



**Pythagoras VBA
&
Pythagoras Object Model**

For Pythagoras 24.00
December 11 , 2023

Table of Contents

Table of Contents	i
Chapter 1 - Introduction.....	1-1
<i>How to read this manual?</i>	<i>1-1</i>
<i>General.....</i>	<i>1-1</i>
Remarks.....	1-1
<i>Pythagoras VBA</i>	<i>1-1</i>
1. Main differences between Microsoft VBA and Pythagoras VBA.....	1-1
2. The Pythagoras Object Model	1-2
3. Pythagoras Macro Libraries	1-3
4. Access to Protected and Invisible objects.....	1-3
<i>Getting started.....</i>	<i>1-3</i>
<i>Exporting and importing VBA Sources and Forms.....</i>	<i>1-5</i>
1. Export VBA Sources and Forms	1-6
2. Import VBA Sources and Forms.....	1-6
<i>Events.....</i>	<i>1-6</i>
1. Calling Event macros	1-6
2. Type of events	1-6
<i>Private Tools.....</i>	<i>1-8</i>
Chapter 2 - Application	2-1
<i>Properties.....</i>	<i>2-3</i>
GetPrivateTools	2-3
PrivateToolMenuEnabled.....	2-3
PrivateToolMenuString	2-4
PrivateToolSelected.....	2-4
PrivateToolCursor	2-4
PrivateToolSnapping.....	2-4
PrivateToolTargetEnabled.....	2-5
PrivateToolTargetType	2-5
PrivateToolTargetPosition.....	2-5
LicenceNumber.....	2-6
Language	2-6
Version	2-6
VersionStr	2-6
ApplicationDirectory.....	2-7
UserProfileDirectory	2-7
PublicDirectory.....	2-7
TemporaryDirectory	2-7
PythagorasLocalDirectory	2-8
CodeDescriptionFile	2-8
HiliteSelections	2-8
SystemClipboard.....	2-8
SystemDate.....	2-9
AreaAnnotationFormat	2-9
SystemParameters	2-9
IgnoreDataErrors.....	2-9
NilUUID	2-10
NegativeInfinity.....	2-10
PositiveInfinity.....	2-10
NaN	2-10
<i>Methods.....</i>	<i>2-11</i>
NewToolbarButton	2-11
RemoveToolbarButtons.....	2-11

EnableToolBarButton.....	2-11
NewUserMenu.....	2-11
RemoveUserMenu	2-12
AppendUserMenuItem	2-12
AppendUserPopupMenu.....	2-12
AppendUserSubMenuItem.....	2-12
RemoveUserMenuItem.....	2-13
RemoveUserSubMenuItem.....	2-13
EnableUserMenuItem	2-13
CheckUserMenuItem	2-14
OpenURL.....	2-14
OpenDocument	2-14
GetNewDocument.....	2-15
PutDocumentOnTop.....	2-15
GetActiveDocument	2-15
Wait	2-15
ContinueEvent.....	2-15
StopEvent.....	2-16
SetEventKeys.....	2-16
GetEventKeys	2-17
ComposeFileName.....	2-17
ConvertImage	2-17
GetDocuments.....	2-18
GetSymbolLibrary.....	2-18
GetCoordinateReferenceSystems.....	2-19
GetTiePointsSetCollection.....	2-19
GetDialogs.....	2-19
TileWindows.....	2-19
CascadeWindows.....	2-20
AddToRecentTextFiles	2-20
OpenExternalFile	2-20
DownloadFile	2-21
GetPathName.....	2-21
GetFileName.....	2-21
SplitFileName	2-21
ChangeExtension.....	2-22
SplitURI	2-22
ComposeURI.....	2-22
Intersection2Lines	2-23
Intersection	2-23
BestLine	2-24
BestArc.....	2-24
BestCircle	2-24
ProjectionOnLine	2-24
OptionSupported	2-25
MacroOptionSupported	2-25
ColumnValue.....	2-25
SelectCondition.....	2-25
NameCategory.....	2-26
RowSetDefinition.....	2-26
MatchCondition	2-26
WhereCondition	2-27
ColumnDefinition.....	2-27
CreateUUID.....	2-27
IsNilUUID.....	2-28
EqualUUID	2-28

UUIDToString	2-28
StringToUUID	2-28
UUIDToDate	2-29
Parse	2-29
TrueColorToColor	2-30
ColorToTrueColor	2-30
GetUserDefaults	2-30
GetAttributesSet	2-30
SetAttributesSet	2-31
SetVBADefaultAttributes	2-31
IsNegativeInfinity	2-31
IsPositiveInfinity	2-32
IsInfinity	2-32
IsNaN	2-32
SortAll	2-32
Sort	2-33
SortValueIndex	2-33
IncPointIdent	2-33
InitSelectionCriteria	2-34
ActivateTool	2-34
SetStatusBarText	2-35
InitVectoriseData	2-35
<i>XML Methods</i>	2-35
CreateXMLDocument	2-35
ReadXMLDocument	2-38
ParseStringAsXML	2-38
GetXMLErrorPosition	2-39
XMLReplaceWhites	2-39
XMLCollapseWhites	2-39
XMLWhiteSpaceNormalize	2-39
XMLStringToBoolean	2-40
XMLStringToInteger	2-40
XMLDecimalStringToReal	2-40
XMLStringToReal	2-41
XMLStringToDateTime	2-41
XMLStringToTime	2-41
XMLStringToDate	2-42
XMLStringToUTCDateTime	2-42
XMLStringToUTCTime	2-42
XMLDivideString	2-42
XMLStringToBooleans	2-43
XMLStringToIntegers	2-43
XMLDecimalStringToReals	2-43
XMLStringToReals	2-43
XMLStringToDateTimesZ	2-44
XMLStringToTimesZ	2-44
XMLStringToDatesZ	2-44
XMLStringToDateTimes	2-45
XMLStringToDates	2-45
XMLStringToTimes	2-45
XMLStringToUTCDateTimes	2-45
XMLStringToUTCTimes	2-45
XMLBooleanToString	2-46
XMLIntegerToString	2-46
XMLRealToDecimalString	2-46
XMLRealToString	2-46

XMLDateTimeToString.....	2-47
XMLTimeToString	2-47
XMLDateToString	2-47
XMLUTCDateTimeToString	2-48
XMLUTCTimeToString.....	2-48
XMLUniteStrings	2-48
XMLBooleansToString	2-48
XMLIntegersToString.....	2-48
XMLRealsToDecimalString	2-49
XMLRealsToString.....	2-49
XMLDatesToString	2-50
XMLUTCDatesToString	2-50
XMLUTCTimesToString.....	2-50
XMLValidName	2-50
Chapter 3 - PrivateTool.....	3-1
<i>Properties.....</i>	<i>3-1</i>
Active	3-1
Default	3-2
Tooltip	3-2
MenuEnabled.....	3-2
MenuString	3-2
Snapping.....	3-3
KeepSelection	3-3
TargetEnabled.....	3-3
TargetType	3-4
TargetPosition.....	3-4
TargetLineStyle	3-4
TargetLineWidth.....	3-5
TargetLineColor	3-5
See TargetLineStyle.....	3-5
Cursor	3-5
OnPrivateTool, OnMouseMove, OnMouseDown, OnMouseUp, OnMouseDoubleClick, OnRightMouseDown, OnRightMouseUp, OnRightMouseDoubleClick, OnMiddleMouseDown, OnMiddleMouseUp, OnMiddleMouseDoubleClick	3-5
<i>Methods.....</i>	<i>3-6</i>
TargetLine.....	3-6
See TargetLineStyle.....	3-6
RemoveTargetPoint.....	3-6
RemoveTargetLine.....	3-6
Chapter 4 - PrivateTools.....	4-1
<i>Methods.....</i>	<i>4-1</i>
GetFirst.....	4-1
GetNext.....	4-1
GetActivePrivateTool	4-2
SetPrivateToolActive.....	4-2
AddPrivateTool.....	4-2
RemovePrivateTool	4-3
Chapter 5 - Document.....	5-1
<i>Properties.....</i>	<i>5-2</i>
ReadOnly	5-2
NewGroupId.....	5-2
Name	5-3

SnappedObjectPosition	5-3
EventCursorPosInDoc.....	5-3
EventCursorPosInWindow	5-4
Info.....	5-4
Extent.....	5-4
ActiveOverlay	5-4
Scale.....	5-5
ZoomFactor	5-5
ActiveSheet.....	5-5
Database	5-6
LockedDatabase, DifferentDatabaseLocks, NrOfTables, NrOfAttributeTables, NrOfDictionaryTables, NrOfViews.....	5-6
NextPointIdent.....	5-6
<i>Methods.....</i>	<i>5-6</i>
Save.....	5-6
SaveAs.....	5-6
PrintToPDF	5-7
CloseDoc	5-7
NewState	5-8
EndState.....	5-8
ClearUndo	5-9
UpdateScreen	5-9
RedrawDocument.....	5-9
GetOverlay	5-9
GetSelection	5-10
GetFlaggedObjects	5-10
GetObjectUnderCursor	5-10
ObjectReferenceToCadObject.....	5-11
GetKeyboardEventData.....	5-11
Paste	5-11
ImportShape	5-12
ImportShapeRegion	5-13
ImportDWG (ImportDXF)	5-13
ImportDataCollector.....	5-14
GetActiveCoordinateSystem.....	5-14
GetCoordinateReferenceSystem.....	5-14
SetCoordinateReferenceSystem	5-15
TransformXYTo	5-15
TransformXYFrom	5-16
TransformPossible.....	5-16
Center	5-16
Zoom	5-17
ZoomRectangle	5-17
FitInWindow	5-17
ShowAll	5-18
GetSheets.....	5-18
GetLayers	5-18
GetLayerGroups.....	5-19
GetSymbolLibrary.....	5-19
GetWindow.....	5-19
GetTerrainModels.....	5-20
GetSubDocuments	5-20
GetTiePointsSetCollection.....	5-20
FirstTable, NextTable, TableByName, CreateTable, ImportDFB, DropTable, CopyTableSkeleton, CopyTable, FirstDBView, NextDBView, DBViewByName, CreateSimpleQuery,	

CreateAggregationQuery, DropQuery, CopyQuery, CopyDatabaseSkeleton, CopyDatabase, PurgeDatabase, DropDatabase, LockDatabase, UnlockDatabase, ActualizeQueries	5-21
ShowDataViewer.....	5-21
GetThematicMaps.....	5-21
GetViews.....	5-21
PointIdentUsed	5-22
Vectorise.....	5-22
GetCameras	5-23
Chapter 6 - Documents.....	6-1
<i>Properties</i>	6-1
NrOfDocuments	6-1
<i>Methods</i>	6-1
GetFirst.....	6-1
GetNext.....	6-1
Chapter 7 - SubDocument	7-1
<i>Properties</i>	7-1
Name	7-1
Active	7-1
Main.....	7-1
Status	7-2
NameRelative	7-2
DifferentDatabaseLock.....	7-2
<i>Methods</i>	7-2
GetDocument.....	7-2
Chapter 8 - SubDocuments	8-1
<i>Properties</i>	8-1
NrOfSubDocuments	8-1
<i>Methods</i>	8-1
GetFirst.....	8-1
GetNext.....	8-1
AddSubDocument.....	8-2
RemoveSubDocument.....	8-2
GetMain.....	8-2
GetActive	8-3
SetActive.....	8-3
GetDocument.....	8-3
Chapter 9 - View.....	9-1
<i>Properties</i>	9-1
Name	9-1
Active	9-1
Visible	9-1
Chapter 10 - Views.....	10-1
<i>Properties</i>	10-1
NrOfViews.....	10-1
<i>Methods</i>	10-1
GetFirst.....	10-1
GetNext.....	10-2
GetView	10-2
GetActive	10-2
SetActive.....	10-2
Chapter 11 - ThematicMap	11-1
<i>Properties</i>	11-1
Name	11-1

Active	11-1
Chapter 12 - ThematicMaps	12-1
<i>Properties</i>	12-1
NrOfThematicMaps	12-1
<i>Methods</i>	12-1
GetFirst.....	12-1
GetNext.....	12-2
Chapter 13 - Sheet.....	13-1
<i>Properties</i>	13-1
Name	13-1
Active	13-1
PaperWidth	13-2
PaperHeight	13-2
Margins	13-2
ScaleFactor	13-3
Scale.....	13-3
Position	13-3
PageAngle.....	13-4
Floating	13-4
<i>Methods</i>	13-4
GetAllObjects	13-4
CreatePoint.....	13-5
CreateLine.....	13-5
CreateArc.....	13-5
CreateCircle	13-5
CreateText.....	13-5
CreateRichText	13-6
CreateClothoid	13-6
CreateHorizontalCurve.....	13-6
CreateCurve.....	13-6
CreatePolygon	13-7
CreatePath	13-7
CreateImage.....	13-7
InsertImage	13-7
CreateCompound	13-8
Duplicate	13-8
PrintOut	13-8
PrintToPDF.....	13-9
SetPrintArea.....	13-9
ClearPrintArea	13-9
Chapter 14 - Sheets.....	14-1
<i>Properties</i>	14-1
NrOfSheets.....	14-1
<i>Methods</i>	14-1
GetFirst.....	14-1
GetNext.....	14-2
GetSheet.....	14-2
GetActive	14-2
SetActive.....	14-2
Chapter 15 - Overlay	15-1
<i>Properties</i>	15-1
OverlayType.....	15-1
<i>Methods</i>	15-2
CreatePoint.....	15-2

