object Model			
1				
	Coming			

VB/VBA (OLE)

Usin	Using Iltem Object Model			
1				
	Coming soon			
Usin	Using PlanSwift Object Model			
1				
	Coming soon			

ResultAsString

ResultAsString

Returns the property result of the property.

Syntax:

Procedure: ResultAsString: String;

API Calls

Delphi

Using PlanSwift Object Model

C#

Using Iltem Object Model
Using PlanSwift Object Model
1
Coming soon

VB/VBA (OLE)

Usin	Using Iltem Object Model		
1	Coming soon		
Using PlanSwift Object Model			
Usin	g PlanSwift Object Model		

ResultAsInteger

ResultAsInteger

Returns the property result as an integer if possible.

Syntax:

Procedure: ResultAsInteger: Integer;

API Calls

Delphi

Using PlanSwift Object Model	Expand source
------------------------------	---------------

C#

Using Iltem Object Model

Using PlanSwift Object	Model		
1			
Coming soon			

> Expand source

VB/VBA (OLE)

Using Iltem Object Model			
1	Coming soon		
Using PlanSwift Object Model			
1	Coming soon		

ResultAsFloat

ResultAsFloat

Returns the property resultasa Double.

Syntax:

Procedure: ResultAsFloat: Double;

API Calls

Delphi

Using PlanSwift Object Model

C#

Using IItem Object Model
Using PlanSwift Object Model
1

VB/VBA (OLE)

Using IItem Object Model			
1	Coming soon		
Usin	Using PlanSwift Object Model		
1	Coming soon		

ResultAsVariant

ResultAsVariant

Returns the property result as a Variant;

Syntax:

Procedure: ResultAsVariant: Variant;

API Calls

Delphi

Using	PlanSwift	Object	Model
-------	-----------	--------	-------

C# Using IItem Object Model Using PlanSwift Object Model 1 Coming soon

> Expand source

VB/VBA (OLE)

Using Iltem Object Model				
1				
	Coming soon			
Usin	Using PlanSwift Object Model			
1				
	Coming soon			

ScriptType

ScriptType

Gets or Sets the ScriptType attribute for this property.

Syntax:

Procedure: ScriptType: ScriptTypes;

- stEvent = 0 stMethod = 1
- •

API Calls

Delphi

Using PlanSwift Object Model

C#

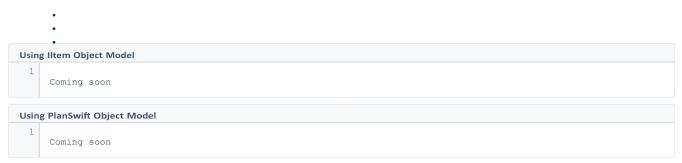
1

Using Iltem Object Model

Using PlanSwift Object Model

Coming soon

VB/VBA (OLE)



ScriptLanguage

ScriptLanguage

Gets or Sets the ScriptLanguage attribute for this property.

Syntax:

```
Procedure: ScriptLanguage: ScriptLanguages;
sIPascal = 0 sIBasic = 1 sIExecute = 2
```

API Calls

Delphi

Using PlanSwift Object Model

> Expand source

C#

Using IItem Object Model

Using PlanSwift Object Model

Coming soor

VB/VBA (OLE)

1

Using Iltem Object Model 1 Coming soon Using PlanSwift Object Model 1

Coming soon

ScriptParameters

ScriptParameters

Gets or Sets the ScriptParameters attribute for this property. This string is a CRLF delimited list of Parameter names.

Syntax:

Procedure: ScriptParameters: String;

API Calls

Delphi

Using PlanSwift Object Model

C#

Using IItem Object Model	> Expand source
Using PlanSwift Object Model	
1 Coming soon	

> Expand source

VB/VBA (OLE)

Using Iltem Object Model		
1	Coming soon	
Usin	g PlanSwift Object Model	

Coming soon

SimpleList

SimpleList

Gets or Sets the SimpleList attribute for the property. If ListType = ItSimpleList, the SimpleList attribute will be the CRLF delimited string of list items.

Syntax:

Procedure: SimpleList: String;

	'I Calls	
phi		
Using	g Iltem Object Model	> Expand source
Using	g PlanSwift Object Model	
1	Coming soon	

C#

Using Iltem Object Model	> Expand source
Using PlanSwift Object Model	
1 Coming soon	

VB/VBA (OLE)

SliderMax

SliderMax

Gets or Sets the SliderMax attribute for the property.

Usin	Using Iltem Object Model		
1	Coming soon		
Usin	Using PlanSwift Object Model		
1	Coming soon		

Syntax:

Procedure: SliderMax: Integer;

API Calls

Using PlanSwift Object Model	Expand source

C#

Delphi

Using IItem Object Model	> Expand source
Using PlanSwift Object Model	
1 Coming soon	

VB/VBA (OLE)

Usin	Using Iltem Object Model		
1	Coming soon		
Usin	g PlanSwift Object Model		
1	Coming soon		

SliderMin

SliderMin

Gets or Sets the SliderMin attribute for this property.

Syntax:

Procedure: SliderMin: Integer;

API Calls

Delphi

Using PlanSwift Object Model	> Expand source
------------------------------	-----------------

C#

Using Iltem Object Model	> Expand source
Using PlanSwift Object Model	
1 Coming soon	

VB/VBA (OLE)

Usin	Using Iltem Object Model		
1			
	Coming soon		
Using PlanSwift Object Model			
	g rianswirt object model		
1			

SliderShowTicks

SliderShowTicks

Gets or Sets the SliderShowTicks attribute for this property.

Syntax:

Procedure: SliderShowTicks: Boolean;

API Calls Delphi Using PlanSwift Object Model > Expand source C# Using Iltem Object Model > Expand source Using PlanSwift Object Model

VB/VBA (OLE)

Usin	Using IItem Object Model		
1			
	Coming soon		
Usin	Using PlanSwift Object Model		
1			
	Coming soon		

SliderTickFrequency

SliderTickFrequency

Gets or Sets the SliderTickFrequency attribute for this property.

Syntax:

Procedure: SliderTickFrequency: Integer;

API Calls

Delphi

Using PlanSwift Object Model	Expand source
------------------------------	---------------

C#

Using Iltem Object Model	Expand source
Using PlanSwift Object Model	
1 Coming soon	

VB/VBA	B/VBA (OLE)	
Using	g Iltem Object Model	
1	Coming soon	
Usin	g PlanSwift Object Model	
1	Coming soon	
Curt	anallidaan	

SystemHidden

SystemHidden

Returns True if the property is Hidden by the system.

Syntax:

Procedure: SystemHidden: Boolean;

API Calls

Delphi

Using PlanSwift Object Model	> Expand source

C#

Using IItem Object Model	> Expand source
Using PlanSwift Object Model	

VB/VBA (OLE)

1

Usin	Using Iltem Object Model	
1	Coming soon	
Using PlanSwift Object Model		
Usin	g PlanSwift Object Model	

SystemLocked

SystemLocked

Returns True if the property is locked by the system.

Syntax:

Procedure: SystemLocked: Boolean;

API Calls

Delphi

Using PlanSwift Object Model	> Expand source

C#	
Using IItem Object Model	Expand source
Using PlanSwift Object Model	
1 Coming soon	
VB/VBA (OLE)	

Usin	Using Iltem Object Model	
1	Coming soon	
Usin	g PlanSwift Object Model	
1	Coming soon	

TreeList

TreeList

Gets or Sets the TreeList attribute of the property. If ListType = ItTreeList this attribute will contain the full path to the treelist item to use for a root item in the list.

> Expand source

Syntax:

Procedure: TreeList: String;

API Calls

Delphi

Using PlanSwift Object Model

C#

Cn	
Using Iltem Object Model	> Expand source
Using PlanSwift Object Model	
1 Coming soon	



ExecuteScript

Executes the script property, passing a CRLF delimited list of parameters. Returns the value assigned to Result in the script.

Syntax:

Procedure:ExecuteScript(ParamList: String = ''): Variant;

Expand source

C#	
Using Iltem Object Model	Expand source
Using PlanSwift Object Model	
1 Coming soon	

vb/vba (ole) Units

Units

Gets or Sets the Units attribute for the property.

Usin	g Iltem Object Model		
1	Coming soon		
Usin	Using PlanSwift Object Model		
1	Coming soon		

Syntax:

Procedure: Units: String;

API Calls

Delphi Using PlanSwift Object Model C#

Using Iltem Object Model	Expand source
Using PlanSwift Object Model	

VB/VBA (OLE)

Using Iltem Object Model	
1	Coming soon
Usin	g PlanSwift Object Model
1	Coming soon

UserHidden

UserHidden

Gets or Sets the UserHidden attribute for the property.

Syntax:

Procedure: UserHidden: Boolean;

API Calls

Delphi

Using PlanSwift Object Model

> Expand source

C#

Using Iltem Object Model	> Expand source
Using PlanSwift Object Model	
l Coming soon	

VB/VBA (OLE)

1

Using Iltem Object Model

Coming soon

Using PlanSwift Object Model

1 Coming soon

UserLocked

UserLocked

Gets or Sets the UserLocked attribute of the property.

Syntax:

Procedure: UserLocked: Boolean;

API Calls

Delphi

Using PlanSwift Object Model



VB/VBA (OLE)

Using Iltem Object Model	
1	Coming soon
Usin	g PlanSwift Object Model
1	Coming soon

IsBeta

IsBeta

Returns True if Beta user, False if not.

Syntax:

Procedure: IsBeta: Boolean;

Code Reference:

1. Navigate to Plugin Store->Tool Manager and create a new Plugin

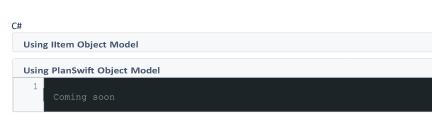
2. Set the plugin type to Script Code and open the Editor

3. Copy Code into the editor
4. Press run

API	Cal	ls
API	Cal	IS

Delphi





VB/VBA (OLE)

IsJobOpen

IsJobOpen

Tests whether the PlanSwift application actually has a "Job" opened in the editor.

Usin	g Iltem Object Model
1	Coming soon
Usin	g PlanSwift Object Model
1	Coming soon

Syntax:

Procedure: IsJobOpen: Wordbool;

API Calls

Delphi

Using PlanSwift Object Model	> Expand source

C#	
Using IItem Object Model	> Expand source
Using PlanSwift Object Model	
1 Coming soon	
VB/VBA (OLE)	
Using IItem Object Model	
1 Coming soon	

Usin	Using PlanSwift Object Model		
1			
	Coming soon		

IsUnlocked

IsUnlocked

Checks the product activation status of a plugin. If AllowUnlock is true, the user is prompted to Activate if needed.

Syntax:

Procedure: IsUnlocked(AProduct: String; AMajorVer: Integer; AMinorVer: Integer; AllowUnlock: Boolean): Boolean;

Code Reference:

- 1. Create a New Form application
- 2. Add a button to the form
- 3. Add PlanSwift to reference (Planswift9_tlb in the uses)
- 4. Copy code to button onclick event

API Calls

Delphi

Using PlanSwift Object Model

> E

C#

Using Iltem Object Model

Using PlanSwift Object Model

Coming soor

VB/VBA (OLE) Using Iltem Object Model Coming soon Using PlanSwift Object Model Coming soon Coming soon

NewBlankPage

NewBlankPage

Creates a blank page in the current job and returns the Page Item that was created.

Syntax:

Procedure: NewBlankPage(const AName: WideString; AWidth, AHeight, ADPI: Integer; const AScale: WideString): Iltem;

Code Reference:

- 1. Create a New Form application
- 2. Add a button to the for
- 3. Add PlanSwift to reference (Planswift9_tlb in the uses)
- 4. Copy code to button onclick event

API Calls

Delphi

Using PlanSwift Object Model

> Expand source

C#	¥	
	Using IItem Object Model) E
	Using PlanSwift Object Model	
	1. Coming soon	

VB/VBA (OLE)

Usin	Using Iltem Object Model	
1	Coming soon	
Using PlanSwift Object Model		
1	Coming soon	

NewChangeGroup

NewChangeGroup

Starts a new change group.

Syntax:

Procedure: NewChangeGroup(GroupName: String);

Code Reference

- 1. Create a New Forms Application
- 2. Add a PlanSwift to the References (Planswift_Tlb)
- 3. Add a button to the form
- 4. Copy code below to the onclick event of the button

5. Compile and run

API Calls

Delphi		
Usin	g litem Object Model	> Expand source
Usin	g PlanSwift Object Model	
1	Coming soon	

C#

Using Iltem Object Model	> Expand source

Using Pla	nSwift Ob	ject Mode
-----------	-----------	-----------

Coming soon

VB/VBA (OLE)

1



NewItem

Creates a new IItem as a child of ParentPath.

Syntax:

Usin	Using Iltem Object Model		
1	Coming soon		
Using PlanSwift Object Model			
1	Coming soon		

Procedure: NewItem(ParentPath, ItemType: String; AName: String = ''): IItem;

API Calls		
Delphi		
Using PlanSwift Object Model		
#		
Using IItem Object Model		
Using PlanSwift Object Model		
1 Coming soon		
B/VBA (OLE)		
Using Iltem Object Model		
1 Coming soon		

Using PlanSwift Object Model

1 Coming soon

NewItemEx (2)

NewItemEx

Creates a new IItem as a child of ParentPath. If EditProperties is true, then the property editor will display when the item is created.

Syntax:

Procedure: NewItemEx(ParentPath, ItemType: String; AName: String = "; EditProperties: Boolean): IItem;

API Calls

Delphi

Using PlanSwift Object Model

C#

Using Iltem Object Model	> Expand source
Using PlanSwift Object Model	

> Expand source

Coming soon

VB/VBA (OLE)

Using Iltem Object Model		
1	Coming soon	
Usin	g PlanSwift Object Model	
1		

Coming soon

NewJobEx

NewJobEx

Starts a "new" job in the PlanSwift application.

Syntax:

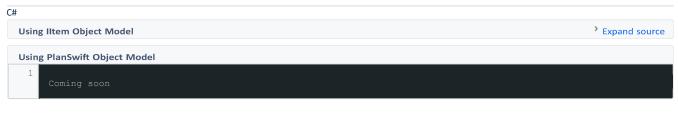
Procedure: NewJobEx(const JobName: WideString = ''): Wordbool;

API Calls

Delphi

Using PlanSwift Object Model

> Expand s



VB/VBA (OLE)

Usin	Using Iltem Object Model	
1	Coming soon	
Using PlanSwift Object Model		

Coming soon

NewPoint (2)

NewPoint

1

Adds a new point to ItemPath at the X, Y coordinates. If the Item is not found or is not a drawing object, this procedure will be ignored.

Syntax:

Procedure: NewPoint(ItemPath: String; X, Y: Double);

Code Reference:

- 1. Create a New Forms Application
- 2. Add a PlanSwift to the References (Planswift_Tlb)
- 3. Add a button to the form
- 4. Copy code below to the onclick event of the button
- 5. Compile and run

API Calls



C#	
Using Iltem Object Model	> Expand s
Using PlanSwift Object Model	
1 Coming soon	

VB/VBA (OLE)

11-1-1	
Using	g Iltem Object Model
1	
	Coming soon
Usin	g PlanSwift Object Model
1	
-	
	Coming soon

NewSection (2)

NewSection

Returns the newly created section as an IItem, as a child item of ParentPath. If ParentPath is not found or is not a digitizer item, this function returns Nil.

Syntax:

Procedure: NewSection(ParentPath: String; SectionName: String = "): IItem;

API Calls

Delphi

Using PlanSwift Object Model

Expand source

C#

Using Iltem Object Model	Expand source
Using PlanSwift Object Model	

VB/VBA (OLE)

Usin	Using Iltem Object Model		
1	Coming soon		
Using PlanSwift Object Model			
1	Coming soon		

OnClose

OnClose

Triggered when the PlanSwift application closes.

Syntax:

Procedure: OnClose;

API Calls

Using	g Iltem Object Model	Expand source	
Using PlanSwift Object Model			
1	Coming soon		

C#

Using	g Iltem Object Model	> Expand source		
Using PlanSwift Object Model				
1	Coming soon			

VB/VBA (OLE) Using Iltem Object Model Coming soon Using PlanSwift Object Model Coming soon Coming soon

OnDoneRecordingDigitizer

OnDoneRecordingDigitizer

Triggered when an item section finishes recording.

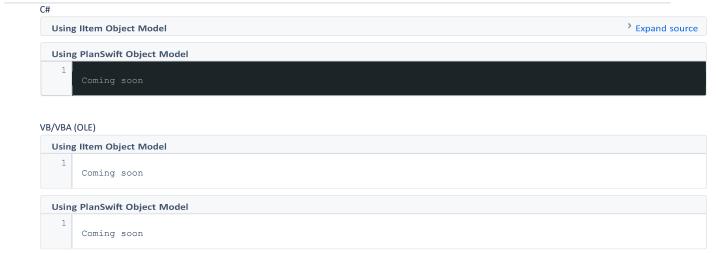
Syntax:

Procedure: OnDoneRecordingDigitizer(ItemPath: String);

API Calls

Delphi

Usin	Using Iltem Object Model Expand source	
Usin	g PlanSwift Object Model	
1		
	Coming soon	



OnCopyItem

OnCopyItem

Triggered when the PlanSwift application copies an item.

Syntax:

Procedure: OnCopyItem;

API Calls

Delphi			
Usin	g Iltem Object Model	> Expand source	
Usin	Using PlanSwift Object Model		
1	Coming soon		

C#

Using IItem Object Model	Expand source	
Using PlanSwift Object Model		
1 Coming soon		

VB/VBA (OLE)

Usin	Using Iltem Object Model		
1	Coming soon		
Usin	Using PlanSwift Object Model		
1	Coming soon		

OnDigitizerSectionChanged

OnDigitizerSectionChanged

Triggered when the PlanSwift application focuses a new section item.

Syntax:

Procedure: OnDigitizerSectionChanged;

API Calls

Delphi		
Using IItem Object Model		> Expand source
Usin	g PlanSwift Object Model	
1	Coming soon	

C#

Using Iltem Object Model	Expand source
Using PlanSwift Object Model	
1 Coming soon	

VB/VBA (OLE)

Usin	Using Iltem Object Model		
1	Coming soon		
Using PlanSwift Object Model			
1	Coming soon		

OnDoneRecording

OnDoneRecording

Triggered when an item has finished recording.

Syntax:

Procedure: OnDoneRecording(ItemPath: String);

API Calls

Delphi

Using IItem Object Model	> Expand source
Using PlanSwift Object Model	
Coming soon	

C#

Using Iltem Object Model
Cxpand source

Using PlanSwift Object Model

Coming soon

VB/VBA (OLE)

Using IItem Object Model	
Coming soon	
Commig Soon	
Using PlanSwift Object Model	
Coming soon	

OnItemChange

OnItemChange

Triggered when an item, specified by ItemPath, has been changed.

Syntax:

Procedure: OnItemChange(ItemPath: String);

API Calls

Using Iltem Object Model > Expand source Using PlanSwift Object Model 1 Coming soon Coming soon

C#

#
Using Iltem Object Model
Using PlanSwift Object Model
Coming soon

VB/VBA (OLE)

Usin	Using Iltem Object Model		
1	Coming soon		
Using PlanSwift Object Model			

OnItemDelete

OnItemDelete

Triggered when an item is deleted. ItemPath specifies which item is deleted.

Syntax:

Procedure: OnItemDelete(ItemPath: String);

API Calls

Delphi

1

Using Iltem Object Model
Cusing PlanSwift Object Model

Coming soon

C#

Using Iltem Object Model > Expand source

Using PlanSwift Object Model

l Coming soon

VB/VBA (OLE)

Usin	Using Iltem Object Model		
1	Coming soon		
Usin	Using PlanSwift Object Model		
1	Coming soon		

OnJobClose

OnJobClose

Triggered when the current job closes.

Syntax:

Procedure: OnJobClose;

API Calls

Delphi

Using Iltem Object Model

> Expand source

Usin	Using PlanSwift Object Model		
1	Coming soon		

C#

Using Iltem Object Model	> Expand source
Using PlanSwift Object Model	
1 Coming soon	

VB/VBA (OLE)

Usin	Using Iltem Object Model		
1	Coming soon		
Usin	Using PlanSwift Object Model		
1			
	Coming soon		

OnJobOpen

OnJobOpen

Triggered when a new job is opened.

Syntax:

Procedure: OnJobOpen;

API Calls

Delphi Using Iltem Object Model

1
2 PlanSwift and FreshDesk Code:
3
4
5 Procedure OnJobOpen;
6 Begin
7 // Process as needed.
8 End;
9
PlanSwift.OnJobOpen := OnJobOpen;

Using PlanSwift Object Model

Coming soon

C#

Using Iltem Object Model	Expand source
Using PlanSwift Object Model	
1 Coming soon	

VB/VBA (OLE)

OnNewItem

OnNewItem

Triggered when a new item, specified by ItemPath, is created.

Usin	Using Iltem Object Model		
1	Coming soon		
Using PlanSwift Object Model			
1	Coming soon		

Syntax:

Procedure: OnNewItem(ItemPath: String);

API Calls

Delphi

Usin	g Iltem Object Model	> Expand source
Usin	g PlanSwift Object Model	
1	Coming soon	

> Expand source

C#

Using Iltem Object Model

Using PlanSwift Object Model

1 Coming soon

VB/VBA (OLE)

Usin	Using Iltem Object Model			
1	Coming soon			
Usin	Using PlanSwift Object Model			
1	Coming soon			

OnNewJob

OnNewJob

Triggered when the application starts a "new" job.

Syntax:

Procedure: OnNewJob;

API Calls

Delphi

Using	g Iltem Object Model	> Expand source	
Usin	Using PlanSwift Object Model		
1	Coming soon		

C#

Using IItem Object Model	> Expand source
Using PlanSwift Object Model	
Coming soon	

VB/VBA (OLE)

Using	Using IItem Object Model			
1	Coming soon			
Using	Using PlanSwift Object Model			
1	Coming soon			

OnSelectedPageChange

OnSelectedPageChange

Triggered when the application changes to a "new" page in the job.

Syntax:

Procedure: OnSelectedPageChange;

API Calls

Delphi

Usin	g Iltem Object Model	> Expand source
Usin	g PlanSwift Object Model	
1	Coming soon	

C#

Usi	ing Iltem Object Model	> Expand source
Usi	ing PlanSwift Object Model	
1	Coming soon	

VB/VBA (OLE) Using litem Object Model Coming soon Using PlanSwift Object Model

Coming soon

OnSelectedSelectionChanged

OnSelectedSelectionChanged

Triggered when the application changes to a new selection.

Syntax:

Procedure: OnSelectedSelectionChanged;

API Calls

Delphi

Usin	Using litem Object Model > Expand source		
Usin	Using PlanSwift Object Model		
1	Coming soon		



Using Iltem Object Model	
1	Coming soon
Usin	g PlanSwift Object Model

OnS	electio	onCha	nged

Coming soon

OnSelectionChanged

Triggered when the application changes focus to a new "selectable" item in the editor.

Syntax:

Procedure: OnSelectionChanged;

Using IItem Object Model	> Expand source
Using PlanSwift Object Model	
1 Coming soon	
C#	

VB/VBA (OLE)

Usin	Using IItem Object Model		
1	Coming soon		
Using PlanSwift Object Model			
1	Coming soon		

OpenJob

OpenJob

Opens the job specified by JobPath. Returns True if successful, false if the job could not be found or opened.

Syntax:

Procedure: OpenJob(JobPath: String): Boolean;

Code Reference:

1. Navigate to Plugin Store->Tool Manager and create a new Plugin

- 2. Set the plugin type to Script Code and open the Editor
- 3. Copy Code into the editor
- 4. Press run

API Calls

Delphi

Using PlanSwift Object Model

C#

Using Iltem Object Model

> Expand source

> Expand source

Using PlanSwift Object Model

Coming soo

VB/VBA	(OLE)
Usin	g Iltem Object Model
1	Coming soon
Usin	g PlanSwift Object Model
1	Coming soon
0	

OpenJobEx

OpenJobEx

Shows the PlanSwift Open Job Dialog for the user to select a job to open.

Syntax:

Procedure: OpenJobEx;

Code Reference:

- 1. Create a New Forms Application
- 2. Add a PlanSwift to the References (Planswift_Tlb)
- 3. Add a button to the form
- 4. Copy code below to the onclick event of the button
- 5. Compile and run

Usin	Using Iltem Object Model	
1	Coming soon	
Usin	g PlanSwift Object Model	
1	Coming soon	

API Calls	
Delphi	
Using PlanSwift Object Model	Expand source
C#	
Using Iltem Object Model	> Expand source
Using PlanSwift Object Model	
1 Coming soon	

vb/vba (OLE) PointCount (2)

PointCount

Returns the number of digitizer points on the item section.

Syntax:

Procedure: PointCount(ItemPath: String): Integer;

Code Reference:

1. Navigate to Plugin Store->Tool Manager and create a new Plugin

2. Set the plugin type to Script Code and open the Editor

- 3. Copy Code into the editor
- 4. Press run

API Calls

Del	

Using PlanSwift Object Model

C#

Using IItem Object Model

Using PlanSwift Object Model



VB/VBA (OLE)

Usin	Using IItem Object Model		
1	Coming soon		
Usin	Using PlanSwift Object Model		
1	Coming soon		

PostChanges

PostChanges

Post changes made since call to NewChangeGroup.

Syntax:

Procedure: PostChanges; Code

Reference:

1. Create a New Forms Application

- 2. Add a PlanSwift to the References (Planswift_Tlb)
- 3. Add a button to the form
- 4. Copy code below to the onclick event of the button
- 5. Compile and run

API Calls

Delphi

Using PlanSwift Object Model

C#

Using Iltem Object Model

Using PlanSwift Object Model

Coming soon

VB/VBA (OLE)

1

Root

Root

Returns the Root tree object.

Usin	Using Iltem Object Model		
1			
	Coming soon		
Usin	Using PlanSwift Object Model		
1			
	Coming soon		

> Expand source

Expand source

Syntax:

Procedure: Root: IItem

Code Reference:

- 1. Navigate to Plugin Store->Tool Manager and create a new Plugin
- 2. Set the plugin type to Script Code and open the Editor
- 3. Copy Code into the editor
- 4. Press run

API Calls

Delphi

Using PlanSwift Object Model

C#

Using Iltem Object Model

Using PlanSwift Object Model

1 Coming so

VB/VBA (OLE)

Usin	Using IItem Object Model		
1	Coming soon		
Usin	Using PlanSwift Object Model		
1	Coming soon		

SaveScreenShot

SaveScreenShot

Save a screenshot of the active monitor to a specified filespec.

Syntax:

Procedure: SaveScreenShot(const FileName: WideString; Prompt: WordBool): WordBool;

Code Reference:

1. Create a New Form application

- 2. Add a button to the form
- 3. Add PlanSwift to reference (Planswift9_tlb in the uses)
- 4. Copy code to button onclick event

API Calls

Delphi

Using PlanSwift Object Model

> Expand source

C# Using Iltem Object Model Using PlanSwift Object Model 1 Coming soon VB/VBA (OLE) Using Iltem Object Model 1 Coming soon Using PlanSwift Object Model 1 Coming soon

SelectedItem

SelectedItem

Returns the currently selected Item, or Nil if no item is selected.

Syntax:

Procedure: SelectedItem: IItem

API Calls

Usin	g Iltem Object Model	> Expand source	
Usin	Using PlanSwift Object Model		
1	Coming soon		

C#

Using IItem Object Model	Expand source
Using PlanSwift Object Model	
1 Coming soon	

VB/VBA (OLE)

Usin	Using Iltem Object Model		
1	Coming coon		
	Coming soon		
Usin	Using PlanSwift Object Model		
1			
	Coming soon		

SelectedPage

SelectedPage

Returns the currently selected page item or Nil if no page is selected.

Syntax:

Procedure: SelectedPage: Iltem;

API Calls Delphi	
Using PlanSwift Object Model	
C#	
Using Iltem Object Model	
Using PlanSwift Object Model	
1	
Coming soon	
VB/VBA (OLE)	
VB/VBA (OLE)	
VB/VBA (OLE) Using Iltem Object Model Coming soon	
VB/VBA (OLE) Using litem Object Model	

SelectItemDialog

SelectItemDialog

Displays the Select Item dialog to the user, then returns the selected item.

Syntax:

Procedure: SelectItemDialog(Header: String; Title: String; RootItemID: String): IItem;

Code Reference:

1. Create a New Form application

2. Add a button to the form

- 3. Add PlanSwift to reference (Planswift9_tlb in the uses)
- 4. Copy code to button onclick event

API Calls

Delphi

Using PlanSwift Object Model

Expand source

Expand source

C#

Using Iltem Object Model

Using PlanSwift Object Model

Coming soor

VB/VBA (OLE)

1

Usin	g Iltem Object Model
1	Coming soon
Usin	g PlanSwift Object Model
1	Coming soon

SelectionList SelectionList

Returns an ISelectionList object of all the selected items.

Syntax:

Procedure: SelectionList: ISelectionList;

Code Reference:

- 1. Create a New Form application
- 2. Add a button to the form
- 3. Add PlanSwift to reference (Planswift9_tlb in the uses)
- 4. Copy code to button onclick event

API Calls

Delphi

Using PlanSwift Object Model

C#

Using Iltem Object Model
Coming soon
Using PlanSwift Object Model

Using Iltem Object Model

Using PlanSwift Object Model

Coming soon

VB/VBA (OLE) SetPoint (2)

SetPoint

Sets the XY coordinates of the specified point.

Syntax:

Procedure: SetPoint(ItemPath: String; PointIndex: Integer; X, Y: Double);

Code Reference:

- 1. Create a New Forms Application
- 2. Add a PlanSwift to the References (Planswift_Tlb)
- 3. Add a button to the form
- 4. Copy code below to the onclick event of the button
- 5. Compile and run

API Calls

Delphi

Using PlanSwift Object Model

> Expand source

Using	g PlanSwift Object Model	
1	Coming soon	
VB/VBA	(OLE)	
VB/VBA		
Using	(OLE) g litem Object Model	

SetPropertyFormula (2)

SetPropertyFormula

Sets the Items Property Formula to the specified value. This will also create a new Property with the default Type as Text if the property does not exist.

Syntax:

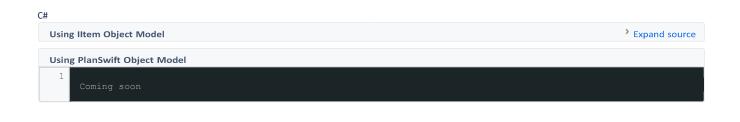
Procedure: SetPropertyFormula(ItemPath, PropertyName, Value: String);

API Calls

Delphi

Using PlanSwift Object Model

> Expand source



> Expand source

VB/VBA (OLE)

1

Using Iltem Object Model

Using PlanSwift Object Model

SetSelected

SetSelected

Set the job "object" to either selected or not selected based on the specified itempath.

Syntax:

Procedure: SetSelected(const ItemPath: WideString; Value: WordBool);

Code Reference:

- 1. Create a New Form application
- 2. Add a button to the form
- 3. Add PlanSwift to reference (Planswift9_tlb in the uses)
- 4. Copy code to button onclick event

API Calls

Delphi

Using PlanSwift Object Model	> Expand source
C#	
Using Iltem Object Model	Expand source
Using PlanSwift Object Model	

VB/VBA (OLE)

Usin	Using IItem Object Model		
1	Coming soon		
Usin	g PlanSwift Object Model		
1	Coming soon		

SetZoom

SetZoom

Defines the current "zoom" scale factor for the active page.

Syntax:

Procedure: Set_Zoom(Value: Double); Code

Reference:

- 1. Create a New Form application
- 2. Add a button to the form
- 3. Add PlanSwift to reference (Planswift9_tlb in the uses)
- 4. Copy code to button onclick event

API Calls

Delphi

Usin	Using Iltem Object Model		
1	Coming soon		
Usin	g PlanSwift Object Model		
1	Coming soon		



VB/VBA (OLE)

ISelectionList

ISelectionList

A simple list that contains all of the items selected when the list is created.

Syntax:

Procedure: Coming soon

API Calls

De	lphi	
	Using	; lltem Object Model
	11-1	
	Using	g PlanSwift Object Model
	1	Coming soon



VB/VBA (OLE)

1 Coming soon	Usin	g Iltem Object Model
	1	Coming soon

Usin	Using PlanSwift Object Model	
1		
	Coming soon	

Count

Count

Returns the number of objects in the list.

Syntax:

Procedure: Count: Integer;

API Calls

Delphi

Using Iltem Object Model

Using PlanSwift Object Model

Coming soon

C#

Using IItem Object Model

Using PlanSwift Object Model

_____Coming s

VB/VBA (OLE)

1	Coming soon
Usin	g PlanSwift Object Model

Items (1)

Items

Read-only collection of items. As with all collections, Index is a 0 based value.

Syntax:

Procedure: Items(Index: Integer): IItem;

API Calls

Delphi

Usin	g Iltem Object Model	> Expand source
Usin	g PlanSwift Object Model	
1	Coming soon	

C#

Using IItem Object Model

Using	PlanSwift	Object	Model

Coming soon

VB/VBA (OLE)

Usin	g Iltem Object Model
1	Coming soon
Usin	g PlanSwift Object Model
1	Coming soon

SetActiveTab (TabName: String)

SetActiveTab (TabName: String)

Passing the name of the tab to the method. will set the active tab in PlanSwift.

API Calls

C#	
Usin	g PlanSwift Object Model
1	
2	
3	IPlanSwift planSwift = new IPlanSwift();
4	while (!planSwift.IsLoaded)
5	
6	Sleep(10);
7	
,	<pre>planSwift.SetActiveTab("Home");</pre>

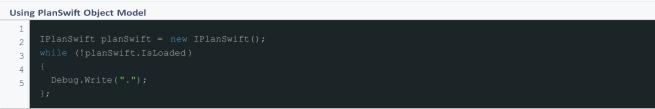
IsLoaded

IsLoaded

Will Wait For PlanSwift to finish loading.

API Calls

C#



Scripting Documentation

Scripting Documentation

PlanSwift's internal scripting provides the means to access PlanSwift's internal structure in a coding environment to help automate PlanSwift functionality. Scripting also provides the developer the capability to write and automate drawing tasks as well as complex formula calculations. PlanSwift's internal scripting functionality executes faster than PlanSwift's API functionality. Scripting for PlanSwift's root settings are documented in the PlanSwift Structure section. Click here to see them.

PlanSwift does not provide technical support for this function.

Using API Methods in Scriptin g Scripting Interfac e Scripting - Function s Developer Docs -- Freshdesk Xfe r

Using API Methods in Scripting

Using API Methods in Scripting

PlanSwift's Scripting Interface allows you to hook into PlanSwift's API and use it, or you can use the default Scripting Reference. If you want to write something inside of PlanSwift, just for PlanSwift, then you can use the Scripting Interface.

Documentation on the Scripting Interface and the Scripting Reference are in process.

Scripting Interface

Scripting Interface

To use the Scripting Interface, follow the steps below.

1. Click on Plugins on the PlanSwiftMain Menu Ribbon BafFigure 1).

Home Page Tools	View Estimating Lists	Templates Settings F	Reports Help Plugin:	s Search Undo
New Open Print Email Job	Pa	it Zoom Zoom Pan ge In Out	Scale Dimension	Area Linear Segment
Pages, Bookmarks Figure 1	Navigate	200m / Pan	ideos	Read Our

2. Click on the Tools Manageron the Plugins Ribbon-bar Men(Figure 2).

Figure 2

3. In the **Tools Manager** window (Figure 3), click on the green plus (+). Note that the yellow folder icon allows a new folder to be created for storage of plugins. The blue gear icon allows the viewing of properties for any plugin in the window below. The red "X" icon allows for a plugin to be deleted. The green triangular icon allows a selected plugin to be executed.

🥏 Tools Manager	×
□ + 🕸 🛛 🕨	
Search:	
Name	
· · · · · · · · · · · · · · · · · · ·	
Figure 3	

4. In the Properties - [New Plugin] window, give the new plugin a name, such as "Stucco" or whatever name you choose for the new plugin (Figure 4).

Icon		argeIcon	_
*	k	-	
Plugin Type Executable	142	0	
Executable			
Parameters			
			•••
🗌 OnStartup			
🗌 On Image Menu			
🗌 On Ribbon Bar			

Figure 4

5. Select the Plugin Type by clicking on the **Plugin Type** selection arrow and then selecting **Script Code** from the drop-down menu (Figure 5).

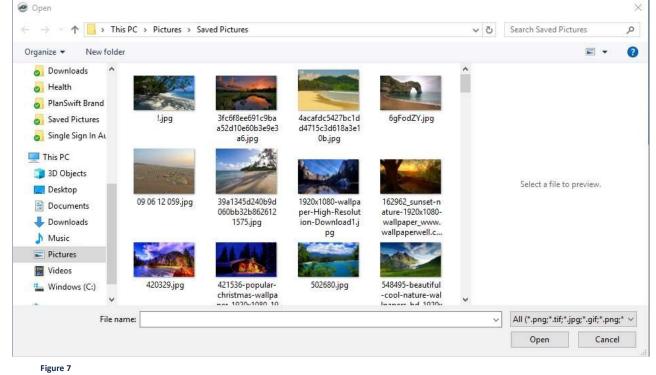
Nam	e			
Stud	cco			Felice
Icon		LargeIcor		
Exe She Exe Scri	in Type cutable il Execute cutable pi Code			
	OnStartup On Image Menu On Ribbon Bar			
out	Advanced Form	4	Ok	Cance

- Figure 5
- 6. The Plugin Type is now displayed as Script Code (arrow 1 in Figure 6). Doubleclicking on the Icon (arrow 1) or the LargeIcon (arrow 2) allows the selection of a small Icon or a LargeIcon. Double-click on the LargeIcon (arrow 3).

Name		
Stucco		
Icon	2 LargeIcon	3
Plugin Type Script Code	1	
Script Code		
		Edit
OnStartup		
🗌 On Image Menu		
🗌 On Ribbon Bar		
		21
		Ok Can

Figure 6

7. This opens an Explorer window (Figure 7) allowing the selection of an icon. From here, it is possible to navigate to a different directory and select an icon, then click on **Open** to use that selected icon; but, for now, select Cancel to retain the default green puzzle-piece icon.



8. The next step is to select where you want the new icon to appear: OnStartup, On Image Menu, or On Ribbon Bar. For this exercise, select On Ribbon Bar (arrow 1), select Tools from the Ribbon Tab drop-down (arrow 2), select Takeoff Item from the Ribbon Group drop-down (arrow 3), and then click on Ok.

e				
:co				
		LargeIcon		
	•		÷	
				15
ot Code				*
t Code				
OnStartup	4		2	3
	1		Ĩ	1
On Image Menu				
On Image Menu	Ribbon	Tab	Ribbon Group	1
On Image Menu On Ribbon Bar	Ribbon Tools	Tab	Ribbon Group Takeoff Item	Ż
		Tab		j
	in Type pt Code ot Code	in Type pt Code ot Code	in Type pt Code	LargeIcon

Figure 8

9. Click on the Tools tab on the Main Ribbon-bar, and you will see the new "Stucco" plugin displayed in the Takeoff Item group (Figure 9).

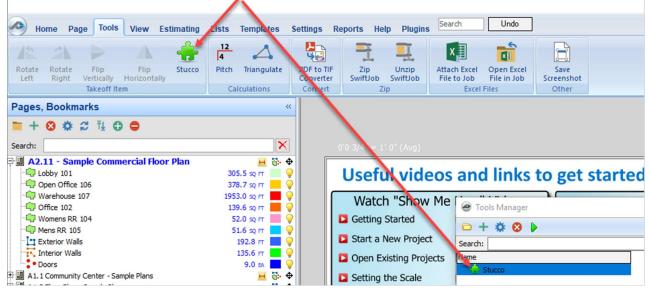


Figure 9

10. Double-click on the Stucco plugin in the Tools Manager window to re-open the Properties - [Stucco] window (Figure 10). Click on Edit.

Name				
Stucco				••
Icon	1	argeIcon		
٠			÷	
Plugin Type				
Script Code				•
Script Code				15
□ OnStartup				
On Image Menu			Ribbon Group	
🗌 On Image Menu	Ribbon Tab			
	Ribbon Tab Tools		Takeoff Item	•
🗌 On Image Menu	Contraction of the second	•	Takeoff Item	•

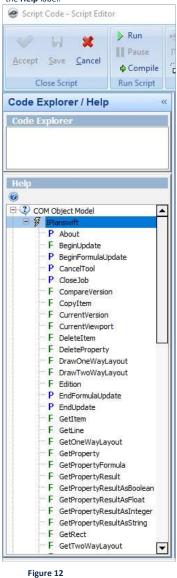
Figure 10

11. This opens the Script Code - Script Editor window (Figure 11). The large Script Editor window on the right is where plugin code is written. The Code Explorer window will list variables and procedures as they are coded into the Script Editor window. The Help window contains the COM Object Model and Scripting selections available to use in the Script Editor window. Clicking on the "+" symbol opens the folder and subfolders. Click on the "+" left of COM Object Model, then click on the "+" left of the IPlanswift folder.

Script Code - Script Editor	9 <u>730</u>	×
Image: Serie pt Image: Serie pt		
Code Explorer / Help « ¹		
Itop: Ø Ø Ø Stopper Model B: Q: Stopping		
Watch Window Debug Output		

Figure 11

12. The COM Object Model selections are now available in the Help window (Figure 12). Click on Current Version, then click on the circled question mark directly below the Help label.



13. Clicking on CurrentVersion (arrow 1 of Figure 13), then clicking on the question mark (arrow 2), opens the CurrentVersion window (arrow 3), which provides the Declaration form of the item selected and a Source Code Example. Source code may be copied and pasted into the Script Editor window and then modified as needed, or the CurrentVersion selection can be double-clicked on and will appear in the Script Editor window at the cursor's last position. The Close Script, Run Script, Debug Script, Break Points, and Script Parameter sections provide the code editing functions useful in programming API's.

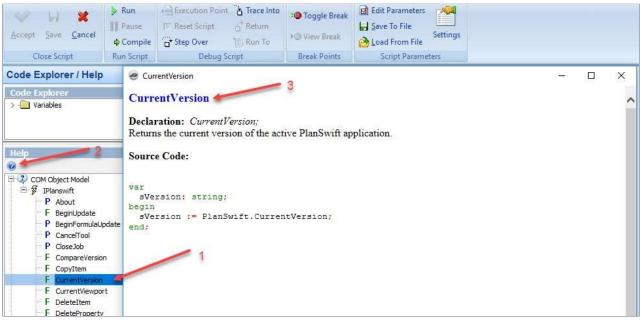


Figure 13

Scripting - Functions

Scripting - Functions

Forms

- NewRadioButton
- NewForm
- NewButton
- NewEdit
- NewComboBox
- NewColorBox
- NewCheckBox
- NewLabel

Math Functions

- DecToEnglish
- DistanceBetweenPointsScaled
- DistanceFromLine
- ExtendLine
- GetDistanceFromLine
- GetIntersectPoint
- GetPI
- Max
- Min
- RoundDown
- DistanceBetweenPoints
- RoundToNearest
- RoundUp
- AngleBetweenPointsUnScaled
- ParallelLine
- Pi
- PointOnAngle
- TrimToArea

Update Methods

- BeginUpdate (2)
- BeginFormulaUpdate (2)
- CurrentVersion (2)
- EndFormulaUpdate (2)
- EndUpdate (2)
- ImageRefresh
- RefreshImage
- NewChangeGroup (2)
- PostChanges (2)

Windows Controls

- CurrentViewport (2)
- FindWindow
- FocusMainWindow
- FocusWindow
- SendKey
- SendKeys

User Input

- GetIntersectPoint (2)
- GetLine (2)
- GetPoint (2)
- GetRect (2)
- HitTest
- NewPoint (3)
- ResultPointX

- ResultPointY
- •
- •

ltem s

- •
- •
- •

- • •

- •
- •
- •
- •
- •
- •
- •
- •
- •
- •

- •
- •
- •
- •
- •
- •
- •

- •
- •
- •
- •
- • •

- •
- •
- •

Misc

- • • •

- •

ResultPointX2 ResultPointY2

BringToFront ChildCount (2) ChildItem (2) ClearSelection Copyltem (2) DeleteItem GetItem (2) GetSelectionList IsType ListData MoveltemTo NewItem (3) ParentItemGUID ParentItem (2) SelectedItem (2) ShowLabel StartRecording SelectedPage (2) SendToBack SelectedPageGUID SelectItem ZoomToItem

Sections

AddPoint DeletePoint DrawOneWayLayout (2) DrawTwoWayLayout (2) GetOneWayLayout (2) GetTwoWayLayout (2) NewSection (3) NewSubtractSection PointX PointY PointCount (3) SetPoint (3)

Properties

DeleteProperty GetGUIDfromPath GetDobTotal (2) GetPropertyCount GetPropertyFormula GetPropertyResult GetResultAsBoolean GetResultAsInteger GetResultAsInteger GetPropertyName GetPropertyAttribute SetPropertyAttribute SetPropertyAttribute

BeepAcknowledged CompareVersion (2) CurrentUser ExecuteScript (3) GetActionLog GetPairedValue

GetResult

- IsUnlocked (2)
- KeyDown
- KeyUp

- License
 OpenJob (2)
 NewBlankPage (2)
 SetRecordMode
- SetImagePropertyFromFile
 SetEncrypted
- RGB
- ReapplyAllOfType • OpenJobEx (2)
- SetResult
- StopRecording

- Dialogs EditScriptProperty EditItem

 - Custom Dialogs
 - MessageDialog
 - My Color Dialog
 - ScriptMessageDialog
 - SelectItemDialog (2

Global Variables and Constant s

Forms

Forms

NewRadioButto n		
NewForm		
NewLabel		
NewCheckBo x		
NewComboBo x		
NewButto n		
NewColorBo x		
NewEdit		

NewButton

NewButton

Creates a new TButton and sets the Left, Top, Caption and ModalResult Properties as specified. Do not attempt to destroy or free a TButton created with NewButton.

Syntax:

Procedure: NewButton(Left, Top: Integer; Caption: String; Modalresult: Integer): TButton;

Code Reference:

Scripting

1. Navigate to Plugin Store->Tool Manager and create a new Plugin

2. Set the plugin type to Script Code and open the $\ensuremath{\mathsf{Editor}}$

3. Copy Code into the editor

4. Press run

API Call:

NewCheckBox

NewCheckBox

Creates a new TCheckBox and sets the Left, Top, Caption and Checked Properties as specified. Do not attempt to destroy or free a TCheckBox created withNewCheckBox.

Syntax:

Procedure: NewCheckBox(Left, Top: Integer; Caption: String; Checked: Boolean): TCheckBox;

Code Reference:

Scripting

1. Navigate to Plugin Store->Tool Manager and create a new Plugin

2. Set the plugin type to Script Code and open the Editor

3. Copy Code into the editor

4. Press run

API Call:

NewColorBox

NewColorBox

Creates a new TColorBox and sets the Left, Top and Selected Properties as specified. Do not attempt to destroy or free a TColorBox created with NewColorBox.

Syntax:

Procedure: NewColorBox(Left, Top: Integer; Selected: Integer): TColorBox; Code Reference: 1. Navigate to Plugin Store->Tool Manager and create a new Plugin

- 2. Set the plugin type to Script Code and open the Editor
- 3. Copy Code into the editor
- 4. Press run

API Call:

Scripting

NewComboBox

NewComboBox

Creates a new TComboBox and sets the Left, Top and Text Properties as specified. Do not attempt to destroy or free a TComboBox created withNewComboBox.

Syntax:

Procedure: NewComboBox(Left, Top: Integer; Text: String): TComboBox; Code Reference: 1. Navigate to Plugin Store->Tool Manager and create a new Plugin

- 2. Set the plugin type to Script Code and open the $\ensuremath{\mathsf{Editor}}$
- 3. Copy Code into the editor
- 4. Press run

API Call:

Scripting

NewEdit

NewEdit

Creates a new TEdit and sets the Left, Top and Text Properties as specified. Do not attempt to destroy or free a TEdit created with NewEdit.

Syntax:

Procedure: NewEdit(Left, Top: Integer; Text: String): TEdit; Code Reference: 1. Navigate to Plugin Store->Tool Manager and create a new Plugin

- 2. Set the plugin type to Script Code and open the Editor
- 3. Copy Code into the editor
- 4. Press run

API Call:

Scripting

NewForm

NewForm

Creates a new TForm object and sets the width, height, and caption as specified. Do not attempt to destroy or free forms created with NewForm.

Syntax:

Procedure: NewForm(Width, Height: Integer; Caption: String): TForm; Code Reference: 1. Navigate to Plugin Store->Tool Manager and create a new Plugin

- 2. Set the plugin type to Script Code and open the Editor
- 3. Copy Code into the editor
- 4. Press run

API Call:

Scripting

NewLabel

NewLabel

Creates and returns a new TLabel object and sets the Left, Top and Caption properties. Do not attempt to destroy or free labels created with NewLabel.Declaration:

Syntax:

Procedure: NewLabel(Left, Top: Integer; Caption: String): TLabel;

API Call:

Scripting

NewRadioButton

NewRadioButton

Creates a new TRadioButton and sets the Left, Top, Caption and Checked Properties as specified. Do not attempt to destroy or free a TRadioButton created withNewRadioButton.

Syntax:

Procedure: NewRadioButton(Left, Top: Integer; Caption: String; Checked: Boolean): TRadioButton;

Scripting

Code Reference:

- Navigate to Plugin Store->Tool Manager and create a new Plugin
 Set the plugin type to Script Code and open the Editor
- 3. Copy Code into the editor
- 4. Press run

API Call:

Math Functions

Math Functions

DecToEnglis h
DistanceBetweenPointsScale d
DistanceFromLin e
ExtendLin e
GetDistanceFromLin e
GetIntersectPoin t
GetPI
Max
Min
RoundDow n
DistanceBetweenPoint s
RoundToNeares t
RoundU p
AngleBetweenPointsUnScale d
ParallelLin e
Pi
PointOnAngl e
TrimToAre a

DecToEnglish

DecToEnglish

Converts a given dimension into its string representation.

Syntax:

Procedure: DecToEnglish(Feet: Double): String;

API Call:

Scripting

DistanceBetweenPointsScaled

DistanceBetweenPointsScaled

Returns the angle between 2 points given by p1 and p2 coordinates (based on scale factor for the page).

Syntax:

Procedure: DistanceBetweenPointsScaled(p1X, p1Y, p2X, p2Y: Double): Double;

API Call:

Scripting

DistanceFromLine

DistanceFromLine

Returns the distance of a point given by *p3* is from a line, given by *p1* and *p2*.

Syntax:

Procedure: DistanceFromLine(p1x, p1y, p2x, p2y, p3x, p3y: Double): Double;

API Call:

Scripting

ExtendLine

ExtendLine

Calculates the points to extend a line given by *p1* and *p2* a given Distance, then returns the new points in variables *p3* and *p4*.

c.		• ~		
31	/[1	١d	x	а.

Procedure: ExtendLine(p1x, p1y, p2x, p2y, Distance: Double; var p3x: Double; var p3y: Double; var p4x: Double; var p4y: Double);

API Call:		
Scripting	Expand source	ce

GetDistanceFromLine

GetDistanceFromLine

Coming soon

Syntax: Procedure: GetDistanceFromLine

API Call:

Scripting

GetIntersectPoint

GetIntersectPoint

Calculates at what point Line 1, given by p1 and p2, intersects with Line 2, given by p3 and p4, and returns the result point in p5. Returns 1 (True) if the lines intersect

or 0 (False) if the lines are parallel.

Syntax:

Procedure: GetIntersectPoint(p1x, p1y, p2x, p2y, p3x, p3y, p4x, p4y: Double; var p5x: Double; var p5y: Double): Integer;

API Call:

Scripting

GetPI

GetPI

Gets Pi.		
Syntax: Procedure:		
GetPI API Call:		
Scripting		> Expand source

Max

Max

Returns the larger of the values passed.

Syntax:

Procedure: Max(Value1, Value2: Double): Double;

API Call:

Scripting

Min

Min

Returns the smaller of the values passed.

Syntax:

Procedure: Min(Value1, Value2: Double): Double;

API Call:

Scripting

RoundDown

RoundDown

Rounds the given value down to the nearest whole number.

Syntax:

Procedure: RoundDown(Val: Double): Integer;

API Call:

Scripting

DistanceBetweenPoints

DistanceBetweenPoints

Returns the distance between 2 points specified by p1 and p2 coordinates.

Syntax:

Procedure: DistanceBetweenPoints(p1X, p1Y, p2X, p2Y: Double): Double;

API Call:

Scripting

RoundToNearest

RoundToNearest

Rounds the given value down to the nearest value as defined by precision.

Syntax:

Procedure: RoundToNearest(Val, Precision: Double): Double;

API Call:

Scripting

RoundUp

RoundUp

Rounds the given value up to the nearest integer.

Syntax:

Procedure: RoundUp(Val: Double): Integer

API Call:

Scripting

AngleBetweenPointsUnScaled

AngleBetweenPointsUnScaled

Returns the angle between 2 points given by p1 and p2 coordinates.

Syntax:

Procedure: AngleBetweenPointsUnScaled(p1X, p1Y, p2X, p2Y: Double): Double;

API Call:

Scripting

ParallelLine

ParallelLine

Calculates the points to form a new line parallel line offset Distance from the original, specified by p1 and p2, then returns the new points in variables p3 and p4.

Syntax:

Procedure: ParallelLine(p1x, p1y, p2x, p2y, Distance: Double; var p3x: Double; var p3y: Double; var p4y: Double);

API Call:

Scripting

Pi

Pi

Returnsthe numeric value for Pi (3.1415926535897932384626433832795).

Syntax:

Procedure: Pi: Double;

API Call:

Scripting

PointOnAngle

PointOnAngle

Calculates a new point given by *p1* a given *Distance and Angle* then returns the result point in *p2*.

Syntax:

Procedure: PointOnAngle(p1x, p1y, Angle, Distance: Double; var p2x: double; var p2y: double);

API Call:

Scripting

TrimToArea

TrimToArea

Trims the ends of a Segment object to the boundaries of the given Area object.

Syntax:

Procedure: TrimToArea(AreaPath, SegmentPath: String);

API Call:

Scripting

Update Methods

Update Methods

BeginUpdate (2)ReginFormulaUpdate (2)CurrentVersion (2)EndFormulaUpdate (2)IndUpdate (2)Refers hRefreshImag eNewChangeGroup (2)PostChanges (2)

BeginUpdate (2)

BeginUpdate

Temporarily suspends program updates.

Syntax: Procedure: BeginUpdate;

API Call:

Scripting

BeginFormulaUpdate (2)

BeginFormulaUpdate

Temporarily suspends automatic property calculations.

Syntax:

Procedure: BeginFormulaUpdate;

API Call:

Scripting

CurrentVersion (2) currentVersion

Returns the current versions of PlanSwift.

Syntax:

Procedure: CurrentVersion: String;

API Call:

Scripting

EndFormulaUpdate (2)

EndFormulaUpdate

Ends the temporary suspension of automatic property calculations.

Syntax:

Procedure: EndFormulaUpdate;

API Call:

Scripting

EndUpdate (2)

EndUpdate

Ends the temporary suspension of program updates.

Syntax:

Procedure: EndUpdate;

API Call:

Scripting

ImageRefresh

ImageRefresh

Refreshes the current screen image. Same as RefreshImage.

Syntax:

Procedure: ImageRefresh;

API Call:

Scripting

RefreshImage

RefreshImage

Refreshes the current screen image. Same asImageRefresh.

Syntax:

Procedure: RefreshImage

API Call:

Scripting

NewChangeGroup (2)

NewChangeGroup

Creates a new program change group.

Syntax:

Procedure: NewChangeGroup(AName: String);

API Call:

Scripting

PostChanges (2)

PostChanges

Post all opened change groups to the program.

Syntax: Procedure: PostChanges;

API Call:

Scripting

Windows Controls

Windows Controls

CurrentViewport (2) FindWindo w FocusMainWindo w FocusWindo w SendKe y SendKey s

CurrentViewport (2)

CurrentViewport

In PlanSwift, the documentation window shows "BeginForumulaUpdate", not CurrentViewPort. ???? Coming soon

Syntax:

Procedure: Coming soon

API Call:

Scripting

FindWindow

FindWindow

Finds a window based on the given criteria. Returns the window handle if successful or 0 if the window is not found.

Syntax:

Procedure: FindWindow(StartsWith, Contains, Excludes: String; Exact: Boolean): Integer; Contains, Excludes, and Exact are optional.

API Call: Scripting

FocusMainWindow

FocusMainWindow

Put user focus back on application main window.

Syntax: Procedure:

FocusMainWindow;

API Call:

Scripting

FocusWindow

FocusWindow

Put user focus back on application main window.

Syntax:

Procedure: FocusWindow(Hwnd: Integer);

API Call:

Scripting

SendKey

SendKey

Sends the given KeyCode to the active PlanSwift control. (Same as TypeKey).

Syntax:

Procedure: SendKey(AKey: Integer);

API Call:

Scripting

SendKeys

SendKeys

Sends a string of keystrokes to the active PlanSwift control.

Syntax:

Procedure: SendKeys(AKeys: String);

API Call:

Scripting

User Input

User Input

GetIntersectPoint (2)		
GetLine (2)		
GetPoint (2)		
GetRect (2)		
HitTes t		
NewPoint (3)		
ResultPoint X		
ResultPoint Y		
ResultPointX 2		
ResultPointY 2		

GetIntersectPoint (2)

GetIntersectPoint

Coming soon.
Syntax: Procedure:
GetIntersectPoint
API Call:
Scripting
Expand source

GetLine (2)

GetLine

Prompts the user to click 2 points on the activeplan to define a line then returns the coordinates in *p1* and *p2*. Returns 1 if the function is successful or 0 if the user cancels.

Syntax:

Procedure: GetLine(Var p1x: double; Var p1y: double; Var p2x: double; Var p2y: double; Hint: String): Integer;

Scripting API Call:

GetPoint (2)

GetPoint

GetPoint prompts the user to select a point by clicking on the activeplan, then returns the point coordinates in X and Y. If the user clicks a valid point, the result is 1 (True); otherwise, the result is 0 (False).

Syntax:

Procedure: GetPoint(Var X: Double; Var Y: Double; Hint: String): Integer;

Scripting API Call:

GetRect (2)

GetRect

Prompts the user to click 2 points on the active plan to define a rectangle, then returns the coordinates in *p1* and *p2*. Returns 1 if the function is successful or 0 if the user cancels.

Syntax:

Procedure: GetRect(Var p1x: double; Var p1y: double; Var p2x: double; Var p2y: double; Hint: String): Integer;

Scripting API Call:

HitTest

HitTest

Coming soon

Syntax: Procedure:

HitTest API Call:

Scripting

NewPoint (3)

NewPoint

Adds a new point to ItemPath at the X, Y coordinates. If the Item is not found or is not a drawing object, this procedure will be ignored.

Syntax:

Procedure: NewPoint(ItemPath: String; X, Y: Double);

API Call:

Scripting

ResultPointX

ResultPointX

Returns the x coordinate from the last *Getpoint, Getline or GetRect*.

Syntax:

Procedure: ResultPointX: Double;

API Call:

Scripting

ResultPointY

ResultPointY

Returns the y coordinate from the last *Getpoint, Getline or GetRect*.

Syntax:

Procedure: ResultPointY: Double;

API Call:

Scripting

ResultPointX2

ResultPointX2

Returns the x2 coordinate from the last *Getline or GetRect*.

Syntax:

Procedure: ResultPointX2: Double;

API Call:

Scripting

ResultPointY2

ResultPointY2

Returns the y2 coordinate from the last *Getline or GetRect*.

Syntax:

Procedure: ResultPointY2: Double;

API Call:

Scripting

Items

Items

BringToFron t	
ChildCount (2)	
ChildItem (2)	
ClearSelectio n	
Copyltem (2)	
Deletelte m	
Getltem (2)	
GetSelectionLis t	
IsType	
ListData	
MoveltemTo	
NewItem (3)	
ParentitemGUI D	
Parentitem (2)	
SelectedItem (2)	
ShowLabel	
StartRecordin g	
SelectedPage (2)	
SendToBac k	
SelectedPageGUI D	

BringToFront

BringToFront

Brings the given item to the forefront, above all other items.

Syntax:

Procedure: BringToFront(ItemPath: String);

API Call:

Scripting

ChildCount (2)

ChildCount

Returns the number of child items for the item.

Syntax:

Procedure: ChildCount(ItemPath: String): Integer;

API Call:

Scripting

ChildItem (2)

ChildItem

Returns the full path of the child item at position Index in the list. If the child item does not exist, an empty string is returned.

Syntax:

Procedure: ChildItem(ItemPath: String; Index: Integer): String;

API Call:

Scripting

ClearSelection

ClearSelection

Un-selects all currently selected items.

Syntax: Procedure:

ClearSelection;

API Call:

Scripting

Copyltem (2)

CopyItem

Creates a copy of Item under Parent and returns the ID of the new item. If IncludeChildren is true, child items will be copied also. If SkipSections is true, digitized sections

will be duplicated also.

Syntax:

Procedure: CopyItem(Item: String; Parent: String; IncludeChildren: boolean; SkipSections: boolean): String;

Scripting API Call:

DeleteItem

Deleteltem

Deletes the given item from the system. Returns 1 (True) is successful, otherwise 0 (False).

Syntax:

Procedure: DeleteItem(ItemPath: String): Integer;

API Call:

Scripting

GetItem (2)

GetItem

Returns the item given by FullPath. Returns Nil if the object is not found.

Syntax:

Procedure: GetItem(sItemPath: string): string;

API Call:

Scripting

GetSelectionList

GetSelectionList

Returns a list of order list of GUIDs assigned to sections.

Syntax:

Procedure: GetSelectionList;

API Call:

Scripting

IsType

IsType

Returns 1 (True) if the item is of type given, otherwise returns 0 (False).

Syntax:

Procedure: IsType(ItemPath, Type: String): Integer;

API Call:

Scripting

ListData

ListData

Coming soon

Syntax: Procedure:

ListData; API Call:

Scripting

MoveltemTo

MoveltemTo

Returns True if the given item is successfully moved to a new parent item. MoveAction is optional, can be Above, Below or IntoTop; otherwise will default to IntoBottom.

Syntax:

Procedure: MoveItemTo(ItemPath, NewParent, MoveAction: String): Boolean;

Scripting

API Call:

NewItem (3)

NewItem

Creates a new child item for the given item. ItemType is optional and allows you to set the type of item to create. Name is optional and sets the name for the new child item.

Syntax:

Procedure: NewItem(ItemPath, ItemType, Name: String): String;

API Call:

Scripting

ParentItemGUID

ParentItemGUID

Returns the GUID (globally unique identifier) of the Parent Item for the given item.

Syntax:

Procedure: ParentItemGUID(ItemPath: String): String;

API Call:

Scripting

ParentItem (2)

ParentItem

Returns the parent item for the given item. If the function fails an empty string is returned.

Syntax:

Procedure: ParentItem(ItemPath: String): String;

API Call:

Scripting

SelectedItem (2)

SelectedItem

Returns the full path to the currently selected item. If no item is selected an empty string is returned.

Syntax:

Procedure: SelectedItem: String;

API Call:

Scripting

ShowLabel

ShowLabel

Sets the visibility of an item's label.

Syntax:

Procedure: ShowLabel(ItemPath: String; Visible: Boolean);

API Call:

Scripting

StartRecording

StartRecording

ItemPath is optional. If provided, ItemPath must be a digitizer object. If <itempath< i=""> is omitted, Planswift will attempt to record the currently selected item, if any. Returns 1 (True) if successful, otherwise returns 0 (False).

Syntax:

Procedure: StartRecording(ItemPath: String): Integer;

API Call:

Scripting

SelectedPage (2)

SelectedPage

Returns the full path to the currently selected page. If no page is selected, an empty string is returned.

Syntax:

Procedure: SelectedPage: String;

API Call:

Scripting

SendToBack

SendToBack

Sends the given item to the back, behind all other items.

Syntax:

Procedure: SendToBack(ItemPath: String);

API Call:

Scripting

SelectedPageGUID

SelectedPageGUID

Returns the GUID for the currently selected page. If no page is selected returns an empty string.

Syntax:

Procedure: SelectedPageGUID: String;

API Call:

Scripting

SelectItem

SelectItem

Set the given items selected status to Selected. Returns True if successful, False if the operation failed.

Syntax:

Procedure: SelectItem(ItemPathorGUID: string; Selected: boolean): Boolean;

API Call:

Scripting

ZoomToltem

ZoomToltem

Redisplays the current view to a selected takeoff item on the page. A value can be assigned to allow for user sizeable margins around the viewed object (default = 30).

Syntax:

Procedure: ZoomToltem(sltemPath: string; Marginsize: string); integer;

API Call:

Scripting

Sections

Sections

AddPoint		
DeletePoin t		
DrawOneWayLayout (2)		
DrawTwoWayLayout (2)		
GetOneWayLayou t		
GetTwoWayLayout (2)		
NewSection (3)		
NewSubtractSectio n		
PointX		
PointY		
PointCount (3)		
SetPoint (3)		

AddPoint

AddPoint

Adds a new point given by X, Y to the item. ItemPath must specify an existing digitizer object or the procedure fails.

Syntax:

Procedure: AddPoint(ItemPath: String; X, Y: Double);

API Call:

Scripting

DeletePoint

DeletePoint

Deletes the point at position Index.

Syntax:

Procedure: DeletePoint(ItemPath: String; Index: Integer);

API Call:

Scripting

DrawOneWayLayout (2)

DrawOneWayLayout

Function used to perform segment layouts at a specified span, horizontal run, as well as spacing.

Syntax:

Procedure: DrawOneWayLayout(altem: string; oSpanPnt1X; oSpanPnt1Y, oSpanPnt2X, oSpanPnt2Y, oRunPnt1X, oRunPnt1Y, oRunPnt2Y: double; blncludeFirst, blncludeLast: boolean; nSpacing: double; aAreaSection: string): boolean;

Arguments: Altem: String

Specifies the area section to assign the layout segments to.

SpanLineX and SpanLineY: *Double* Direction span start and endpoint.

RunLineX and RunLineY: *Double* Horizontal (side to side) run direction of area to populate. Requires a start and endpoint;

bIncludeFirst: Boolean

Specifies whether to include a segment at the "start" run point. Even if it does not fall within the spacing range.

bincludeLast: Boolean

Specifies whether to include a segment at the "last" run point. Even if it does not fall within the spacing range.

nSpacing: Double Specifies the "run" spacing used when laying out segment objects.

AArea: String (optional parameter)

Specifies a defined "Area Segment" to trim/extend laid segments to. Supply either the path or GUID (globally unique identifier) to the area section. Or, empty double-quotes for no trim/extending required.

API Call:

DrawTwoWayLayout (2)

DrawTwoWayLayout

Function used to perform segment layouts (in 2 directions) at a specified span, horizontal run, as well as spacing.

Syntax:

Procedure: DrawTwoWayLayout(altem: string; oSpanPnt1X; oSpanPnt1Y, oSpanPnt2X, oSpanPnt2Y, oRunPnt1X, oRunPnt1Y, oRunPnt2Y; double; blncludeFirst, blncludeLast: boolean; nSpacing: double; aAreaSection: string): boolean;

Arguments:

Altem: String

Specifies the area section to assign the layout segments to.

SpanLineX and SpanLineY: Double Direction span start and endpoint.

RunLineX and RunLineY: Double

Horizontal (side to side) run direction of area to populate. Requires a start and endpoint;

bincludeFirst: Boolean

Specifies whether to include a segment at the "start" run point. Even if it does not fall within the spacing range.

bincludeLast: Boolean

Specifies whether to include a segment at the "last" run point. Even if it does not fall within the spacing range.

Scripting

nSpacing: Double

Specifies the "run" spacing used when laying out segment objects.

AArea: String (optional parameter)

Specifies a defined "Area Segment" to trim/extend laid segments to. Supply either the path or GUID to the area section. Or, empty double-quotes for no trim/extending required.

API Call: GetOneWayLayout

GetOneWayLayout

Function used to perform segment layouts at a specified span, horizontal run, as well as spacing.

Syntax:

Procedure: GetOneWayLayout(altem: string; sSpanHint, sRunHint: string; blncludeFirst, blncludeLast: boolean; nSpacing: double; aAreaSection: string): boolean;

Arguments:

Altem: String Specifies the area section to assign the layout segments to.

sSpanHint: String

Hint message displayed on mouse cursor indicating to pick the "span" direction.

sRunHint: String

Hint message displayed on mouse cursor indicating to pick the "run" direction.

bIncludeFirst: Boolean

Specifies whether to include a segment at the "start" run point. Even if it does not fall within the spacing range.

Scripting

bincludeLast: Boolean

Specifies whether to include a segment at the "last" run point. Even if it does not fall within the spacing range.

nSpacing: Double

Specifies the "run" spacing used when laying out segment objects.

AArea: String (optional parameter)

Specifies a defined "Area Segment" to trim/extend laid segments to. Supply either the path or GUID to the area section. Or, empty double-quotes for no trim/extending required.

API Call: GetTwoWayLayout (2)

GetTwoWayLayout

Function used to perform segment layouts (in 2 directions) at a specified span, horizontal run, as well as spacing.

Syntax:

Procedure: GetTwoWayLayout(altem: string; sSpanHint, sRunHint: string; blncludeFirst, blncludeLast: boolean; nSpacing: double; aAreaSection: string): boolean;

Arguments: Altem: String

Specifies the area section to assign the layout segments to.

sSpanHint: String

Hint message displayed on mouse cursor indicating to pick the "span" direction.

sRunHint: String

Hint message displayed on mouse cursor indicating to pick the "run" direction.

Scripting

bincludeFirst: Boolean

Specifies whether to include a segment at the "start" run point. Even if it does not fall within the spacing range.

bincludeLast: Boolean

Specifies whether to include a segment at the "last" run point. Even if it does not fall within the spacing range.

nSpacing: Double

Specifies the "run" spacing used when laying out segment objects.

AArea: String (optional parameter)

Specifies a defined "Area Segment" to trim/extend laid segments to. Supply either the path or GUID to the area section. Or, empty double-quotes for no trim/extending required.

API Call:

NewSection (3)

NewSection

Adds a new section to a digitized type item and returns the full path to the new section. If *ParentPath* does not exist, or is not a digitzer item, this function fails and returns an empty string. *SectionName* is optional.

Syntax:

Procedure: NewSection(ParentPath, SectionName: String): String;

API Call:

Scripting

NewSubtractSection

NewSubtractSection

Subtracts a new section from a digitized type item and returns the full path to the new section. If *ParentPath* does not exist or is not a digitzer item, this function fails and returns an empty string. *SectionName* is optional.

Syntax:

Procedure: NewSubtractSection;

API Call:

Scripting

PointX

PointX

Returns the X coordinate of the point given by Index. If this function fails the return value is -1.

Syntax:

Procedure: PointX(ItemPath: String; Index: Integer): Double;

API Call:

Scripting

PointY

PointY

Returns the Y coordinate of the point given by Index. If this function fails the return value is -1.

Syntax:

Procedure: PointY(ItemPath: String; Index: Integer): Double;

API Call:

Scripting

PointCount (3)

PointCount

Returns the number of points recorded for the section.

Syntax:

Procedure: PointCount(ItemPath: String): Integer;

API Call:

Scripting

SetPoint (3)

SetPoint

Sets the X, Y coordinates of the given point.