CreateLine.....	15-2
CreateArc.....	15-3
CreateCircle.....	15-3
CreateText.....	15-3
CreateRichText.....	15-4
CreateClothoid.....	15-4
CreateHorizontalCurve.....	15-5
CreateCurve.....	15-5
CreatePolygon.....	15-6
CreatePath.....	15-7
CreatePathElement.....	15-7
CreatePathElement3D.....	15-7
CreateImage.....	15-8
InsertImage.....	15-8
CreateCompound.....	15-9
GetAllObjects.....	15-9
GetCoordinateSystems.....	15-9
CreateTempCoordinateSystem.....	15-10
CreateUserCoordinateSystem.....	15-10
DeleteCoordinateSystem.....	15-10
Clip.....	15-11
XYToPoint.....	15-11

Chapter 16 - CadObject.....16-1

<i>Properties.....</i>	<i>16-2</i>
ObjectType.....	16-2
Comment.....	16-2
Link.....	16-2
Color.....	16-3
TrueColor.....	16-3
DisplayLevel.....	16-3
GroupId.....	16-3
Container.....	16-4
Layer.....	16-4
Overlay.....	16-4
CanBeDeleted.....	16-4
CanChangeOverlay.....	16-5
ObjectReference.....	16-5
Selected.....	16-5
Flagged.....	16-5
Data.....	16-6
Sheet.....	16-6
<i>Methods.....</i>	<i>16-6</i>
GetPoint.....	16-6
GetLine.....	16-6
GetArc.....	16-7
GetCircle.....	16-7
GetClothoid.....	16-7
GetCurve.....	16-7
GetText.....	16-7
GetRichText.....	16-8
GetPolygon.....	16-8
GetSolid.....	16-8
GetImage.....	16-8
GetPath.....	16-8
GetCoordinateSystem.....	16-9
GetCompound.....	16-9

GetAnnotation	16-9
GetDocument	16-9
GetSubDocument	16-9
GetReferencingObjects	16-10
SetData	16-10
DeleteData	16-10
ConvertToPolyline	16-11
SetLink	16-11
GetQueryData	16-11
CreateAnnotation	16-11
Chapter 17 - Point	17-1
<i>Properties</i>	17-1
PointIdent	17-1
Coordinates	17-2
Elevation	17-2
ZoomFactor	17-2
PointStyle	17-2
<i>Methods</i>	17-3
GetCadObject	17-3
GetOrientation	17-3
SetOrientation	17-3
GetScaleFactor	17-3
SetScaleFactor	17-4
Chapter 18 - Line	18-1
<i>Properties</i>	18-1
LineStyle	18-2
Scale	18-2
Width	18-2
Arrows	18-2
LineLength	18-2
Bearing	18-3
DeltaZ	18-3
SlopeLength	18-3
<i>Methods</i>	18-3
GetCadObject	18-3
GetBeginPoint	18-4
GetEndPoint	18-4
CDZtoXYZ	18-4
XYZtoCDZ	18-4
Invert	18-4
Chapter 19 - Arc	19-1
<i>Properties</i>	19-1
ArcLength	19-1
Radius	19-2
DeltaZ	19-2
SlopeLength	19-2
ArcAperture	19-2
Center	19-2
Center3D	19-3
<i>Methods</i>	19-3
GetCadObject	19-3
GetBeginPoint	19-3
GetEndPoint	19-3
CDZtoXYZ	19-3
XYZtoCDZ	19-4

Chapter 20 - Circle	20-1
<i>Properties</i>	20-1
Radius	20-1
Center	20-1
RadiusPnt	20-2
<i>Methods</i>	20-2
GetCadObject	20-2
Chapter 21 - Curve	21-1
<i>Properties</i>	21-1
NrOfPoints	21-1
NrOfSegments	21-2
Curvature (obsolete)	21-2
MaxRounding (obsolete)	21-2
ThreeD	21-2
Elevation	21-2
StartTangent (obsolete)	21-3
EndTangent (obsolete)	21-3
<i>Methods</i>	21-3
GetCadObject	21-3
GetCurvePoints	21-4
GetCurveSegments	21-4
GetBearingAt	21-4
Chapter 22 - Clothoid	22-1
<i>Properties</i>	22-1
Length	22-1
DeltaZ	22-2
SlopeLength	22-2
K (or ClothoidConstant)	22-2
StartRadius	22-2
EndRadius	22-2
Mirrored	22-3
StartCenter	22-3
EndCenter	22-3
<i>Methods</i>	22-3
GetCadObject	22-3
GetBeginPoint	22-4
GetEndPoint	22-4
CDZtoXYZ	22-4
XYZtoCDZ	22-4
Chapter 23 - Text	23-1
<i>Properties</i>	23-1
Position	23-1
Content	23-2
Orientation	23-2
Angle	23-2
Style	23-2
Weight	23-3
PointSize	23-3
HorizontalAlignment	23-3
VerticalAlignment	23-3
Underline	23-3
Border	23-4
LineSpacing	23-4
Opaque	23-4
NrOfLines	23-4

<i>Methods</i>	23-5
GetCadObject.....	23-5
GetAngle	23-5
GetContent.....	23-5
GetFontName.....	23-5
GetFont.....	23-6
GetFont (obsolete format)	23-6
SetFont	23-6
SetFont (obsolete format)	23-6
Chapter 24 - RichText	24-1
<i>Properties</i>	24-1
Position	24-1
Content.....	24-1
Orientation	24-2
Angle	24-2
HorizontalAlignment.....	24-2
VerticalAlignment.....	24-2
Opaque	24-2
Border.....	24-2
NrOfLines	24-2
<i>Methods</i>	24-2
GetCadObject.....	24-2
GetAngle	24-2
Chapter 25 - Path	25-1
<i>Properties</i>	25-1
NrOfObjects	25-1
NrOfElements	25-1
TotalLength	25-2
TotalSlopeLength.....	25-2
DeltaZ.....	25-2
StartingChainage.....	25-2
<i>Methods</i>	25-2
GetCadObject	25-3
GetPathObjects.....	25-3
GetPathElements.....	25-3
CDZtoXYZ	25-4
XYZtoCDZ	25-5
Chapter 26 - Road	26-1
<i>Properties</i>	26-1
Color	26-1
TotalLength	26-1
StartingChainage	26-2
<i>Methods</i>	26-2
GetCadObject	26-2
GetColor	26-2
SetColor.....	26-2
GetProfileData	26-3
GetHorizontalAlignmentData	26-3
GetSuperElevationData	26-4
GetRD_CrossSectionPositions.....	26-4
GetRD_ReferenceObjects	26-5
GetCenterLine.....	26-5
CDZtoXYZ	26-6
XYZtoCDZ	26-6
GetCrossSectionData	26-7

Chapter 27 - Polygon	27-8
<i>Properties.....</i>	<i>27-8</i>
NrOfObjects	27-8
Pattern.....	27-9
Border.....	27-9
Opaque	27-9
BackgroundColor	27-9
BackgroundPattern	27-10
Scale.....	27-10
Area	27-10
Perimeter	27-10
TotalSlopeLength.....	27-10
Centroid	27-10
<i>Methods.....</i>	<i>27-11</i>
GetCadObject	27-11
GetPolygonObjects	27-11
GetPolygonPathElements	27-12
AlignPolygon.....	27-12
GetPolygonAlignment	27-13
PointInsidePolygon	27-13
Chapter 28 - Solid.....	28-14
<i>Properties.....</i>	<i>28-14</i>
Opaque	28-14
Intensity.....	28-14
Volume	28-15
<i>Methods.....</i>	<i>28-15</i>
GetCadObject	28-15
GetBottomPnts.....	28-15
GetTopPnts	28-16
GetBottomIdents	28-16
GetTopIdents	28-17
Chapter 29 - Image	29-1
<i>Properties.....</i>	<i>29-1</i>
Monochrome.....	29-1
ColorDepth	29-2
Width & Height	29-2
DPMx & DPMY	29-2
WidthInPixels & HeightInPixels	29-2
Position	29-3
FileName.....	29-3
NameRelative	29-3
Intensity.....	29-3
Inverted	29-4
Transparent	29-4
<i>Methods.....</i>	<i>29-4</i>
GetCadObject	29-4
SetReduce_Enlarge.....	29-4
GetReduce_Enlarge	29-5
SetScale	29-5
GetScale.....	29-5
GetOrientation	29-5
SetOrientation	29-5
PixelRect.....	29-6
PositionRect.....	29-6
ImageData.....	29-6

ImageDataSize.....	29-7
Chapter 30 - Compound.....	30-1
<i>Properties</i>	30-2
AllFlagged	30-2
AllSelected	30-2
NrOfElements	30-2
<i>Methods</i>	30-2
GetCadObject	30-2
GetElements.....	30-3
Chapter 31 - Annotation	31-1
<i>Properties</i>	31-1
TextPosition.....	31-1
TextContent.....	31-2
TextResult.....	31-2
TextOrientation	31-2
TextAngle	31-3
TextStyle.....	31-3
TextWeight.....	31-3
TextPointSize.....	31-3
TextHorizontalAlignment.....	31-3
TextVerticalAlignment.....	31-4
TextUnderline	31-4
TextBorder	31-4
TextLineSpacing.....	31-4
TextOpaque	31-5
TextFreeze	31-5
TextColor.....	31-5
AnnotationType.....	31-5
SymbolType	31-6
SymbolColor.....	31-6
SymbolScale	31-6
NrOfLines	31-6
<i>Methods</i>	31-6
GetCadObject	31-7
GetTextAngle	31-7
GetTextFont.....	31-7
SetTextFont.....	31-7
SetTextFont (obsolete format).....	31-8
GetReferencedObjects.....	31-8
GetSymbolStyle.....	31-8
SetSymbolStyle	31-8
Chapter 32 - CoordinateSystem	32-1
<i>Properties</i>	32-1
Overlay	32-1
TypeOfCS	32-1
Name	32-1
Origin	32-2
ZOfOrigin	32-2
XAxisPnt.....	32-2
YAxisPnt.....	32-2
TransformationParameters.....	32-2
<i>Methods</i>	32-3
SetActive.....	32-3
Chapter 33 - CoordinateSystems	33-1

<i>Properties</i>	33-1
NrOfCoordinateSystems	33-1
<i>Methods</i>	33-1
GetFirst.....	33-1
GetNext.....	33-1
Chapter 34 - CoordinateReferenceSystem	34-1
<i>Properties</i>	34-1
Name	34-1
LocalName	34-1
ShortName.....	34-1
CRSType.....	34-2
Enabled	34-2
<i>Methods</i>	34-2
TransformXYTo	34-2
TransformToPossible.....	34-3
Chapter 35 - CoordinateReferenceSystems	35-1
<i>Properties</i>	35-1
NrOfCoordinateReferenceSystems.....	35-1
<i>Methods</i>	35-1
GetFirst.....	35-1
GetNext.....	35-1
GetCoordinateReferenceSystem.....	35-2
Chapter 36 - TiePointsSetCollection	36-1
<i>Properties</i>	36-1
Count.....	36-1
<i>Methods</i>	36-1
GetFirst.....	36-1
GetNext.....	36-1
GetTiePointsSet	36-2
Chapter 37 - TiePointsSet	37-1
<i>Properties</i>	37-1
Name	37-1
NrOfElements	37-1
CRSFrom.....	37-1
CRSTo.....	37-2
<i>Methods</i>	37-2
GetFirstElement	37-2
GetNextElement	37-2
Chapter 38 - Selection, FlaggedObjects, AllObjects	38-1
<i>Properties</i>	38-3
Overlay	38-3
Sheet.....	38-3
Wallpaper.....	38-3
NrOfObjects	38-3
NrOfWholeObjects.....	38-4
Color	38-4
TrueColor	38-4
ImageIntensity	38-5
Comment	38-5
DisplayLevel.....	38-5
Layer	38-5
PointStyle	38-6
LineStyle	38-6
LineWidth	38-6

LineArrows.....	38-6
TextStyle.....	38-7
TextWeight.....	38-7
TextPointSize.....	38-7
TextHorizontalAlignment.....	38-8
TextVerticalAlignment.....	38-8
TextUnderline.....	38-8
TextBorder.....	38-9
TextLineSpacing.....	38-9
TextOpaque.....	38-9
TextOrientation.....	38-10
TextAngle.....	38-10
CurveCurvature (obsolete).....	38-10
CurveMaxRounding (obsolete).....	38-10
PolygonPattern.....	38-11
PolygonBorder.....	38-11
PolygonOpaque.....	38-11
PolygonBackgroundColor.....	38-11
PolygonBackgroundPattern.....	38-12
SolidOpaque.....	38-12
SolidIntensity.....	38-12
CanChangeOverlay.....	38-12
Container.....	38-12
AnnotationTextStyle.....	38-13
AnnotationTextWeight.....	38-13
AnnotationTextPointSize.....	38-13
AnnotationTextHorizontalAlignment.....	38-14
AnnotationTextUnderline.....	38-14
AnnotationTextBorder.....	38-14
AnnotationTextLineSpacing.....	38-15
AnnotationTextOpaque.....	38-15
AnnotationTextOrientation.....	38-15
AnnotationTextColor.....	38-16
AnnotationSymbolColor.....	38-16
AnnotationSymbolScale.....	38-16
<i>Methods.....</i>	<i>38-17</i>
Count.....	38-17
GetFirst.....	38-17
GetNext.....	38-18
ChangeOverlay.....	38-18
Copy.....	38-19
Cut.....	38-19
Clear.....	38-19
Move.....	38-19
Rotate.....	38-20
Group.....	38-20
Ungroup.....	38-20
Unpack.....	38-21
DeleteData.....	38-21
AddAnnotation.....	38-21
AdjustDisplayLevel.....	38-22
SetZ.....	38-22
ScalePoints.....	38-22
ScaleLines.....	38-22
ScaleTexts.....	38-23
ScalePolygons.....	38-23

AlignPolygons.....	38-23
ConvertToPolylines.....	38-23
ConvertObjectsToPolylines (obsolete).....	38-23
GetFirstPoint.....	38-24
GetNextPoint.....	38-24
IntersectingLines	38-24
Intersections.....	38-25
UnderShoot.....	38-25
OverShoot	38-25
SetTextFont	38-26
SetAnnotationTextFont.....	38-26
SetAnnotationSymbolStyle	38-26
Transform	38-27
ClearSet.....	38-27
MakeSet.....	38-28
MakeSetDB	38-28
MakeSetTextual	38-29
Toggle	38-29
Include.....	38-29
Exclude.....	38-30
InSet.....	38-30
ApplyThematics.....	38-30
GetThematics.....	38-31
RemoveThematics	38-31
ExportShape.....	38-31
GetTerrainModel.....	38-32

Chapter 39 - TerrainModel.....39-1

<i>Properties.....</i>	<i>39-2</i>
Name	39-2
zMin.....	39-2
zMax	39-2
NrOfTriangles.....	39-2
NrOfVertexes	39-2
EnclosingRectangle	39-3
Visible	39-3
<i>Methods.....</i>	<i>39-3</i>
Boundary.....	39-3
Elevation.....	39-3
AdjustElevationOfVertex	39-4
AdjustAllElevations.....	39-4
Slope.....	39-4
GetTriangle.....	39-4
GetFirstTriangle	39-4
GetNextTriangle	39-5
GetFirstVertex.....	39-5
GetNextVertex	39-5
CrossSection.....	39-5
CrossSectionPnts.....	39-6
CalculateVolume.....	39-7
Duplicate	39-8
AddVertex	39-8
RemoveVertex.....	39-8
MoveVertex	39-8
AddTriangle.....	39-9
RemoveTriangle	39-9
SwapDiagonal.....	39-9

Chapter 40 - TerrainModels.....	40-1
<i>Properties.....</i>	<i>40-1</i>
NrOfTerrainModels.....	40-1
<i>Methods.....</i>	<i>40-1</i>
GetFirst.....	40-1
GetNext.....	40-1
FindTerrainModel	40-2
NewTerrainModel	40-2
DeleteTerrainModel.....	40-2
CalculateVolume.....	40-3
Chapter 41 - Triangle.....	41-1
<i>Methods.....</i>	<i>41-1</i>
Adjacent.....	41-1
Vertexes	41-1
Chapter 42 - Database	42-1
<i>Properties.....</i>	<i>42-1</i>
Document.....	42-1
LockedDatabase	42-2
DifferentDatabaseLocks.....	42-2
NrOfTables.....	42-2
NrOfAttributeTables.....	42-2
NrOfDictionaryTables	42-2
NrOfDBViews.....	42-3
<i>Methods.....</i>	<i>42-3</i>
FirstTable	42-3
NextTable	42-3
TableByName	42-3
CreateTable	42-4
ImportDBF.....	42-4
DropTable	42-5
CopyTableSkeleton	42-5
CopyTable	42-6
FirstDBView.....	42-6
NextDBView	42-6
DBViewByName	42-6
CreateSimpleQuery	42-7
CreateAggregationQuery	42-7
DropQuery	42-8
CopyQuery	42-8
CopyDatabaseSkeleton.....	42-8
CopyDatabase.....	42-9
PurgeDatabase.....	42-9
DropDatabase.....	42-9
CloseDatabase	42-10
NewState	42-10
EndState.....	42-10
LockDatabase	42-11
UnlockDatabase.....	42-11
ActualizeQueries	42-11
Chapter 43 - DBTable	43-1
<i>Properties.....</i>	<i>43-2</i>
Document.....	43-2
Name	43-2
Description	43-3
TableType	43-3

ObjectTypes.....	43-3
Access	43-3
NrOfColumns	43-4
NrOfRows	43-4
DataStamp.....	43-4
Database	43-4
<i>Methods.....</i>	<i>43-5</i>
Identical	43-5
GetColumn	43-5
AddColumn.....	43-5
DropColumn.....	43-6
InsertRow.....	43-6
FirstRow	43-7
NextRow	43-7
LastRow	43-7
PreviousRow.....	43-8
GetRows.....	43-8
RowById	43-8
DeleteRow.....	43-9
FirstSubDocument.....	43-9
NextSubDocument.....	43-9
GetSubDocuments	43-10
Clear	43-10
FastClear	43-10
CopyData	43-10
Chapter 44 - DBTableColumn.....	44-1
<i>Properties.....</i>	<i>44-1</i>
Table.....	44-1
SeqOrder.....	44-1
Name	44-1
Description	44-2
DataType	44-2
ObjectTypes.....	44-2
Display.....	44-3
MaybeNull.....	44-3
Changeable	44-3
Minimum	44-3
Maximum.....	44-3
Default.....	44-4
Fractional	44-4
ReferencedColumn.....	44-4
PendingAction.....	44-5
Indexed.....	44-5
Unique	44-5
Format.....	44-5
<i>Methods.....</i>	<i>44-6</i>
Move.....	44-6
GetRow	44-6
GetRows.....	44-6
Chapter 45 - DBQuery.....	45-1
<i>Properties.....</i>	<i>45-1</i>
Database	45-1
QueryType	45-2
MainBaseDefinition, AuxBaseDefinition	45-2
Name	45-2

Description	45-2
Where	45-2
Preferences	45-3
Match	45-3
Validity	45-3
ErrorIndex	45-5
SyntaxError	45-5
ErrorStart, ErrorEnd	45-6
NrOfWhereConditions	45-6
NrOfColumns	45-6
CadObjectLinked	45-6
ObjectTypes	45-6
Overlay	45-7
Sheet	45-7
<i>Methods</i>	45-7
GetWhere	45-7
AddWhere	45-7
AlterWhere	45-8
MoveWhere	45-8
DropWhere	45-8
AddColumn	45-9
GetColumn	45-9
DropColumn	45-9
GetRows	45-10
RowById	45-10
Chapter 46 - DBQueryColumn	46-1
<i>Properties</i>	46-1
Query	46-1
SeqOrder	46-1
Definition	46-1
DataType	46-2
<i>Methods</i>	46-2
Move	46-2
Chapter 47 - DBRow	47-1
<i>Properties</i>	47-1
Table	47-1
Query	47-1
LinkedObject	47-1
Id	47-2
Overlay	47-2
Sheet	47-2
<i>Methods</i>	47-2
GetAll	47-2
GetValue	47-3
SetValue	47-3
GetStringValue	47-3
Update	47-3
Identical	47-4
Chapter 48 - DBRows	48-1
<i>Properties</i>	48-1
Table	48-1
Query	48-1
Subdocument	48-1
ByColumn	48-2
<i>Methods</i>	48-2

FirstRow	48-2
NextRow	48-2
LastRow	48-2
PreviousRow	48-3
Chapter 49 - DBSubDocuments	49-1
<i>Properties</i>	49-1
Table.....	49-1
NrOfSubDocuments	49-1
<i>Methods</i>	49-1
FirstSubDocument.....	49-1
FirstSubDocument.....	49-2
Chapter 50 - Layer	50-1
<i>Properties</i>	50-1
Name	50-1
Protected	50-1
Visible	50-1
Wallpaper.....	50-2
MinimumScale.....	50-2
MaximumScale.....	50-2
CanBeDeleted	50-2
<i>Methods</i>	50-2
SetMinimumAndMaximumScale	50-2
Chapter 51 - Layers	51-1
<i>Properties</i>	51-1
NrOfLayers	51-1
Protected	51-1
Visible	51-1
Wallpaper.....	51-2
<i>Methods</i>	51-2
GetFirst.....	51-2
GetNext.....	51-2
GetActiveLayer.....	51-2
SetActiveLayer	51-3
GetLayer	51-3
NewLayer	51-3
DeleteLayer	51-3
DeleteEmptyLayers.....	51-4
SetProtected, SetVisible, SetSnappingDisabled.....	51-4
Chapter 52 - LayerGroup.....	52-1
<i>Properties</i>	52-1
Name	52-1
NrOfLayers	52-1
<i>Methods</i>	52-2
GetFirst.....	52-2
GetNext.....	52-2
SetProtected.....	52-2
SetVisible.....	52-2
SetSnappingDisabled.....	52-3
SetMinimumAndMaximumScale	52-3
AddLayer	52-3
RemoveLayer.....	52-3
Chapter 53 - LayerGroups.....	53-1
<i>Properties</i>	53-1
NrOfGroups	53-1

<i>Methods</i>	53-1
GetFirst.....	53-1
GetNext.....	53-1
NewGroup.....	53-2
DeleteGroup.....	53-2
SetActive.....	53-2
GetActive.....	53-2
Chapter 54 - Symbol	54-1
<i>Properties</i>	54-1
TypeOfSymbol.....	54-1
Class.....	54-2
Group.....	54-2
Item.....	54-2
FixedColor.....	54-2
ScaleDependent.....	54-2
CanBeRemoved.....	54-3
<i>Methods</i>	54-3
GetDocument.....	54-3
GetPointSymbol.....	54-3
GetLineSymbol.....	54-3
GetPattern.....	54-4
GetSectionSymbol.....	54-4
Chapter 55 - PointSymbol	55-1
<i>Properties</i>	55-1
Rotatable.....	55-1
Limits.....	55-1
<i>Methods</i>	55-2
Chapter 56 - LineSymbol	56-1
<i>Properties</i>	56-1
Width.....	56-1
Length.....	56-1
<i>Methods</i>	56-1
Chapter 57 - Pattern	57-1
<i>Properties</i>	57-1
<i>Methods</i>	57-1
Chapter 58 - SectionSymbol	58-1
<i>Properties</i>	58-1
<i>Methods</i>	58-1
Chapter 59 - SymbolGroup	59-1
<i>Properties</i>	59-1
TypeOfSymbol.....	59-1
Class.....	59-1
Group.....	59-2
NrOfSymbols.....	59-2
<i>Methods</i>	59-2
GetDocument.....	59-2
GetFirst.....	59-2
GetNext.....	59-2
Chapter 60 - Symbols	60-1
<i>Properties</i>	60-1
TypeOfSymbol.....	60-1
Class.....	60-1
NrOfSymbols.....	60-2

NrOfGroups	60-2
<i>Methods</i>	60-2
GetDocument.....	60-2
GetFirstSymbol	60-2
GetNextSymbol	60-3
GetSymbol.....	60-3
GetFirstGroup.....	60-3
GetNextGroup.....	60-3
GetGroup	60-3
RemoveAllUnusedSymbols.....	60-3
Chapter 61 - Library	61-1
<i>Properties</i>	61-1
Class.....	61-1
<i>Methods</i>	61-1
GetDocument.....	61-1
GetPointSymbols.....	61-2
GetLineSymbols.....	61-2
GetPatterns.....	61-2
GetSectionSymbols.....	61-2
Add.....	61-2
Remove	61-3
CreateSymbol	61-3
RemoveAllUnusedSymbols.....	61-3
Chapter 62 - SymbolContainer	62-1
<i>Methods</i>	62-1
AddLine.....	62-1
AddColoredLine.....	62-1
AddArc.....	62-2
AddColoredArc	62-2
AddCircle	62-2
AddColoredCircle.....	62-3
AddText.....	62-3
AddColoredText	62-3
AddPolygon	62-4
AddColoredPolygon.....	62-4
Chapter 63 - UserDefaults.....	63-1
<i>Properties</i>	63-1
NrUserDefaults	63-1
<i>Methods</i>	63-1
GetFirst.....	63-1
GetNext.....	63-2
SetActive.....	63-2
GetActive	63-2
RemoveUserDefault.....	63-2
AddUserDefault	63-3
Chapter 64 - UserDefault.....	64-1
<i>Properties</i>	64-1
Group	64-1
Name	64-1
Active	64-2
AttributesSet.....	64-2
<i>Methods</i>	64-2
GetFullName	64-2
Chapter 65 - AttributesSet	65-1