Syntax:

Procedure: SetPoint(ItemPath: String; Index: Integer; X, Y: Double);

API Call:

Scripting

Properties

Properties

DeletePropert y	
GetGUIDfromPat h	
GetJobTotal (2)	
GetPropertyCoun t	
GetPropertyFormul a	
GetPropertyResul t	
GetResultAsBoolea n	
GetResultAsFloa t	
GetResultAsIntege r	
GetResultAsStrin g	
GetPropertyNam e	
GetPropertyAttribut e	
GetPropertyAttributeLis t	
SetPropertyAttribut e	
SetPropertyFormula (3)	
DeleteProperty	

DeleteProperty

Deletes a property from the item.

Syntax:

Procedure: DeleteProperty(ItemPath, PropertyName: String);

API Call:

Scripting	> Expand source
-----------	-----------------

GetGUIDfromPath

GetGUIDfromPath

Returns the GUID for the item based upon the path for the item.

Syntax:

Procedure: GetGUIDfromPath;

API Call:

Scripting

GetJobTotal (2)

GetJobTotal

Retrieves the total number of items of a certain type in the entire opened job.

Syntax:

Procedure: GetJobTotal(sPropertyname: string; sItemType: string = "): Double;

API Call:	
Scripting	> Expand source

GetPropertyCount

GetPropertyCount

Returns the number of properties for the given item.

Syntax:

Procedure: GetPropertyCount(ItemPath: String): Integer;

API Call:		
Scripting		> Expand source

GetPropertyFormula

GetPropertyFormula

Returns the formula for the given property.

Syntax:

Procedure: GetPropertyFormula(ItemPath, PropertyName: String): String;

API Call:	
Scripting	Expand source

GetPropertyResult

GetPropertyResult

Returns the calculated result of the given property as a variant.

Syntax:

Procedure: GetPropertyResult(ItemPath, PropertyName: String): Variant;

API Call:	
Scripting	Expand source

GetResultAsBoolean

GetResultAsBoolean

Returns the calculated result of the given property.

Syntax:

Procedure: GetResultAsBoolean(ItemPath, PropertyName: String; Default: Boolean): Boolean;

API Call:	
Scripting	Expand source

Default is an optional return value for the function in case of failure. If Default

GetResultAsFloat

GetResultAsFloat

Returns the calculated result of the given property.

Syntax:

Procedure: GetResultAsBoolean(ItemPath, PropertyName: String; Default: Double): Double;

API Call:

Scripting

> Expand source

is not provided it defaults to 0.

GetResultAsInteger

GetResultAsInteger

Returns the calculated result of the given property.

is not provided it defaults to 0.

Syntax:

Procedure: GetResultAsInteger(ItemPath, PropertyName: String; Default: Integer): Integer;

Default is an optional return value for the function in case of failure. If Default

API Call:

Scripting

Default is an optional return value for the function in case of failure. If Default

GetResultAsString

GetResultAsString

Returns the calculated result of the given property. is not provided it defaults to an empty string.

Syntax:

Procedure: GetResultAsString(ItemPath, PropertyName: String; Default: String): String;

API Call:

Scripting

GetPropertyName

GetPropertyName

Returns the name of the *nth* property in the property list.

Syntax:

Procedure: GetPropertyName(ItemPath: String; Index: Integer): String;

API Call:

Scripting

GetPropertyAttribute

GetPropertyAttribute

Returns the value of the given Item Property Attribute.

Syntax:

Procedure: GetPropertyAttribute(Itempath, PropertyName, AttributeName: String): String;

API Call:

Scripting

GetPropertyAttributeList

GetPropertyAttributeList

Returns the value of the given Item Property Attribute.

Syntax:

Procedure: GetPropertyAttributeList(Itempath, PropertyName: String): String;

API Call:

Scripting

SetPropertyAttribute

SetPropertyAttribute

Attempts to set the Item Property Attribute to the given Value.

Syntax:

Procedure: SetPropertyAttribute(ItemPath, PropertyName, AttributeName, Value: String);

API Call:

Scripting

SetPropertyFormula (3)

SetPropertyFormula

Sets the given property to the value specified if possible.

Syntax:

Procedure: SetPropertyFormula(ItemPath, PropertyName: String; Value: Variant; Type: String);

API Call:

Scripting

Misc

Misc

BeepAcknowledge d
CompareVersion (2)
CurrentUse r
ExecuteScript (3)
GetActionLo g
GetPairedValu e
GetResult
IsUnlocked (2)
KeyDown
KeyUp
License
OpenJob (2)
NewBlankPage (2)
SetRecordMod e
SetImagePropertyFromFil e
SetEncrypte d
RGB
ReapplyAllOfTyp e
OpenJobEx (2)
SetResult

BeepAcknowledged

BeepAcknowledged

Sends an instruction to the computer to trigger an audible beep.

Syntax:

Procedure: BeepAcknowledged;

API Call:

Scripting

CompareVersion (2)

CompareVersion

Compares two different versions of PlanSwift.

Syntax:

Procedure: Compare Version(arg1, arg2): integer;

API Call:

Scripting

CurrentUser

CurrentUser

Returns the username of the current user.

Syntax: Procedure: CurrentUser:	
String	
API Call:	
Scripting	Expand sour

ExecuteScript (3)

ExecuteScript

Executes the script property Property Path and returns the result as a variant. Paramx is the optional string parameters to pass to the script. Failure to pass required

parameters could lead to errors or failure. All scripts should check for invalid parameters and exit gracefully.

Syntax:

Procedure: ExecuteScript(PropertyPath, param1, param2, param3, param4, param5, param6, param7, param8, param9: String;): Variant;

API Call:

Scripting

GetActionLog

GetActionLog

Returns to a string the entire contents of the application Action Log.

Syntax:	
Procedure: GetActionLog;	
API Call:	

GetPairedValue

GetPairedValue

Passed a search string and a set of paired strings, the result will be the value assigned to the search string.

Syntax:

Procedure: GetPairedValue(sSearchStr: string; sStringSet: string): string;

API Call:

Scripting

GetResult

GetResult

Coming soon. Returns the calculated result from the given property ???.

Syntax:	
Procedure: GetResult;	
API Call:	

570

IsUnlocked (2)

IsUnlocked

Checks the product activation status of a plugin. If AllowUnlock is true the user is prompted to Activate if needed.

Syntax:	Procedure: IsUnlocked(AProduct: String; AMajorVer: Integer; AMinorVer: Integer; AllowUnlock: Boolean): Boolean;	
API Call:		
Scripting		> Expand source

KeyDown

KeyDown

Pushes a Keydown event to PlanSwift.

Syntax: Procedure:		
KeyDown;		
API Call:		
Scripting		> Expand source

KeyUp

KeyUp

Pushes a Keyup event to PlanSwift.

Syntax: Procedure:

KeyUp;

API Call:

Scripting

* Expand source

License

License

Coming soon

Syntax: Procedure:

License; API Call:

Scripting

OpenJob (2)

OpenJob

Opens the job specified by JobPath. Returns True if successful, false if the job could not be found or opened.

Syntax:

Procedure: OpenJob(JobPath: String): Boolean;

API Call:

Scripting

NewBlankPage (2)

NewBlankPage

Creates a blank page in the current job and returns the Page Item that was created.

Syntax:

Procedure: NewBlankPage(AName: string; AWidth, AHeight, ADPI: integer; AScale: string): string;

API Call:	
Scripting	Expand source

SetRecordMode

SetRecordMode

Set the digitizer record mode to either "Box" mode or "Point to Point" mode. "Box" is the only valid setting; anything else will set the mode to "Point to Point".

Syntax:

Procedure: SetRecordMode(Mode: String);

API Call:

Scripting

SetImagePropertyFromFile

SetImagePropertyFromFile

Assigns an image to a property within an item. Item is passed with a filespec parameter.

Syntax:

Procedure: SetImagePropertyFromFile(sItemPath: string; sPropertyItem: string; sFilespec: string): integer;

API Call:	
Scripting	> Expand source

SetEncrypted

SetEncrypted

Sets encryption on scripted plugins.

Syntax: Procedure:

SetEncrypted;

API Call:

Scripting

RGB

RGB

Passed RGB integers will return the "color" integer in ARGB format.

Syntax:

Procedure: RGB(nRed: integer; nGreen: integer; nBlue: integer): integer;

API Call:

Scripting

ReapplyAllOfType

ReapplyAllOfType

Coming soon.

Syntax:

Procedure: ReapplyAllOfType(sType: string);

API Call:

Scripting

OpenJobEx (2)

OpenJobEx

Shows the Open Job Dialog for the user to select a job to open.

Syntax:

Procedure: OpenJobEx;

API Call:

Scripting

SetResult

SetResult

Assigns the result of the script to the passed value.

Syntax:

Procedure: SetResult(nResult: string);

API Call:

Scripting

StopRecording

StopRecording

Forces a termination on any mouse takeoff recordings that got started.

Syntax:

Procedure: StopRecording;

API Call:

Scripting

Dialogs

Dialogs

EditScriptPropert y EditIte m Custom Dialog s MessageDialo g My Color Dialo g ScriptMessageDialo g

SelectItemDialog (2)

EditScriptProperty

EditScriptProperty

Loads the specified script property into the script editor and displays to the user for editing.

Syntax:

Procedure: EditScriptProperty(ItemPath, PropertyName: String);

API Call:

Scripting

EditItem

EditItem

Loads the given item into the Item Editor, then displays to the user for editing. If ItemPath does not exist, or if the user cancels the dialog, the function fails and returns

False.

Syntax:

Procedure: EditItem(ItemPath: String): Boolean;

API Call:

Scripting

Custom Dialogs

Custom Dialogs

One of the great new features in PlanSwift9 is the ability to create reusable dialogs using stored items and properties; simply design an item with only the desired properties set as *Input*.

Scripting

Syntax:

Procedure: Coming soon

API Call:

MessageDialog

MessageDialog

Displays a dialog with a corresponding message.

Syntax: Procedure:	
MessageDialog();	
API Call:	
Scripting	Expand source

My Color Dialog

My Color Dialog

Coming soon

Syntax: Procedure: Coming	
soon	
API Call:	
Scripting	> Expand source

ScriptMessageDialog

ScriptMessageDialog

Coming soon

Syntax: Procedure: ScriptMessageDialog;

API Call:

Scripting

SelectItemDialog (2)

SelectItemDialog

Displays the ItemDialog with specified parameters that were passed as arguments.

Syntax

Procedure: SelectItemDialog(AHeader: String = "; ACaption: String = "; RootItem: String = ");

API Call:	
Scripting	> Expand source

Global Variables and Constants

Global Variables and Constants

Coming soon

Syntax:	
Procedure: Coming soon	
API Call:	
Scripting	> Expand source

Developer Docs -- Freshdesk Xfer

Developer Docs–Freshdesk Xfer

COM Object Model - Events -- xfer from Freshdesk

COM Object Model Events – Transfer from FreshDesk

OnDoneRecording - FD

OnDoneRecording

Called when recording of a section or the stop button pressed.
Declaration: ???
API Calls Delphi
Using Iltem Object Model
Using PlanSwift Object Model
1
C#

Using	g Iltem Object Model		
1			
Using	g PlanSwift Object Model		
1			

Using Iltem	Using Iltem Object Model				
1					
Using Plans	Using PlanSwift Object Model				
1					

Pascal Scripting (OLE)

Item	Item Object Model				
1					
Deat					
ROOT	Object Model				
1					

Pascal Scripting

Item	Item Object Model				
1					
Usin	g the PlanSwift Object Model				
1					

VB/VBA (OLE)

¹⁶³¹ OnJobOpen - FD

OnJobOpen

Triggered when a new job is opened in PlanSwift.

Declaration: OnJobOpen;

API Calls

Delphi	
Using Iltem Object Model	> Expand source
Using PlanSwift Object Model	
1	

C#

Usin	Iltem Object Model
1	
Usin	; PlanSwift Object Model
1	

VB/VBA (OLE)

Usin	Using Iltem Object Model		
1			
Usin	g PlanSwift Object Model		
1			

Pascal Scripting (OLE)

Item Object Model		
1		
Root	t Object Model	
Root	t Object Model	

Pascal Scripting

1

Item Object Model	
1	
Using the PlanSwift Object Model	

1633

OnNewItem - FD

OnNewItem

Triggered when a new item, specified by ItemPath, is created.

Declaration: OnNewItem(ItemPath: String);

API Calls

Delphi

Usin	g lltem Object Model	Expand source	
Usin	Using PlanSwift Object Model		
1			

C#

VB/VBA (OLE)

Usin	g Iltem Object Model
1	
Usin	g PlanSwift Object Model
1	

Pascal Scripting (OLE)

Item	Object Model
1	
Root	t Object Model
1	

Item	Object Model
1	
Usin	g the PlanSwift Object Model
1635	

OnSelectionChanged - FD

OnSelectionChanged

Triggered when the PlanSwift application changes focus to a new "selectable" item in the editor.

Declaration: OnSelectionChanged;

API Calls

Delphi

Using IItem Object Model	Expand source		
Using PlanSwift Object Model	Using PlanSwift Object Model		
1			

C#

Using Iltem Object Mod	del		
1			
Using PlanSwift Object	: Model		
Using PlanSwift Object	: Model		
Using PlanSwift Object	t Model		

VB/VBA (OLE)

Using Iltem Object Model			
1			
Usin	g PlanSwift Object Model		
1			

Pascal Scripting (OLE)

Item	Object Model
1	
Root	t Object Model
1	t Object Model

Pascal Scripting

ltem	Object Model
1	
Usin	g the PlanSwift Object Model

1637

OnltemDelete - FD

OnItemDelete

Triggered when an item is deleted. ItemPath specifies which item was deleted.

Declaration: OnItemDelete(ItemPath: String);

API Calls

De	elphi
	Using Iltem Object Model
	Using PlanSwift Object Model

C#

1

Using Iltem Object Model	
1.	
Using PlanSwift Object Model	
1 .	

VB/VBA (OLE)

Usin	g Iltem Object Model
1	
Usin	ng PlanSwift Object Model
1	

Pascal Scripting (OLE)

Item	Object Model
1	
Root	t Object Model
Root	t Object Model
	t Object Model

Pascal Scripting

1

1639

P	tem	Object Model
	1	

Using the PlanSwift Object Model

OnItemChange - FD

OnItemChange

Triggered when an item, specified by ItemPath has been changed.

Declaration: OnItemChange(ItemPath: String);

API Calls

Delphi		
Usin	g Iltem Object Model	> Expand source
Usin	g PlanSwift Object Model	
1		

C#

Using Iltem Object Model		
1		
Using PlanSwift Object Model		
1		

VB/VBA (OLE)

Usin	g Iltem Object Model
1	
Usin	g PlanSwift Object Model
1	

Pascal Scripting (OLE)

Item	n Object Model
1	
Boot	t Object Model
ROOI	
1	

Pascal Scripting

Item	n Object Model
1	

Usin	g the PlanSwift Object Model	
1		

OnCopyItem - FD

OnCopyItem

Triggered when the PlanSwift application copies an item.

Declaration: OnCopyItem;

API Calls

Delphi		
Usin	g lltem Object Model	> Expand source
Usin	g PlanSwift Object Model	
1		

C#

Usin	g IItem Object Model
1	
Usin	g PlanSwift Object Model
1	

VB/VBA (OLE)

Usin	g Iltem Object Model
1	
11-1-	
Usin	g PlanSwift Object Model
1	

Pascal Scripting (OLE)

n Object Model
ot Object Model

Pascal Scripting

Item Object Model

¹⁶⁴³ OnSelectedPageChange - FD

OnSelectedPageChange

Triggered when the PlanSwift application changes to a "new" page in the job.

Declaration: OnSelectedPageChange;

API Calls

Delphi

Usin	g Iltem Object Model
Usin	g PlanSwift Object Model
1	

C#

Using Iltem Object Model
1
Using PlanSwift Object Model
1

VB/VBA (OLE)

Usir	ng litem Object Model
1	
Usir	ng PlanSwift Object Model
1	

Pascal Scripting (OLE)

Item	Object Model
1	
Root	t Object Model
Root 1	t Object Model
1	t Object Model

Pascal Scripting

Item Object Model

1645

OnNewJob - FD

1

Using the PlanSwift Object Model

OnNewJob

Triggered when the PlanSwift application starts a "new" job.

Declaration: OnNewJob;

API Calls

Delphi	
Using Iltem Object Model	> Expand source
Using PlanSwift Object Model	
1	

C#

Us	Using Iltem Object Model		
:			
Us	ing PlanSwift Object Model		

VB/VBA (OLE)

Usin	Using Iltem Object Model			
1				
Usin	g PlanSwift Object Model			
1				

Pascal Scripting (OLE)

Item	Object Model
1	
Root	t Object Model
Root	t Object Model

Pascal Scripting

Item Object Model			
1647			

OnJobClose - FD

OnJobClose

Triggered when the current job closes.

Declaration: OnJobClose;

API Calls

Delphi

Usin	g Iltem Object Model
Usin	g PlanSwift Object Model
1	

C#

U	ng Iltem Object Model	
L.	ng PlanSwift Object Model	

VB/VBA (OLE)

Usin	g lltem Object Model
1	
Usir	ng PlanSwift Object Model
1	

Pascal Scripting (OLE)

Item	Object Model
1	
Root	t Object Model
1	

Pascal Scripting

Item Object Model			

1649

1

Using the PlanSwift Object Model

OnClose - FD

OnClose

Triggered when the PlanSwift application closes.

Declaration: OnClose;

API Calls

Delphi	
Usin	g litem Object Model
1	
Linia	
USIN	ng PlanSwift Object Model
1	

C#

Using Iltem Object Model		
1		
Using P	lanSwift Object Model	

VB/VBA (OLE)

1

Using Iltem Object Model		
1		
Usin	g PlanSwift Object Model	
Usin 1	g PlanSwift Object Model	

Pascal Scripting (OLE)

Item Object Model	
1	
Root	t Object Model
Root 1	t Object Model

Item Object Model

1

Using the PlanSwift Object Model

OnDoneRecordingDigitizer - FD

OnDoneRecordingDigitizer

Triggered when an item section finishes recording.

Declaration: OnDoneRecordingDigitizer(ItemPath: String);

API Calls

Delphi

1

Using IItem Object Model

Using PlanSwift Object Model

C#

Using Iltem Object Model			
1			
Usin	Using PlanSwift Object Model		
1			

VB/VBA (OLE)

Using IItem Object Model		
1		
Using PlanSwift Object Model		
03111	ig Planswitt Object Model	
1		

Pascal Scripting (OLE)

Item Object Model		
1		
Root	Root Object Model	
1		

Item	Object Model
1	
Usin	g the PlanSwift Object Model
1	

OnSelectedSelectionChanged - FD

OnSelectedSelectionChanged

Triggered when the PlanSwift application changes to a new selection.

Declaration: OnSelectedSelectionChanged;

API Calls

Delphi

Using Iltem Object Model

Usir	Using PlanSwift Object Model			
1				

> Expand source

C#

Using IItem Object Model	
ng PlanSwift Object Model	

VB/VBA (OLE)

Using IItem Object Model	
1	
Using PlanSwift Object Model	
1	

Pascal Scripting (OLE)

ltem	o Object Model
1	
Root Object Model	
Root	t Object Model
Root	

Pascal Scripting

tem	tem Object Model		
1			
Using the PlanSwift Object Model			

1655

OnDigitizerSectionChanged - FD

OnDigitizerSectionChanged

Triggered when the PlanSwift application focuses a new section item.

Declaration: OnDigitizerSectionChanged;

API Calls

Delphi

Using Iltem Object Model

Using PlanSwift Object Model

1

C#

Usin	Using Iltem Object Model		
1			
Usin	Using PlanSwift Object Model		
1			

VB/VBA (OLE)

Usin	g Iltem Object Model
1	
Usin	g PlanSwift Object Model
1	

Pascal Scripting (OLE)

Item	Item Object Model		
1			
Root	Object Model		
1			

Item Object Model		
1		
Usin	g the PlanSwift Object Model	
1		

COM Object Model - Procedures -- Xfer from Freshdesk

COM Object Model - Procedures – Xfer from Freshdesk

BeginUpdate - FD

BeginUpdate

Temporarily suspends program updates.

Syntax:

Function: IPlanswift.BeginUpdate; Code Reference:

- 1. Create a New Forms Application
- Add a Planswift to the References (Planswift_Tlb)
- 2. 3. Add a button to the form
- 4. Copy code below to the onclick event of the button
- 5. Compile and run

API Calls

Delphi

Usin	g lltem Object Model	Expand source
Usin	g PlanSwift Object Model	
1		

C#

Usin	ltem Object Model
1	
Usin	PlanSwift Object Model
1	
Usin 1	PlanSwift Object Model

VB/VBA (OLE)

Usin	g Iltem Object Model
1	
Usin	g PlanSwift Object Model
1	

Pascal Scripting (OLE)

Item	Object Model
1	
Root	Object Model
1	

1				
Using the Plan	Swift Object Model			
1				

SetPropertyFormula - FD

SetPropertyFormula

Sets the Items Property Formula. (note) this will also create a new Property with the default a default Type as Text if the property does not exist.

Syntax:

Procedure: IPlanswift.SetPropertyFormula(PropertyName, value: String);

Code Reference:

- 1. Create a New Forms Application
- 2. Add a Planswift to the References (Planswift_Tlb)
- 3. Add a button to the form
- 4. Copy code below to the onclick event of the button
- 5. Compile and run

API Calls

Delphi

Using Iltem Object Model	Expand source
Using PlanSwift Object Model	
1	

C#

Using IItem Object Model		
ng PlanSwift Object Model		
g ranswitt object model		

VB/VBA (OLE)

Usin	g Iltem Object Model
1	
Usin	g PlanSwift Object Model
1	

Pascal Scripting (OLE)

Item	Item Object Model		
1			
Root	Object Model		
1			

Item Object Model		
1		
Using the PlanSwift Object Model		

Using the PlanSwift Object Model

About - FD

About

Shows the About Planswift Dialog Syntax:

Procedure: IPlanswift.About; Code Reference:

- 1. Create a New Forms Application
- 2. Add a Planswift to the References (Planswift_Tlb)
- Add a button to the form
- 4. Copy code below to the onclick event of the button
- 5. Compile and run

API Calls

Delphi Using Iltem Object Model Using PlanSwift Object Model 1

C#

Using IItem Object Model
Using PlanSwift Object Model
1

VB/VBA (OLE)

Usin	g Iltem Object Model
1	
Usin	g PlanSwift Object Model
1	

Pascal Scripting (OLE)

Item Object Model	
1	
Root Object Model	

Pascal Scripting

Item Object Model

OpenJobEx - FD

OpenJobEx

Opens the "Open Job" Dialog Box once the "Job Dialog" box appears. The com will suspend until either a job is opened or the Cancel button is pressed.

Syntax:

Procedure: IPIanSwift.OpenJobEx;

Code Reference:

- 1. Create a New Forms Application
- 2. Add a Planswift to the References (Planswift_Tlb)
- 3. Add a button to the form
- 4. Copy code below to the onclick event of the button
- 5. Compile and run

API Calls

Delphi

Usin	g Iltem Object Model	Expand source
Usin	g PlanSwift Object Model	
1		

C#

Usin	g Iltem Object Model
1	
Usin	g PlanSwift Object Model
1	

VB/VBA (OLE)

	Usin	g Iltem Object Model	
	1		
	Usin	g PlanSwift Object Model	
	1		
ltem	Object	Model	
1			

Root Object Model

Pascal Scripting (OLE)

Pascal Scripting

1

Item Object Model

BeginFormulaUpdate - FD

BeginFormulaUpdate

Signals the beginning of a formula change operation.

Syntax:

Procedure: BeginFormulaUpdate; Code Reference:

- 1. Create a New Forms Application
- 2. Add a Planswift to the References (Planswift_Tlb)
- 3.
 Add a button to the form
- 4. Copy code below to the onclick event of the button
- 5. Compile and run

API Calls

Delphi

Usin	g litem Object Model	Expand source
Usin	ng PlanSwift Object Model	
1		

C#

Using Iltem Object Model	
1	
Using PlanSwift Object Model	
1	

VB/VBA (OLE)

Usin	g Iltem Object Model
1	
11-1	
Usin	g PlanSwift Object Model
1	

Item Obje	ect Model	
1		
Root Obje	ect Model	

Pascal Scripting (OLE)

Pascal Scripting

1

Item Object Model

EndUpdate

Calls the PlanSwift EndUpdate procedure Syntax: Procedure: PlanSwift.EndUpdate; Code

Instruction:

a. Create a New Project b. Add Planswift Reference Usage

API Calls

Delphi

1

Using Iltem Object Model

Using PlanSwift Object Model

C#

1			
Using PlanS	wift Object Model		
1			

VB/VBA (OLE)

Using	g Iltem Object Model
1	
Usin	g PlanSwift Object Model
1	
	1 Usin

Pascal Scripting (OLE)

Item	Object Model
1	
Root	t Object Model
Roo t	t Object Model
	t Object Model

SetPoint - FD

SetPoint

Sets the digitizer point specified by *PointIndex* to the given *X*, *Y* coordinates.

Syntax:

Procedure: IPlanswift.SetPoint(PointIndex: Integer; X, Y: Double);

Code Reference:

- 1. Create a New Forms Application
- 2. Add a Planswift to the References (Planswift_Tlb)
- Add a button to the form
 Add a button to the onclick event of the button
 Copy code below to the onclick event of the button
 Compile and run

API Calls

Delphi

Using Iltem Object Model 1

1	
2	<pre>procedure TForm1.GetSetPoint(sender: TObject);</pre>
3	var
4	ps: IPlanSwift;
5	area,sect: IItem;
6	<pre>xs,cx,cy: Extended;</pre>
7	pgw,pgh,p1x,p1y,p2x,p2y,p3x,p3y,p4x,p4y: Extended ;
8	begin
9	<pre>ps := coPlanswift.Create;</pre>
10	<pre>area := ps.GetItem('Job\Takeoff');</pre>
11	<pre>area := area.NewItem('Area','SetPointArea');</pre>
12	<pre>sect := area.NewSection('SetPoint Area Section');</pre>
13	<pre>pgw := ps.SelectedPage.GetPropertyResultAsInteger('PageWidth',0);</pre>
14	<pre>pgh := ps.SelectedPage.GetPropertyResultAsInteger('PageHeight',0);</pre>
15	<pre>xs := ps.SelectedPage.GetPropertyResultAsFloat('ScaleX',0);</pre>
16	cx := pgw /2;
17	cy := pgh /2;
18	p1x := cx - 20 * xs;
19	ply := cy - 10 * xs;
20	<pre>sect.NewPoint(plx,ply);</pre>
21	p2x := cx + 10 * XS;
22	p2y := p1y;
23	<pre>sect.NewPoint(p2x,p2y);</pre>
24	p3x := p2x;
25	p3y := cy + 10 * xs;
26	<pre>sect.NewPoint(p3x,p3y);</pre>
27	p4x := cx - 10 * xs;
28	p4y := p3y;
29	<pre>ps.NewPoint(sect.GUID,p4x,p4y);</pre>
30	ShowMessage('Now will fix the first point by using set point');
31	p1x := cx - 10 * XS;
32	<pre>ps.SetPoint(sect.guid,0,p1x,p1y);</pre>
33	
	end;

Using PlanSwift Object Model

1

C#

Using Iltem Object Model

1	

Using PlanSwift Object Model

VB/VBA (OLE)

•

1

Using IIte	Using Iltem Object Model		
1			
Using Pla	nSwift Object Model		
1			
_			

Pascal Scripting (OLE)

Item Object Model		
Root Object Model		

Pascal Scripting

Ite	em Object Model
	1
Us	ing the PlanSwift Object Model
	1

EndFormulaUpdate - FD

EndFormulaUpdate

Signals an end to the formula update operation Syntax:

Procedure: EndFormulaUpdate

Code Reference:

- 1. Create a New Forms Application
- 2. Add a Planswift to the References (Planswift_Tlb)
- 3. Add a button to the form
- 4. Copy code below to the onclick event of the button
- 5. Compile and run

API Calls

Delphi

Using	lltem Object Model		
Using	PlanSwift Object Mc	del	
1			

Using Iltem Object Model			
1			
		_	
Using PlanSwift Object Model			
1			

VB/VBA (OLE)

Using Iltem Object Model			
1			
Usin	Using PlanSwift Object Model		
1			

Pascal Scripting (OLE)

Item	Item Object Model	
1		
Root	t Object Model	
Root	t Object Model	
	t Object Model	

Pascal Scripting Item Object Model Using the PlanSwift Object Model 1

•

•

CancelTool - FD

CancelTool

Cancels the currently active tool in PlanSwift.

Syntax: Procedure: IPIanSwift.CancelTool; Code

Reference:

- 1. Create a New Forms Application
- 2. Add a Planswift to the References (Planswift_Tlb)
- 3. Add a button to the form
- 4. Copy code below to the onclick event of the button
- 5. Open Planswift and select a digitizer object
- 6. Compile and run

API Calls

Delphi

Using Ilte	m Object Model		
Using Pla	nSwift Object Mo	del	
1			

C#

Usin	g Iltem Object Model
1	
Usin	ng PlanSwift Object Model
1	

VB/VBA (OLE)

Using Iltem	Object Model
1	
Using Plans	Swift Object Model
Using Plans	Swift Object Model
	Swift Object Model

Pascal Scripting (OLE)

el 👘

CloseJob

Closes the currently opened job.

Syntax:

Procedure: IPlanswift.CloseJob; Code Reference:

- 1. Create a New Forms Application
- 2. Add a Planswift to the References (Planswift_Tlb)
- Add a button to the form
- 4. Copy code below to the onclick event of the button
- 5. Compile and run

API Calls

Delphi				
Using IIte	em Object Mode			
Using Pla	anSwift Object N	lodel		
1				

C#

U	sing Iltem Object Model
	1
U	sing PlanSwift Object Model
	1

VB/VBA (OLE)

Usir	ng litem Object Model
1	
Usir	ng PlanSwift Object Model

Pascal Scripting (OLE)

Item	Item Object Model	
1		
Root	t Object Model	

Usin	g the PlanSwift Object Model	
1		

PostChanges - FD

PostChanges

Post changes made since call to NewChangeGroup. (See New Change Group).

Syntax:

Reference: IPlanswift.PostChanges; Code

Reference:

- 1. Create a New Forms Application
- 2. Add a Planswift to the References (Planswift_Tlb)
- 3. Add a button to the form
- 4. Copy code below to the onclick event of the button
- 5. Compile and run

API Calls

Delphi

1

Using Iltem Object Model

Using PlanSwift Object Model

C#

Usin	ng Iltem Object Model
1	

Using PlanSwift Object Model

VB/VBA (OLE)

Using Iltem Object Model			
1			
Usin	Using PlanSwift Object Model		
1			

Pascal Scripting (OLE)

Item Object Model			
1			
Roo	Root Object Model		
1			

Using the PlanSwift Object Model

1

NewPoint - FD

NewPoint

Creates a new digitizer point at the X, Y coordinates.

Syntax:

Code Reference:

- 1. Create a New Forms Application
- 2. Add a Planswift to the References (Planswift_Tlb)
- 3. Add a button to the form
- 4. Copy code below to the onclick event of the button
- 5. Compile and run

API Calls

Delphi

Using Iltem Object Model	> Expand source			
Using PlanSwift Object Model				
1				

Procedure: IPlanswift.NewPoint(X, Y: Double);

C#

Using Iltem Object Model			
Using PlanSwift Object Model			

VB/VBA (OLE)

Usin	Using Iltem Object Model			
1				
Usin	Using PlanSwift Object Model			
1				

Pascal Scripting (OLE)

Item Object Model		
1		
Root Object Model		
1		

NewChangeGroup - FD

NewChangeGroup

Starts a new change group. This will start the store of all com events taking place as an undo point until a postchages event is called (See Post Changes).

Syntax:

Procedure: NewChangeGroup(GroupName: String);

Code Reference:

- 1. Create a New Forms Application
- 2. Add a Planswift to the References (Planswift_Tlb)
- 3. Add a button to the form
- 4. Copy code below to the onclick event of the button
- 5. Compile and run

API Calls

Delphi

Usin	g Iltem Object Model	> Expand source	
Using PlanSwift Object Model			
1			

C#

Using Iltem Object Model			
ng PlanSwift Object Model			

VB/VBA (OLE)

Usin	Using Iltem Object Model			
1				
Usin	Using PlanSwift Object Model			
1				

Pascal Scripting (OLE)

Item	Item Object Model	
1		
Root Object Model		
1		

COM Object Model - Functions -- Xfer from Freshdesk

COM Object Model - Functions – Xfer from Freshdesk

SaveScreenShot - FD

SaveScreenShot

Save a screenshot of the active monitor to a specified filespec.

Syntax:

Function: SaveScreenShot(constFileName: WideString; Prompt: WordBool): WordBool;

Code Reference:

- 1. Create a New Form application
- 2. Add a button to the form
- Add Planswift to reference (Planswift9_tlb in the uses)
- 4. Copy code to button onclick event

API Calls

Delphi

1

Using Iltem Object Model	Expand source
Using PlanSwift Object Model	

C#

Using Iltem Object Model		
1		
	ļ ,	
Using PlanSwift Object Model		

VB/VBA (OLE)

1

Usin	g Iltem Object Model				
1					
Usin	Using PlanSwift Object Model				
1					

Pascal Scripting (OLE)

Item Object Model			
1			
Root Object Model			
1			

Item Object Model		

Using the	PlanSwift	Object	Model
-----------	-----------	--------	-------

GetRect - FD

GetRect

1

Prompts the user to click 2 points on the active plan to define a rectangle, then returns the coordinates in p1 and p2.

Returns 1 if the function is successful or 0 if the user cancels.

Syntax:

Function: GetRect(Varp1x: double; Varp1y: double; Varp2x: double; Varp2y: double; Hint: String): Integer;

Code Reference:

- 1. Create a New Form application
- 2. Add a button to the form
- 3. Add Planswift to reference (Planswift9_tlb in the uses)
- 4. Copy code to button onclick event

API Calls

Delphi

Using Iltem Object Model	Expand source

Using PlanSwift Object Model

C#

		Using IItem Object Model
		1
Using PlanSwift Object Model	- 1	Using PlanSwift Object Model

VB/VBA (OLE)

Using Iltem Object Model				
1				
Using PlanSwift Object Model				
1				

Pascal Scripting (OLE)

Item	Object Model
1	
Duri	
ROOT	: Object Model
1	

ltem	Object	Mode
------	--------	------

Using the	PlanSwift	Object	Model
-----------	-----------	--------	-------

GetLine - FD

GetLine

1

Prompts the user to click 2 points on the active plan to define a line then returns the coordinates in *p1 and p2*.

Returns 1 if the function is successful or 0 if the user cancels.

Syntax:

Function: GetLine(constToolHint: WideString): ILine;

Code Reference:

- 1. Create a New Form application
- 2. Add a button to the form
- 3. Add Planswift to reference (Planswift9_tlb in the uses)
- 4. Copy code to button onclick event

API Calls

Delphi

1

Using Iltem Object Model	Expand source
Using PlanSwift Object Model	

C#

	Using Iltem Object Model
	1
1	
	Using PlanSwift Object Model

VB/VBA (OLE)

Usin	g Iltem Object Model
1	
Usin	g PlanSwift Object Model
1	

Pascal Scripting (OLE)

Item	Item Object Model			
1				
Root	t Object Model			
1				

Item Object Model	Item	Object	Model
-------------------	------	--------	-------

Using the	PlanSwift	Object	Model
-----------	-----------	--------	-------

GetPropertyResult - FD

GetPropertyResult

Returns the calculated result from the given property.