<i>Properties</i>	65-2
PointAttributes	65-2
LineAttributes	65-2
PolygonAttributes.....	65-2
SolidAttributes.....	65-2
TextAttributes	65-2
ImageAttributes.....	65-3
TrueColor	65-3
HasColor	65-3
Layer	65-3
HasLayer.....	65-4
DisplayLevel.....	65-4
HasDisplayLevel.....	65-4
Comment	65-4
HasComment	65-4
PointConfirmation.....	65-5
HasPointConfirmation	65-5
Copy.....	65-5
<i>Methods</i>	65-5
GetAnnotationAttributes	65-5
SetAnnotationAttributes.....	65-6

Chapter 66 - PointAttributes 66-1

<i>Properties</i>	66-1
PointStyle	66-1
HasStyle	66-1
TrueColor	66-2
HasColor	66-2
Layer	66-2
HasLayer.....	66-2
DisplayLevel.....	66-2
HasDisplayLevel.....	66-3
Comment	66-3
HasComment	66-3
PointIdent	66-3
HasPointIdent.....	66-3
ZoomFactor	66-4
HasZoomFactor	66-4
HasScaleFactor.....	66-4
Copy.....	66-4
<i>Methods</i>	66-5
GetScaleFactor	66-5
SetScaleFactor	66-5

Chapter 67 - LineAttributes 67-1

<i>Properties</i>	67-1
LineStyle	67-1
HasStyle	67-1
TrueColor	67-2
HasColor	67-2
Layer	67-2
HasLayer.....	67-2
DisplayLevel.....	67-2
HasDisplayLevel.....	67-3
Comment	67-3
HasComment	67-3
HiddenEndPoints.....	67-3

HasHiddenEndPoints	67-3
Arrows.....	67-4
HasArrows.....	67-4
Width	67-4
HasWidth.....	67-4
Scale.....	67-4
HasScale.....	67-5
Curvature (obsolete)	67-5
FullRounding (obsolete)	67-5
HasCurvatureAndFullRounding (obsolete)	67-5
Copy.....	67-5
Chapter 68 - PolygonAttributes	68-1
<i>Properties</i>	68-1
Pattern.....	68-1
HasPattern.....	68-1
Border.....	68-2
HasBorder.....	68-2
TrueColor	68-2
HasColor	68-2
Layer	68-2
HasLayer.....	68-3
DisplayLevel.....	68-3
HasDisplayLevel.....	68-3
Comment	68-3
HasComment	68-4
BGPattern.....	68-4
BGColor	68-4
HasBGPattern.....	68-4
Scale.....	68-4
HasScale.....	68-5
Opaque	68-5
HasOpaque.....	68-5
Copy.....	68-5
Chapter 69 - SolidAttributes	69-6
<i>Properties</i>	69-6
TrueColor	69-6
HasColor	69-6
Layer	69-6
HasLayer.....	69-7
DisplayLevel.....	69-7
HasDisplayLevel.....	69-7
Comment	69-7
HasComment	69-8
Opaque	69-8
HasOpaque.....	69-8
Intensity.....	69-8
HasIntensity.....	69-8
Copy.....	69-9
Chapter 70 - TextAttributes	70-1
<i>Properties</i>	70-1
Content.....	70-1
Style.....	70-1
HasStyle	70-2
PointSize.....	70-2
HasPointSize	70-2

Underline.....	70-2
HasUnderline.....	70-3
Border.....	70-3
HasBorder.....	70-3
Weight.....	70-3
HasWeight.....	70-3
LineSpacing.....	70-4
HasLineSpacing.....	70-4
HorizontalAlignment.....	70-4
HasHorizontalAlignment.....	70-4
VerticalAlignment.....	70-4
HasVerticalAlignment.....	70-5
Orientation.....	70-5
HasOrientation.....	70-5
Opaque.....	70-5
HasOpaque.....	70-6
TrueColor.....	70-6
HasColor.....	70-6
Layer.....	70-6
HasLayer.....	70-6
DisplayLevel.....	70-7
HasDisplayLevel.....	70-7
HasFont.....	70-7
Copy.....	70-7
<i>Methods.....</i>	<i>70-8</i>
GetFontName.....	70-8
GetFont.....	70-8
GetFont (obsolete format).....	70-8
SetFont.....	70-8
SetFont (obsolete format).....	70-8
Chapter 71 - ImageAttributes.....	71-1
<i>Properties.....</i>	<i>71-1</i>
TrueColor.....	71-1
HasColor.....	71-1
Layer.....	71-1
HasLayer.....	71-2
DisplayLevel.....	71-2
HasDisplayLevel.....	71-2
Comment.....	71-2
HasComment.....	71-2
Copy.....	71-3
Chapter 72 - AnnotationAttributes.....	72-4
<i>Properties.....</i>	<i>72-5</i>
TextContent.....	72-5
TextStyle.....	72-5
HasTextStyle.....	72-5
TextPointSize.....	72-5
HasTextPointSize.....	72-6
TextUnderline.....	72-6
HasTextUnderline.....	72-6
TextBorder.....	72-6
HasBorder.....	72-6
TextWeight.....	72-7
HasTextWeight.....	72-7
TextLineSpacing.....	72-7

HasTextLineSpacing	72-7
TextHorizontalAlignment.....	72-8
HasTextHorizontalAlignment.....	72-8
TextOrientation	72-8
HasTextOrientation	72-8
TextOpaque	72-9
HasTextOpaque	72-9
TextTrueColor	72-9
HasTextColor	72-9
FreezeText.....	72-9
HasFreezeText	72-10
HasTextFont.....	72-10
SymbolType	72-10
HasSymbolType.....	72-10
SymbolStyle.....	72-10
HasSymbolStyle	72-11
SymbolTrueColor.....	72-11
HasSymbolColor	72-11
SymbolScale	72-12
HasSymbolScale.....	72-12
FlipRange	72-12
HasFlipRange.....	72-12
PointerQuadrant	72-12
HasPointerQuadrant.....	72-13
Layer	72-13
HasLayer.....	72-13
DisplayLevel.....	72-13
HasDisplayLevel.....	72-13
UnitsAndDecimals	72-14
HasUnitsAndDecimals	72-14
UnitsCoord.....	72-14
HasUnitsCoord	72-14
UnitsLength	72-15
HasUnitsLength	72-15
UnitsArea.....	72-15
HasUnitsArea.....	72-15
UnitsAngle.....	72-16
HasUnitsAngle.....	72-16
DecCoord	72-16
HasDecCoord	72-16
DecLength	72-16
HasDecLength	72-17
DecArea.....	72-17
HasDecArea	72-17
DecAngle.....	72-17
HasDecAngle	72-18
FormatArea	72-18
HasFormatArea.....	72-18
Copy.....	72-18
<i>Methods.....</i>	<i>72-18</i>
GetTextFont.....	72-18
SetTextFont	72-19

Chapter 73 - Stack.....	73-1
<i>Properties.....</i>	<i>73-1</i>
NrOfItems.....	73-1
<i>Methods.....</i>	<i>73-1</i>

Push.....	73-1
Pop.....	73-1
Clear	73-2
Chapter 74 - Queue.....	74-1
<i>Properties.....</i>	<i>74-1</i>
NrOfItems.....	74-1
<i>Methods.....</i>	<i>74-1</i>
PutItem.....	74-1
GetItem.....	74-1
Clear	74-2
Chapter 75 - Tree	75-1
<i>Properties.....</i>	<i>75-1</i>
NrOfItems.....	75-1
<i>Methods.....</i>	<i>75-1</i>
Insert.....	75-1
Delete.....	75-2
Clear	75-2
GetFirst.....	75-2
GetNext.....	75-2
Chapter 76 - Window	76-1
<i>Properties.....</i>	<i>76-1</i>
HasHorScrollBar.....	76-1
HasVertScrollBar.....	76-1
Width	76-1
Height	76-2
Position	76-2
ScreenPosition	76-2
HorScrollThumb.....	76-3
VertScrollThumb.....	76-3
<i>Methods.....</i>	<i>76-3</i>
Minimize.....	76-3
Maximize	76-3
Restore	76-3
SetWindowSize.....	76-4
Chapter 77 - HTMLWindow.....	77-1
<i>Properties.....</i>	<i>77-1</i>
Width	77-1
Height	77-2
Title.....	77-2
<i>Methods.....</i>	<i>77-2</i>
DisplayHTML.....	77-2
Show.....	77-2
Hide	77-2
Chapter 78 - PathElement.....	78-3
<i>Properties.....</i>	<i>78-3</i>
ElementType.....	78-3
Sequence.....	78-3
Length.....	78-4
BeginPoint	78-4
EndPoint.....	78-4
BeginPoint3D.....	78-4
EndPoint3D	78-4
Line	78-5
Arc.....	78-5

Clothoid.....	78-5
Chapter 79 - PathLine.....	79-1
<i>Properties</i>	79-1
<i>Methods</i>	79-1
GetPathElement.....	79-1
Chapter 80 - PathArc.....	80-1
<i>Properties</i>	80-1
Radius	80-1
Center	80-1
Center3D.....	80-1
<i>Methods</i>	80-2
GetPathElement.....	80-2
Chapter 81 - PathClothoid.....	81-1
<i>Properties</i>	81-1
ClothoidConstant	81-1
StartRadius	81-1
EndRadius	81-2
StartAngle	81-2
StartCenter	81-2
EndCenter	81-2
Mirror	81-2
<i>Methods</i>	81-3
GetPathElement.....	81-3
Chapter 82 - CurveSegment.....	82-1
<i>Properties</i>	82-1
SegmentType.....	82-1
P1 & P2.....	82-2
Length.....	82-2
SlopeLength.....	82-2
<i>Methods</i>	82-3
MoveP1 & MoveP2	82-3
GetCurveLine	82-3
GetCurveArc.....	82-3
GetCurveBezier	82-3
Chapter 83 - CurveLine	83-5
<i>Properties</i>	83-5
SegmentType	83-5
P1.....	83-5
P2.....	83-5
Length.....	83-5
SlopeLength.....	83-5
<i>Methods</i>	83-5
MoveP1	83-5
MoveP2	83-5
Chapter 84 - CurveArc	84-6
<i>Properties</i>	84-6
SegmentType.....	84-6
P1.....	84-6
P2.....	84-6
ArcP	84-6
Length.....	84-6
SlopeLength.....	84-6
<i>Methods</i>	84-7

MoveP1	84-7
MoveP2	84-7
MoveArcP	84-7
Chapter 85 - CurveBezier	85-8
<i>Properties</i>	85-8
SegmentType.....	85-8
P1.....	85-8
P2.....	85-8
Length.....	85-8
SlopeLength.....	85-8
<i>Methods</i>	85-8
MoveP1	85-8
MoveP2	85-8
Chapter 86 - GeoMath.....	86-9
<i>Methods</i>	86-9
Distance (or dS).....	86-9
dZ NOT IMPLEMENTED.....	86-10
Projection.....	86-10
Intersections	86-11
CalculateIntersectionPoints	86-11
CalculateCenterArc	86-11
NewPoint.....	86-11
NewLine.....	86-12
NewArc.....	86-12
NewCircle.....	86-12
NewCircle3P.....	86-13
NewPolygon.....	86-13
NewPolygonByPathElements.....	86-13
NewPath	86-14
NewCurve.....	86-14
NewClothoid	86-14
NewPathLine	86-15
NewPathArc.....	86-15
NewPathClothoid.....	86-16
Chapter 87 - GeoObject	87-1
<i>Properties</i>	87-1
Horizontal	87-1
Z.....	87-1
ObjectType.....	87-1
HasColor	87-2
Color	87-2
<i>Methods</i>	87-2
GetPoint.....	87-2
GetLine	87-3
GetArc	87-3
GetCircle	87-3
GetClothoid.....	87-3
GetCurve.....	87-3
GetPolygon.....	87-4
GetPath.....	87-4
Copy.....	87-4
Chapter 88 - GeoPoint	88-1
<i>Properties</i>	88-1
XYZ.....	88-1