Syntax:

1

Function: GetPropertyResult(ItemPath, PropertyName: String): Variant;

Code Reference:

- 1. Navigate to Plugin Store->Tool Manager and create a new Plugin
- 1.
 2.
 3.
 4. Set the plugin type to Script Code and open the Editor
 - Copy Code into the editor
 - Press run

API Calls

Delphi	
Using IItem Object Model	Expand source
Using PlanSwift Object Model	
1	

C#

Usin	Using Iltem Object Model				
1	1				

Using PlanSwift Object Model

VB/VBA (OLE)

Usin	g Iltem Object Model
1	
Usin	g PlanSwift Object Model
1	

Pascal Scripting (OLE)

Item Object Model			
1			
Root	t Object Model		
1			

Item Object Model		

Using the	PlanSwift	Object	Model
-----------	-----------	--------	-------

1694 CompareVersion - FD

CompareVersion

Compares 2 different versions of PlanSwift.

Syntax:

Function: CompareVersion; Code Reference:

- 1. Create a New Form application
- 2. 3.
- Add a button to the form Add Planswift to reference (Planswift9_tlb in the uses) Copy code to button onclick event
- 4.

API Cals

Delphi

Usin	g Iltem Object Model	Expand source
Usin	g PlanSwift Object Model	
1		

C#

Using Iltem Object Model				
1				
Usin	Using PlanSwift Object Model			
1				

VB/VBA (OLE)

Using Iltem Object Model			
1			
Usin	g PlanSwift Object Model		
0.5111			
1			

Pascal Scripting (OLE)

Item Object Model			
Using the PlanSwift Object Model			

SelectionList - FD

SelectionList

- Returns an ISelectionList object of all the selected items.
- Syntax:

Function: SelectionList: ISelectionList;

Code Reference:

- 1. Create a New Form application
- 2. Add a button to the form
- 3. Add Planswift to reference (Planswift9_tlb in the uses)
- 4. Copy code to button onclick event
- •

1

API Calls

Delphi

Usin	g lltem Object Model	> Expand source		
Usin	Using PlanSwift Object Model			
1				

C#

Using Iltem Object Model		
Using PlanSwift Object Model		
1		

VB/VBA (OLE)

Using Iltem Object Model			
1			
Using PlanSwift Object Model			
1			

Pascal Scripting (OLE)

Item Object Model			
1			
Death			
ROOT	Root Object Model		
Using the PlanSwift Object Model			
1			

IsUnlocked - FD

IsUnlocked

Checks the product activation status of a plugin. If AllowUnlock is true the user is prompted to Activate if needed.

Syntax:

Function: IsUnlocked(AProduct: String; AMajorVer: Integer; AMinorVer: Integer; AllowUnlock: Boolean): Boolean;

Code Reference:

- 1. Create a New Form application
- 2. Add a button to the form
- 3. Add Planswift to reference (Planswift9_tlb in the uses)
- 4. Copy code to button onclick event

API Calls

Delphi

Using IItem Object Model	> Expand source
Using PlanSwift Object Model	
1	

C#

Using	lltem Object Model			
1				
Using	Using PlanSwift Object Model			
1				
	PlanSwift Object Model			

VB/VBA (OLE)

Using Iltem Object Model			
1			
Using PlanSwift Object Model			
1			

Pascal Scripting (OLE)

-	
_	

Pascal Scripting

Item	Object Model			
1				
Root	Root Object Model			
Usin	g the PlanSwift Object Model			
1				

GetJobTotal - FD

GetJobTotal

Retrieves the total number of items of a certain type in the entire opened job.

Syntax:

Function: GetJobTotal(const Propertyname: WideString; const ItemType: WideString = ''): Double;

Code Reference:

- 1. Create a New Form application
- 2. Add a button to the form
- 3. Add Planswift to reference (Planswift9_tlb in the uses)
- 4. Copy code to button onclick event
- •
- API Calls

Delphi

Usin	g Iltem Object Model	> Expand source
Usin	g PlanSwift Object Model	
1		

C#

Usin	g Iltem Object Model
1	
Usin	g PlanSwift Object Model
1	

VB/VBA (OLE)

Usin	g Iltem Object Model
1	
Usin	g PlanSwift Object Model
1	

Pascal Scripting (OLE)

-	
-	

Pascal Scripting

Item	Object Model			
1				
Root	Root Object Model			
Usin	g the PlanSwift Object Model			
1				

Copyltem - FD

CopyItem

Creates a copy of Item under Parent and returns the ID of the new item.

If IncludeChildren is true, child items will be copied also.

If *SkipSections* is true, digitized sections will be duplicated also.

Syntax:

Function: CopyItem(Item: String; Parent: String; IncludeChildren: boolean; SkipSections: boolean): String;

Code Reference:

- 1. Create a New Form application
- 2. Add a button to the form
- 3. add Planswift to reference (Planswift9_tlb in the uses)
- 4. copy code to button onclick event

API Calls

Delphi Using Iltem Object Model Using PlanSwift Object Model 1

C#

Using PlanSwift Object Model	

VB/VBA (OLE)

Usin	g Iltem Object Model
1	
Usin	ng PlanSwift Object Model
1	
+	

Pascal Scripting (OLE)

Item Object Model				
1				
Root Object Model				
1				
Pascal Scripting				

Item Object Model			
1			
Lining	g the PlanSwift Object Model		
USIN	g the hanowit object model		
1			

GetProperty - FD

GetProperty

Returns the IPropertyObject specified by ItemPath and PropertyName. Returns Nil if the Item or Property is not found.

Syntax:

Function: GetProperty(ItemPath, PropertyName: String): IPropertyObject;

Code Reference:

- 1. Navigate to Plugin Store->Tool Manager and create a new Plugin
- 2. Set the plugin type to Script Code and open the Editor
- 3. Copy Code into the editor
- 4. Press run

API Calls

Delphi

Using Iltem Object Model >> Expand sources		> Expand source		
Usin	Using PlanSwift Object Model			
1				

C#

Using IItem Object Model
1
Using PlanSwift Object Model

VB/VBA (OLE)

1

Usin	g Iltem Object Model
1	
Liste	
Usin	g PlanSwift Object Model
1	

Pascal Scripting (OLE)

Item	Object Model
1	
Root	t Object Model
1	

GetItem - FD

GetItem

Returns the item given by FullPath. Returns Nil if the object is not found.

Syntax:

Function: GetItem(FullPath: String): IItem;

Code Reference:

- 1. Navigate to Plugin Store->Tool Manager and create a new Plugin
- Set the plugin type to Script Code and open the Editor
- 2. 3. Copy Code into the editor
- 4. Press run

API Calls

Delphi

Using Iltem Object Model	> Expand source
Using PlanSwift Object Model	
1	

C#

Usin	ng litem Object Model
1	
Usir	ng PlanSwift Object Model
1	

VB/VBA (OLE)

Usin	g Iltem Object Model
1	
Usin	g PlanSwift Object Model
0 Jill	
1	

Pascal Scripting (OLE)

Item	Object Model
1	
Root	t Object Medel
1000	t Object Model
1	

Item Object Model

DrawTwoWayLayout - FD

DrawTwoWayLayout

Function used to perform segment layouts (in 2 directions) at a specified span, horizontal run, as well as spacing.

Syntax:

Function: DrawTwoWayLayout(const AItem: WideString; const SpanLine: ILine; const RunLine: ILine; bIncludeFirst: WordBool; bIncludeLast: WordBool; nSpacing: Double; const AArea: WideString): WordBool;

Arguments:

Altem: WideString

Specifies the area section to assign the layout segments to.

SpanLine: ILine Direction span start and endpoint.

. .

RunLine: ILine Horizontal (side to side) run direction of area to populate. Requires a start and endpoint;

bIncludeFirst: WordBool

Specifies whether to include a segment at the "start" run point. Even if it does not fall within the spacing range.

bIncludeLast: WordBool

Specifies whether to include a segment at the "last" run point. Even if it does not fall within the spacing range.

nSpacing: Double

Specifies the "run" spacing used when laying out segment objects.

AArea: WideString (optional parameter)

Specifies a defined "Area Segment" to trim/extend laid segments to. Supply either the path or guide to the area section. Or empty double-quotes for no trim/extending required.

Code Reference:

- 1. Create a New Form application
- 2. Add a button to the form
- 3. Add Planswift to reference (Planswift9_tlb in the uses)
- 4. Copy code to button onclick event

API Calls

Delphi Using Iltem Object Model Using PlanSwift Object Model 1

C#

Usin	g Iltem Object Model
1	
Usin	ng PlanSwift Object Model
1	

VB/VBA (OLE)

Using Iltem Object I	Vlodel		
1			
Using PlanSwift Obj	ect Model		

Pascal Scripting (OLE)

ltem	Object Model
1	
Root	Object Model
1	

Pascal Scripting

Item Object Model			
1			

Using the PlanSwift Object Model

PointCount - FD

PointCount

Returns the number of digitizer points for the item.

Syntax:

Function: PointCount: Integer;

Code Reference:

- 1. Navigate to Plugin Store->Tool Manager and create a new Plugin
- 2. 3. Set the plugin type to Script Code and open the Editor
 - Copy Code into the editor
- 4. Press run

API Calls

Delphi

C#

Usin	g Iltem Object Model	
1		
		_
Usin	g PlanSwift Object Model	
1		

VB/VBA (OLE)

Usin	g Iltem Object Model
1	
Usin	g PlanSwift Object Model
1	

Pascal Scripting (OLE)

Item	o Object Model
1	
_	
Root	t Object Model
1	

Item Object Model

NewItemEx - FD

NewItemEx

Creates a new child item and returns the new item.

If EditProperties is true then the property editor will be displayed when the item is created.

Syntax:

Function: NewItemEx(ItemType, AName: String; EditProperties: Boolean): IItem;

Code Reference:

- 1. Create a New Form application
- 2. Add a button to the form
- 3. Add Planswift to reference (Planswift9_tlb in the uses)
- 4. Copy code to button onclick event

API Calls

> Expand source

C#

Using Iltem Object Model				
1				
Using PlanSwi	ft Object Model			

VB/VBA (OLE)

1

Usin	g Iltem Object Model
1	
Usin	g PlanSwift Object Model
1	

Pascal Scripting (OLE)

ltem	Object Model
1	
Root	Object Model
1	

Pascal Scripting

OpenJob - FD

OpenJob

Opens the job specified by JobPath. Returns True if ssuccessful or false if the job could not be found or opened.

Syntax:

Function: OpenJob(JobPath: String): Boolean;

Code Reference:

- 1. Navigate to Plugin Store->Tool Manager and create a new Plugin
- 1.
 2.
 3.
 4. Set the plugin type to Script Code and open the Editor
 - Copy Code into the editor
 - Press run

API Calls

Delphi

Usin	g Iltem Object Model	> Expand source
Usin	g PlanSwift Object Model	
1		

C#

Usin	ng litem Object Model
1	
Usir	ng PlanSwift Object Model
1	

VB/VBA (OLE)

Usin	g Iltem Object Model
1	
Usin	g PlanSwift Object Model
1	

Pascal Scripting (OLE)

Item Obj	ject Model	
1		
Root Obj	ject Model	
1		

Item Object Model

GetPropertyFormula - FD

GetPropertyFormula

Returns the formula string for the property specified by ItemPath and PropertyName. Returns an empty string (") if the item or property is not found.

Syntax:

Function: GetPropertyFormula(ItemPath, PropertyName: String): String;

Code Reference:

- 1. Navigate to Plugin Store->Tool Manager and create a new Plugin
- 2. Set the plugin type to Script Code and open the Editor
- 3. Copy Code into the editor
- 4. Press run

API Calls

Delphi

Using Iltem	Object Model	Expand source
Using PlanS	wift Object Model	
1		

C#

Usin	ng Iltem Object Model
1	
1.1 - 1 - 2	ng PlanSwift Object Model
Usin	
1	

VB/VBA (OLE)

Usin	Using IItem Object Model	
1		

Using PlanSwift Object Model	
1	

Pascal Scripting (OLE)

Item	Object Model	
1		
Root	t Object Model	
1		

Usin	g the PlanSwift Object Model		
1			

GetTwoWayLayout - FD

GetTwoWayLayout

1

Function used to perform segment layouts (in 2 directions) at a specified span, horizontal run, as well as spacing.

Arguments:

Altem: WideString Specifies the area section to assign the layout segments to.

SpanLine: ILine Direction span start and endpoint.

RunLine: ILine Horizontal (side to side) run direction of area to populate. Requires a start and endpoint;

bIncludeFirst: WordBool Specifies whether to include a segment at the "start" run point. Even if it does not fall within the spacing range.

bIncludeLast: WordBool

Specifies whether to include a segment at the "last" run point. Even if it does not fall within the spacing range.

nSpacing: Double

Specifies the "run" spacing used when laying out segment objects.

AArea: WideString (optional parameter)

Specifies a defined "Area Segment" to trim/extend laid segments to. Supply either the path or guide to the area section. Or, empty double-quotes for no trim/extending required.

Code Reference:

- 1. Create a New Form application
- . 2. Add a button to the form
- 3. Add Planswift to reference (Planswift9_tlb in the uses) •
 - 4. Copy code to button onclick event

API Calls

Delphi

Using IItem Object Model	Expand source
Using PlanSwift Object Model	
1	

C#

Usin	Using Iltem Object Model	
1		
Usin	g PlanSwift Object Model	
1		

VB/VBA (OLE)

Using Iltem Object Model	
1	
Usin	g PlanSwift Object Model

Pascal Scripting (OLE)

Item Object Model	
1	
Deat	Object Model
KOOL	
1	

Item Object Model	
1	
Usin	g the PlanSwift Object Model
1	

1

•

Using the PlanSwift Object Model

SetSelected - FD

SetSelected

Set the PlanSwift job "object" to either selected or not selected based on the specified itempath. Syntax:

Function: SetSelected(const ItemPath: WideString; Value: WordBool);

- Code Reference:
- 1. Create a New Form application
 - 2. Add a button to the form
 - 3. Add Planswift to reference (Planswift9_tlb in the uses)
 - 4. Copy code to button onclick event

API Calls

Delphi

Using Iltem Object Model
Using PlanSwift Object Model

C#

Using Iltem Object Model			
1			
Using PlanSwift Object Mod	el		
1			

VB/VBA (OLE)

Usin	Using IItem Object Model		
1			
Usin	g PlanSwift Object Model		
1			

Item	Object Model
1	
Root	Object Model

Pascal Scripting (OLE)

Pascal Scripting

1

1

SelectItemDialog - FD

SelectItemDialog

Displays the Planswift Select Item dialog to the user, then returns the selected item.

Syntax:

Function: SelectItemDialog(Header: String; Title: String; RootItemID: String): IItem;

Code Reference:

- 1. Create a New Form application
- 2. Add a button to the form
- 3. add Planswift to reference (Planswift9_tlb in the uses)
- 4. copy code to button onclick event

API Calls

Delphi Using Iltem Object Model Using PlanSwift Object Model 1

C#

Using Iltem Object Model
Using PlanSwift Object Model

VB/VBA (OLE)

Usin	g Iltem Object Model
1	
Usin	g PlanSwift Object Model
1	

Item Object Model		
1		
Root Object Model		

Pascal Scripting (OLE)

Pascal Scripting

1

1

٠

•

NewJobEx - FD

NewJobEx

Starts a "new" job in the PlanSwift application.

Syntax:

Function: NewJobEx(const JobName: WideString = ''): Wordbool;

Code Reference:

- 1. Create a New Form application
- 2. Add a button to the form
- 3. add Planswift to reference (Planswift9_tlb in the uses)
- 4. copy code to button onclick event

API Calls

Using Iltem Object Model

Using PlanSwift Object Model

Delphi

sing Iltem Object Model	
sing PlanSwift Object Model	
1	

1

1

C#

VB/VBA	(OLE)
Usin	g Iltem Object Model
1	
Usin	g PlanSwift Object Model
1	

Item Object Model		
1		
Root Object Model		

Pascal Scripting (OLE)

Pascal Scripting

1

1

•

•

CurrentVersion - FD

CurrentVersion

Returns the current version of the active PlanSwift application.

Syntax:

- Function: CurrentVersion; Code Reference:
- 1. Create a New Form application
 - 2. Add a button to the form
 - 3. Add Planswift to reference (Planswift9_tlb in the uses)
 - 4. Copy code to button onclick event

API Calls

Delphi

Usin	g Iltem Object Model
Usin	g PlanSwift Object Model
1	

C#

Using Iltem Obj	ect Model		
1			
Using PlanSwift	Object Model		
Osing Planswin	Object Wodel		
1			

VB/VBA (OLE)

Usin	g Iltem Object Model
1	
Usin	g PlanSwift Object Model
1	

Item Object Model			
1			
Root Object Model			

Pascal Scripting (OLE)

Pascal Scripting

1

NewBlankPage - FD

NewBlankPage

Creates a blank page in the current job and returns the PAge Item that was created.

Syntax:

Function: NewBlankPage(constAName: WideString; AWidth, AHeight, ADPI: Integer; constAScale: WideString): IItem;

Code Reference:

- 1. Create a New Form application
- 2. Add a button to the form
- 3. Add Planswift to reference (Planswift9_tlb in the uses)
- 4. Copy code to button onclick event

API Calls

Delphi

Using IItem Object Model	> Expand source
Using PlanSwift Object Model	
1	

C#

Using IItem Object Model			
1			
Usir	ng PlanSwift Object Model		
1			

VB/VBA (OLE)

Using Iltem Object Model				
1				
Usin	g PlanSwift Object Model			
1				

Pascal Scripting (OLE)

Item Object Model				
1				
Root	: Object Model			
1				
Pascal Scripting				

Item Object Model			
1			
Using the PlanSwift Object Model			

NewItem - FD

NewItem

Creates a new child item and returns the new item.

Syntax:

Function: NewItem(ItemType: String; AName: String = ''): IItem;

Code Reference:

- 1. Navigate to Plugin Store->Tool Manager and create a new Plugin
- 2. Set the plugin type to Script Code and open the Editor
- 3. Copy Code into the editor
- 4. Press run

API Calls

Delphi

Usin	g Iltem Object Model	> Expand source
Usin	g PlanSwift Object Model	
1		

C#

Usin	Using Iltem Object Model				
1					
Usin	g PlanSwift Object Model				
1					

VB/VBA (OLE)

Using Iltem Object Model				
1				
Using PlanSwift Object Model				
1				

Pascal Scripting (OLE)

Item	Item Object Model					
1						

Roo	Root Object Model				
1					

Pascal Scripting

1

Using	g the PlanSwift Object Model
1	

٠

•

IsBeta - FD

IsBeta

Returns True if Beta user, False if not.

- Syntax:
 - Function: IsBeta: Boolean;

Code Reference:

- 1. Navigate to Plugin Store->Tool Manager and create a new Plugin
- 2. Set the plugin type to Script Code and open the Editor
- 3. Copy Code into the editor
- 4. Press run

API Calls

Delphi

Using Iltem Object Model

Using PlanSwift Object Model

1

C#

Usin	Using Iltem Object Model			
1				
Usin	g PlanSwift Object Model			
1				

VB/VBA (OLE)

Using Iltem Object Model			
	1		

Using PlanSwift Object Model

1

 Item Object Model

 1

 Root Object Model

Pascal Scripting

1

Using	g the PlanSwift Object Model
1	

Root - FD

Root

Returns the Root tree object in PlanSwift.

Syntax:

Function: Root: IItem

Code Reference:

- 1. Navigate to Plugin Store->Tool Manager and create a new Plugin
- 2. Set the plugin type to Script Code and open the Editor
- 3. Copy Code into the editor
- 4. Press run

API Calls

Delphi

Using Iltem Object Model

Using PlanSwift Object Model

1

C#

Usin	Using Iltem Object Model			
1				
Using PlanSwift Object Model				
1				

VB/VBA (OLE)

Using IIt	em Object Model		
1			

Using PlanSwift Object Model

1

 Item Object Model

 1

 Root Object Model

٠

•

Pascal Scripting

1

Usin	g the PlanSwift Object Model
1	
1	

٠

•

CurrentViewport - FD

CurrentViewport

Gets the upper-right and lower-left points of the viewport.

Syntax:

Function: CurrentViewport; Code

Reference:

- 1. Create a New Form application
- 2. Add a button to the form
- 3. Add Planswift to reference (Planswift9_tlb in the uses)
- 4. Copy code to button onclick event

API Calls

Delphi

Using Iltem Object Model

Using PlanSwift Object Model

1

C#

Using Iltem Object Model	Usin
1	1
Using PlanSwift Object Model	Usin

VB/VBA (OLE)

1

Using Iltem Object Model	
1	
Using PlanSwift Object Model	

Item Object Model 1 Root Object Model

Pascal Scripting

1

Using	g the PlanSwift Object Model
-	
1	
-	

SelectedPage - FD

SelectedPage

Returns the currently selected page item or nil if no page is selected.

- Syntax:
 - Function: SelectedPage: IItem;

Code Reference:

- 1. Navigate to Plugin Store->Tool Manager and create a new Plugin
- 2. Set the plugin type to Script Code and open the Editor
- 3. Copy Code into the editor
- 4. Press run

API Calls

Delphi

Using Iltem Object Model

Using PlanSwift Object Model

C#

Using Iltem Object Model	
1	
Using PlanSwift Object Model	

VB/VBA (OLE)

Usin	g lltem Object Model
1	
	- Discuit Object Markel

Using PlanSwift Object Model

1

Item Object Model 1 Root Object Model

.

•

•

Pascal Scripting

1

Using	g the PlanSwift Object Model
1	

•

DeleteItem - FD

DeleteItem

Deletes the item specified by *ItemPath* from the system.

Syntax:

Function: DeleteItem(ItemPath: String): Boolean;

Code Reference:

- 1. Navigate to Plugin Store->Tool Manager and create a new Plugin
- 2. Set the plugin type to Script Code and open the Editor
- 3. Copy Code into the editor
- 4. Press run

API Calls

Delphi

Using Iltem Object Model

Using PlanSwift Object Model

1

C#

sing Iltem Object Model	
1	
sing PlanSwift Object Model	

VB/VBA (OLE)

ι	lsing I	Item Obje	ct Model			
	1					

Using PlanSwift Object Model

1

 Item Object Model

 1

 Root Object Model

Pascal Scripting

1

GetOneWayLayout - FD

GetOneWayLayout

Function used to perform segment layouts at a specified span, horizontal run, as well as spacing.

Arguments: Altem: WideString Specifies the area section to assign the layout segments to.

sSpanHint: WideString Hint to user on mouse cursor specifying to select the span line.

sRunHint: WideString Hint to user on mouse cursor specifying to select the run line.

bIncludeFirst: WordBool

Specifies whether to include a segment at the "start" run point. Even if it does not fall within the spacing range.

bIncludeLast: WordBool

Specifies whether to include a segment at the "last" run point. Even if it does not fall within the spacing range.

nSpacing: Double Specifies the "run" spacing used when laying out segment objects.

AArea: WideString (optional parameter)

Specifies a defined "Area Segment" to trim/extend laid segments to. Supply either the path or guid to the area section. Or, empty double-quotes for no trim/extending required.

Syntax:

Function:GetOneWayLayout(const AItem: WideString; const sSpanHint: WideString; const sRunHint: WideString; bIncludeFirst: WordBool; bIncludeLast: WordBool; nSpacing: Double; const AArea: WideString): WordBool; Code Reference:

- 1. Create a New Form application
- 2. Add a button to the form
- 3. Add Planswift to reference (Planswift9_tlb in the uses)
- 4. Copy code to button onclick event

API Calls

Delphi

Using IItem Object Model	Expand source
Using PlanSwift Object Model	
1	

C#

Using	g Iltem Object Model
1	
Using	g PlanSwift Object Model
1	
Using 1	g PlanSwift Object Model

VB/VBA (OLE)

Using Iltem Object Model
1

Using PlanSwift Object Model Pascal Scripting (OLE)

Item	n Object Model
1	
Root	t Object Model
Root	t Object Model

Item Object Model			
1			
Using the PlanSwift O	aiect Model		

GetPropertyResultAsString - FD

GetPropertyResultAsString

Returns the result value of the given property. Returns Default if the property is not found.

Syntax:

Function: GetPropertyResultAsString(ItemPath, PropertyName: String; Default String = ''): String;

Code Reference:

- 1. Create a New Form application
- 2. Add a button to the form
- Add Planswift to reference (Planswift9_tlb in the uses)
- 4. Copy code to button onclick event

API Calls

Delphi

Using	Iltem Object Model	> Expand source
Using	PlanSwift Object Model	
1		

C#

Usin	Using Iltem Object Model				
1					
Usin	g PlanSwift Object Model				
1					

VB/VBA (OLE)

Usin	g Iltem Object Model
1	
Usin	g PlanSwift Object Model
1	

Pascal Scripting (OLE)

ltem	Object Model
1	
Root	t Object Model
1	

Pascal Scripting

Usin	ng the PlanSwift Object Model
1	

IsJobOpen - FD

IsJobOpen

Syntax:

Function: IsJobOpen: Wordbool;

Code Reference:

- 1. Create a New Form application
- 2. Add a button to the form
- 3. Add Planswift to reference (Planswift9_tlb in the uses)
- 4. Copy code to button onclick event

API Call

Delphi	
Using Iltem Object Model	> Expand source
Using PlanSwift Object Model	
1	

Tests whether the PlanSwift application actually has a "Job" opened in the editor.

C#

Usin	Using Iltem Object Model			
1				
Usin	Using PlanSwift Object Model			
1				

VB/VBA (OLE)

Usin	g Iltem Object Model
1	
Usin	g PlanSwift Object Model
1	

Pascal Scripting (OLE)

Item	n Object Model
1	
Root	t Object Model
1	t Object Model

Usin	ng the PlanSwift Object Model	
1		

DrawOneWayLayout - FD

DrawOneWayLayout

Function used to perform segment layouts at a specified span, horizontal run, as well as spacing.

Syntax:

1

Function: DrawOneWayLayout(const AItem: WideString; const SpanLine: ILine; const RunLine: ILine; bIncludeFirst: WordBool; bIncludeLast: WordBool; nSpacing: Double; const AArea: WideString): WordBool;

Arguments:

Altem: WideString

Specifies the area section to assign the layout segments to.

SpanLine: ILine

Direction span start and endpoint.

RunLine: ILine

Horizontal (side to side) run direction of area to populate. Requires a start and endpoint;

bIncludeFirst: WordBool

Specifies whether to include a segment at the "start" run point. Even if it does not fall within the spacing range.

bIncludeLast: WordBool

Specifies whether to include a segment at the "last" run point. Even if it does not fall within the spacing range.

nSpacing: Double

Specifies the "run" spacing used when laying out segment objects.

AArea: WideString (optional parameter)

Specifies a defined "Area Segment" to trim/extend laid segments to. Supply either the path or guid to the area section. Or, empty double-quotes for no trim/extending required.

Code Reference:

- 1. Create a New Form application
- 2. Add a button to the form
- 3. add Planswift to reference (Planswift9_tlb in the uses)
- 4. copy code to button onclick event

API Calls

Delphi

Using IItem Object Model	> Expand source
Using PlanSwift Object Model	
1	

C#

Usin	g Iltem Object Model
1	
Usin	g PlanSwift Object Model
1	

VB/VBA (OLE)

Using Iltem Object Model	
1	
Using PlanSwift Object Model	

Pascal Scripting (OLE)

ltem	Object Model
1	
Root	t Object Model
Root	t Object Model

Item Object Mo	del			
1				
Using the PlanS	wift Obiect Model			
Using the PlanS	wift Object Model			
Using the PlanS	wift Object Model			

DeleteProperty - FD

DeleteProperty

Deletes PropertyName from ItemPath.

Syntax:

Function: DeleteProperty(ItemPath, PropertyName: String): Boolean;

Code Reference:

- Navigate to Plugin Store->Tool Manager and create a new Plugin
- 1.
 2.
 3. Set the plugin type to Script Code and open the Editor
 - Copy Code into the editor
- 4. Press run

API Calls

Delphi

Using Iltem Object Model	> Expand source
Using PlanSwift Object Model	
1	

C#

Using Iltem Object Model		
1		
Usin	g PlanSwift Object Model	
1		

VB/VBA (OLE)

Usin	ng Iltem Object Model
1	
Usin	ng PlanSwift Object Model
1	

Pascal Scripting (OLE)

Item	Object Model
1	
Root	: Object Model
Root	: Object Model

Item Object Model			

Using the PlanSwift Object Model

1

1

Edition - FD

Edition

Returns the current PlanSwift Edition.

Syntax:

Function: Edition;

Code Reference:

- 1. Navigate to Plugin Store->Tool Manager and create a new Plugin
- 2. Set the plugin type to Script Code and open the Editor
- 3. Copy Code into the editor
- 4. Press run

API Calls

Delphi

C#

Usin	g Iltem Object Model
1	
Usin	g PlanSwift Object Model
1	

VB/VBA (OLE)

Using Iltem Object Model		
1		
Usin	g PlanSwift Object Model	
1		

Pascal Scripting (OLE)

Item	Item Object Model		
1			
Root	t Object Model		
1			

GetPropertyResultAsBoolean - FD

GetPropertyResultAsBoolean

Attempt to return the result of the given property as a boolean value. If the calculated result can not be converted to a boolean value, the default value is returned.

Syntax:

Function: GetPropertyResultAsBoolean(ItemPath, PropertyName: String; Default: Boolean = False): Boolean;

Code Reference:

- 1. Navigate to Plugin Store->Tool Manager and create a new Plugin
- 2. Set the plugin type to Script Code and open the Editor
- 3. Copy Code into the editor
- 4. press run

API Calls

Delphi

Using IItem Object Model	> Expand source
Using PlanSwift Object Model	
1	

C#

Using	g lltem Object Model			
1				

Using	g PlanSwift Object Model			
1				

VB/VBA (OLE)

Usin	g Iltem Object Model
1	
Usin	g PlanSwift Object Model
1	

Pascal Scripting (OLE)

Item	m Object Model	
1		
Roo	ot Object Model	
1		

GetPropertyResultAsInteger - FD

GetPropertyResultAsInteger

Attempts to return the property value as an Integer. If the calculated value can not be converted to an integer, the value given in Default is returned.

Syntax:

Function: GetPropertyResultAsInteger(ItemPath, PropertyName: String; Default: Integer = 0): Integer;

Code Reference:

- 1. Navigate to Plugin Store->Tool Manager and create a new Plugin
- •
 2.
 Set the plugin type to Script Code and open the Editor
- 3. Copy Code into the editor
- 4. Press run

API Calls

Delphi

Usin	ng Iltem Object Model	Expand source
Usin	ng PlanSwift Object Model	
1		

C#

Using	Using Iltem Object Model			
1				
Usin	g PlanSwift Object Model			
1				

VB/VBA (OLE)

Using Iltem Object Model	
1	
Using PlanSwift Object Model	
1	

Item Object Model		
1		
Root Object Model		

Pascal Scripting

1

Using the PlanSwift Object Model

Handle - FD

Handle

Gets the handle of the current PlanSwift application.

Syntax:

Function: Handle: HResult; Code Reference:

- 1. Create a New Form application
- Add a button to the form
- Add PlanSwift to reference (Planswift9_tlb in the uses)
- 2. 3. 4. Copy code to button onclick event

API Calls

Delphi

Usin	ng litem Object Model	> Expand source
Usir	ng PlanSwift Object Model	
1		

C#

Usin	ng Iltem Object Model
1	
Usin	ng PlanSwift Object Model
1	

VB/VBA (OLE)

Using IItem Object Model			
1			
Using PlanSwift Object Model			
1			

Item Object Model			

Pascal Scripting

1

GetZoom - FD

GetZoom

Returns the current "zoom" scale factor for the active page.

Syntax:

Function: Get_Zoom: Double; Code Reference:

- 1. Create a New Form application
- 2. Add a button to the form
 3. Add Planswift to reference
 4. Copy code to button oncli Add Planswift to reference (Planswift9_tlb in the uses)
 - Copy code to button onclick event

API Calls

Delphi

Usin	g Iltem Object Model	> Expand source
Usin	ng PlanSwift Object Model	
1		

C#

Usin	Using IItem Object Model		
1			
Using PlanSwift Object Model			
1			

VB/VBA (OLE)

Using IItem Object Model			
1			
Using PlanSwift Object Model			
1			

Item Object Model		
1		
Root Object Model		

Pascal Scripting

1

GetPropertyResultAsFloat - FD

GetPropertyResultAsFloat

Attempts to return the given property value as a floating point value. If the calculated property value can not be converted, the value supplied by *Default* is returned.

Syntax:

Function: GetPropertyResultAsFloat(ItemPath, PropertyName: String; Default: Double= 0): Double;

Code Reference:

- 1. Navigate to Plugin Store->Tool Manager and create a new Plugin
- 2. Set the plugin type to Script Code and open the Editor
- 3. Copy Code into the editor
- 4. Press run

API Calls

Delphi

Usin	g Iltem Object Model	> Expand source		
Usin	Using PlanSwift Object Model			
1				

C#

Using PlanSwift Object Model			
1			

VB/VBA (OLE)

Usin	Using Iltem Object Model			
1				
Using PlanSwift Object Model				
1				

Pascal Scripting (OLE)

Item Object Model				
1				
Root	Root Object Model			
1				

Using the PlanSwift Object Model

Iltems - Procedures -- Freshdesk Xfer

Iltems - Procedures – Freshdesk Xfer

Procedures - Delete - FD

Procedures - Delete

Deletes the Item and its children from the system. Syntax: Procedure: Delete;

API Calls

Delphi		
Usin	g litem Object Model	Expand source
Usin	g PlanSwift Object Model	
1		

C#

Using Iltem Object Model	
1	
Using PlanSwift Object Model	
1	

VB/VBA (OLE)

Usin	ig litem Object Model
1	
Usin	ng PlanSwift Object Model
1	

Pascal Scripting (OLE)

Item	Item Object Model			
1				
Root	t Object Model			
1				

Item	Object Model
1	
Using	g the PlanSwift Object Model
1	
1767	

Procedures - NewPoint - FD

Procedures - NewPoint

Creates a new digitizer point at the X, Y coordinates.

Procedure: NewPoint(X, Y: Double);

API Calls

Syntax:

Delphi				
Using Ilter	m Object Model			
Using Pla	nSwift Object Mo	del		
1				

C#

Using Iltem C	bject Model		
1			
Using PlanSv	vift Object Model		
1 .			

VB/VBA (OLE)

Using Iltem Object Model	
1	
Using PlanSwift Object Model	
1	

Pascal Scripting (OLE)

Item	Object Model
1	
Root	: Object Model
1	

Pascal Scripting

ltem	Object Model				
1					

Using the PlanSwift Object Model

¹⁷⁶⁹ Procedures - Delete Property - FD

Procedures - Delete Property

Deletes PropertyName from ItemPath.

Syntax:

Function: DeleteProperty(ItemPath, PropertyName: String): Boolean;

API Calls

Delph	Delphi			
Usi	ng IItem Object Model	> Expand source		
Usi	ng PlanSwift Object Model			
1				

C#

Usin	Using Iltem Object Model		
1			
Usin	g PlanSwift Object Model		

VB/VBA (OLE)

1

Usin	g Iltem Object Model
1	
Linin	g PlanSwift Object Model
USIII	
1	

Pascal Scripting (OLE)

Item	Object Model
1	
Deed	
ROOT	: Object Model
1	

Item	n Object Model
1	
Usin	g the PlanSwift Object Model
1	
1771	

Procedures - SetPropertyFormula - FD

Procedures - SetPropertyFormula

Sets the given property formula to value.

Syntax:

Procedure: SetPropertyFormula(PropertyName, value: String);

API Calls

Delphi

Using	ng Iltem Object Model	
Usin	ng PlanSwift Object Model	
1		

C#



VB/VBA (OLE)

1		
1		
Jsing PlanSwift Object Model		

Pascal Scripting (OLE)

Item	Object Model
1	
ROOI	t Object Model
1	

ltem	n Object Model	
1		
Usin	ng the PlanSwift Object Model	
1		

1773

Procedures - SetPoint - FD

Procedures - SetPoint

Sets the digitizer point specified by *PointIndex* to the given *X*, *Y* coordinates.