XY.....	88-1
X.....	88-1
Y.....	88-2
Z.....	88-2
Methods.....	88-2
GetGeoObject.....	88-2
Chapter 89 - GeoLine	89-1
Properties.....	89-1
P1 and P2	89-1
Bearing.....	89-1
Angle	89-2
Methods.....	89-2
GetGeoObject.....	89-2
Chapter 90 - GeoArc	90-1
Properties.....	90-1
C	90-1
P1.....	90-1
P2.....	90-2
Radius	90-2
Methods.....	90-2
GetGeoObject.....	90-2
Chapter 91 - GeoCircle	91-1
Properties.....	91-1
C	91-1
P	91-1
Radius	91-2
Methods.....	91-2
GetGeoObject.....	91-2
Chapter 92 - GeoClothoid	92-1
Properties.....	92-1
P1.....	92-1
P2.....	92-1
R1	92-2
R2	92-2
K.....	92-2
Mirror	92-2
Methods.....	92-2
GetGeoObject.....	92-2
Chapter 93 - GeoCurve	93-1
Properties.....	93-1
NrOfPoints	93-1
Curvature (obsolete)	93-1
MaxRounding (obsolete)	93-2
ThreeD.....	93-2
Elevation.....	93-2
StartTangent (obsolete).....	93-2
EndTangent (obsolete).....	93-2
Methods.....	93-3
SetPoints.....	93-3
GetPoints	93-3
GetGeoObject.....	93-3
Chapter 94 - GeoPolygon.....	94-1
Properties.....	94-1

NrOfPoints	94-1
NrOfPolygonElements	94-1
Area	94-1
Perimeter	94-2
Centroid	94-2
Pattern	94-2
<i>Methods</i>	94-2
SetPoints	94-2
GetPoints	94-3
SetElements	94-3
GetElements	94-3
GetGeoObject	94-3
PointInPolygon	94-4
ClipLine	94-1
ClipArc	94-1
ClipCircle	94-2
ClipCurve	94-2
ClipPolygon	94-3
Chapter 95 - GeoPath	95-1
<i>Properties</i>	95-1
NrOfElements	95-1
Length	95-1
<i>Methods</i>	95-2
SetElements	95-2
GetElements	95-2
GetParallel	95-2
GetGeoObject	95-3
CDToXY	95-3
XYToCD	95-4
Chapter 96 - MatrixOperations	96-5
<i>Methods</i>	96-5
NewMatrix	96-5
NewIdentityMatrix	96-6
IsEqual	96-6
Add	96-7
Multiply	96-7
Transpose	96-7
Inverse	96-8
Solve	96-8
Determinant	96-9
Chapter 97 - HelmertTransformation	97-1
<i>Properties</i>	97-1
NumberOfPassPoints	97-1
TransformationParameters	97-2
StandardDeviation	97-2
<i>Methods</i>	97-2
Calculate	97-2
GetResiduals	97-2
GetDelaunayTriangulation	97-3
Transform	97-3
Chapter 98 - DelaunayTriangulation	98-1
<i>Properties</i>	98-1
NumberOfPoints	98-1
NumberOfTriangles	98-2

<i>Methods</i>	98-2
Initialise.....	98-2
GetFirst.....	98-2
GetNext.....	98-2
GetTriangleVertexes.....	98-2
NaturalNeighbours.....	98-3
PointInTriangulation.....	98-3
Interpolation.....	98-3
Chapter 99 - Camera	99-5
<i>Properties</i>	99-5
Name	99-5
Position	99-5
Direction.....	99-5
Projection.....	99-6
Chapter 100 - Cameras	100-7
<i>Properties</i>	100-7
NrOfCameras	100-7
<i>Methods</i>	100-7
GetFirst.....	100-7
GetNext.....	100-8
GetCamera	100-8
AddCamera.....	100-8
DeleteCamera	100-9
Chapter 101 - XMLDocument	101-10
<i>Properties</i>	101-10
Version	101-10
Standalone.....	101-11
Comments1	101-11
Comments2	101-11
Comments3	101-11
Main.....	101-11
<i>Methods</i>	101-12
Clear	101-12
WhiteSpaceNormalize	101-12
Save.....	101-12
GetAsString.....	101-13
Chapter 102 - XMLItem.....	102-15
<i>Properties</i>	102-15
ItemType	102-15
Text.....	102-15
PI.....	102-15
Comment	102-16
Element.....	102-16
Document.....	102-16
NextItem	102-16
WhiteSpace.....	102-17
Parent	102-17
Chapter 103 - XMLItems.....	103-19
<i>Properties</i>	103-19
Parent	103-19
NrOfItems.....	103-19
CurrentItem.....	103-20
FirstItem	103-20
NextItem	103-20

NrOfElements	103-20
FirstElement	103-20
NextElement	103-21
<i>Methods</i>	103-21
Clear	103-21
Copy.....	103-21
WhiteSpaceNormalize	103-21
AddText.....	103-21
AddPI	103-22
AddComment.....	103-22
AddElement.....	103-23
RemoveItem.....	103-23
Chapter 104 - XMLText.....	104-25
<i>Properties</i>	104-25
Item.....	104-25
NextItem	104-25
Document.....	104-25
Content.....	104-25
SpaceBefore.....	104-26
Parent	104-26
Chapter 105 - XMLPI.....	105-27
<i>Properties</i>	105-27
Item.....	105-27
NextItem	105-27
Document.....	105-27
Target.....	105-27
Content.....	105-28
WhiteSpace.....	105-28
Parent	105-28
Chapter 106 - XMLComment.....	106-29
<i>Properties</i>	106-29
Item.....	106-29
NextItem	106-29
Document.....	106-29
Content.....	106-29
Whitespace	106-30
Parent	106-30
Chapter 107 - XMLElement	107-31
<i>Properties</i>	107-31
Item.....	107-31
NextItem	107-31
Document.....	107-32
Whitespace	107-32
NextElement	107-32
NrOfAttributes	107-32
FirstAttribute.....	107-32
Name	107-33
Value.....	107-33
SubItems.....	107-33
Parent	107-33
<i>Methods</i>	107-34
GetAttribute	107-34
AddAttribute.....	107-34
RemoveAttribute.....	107-34

RemoveAttributes	107-34
CopyAttributes.....	107-34
Clear	107-35
Copy.....	107-35
WhiteSpaceNormalize	107-35
Chapter 108 - XMLAttribute	108-37
<i>Properties</i>	108-37
Element.....	108-37
Name	108-37
Value.....	108-37
WhiteSpace.....	108-37
NextAttribute.....	108-38
Chapter 109 - KML Introduction	109-39
<i>Purpose</i>	109-39
<i>Structure of KML Object</i>	109-39
<i>Example</i>	109-41
Chapter 110 - KMLFolder	110-1
<i>Properties</i>	110-1
Name	110-1
Features.....	110-1
<i>Methods</i>	110-1
GetFeature	110-1
Save.....	110-2
Chapter 111 - KMLFeatures	111-1
<i>Properties</i>	111-1
NrOfFeatures	111-1
IsEmpty	111-1
Iterator	111-1
<i>Methods</i>	111-2
Add.....	111-2
Clear	111-2
Chapter 112 - KMLFeaturesIterator	112-1
<i>Properties</i>	112-1
Valid.....	112-1
EndOfList.....	112-1
<i>Methods</i>	112-1
GetFirst.....	112-1
GetNext.....	112-2
Insert.....	112-2
Remove	112-2
Chapter 113 - KMLFeature	113-1
<i>Properties</i>	113-1
FeatureType.....	113-1
<i>Methods</i>	113-1
GetPlacemark	113-1
GetFolder.....	113-1
Chapter 114 - KMLPlacemark.....	114-1
<i>Properties</i>	114-1
Name	114-1
Geometry.....	114-1
<i>Methods</i>	114-1
GetFeature	114-1
SetLineStyle.....	114-2

GetLineStyle.....	114-2
SetPolyStyle.....	114-2
GetPolyStyle.....	114-2
SetIconStyle.....	114-2
GetIconStyle.....	114-3
Chapter 115 - KMLGeometry.....	115-1
<i>Properties.....</i>	<i>115-1</i>
GeometryType.....	115-1
<i>Methods.....</i>	<i>115-1</i>
GetPoint.....	115-1
GetLineString.....	115-1
GetPolygon.....	115-2
Chapter 116 - KMLPoint.....	116-1
<i>Properties.....</i>	<i>116-1</i>
GeoPoint.....	116-1
<i>Methods.....</i>	<i>116-1</i>
GetGeometry.....	116-1
Chapter 117 - KMLLineString.....	117-1
<i>Properties.....</i>	<i>117-1</i>
Coordinates.....	117-1
LinearRing.....	117-1
AltitudeMode.....	117-1
Extruded.....	117-2
<i>Methods.....</i>	<i>117-2</i>
GetGeometry.....	117-2
Chapter 118 - KMLPolygon.....	118-1
<i>Properties.....</i>	<i>118-1</i>
AltitudeMode.....	118-1
Extruded.....	118-1
OuterBoundary.....	118-1
InnerBoundary.....	118-2
<i>Methods.....</i>	<i>118-2</i>
GetGeometry.....	118-2
Chapter 119 - KMLInnerBoundary.....	119-1
<i>Properties.....</i>	<i>119-1</i>
NrOfCutouts.....	119-1
IsEmpty.....	119-1
Iterator.....	119-1
<i>Methods.....</i>	<i>119-2</i>
Add.....	119-2
Clear.....	119-2
Chapter 120 - KMLInnerBoundaryIterator.....	120-1
<i>Properties.....</i>	<i>120-1</i>
Valid.....	120-1
EndOfList.....	120-1
<i>Methods.....</i>	<i>120-2</i>
GetFirst.....	120-2
GetNext.....	120-2
Insert.....	120-2
Remove.....	120-2
Chapter 121 - KMLCoordinates.....	121-4
<i>Properties.....</i>	<i>121-4</i>
NrOfCoordinates.....	121-4

IsEmpty.....	121-4
Iterator.....	121-4
Methods.....	121-4
Add.....	121-4
Clear	121-5
Chapter 122 - KMLCoordinatesIterator	122-1
Properties.....	122-1
Valid.....	122-1
EndOfList.....	122-1
Methods.....	122-2
GetFirst.....	122-2
GetNext.....	122-2
Insert.....	122-2
Remove	122-2
Chapter 123 - KMLCoordinate	123-1
Properties.....	123-1
GeoPoint	123-1
Chapter 124 - JSON.....	124-1
Properties.....	124-1
Type.....	124-1
JSONObject.....	124-1
JSONArray	124-1
JSON.....	124-2
Methods.....	124-2
AssignObject	124-2
AssignArray.....	124-2
Clear	124-2
AssignJSON	124-2
Search.....	124-3
Chapter 125 - JSONObject.....	125-1
Properties.....	125-1
JSON.....	125-1
NrOfElements	125-1
Methods.....	125-2
AddNameValue	125-2
GetFirst.....	125-2
GetNext.....	125-3
Search	125-3
Chapter 126 - JSONArray	126-1
Properties.....	126-1
JSON.....	126-1
NrOfElements	126-1
Methods.....	126-1
AddValue	126-1
GetFirst.....	126-2
GetNext.....	126-2
Search	126-2
Chapter 127 - Units	127-1
Properties.....	127-2
UnitOfLength	127-2
AngleUnit.....	127-2
HorizontalAngleClockwise.....	127-2
VertAngleType	127-3

XYZRepresentation	127-3
RectangleCoordRepr	127-3
ScaleFormat	127-3
ChainageFormat	127-3
DecimalSeparator	127-4
ItemSeparator	127-4
DecimalPlacesForAngles	127-4
DecimalPlacesForLengths	127-4
DecimalPlacesForXYCoords	127-5
DecimalPlacesForAreas	127-5
DecimalPlacesForVolumes	127-5
LongDateFormat	127-5
ShortDateFormat	127-6
TimeFormat	127-6
AM	127-6
PM	127-7
ValidLongDateFormat	127-7
ValidShortDateFormat	127-7
ValidTimeFormat	127-7
DateTimeFixedLength	127-7
MaximumYear	127-8
<i>Methods</i>	127-8
SetTransformation	127-8
GetTransformation	127-9
ConvertCoordinatesToUser	127-9
ConvertCoordinatesToInternal	127-10
ConvertLengthToUser	127-10
ConvertLengthToInternal	127-11
ConvertAngleToUser	127-11
ConvertAngleToInternal	127-11
ConvertAreaToUser	127-11
ConvertAreaToInternal	127-12
ConvertVolumeToUser	127-12
ConvertVolumeToInternal	127-12
StrConvertCoordinatesToUser	127-12
StrConvertCoordinatesToInternal	127-13
StrConvertLengthToUser	127-13
StrConvertLengthToInternal	127-13
StrConvertChainageToUser	127-13
StrConvertAngleToUser	127-14
StrConvertAngleToInternal	127-14
StrConvertAreaToUser	127-14
StrConvertAreaToInternal	127-15
StrConvertVolumeToUser	127-15
StrConvertVolumeToInternal	127-15
FormatLongDate	127-15
FormatShortDate	127-15
FormatTime	127-16
StringToDateTime	127-16
StringToDate	127-16
StringToTime	127-16
GetPreferences	127-17
SetPreferences	127-17
Chapter 128 - Comm	128-1
<i>Properties</i>	128-1
Baudrate	128-1