Syntax:

Procedure: SetPoint(PointIndex: Integer; X, Y: Double);

API Calls

Delphi

Using IItem Object Model	> Expand source
Using PlanSwift Object Model	
1	

C#

Using Iltem Object Model		
Using PlanSwift Object Model		

VB/VBA (OLE)

Usin	g Iltem Object Model
1	
Linin	
Usin	g PlanSwift Object Model
1	

Pascal Scripting (OLE)

Item	Object Model
1	
ROOT	t Object Model
1	

Pascal Scripting

ltem	Object Model			
1				

Using the PlanSwift Object Model

Iltems - Property -- Xfer from Freshdesk

Iltems - Property --Xfer from Freshdesk

Property - ItemType - FD

Property - ItemType

Gets or Sets the *Type* property for the Item. Declaration: *ItemType: String;*

API Calls

Delphi			
Using Iltem Object Model	> Expand source		
Using PlanSwift Object Model			
1			

C#

Using IItem Object Model	
Using PlanSwift Object Model	

VB/VBA (OLE)

Using Iltem Object Model			
1			
Licin	g PlanSwift Object Model		
USIII			
1			

Pascal Scripting (OLE)

Item	Item Object Model			
1				
ROOT	t Object Model			
1				

ltem	Item Object Model		
1			
Using the PlanSwift Object Model			
Usin	g the Planswitt Object Model		
Usin 1			

¹⁷⁷⁸ Property - Name - FD

Property - Name

Gets or Sets the Name property for the item

Declaration: Name: String;

API Calls

Delphi

Using PlanSwift Object Model

1

C#

Using Iltem Object Model
1
Using PlanSwift Object Model
1

VB/VBA (OLE)

Using Ilter	n Object Model		
1			
Using Plan	Swift Object Model		
Using Plan	Swift Object Model		
1	Swift Object Model		

Pascal Scripting (OLE)

Item Object Model	
1	
Root Object Model	
1	

ltem	Item Object Model			
1				
Usin	Using the PlanSwift Object Model			
1				

¹⁷⁸⁰ Iltems - Functions -- Xfer from Freshdesk

IItems - Functions – Xfer from Freshdesk

Functions - GUID - FD

Functions - GUID

Returns the GUID for the Item.

Syntax:

Function: Iltem.GUID: String;

API Calls

Delphi Using Iltem Object Model Using PlanSwift Object Model 1

C#

Using Iltem Object Model		
Using PlanSwift Object Model		

VB/VBA (OLE)

Usin	Using IItem Object Model			
1				
Usin	Using PlanSwift Object Model			
1				

Pascal Scripting (OLE)

Item	Item Object Model			
1	1			
Root	Root Object Model			
1	1			

Item	Object Model
1	
Usin	g the PlanSwift Object Model
1	
1783	

Function - GetPropertyResultAsString - FD

Function - GetPropertyResultAsString

Returns the result value of the given property. Returns *Default* if the property is not found.

Syntax:

Function: IItem.GetPropertyResultAsString(ItemPath, PropertyName: String; Default String = ''): String;

API Calls

Delphi		
Usin	g Iltem Object Model	irce
Usin	g PlanSwift Object Model	
1		

C#

Using Iltem Object Mo	del		
1			
Using PlanSwift Object	: Model		
1			

VB/VBA (OLE)

Usin	g Iltem Object Model
1	
Usin	g PlanSwift Object Model
1	

Pascal Scripting (OLE)

Item	n Object Model
1	
-	
Root	t Object Model
Root	t Object Model

Item	Object Model
1	
Usin	g the PlanSwift Object Model
1	
1785	

NewProperty - FD

NewProperty

	Creates a new property as specified and returns the new IPropertyObject.	
	 ptNumber = 0 ptColor = 1 ptText = 2 ptMemo = 3 ptCheckBox = 4 ptPath = 5 ptImage = 6 ptLargeImage = 7 ptType = 8 ptScript = 9 ptFile = 10 ptLargeFile = 11 ptFileName = 12 ptConnectionString = 13 ptSlider = 14 ptDimension = 15 	
Delphi	API Calls	
	Using Iltem Object Model	> Expa
	Using PlanSwift Object Model	
C#	•	
Using	lltem Object Model	
1		
Using	PlanSwift Object Model	
1		

VB/VBA (OLE)

Usin	g Iltem Object Model
1	
Usin	g PlanSwift Object Model
1	

Pascal Scripting (OLE)

Item	Object Model
1	
Root	Object Model
Pascal S	ripting
Item	Object Model
1	
Usin	g the PlanSwift Object Model
1	

Functions - Propertyltem - FD

Functions - Propertyltem

Returns the IPropertyObject at the given index.

Syntax:

Function: IItem.PropertyItem(Index: Integer);

API Calls

Delphi

1

Using Iltem Object Model		

Using PlanSwift Object Model

C#

Usin	g Iltem Object Model
1	
Usin	g PlanSwift Object Model
1	

VB/VBA (OLE)

Usin	g Iltem Object Model
1	
Usin	g PlanSwift Object Model
Usin 1	g PlanSwift Object Model

Pascal Scripting (OLE)

Item O	Dbject Model
1	
Root C	Dbject Model
Root C	Dbject Model
1	Dbject Model



Functions - ChildItem - FD

Functions - ChildItem

Returns the child item at the given index position. Syntax:

Function: IItem.ChildItem(Index: Integer): IItem;

API Calls

Delphi	
Using Iltem Object Model	> Expand source
Using PlanSwift Object Model	
1	

C#

Using Iltem Object Model		
1		
Using PlanSwift Object Model		

VB/VBA (OLE)

Using Iltem Object Model	
1	
Using PlanSwift Object Model	
1	

Pascal Scripting (OLE)

Item Object Model		
1		
Root Object Model		
1		

Pascal Scripting 1791 Functions - GetPropertyResult - FD

Functions - GetPropertyResult

Returns the calculated result from the given property.

Syntax:

Function: GetPropertyResult(ItemPath, PropertyName: String): Variant;

API Calls

Delphi

Usin	ng litem Object Model	
Usin	ng PlanSwift Object Model	
1		

C#

1 Using PlanSwift Object Model	Using IIte	m Object Model			
Using PlanSwift Object Model	1				
Using PlanSwift Object Model					
Using PlanSwift Object Model					
Using PlanSwift Object Model					
	Using Pla	nSwift Object Mode	el		
	1				

VB/VBA (OLE)

Using Iltem Object Model	
1	
Using PlanSwift Object Model	
1	

Pascal Scripting (OLE)

Item	Object Model
1	
Root	: Object Model
Root	: Object Model
	: Object Model

Pascal Scripting 1793 Functions - GetItemByGUID - FD

Functions -GetItemByGUID

Returns the child item specified by aGUID.

Syntax:

Function: IItem.GetItemByGUID(aGUID: String): IItem;

Item	Object Model
1	
Using	g the PlanSwift Object Model
1	

API Calls

Delphi

Using Iltem Object Model	Expand source
Using PlanSwift Object Model	
1	

C#

Usin	g Iltem Object Model
1	
Usin	g PlanSwift Object Model
1	

VB/VBA (OLE)

Using Iltem Object Model		
1		
Usin	g PlanSwift Object Model	
1		

Pascal Scripting (OLE)

Item Object Model		
1		
Root Object Model		
1		

Functions - ParentItem - FD

Functions - ParentItem

Returns the parent to the Item. Syntax:

Function:IItem.ParentItem: IItem;

API Calls

Delphi		
Using IItem Object Model	> Expand source	

Usin	g PlanSwift Object Model
1	

C#

Usin	Using Iltem Object Model		
1	1.		
Usin	sing PlanSwift Object Model		
1	1		

VB/VBA (OLE)

Usin	Using Iltem Object Model		
1			
Usin	g PlanSwift Object Model		
USIII			
1			

Pascal Scripting (OLE)

ltem	Item Object Model		
1			
Root	Object Model		
1			

Item	Object Model
1	
Usin	g the PlanSwift Object Model

1797 Functions - FullPath - FD

Functions - FullPath

Returns the full path to the Item.	
Syntax:	

Function: IItem.FullPath: String;

API Calls

Delphi		
Usin	ng litem Object Model	> Expand source
Usin	ng PlanSwift Object Model	
1		

C#

Usin	Using Iltem Object Model		
1			
Usin	g PlanSwift Object Model		

VB/VBA (OLF)

Usin	g Iltem Object Model	
1		
Usir	g PlanSwift Object Model	
1		

Pascal Scripting (OLE)

Item Object Model			
1	1		
Root	Root Object Model		
1	1		

Pascal Scripting 1799

Functions - GetPropertyResultAsBoolean - FD

Functions - GetPropertyResultAsBoolean

Attempt to return the result of the given property as a boolean value. If the calculated result cannot be converted to a boolean value, the default value is returned.

Syntax:

Function: IItem.GetPropertyResultAsBoolean(ItemPath, PropertyName: String; Default: Boolean = False): Boolean;

API Calls

Delphi

Usin	ng Iltem Object Model	> Expand source
Usin	ng PlanSwift Object Model	
1		

C#

Using Iltem Object Model			
1			
Using PlanSwift Object Model			
1			

VB/VBA (OLE)

Using Iltem Object Model		
1		
Using PlanSwift Object Model		
1		

Pascal Scripting (OLE)

Item Object Model			
Root Object Model			

Pascal Scripting 1801

Functions - Edit - FD

Functions - Edit

Displays the Item in the Editor Dialog.
Syntax:

Function: IItem.Edit(ShowAdvanced: Boolean = True): Boolean;

Item Object Model

1

Using the PlanSwift Object Model

1

API Calls

Delphi

Using Iltem Object Model	Expand source
Using PlanSwift Object Model	
1	

C#

Using Iltem Object Model			
1			
Using PlanSwift Object Model			
1			

VB/VBA (OLE)

Using Iltem Object Model		
1		
Using PlanSwift Object Model		
1		

Pascal Scripting (OLE)

Item Object Model	
1	
Root Object Model	
Root Object Model	

Pascal Scripting 1803 Functions - PropertyCount - FD

Functions - PropertyCount

Returns the number of properties for this item. Syntax:

Function:IItem.PropertyCount: Integer;

API Calls

Delphi

Using Iltem Object Model	> Expand source	
Using PlanSwift Object Model		
1		

C#

Usin	sing Iltem Object Model	
1		
Usin	lsing PlanSwift Object Model	
1		

VB/VBA (OLE)

Using IItem Object Model		
1		
Using PlanSwift Object Model		
1		

Pascal Scripting (OLE)

Item Object Model		

ltem	Item Object Model		
1			
Using	g the PlanSwift Object Model		
1			

Functions - NewSection - FD

Functions - NewSection

Creates a new section for the Item. If the Item is not a draw object this function returns *Nil*.

Syntax:

Function: IItem.NewSection(AName: String = ''): IItem;

API Calls

Delphi

Using litem Object Model	Expand source
Using PlanSwift Object Model	
1	

C#

Using IItem Object Model		
1		
Usin	ng PlanSwift Object Model	
1		

VB/VBA (OLE)

Usin	g Iltem Object Model
1	
Usin	g PlanSwift Object Model
1	

Pascal Scripting (OLE)

Item Object Model			
1			
Root	Root Object Model		
1			

Item Object Model			
1			
Usin	Using the PlanSwift Object Model		

1807

1

Functions - NewItem - FD

Functions - NewItem

Creates a new child item and returns the new item.

Syntax:

Function: IItem.NewItem(ItemType: String; AName: String = ''): IItem;

API Calls

Delphi

1

Using Iltem Object Model

Using PlanSwift Object Model

C#

Using Iltem Object Model		
sing PlanSwift Object Model		
1		

VB/VBA (OLE)

Using	Using Iltem Object Model		
1			
Using	g PlanSwift Object Model		
1			
	g PlanSwift Object Model		

Pascal Scripting (OLE)

Item	Item Object Model		
1			
Root	t Object Model		
1			

Item	Item Object Model		
1			
Usin	Using the PlanSwift Object Model		
1			

1809

Functions - NewItemEx - FD

Functions - NewItemEx

Creates a new child item and returns the new item.

If EditProperties is true then the property editor will be displayed when the item is created.

Syntax:

Function: IItem.NewItemEx(ItemType, AName: String; EditProperties: Boolean): IItem;

API Calls

Delphi

Using IItem Object Model	Expand source
Using PlanSwift Object Model	
1	

C#

Using	g Iltem Object Model		
1			
Using PlanSwift Object Model			
1			

VB/VBA (OLE)

Usin	g Iltem Object Model
1	
Usin	g PlanSwift Object Model
1	

Pascal Scripting (OLE)

ltem	Object Model	
1		
Root Object Model		
Root	t Object Model	
Root		

Pascal Scripting

Item	Object Model
1	
Usin	g the PlanSwift Object Model
1	
1811	
Fun	ctions - GetPoint - FD

Functions - GetPoint

Returns the IPoint object from the given index position.

Syntax:Function: IItem.GetPoint(PointIndex: Integer): IPoint

API Calls	
Delphi	

1

1

Using Iltem Object Model

Using PlanSwift Object Model

C#

Using Iltem Object Model

Using PlanSwift Object Model

1		

VB/VBA (OLE)

ng Iltem Object Model
ng PlanSwift Object Model

Pascal Scripting (OLE)

1

Item Object Model

Root Object Model

1

Item Object Model	
1	
Using the PlanSwift Object Model	
1	

Functions - GetItem - FD

Functions - GetItem

Returns the item given by FullPath. Returns Nil if the object is not found. Syntax:

Function: IItem.GetItem(FullPath: String): IItem;

API Calls Delphi

Using IItem Object Model

Using PlanSwift Object Model

1

1

1

C#

Using Iltem Object Model



Using PlanSwift Object Model

VB/VBA (OLE)

Using IItem Object Model
1
Using PlanSwift Object Model
1
ascal Scripting (OLE)
Item Object Model
1
Root Object Model
1

Pascal Scripting

ltem	Object Model
1	
Usin	g the PlanSwift Object Model
1	

Functions - CanRecord - FD

Functions - CanRecord

Returns true if the item is record-able item.

Syntax:

Function: IItem.CanRecord: Boolean;

API Calls

Delphi

Usin	g litem Object Model	> Expand source
Usin	g PlanSwift Object Model	
1		

C#

Usiı	ng Iltem Object Model			
1				

Using PlanSwift Object Model

VB/VBA (OLE)

1

1

Using Iltem Object Model		
1		
Usin	Using PlanSwift Object Model	

Pascal Scripting (OLE)

Item	Object Model
1	
Root	Object Model
1	

Pascal Scripting

Item	n Object Model
1	
Usin	ng the PlanSwift Object Model
1	

Functions - DeleteItem - FD

Functions - Deleteltem

Deletes the item specified by *ItemPath* from the system.

Syntax:

Function: IItem.DeleteItem(ItemPath: String): Boolean;

API Calls

Delphi

Using Iltem Object Model
Using PlanSwift Object Model

1

C#

Using Iltem Object Model
1
Using PlanSwift Object Model

VB/VBA (OLE)

Usin	g Iltem Object Model
1	
Usin	g PlanSwift Object Model
1	

Pascal Scripting (OLE)

Item	Object Model
1	
Root	: Object Model
1	

ltem	Item Object Model	
1		
Usin	g the PlanSwift Object Model	
1		

¹⁸¹⁶ Functions - GetProperty - FD

Functions - GetProperty

Returns the IPropertyObject specified by *ItemPath* and *PropertyName*. Returns *Nil* if the Item or Property is not found. Syntax:

Function: IItem.GetProperty(ItemPath, PropertyName: String): IPropertyObject;

API Calls

Delphi		
Usir	ng litem Object Model	> Expand source
Usir	ng PlanSwift Object Model	

C#

Using Iltem Object Model
1
Using PlanSwift Object Model
1

VB/VBA (OLE)

Usin	g Iltem Object Model
1	
Usin	g PlanSwift Object Model
1	

Pascal Scripting (OLE)

Item	Item Object Model	
1		
Root Object Model		
1		

Item	Object Model	
1		
USIII	g the PlanSwift Object Model	
1		

1818 Functions - GetPropertyResultAsInteger - FD

Functions - GetPropertyResultAsInteger

Attempts to return the property value as an Integer. If the calculated value cannot be converted to an integer, the value given in Default is returned.

Syntax:

Function: IItem.GetPropertyResultAsInteger(ItemPath, PropertyName: String; Default: Integer = 0): Integer;

API Calls

Delphi	
Using IItem Object Model	> Expand source
Using PlanSwift Object Model	
1	

C#

Using Iltem Object Model		
1		
Using PlanSwift Object Model		
1		

VB/VBA (OLE)

Using Iltem Object Model		
1		
Usin	g PlanSwift Object Model	
1		

Pascal Scripting (OLE)

Item Object Model			
1			
Root	Root Object Model		
1			

Item Object Model		
1		
Usin	g the PlanSwift Object Model	
1		
T		

1820

Functions - PointCount - FD

Functions - PointCount

Returns the number of digitizer points for the item.

Syntax:

Function:IItem.PointCount: Integer;

API Calls

Delphi

Using IItem Object Model	> Expand source
Using PlanSwift Object Model	
1	

C#

1.		
Using PlanSwift Object Model		
í		

VB/VBA (OLE)

Using Iltem Object Model			
1			
Usin	g PlanSwift Object Model		
1			

Pascal Scripting (OLE)

Item Object Model			
1			
Root	Root Object Model		
1			

Item	Item Object Model		
1			
Using	Using the PlanSwift Object Model		
1			

1822

Functions - GetPropertyFormula - FD

Functions - GetPropertyFormula

Returns the formula string for the property specified by *ItemPath* and *PropertyName*. Returns an empty string (") if the item or property is not found.

Syntax:

Function: IItem.GetPropertyFormula(ItemPath, PropertyName: String): String;

API Calls

Delphi

Using Iltem Object Model	> Expand source
Using PlanSwift Object Model	
1	

C#

Using Iltem Object Model			
1			
Using PlanSwift Object Model			
1			

VB/VBA (OLE)

Usin	g Iltem Object Model
1	
Usin	g PlanSwift Object Model
1	

Pascal Scripting (OLE)

Item Object Model			
1			
Root	Root Object Model		
1			

Item Object Model				
1				
Usin	Using the PlanSwift Object Model			
1				

1824

Functions - ChildCount - FD

Functions - ChildCount

Returns the number of child items for the item.

Syntax:

Function: IItem.ChildCount: Integer;

API Calls

Delphi

Using IItem Object Model	> Expand source
Using PlanSwift Object Model	
1	

C#

Using Iltem Object Model				
1				
Using PlanSwift Object Model				
1				

VB/VBA (OLE)

Using Iltem Object Model				
1				
Usin	Using PlanSwift Object Model			
1				

ltem	Object Model
1	
Root	t Object Model
Root	t Object Model

Item Object Model				
1				
Usin	Using the PlanSwift Object Model			
1				

1826

Functions - DoRecord - FD

Functions - DoRecord

Begins recording digitizer points for the Item. Returns False if no points are recorded.

Syntax:

Function: DoRecord: Boolean;

API Calls

Delphi		
Using IItem Object Model	> Expand source	
Using PlanSwift Object Model		
1		

C#

Using Iltem Object Model			
1			
Using PlanSwift Object Model			
1			

VB/VBA (OLE)

Usin	g Iltem Object Model			
1				
Usin	Using PlanSwift Object Model			
1				

ltem	n Object Model
1	
Root	t Object Model
Root	t Object Model

Item	Object Model			
1				
Usin	Using the PlanSwift Object Model			
1				
-				

1828

Functions - GetPropertyResultAsFloat - FD

Functions - GetPropertyResultAsFloat

Attempts to return the given property value as a floating point value. If the calculated property value cannot be converted, the value supplied by *Default* is returned.

Syntax:

Function: IItem.GetPropertyResultAsFloat(ItemPath, PropertyName: String; Default: Double = 0): Double;

API Calls

Delphi	
--------	--

Usin	g Iltem Object Model	Expand source
Usin	g PlanSwift Object Model	
1		

C#

Using Iltem Object Model	
1	
Using DanSwift Object Model	
Using PlanSwift Object Model	

VB/VBA (OLE)

Using Iltem Object Model	
1	
Usin	g PlanSwift Object Model
1	

Item	Object Model
1	
Root	: Object Model
Root	: Object Model

Pascal Scripting		
Item	Object Model	
1		
Usin	g the PlanSwift Object Model	
1		
1830		

IPoint - Property -- Xfer from Freshdesk

IPoint - Property xx -Xfer from Freshdesk

Property - X - FD

Property - X

Gets or Sets the X coordinate for the IPoint. Syntax:

Property:IItem.IPoint(X: Double; Y: Double);

API Calls

Delphi		
Using IItem Object Model	> Expand source	
Using PlanSwift Object Model		
1		

C#

Using	g Iltem Object Model
1	
Usin	g PlanSwift Object Model
1	

VB/VBA (OLE)

Using IItem Object Model	
1	
Usin	g PlanSwift Object Model
	5 · · ·
1	

Pascal Scripting (OLE)

Item Object Model	
1	
Root	t Object Model
1	

Item Object Model			
1			
Hein	Using the PlanSwift Object Model		
OSIII			
1			

1833

Property - Y - FD

Property - Y

Get or Sets the Y coordinate of the IPoint.

Syntax:

Property: IItem.IPoint(X:Double; Y: Double);

API Calls

Delphi		
	Using	g Iltem Object Model
	Usin	g PlanSwift Object Model
	1	
	1	

C#

Usin	g Iltem Object Model
1	
Usin	g PlanSwift Object Model
1	

VB/VBA (OLE)

Using Iltem Object Model			
1			
Usin	Using PlanSwift Object Model		
1			

Pascal Scripting (OLE)

Item Object Model			
1			
Root Object Model			
1			
1			

Pascal Scripting

Item Object Model

Using the PlanSwift Object Model

1

1

1835 IPropertyObject - Procedures -- Xfer from FreshDesk

IPropertyObject - Procedures -- Xfer from FreshDesk

Procedures - EditScript - FD

Procedures - EditScript

Opens the script property in the script editor. If the property is not of type ptScript this method is ignored.

Syntax:

Procedure: IPropertyObject.EditScript;

API Calls

Delphi	
Usir	ng litem Object Model
Usiı	ng PlanSwift Object Model
1	

C#

Using Iltem Object Model		
1		
Using PlanSwift Object Model		
1		

VB/VBA (OLE)

Using Iltem Object Model			
1			
Usin	g PlanSwift Object Model		
1			

Pascal Scripting (OLE)

Item Object Model		
1		
Root Object Model		
1		

Item Object Model		
1		
Using the PlanSwift Object Model		
	0 ···· · ····· · ····	
1		

1838 IPropertyObject - Functions -- Xfer from Freshdesk

IPropertyObject - Functions – Xfer from Freshdesk

1839

Functions - MeetsInputCondition - FD

Functions - MeetsInputCondition

Returns true if the InputCondition has been met.

Syntax:

Function: MeetsInputCondition;

API Calls

Delphi

Expand source		
Using PlanSwift Object Model		

C#

Using Iltem Object Model			
1			
		_	

Using PlanSwift Object Model

VB/VBA (OLE)

1

Using Iltem Object Model			
1			
Using PlanSwift Object Model			
1			

Pascal Scripting (OLE)

Item Object Model			
1			
Root Object Model			
1			

Item	Item Object Model		
1			
Usin	g the PlanSwift Object Model		
1			
1841			

Functions - ExecuteScript - FD

Functions - ExecuteScript

Executes the script property, passing a CRLF delimited list of parameters. Returns the value assigned to Result in the script.

Syntax:

Function: ExecuteScript(ParamList: String = ''): Variant;

API Calls

Delphi				
Using IItem Object Model	> Expand source			
Using PlanSwift Object Model				
1				

C#

Usin	Using Iltem Object Model			
1				
Usin	g PlanSwift Object Model			
1				

VB/VBA (OLE)

Using Iltem Object Model		
1		
Usin	g PlanSwift Object Model	
Usin 1	g PlanSwift Object Model	

Pascal Scripting (OLE)

ltem	Item Object Model			
1				
Root	Object Model			
1				

Pascal Scripting 1843 Functions - ResultAsString - FD

Functions - ResultAsString

Returns the property result of the property.

Syntax:

Function: ResultAsString: String;

Delphi				
Using Iltem Ol	oject Model			>
Using PlanSwi	ft Object Model			
1				

C#

Using Iltem Object Model		
1		
Using PlanSwift Object Model		
1		

VB/VBA (OLE)

Usin	Using Iltem Object Model			
1				
Usin	g PlanSwift Object Model			
1				

Pascal Scripting (OLE)

Item	Item Object Model				
1					
Root	t Object Model				
1					

Pascal Scripting 1845 Functions - System Locked - FD

Functions - System Locked

Returns True if the property is locked by the system.

Syntax:

Function: SystemLocked: Boolean;

Item	Object Model
1	
Using	the PlanSwift Object Model
1	

Delphi

Using Iltem Object Model	Expand source
Using PlanSwift Object Model	
1	

C#

Usin	Using Iltem Object Model			
1				
Usin	g PlanSwift Object Model			
1				

VB/VBA (OLE)

Using Iltem Object Model			
1			
Usin	g PlanSwift Object Model		
1			

Pascal Scripting (OLE)

Item Object Model	
1	
Root Object Model	
Root Object Model	

Functions - ResultAsInteger - FD

Functions - ResultAsInteger

Returns the property result as an integer if possible. Syntax: Function: ResultAsInteger: Integer;

API Calls

Delphi

Using Iltem Object Model

Using PlanSwift Object Model

1

C#



VB/VBA (OLE)

Using IIte	em Object Model		
1			
Using Pla	anSwift Object Model		
1			

Pascal Scripting (OLE)

Item Object Model		
1		
Root	t Object Model	
1		

Pascal Scripting

Item C	Dbject Model
1	
Using	the PlanSwift Object Model
1	
1849	

Functions - System Hidden - FD

Functions - System Hidden

Returns True if the property is Hidden by the system.

Syntax:

Function: SystemHidden: Boolean;

API Calls

Delphi		
Using Iltem Object Model	Expand source	
Using PlanSwift Object Model		
1		

C#

Usin	g Iltem Object Model
1	
Usin	g PlanSwift Object Model
1	

VB/VBA (OLE)

Usin	Using Iltem Object Model		
1			
Usin	Using PlanSwift Object Model		
1			
1			

Pascal Scripting (OLE)

ltem	Object Model
1	
Root	Object Model
1	

Pascal Scripting
1851

Functions - ResultAsVariant - FD

Functions - ResultAsVariant

Returns the property result as a Variant; Syntax:

Function: ResultAsVariant: Variant;

Delphi

Using Iltem Object Model
Using PlanSwift Object Model
1

C#

Using Iltem Object Model

1			
Usin	ng PlanSwift Object Model		
1			

VB/VBA (OLE)

Using	g Iltem Object Model
1	
Using	g PlanSwift Object Model
1	

Pascal Scripting (OLE)

Item	Object Model
1	
Root	: Object Model

Pascal Scripting 1853

Functions - PropertyType - FD

Functions - PropertyType

Returns the Type attribute for the property.

Syntax:

Function: PropertyType: String;

Item Object Model			
1			
Using the PlanSwift Object Model			
1			

Delphi

Using Iltem Object Model	> Expand source
Using PlanSwift Object Model	
1	

C#

Usin	Using Iltem Object Model			
1				
Usin	Using PlanSwift Object Model			
1				

VB/VBA (OLE)

Using Iltem Object Model				
1				
Usin	Using PlanSwift Object Model			
1				

Pascal Scripting (OLE)

Item Object Model	
1	
Root Object Model	
Root Object Model	

Functions - ResultAsFloat - FD

Functions - ResultAsFloat

Returns the Type attribute for the property.

Syntax:

Function: PropertyType: String;

API Calls

Del	nhi	
Dei	pill	

Using Iltem Object Model

Using PlanSwift Object Model

C#

Using IItem Object Model			
Using PlanSwift Object Model			
1			

VB/VBA (OLE)

Using Iltem Object Model			
1			
Using	g PlanSwift Object Model		
1			

Pascal Scripting (OLE)

Item Object Model			
1			
Root	Object Model		
1			

Pascal Scripting

Item	Item Object Model			
1				

Using the PlanSwift Object Model

IPropertyObject - Property -- Xfer from Freshdesk

IPropertyObject - Property – Xfer from Freshdesk

Property - Expression - FD

Property - Expression

Gets or Sets the Expression attribute for the property. Syntax:

Property: IPropertyObject.Expression: Boolean;

API Calls

Delphi		
Usin	g Iltem Object Model	> Expand source
Usin	g PlanSwift Object Model	
1		

C#

Using Iltem Object Model		
1		
Using PlanSwift Object Model		
1		

VB/VBA (OLE)

Using Iltem Object Model		
1		
Usin	Using PlanSwift Object Model	
1		

Item Object Model	
1	
Root Object Model	
1	

Item	ltem Object Model		
1			
Using the PlanSwift Object Model			
1			

1860

Property - TreeList - FD

Property - TreeList

Gets or Sets the TreeList attribute of the property. If ListType = ItTreeList this attribute will contain the full path to the treelist item to use for a root item in the list.

Syntax:

Property: TreeList: String;

API Calls

Delphi

Using Iltem Object Model	> Expand source
Using PlanSwift Object Model	
1	

C#

Usin	ng litem Object Model
1	
Usin	ng PlanSwift Object Model
1	

VB/VBA (OLE)

Usin	g Iltem Object Model	
1		
Using PlanSwift Object Model		
1		

Item Object Model		
1		
Root	t Object Model	
1		

1862

Property - SimpleList - FD

Property - SimpleList

Gets or Sets the SimpleList attribute for the property. If ListType = ItSimpleList, the SimpleList attribute will be the CRLF delimited string of list items.

Syntax:

Property: SimpleList: String;

API Calls

Delphi

Using II	tem Object Model	Expand source
Using P	lanSwift Object Model	
1		

C#

Usin	Using Iltem Object Model		
1			
Usin	Using PlanSwift Object Model		
1			

VB/VBA (OLE)

Using Iltem Object Model			
1			
Usin	Using PlanSwift Object Model		
1			

Item	Item Object Model		
1			
Using the PlanSwift Object Model			
1			

Item Object Model				
1				
Root	t Object Model			
Root	t Object Model			

1864

Property - SliderMax - FD

Property - SliderMax

Gets or Sets the SliderMax attribute for the property.

Syntax:

Property: SliderMax: Integer;

API Calls

Delphi		
Usin	g Iltem Object Model	Expand source
Usin	g PlanSwift Object Model	
1		

C#

Using	Using Iltem Object Model				
1					
Using	g PlanSwift Object Model				
1					

VB/VBA (OLE)

Usin	g Iltem Object Model
1	
Usin	g PlanSwift Object Model
1	

Pascal Scripting (OLE)

Item O	Dbject Model
1	
Deet O	
Root O	Dbject Model
Root O	Dbject Model

Pascal Scripting

Item	Object Model		
1			

Using the PlanSwift Object Model

1866

Property - ListShowSearch - FD

Property - ListShowSearch

Gets or Sets the ListShowSearch attribute for the property.

Syntax:

Property: ListShowSearch: Boolean;

API Calls

Delphi

1

Using Iltem Object Model

Using PlanSwift Object Model

C#

Using Iltem Object Model

Using PlanSwift Object Model

VB/VBA (OLE)

Usin	ng Iltem Object Model
1	
Usin	ng PlanSwift Object Model
1	

	Item Object Model
	1
	Root Object Model
	1
Ρ	iscal Scripting

1			
	e PlanSwift Objec	t Model	
Using th			

1868 Property - InheritAction - FD

Property - InheritAction

1

Gets or sets the InheritAction attribute for this property.

- iaNormal = 0 iaIgnore = 1
- iaInheritFormula = 2
- iaInheritResult = 3 iaFlatten
- = 4
- Syntax:

Property: InheritAction: Inheritactions;

API Calls

Delphi		
Usin	ng litem Object Model	Expand source
Usin	ng PlanSwift Object Model	
1		

C#

Using IItem Object Model	
1	

llcing	PlanSwift	Object	Model
USING	Flanswire	Object	would

VB/VBA (OLE)

1

Using Iltem Object Model				
1				
Usin	ng PlanSwift Object Model			
1				

Pascal Scripting (OLE)

Item	Item Object Model				
1					
Root	: Object Model				
1					

Item	Item Object Model				
1					
1870					

Property - SliderMin - FD

Property - SliderMin

Gets or Sets the SliderMin attribute for this property.

Syntax:

Property: SliderMin: Integer;

API Calls

Delphi				
Using	Using IItem Object Model			
Using	Using PlanSwift Object Model			
1				

C#



VB/VBA (OLE)

Using Iltem Object Model				
1				
Using Pl	anSwift Object Model			
1				

Pascal Scripting (OLE)

Item Ob	ject Model
1	
Root Ob	pject Model
Root Ob	oject Model
	oject Model

Pascal Scripting

1872

Property - InputCondition - FD

Property - InputCondition

Gets or Sets the InputCondition attribute for the property. Syntax:

Property: InputCondition: String;

Delphi

Using litem Object Model > Expand so			
Using PlanSwift Object Model			
1			

C#

Usin	Using Iltem Object Model					
1						
Usin	Using PlanSwift Object Model					
1						

VB/VBA (OLE)

Usin	Using Iltem Object Model				
1					
Usin	g PlanSwift Object Model				
1					

Pascal Scripting (OLE)

Item Object Model	
1	
Root Object Model	
1	

Pascal Scripting 1874 Property - InputUnits - FD

Property - InputUnits

Gets or Sets the InputUnits attribute for the property. Syntax:

Property: InputUnits: String;

Item	Item Object Model				
1					
Using	the PlanSwift Object Model				
1					

Delphi

Using III	tem Object Model		
Using P	lanSwift Object Model		
1			

C#

Using	tem Object	: Model		
1				
_				
Using	lanSwift O	bject Model		
Using	PlanSwift O	bject Model		
	PlanSwift O	bject Model		

VB/VBA (OLE)

Usin	g Iltem Object Model
1	
Liste	- Discuift Ohiost Madel
USIN	g PlanSwift Object Model
1	g Planswift Object Model
1	g Planswift Object Model

Pascal Scripting (OLE)

Item	Item Object Model		
1			
Root	t Object Model		
1			

Property - IsInput - FD

Property - IsInput

Gets or Sets the IsInput attribute for the property.

Syntax:

Property: IsInput: Boolean;

API Calls

Delphi

Using IItem Object Model	> Expand source
Using PlanSwift Object Model	
1	

C#

Usin	g Iltem Object Model
1	
Usin	ig PlanSwift Object Model
Usin 1	g PlanSwift Object Model

VB/VBA (OLE)

Using	; Iltem Object Model
1	
Usin	g PlanSwift Object Model
1	

Pascal Scripting (OLE)

Item	Object Model
1	
Root	t Object Model
1	

Pascal Scripting

ltem	Item Object Model		
1			
Linin	g the PlanSwift Object Model		

Property - Units - FD

Property - Units

Gets or Sets the Units attribute for the property.

Syntax:

1

Property: Units: String;

Delphi

Usin	g Iltem Object Model
Usin	ng PlanSwift Object Model
1	

C#

Using IIte	em Object Mo	del		
1				
Using Pla	nSwift Objec	t Model		
1				

VB/VBA (OLE)

Using Ilte	m Object Model		
1			
Using Pla	nSwift Object Mode	I	
1			

Pascal Scripting (OLE)

Item	Object Model
1	
Root	t Object Model
1	

Pascal Scripting

Item Object Model	
1	

Using the PlanSwift Object Model

Property - SliderShowTicks - FD

Property - SliderShowTicks

Gets or Sets the SliderShowTicks attribute for this property.

Syntax:

Property: SliderShowTicks: Boolean;

API Calls

Delphi Using Iltem Object Model Using PlanSwift Object Model 1

C#

VB/VBA (OLE)

Pascal Scripting (OLE)

Item Object Model		
1		
Root	t Object Model	
Root	t Object Model	

Item	Item Object Model		
1			
Usin	Using the PlanSwift Object Model		

Property - SliderTickFrequency - FD

Property - SliderTickFrequency

Gets or Sets the SliderTickFrequency attribute for this property.

Syntax:

Property: SliderTickFrequency: Integer;

API Calls

Delphi	
Usin	g Iltem Object Model
Usir	g PlanSwift Object Model
1	

C#

Using Iltem Object Model	
1	
Usir	ng PlanSwift Object Model

VB/VBA (OLE)

1

Usin	ng Iltem Object Model	
1		
Usir	ng PlanSwift Object Model	
1		

Pascal Scripting (OLE)

Item	Object Model
1	
Root	Object Model
1	

Item Object Model		
1		
Using	the PlanSwift Object Model	
1		

Property - DecimalPlaces - FD

Property - DecimalPlaces

Gets or Sets the DecimalPlaces attribute for the property.

Syntax:

Property: IPropertyObject.DecimalPlaces: Integer;

API Calls

Delphi		
Using	lltem Object Model	Expand source
Using	PlanSwift Object Model	
1		

C#

Using Iltem Object	Model		
1			
Using PlanSwift Ob	oject Model		
1			

VB/VBA (OLE)

Using Iltem Object Model			
1			
Using PlanSwi	ft Object Model		
1			

Pascal Scripting (OLE)

Root Object Model		

Pascal Scripting 1885

Property - ListShowOnlyTypes - FD

Property - ListShowOnlyTypes

Gets or Sets the ListShowOnlyTypes attribute for this property.

Syntax:

Property: ListShowOnlyTypes: String;

API Calls

Delphi

Using Iltem Object Model		
Using PlanSwift Object Model		
1		

C#

Using Iltem Object Model
1,
Using PlanSwift Object Model
1,

VB/VBA (OLE)

Usin	g Iltem Object Model
1	
Usin	g PlanSwift Object Model
1	

Pascal Scripting (OLE)

Item	n Object Model
1	
Roo	t Object Model
1	

Item	Item Object Model		
1			
11-1			
Using the PlanSwift Object Model			
1			

Property - InheritPullForm - FD

Property - InheritPullForm

Gets or Sets the InheritPullFrom attribute for this property.

Syntax:

Property: InheritPullFrom: String;

API Calls

Delphi

Using Iltem Object Model	> Expand source
Using PlanSwift Object Model	
1	

C#

Using IItem Object Model	
1	
Using PlanSwift Object Model	
1	

VB/VBA (OLE)

Usin	g Iltem Object Model		
1			
Usin	Using PlanSwift Object Model		
1			

Pascal Scripting (OLE)

Item	Item Object Model			
1				
Root	Root Object Model			
1				

Item	Object Model
1	
Usin	g the PlanSwift Object Model
1	
1889	

Property - ListType - FD

Property - ListType

Gets or Sets the ListType attribute for the property.

Syntax:

Property: ListType: ListTypes;

API Calls

Delphi

Using IItem Object Model

Using PlanSwift Object Model

1

C#

Using Iltem	Object Model		
1			
Using PlanS	wift Object Model		

VB/VBA (OLE)

Usin	g Iltem Object Model
1	
Usin	g PlanSwift Object Model

Pascal Scripting (OLE)

Item Object Mod	lel		
1			
Root Object Model			
1			

ltem	Object Model
1	
Usin	g the PlanSwift Object Model
1	
-	



Property - ScriptParameters - FD

Property - ScriptParameters

Gets or Sets the ScriptParameters attribute for this property. This string is a CRLF delimited list of Parameter names.

Syntax:

Property: ScriptParameters: String;

API Calls

Delphi

Using Iltem Object Model	> Expand source
Using PlanSwift Object Model	
1	

C#

Usin	Using Iltem Object Model			
1				
Usin	Using PlanSwift Object Model			
1				

VB/VBA (OLE)

Usin	g Iltem Object Model		
1			
Usin	Using PlanSwift Object Model		
1			

Pascal Scripting (OLE)

Item	Object Model		
1			
Root	Root Object Model		
1			

Item Object Model 1 Using the PlanSwift Object Model 1 1893

Property - UserHidden - FD

Property - UserHidden

Gets or Sets the UserHidden attribute for the property.

Syntax:

Property: UserHidden: Boolean;

API Calls

Delphi

Using Iltem Object Model

Using PlanSwift Object Model

1

C#

Using Iltem Object Model

Using PlanSwift Object Model

VB/VBA (OLE)

1

Usin	g lltem Object Model
1	
Usin	g PlanSwift Object Model
1	

Item	Item Object Model		
1			
Using the PlanSwift Object Model			
1			

1895

Property - ScriptLanguage - FD

Property - ScriptLanguage

Gets or Sets the ScriptLanguage attribute for this property.

- slPascal = 0 slBasic = 1
- slExecute = 2

• Syntax:

Property: ScriptLanguage: ScriptLanguages;

API Calls

Delphi Using Iltem Object Model Using PlanSwift Object Model 1

C#

Usin	Using Iltem Object Model				
1					
Usin	g PlanSwift Object Model				
1					

VB/VBA (OLE)

Using IItem Object Model		
1		
Usin	g PlanSwift Object Model	
1		

Item Object Model		
1		
Root	t Object Model	
1		

Pascal Scripting

Item Object Model		
1		
Usin	g the PlanSwift Object Model	

Property - Group - FD

Property - Group

Gets or Sets the Group attribute for the property.

Syntax:

Property: Group: String;

API Calls

Delphi
Using Iltem Object Model

Using PlanSwift Object Model

1

C#

Using Iltem Object Model			
1			
Usir	ng PlanSwift Object Model		
1			

VB/VBA (OLE)

Using Iltem Object Model		
1		
Using	PlanSwift Object Model	
1		

Pascal Scripting (OLE)

Item Object Model			
1			
Root Object Model			
1			

Pascal Scripting

1899

Property - ListPropertiesToSet - FD

Property - ListPropertiesToSet

Gets or Sets the ListPropertiesToSet attribute for this property.

Syntax:

Property: ListPropertiesToSet: String;

API Calls

Delphi Using Iltem Object Model Using PlanSwift Object Model 1

C#

Using Iltem Object Model

 1

 Using PlanSwift Object Model

 1

VB/VBA (OLE)

Using IItem Object Model		
1		
Usin	g PlanSwift Object Model	
1		

Pascal Scripting (OLE)

Item Object Model		
1		
Root	t Object Model	
1		

Pascal Scripting 1901

Property - ListColumnAutoWidth - FD

Property - ListColumnAutoWidth

Gets or Sets the ListColumnAutoWidth attribute for the property.

Syntax:

Property: ListColumnAutoWidth: Boolean;

Item	ltem Object Model		
1			
Using	the PlanSwift Object Model		
1			

API Calls

Delphi

Using Iltem Object Model	> Expand source
Using PlanSwift Object Model	
1	

C#

Usin	Using Iltem Object Model		
1			
Usin	Using PlanSwift Object Model		
1			

VB/VBA (OLE)

Using Iltem Object Model		
1		
Using PlanSwift Object Model		
1		

Pascal Scripting (OLE)

Item Object Model		
1		
Root Object Model		
Root Object Model		

Property - List - FD

Property - List

Gets or Sets the List attribute for the property. If ListType = ItList then this string will be the full path to the PlanSwift List Object as defined on the List tab on the main ribbon bar.

Syntax:

Property: List: String;

API Calls

Delphi		
Usin	ng litem Object Model	> Expand source
Usin	ng PlanSwift Object Model	
1		

C#

Using Iltem Object Model	
1	
Using	PlanSwift Object Model
1	

VB/VBA (OLE)

Using IItem Object Model			
1			
Usin	Using PlanSwift Object Model		
1			

Pascal Scripting (OLE)

Item	Item Object Model		
1			
Root	Root Object Model		
1			

Item Object Model		
1		
	Using the PlanSwift Object Model	
Usin	ig the PlanSwift Object Model	
Usin 1	ig the Planswift Object Model	

¹⁹⁰⁵ Property - Formula - FD

Property - Formula

Gets or Sets the Formula attribute for the property.

Syntax:

Property: Formula: String;

API Calls

Delphi
Using Iltem Object Model

Using PlanSwift Object Model

1

C#

Using Iltem Object Model	
ng PlanSwift Object Model	

VB/VBA (OLE)

Using Iltem Object Model			
1			
Usin	Using PlanSwift Object Model		
1			

Pascal Scripting (OLE)

Item	Item Object Model		
1			
ROOT	Root Object Model		
1			

Item Object Model		
1		
Using the PlanSwift Object Model		
- O Jill		
1		

1907

Property - Image Transparent - FD

Property - Image Transparent

Gets or Sets the ImageTransparent attribute for this property.

API Call:

Syntax:

ImageTransparent: Boolean;

API Calls

Using Iltem Object Model	> Expand source
Using PlanSwift Object Model	
1	

C#

Usir	g Iltem Object Model	
1		
		_
Usir	g PlanSwift Object Model	
1		

VB/VBA (OLE)

Using Iltem Object Model	
1	
Using PlanSwift Object Model	
1	

Item Object Model		
1		
Root Object Model		
Root		
1		

ltem	Item Object Model			
1				
Using the PlanSwift Object Model				

1909

Property - CompileDenyWrite - FD

Property - CompileDenyWrite

Gets or Sets the CompileDenyWrite attribute for this property.

Syntax:

Property: IPropertyObject.CompileDenyWrite: Boolean;

API Calls

Delphi

Using Iltem Object Model	> Expand source
Using PlanSwift Object Model	
1	

C#

Using IItem Object Model		
1		
Using PlanSwift Object Model		
1		

VB/VBA (OLE)

Using IItem Object Model		
1		
Using PlanSwift Object Model		
1		

Item Object Model		
1		
Root Object Model		
1		

Item Object Model		
1		
Usin	g the PlanSwift Object Model	
1		

1911

Property - UserLocked - FD

Property - UserLocked

Syntax:

Gets or Sets the UserLocked attribute of the property.

Property: UserLocked: Boolean

API Calls

Delphi			
Using Ilten	n Object Model		> Expand source
Using PlanSwift Object Model			
1			

C#

Usir	g Iltem Object Model
1	
Usir	ng PlanSwift Object Model
1	

VB/VBA (OLE)

Usin	g Iltem Object Model
1	
Usin	g PlanSwift Object Model
1	

Item	n Object Model
1	
Root	t Object Model
1	

Item	Item Object Model				
1					
Usin	Using the PlanSwift Object Model				
1					

1913

Property - IsInherited - FD

Property - IsInherited

Gets or Sets the IsInherited attribute for this property.

Syntax:

Property: IsInherited: Boolean;

API Calls

Delphi Using Iltem Object Model Using PlanSwift Object Model 1

C#

Using Iltem Object Model	
1	
Using PlanSwift Object Model	
1	

VB/VBA (OLE)

Usin	g Iltem Object Model
1	
Usin	g PlanSwift Object Model
1	

1

Pascal Scripting

ltem	Item Object Model				
1	1				

Using the PlanSwift Object Model

¹⁹¹⁵ Property - CompileDenyRead - FD

Property - CompileDenyRead

Gets or Sets the CompileDenyRead attribute for this property. Syntax:

Property: IPropertyObject.CompileDenyRead: Boolean;

API Calls

Delphi

Using IItem Object Model	> Expand source
Using PlanSwift Object Model	
1	

C#

Using Iltem	Object Model		
1			
Using Plans	Swift Object Model		
1			

VB/VBA (OLE)

Usin	g Iltem Object Model		
1			
Using PlanSwift Object Model			
1			

Item	n Object Model
1	
Root	t Object Model
1	

Item Object Model				
1				
Using the PlanSwif	t Object Model			
1				
1917				

Property - InputType - FD

Property - InputType

Gets or Sets the InputType attribute for the property.

inpStoreLocal = 0 inpStoreParent = 1 Syntax: *InputType: InputTypes;*

API Calls

Delphi

Using Iltem Object Model	Expand source
Using PlanSwift Object Model	
1	

C#

Using Iltem Object Model	
Using PlanSwift Object Model	

VB/VBA (OLE)

Usin	g Iltem Object Model
1	
Usin	g PlanSwift Object Model
1	

Item	Item Object Model				
1					
Root	Object Model				
1					

Iten	Object Mode	el				
1						

Using the PlanSwift Object Model

1919

Property - Adjust - FD

Property - Adjust

Syntax:

Gets or Sets the Adjust attribute for the property.

Property: Adjust: String;

API Calls

Delphi

Usin	g Iltem Object Model	> Expand source
Usin	g PlanSwift Object Model	
1		

C#

Using Iltem (Using Iltem Object Model				
1					
Using PlanSv	wift Object Model				

VB/VBA (OLE)

Using Ilte	m Object Model		
1			
Using Pla	nSwift Object Model		
-	•		
1			
-			

Pascal Scripting (OLE)

Item	Item Object Model				
1					
Root	t Object Model				
1					

Item	Item Object Model				
1					
Usin	ng the PlanSwift Object Model				
1					
Usin 1	ng the PlanSwift Object Model				

1921

Property - ListResultColumn - FD

Property - ListResultColumn

Gets or Sets the ListResultColumn attribute for the property. If the ListType = ItList, this attribute specifies which column to return for the result.

Syntax:

Property: ListResultColumn: String;

API Calls

Delphi

Usin	g Iltem Object Model	> Expand source
Usin	g PlanSwift Object Model	
1		

C#

Usin	g Iltem Object Model
1	
	-
Usin	g PlanSwift Object Model
1	

VB/VBA (OLE)

Usin	g Iltem Object Model
1	
Usin	g PlanSwift Object Model
1	

Item	Object Model
1	
Poot	Object Model
ROOL	
1	

1

Pascal Scripting

ltem	Object Model			
1				

Using the PlanSwift Object Model

¹⁹²³ Property - CalculateBeforeInherit - FD

Property - CalculateBeforeInherit

Gets or Sets the CalculateBeforeInherit attribute for the property
Syntax:

Property: IPropertyObject.CalculateBeforeInherit: Boolean;

API Calls

Delphi

Using Iltem Object Model	> Expand source
Using PlanSwift Object Model	
1	

C#

Using IItem Object Model		
1		
Using PlanSwift Object Model		

VB/VBA (OLE)

1

Usin	g Iltem Object Model
1	
Usin	g PlanSwift Object Model
1	

Item Object Model			
1			
Root	Object Model		
1			

Item	Item Object Model		
1			
Usin	ng the PlanSwift Object Model		
1			

1925

Property - ListFromProperty - FD

Property - ListFromProperty

Gets or Sets the ListFromProperty attribute for the property; Syntax:

Property: ListFromProperty: Boolean;

API Calls

Delphi

Using Iltem Object Model

Using PlanSwift Object Model

C#

Using Iltem Object Model

Using PlanSwift Object Model

VB/VBA (OLE)

Using Iltem Object Model

Using PlanSwift Object Model

Pascal Scripting (OLE)

Item Object Model

Root Object Model

Item Object Model

Using the PlanSwift Object Model

Property - PlugInToExecuteButtonCaption - FD

Property - PlugInToExecuteButtonCaption

Gets or Sets the $\mathsf{PlugInToExecuteButtonCaption}$ attribute for this property.

Syntax:

Property: PlugInToExecuteButtonCaption: String;

API Calls

Delphi

Using Iltem Object Model	> Expand source

Usin	g PlanSwift Object Model
1	

C#

Using Iltem Object Model	
1	
Using PlanSwift Object Model	

VB/VBA (OLE)	
Using Iltem Object Model	
1	
Using PlanSwift Object Model	
1	

Pascal Scripting (OLE)

Item	Object Model
1	
Root	t Object Model
Root	t Object Model

Item	Item Object Model	
1		
Using the PlanSwift Object Model		

Property - ListVisibleColumnsInDropdown - FD

Property - ListVisibleColumnsInDropdown

Gets or Sets the ListVisibleColumnsInDropdown attribute for this property.

Syntax:

Property: ListVisibleColumnsInDropdown: String;

API Calls

Delphi		
Using I	ltem Object Model	Expand source
Using F	PlanSwift Object Model	
1		

C#

Usin	Using IItem Object Model	
1	1	
Usin	Using PlanSwift Object Model	
1		

VB/VBA (OLE)

Usin	g Iltem Object Model
1	
Usin	g PlanSwift Object Model
1	

Pascal Scripting (OLE)

Item	Object Model		
1			
Root	Root Object Model		
1			

Item	Object Model
1	
Usin	g the PlanSwift Object Model
1	

1930 Property - ScriptType - FD

Property - ScriptType

Gets or Sets the ScriptType attribute for this property.

Syntax:

Property: ScriptType: ScriptTypes;

API Calls

Delphi		
Usin	ng Iltem Object Model	> Expand source
Usin	ng PlanSwift Object Model	
1		

C#

Usin	g Iltem Object Model
1	
Usin	g PlanSwift Object Model
1	

VB/VBA (OLE)

Usin	g Iltem Object Model
1	
Usin	g PlanSwift Object Model
0 Sill	
1	

Pascal Scripting (OLE)

Item	Object Model
1	
Root	t Object Model
1	

Pascal Scripting 1932

Property - ListShow1Level - FD

Property - ListShow1Level

Gets or Sets the ListShow1Level attribute for this property. Syntax:

Property: ListShow1Level: Boolean;

API Calls

Delphi

Usin	g Iltem Object Model	Expand source
Usin	g PlanSwift Object Model	
1		

C#

Usin	Using Iltem Object Model			
1				
Usin	g PlanSwift Object Model			
1				

VB/VBA (OLE)

Usin	g Iltem Object Model
1	
Usin	g PlanSwift Object Model
1	

Pascal Scripting (OLE)

Pascal Scripting 1934

Property - PluginToExecute - FD

Property - PluginToExecute

Gets or Sets the PlugInToExecute attribute for this property.

Syntax:

Property: PlugInToExecute: String;

Item Object Model

1

Using the PlanSwift Object Model

1

API Calls

Delphi

Using Iltem Object Model	> Expand source
Using PlanSwift Object Model	
1	

C#

Using	Using Iltem Object Model			
1				
Using	Using PlanSwift Object Model			
1				

VB/VBA (OLE)

Using IItem Object Model		
ng PlanSwift Object Model		

Pascal Scripting (OLE)

ltem	Object Model
1	
Root	Object Model
1	

Pascal Scripting 1936

Property - ListReturnFullPath - FD

Property - ListReturnFullPath

Gets or Sets the ListReturnFullPath for this property.
Syntax:
 Property: ListReturnFullPath: Boolean;

API Calls

Delphi

Usin	g Iltem Object Model	> Expand source
Usin	g PlanSwift Object Model	
1		

C#

Using Iltem Object Model		
1		
Using PlanSwift Object Model		
1		

VB/VBA (OLE)

Usin	Using IItem Object Model			
1				
Usin	g PlanSwift Object Model			
1				

Pascal Scripting (OLE)

ltem	Object Model
1	
_	
Root	Object Model
1	

Pascal Scripting 1938 Property -- Name - FD

Property -- Name

Gets or Sets the Name property for the item.
Syntax: Property: Name: String;

API Calls

Delphi

Item Object Model		
1		
Using	Using the PlanSwift Object Model	
1		

Using Iltem O	bject Model	Expand source	
Using PlanSw	Using PlanSwift Object Model		
1			

C#

Usiı	ng Iltem Object Model
1	
Usi	ng PlanSwift Object Model
1	

VB/VBA (OLE)

Usin	Using Iltem Object Model	
1		
Usin	g PlanSwift Object Model	
1		

Pascal Scripting (OLE)

Item	Item Object Model	
1		
Deet		
ROOL	Object Model	
1		

Pascal Scripting
1940

Property - CompileDenyOLE - FD

Property - CompileDenyOLE

Gets or Sets the CompileDenyOLE attribute for this property. Syntax:

Property: IPropertyObject.CompileDenyOLE: Boolean;

API Calls

Delphi

Using Iltem Object Model		
Using PlanSwift Object Model		

C#

Using IItem Object Model	Expand source	
Using PlanSwift Object Model		
1.		

VB/VBA (OLE)

Usin	g Iltem Object Model
1	
Usin	g PlanSwift Object Model
1	
1	

Pascal Scripting (OLE)

Item	Item Object Model	
1		
Deet		
KUUL	Root Object Model	
1		

ltem	Item Object Model	
1		
Using the PlanSwift Object Model		
1		

Scripting - Functions -- Xfer from Freshdesk

Scripting - Functions – Xfer from Freshdesk

Functions - New Label - FD

Functions - New Label

Creates and returns a new TLabel object and sets the Left, Top and Caption properties. Do not attempt to destroy or free labels created with NewLabel.

Declaration:

<!--startsyntax-->Function: NewLabel(Left, Top: Integer; Caption: String): TLabel;<!--endsyntax-->

API Calls

Delphi

Scripting

> Expand source

Functions - NewForm - FD

Functions - NewForm

Create a new TForm object and sets the width, height, and caption as specified. Do not attempt to destroy or free forms created with NewForm.

Syntax:

Function: NewForm(Width, Height: Integer; Caption: String): TForm;

Code Reference:

- 1. Navigate to Plugin Store->Tool Manager and create a new Plugin
- 2. Set the plugin type to Script Code and open the Editor
- 3. Copy Code into the editor
 4. Press run

API Call

Delphi

Scripting

> Expand source

Functions - NewComboBox - FD

Functions - NewComboBox

Creates a new TComboBox and sets the Left, Top and Text Properties as specified. Do not attempt to destroy or free a TComboBox created with NewComboBox. Syntax:

Function: NewComboBox(Left, Top: Integer; Text: String): TComboBox

Code Reference:

- 1. Navigate to Plugin Store->Tool Manager and create a new Plugin
- 2. Set the plugin type to Script Code and open the Editor
- 3. Copy Code into the editor
 4. Press run

API Calls

Delphi

Scripting

Property - NewCheckBox - FD

Property - NewCheckBox

Creates a new TCheckBox and sets the Left, Top, Caption and Checked Properties as specified. Do not attempt to destroy or free a TCheckBox created with NewCheckBox. Syntax:

Function: NewCheckBox(Left, Top: Integer; Caption: String; Checked: Boolean): TCheckBox;

Code Reference:

- 1. Navigate to Plugin Store->Tool Manager and create a new Plugin
- 2. Set the plugin type to Script Code and open the Editor
- 3. Copy Code into the editor4. Press run

API Call

Delphi

Scripting

Functions - NewButton - FD

Functions - NewButton

Creates a new TButton and sets the Left, Top, Caption and ModalResult Properties as specified. Do not attempt to destroy or free a TButton created with NewButton. Syntax:

Function: NewCheckBox(Left, Top: Integer; Caption: String; Checked: Boolean): TCheckBox;

Code Reference:

- 1. Navigate to Plugin Store->Tool Manager and create a new Plugin
- 2. Set the plugin type to Script Code and open the Editor
- 3. Copy Code into the editor
 4. Press run

API Calls

Delphi

Scripting

Functions - NewEdit - FD

`Functions - NewEdit

Creates a new TEdit and sets the Left, Top and Text Properties as specified. Do not attempt to destroy or free a TEdit created with NewEdit.

Syntax:

Function: NewEdit(Left, Top: Integer; Text: String): TEdit;

Code Reference:

- 1. Navigate to Plugin Store->Tool Manager and create a new Plugin
- 2. Set the plugin type to Script Code and open the Editor
- 3. Copy Code into the editor
 4. press run

API Calls

Delphi

Scripting

Property - NewRadioButton - FD

Property - NewRadioButton

Creates a new TRadioButton and sets the Left, Top, Caption and Checked Properties as specified. Do not attempt to destroy or free a TRadioButton created with NewRadioButton.

Syntax:

Function: NewRadioButton(Left, Top: Integer; Caption: String; Checked: Boolean): TRadioButton;

Code Reference:

- 1. Navigate to Plugin Store->Tool Manager and create a new Plugin
- 2. Set the plugin type to Script Code and open the Editor
- 3. Copy Code into the editor
- 4. Press run

API Calls

Delphi	
Scripting	Expand source

Property - NewColorBox - FD

Property - NewColorBox

Creates a new TColorBox and sets the Left, Top and Selected Properties as specified. Do not attempt to destroy or free a TColorBox created with NewColorBox. Syntax:

Function: NewColorBox(Left, Top: Integer; Selected: Integer): TColorBox;

Code Reference:

- 1. Navigate to Plugin Store->Tool Manager and create a new Plugin
- 2. Set the plugin type to Script Code and open the Editor
- 3. Copy Code into the editor
 4. Press run

API Calls

Delphi

Scripting

Functions - Math Functions -- Xfer from Freshdesk

Functions - Math Functions - Xfer from Freshdesk

Math Functions - RoundUp - FD

Math Functions - RoundUp

Rounds the given Val up to the nearest integer.

Declaration:

RoundUp(Val: Double): Integer

Source Code

Delphi

Scripting

Math Functions - ParallelLine - FD

Math Functions - ParallelLine

Calculates the points to form a new line parallel line offset Distance from the original, specified by p1 and p2 then returns the new points in variables p3 and p4.

Declaration:

ParallelLine(p1x, p1y, p2x, p2y, Distance: Double; var p3x: Double; var p3y: Double; var p4x: Double; var p4y: Double);

Source Code

Delphi

Scripting

Math Functions - RoundToNearest - FD

Math Functions - RoundToNearest

Rounds the given value down to the nearest value as defined by precision.

Declaration:

RoundToNearest(Val, Precision: Double): Double;

Source Code

Delphi

Scripting

Math Functions - DistanceFromLine - FD

Math Functions - DistanceFromLine

Returns the distance of a point given by *p3* is from a line, given by *p1* and *p2*.

Declaration:

DistanceFromLine(p1x, p1y, p2x, p2y, p3x, p3y: Double): Double;

Source Code

Delphi

Scripting

Math Functions - DistanceBetweenPoints - FD

Math Functions - DistanceBetweenPoints

Returns the distance between 2 points specified by p1 and p2 coordinates.

Declaration:

DistanceBetweenPoints(p1X, p1Y, p2X, p2Y: Double): Double;

Source Code

Delphi

Scripting

Math Functions - GetIntersectPoint - FD

Math Functions - GetIntersectPoint

Calculates at what point Line 1, given by p1 and p2 intersects with Line 2, given by p3 and p4 and returns the result point in p5. Returns 1 (True) if the lines intersect or 0 (False) if the lines are parallel.

Declaration:

GetIntersectPoint(p1x, p1y, p2x, p2y, p3x, p3y, p4x, p4y: Double; var p5x: Double; var p5y: Double): Integer;

Source Code

Delphi

Scripting

Math Functions - RoundDown - FD

Math Functions - RoundDown

Rounds the given Val down to the nearest integer.

Declaration:

RoundDown(Val: Double): Integer

Source Code

Delphi

Scripting

Math Functions - Pi - FD

Math Functions - Pi

Returns the numeric value for Pi (3.1415926535897932384626433832795).

Syntax: Pi: Double;

Source Code

Delphi

Scripting

Math Functions - DecToEnglish - FD

Math Functions - DecToEnglish

Converts a given dimension into its string representation.

Syntax:

Function: DecToEnglish(Feet: Double): String;

Code Reference:

- 1. Navigate to Plugin Store->Tool Manager and create a new Plugin
- 2. Set the plugin type to Script Code and open the Editor
 3. Copy Code into the editor
 4. press run

API Calls

Delphi

Scripting

Math Functions - ExtendLine - FD

Math Functions - ExtendLine	
	Calculates the points to extend a line given by p1 and p2 a given Distance then returns the new points in variables p3 and p4.
Declaration:	
	ExtendLine(p1x, p1y, p2x, p2y, Distance: Double; var p3x: Double; var p3y: Double; var p4x: Double; var p4y: Double);
Source Code	
Delphi	
Scripting	> Expand source

Math Functions - AngleBetweenPointsUnScaled - FD

Math Functions - AngleBetweenPointsUnScaled

Returns the angle between 2 points given by p1 and p2 coordinates.

Declaration:

AngleBetweenPointsUnScaled(p1X, p1Y, p2X, p2Y: Double): Double;

Source Code

Delphi

Scripting

Math Function - Min - FD

Math Function - Min

Returns the smaller of the values passed. Declaration:

Min(Value1, Value2: Double): Double;

Source Code

Delphi

Scripting

Math Function - Max - FD

Math Function - Max

Returns the larger of the values passed. Declaration:

Max(Value1, Value2: Double): Double;

Source Code

Delphi

Scripting

Math Functions - PointOnAngle - FD

Math Functions - PointOnAngle

Calculates a new point given by p1 a given Distance and Angle then returns the result point in p2.

Declaration:

PointOnAngle(p1x, p1y, Angle, Distance: Double; var p2x: double; var p2y: double);

Source Code

Delphi

Scripting

Math Functions - Procedures -- Xfer from Freshdesk

Math Functions - Procedures - Xfer from Freshdesk

Procedures - TrimToArea - FD

Procedures - TrimToArea

Declaration:

TrimToArea(AreaPath, SegmentPath: String);

Source Code

Delphi

Scripting

Trims the ends of a Segment object to the boundaries of the given Area object.

Update Method - Functions -- Xfer from Freshdesk

Update Method - Functions -- Xfer from Freshdesk

Update Method - CurrentVersion - FD

Update Method - CurrentVersion

Returns the current versions of Planswift. Declaration: *CurrentVersion: String;*

Source Code

Delphi

Scripting

Update Methods - Procedures -- Xfer from Freshdesk

Update Methods - Procedures -- Xfer from Freshdesk

Update Methods - EndUpdate - FD

Update Methods - EndUpdate

Ends the temporary suspension of program updates.

Declaration: EndUpdate;

Source Code

Delphi

Scripting

Update Methods - BeginFormulaUpdate - FD

Update Methods - BeginFormulaUpdate

Temporarily suspends automatic property calculations. Declaration: *BeginFormulaUpdate;*

Source Code

Delphi

Scripting

Update Methods - Begin Update - FD

Update Methods - BeginUpdate

Temporarily suspends program updates. Declaration: *BeginUpdate;*

Source Code

Delphi

Scripting

Update Methods - RefreshImage - FD

Update Methods - RefreshImage

Refreshes the current screen image. Same as ImageRefresh.

Declaration: RefreshImage

Source Code

Delphi

Scripting

Update Methods - NewChangeGroup - FD

Update Methods - NewChangeGroup

Creates a new program change group.

Declaration:

NewChangeGroup(AName: String);

Source Code

Delphi

Scripting

Update Methods - PostChanges - FD

Update Methods - PostChanges

Post all opened change groups to the program. Declaration: *PostChanges;*

Source Code

Delphi

Scripting

Update Methods - EndFormulaUpdate - FD

Update Methods - EndFormulaUpdate

Ends the temporary suspension of automatic property calculations. Declaration: *EndFormulaUpdate;*

Source Code

Delphi

Scripting

Update Methods- ImageRefresh - FD

Update Methods - ImageRefresh

Refreshes the current screen image. Same as RefreshImage.

Declaration:

ImageRefresh;

Source Code

Delphi

Scripting

Windows Controls - Functions -- Xfer from Freshdesk

Windows Controls - Functions -- Xfer from Freshdesk

Windows Controls - FindWindow - FD

Windows Controls - FindWindow

Finds a window based on the given criteria. Returns the window handle if successful or 0 if the window is not found.

Contains, Excludes and Exact are optional.

Declaration:

FindWindow(StartsWith, Contains, Excludes: String; Exact: Boolean): Integer;

Source Code

Delphi

Scripting

Windows Controls - FocusWindow - FD

Windows Controls - FocusWindow

Gives focus to the window given by Hwnd.

Declaration:

FocusWindow(Hwnd: Integer);

Source Code

Delphi

Scripting

Windows Controls - Procedures -- Xfer from Freshdesk

Windows Controls - Procedures -- Xfer from Freshdesk

Windows Controls - Send Key - FD

Windows Controls - Send Key

Sends the given KeyCode to the active PlanSwift control.

Declaration: SendKey(AKey: Integer);

Source Code

Delphi

Scripting

Windows Controls - SendKeys - FD

Windows Controls - SendKeys

Declaration:

SendKeys(AKeys: String);

Source Code

Delphi

Scripting

Sends a string of keystrokes to the active PlanSwift control. Same as TypeKeys

User Input - Functions -- Xfer from Freshdesk

User Input - Functions -- Xfer from Freshdesk

User Input - ResultPointV2 - FD

User Input - ResultPointV2

Returns the y2 coordinate from the last Getline or GetRect.

Declaration: ResultPointY2: Double;

Source Code

Delphi

Scripting

User Input - GetPoint - FD

User Input - GetPoint

GetPoint prompts the user to select a point by clicking on the active plan, the returns the point coordinates in X and Y.

If the user clicks a valid point, the result is 1 (True), otherwise the result is 0 (False).

Declaration:

GetPoint(Var X: Double; Var Y: Double; Hint: String): Integer;

Source Code

Delphi

Scripting

User Input - ResultPointX2 - FD

User Input - ResultPointX2

Returns the x2 coordinate from the last Getline or GetRect.

Declaration: ResultPointX2: Double;

Source Code

Delphi

Scripting

User Input - GetLine - FD

User Input - GetLine

Prompts the user to click 2 points on the active plan to define a line then returns the coordinates in p1 and p2.

Returns 1 if the function is successful or 0 if the user cancels.

Declaration:

GetLine(Var p1x: double; Var p1y: double; Var p2x: double; Var p2y: double; Hint: String): Integer;

Source Code

Delphi	
Scripting	> Expand source

User Input - ResultPointX - FD

User Input - ResultPointX

Returns the x coordinate from the last Getpoint, Getline or GetRect.

Declaration: ResultPointX: Double;

Source Code

Delphi

Scripting

User Input - GetRect - FD

User Input - GetRect

Prompts the user to click 2 points on the active plan to define a rectangle then returns the coordinates in *p1 and p2*.

Returns 1 if the function is successful or 0 if the user cancels.

Declaration:

GetRect(Var p1x: double; Var p1y: double; Var p2x: double; Var p2y: double; Hint: String): Integer;

API Call

Delphi	
Scripting	Expand source

User Input - ResultPointY - FD

User Input - ResultPointY

Returns the y coordinate from the last Getpoint, Getline or GetRect.

Declaration: ResultPointY: Double;

Source Code

Delphi

Scripting

Items - Functions -- Xfer from Freshdesk

Items - Functions -- Xfer from Freshdesk

Items - SelectedPage - FD

Items - SelectedPage

Returns the full path to the currently selected page. If no page is selected and an empty string is returned.

Declaration: SelectedPage: String;

Source Code

Delphi

Scripting

Items - IsType - FD

Items - IsType

Declaration:

IsType(ItemPath, Type: String): Integer;

Source Code

Delphi

Scripting

Returns 1 (True) if the item is of type given, otherwise returns 0 (False).

Items - ChildItem - FD

Items - ChildItem

Returns the full path of the child item at position *Index* in the list. If the child item does not exist an empty string is returned. Declaration:

ChildItem(ItemPath: String; Index: Integer): String;

Source Code

Delphi

Scripting

Items - ParentItem - FD

Items - ParentItem

Returns the parent item for the given item. If the function fails an empty string is returned.

Declaration:

ParentItem(ItemPath: String): String;

Source Code

Delphi

Scripting

Items - StartRecording - FD

Items - StartRecording

ItemPath is optional. If provided, ItemPath must be a digitizer object.

If is omitted, PlanSwift will attempt to record the currently selected item, if any.

Returns 1 (True) if successful, otherwise returns 0 (False).

Declaration:

StartRecording(ItemPath: String): Integer;

Source Code

Delphi

Scripting

Items - MoveltemTo - FD

Items - MoveltemTo

Returns True if the given item is successfully moved to a new parent item.

MoveAction is optional, can be Above, Below or IntoTop otherwise will default to IntoBottom.

Declaration:

MoveItemTo(ItemPath, NewParent, MoveAction: String): Boolean;

Source Code

Delphi

Scripting

Items - SelectedItem - FD

Items - SelectedItem

Returns the full path to the currently selected item If no item is selected an empty string is returned. Declaration: SelectedItem: String;

Source Code

Delphi

Scripting

Items - NewItem - FD

Items - NewItem

Creates a new child item for the given item. ItemType is optional and allows you to set the type of item to create. Name is optional, sets the name for the new child item.