StopBits.....	128-1
Parity.....	128-2
HandShaking.....	128-2
NrOfBits.....	128-2
Port.....	128-2
<i>Methods.....</i>	<i>128-2</i>
OpenLink.....	128-2
CloseLink.....	128-3
Read.....	128-3
Write.....	128-3
ClearInputBuffer.....	128-4
Chapter 129 - Timer.....	129-1
<i>Properties.....</i>	<i>129-1</i>
ElapsedTime.....	129-1
Interval.....	129-1
CallBackMacro.....	129-1
<i>Methods.....</i>	<i>129-2</i>
Start.....	129-2
Stop.....	129-2
Chapter 130 - Progress.....	130-1
<i>Properties.....</i>	<i>130-1</i>
CancelButton.....	130-1
Message.....	130-1
Title.....	130-2
Percentage.....	130-2
Cancelled.....	130-2
<i>Methods.....</i>	<i>130-2</i>
Start.....	130-2
Stop.....	130-3
Chapter 131 - Fonts.....	131-1
<i>Properties.....</i>	<i>131-1</i>
NrOfFonts.....	131-1
<i>Methods.....</i>	<i>131-1</i>
GetFirst.....	131-1
GetNext.....	131-1
Chapter 132 - Folder.....	132-1
<i>Properties.....</i>	<i>132-1</i>
NrOfFolders.....	132-1
NrOfFiles.....	132-1
Path.....	132-2
<i>Methods.....</i>	<i>132-2</i>
Assign.....	132-2
FirstFolder.....	132-2
NextFolder.....	132-2
FirstFile.....	132-2
NextFile.....	132-2
Chapter 133 - File.....	133-1
<i>Properties.....</i>	<i>133-1</i>
Drive.....	133-1
Path.....	133-1
Name.....	133-1
Extention.....	133-2
FullName.....	133-2
FileType.....	133-2

Application	133-2
Methods.....	133-3
Assign.....	133-3
EncodeToBase64	133-3
Chapter 134 - ODBCDataSource	134-1
Methods.....	134-1
Connect	134-1
Select.....	134-2
FetchNext	134-2
GetData.....	134-2
Cancel.....	134-3
FreeStmt	134-3
Chapter 135 - HTTP	135-1
Properties.....	135-1
Status	135-1
StatusText	135-1
ResponseText	135-2
Methods.....	135-2
DoOpen.....	135-2
SetRequestHeader	135-2
Send.....	135-2
ResponseBody.....	135-3
GetAllResponseHeaders.....	135-3
SetTimeOuts.....	135-3
Chapter 136 - Dialogs.....	136-4
Methods.....	136-4
RequestFileOpen	136-4
RequestFilesOpen.....	136-5
RequestFolderOpen	136-5
RequestFileSave	136-6
RequestTextFileView	136-6
GetTrueColor	136-6
Chapter 137 - Console.....	137-1
Properties.....	137-1
Visible	137-1
Methods.....	137-1
ReadLine.....	137-1
WriteString.....	137-2
WriteLine	137-2
Chapter 138 - Form Modules.....	138-1
Form Events	138-2
Initialize.....	138-2
Modality.....	138-3
Terminate.....	138-3
QueryClose.....	138-3
Click.....	138-3
DropButtonClick.....	138-3
DblClick.....	138-4
Change.....	138-4
Form Control Methods and Properties.....	138-4
Form Object.....	138-5
Caption	138-5
Floating	138-5
Show.....	138-5

Hide	138-5
OpenForm	138-6
CloseForm	138-6
GetPosition	138-6
SetPosition.....	138-6
Width	138-6
Height	138-6
ScreenWidth	138-6
ScreenHeight.....	138-7
<i>Common properties.....</i>	<i>138-7</i>
SetFocus.....	138-7
Visible	138-7
Enabled	138-7
ControlTipText.....	138-7
<i>TextEdit Control.....</i>	<i>138-8</i>
Value (= Text).....	138-8
MaxLength	138-8
Locked	138-8
SetFocus, Visible, Enabled, ControlTipText	138-8
<i>Label Control.....</i>	<i>138-8</i>
Caption	138-8
Visible, Enabled, ControlTipText.....	138-8
<i>CommandButton Control.....</i>	<i>138-8</i>
Value.....	138-8
Caption	138-9
Visible, Enabled, SetFocus, ControlTipText	138-9
<i>ToggleButton Control.....</i>	<i>138-9</i>
Value.....	138-9
Caption	138-9
Visible, Enabled, SetFocus, ControlTipText	138-9
<i>CheckBox Control</i>	<i>138-9</i>
Value.....	138-9
Caption	138-10
Visible, Enabled, SetFocus	138-10
<i>OptionButton Control.....</i>	<i>138-10</i>
Value.....	138-10
Caption	138-10
Visible, Enabled, SetFocus, ControlTipText	138-10
<i>SpinButton Control.....</i>	<i>138-10</i>
Value.....	138-10
Locked	138-10
Visible, Enabled, SetFocus, ControlTipText	138-11
<i>Frame Control.....</i>	<i>138-11</i>
Caption	138-11
ControlTipText.....	138-11
<i>ListBox Control.....</i>	<i>138-11</i>
ColumnTitle (synonym ListHeader)	138-11
Text.....	138-11
ListCount	138-11
ListIndex.....	138-11
List.....	138-12
ColumnWidths	138-12
AddItem	138-12
RemoveItem.....	138-12
Clear	138-13
Selected.....	138-13

SetFocus, Visible, Enabled, ControlTipText	138-13
<i>DropDownList Control</i>	138-13
Text	138-13
ListCount	138-13
ListIndex	138-13
List	138-14
AddItem	138-14
RemoveItem	138-14
Clear	138-14
SetFocus, Visible, Enabled, ControlTipText	138-14
<i>Combobox Control</i>	138-15
Value	138-15
Text	138-15
MaxLength	138-15
Clear	138-15
ListCount, ListIndex, List, AddItem, RemoveItem	138-15
SetFocus, Visible, Enabled, ControlTipText	138-15
<i>Multipage Control</i>	138-15
Caption	138-15
Value	138-16
Visible, Enabled, ControlTipText	138-16

Chapter 139 - Pythagoras Data Types..... 139-1

XYZ	139-1
XY	139-1
XYV	139-1
KMLGeoPoint	139-1
RGB	139-2
KMLColor	139-2
Rectangle	139-2
ObliqueRectangle	139-2
Transformation	139-3
TransformationParameters	139-3
Style	139-3
SelectionCriteria	139-4
RowSetDefinition	139-5
SelectCondition	139-5
MatchCondition	139-5
WhereCondition	139-6
ColumnDefinition	139-6
DocInfo	139-7
WindowCoord	139-8
Line2D	139-8
Line3D	139-8
LinePair	139-8
ColumnValue	139-9
ValueIndex	139-9
UUID	139-9
ProfileElement	139-9
HorizontalTransition	139-10
SuperElevationPoint	139-10
SuperElevationData	139-10
SystemParameters	139-10
Font	139-11
Location	139-11
DTMReduce	139-11
VolumeCalculationDTM	139-11

<i>VolumeCalculationDTMs</i>	139-12
<i>VolumeCalculationResult</i>	139-12
<i>VectoriseData</i>	139-12
Chapter 140 - Constants	140-1
<i>Language Constants</i>	140-1
Button Display constants	140-1
Icon Display constants	140-1
Default button constants	140-1
Modality constants	140-1
MsgBox return value constants	140-1
Types constants	140-1
Compare constants	140-1
String constants	140-1
Day of week constants	140-2
File System constants	140-2
<i>Pythagoras Object Model Constants</i>	140-2
General	140-2
Ok - Not Ok	140-2
SortOrder	140-2
Group	140-2
Language	140-2
State of Subdocument	140-3
SaveBeforeClose	140-3
Overlay type	140-3
Tool type	140-3
Colors	140-4
Length Units	140-5
Angle Units	140-5
Vertical Angle	140-5
XYZ Representation	140-5
Representation of rectangular coordinates	140-5
Angle type	140-5
Scale Format	140-5
Chainage Format	140-5
Point Style	140-5
Line Style	140-6
AnnotationStyle	140-6
Arrows on end point of lines	140-6
Document Version	140-6
Selection	140-7
Compare	140-7
Type of intersection	140-7
ObjectTypes	140-7
ElementType	140-7
AnnotationKind	140-7
AnnotationType	140-8
Font source	140-8
Codepage	140-8
Text Style	140-8
Text Weight	140-8
Text: horizontal alignment	140-8
Text: vertical alignment	140-8
Text: underline	140-8
Text: border	140-9
Text: line spacing	140-9
Text: orientation	140-9

Images: ColorDepth	140-9
Polygon: styleId	140-9
Polygon: border	140-9
KML: Type of KML Objects	140-9
KML: Altitude Mode	140-10
PTPosition	140-10
Results from PointInPolygon method	140-10
Type of coordinate system	140-10
Paste operations	140-10
Symbol Type	140-10
View: Visible attributes	140-10
Symbol Class	140-10
Database: type of table constants	140-10
Database: name category constants	140-10
Database: table access constants	140-11
Database: reaction on pending references constants	140-11
Database: ptRowSetType	140-11
Database: RowSet Base Overlay (PtRowSetBaseOverlay)	140-11
Database: PtQueryType	140-11
Database: PtMatchConditionKind	140-11
Database: PtObjectFunction	140-11
Database: PtAggregateFunction	140-11
Database: PtSyntaxError	140-12
Database: PtQueryValidity	140-12
Table Import: PtImportCompletion	140-13
Vertical Curve : Fixed/Variable Radius or Length	140-13
Vertical Curve	140-13
Path Sequence	140-14
Path Element Type	140-14
Road Cross Section Element Type	140-14
Road Horizontal Alignment Transition Type	140-14
Road Superelevation Point Type	140-14
Annotation format Areas	140-14
Serial port settings : stop bits	140-14
Serial port settings : parity	140-14
Serial port settings : handshaking	140-14
Serial port settings : read/write	140-14
User Menu	140-14
Type of Toolbar Buttons	140-14
Bitmaps - Private Tool	140-15
Cursors	140-15
End Of Cross Section	140-15
Elevation/Value unknown	140-15
Import DWG parameters	140-15
Export Shape parameters	140-16
Program options	140-16
Macro options	140-16
Pythagoras specific types	140-16
Type of Coordinate Reference System	140-16
Type of Interpolations	140-16
Synchronize Dictionary Data Options	140-16
VectorisationType	140-16
PtPairing	140-17
PtOverlappingPointSelection	140-17
Decimal Places	140-17
Charsets	140-17

<i>XMLDocumentVersion</i>	140-21
<i>XMLItemType</i>	140-21
<i>XMLWhiteSpaceFacet</i>	140-21
<i>XMLCompletionCode</i>	140-21
<i>Pythagoras virtual key codes Constants</i>	140-22
Windows.....	140-22
Chapter 141 - Error handling	141-1
<i>Introduction</i>	141-1
<i>Pythagoras VBA errors</i>	141-1
<i>Pythagoras errors</i>	141-3
Chapter 142 - Pythagoras VBA language elements	3
<i>Option Statement</i>	3
Base.....	3
Explicit.....	3
Compare.....	3
MSCompatible.....	3
PlusMSCompatible.....	3
<i>Special values</i>	4
Nothing.....	4
Empty.....	4
Null.....	5
<i>Operators</i>	5
Arithmetic Operators.....	5
String Operators.....	5
Comparison Operators.....	5
Logical Operators.....	5
<i>Variable Declaration</i>	5
Const.....	5
Dim.....	6
User-Defined type.....	6
Enumeration type.....	7
ReDim.....	7
Erase.....	7
<i>Procedure & Function Declaration</i>	7
Sub.....	7
Function.....	8
<i>Calling Procedures and Functions</i>	8
<i>Program Structure and flow</i>	9
Do ... Loop.....	9
Exit.....	10
For ... Next.....	10
If ... Then ... Else.....	10
Select Case ... End Select.....	11
With ... End With.....	11
<i>Data Type Conversion Functions</i>	12
CBool.....	12
CByte.....	12
CDBl.....	12
CInt.....	12
CLng.....	12
CLngLng.....	13
CLngPtr.....	13
CSng.....	13
CStr.....	13
CDate.....	13

CVErr.....	14
Fix.....	14
Int.....	14
Round.....	14
Str.....	14
Val.....	15
<i>Err Object.....</i>	<i>15</i>
Clear.....	15
Description.....	15
Number.....	15
On Error.....	15
Raise.....	16
Resume.....	16
Source.....	17
<i>File Handling.....</i>	<i>17</i>
FreeFile.....	17
Open.....	17
Close.....	17
Reset.....	18
Print.....	18
Line Input.....	18
Write.....	18
Input.....	18
Put.....	19
Get.....	19
Seek Function.....	19
Seek Statement.....	19
Lock.....	20
Unlock.....	20
Kill.....	20
Loc.....	20
Eof.....	20
Lof.....	21
FileAttr.....	21
FileCopy.....	21
FileLen.....	21
Dir.....	21
MkDir.....	22
<i>Interaction Functions.....</i>	<i>22</i>
InputBox.....	22
MsgBox Function/Statement.....	23
<i>Math Functions.....</i>	<i>23</i>
Abs.....	23
Atn.....	23
Cos.....	23
Exp.....	23
Log.....	23
Sgn.....	23
Sin.....	23
Sqr.....	23
Tan.....	24
<i>String Manipulation.....</i>	<i>24</i>
ASCII and Unicode strings.....	24
Asc.....	24
AscW.....	24
Chr.....	24

ChrW	25
Format.....	25
InStr.....	26
InStrRev.....	26
LCase	27
Left.....	27
Len.....	27
LenB.....	27
LenW	27
StrConv.....	28
LTrim	28
Mid Function.....	28
Mid Statement.....	28
Right.....	28
RTrim.....	29
Space	29
Spc.....	29
StrComp.....	29
String.....	29
Tab	30
Trim.....	30
UCase.....	30
<i>Date functions.....</i>	<i>30</i>
Date literals	30
Date.....	30
DateAdd.....	30
DateDiff.....	31
DatePart.....	31
DateSerial.....	32
Day	32
Hour.....	32
Minute.....	32
Month.....	32
MonthName.....	33
Now.....	33
Second	33
Time.....	33
TimeSerial.....	33
TimeValue	33
Weekday	34
WeekdayName.....	34
Year.....	34
<i>Various.....</i>	<i>34</i>
TypeName.....	34
VarType.....	35
IsDate, IsEmpty, IsNull, IsError, IsNumeric, IsObject.....	35
IsEmptyArray	35
UBound	35
LBound.....	36
Beep.....	36
AppActivate.....	36
Shell.....	36
Declare	37
End.....	38

Chapter 1 - Introduction

How to read this manual?

This manual is in the first place a reference manual. There are nearly 100 classes with in total more than thousand properties and methods. Too much to be remembered by a normal human being.

Reading this introduction is a must, although the sections "Events" and "Private Tool" may be skipped on the first reading. Afterwards read the short description at the start of each chapter and try to understand the meaning of each class. Having a look to the names of properties and methods can help to get an insight in the purpose and possibilities of the class.

It is evident that for the classes that are related to Pythagoras Objects, general knowledge of Pythagoras is required. This manual does not explain the meaning of Layer, Sheet, Subdocument, Symbol, or whatever Pythagoras term.

Once you start to understand the major classes: Application, Document, Overlay, CadObject, ... you can start to write simple applications and step by step you will get an insight of the powerful features of the Pythagoras VBA Object Model.

If you are not familiar with the language VBA, we advise to read first "VBA for dummies". **Fout!**

Verwijzingsbron niet gevonden. Pythagoras VBA Language Elements and program and test some simple programs before starting reading other chapters.

General

Remarks

The examples used in this manual are strictly educational. They all compile, but it is not guaranteed that they will also run, for lack of a proper code 'environment'

Pythagoras VBA

What is Pythagoras VBA? Pythagoras VBA is a programming environment in Pythagoras that allows the user to write its own extensions to Pythagoras. The programming language "Pythagoras VBA" is nearly identical to the language used by the Microsoft products *Visual Basic* and MS-Office VBA. The way Pythagoras objects (Document, Point, Line, etc) are accessed from VBA is similar to the access of objects in the MS-Office products.

1. Main differences between Microsoft VBA and Pythagoras VBA

The programming environment of both systems is completely different. The programming environment of Pythagoras VBA is described later in this manual

The main language principles of Pythagoras VBA and Microsoft VBA are the same. So the declaring subroutines, variables, the use of control structures and error handling is the same.

The following language VB language data types are not supported in Pythagoras VBA:

- Currency
- Decimal
- Collections

In Pythagoras VBA, variables **must** be predefined.

Most standard functions of Visual Basic are also available in Pythagoras VBA. The standard VB mathematical functions, string functions, and file input/output functions are provided. An overview of the Pythagoras VBA language elements is given further in this manual.

Dates in Pythagoras internally are implemented rather as in both .Net and UUIDs than in Microsoft VBA, one of the main differences is different zero base for absolute dates. In Microsoft VB a zero date is December the 30th of 1899, in Pythagoras VBA it is January the 1st of 1 AD in proleptic Gregorian calendar. When converting absolute dates to numeric the result will be different: Pythagoras VBA date will be 693593 days more than Microsoft VBA.

In contrast to Microsoft VBA, the Pythagoras Object Model is not accessible from outside. Calls to other COM components are possible from Pythagoras Macros.

2. The Pythagoras Object Model

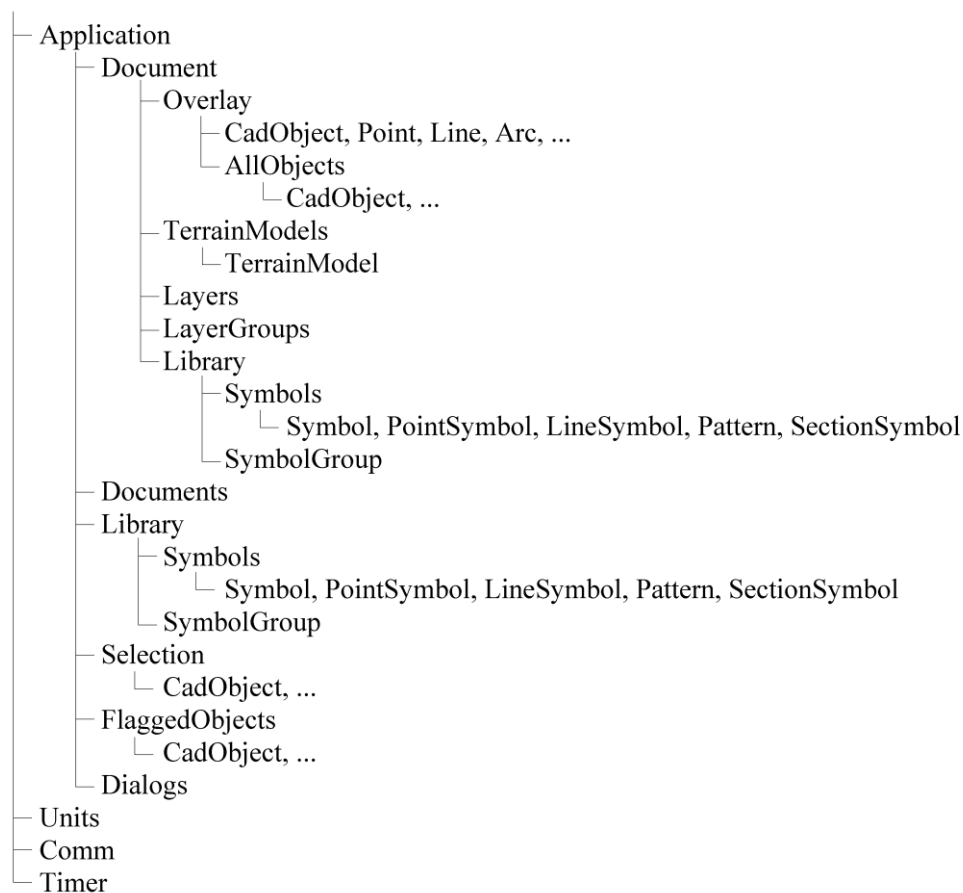
The Pythagoras Object Model allows access to nearly all Pythagoras objects such as documents, drawing elements (lines, points, ...), layers, the selection, to name just a few. This model is very powerful and it allows programmers to extend the functionality of Pythagoras. The main part of this manual describes the Pythagoras Object model in detail.

The Pythagoras Object Model contains also classes that are not related to Pythagoras Objects. Some examples: Folder, File, Comm, Timer, Progresss, Dialogs, ODBCDataSource, HelmertTransformation, AffineTransformation, DelaunayTriangulation, GeoObject, GeoMath, GeoPoint, GeoLine, .. These classes will make it much easier to write some applications without the need to interface with DLL's.

Almost all properties and methods do just the action described in this manual and have no side effects. A few properties and methods result in clearing the Undo-system of the document on which they have effect.

An example: deleting a layer will clear the Undo-system of the document. The reason: undo/redo is only applicable to objects and not to layers, sheets, subdocuments, etc. Allowing Undo could make the document inconsistent. The description mentions this side effect for all the properties and methods involved.

Overview of the Pythagoras Object Model



This overview only represents the main dependencies of the Pythagoras Objects. We recommend consulting this overview regularly while studying the Pythagoras Object Model.

3. Pythagoras Macro Libraries

VBA subroutines are called Macros. One or more macros are stored in a module. One or more modules make up a library. A library can contain Code Modules and Form Modules (Class Modules are not supported in the current version of Pythagoras VBA). Libraries are stored either in the Macro System Libraries file (Pythagoras.mlb) or in a Pythagoras document. So the Pythagoras.mlb file may contain multiple libraries, and each library may contain multiple macros. When Pythagoras is launched, the Pythagoras.mlb file will be loaded in memory. If a *Startup Library* is specified, the startup library will be compiled. If the startup library contains a macro with the name *OnStartup*, then this macro will be executed.

This way, the default environment of Pythagoras can be personalized at startup of Pythagoras.

When a document contains one or more libraries, these libraries will be read in memory when the document is opened and becomes the top window. Similar as with the system libraries, a document library may contain a Startup Library. This feature allows personalizing the Pythagoras environment when a specific document is opened.

At a given moment only one macro library, either a system- or a document macro library, can be active. This means that only macros of the active (loaded) library can be executed. It is not possible to call macros from other libraries.

If the active library contains *OnDocOpen* macro it will be executed every time a drawing is opened in Pythagoras. If the active library contains *OnDocClose* macro it will be executed every time a drawing is about to be closed. It is recommended to use these events only in system libraries.

4. Access to Protected and Invisible objects

VBA can not change objects which are protected. If the VBA programmer wants to do so, the protection of the layer has to be removed. No error or exception is given.

Operations on objects, which are in a layer, which is not visible, will be executed without any warning.

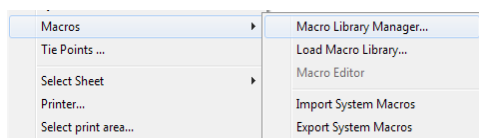
Getting started

The design and maintenance of a macro is best done in a document macro library. A document library can be easily moved to the system library. The document with its libraries can be easily passed on to other users.

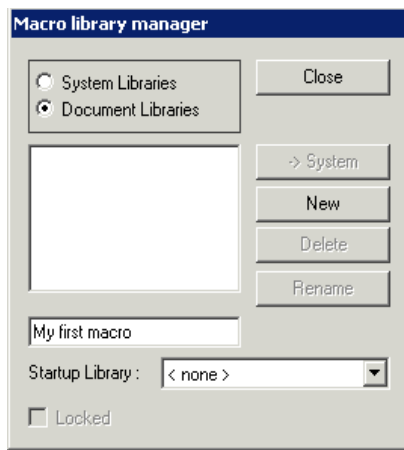
Step 1 : open a new document and give the document a name.

Step 2 : create a document library

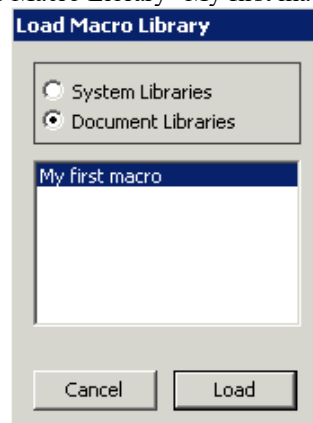
a) Select *File, Macros, Macro Library Manager...*



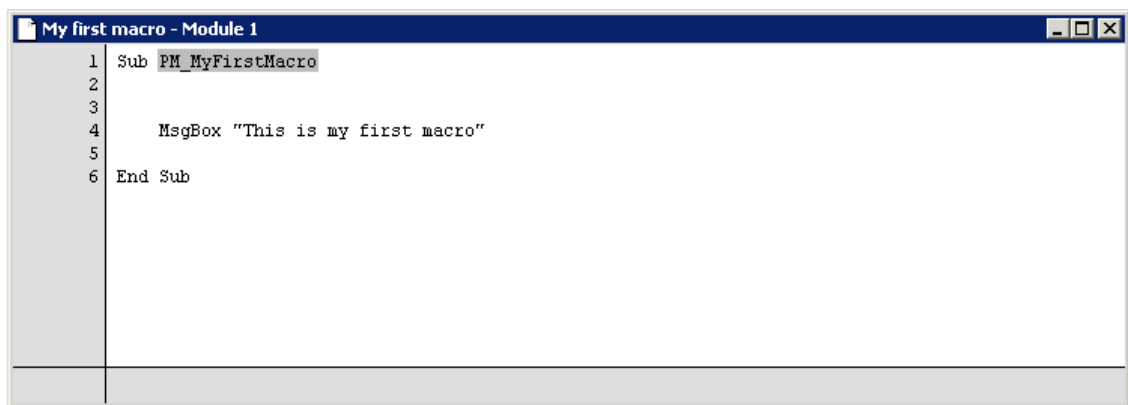
b) Select the radio button "Document Libraries", type the name of the new library and click *New*.