Declaration:

NewItem(ItemPath, ItemType, Name: String): String

Source Code

Delphi

Scripting

Items - ChildCount - FD

Items - ChildCount

Returns the number of child items for the item.

Declaration:

ChildCount(ItemPath: String): Integer;

Source Code

Delphi

Scripting

Items - DeleteItem - FD

Items - DeleteItem

Deletes the given item from the system. Returns 1 (True) is successful, otherwise 0 (False).' Declaration:

DeleteItem(ItemPath: String): Integer;

Source Code

Delphi

Scripting

Items - Procedures -- Xfer from Freshdesk

Items - Procedures -- Xfer from Freshdesk

Items - ShowLabel - FD

Items - ShowLabel

Sets the visibility of an items label. Declaration:

ShowLabel(ItemPath: String; Visible: Boolean);

Source Code

Delphi

Scripting

Sections - Functions -- Xfer from Freshdesk

Sections - Functions -- Xfer from Freshdesk

Sections - PointCount - FD

Sections - PointCount

Returns the number of points recorded for the section.

Declaration:

PointCount(ItemPath: String): Integer;

Source Code

Delphi

Scripting

Sections - NewSection - FD

Sections - NewSection

Adds a new section to a digitized type item and returns the full path to the new section.

If ParentPath does not exist, or is not a digitzer item, this function fails and returns an empty string. SectionName is optional.

Declaration:

NewSection(ParentPath, SectionName: String): String;

Source Code

Delphi		
Scripting	> Expand sou	urce

Sections - PointX - FD

Sections - PointX

Returns the X coordinate of the point given by *Index If this function fails the return value is -1.* Declaration:

PointX(ItemPath: String; Index: Integer): Double;

Source Code

Delphi
Scripting Expand source

Scripting - PointY - FD

Scripting - PointY

Returns the Y coordinate of the point given by *Index* If this function fails the return value is -1. Declaration:

PointY(ItemPath: String; Index: Integer): Double;

Source Code

Delphi

Scripting

Sections - Procedures -- Xfer from Freshdesk

Sections - Functions -- Xfer from Freshdesk

Sections -SetPoint - FD

Sections - SetPoint

Sets the X, Y coordinates of the given point. Declaration:

SetPoint(ItemPath: String; Index: Integer; X, Y: Double);

Source Code

Delphi

Scripting

Sections - AddPoint - FD

Sections - AddPoint

Adds a new point given by X, Y to the item.

ItemPath must specify an existing digitizer object or the procedure fails.

Declaration:

AddPoint(ItemPath: String; X, Y: Double);

Source Code

Delphi

Scripting

Sections - DeletePoint - FD

Sections - DeletePoint

Deletes the point at position Index.

Declaration:

DeletePoint(ItemPath: String; Index: Integer);

Source Code

Delphi

Scripting

Properties - Functions -- Xfer from Freshdesk

Properties - Functions -- Xfer from Freshdesk

Properties - GetPropertyResult - FD

Properties - GetPropertyResult

Returns the calculated result of the given property as a variant.

Declaration:

GetPropertyResult(ItemPath, PropertyName: String): Variant;

Source Code

Delphi

Scripting

Properties - GetResultAsInteger - FD

Properties - GetResultAsInteger

Returns the calculated result of the given property.

Default is an optional return value for the function in case of failure. If Default is not provided it defaults to 0

Declaration:

GetResultAsInteger(ItemPath, PropertyName: String; Default: Integer): Integer;

Source Code

Delphi

Properties - GetPropertyAttributeList - FD

Properties - GetPropertyAttributeList

Returns a Name=Value list of the given property attributes.

Declaration:

GetPropertyAttributeList(Itempath, PropertyName: String): String;

Source Code

Delphi

Scripting

Properties - GetPropertyFormula - FD

Properties - GetPropertyFormula

Returns the formula for the given property.

Declaration:

GetPropertyFormula(ItemPath, PropertyName: String): String;

Source Code

Delphi

Scripting

Properties - GetPropertyCount - FD

Properties - GetPropertyCount

Returns the number of properties for the given item.

Declaration:

GetPropertyCount(ItemPath: String): Integer;

Source Code

Delphi

Scripting

Properties - GetPropertyName - FD

Properties - GetPropertyName

Returns the name of the *nth* property in the propertylist.

Declaration:

GetPropertyName(ItemPath: String; Index: Integer): String;

Source Code

Delphi

Scripting

Properties - GetResultAsFloat - FD

Properties - GetResultAsFloat

Returns the calculated result of the given property.

Default is an optional return value for the function in case of failure. If Default is not provided it defaults to 0.

Declaration:

GetResultAsBoolean(ItemPath, PropertyName: String; Default: Double): Double;

Source Code

Delphi	
Scripting	Expand source

Properties - GetResultAsBoolean - FD

Properties - GetResultAsBoolean

Returns the calculated result of the given property.

Default is an optional return value for the function in case of failure. If Default is not provided it defaults to FALSE.

Declaration:

GetResultAsBoolean(ItemPath, PropertyName: String; Default: Boolean): Boolean;

Source Code

Delphi

Scripting

Properties - GetResultAsString - FD

Properties - GetResultAsString

Returns the calculated result of the given property.

Default is an optional return value for the function in case of failure. If Default is not provided it defaults to an empty string.

Declaration:

GetResultAsString(ItemPath, PropertyName: String; Default: String): String;

Source Code

Delphi

Scripting

Properties - GetPropertyAttribute - FD

Properties - GetPropertyAttribute

Returns the value of the given Item Property Attribute

Declaration:

GetPropertyAttribute(Itempath, PropertyName, AttributeName: String): String;

Source Code

Delphi

Scripting

Properties - Procedures -- Xfer from Freshdesk

Properties - Procedures -- Xfer from Freshdesk

Properties - SetPropertyAttribute - FD

Properties - SetPropertyAttribute

Attempts to set the Item Property Attribute to the given Value.

Declaration:

SetPropertyAttribute(ItemPath, PropertyName, AttributeName, Value: String);

Source Code

Delphi

Scripting

Properties - Set PropertyFormula - FD

Properties - SetPropertyFormula

Sets the given property to the value specified if possible.

Declaration

SetPropertyFormula(ItemPath, PropertyName: String; Value: Variant; Type: String);

Properties - Delete Property - FD

Properties - Delete Property

Deletes a property from the item.

Declaration:

DeleteProperty(ItemPath, PropertyName: String);

Source Code

Delphi

Scripting

Misc - Functions -- Xfer from Freshdesk

Misc - Functions -- Xfer from Freshdesk

Misc - ExecuteScript - FD

Misc - ExecuteScript

Executes the script property PropertyPath and returns the result as a variant.

Paramx is the optional string parameters to pass to the script. Failure to pass required parameters could lead to errors or failure. All scripts should check for invalid parameters and exit gracefully.

Declaration:

ExecuteScript(PropertyPath, param1, param2, param3, param4, param5, param6, param7, param8, param9: String;): Variant;

Source Code

Delphi	
Scripting	> Expand source

Misc - Current User - FD

Misc - Current User

Returns the username of the current user. Declaration: *CurrentUser: String*

Source Code

Delphi

Scripting

Dialogs - Function -- Xfer from Freshdesk

Dialogs - Function -- Xfer from Freshdesk

Dialogs - EditItem - FD

Dialogs - EditItem

Loads the given item into the Item Editor then displays to the user for editing.

If ItemPath does not exist or if the user cancels the dialog the function fails and returns False.

Declaration:

EditItem(ItemPath: String): Boolean;

Source Code

Delphi	
Scripting	> Expand source

Dialogs - ScriptMessageDialog - FD

Dialogs - ScriptMessageDialog

Declaration: ScriptMessageDialog

Source Code

Delphi

Scripting

Dialogs - Message Dialog - FD

Dialogs - Message Dialog Declaration:

MessageDialog();

Dialogs - SelectItemDialog - FD

Dialogs - SelectItemDialog

Displays the PlanSwift ItemDialog with specified parameters that were passed as arguments.

Declaration:

SelectItemDialog(AHeader: String = "; ACaption: String = "; RootItem: String = ");

Source Code

Delphi

Scripting

Dialogs - Procedures -- Xfer from Freshdesk

Dialogs - Procedures -- Xfer from Freshdesk

Dialogs - EditScriptProperty - FD

Dialogs - EditScriptProperty

Loads the specified script property into the script editor and displays to the user for editing.

Declaration:

EditScriptProperty(ItemPath, PropertyName: String);

Source Code

Delphi

Scripting

Dialogs - Objects -- Xfer from Freshdesk

Dialogs - Objects -- Xfer from Freshdesk

Dialogs - CustomDialogs - FD

Dialogs - Custom Dialogs

One of the great new features in PlanSwift9 is the ability to create reusable dialogs using stored items and properties, simply design an item with only the desired properties set as *Input*.

Source Code:

Delphi

Scripting

PlanSwift SDK -- Xfer from Freshdesk

PlanSwift SDK -- Xfer from Freshdesk

PlanSwift9 SDK - FD

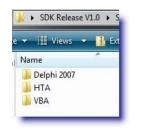
PlanSwift9 SDK

Description:

PlanSwift9 SDK Version 1.2 is now available. This SDK contains the necessary source code for Delphi, Excel, C#, VBA, and HTA. There are several examples for you to use.

What's in the package:

The zip file contains 3 folders:



Delphi Source Code:

Name	Туре
excelform	Delphi Form
🔀 excelform	Delphi Source File
excelform.dcu	DCU File
🚰 main	Delphi Form
🔭 main	Delphi Source File
main.dcu	DCU File
🔀 PlanSwift_TLB	Delphi Source File
PlanSwift_TLB.dcu	DCU File
readme	Text Document
🛅 SDK_Demo	Delphi Project File
🛅 SDK_Demo	Delphi Project File
SDK_Demo	Application
SDK_Demo.dproj.local	LOCAL File
SDK_Demo.identcache	IDENTCACHE File
SDK_Demo.res	RES File
Work Order.dotx	DOTX File

Sample HTA

Туре
HTML Application

Sample Excel Addin

Name	Туре				
📲 PlanSwift	Microsoft Office Excel Ad				
Readme	Text Document				

2045 Development Archives - OLE Automation Manual

• •What is COM? What is OLE Automation?

What is COM?

Component Object Model (COM) is an interface standard for software componentry introduced by Microsoft in 1993. It is used to enable interprocess communication and dynamic object creation in any programming language that supports the technology. The term COM is often used in the software development industry as an umbrella term that encompasses the OLE, OLE Automation, and ActiveX, COM+DCOM technologies.

The essence of COM is a language-neutral way of implementing objects that can be used in environments different from the one they were created in, even across machine boundaries. For well-authored components, COM allows reuse of objects with no knowledge of their internal implementation, as it forces component implementers to provide well-defined interfaces that are separate from the implementation. The different allocation semantics of languages are accommodated by making objects responsible for their own creation and destruction through reference-counting. Casting between different interfaces of an object is achieved through the QueryInterface() function. The preferred method of inheritance within COM is the creation of sub-objects to which method calls are delegated.

Although the interface standard has been implemented on several platforms, COM is primarily used with Microsoft Windows. COM is expected to be replaced at least to some extent by the Microsoft .NET framework, and support for Web Services through the Windows Communication Foundation (WCF). However, COM objects can still be used with all .NET languages without problems. Networked DCOM uses binary proprietary formats, while WCF encourages the use of XML-based SOAP messaging. COM is very similar to other component software interface standards, such as CORBA and Java Beans, although each has its own strengths and weaknesses. It is likely that the characteristics of COM make it most suitable for the development and deployment of desktop applications, for which it was originally designed.

What is OLE Automation?

In Microsoft Windows applications programming, OLE Automation (later renamed by Microsoft to just Automation, although the old term remained in widespread use), is an inter-process communication mechanism based on Component Object Model (COM) that is intended for use by Scripting Languages -originally Visual Basic, but now many languages that run on Windows. It provides an infrastructure whereby applications called automation controllers can access and manipulate (i.e. set properties of or call methods on) shared automation objects that are exported by other applications. It supersedes Dynamic Data Exchange (DDE), an older mechanism for applications to control one another. As with DDE, in OLE Automation the automation controller is the "client" and the application exporting the automation objects is the "server".

Enabling the AllowManualActivation Property

The "Enable Manual Activations" checkbox in the license manager has been hidden so that it is no longer available to the user. Manual Activation may still be enabled through U-T-H Settings.

To enable manual activation:

- 1. Enable U-T-H (Under the Hood)
- 2. Select U-T-H from the main menu, right click on "Settings" and select "Properties"
- 3. In the Properties Form, click the "Advanced" button.
- 4. In the "Other" section, near the bottom (before "Sales Tax 1" and "Sales Tax 2"), look for a property named "AllowManualActivation".

Name:	AllowManualActivation	
Type:	CheckBox	Expression
Group:	Other	*
Fool Hint:		
	Remember Value	Parse Formula

- 5. If this property exists, check the checkbox to set its value to "True".
- 6. If this property does not exist, create a new property called "AllowManualActivation", with a Type of "Checkbox" and a Group of "Other".

Save the new property and then check the checkbox to set its value to "True".

SalesTax1	0	%	0.00
SalesTax2	0	%	0.00
AllowManualActivation	V		True

7. Manual activation should now be enabled.

Contact Us

To contact PlanSwift:

For existing customers, give us a call at888-752-6794, option 1. If you're calling about a simple issue such as activation, one of our customer care specialists can take of you right away. If your issue is more involved, we'll open a case and have a technical support rep follow up. We work all cases in the order they are received - follow up times vary but we will get back to you as soon as possible

If you are interested in purchasing PlanSwiftgive us a call at888-752-6794, option 2. One of our amazing Sales Team members will be happy to help you.

You can send an e-mail toupport@planswift.comPlease include your name, phone number, company name, and a detailed description of your issue. If you want us to call you, let us know the best times to reach you Note: to ensure that you receive our response, please addupport@planswift.comto your email contact list or safe senders list.)





Related articles

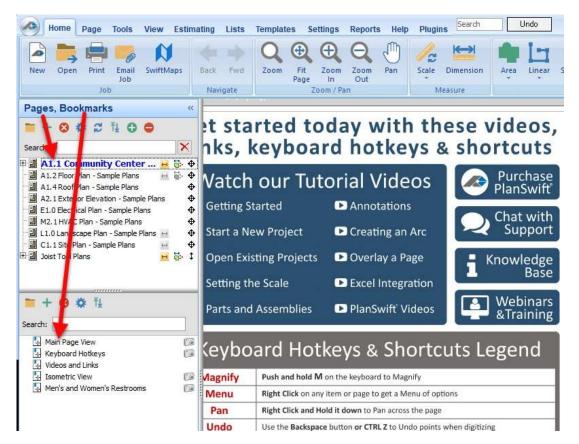
E Contact Us

Draft/Needs Update

- (Missing Images) How To: Add a Bookmark
- (Needs Images + Updated Link) Online Forum
- Area Pro (Check Links)
- Concrete for Budget Takeoff (check links)
- Digitizing or Moving Beyond the Current View (Zoom) (Needs Images)
- Drag and Drop Folders from Templates to Estimating (Needs Images/Video)
- Export to MS Project for PlanSwift version 9 (Check Links)
- How To: Add Attachments to a Project. (Needs Images)
- How To: Create Advanced Parts Using Expressions (Needs Updates)
- How To: Use the Fill Down and Fill With (Needs Images)
- Import a Job from a PlanSwift E-Mail
- Multiline Tool (Needs Image)
- Painting (Check Links)
- Plumbing Standard (Check Links)
- Select and Set (Check Links)
- Shape Stamper (Check Links)
- Subtracting a Section (Needs images)
- Switch Pages from Takeoff Summary (Needs Images)

(Missing Images) How To: Add a Bookmark

Bookmarks provide an easy way to quickly move to pages and even specific areas of pages. By default, PlanSwift docks the Bookmarks to the Pages window. The Pages window is at the top, the Bookmarks window below that.

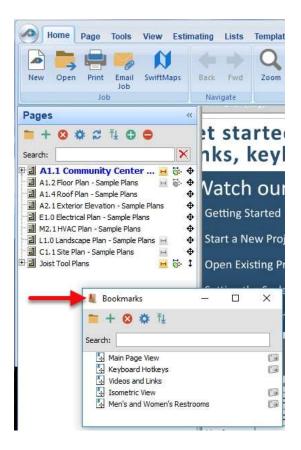


If you wish, you can undock the Bookmarks window by right-clicking the Pages, Booksmarks window header, clicking on Bookmarks, then clicking on Undock.

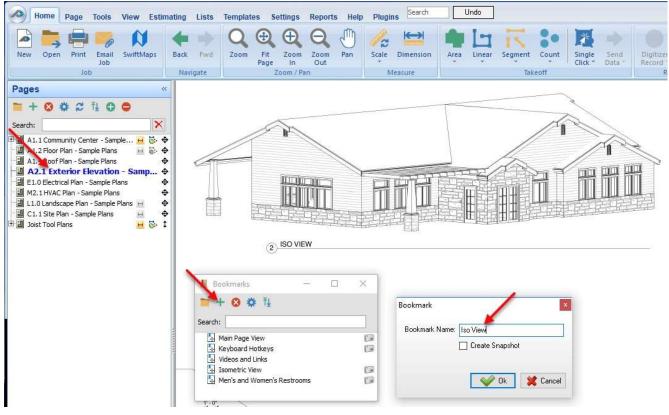
Home Page Tools V	iew Estimatin) Lists	Templates	Settings R
New Open Print Email S Job	wiftMaps Bar	/	Zoom Fit Pag	e /
Pages, Bookmarks	« Pages	avy ate		Zom/Pan
► + 🛛 🌞 🞜 № 🖸 Search:	Bookmarks Reset Layout	>	Undock Dock Left	ľ
Image: All and the second se		atcl	Dock Right Dock With	
A2.1 Exterior Elevation - Sample F E1.0 Electrical Plan - Sample Plans		etting S	Started	DA

🜡 Bookmarks — [) X
💳 🕂 🕴 🏘 Tł	
earch:	
🔄 Main Page View	12
🔄 Keyboard Hotkeys	1
🔄 Videos and Links	
	(4573)
Isometric View	100

The undocked window can now be dragged to wherever it is convenient for you.



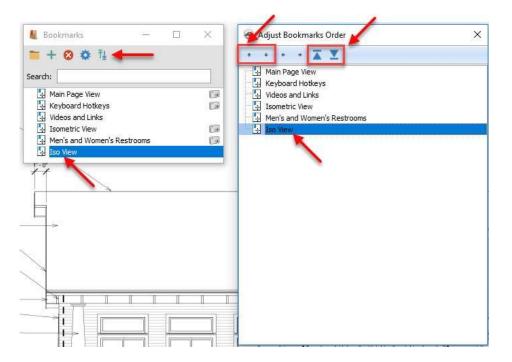
To add a **Bookmark**, first select the page from the **Pages** window that you would like to bookmark. If you want to bookmark a particular area of the page, pan to that area (right-click and drag). If you want it zoomed in on, use the mouse wheel to zoom to the level that you would like. Bookmarks automatically store both the pan and zoom settings in place at the time you create the bookmark. Bookmarks may also be organized and stored in folders. To set a bookmark, click on the green plus (+) in the **Bookmarks** window, enter the new name for the bookmark in the **Bookmark** window that opens, and click on **OK**.



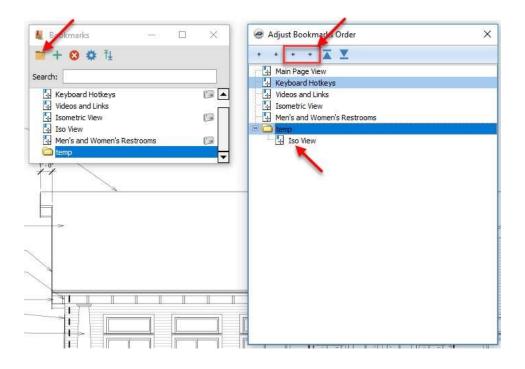
The new bookmark appears in the Bookmark window.

Bookmarks		×
<mark>≡ + ⊗ ‡</mark> ∄		
Search:		
Main Page View		13
🔄 Keyboard Hotkeys		1
🔄 Videos and Links		
Isometric View		1.4
🔄 Men's and Women's Res	trooms	120
🔄 Iso View 🗲		

If you want to move the new bookmark to a higher level, click on the bookmark, then click on the up/down arrows button. This opens the Adjust Bookmarks Order window, where you can click on any bookmark, then click on the arrow keys to move the bookmark. The two large up and down arrows on the right move the bookmark all the way to the top or bottom of the list. The two small up and down arrows on the left move the bookmark up or down one line for each click.



The left and right arrows allow you to move the bookmark into or out of any folder you can create using the folder button in the Bookmarks window.



Once the bookmark is added, simply click on it to immediately view the page (and the location on the page) saved with the bookmark.

Enter a name for the **Bookmark**, check the box for **Create Snapshot** if you want to capture a small image of the bookmarked area, then click **Ok**. To view the snapshot of the **Bookmark**, click in the camera icon to the right of the bookmark name.

Now there is a created **Bookmark** in the **Bookmark** window. Anytime a page is open, clicking the desired **Bookmark** in the list will automatically change the page to a view of that specific **Bookmark**.

(Needs Images + Updated Link) Online Forum

Visit the PlanSwift Forum for help OR Conline Forum Button 2

The PlanSwift Online Forum is your place to connect with other users of PlanSwift to share and learn new ideas. You can also connect with PlanSwift experts inside the forum to gain great information.

1. To connect with the forum from inside PlanSwift, you need to have an internet connection, then select the "PlanSwift Online Forum" button.

ļ	Join	the	Plans	Swift	forum	to	share	ideas	and	get	help	OR	Online	Forum	Selecte	ed 2

2. You will need to register to become a forum member to post questions and comments. You can register by clicking here.

Export to MS Project for PlanSwift version 9 (Check Links)

. Main Form

1.1. PlanSwift default names of properties column and

user-defined names

Description: These fields are made for multi-language support and User Defined Export of Costs and Prices (where <u>Price Each</u> or <u>Cost Each</u> = MS Project <u>Resource Standard Rate</u> and <u>Price Total</u> or <u>Cost Total</u> = MS Project <u>Task or Assignment Costs</u>). In the main case, they do not need edition.

1.2. Parts exporting to Resources options

Description: Here are five combos. They manage the way that each different Part Type is exported to MS Project. Available options are Material, Work and Cost (2007 and up).

1.3. The Logic of Export is (for now fixed L):

- 1.3.1. If Item Type is Folder, Digitizer or Assembly it will create Tasks;
- 1.3.2. If Item Type is Part, Material, Labor, Equipment, Sub or Other it will check Parts exporting options and create Resources and add Assignments(There is a checkfor resource uniqueness key is Part Type and Part Name)

1.4. The Result is a Copy of Estimating but in MS Project

BUT ONLY IF the logic of used methodology in estimating is equal to Logic of Export. In other words: in the estimating must not be used Items with Type â€^TITEMâ€[™] and Parts are Childs of â€^TTasksâ[™] and do not have their own SubChilds.

- See more at: http://www.planswift.com/phpkb/article/export-to-ms-project-for-planswift-version-9-2227.html#sthash.GGfLLnjj.dpuf

How To: Create Advanced Parts Using Expressions (Needs Updates)

In this article, we will cover how to create advanced parts using expressions.

An expression in a programming language is a combination of values, variables, operators, and functions that are interpreted (*evaluated*) according to the particular rules of precedence and of association for a particular programming language, which computes and then produces (*returns*, in a stateful environment) another value.

The expression is said to *evaluate to* that value. As in mathematics, the expression *is* (or can be said to *have*) its evaluated value; the expression is a representation of that value.

Requirements: Version 8.5 Difficulty - Moderate

This part will be a drywall item where you can select the sheet size, sheet thickness, and type of drywall. Based on the selections you make, you can determine a unit price to apply to the sheet count that is generated. This will allow you to create one complex part rather than several individual parts.

1. Click on the "New Part" button located at the top of the estimating window to add a new item.

2. Give the part a name. In this example, we will use "Drywall Template".

Tip: At this point, it is good practice to think about all of the variables you will need for your part. Map it out with paper and pencil first, this will help you to think it through before you start writing the necessary code.

1. In my example, I have determined that I need to create the following properties (shown in blue):

Sheet Width and Sheet Height (to specify the sheet size I want). Check the box to set it as an input.

Layers for Side A and Layers for Side B (this will allow me the flexibility to specify different layers for each side i.e. Side A specifies - 1 layer and side B - 2 layers). Check the box to set it as an input.

Sheet Thickness (1/4", 1/2", 5/8", etc.; set the type to Text and check the box for input).

Type (Std. Gyp, Type-X, Moisture Resistant etc.) Set the type to Text and check the box for input.

SqFt Calc (for calculating the gross Sq Ft of board)

1/4" Std Gyp, 1/2" Std Gyp, 5/8" Std Gyp, 1/4" Type-x, 1/2" Type-x, 5/8" Type-x, 1/4" MR, 1/2" MR, and 5/8" MR (for entering the price per square foot). 1/4" Price, 1/2" Price, and 5/8" Price (this is where we will write the expressions). After entering the name, change the type to Expression (Pascal).

We will also take advantage of the List feature as we create the new properties. For sheet Thickness, create a list of 1/4", 1/2", 5/8" and check the box to only select from this list. For Type, create a list of Std. Gyp, Type-X, Moisture Resistant and check the box to only select from this list. You can change the names or add more to suit your own needs; just remember to add them in to the expressions as well.

Tip: When creating new properties, I like to utilize the Group feature. You will notice I have created a group for the properties that I have set to be input items. This will not affect the calculations in any way but just keeps things organized.

1. Start with the basic calculations first.

Let's do the Gross Square footage first. You can see we have used ([Area]* ([Layers Side A] + [Layers Side B])). This is giving you the correct square footage to start the calculation.

Now we can build the QTY calculation - RoundUp[[Gross SqFT] / ([Sheet Width * Sheet Height)). This will take the gross square footage we just calculated above and divide the footage by the square footage of the sheet size. This will give us the sheet count required and round to the nearest full sheet.

The Price Each calculation is where we will use the nested property - [[Size] price] this takes the value from the property "Size" and appends it to the text " price" thus creating the property of, for example, "1/4" Price". Note: Be mindful of any spaces you create in property names as these are recognized as characters. In this example, there is a space before "price".

After [[Size] price], which is going to be the price per square foot, add *([Sheet Width]*[Sheet Height]). This will now calculate the price per square foot and multiply by the square footage of the sheet size. This will give you the price per sheet.

The Price Total is simply left as the QTY multiplied by the Price of each sheet.

2. Now for the Expressions...

Open the Edit Formula window by clicking on the ellipses button (💫) in the value field. This will open the Edit Formula window. This window allows a greater view of the formula you are creating. This is the area where everything is case sensitive.

Let's start with the IF statement - If('[Type]' = 'Std. Gyp.') then Result := [1/4" std gyp] else

This is saying if the 'Type' property is exactly equal to the value 'Std. Gyp.' then use the result that is in the '1/4" std gyp' property. If it isn't, then read the next line. This repeats until the code reads **Else Result := 0** thus if none of the criteria match, use 0.

3. You can copy this formula and paste it into the other properties for the other sizes; just edit the [1/4"...] to match the appropriate size.

4. We're almost ready to try our new part! Now just populate the price per square foot for all of the different types you have created.

5. Here is one more expression you can use if you like. This one can be set up in the **Name** property. Double click on the Name property and once again set the 'Type' to *Expression (Pascal)*.

IF ([Area] = 0) then Result := 'Drywall Template' else Result := '[Sheet Width] X [Sheet Height] X [Size] [Type]'

This states that when the [Area] is equal to 0 use 'Drywall Template' for the Name; if not, use the sheet height, width, size and type based on the selections chosen from the inputs.

With all of the calculations completed, your part is ready for use. Try it on a practice area by dragging it from the SwiftDepot window on to your Digitizer. It should open the 'Input' window and you should be able to select from your drop-down menu: the size and the type of drywall, the sheet height and width, and the number of layers on each side of the wall. If it is a ceiling, just use one of the sides.

It is always a good practice to try all of the different combinations from your newly created part to check for any errors and correct them. With the addition of the expressions, this type of dynamic part can be created for any number of building components. Drywall, concrete, metal and wood studs, pricing, names, grades, gauges. The possibilities are almost limitless!

Import a Job from a PlanSwift E-Mail

1. Open the PlanSwift email in your default email client (Outlook, Yahoo, Gmail, etc...) The email should look something like the information listed below.

A PlanSwift user has sent you some files.

- Click the following link to download:
- http://share.planswift.com/download/?file=J86E1SS3-7AN8-HD4S-C06A-4LIVW6VBBW5 This file will be available for download for 30 days from today (expiring 04/08/11).

PlanSwift is the #1 takeoff and estimating software, and it comes with powerful, yet easy to use, on-screen digitizer and takeoff tools. Discover how much time you can save with PlanSwift. Available for download at http://www.planswift.com/requesttrial

- 2. Click the first link in the email.
- 3. A dialog box will appear asking you to save the file. Save the file to your desktop or location of your choice.
- 4. Open PlanSwift, if it is not already open. (If you do not have a copy of PlanSwift, click here).
- 5. Click on the tab named Other and click the Unzip SwiftJob button.
- 6. The Open dialog will appear. Select the file you saved in step 3 and press Open.
- 7. The Import dialog will appear. Select your storage location (the default location is your local computer).
- 8. Name the job (IMPORTANT: rename the job to avoid duplicates).
- 9. Press Ok.

This will now start the import process. Congratulations you have just imported your first job!

How To: Add Attachments to a Project. (Needs Images)

To add attachments to your project, start by opening the attachments sidebar window in PlanSwift.

- 1. Allows you to create a New Folder for organizing attachments.
- 2. Allows you to add attachments by opening Windows Explorer. Simply select the desired files in Windows Explorer, and click Open.
- 3. Allows you to delete any files or folders in the **Attachments** sidebar.
- 4. Allows you to view and edit basic and advanced properties regarding the selected Attachment. For more information about advanced properties, click hereto watch the video.
- 5. Directional arrows allow you to organize the attachments by moving them up, down, left or right. Note: Moving files left or right is only useful when placing or removing files from folders.

There is also the option of adding desired attachments to the Attachment sidebar window simply via drag and drop.

If you have multiple attachments for one job, you have the option of searching for a specific attachment or file via the Search bar located at the top of the Attachment sidebar window.

Subtracting a Section (Needs images)

To subtract a section from an already existing digitized area, first select the digitized area you will be editing. Once selected, the perimeter of the digitized area should be highlighted red.

Then, right click the digitized area and select 'Subtract from Section'.

From there, continue to digitize the area you would like to remove. Please note that you must stay within the bounds of the area you are subtracting from.



Drag and Drop Folders from Templates to Estimating (Needs Images/Video)

This feature was added into version 9.3. It allows for the user to drag a folder from the template window and drop it and its contents into the Estimate window.

This function greatly improves the user's ability to create specific job templates of takeoff items and parts specific to a construction sector such as, but not limited to:

- Commercial
- Residential
- Multi-Family Hotel
- •

Note: This approach is more streamlined than copying and pasting from the Templates tab.

Switch Pages from Takeoff Summary (Needs Images)

Available in 9.3+.

- 1. In case you are not in Takeoff Summary, click Takeoff Summary.
- 2. Select the drop-down box at the top of the Takeoff Summary sidebar.
- 3. Select the page you wish to switch to.

Takeoff Summary Page Switch

Digitizing or Moving Beyond the Current View (Zoom) (Needs Images)

Hover Scrolling - Wheel Mouse Pan - Zoom / Pan Tool - Scroll Bars

Hover Scrolling:

- 1. As your mouse nears the top, right, bottom, or left of the plan/takeoff screen you will notice two transparent blocks of blue (see the Hover Scroll image below). The darker of the two has an arrow in it.
- 2. Hovering (not clicking) your mouse in the darker blue box will make the plan scroll quickly in the direction of the arrow.
- 3. Hovering your mouse in the lighter blue box will make the plan scroll slowly in the direction of the arrow.
- 4. While scrolling you may press the Spacebar to reverse the direction of the scroll (see the Hover Scroll after Spacebar image below.) This is useful if you went too far and don't want to move the mouse to the other side of the screen to scroll down.

Note: If you reach the edge of the PlanSwift drawing area, you will be presented with red boxes instead of blue. This means that you cannot scroll anymore in that direction (see the Red Hover image below).

Hover Scroll:

Hover Scolling

Hover Scroll after Space Bar:

Hover Scolling - Back Red

Hover:

Red Hover - Edge of page

Wheel Mouse Pan:

- 1. Hold your wheel mouse button down and "drag" the plan.
- 2. Release the button to continue digitizing or repeat step 1 to pan some more.

Pan via the Zoom / Pan tool:

1. This tool is discussed in detail in the Zoom and Pan section.

Scroll Bars:

1. On the right and the bottom of the current view, you will see scrollbars. You can click and drag these scroll bars to move around the plan.

How To: Use the Fill Down and Fill With (Needs Images)

This article will explain how to use the Fill Down and Fill With options in the right click menus on the Estimating, Templates and Reports screen.

Requirements:

- PlanSwift 9.5.8.6
- Property must exist on the item

First, select the cells in the grid that you want to update. This can be done one of two ways. Hold the CTRL key on your keyboard and click the cells in the column that you want to update, or click the first cell, then hold the SHIFT key and select the last cell to select the range.

Right-click on one of the selected cells and select Fill With.

You will be prompted to enter a value in the dialog box.

Take note that Locked properties will not get overwritten. If your field contains a formula that you do not want over-written, it is recommended that you lock those properties. Repeat these steps to use the **Fill Down** option. The only difference it that it uses the value in the topmost selected cell to fill the other selections.

Painting (Check Links)

Painting

Painting for plugin is a quick and easy database intended for swift takeoff of Painting on a budget basis. This database covers all the major type of takeoff issues that a Painting contractor will encounter.

Plugin Sections

The database is sequenced into :

- Paint Interior Wall 1 Side
- Paint Interior Wall 2 Sides
- Paint Interior Trim
- Paint Exterior Wall by LF
- Paint Exterior Wall by SF
- Paint Exterior Trim
- CMU & Concrete Specialties
- Paint Ceilings
- Paint Windows
- Paint Doors & Frames
- Paint Floors
- Paint Stairwells
- Paint Steel Shapes

*** Note, that the order of these conditions are set correctly as we have set up the data. The Planswift program will move these conditions around as you use the Plugin. This is a known issue and the fix is under way with a future Planswift upgrade.

Takeoff View:

This is a screenshot of the takeoff window. Notice the sequence of the folders on the right side.

Each folder is discussed in detail below.

Paint Wall on 1 Side

This section is to be used when you are painting a wall on one side only. There are child folders for different paint substrates such as:

- Paint Wall on 1 Side GWB
- Paint Wall on 1 Side CMU
- Paint Wall on 1 Side Concrete
- Paint Wall on 1 Side Plaster

Paint Wall on 1 Side Wood

Stain Wall on 1 Side Wood

Each sub section in turn is organized as follows. For this example we are using the GWB as an example:

Paint Wall on 1 Side GWB

Paint Wall Primer & 2 Coats on 1 Side GWB 8'

Paint Wall Primer & 2 Coats on 1 Side GWB 9'

Paint Wall Primer & 2 Coats on 1 Side GWB 10'

Paint Wall Primer & 2 Coats on 1 Side GWB 11'

Paint Wall Primer & 2 Coats on 1 Side GWB 12'

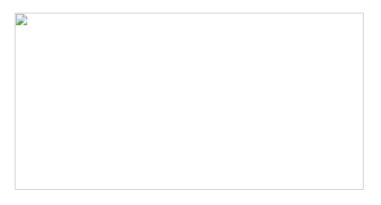
Input other height

Paint Wall Primer & 2 Coats on 1 Side GWB 15'

*** Note, that the order of these conditions are set correctly as we have set up the data. The Planswift program will move these conditions around as you use the Plugin. This is a known issue and the fix is under way with a future Planswift upgrade.

The first 5 conditions are pre-set so that you do not have to input the height of the wall. These heights are set up and the math is completed to calculate the SF of the 1 sided wall that you are to paint. As an example, if you have a 9 foot wall to takeoff select this condition, takeoff the length of the wall and this length is calculated times the height of 9 foot for you. This method will cover 85% of what the average use will encounter.

For the times you have a wall of a different height click on the "Input other Height" folder and then the conditions inside this folder. As you use this condition the 1st question you will see will be the height of the wall. This is pre-set to 15 feet. Change to reflect the actual wall height you want to see. A screen shot of this condition is as follows.



When you input the wall height the description will change to reflect the wall height of the takeoff.

Paint Wall on 2 Sides

This section is to be used when you are painting a wall on 2 sides. There are child folders for different paint substrates such as:

Paint Wall on 2 Sides GWB

Paint Wall on 2 Sides CMU

Paint Wall on 2 Sides Concrete

Paint Wall on 2 Sides Plaster

Paint Wall on 2 Sides Wood

Stain Wall on 2 Sides Wood

Each sub section in turn is organized as follows. For this example we are using the GWB as an example:

Paint Wall on 1 Side GWB

Paint Wall Primer & 2 Coats on 2 Sides GWB 8'

Paint Wall Primer & 2 Coats on 2 Sides GWB 9'

Paint Wall Primer & 2 Coats on 2 Sides GWB 10'

Paint Wall Primer & 2 Coats on 2 Sides GWB 11'

Paint Wall Primer & 2 Coats on 2 Sides GWB 12'

Input other height

Paint Wall Primer & 2 Coats on 2 Sides GWB 15'

*** Note, that the order of these conditions are set correctly as we have set up the data. The Planswift program will move these conditions around as you use the Plugin. This is a known issue and the fix is under way with a future Planswift upgrade.

The first 5 conditions are pre-set so that you do not have to input the height of the wall. These heights are set up and the math is completed to calculate the SF of the 2 sided wall that you are to paint. As an example if you have a 9 foot wall to takeoff select this condition. Takeoff the length of the wall and this length is calculated times the height of 9 foot for you time both sides of the wall. This method will cover 85% of what the average use will encounter.

For the times you have a wall of a different height click on the "Input other Height" folder and then the conditions inside this folder. As you use this condition the 1st question you will see will be the height of the wall. This is pre-set to 15 feet. Change to reflect the actual wall height you want to see.

Paint Interior Trim

This section is to be used when you are painting trim. There are child folders for different paint substrates such as:

Paint Interior Trim

Stain Interior Trim

Each sub section in turn is organized as follows. For this example we are using the Paint Interior Trim as an example:

Paint Interior Trim

Paint Primer & 2 Coats @ Crown Molding

- Paint Primer & 2 Coats @ Picture Rail
- Paint Primer & 2 Coats @ Cornice Molding
- Paint Primer & 2 Coats @ Chair Rail
- Paint Primer & 2 Coats @ Map Rail
- Paint Primer & 2 Coats @ Wall Cap
- Paint Primer & 2 Coats @ Baseboard

*** Note, that the order of these conditions are set correctly as we have set up the data. The Planswift program will move these conditions around as you use the Plugin. This is a known issue and the fix is under way with a future Planswift upgrade.

Click on the green button for the condition that you want. Input the width of the trim. For a starting point all trim is assumed to be 1 foot wide. Then takeoff the length of the trim.

The Stain Interior Trim works exactly as the paint.

Paint Exterior Wall by LF

This section is to be used when you are painting an exterior wall and taking off the wall by lineal feet. All exterior wall in this section is assumed to be a one sided wall. There are child folders for different paint substrates such as:

- Paint Exterior Wall CMU
- Paint Exterior Wall Concrete Wall
- Paint Exterior Wall EIFS Wall
- Paint Exterior Wall Metal Siding
- Paint Exterior Wall Clapboards
- Paint Exterior Wall Shingles
- Stain Exterior Wall Clapboards

Stain Exterior Wall Shingles

Each sub section in turn is organized as follows. For this example we are using the CMU as an example:

Paint Exterior Wall CMU

Paint Exterior CMU Block Filler & 2 Coats 1 Side 8'

Paint Exterior CMU Block Filler & 2 Coats 1 Side 9'

Paint Exterior CMU Block Filler & 2 Coats 1 Side 10'

Paint Exterior CMU Block Filler & 2 Coats 1 Side 11'

Paint Exterior CMU Block Filler & 2 Coats 1 Side 12'

Input other height

Paint Exterior CMU Block Filler & 2 Coats 1 Side 15'

*** Note, that the order of these conditions are set correctly as we have set up the data. The Planswift program will move these conditions around as you use the Plugin. This is a known issue and the fix is under way with a future Planswift upgrade.

The first 5 conditions are pre-set so that you do not have to input the height of the wall. These heights are set up and the math is completed to calculate the SF of the wall that you are to paint. As an example if you have a 9 foot wall to takeoff select this condition, takeoff the length of the wall and this length is calculated times the height of 9 foot for you. This method will cover 85% of what the average use will encounter.

For the times you have a wall of a different height click on the "Input other Height" folder and then the conditions inside this folder. As you use this condition the 1st

question you will see will be the height of the wall. This is pre-set to 15 feet. Change to reflect the actual wall height you want to see When you input the wall height the

description will change to reflect the wall height of the takeoff.

Paint Exterior Wall by SF

This section is to be used when you are painting an exterior wall and taking off the wall with the elevation view by square feet. This section contains all the direct conditions. There are no child folders to this section.

Paint Exterior Wall CMU Wall Block Filler & 2 Coats

Paint Exterior Wall Concrete Wall Block Filler & 2 Coats

Paint Exterior Wall EIFS Wall Primer & 2 Coats

Paint Exterior Wall Metal Siding Primer & 2 Cots

Paint Exterior Wall Clapboards Primer & 2 Coats

Paint Exterior Wall Shingles Primer & 2 Coats

Stain Exterior Wall Clapboards Sealer & 2 Coats

Stain Exterior Wall Shingles Sealer & 2 Coats

Pick the condition you wish to use, by clicking on the green button, then takeoff the area in SF of the area to be painted.

Use the Boxout function of Planswift to deduct exterior openings from the base area.

CMU & Concrete Specialties

This is a small section to be used for specialty conditions with CMU or Concrete walls. This section contains the following types of coatings.

CMU-Concrete Stain Cover

CMU-Concrete Block Filler

CMU-Concrete Elastomeric Hydro Coating CMU-

Concrete Clear Hydro Sealer

Pick the condition you wish to use, by clicking on the green button, then takeoff the area in SF of the area to be painted.

Paint Windows

This is a section to be used for the painting of windows. Windows come in a variety of sizes, materials, types, and manufactures. All of this is ignored in this section, and we are using the Window ID for each type of Window as identified by the Architect or Engineer.

Choose the window ID by number, in the properties box you can change the description to match the specifications of the window. Once this is done, then count each of this type of window.

In this section you have a choice for a window that is painted and also for a wood window that is stained.

Paint Doors & Frames

This is a small section to be used for the painting of doors. Doors are broken apart by single or double doors, the transoms and sidelights, then for specialty doors. Doors and frames are further broken apart by being hollow metal or wood. The door size is not a consideration as this makes little difference in the amount of paint, or the man hour per door leaf. The sections are as follows. Paint Single Doors

- Paint Double Doors
- Transoms Sidelights Borrow Light
- Specialty Doors

In turn, each section is further broken down into conditions for each section. For example at Paint Single Door you have the following door types to paint.

- Paint Single HM Frame & HM Door
- Paint Single HM Frame & Wood Door
- Paint Single Wood Frame & Wood Door
- Stain Single Wood Frame & Wood Door

Then in turn each of the above conditions will have separate conditions for the door and frame. This way you can adjust the cost of each as desired. At double doors the math is set up to provide the painting for 1 double frame and 2 door leafs.

Paint Ceilings

This is a section to be used for the painting of ceilings. There are no child folders in this section, as it directly contains all the conditions for the ceiling painting. Here is a screen shot of all the conditions for painting ceilings.

Paint Stairwells

This is a section to be used for the painting of stairwells. This section is broken down into the components of the stairwell starting from the inside of the exterior walls, and then of the stairwell parts. Each type of conditions has the takeoff method for the type of takeoff needed. There are no child folders in this section, as it directly contains all the conditions for the stairwell painting. Here is a screen shot of all the conditions for painting stairwells.

Paint Floors

This is a section to be used for the painting of floors. There are no child folders in this section, as it directly contains all the conditions for the floor painting. All floors are taken off by SF. These are the conditions in this section:

- Floor Power Acid Wash
- **Concrete Floor Sealer**
- **Concrete Floor Stain**
- Concrete Floor Non-Traffic Enamel
- Concrete Traffic Wear Enamel
- Concrete Floor Paint

Paint Steel Shapes

Painting Steel is much more complex than other issue. The shape of the steel determines the square foot of paint are per lineal feet of steel. All steel is calculated in rough this manner to determine the SF of the paint area.

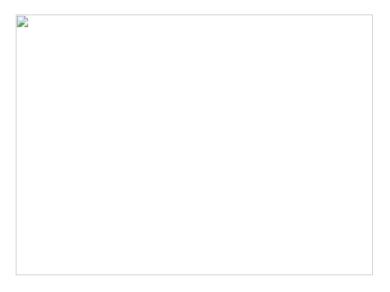
Count of steel pieces x LF of each Steel x SF per LF of Steel = SF of Paint Area

The user must input the following

LF of Each Steel Piece

- SF per LF of Steel
- Steel Weight per LF

Using the example of a W-Flange beam here is the properties box for input of this information.



The questions might vary depending on the type of steel that is being painted. Also as a user be careful on the SF per LF you set up. This will vary depending on the position of the steel in the structure.

The following steel shapes are available for takeoff in the database.

- Paint Steel W-Flange
- Paint Steel Square Tube
- Paint Steel Rectangle Tube
- Paint Steel C Channel
- Paint Steel MC Channel
- Paint Steel Tee
- **Paint Steel Angle**

Each section has detailed conditions for each major size of steel. In general the thickness of the steel is not considered. For example the steel section with the MC Channel conditions opened is show here.

Support

The developer of this PlanSwift library has worked in the construction software database industry for 15 years and works daily with estimators of all types. Database Solutions planswift@database-solutions.com

- See more at: http://www.planswift.com/phpkb/article/painting-2255.html#sthash.mDKt9TAS.dpuf

Plumbing Standard (Check Links)

Plumbing Standard for plugin is a quick and easy database intended for swift takeoff of plumbing on a budget basis. This database covers all the major type of takeoff issues that a plumbing contractor will encounter.

To Install

Open PlanSwift. Go to the "Setting" Tab Click on the "Import Plugin Package" Follow the directions provided by Planswift. After installation you will find "Plumbing Standard" in the "Templates" Tab ready for your use.

To Uninstall

Go to the plugin store tab and select Uninstall Plugins

Plugin Sections

The database is sequenced into :

Demo Domestic Water Drainage Sanitary Waste Drainage & Sanitary Misc Fixtures Plumbing Equipment Medical Gas System Vacuum System Compressed Air System

Takeoff View:

This is a screenshot of the takeoff window. Notice the sequence of the folders on the right side.

Each folder is discussed in detail below.

Demo

The Demo section provides takeoff for miscellaneous issues that a plumbing contractor will encounter such as demolitions and removal. The child folders here are: Plumbing Demolition

Plumbing Remove & Re-install Cut & Patch Core Drill at Floor Core Drill at Wall

To use this section open up the folder you wish, click on the green button to takeoff the condition and mark this condition on the drawings.

All conditions in this section are count activities.

Domestic Water

The domestic water section provides takeoff for water piping of different types and diameters. The piping is set up on a budget basis with the lineal footage of the pipe as the only takeoff consideration. This section is organized as a parent-child folder arrangement. Under each major folder is a child folder with the material spec of the pipe. The parent child folders are organized as follows:

Domestic Water - Underground

Domestic Water Underground Copper Sweat Domestic Water Underground Copper Press Fit Domestic Water Underground CPVC Sch 80 Domestic Water Underground CPVC Flowguard Domestic Water Underground PEX

Domestic Water - Interior

Domestic Water Interior Copper Sweat Domestic Water Interior Copper Press Fit Domestic Water Interior CPVC Sch 80 Domestic Water Interior CPVC Flowguard Domestic Water Interior PEX

Domestic Water Misc

Water Manifold Water Flexible Connectors & Hose Water Backflow Preventer Water Anti Siphon Vacuum Breaker Water Pressure Regulators

In each child folders the takeoff conditions are there for the each size of pipe.

This is a screen shot of this section with one child folder opened up.

The fittings of the pipe are not available in this section. There will soon be a different assembly template database for detailed pipe takeoff with fittings.

Drainage

The drainage section provides takeoff for drainage piping of different types and diameters. The piping is set up on a budget basis with the lineal footage of the pipe as the only takeoff consideration. This section is organized as a parent-child folder arrangement. Under each major folder is a child folder with the material spec of the pipe. The parent child folders are organized as follows:

Drainage - Underground

Drainage Underground Cast Iron SV Drainage Underground Cast Iron XH Drainage Underground Cast Iron Ho-Hub Drainage Underground PVC DWV Drainage Underground ABS DWV

Drainage - Aboveground

Drainage Above Ground Cast Iron SV Drainage Above Ground Cast Iron XH Drainage Above Ground Cast Iron Ho-Hub Drainage Above Ground PVC DWV Drainage Above Ground ABS DWV

This is a screen shot of this section with one child folder opened up.

Sanitary Waste

The Sanitary Waste section provides takeoff for waste piping of different types and diameters. The piping is set up on a budget basis with the lineal footage of the pipe as the only takeoff consideration. This section is organized as a parent-child folder arrangement. Under each major folder is a child folder with the material spec of the pipe. The parent child folders are organized as follows:

Waste - Underground

Waste Underground Cast Iron SV Waste Underground Cast Iron XH Waste Underground Cast Iron Ho-Hub Waste Underground Duriron Waste Underground PVC DWV Waste Underground ABS DWV Waste Underground Polypropylene DWV Waste Underground Fuseal

Waste - Above Ground

Waste Inside Cast Iron SV
Waste Inside Cast Iron XH
Waste Inside Cast Iron Ho-Hub
Waste Inside Duriron
Waste Inside PVC DWV
Waste Inside ABS DWV
Waste Inside Polypropylene DWV
Waste Inside Fuseal
Waste Copper DWV
This is a screen shot of this section with one child folder opened up.

Drainage & Sanitary Misc

The Drainage & Sanitary Misc section provides takeoff for miscellaneous issues of drainage and waste piping that you will encounter to takeoff. This is a section of equipment and fixtures required to be taken off with drainage and waste piping. Each child folder of the section has the conditions.

Sanitary Drains Roof Drains Roof Penetration and Boot Area Drain Trench Drains Sanitary Pumps Grease Interceptors Package Sanitary Pump Station

This is a screen shot of this section with one child folder opened up.

Fixtures

The Fixtures section provides takeoff for all types of fixtures that you will need for standard takeoff issues. All fixtures are taken off by the count method. Each child folder of the section has the conditions.

Fixtures - Bath
Fixtures - Laundry
Fixtures - Medical
Fixtures - Emergency
Fixtures - Drinking Fountain
Fixtures - Kitchen
Fixtures - Other
This is a screen shot of this section with one child folder opened up.

Medical Gas System and Vacuum System

The Medical Gas system and the Vacuum system both work in the same manner so they are discussed together. For each section there is a folder for the pipe, and then another folder for the outlets.

The folders are as follows and each folder has the conditions you need for takeoff.

Medical Gas System Oxy/Med GasPipe - Copper Medical Gas Outlets

Vacuum System Vacuum Pipe - Copper Vacuum Outlets

This is a screen shot of this section with one child folder opened up.

Compressed Air System

The Compressed Air System is compressed of several folders with the compressed air pipe and drops then another folder for the equipment. The folders are as follows and each folder has the conditions you need for takeoff.

Compressed Air Pipe Copper Compressed Air Pipe Black Steel Compressed Air Pipe Gal Steel Compressed Air Pipe Polypropylene Compressed Air Equipment

This is a screen shot of this section with one child folder opened up. Notice the drops are with the pipe as a count condition.

Upgrades

The Budget Concrete Template will receive an upgrade in the spring and fall of 2012. As a purchaser of the template you can receive this upgrade at no cost. If you own the database I will put your suggestions into the upgrades. Just email the developer at planswift@database-solutions.com

Support

The developer of this PlanSwift library has worked in the construction software database industry for 15 years and works daily with estimators of all types.

Database Solutions planswift@database-solutions.com

- See more at: http://www.planswift.com/phpkb/article/plumbing-standard-2254.html#sthash.9g5YLwfu.dpuf

Multiline Tool (Needs Image)

Multiline Tool 📄

Items Included:

Multiline

Requirements

No special requirements.

Install Notes

Files Needed Multiline.SwiftPluginPackage from PlanSwift Plugin Store

Steps

- 1. Open PlanSwift if it is not already open.
- 2. Go to Plugin Store Tab and make the necessary steps for downloading the plugin.
- 3. When download is finished choose "Open". This will process the installation.

After installation you will find Multiline Tool in Linear Dropdown Menu on the Home Menu Tab.

Removal Notes

Steps

- 1. Open PlanSwift if it is not already open.
- 2. Go To Plugin Store Tab
- 3. Click on the Uninstall Plugin button in Plugin Tools Section
- 4. Choose "Multiline" and proceed with Next

Usage

This tool is for fast drawing of a multiple parallel lines. Custom options include:

- # of Lines Number of additional elements, that will be created -
- Offset Distance between elements ·
- Justification the side where the additional sections(items) will be created \cdot
- Each Section As New Item ChekBox ·
- New Item Type List ·
- Auto Generate ChekBox

Methods(available in Multiline Section Only - 🔊)

- 1. Draw Multiline for drawing the additional sections for first time
- 2. Refresh Multiline for redrawing additional sections in these events

OnPointMoved



Select and Set (Check Links)

Select and Set allows you to select sections and set properties on all the selected sections. You can use for multiplying your sections also.

How to use:

- 1. Select the sections of the items you want to set a property on.
- 2. Click on the Select and Set Icon (Home Tab).
- 3. Type in the property you want to set.
- 4. Type in the text value you want on that property you specified.

You can view in the Estimating tab (with sections showing) the properties you set on the sections. You can turn on show sections by clicking on the filter button and checking sections. Also add properties you want as columns with the columns button.

Warning:

Be very careful what property you set. It will overwrite the existing value on that property. May not work well with counts should work fine with label counts. Planswift does not provide support for this plugin.

For More Information : https://plugins.planswift.com/plugin-details/planswift-tools/select-and-set/

Shape Stamper (Check Links)

How to use:

1. Select the section you want the shape of.

2. Click on the **"Get Shape"** icon on the top bar.

3. Select the sections you want to apply that shape to.

4. Click on the "Stamp Shape" icon on the top bar. You can select a item and get its shape. Then stamp out other items with that same shape.

Warning: AS IS Be very careful. It will overwrite the existing value on the DigitizerData property. May not work well with counts because you may have double stacked nodes which will make two counts on top of each other. Depending on how you recorded the item you got the shape from. **PlanSwift does not provide support for this plug-in**.

For More Information: https://plugins.planswift.com/plugin-details/planswift-tools/shape-stamper/

Area Pro (Check Links)

Items Included:

Items Area Pro Page Items Area Pro Section **Functionality:**

Area Pro Plug-in will install a new digitizer item, enables you to receive calculated information for Area Items™ with Subtraction about:

Total Openings Area Total Openings Linear Total Gross Area

Gross Linear Total

Gross Volume Installation:

Open PlanSwift if it is not already open.

Go to "Download Plugins" Sidebar and make the necessary steps for downloading the plugin.

When download is finished choose

For more information : https://support.planswift.com/solution/articles/13000002693-area-pro

Concrete for Budget Takeoff (check links)

Concrete for Budget Takeoff plugin is a quick and easy database intended for swift takeoff of concrete without a lot of detail. This database covers all the major type of concrete and is intended to all for takeoff to gather the SF of forming, CY of concrete and the SF of finish.

To Install

Open PlanSwift. Go to the "Setting" Tab Click on the "Import Plugin Package" Follow the directions provided by Planswift. After installation you will find "Budget Concrete" in the "Templates" Tab ready for your use.

To Delete

Go to the "Templates" Tab Right click on the "Budget Concrete" Tab Click on "Delete Tab"

Plugin Sections

The database is sequenced into :

- Footings & Pile Caps
- Columns & Piers
- Walls
- Slab SOG
- Slab Elevated
- Slab Misc Work
- Misc Concrete
- Reinforcing
- Concrete Purchase

Finish, Cure & Seal

This is a screenshot of the takeoff window. Notice the sequence of the folders on the right side.

Footings & Pile Caps

The footings & pile cap section provides takeoff of concrete, forms and finish for the following conditions:

Continuous Footings Assembly - Formed both sides

Continuous Footings Assembly - Formed one side

Continuous Footings Assembly - not formed

Column or Pier Footing Assembly Rectangle

Column or Pier Footing Assembly Round

Column or Pier Footing Assembly Octagon

Pile Cap Assembly Rectangle

Pile Cap Assembly Round

Pile Cap Assembly Octagon

The assemblies for round shape will use the diameter for both the forming and volume calculations.

The assemblies for octagon will use the lineal foot of one side of the octagon for the forming and volume calculations.

Each assembly has a part for the takeoff of the concrete, the forming, and the finish.

This is a screen shot of the footing & pile cap section.

The takeoff properties for each assembly in this section is as follows:

The user will input the dimension information based on the geometry of the condition and revise the waste if desired.

Columns & Piers

The columns & Piers section provides takeoff of concrete, forms and finish for the following conditions:

Columns - Rectangle Columns - Round Columns - Octagon Piers - Rectangle Piers - Round Piers - Octagon Sonotube Concrete

The assemblies for round shape will use the diameter for both the forming and volume calculations. The assemblies for octagon will use the lineal foot of one side of the octagon for the forming and volume calculations.

The assembly for sonotube allows for sonotube of 8 inches to 42 inches in all standard diameters for sonotube.

Each assembly has a part for the takeoff of the concrete, the forming, and the finish.

This is a screen shot of the Columns & Piers section.

Walls

The Walls section provides takeoff of concrete, forms, bulkhead form, and finish for the following conditions:

Wall Concrete Grade Beam

Stem Wall

Pilasters @ Concrete Wall

Wall Brickshelf

Wall Boxout by Count

Sonotube Concrete

The assembly for walls and grade beam allows for the takeoff of:

Wall Concrete

Wall Form

Wall Bulkhead Form

Wall Column Beam Pockets

Wall Top Finish

The assembly for stem wall allows for the takeoff of:

Wall Concrete

Wall Form

Wall Top Finish

The assembly for pilasters allows for the takeoff of:

Wall Concrete

Pilaster Wall Form Pilaster Top Finish

The assembly for wall Brickshelf allows for the takeoff of:

Brickshelf concrete Deduct

Brickshelf Form

The Wall Boxout assemblies allow for the takeoff of:

Wall Boxout Concrete Deduct

Wall Boxout Form

The takeoff properties of the concrete wall assembly are as follows.

Slab - SOG

The SOG section provides a series of assemblies for taking off different types and conditions of typical slab on grade conditions. The takeoff assemblies are as follows:

Mud Slab

Column Diamonds

SOG Exterior - Edge Formed all Sides

SOG Interior - No Edge Form

SOG Construction Joint Form SOG

Thickened Edge

SOG Thickened Slab by LF

SOG Thickened Slab by Area SOG

Boxout

The assembly for Mud Slab allows for the concrete - form - finish takeoff of a mud slab in the construction work zone.

The assembly for **Column Diamonds** is specific to takeoff the column diamonds of an interior SOG. This is a count assembly where you count the column diamonds then in the properties set the length and width of the diamond

The assembly for SOG Exterior - Edge Formed all Sides is for exterior SOG in which the entire slab is formed

The assembly for **SOG Interior - No Edge Form** is for interior SOG in which the slab is poured up against a wall and therefore an expansion joint is required in lieu of the edge form.

The assembly for **SOG Construction Joint Form** is specifically to take off the bulkhead construction joint form of larger SOG when you have to pour the larger slab in pours by bays.

The assembly **SOG Thickened Edge** is specific for the thickened edge portion only of a SOG. This assembly only brings in the additional formwork and required for the thickened edge. The thickened edge is assumed to be monolithic with the slab. The interior side is assumed to slope at a 45 degree angle up to the nominal SOG. This is concrete only as there is no forming for this condition.

The assembly **SOG Thickened Slab by LF** is for a thickened section of the slab in the interior of the SOG that is take off by lineal feet. For example along a bearing wall line where the SOG is thicker for structural reasons. The slope on both sides is assumed to be at a 45 degree angle. This is concrete only as there is no forming for this condition.

The assembly **SOG Thickened Slab by Area** is for a thickened section of the slab in the interior of the SOG that is take off by area SF. For example at a chimney area where extra support is required. The calculation is a straight area times depth for the concrete. Since the slope on the edge is not taken into account the waste is changed to 10 %. Increase the waste if you desire. This is concrete only as there is no forming for this condition.

The assembly for the **SOB Boxout** is specific to the Boxout of a slab. This assembly will give you the items for the deduct of the concrete in the SOG and the add for the Boxout form. This is an alternative to the using the Boxout feature in the area calculation.

For each of the above assemblies in the properties window you will be asked in the questions pertaining to that condition. For example here is the properties for the SOG Exterior - Edge Formed all Sides.

Slab - Elevated

The elevated slab section provides a series of assemblies for taking off different types and conditions of typical elevated slabs. This includes both slab on metal deck, structural slabs, and pan or dome slabs. The takeoff assemblies are as follows:

Slab on Metal Deck

Slab on Metal Deck Construction Joint

Slab on Metal Deck Boxout

Structural Slab

Pan Slab

Structural or Pan Slab Boxout

Structural or Pan Slab Construction Joint

The assembly for **Slab on Metal Deck** is for a slab on metal deck and the exterior edge form. This assembly will use the average depth of the metal deck concrete. This takes into account the concrete in the flukes of the metal deck. The average depth should be provided by the deck supplier.

The assembly for **Slab on Metal Deck Construction Joint Form** is specifically to take off the bulkhead construction joint form of larger slab when you have to pour the larger slab in pours by bays.

The assembly for the **Slab on Metal Deck Boxout** is specific to the Boxout of the slab. This assembly will give you the items for the deduct of the concrete in the slab and the ad for the Boxout form. This is an alternative to the using the Boxout feature in the area calculation.

The assembly for the **Structural Slab** is specific to a self supporting structural slab that is bottom formed and supported with shoring till the concrete is cured. Additional lines for the bottom form and the shoring are added. The shoring is calculated in cubic feet, therefore the properties will ask for the height above the finish floor of the structural slab.

The assembly for the **Pan Slab** is specific to a self supporting pan filled structural slab that is bottom formed and supported with shoring till the concrete is cured. Additional lines for the bottom form and the shoring are added. The shoring is calculated in cubic feet , therefore the properties will ask for the height above the finish floor of the structural slab.

The assembly for the Structural or **Pan Slab Boxout** is specific to the Boxout of the slab. This assembly will give you the items for the deduct of the concrete in the slab and the add for the Boxout form. This is an alternative to the using the Boxout feature in the area calculation.

The assembly for the Structural or Pan Slab Construction Joint is specifically to take off the bulkhead construction joint form of larger slab when you have to pour the larger slab in pours by bays.

Slab - Misc Work

The Miscellaneous Work slab section provides a series of assemblies for taking off different types of slabs.

Topping Slab

Topping Slab Construction Joint

Topping Slab Boxout

Equipment Pad

The assembly for **Topping Slab** is for a the topping slab that is poured for the finish surface on a slab. In this case the topping slab is not designated as being on grade or elevated, but as a topping slab. The assembly brings in the concrete, forms, and finish of the topping slab.

The assembly for **Topping Slab Construction Joint** is specifically to take off the bulkhead construction joint form of the topping slab when you have to pour the larger slab in pours by bays.

The assembly for the **Topping Slab Boxout** is specific to the Boxout of the slab. This assembly will give you the items for the deduct of the concrete in the slab and the ad for the Boxout form. This is an alternative to the using the Boxout feature in the area calculation.

The assembly for Equipment Pad is used to takeoff the misc equipment pads that you will have for equipment settings.

Misc Concrete

The Miscellaneous Concrete section provides a place for sidewalk and metal pan stair assemblies. The assemblies in this section are as follows:

Sidewalk by LF

Sidewalk by Area

Metal Pan Landings

Metal Pan Risers

The assembly **Sidewalk by LF** is intended to for sidewalks where the user would like to takeoff the sidewalk by the lineal foot. Takeoff the sidewalk by the lineal feet of the centerline of the walk. This assembly will take off the forming, concrete, finish and the liquid curing white pigment finish used on sidewalks. The sidewalk concrete depth is set to 4" and the width is set to 4'. These are the normal dimensions for sidewalks but can be changed by the user if desired.

The assembly **Sidewalk by Area** is intended to for sidewalks where the user would like to takeoff the sidewalk by the area square foot. This assembly will take off the forming, concrete, finish and the liquid curing white pigment finish used on sidewalks. The sidewalk concrete depth is set to 4" as this is the normal depth of sidewalks, but can be changed by the user is need be. The edge form calculation will use the external lineal feet of area.

The assembly Metal Pan Landings is intended for takeoff of the metal pan landings of a stairwell. Count the landings with the takeoff and input at the properties tab for the input of the lending length, width, and the depth of the concrete.

The assembly Metal Pan Risers is intended for takeoff of the metal pan riser of a stairwell. Count the risers with the takeoff and input at the properties tab for the input of the riser, width, and the depth of the concrete.

Forming Misc

The Forming Misc section provides a place for other forming issues that the user may wish to takeoff. The point of the entire database is for budget concrete so these specific conditions are not a part of the assemblies that are listed above. This section is not set up as assemblies but as a section to takeoff the parts for all these conditions. This section is broken down as follows:

- Keyway & Chamfer
- Vapor Barriers

Anchor Bolts & Templates

Rigid Insulation

Set Steel in Concrete

Crack Control

The section for Keyway & Chamfer holds the items for keyway for 2x4, 2x6, and upset keyway. Also Edge chamfer item is here.

The section for Vapor Barriers contains the vapor barriers in 4 mill, 6 mill and for moistop.

The section for Anchor Bolts & Templates contains anchor bolts from 1/2" up to 2" in all nominal sizes. Also included are anchor bolt templates for 2-4-6-and 8 anchor bolt sets.

The section for Rigid Insulation contains rigid insulating for both under slab and foundation insulation form 1" to 3" in all nominal sizes in between.

The section for Set Steel in Concrete contains numerous types of embedded steel shapes and conditions.

The section for Crack Control contains conditions for saw cut of slab, zip strips and hand tool of joints.

Reinforcing

The Reinforcing section provides a place for reinforcing issues that the user other forming issues that the user may wish to takeoff. The point of the entire database is for budget concrete so these specific conditions are not a part of the assemblies that are listed above. This section is not set up as assemblies but as a section to take off the parts for all these conditions. This section is broken down as follows:

Rebar

Rebar CAD Weld Connections

Rebar Threaded Connectors

Rebar Dowels

Welded Wire Mesh

This section will be greatly upgraded in the next release of this database.

The Rebar section provides conditions for all nominal rebar sizes from #3 to # 14.

The Rebar CAD Weld Connections section provides conditions for all rebar CAD weld connections in al nominal rebar sizes from #3 to # 14.

The Rebar Threaded Connectors section provides conditions for all rebar threaded connections in all nominal rebar sizes from #3 to #14.

The Rebar Dowel section provides conditions for all rebar dowels in all nominal rebar sizes from #3 to #8

The Welded Wire Mesh section provides area SF conditions for every wire mesh size available.

Finish, Cure & Seal

The Finish, Cure & Seal section provides a place for finishing activities to placed concrete. The point of the entire database is for budget concrete so these specific conditions are not a part of the assemblies that are listed above. This section is not set up as assemblies but as a section to takeoff the parts for all these conditions. The section is broken down as follows:

Finish Flatwork

Other Finish

Concrete Rubbing & Grind

Curing

Seal & Hardener

The Finish Flatwork Section is the placeholder for specific finish options, color, and shake on products.

The Other Finish Section is a place for top of wall and stair finish.

The Concrete Rubbing & Grind Section is the p[lace for grind, sandblast, rubbing, and bush hammer of the concrete finish.

The Curing Section provides other options for curing.

The Seal & Harder section provides for specific types of sealers and hardeners

Upgrades

The Budget Concrete Template will receive an upgrade in the spring and fall of 2012. As a purchaser of the template you can receive this upgrade at no cost.

If you own the database I will put your suggestions into the upgrades. Just email the developer at planswift@database-solutions.com

Support

The developer of this PlanSwift library has worked in the construction software database industry for 15 years and works daily with estimators of all types.

David Ayers

Database Solutions planswift@database-solutions.com

- See more at: http://www.planswift.com/phpkb/article/concrete-for-budget-takeoff-2226.html#sthash.VSaiaYud.dpuf

Outdated / Don't Need

Callouts

8 This is an error panel
This is a green success panel
This is a purple Note panel
i This is a blue info panel
A This is a yellow "Warning" Panel

PlanSwift Knowledge Base	· · ·27
Release Notes	· · ·28
Release Notes History	· · · ·29
Release Notes v.11.0.00.129 (Release Date: July 10, 2023) · · · · · · · · · · · · · · · · · · ·	· · · 30
Release Notes v.11.0.0.89 (Released: 09/08/2022).	· · ·32
Workaround for Storages Settings	· · ·34
Release Notes v.10.03.0.56 Released: 03/03/2022· · · · · · · · · · · · · · · · · ·	
Release Notes v.10.03.0.50 Released: 07/21/2021 07/21/2021 <t< td=""><td>· · ·37</td></t<>	· · ·37
Release Notes v.10.03.0.48 Released: 05/18/2021. October Notes v.10.03.0.48 Released: 05/18/2021.	· · ·38
Release Notes v.10.02.05.41 Released: 04/30/2020 · · · · · · · · · · · · · · · · · ·	· · · 40
Release Notes v.10.02.05.40 Released: 03/19/2020 ····································	
Release Notes 10.2.4 (03/2019) · · · · · · · · · · · · · · · · · · ·	
Release Notes 10.2 (3/28/2018) · · · · · · · · · · · · · · · · · · ·	
New Sample Content	
New Sample Templates	
New Sample Plans	
New Sample Parts	
New Sample Assemblies	
Release Notes 10.1.1.8 (includes 10.1.1.7)	
Release Notes ConstructConnect Platform Plugin Released: 07/09/2020	
Known Issues Log	
The PlanSwift Customer Portal (MyAccount)	
Registering for a MyAccount Profile	
How an Admin Creates MyAccount Profiles for Users	
Resetting or Changing Your Account Password	
Installing and Licensing PlanSwift	
System Requirements	
Does PlanSwift Run on Windows 10? Windows 11? Windows 365?· · · · · · · · · · · · · · · · · · ·	
Can I Install PlanSwift on a Mac?·····	
Windows Version Compatibility	
Downloading and Installing PlanSwift	
Upgrading PlanSwift	
Using PlanSwift Automatic Updates	
Required (Mandatory) Updates	
How To: Uninstall PlanSwift	
Licensing PlanSwift (Activation)	
Deactivating Your License (to Use on a Different Machine).	
PlanSwift Licensing FAQs.	· · 102
Activation Errors & Troubleshooting	104
Troubleshooting Activation Errors	105
Manually Activating PlanSwift107	

Can I Avoid Manual Activation?	113	
Manual Deactivation		114
Manually Activating a One-Time PlanSwift License		. 118
Manual Licensing FAQs120		· •