Step 3 : Load the Macro Library "My first macro"



Step 4 : Select menu *File, Macros, Macro Editor* to edit the macro.



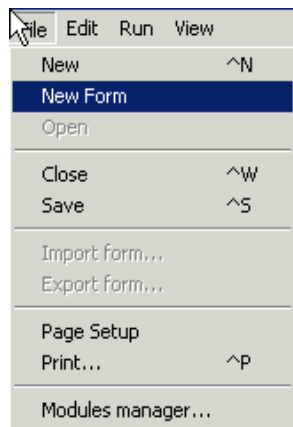
The use of the macro editor is straightforward. Try out the menu items to become familiar with the functions of the macro editor.

When the name of a Sub starts with PM_ (e.g. PM_MyFirstMacro), that macro will be added in the menu *Tools, Run Macro*. The macro can be directly called from there. Use of such macros is useful for testing or for very simple VBA applications.

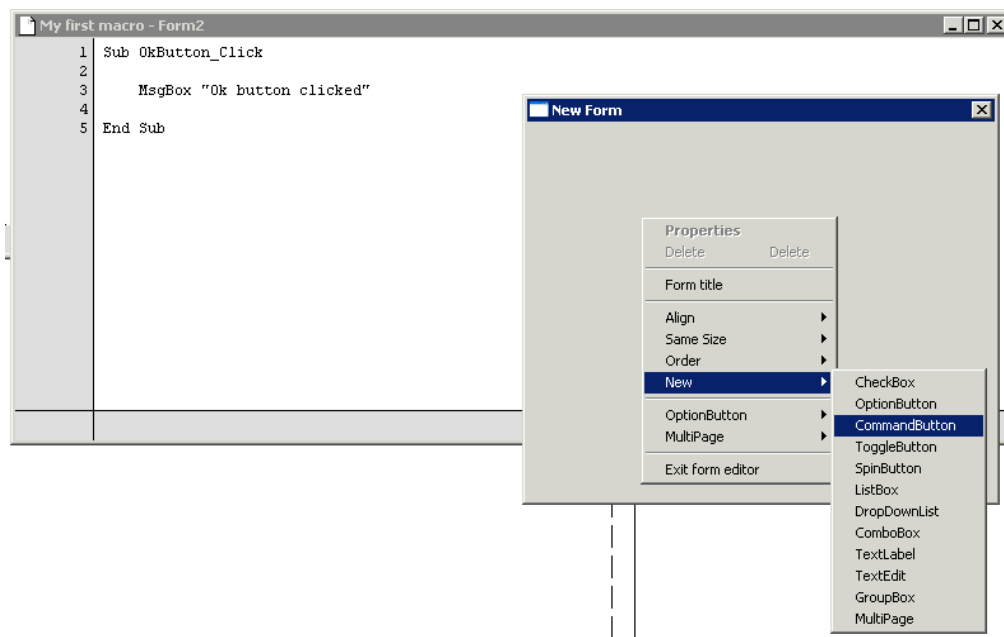
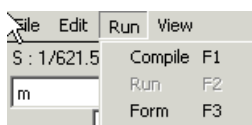
Step 5 : Creating forms

The form resource and the form code are stored in a *Form module*.

Select *New Form* from the *File* menu.



An empty form module will be opened. In the form module you need to write the Form Event macros. Use the menu *Run, Form* to create a new form or the edit an existing one. The right mouse button is used in the form to create new controls, align, etc.



Tips & tricks about forms

- Controls that are created in, or moved to, a GroupBox or MultiPage become a member of the group or multipage. Moving a GroupBox or MultiPage will move also the members.
- When deleting a Groupbox or MultiPage, the members are also deleted. If you don't want this behaviour, you first have to move out the controls.
- The form editor has no undo system. But when quitting the form editor you have the option to save or restore the form. Before doing some complex editing operations, temporary quit and save the form.

Exporting and importing VBA Sources and Forms

VBA Modules and forms can be exported from a Macro Library. The Modules and forms are saved in text format so that they can be opened by any text editor. This makes it easier to compare the sources of a VBA library with another version of the library. Another use could be global search and/or replace. The names of the text files will correspond with there respective modules names. The extension of the text file depends on the type of module.

Code module : **.bas**
Form module : **.frm**
Form : **.frx**

The exported text files can be imported again in the existing, another or in a new Macro Library. The module names and the type will be derived from the name of the text file. Existing files are optionally skipped or overwritten.

1. Export VBA Sources and Forms

When selecting the menu item *Export VBA Sources and Forms...* you need to specify a directory where the sources and forms will be stored. All modules and forms are exported.

When selecting the menu item *Export Form* when a form module has the focus, the form resource is exported. The form can be imported in another form module.

When selecting the menu item *Export Form* when a form module has the focus, **and the CTRL-key is hold down**, the form can be exported in a readable format (CSV-file). This file on its turn can be imported in a spreadsheet program.

2. Import VBA Sources and Forms

When selecting the menu item *Import VBA Sources and Forms...* you need to specify a directory from where the sources and forms will be read. All files in the given directory with the extensions **.bas** and **.frm** will be imported. The **.frx** files (form resources) will be imported in the form source in case the form source was present. If one or more modules already exist, a message will be given with the possibility to overwrite the modules or cancel the operation.

When selecting the menu item *Import Form* when a form module has the focus, a form resource that has been previously exported can be imported. This allows copying a form to another macro.

Events

The main purpose of introducing event macros is to give the VBA user the possibility to filter mouse and other window events. This allows the user to modify the common mouse actions, to create his own shortcuts, to execute code by the creation or opening of a document or to add his own menu items....In short, to customize the overall behavior of Pythagoras.

1. Calling Event macros

The mechanism of the event macros is the following. The general message loop or event filter will trap, if some conditions are met, all events that can have a macro. If an event occurs, it will execute the VBA macro with a predefined fixed name. The VBA macro can execute all required actions, for example moving an object. When the event macro is executed Pythagoras will still execute the default actions, unless the event macro requests otherwise (see *StopEvent*).

Some events are only trapped if a document is opened; some are trapped with or without open documents. Events are not related to a specific document. It is up to the event macro to decide what action may be done or not on a specific document. On the other hand, we can make a restriction between mouse events in profile, document or typical section.

2. Type of events

2.1 Mouse events

OnMouseMove
OnMouseDown
OnMouseUp
OnMouseDoubleClick

(* Windows only *)
OnRightMouseDown
OnRightMouseUp

```
OnRightMouseDown
OnMouseDown
OnMouseUp
OnMouseDown
```

Mouse events are only trapped within a document window and when the tool "Private Tool" (see Private Tool) is selected. Clicking on other parts of the application has no effect.

2.2 Keyboard events

```
OnKeyDown
OnKeyUp
```

The `OnKeyDown` and `OnKeyUp` macros will only be called when a key is in the set of keys initialised by the `Application` method `SetEventKeys`. Initially the set will be empty.

2.3 Loading Macro Library

```
OnStartup:
```

This macro is called when a library is activated

2.4 Deactivating Macro Library

```
OnTermination:
```

This macro is called immediately before a library is deactivated, i.e. a module or a library is compiled or a new library is loaded. This macro is called only if one or more macros of the active library were called.

2.3 Opening a drawing

```
OnDocOpen:
```

This macro is called immediately after a drawing is opened.
It is recommended to use this event only in system libraries.

2.4 Closing a drawing

```
OnDocClose:
```

This macro is called immediately before a drawing is closed.
It is recommended to use this event only in system libraries.

2.5 Selection/Deselection of the private tool

```
OnPrivateTool
```

This macro is called when the private tool is selected or deselected. Selection only occurs if the user selects the private tool item in the Tool menu. Deselection occurs when another tool is selected and in many other cases, e.g. when a new document is opened.

The `OnPrivateTool` macro could check if the private tool is selected (see `Application` method `PrivateToolSelected`) and take appropriate action. This event is useful if in the Private Tool mode a target is displayed. The macro can then control the display and removal of the target.

2.6 User menus

A macro can add a user menu to the menu bar. This user menu can be filled with menu items that have VBA macro call-back macros. The menu can be installed for the application (document window). The installation of the menu is done by a VBA macro. The installation only succeeds if the menu can be added. The menu must have a unique, not empty menu string. You can install a simple menu item or a popup menu items. If a user menu is selected, the call-back function is executed.

Private Tools

Private Tools allow the user to send mouse events to macros (event macros). These macros will then respond on the proper operation of the mouse. When Pythagoras is in a Private Tool Mode, the program searches for 'event macros' and executes them. The event macros that can be used in Pythagoras are:

- OnMouseMove
- OnMouseDown
- OnMouseUp
- OnMouseDoubleClick
- OnRightMouseDown
- OnRightMouseUp
- OnRightMouseDoubleClick
- OnMiddleMouseDown
- OnMiddleMouseUp
- OnMiddleMouseDoubleClick

The Private Tool is by default disabled.

If the property *MenuEnabled* is set TRUE, then the Private Tool can be selected. The property *MenuString* gives the opportunity to change the name of this menu item to a more relevant one.

A VBA macro can a.o.

- Enable Private Tool	<i>MenuEnabled = True</i>
- Change the name	<i>MenuString</i>

When a library is closed (e.g. when another library is loaded), the Private Tools will be removed.

When a Private Tool item is selected, the standard behavior will be the same as in Select mode. When receiving Mouse events, Pythagoras will execute the appropriate event Macro if the Macro is defined in the active VBA library.

An Event Macro can suppress the standard behavior by calling the *StopEvent* method. Standard behavior can be restored by calling the *ContinueEvent* method. StopEvent has to be called before returning from an Event Macro. In other situations, calling StopEvent has no effect.

For more information see objects *PrivateTool* and *PrivateTools*.

Note: the Application methods related to private tools became redundant in Pythagoras 13.

Chapter 2 - Application

Description

The Application object is the top-level object of the Pythagoras Object Model. Most of the other object references are obtained via the Application object.

Example

```
Dim A As Application
Dim Doc As Document

Set A = New Application
A.GetNewDocument Doc
```

Properties

```
GetPrivateTools
PrivateToolMenuEnabled
PrivateToolMenuString
PrivateToolSelected
PrivateToolSnapping
PrivateToolTargetEnabled
PrivateToolTargetType
PrivateToolTargetPosition
LicenceNumber
Language
Version
VersionStr
ApplicationDirectory
UserProfileDirectory
PublicDirectory
TemporaryDirectory
PythagorasLocalDirectory
CodeDescriptionFile
SystemClipboard
SystemDate
SystemParameters
IgnoreDataErrors
NilUUID
NegativeInfinity
PositiveInfinity
NaN
```

Methods

```
NewToolbarButton
RemoveToolbarButtons
EnableToolbarButton
NewUserMenu
RemoveUserMenu
AppendUserMenuItem
AppendUserPopupMenuItem
AppendUserSubMenuItem
RemoveUserMenuItem
RemoveUserSubMenuItem
EnableUserMenuItem
CheckUserMenuItem
OpenURL
OpenDocument
GetNewDocument
PutDocumentOnTop
GetActiveDocument
Wait
ContinueEvent
StopEvent
SetEventKeys
GetEventKeys
ConvertImage
GetDocuments
GetSymbolLibrary
GetCoordinateReferenceSystems
GetTiePointsSetCollection
TileWindows
```


CascadeWindows
 AddToRecentTextFiles
 OpenExternalFile
 GetPathName
 GetFileName
 SplitFileName
 ComposeFileName
 ChangeExtension
 SplitURI
 ComposeURI
 Intersection2Lines
 Intersection
 BestLine
 BestArc
 BestCircle
 ProjectionOnLine
 OptionSupported
 MacroOptionSupported
 ColumnValue
 SelectCondition
 NameCategory
 RowSetDefinition
 MatchCondition
 WhereCondition
 ColumnCondition
 CreateUUID
 IsNilUUID
 UUIDToString
 StringToUUID
 UUIDToDate
 Parse
 TrueColorToColor
 ColorToTrueColor
 GetUserDefaults
 GetAttributesSet
 SetAttributesSet
 SetVBADefaultAttributes
 IsNegativeInfinity
 IsPositiveInfinity
 IsInfinity
 IsNaN
 SortAll
 Sort
 SortValueIndex
 IncPointIdent
 InitSelectionCriteria
 ActivateTool
 SetStatusBarText
 InitVectoriseData

XML Methods

CreateXMLDocument
 ReadXMLDocument
 ParseStringAsXML
 GetXMLErrorPosition
 XMLReplaceWhites
 XMLCollapseWhites
 XMLWhiteSpaceNormalize
 XMLStringToBoolean
 XMLStringToInteger
 XMLDecimalStringToReal
 XMLStringToReal
 XMLStringToDateTime
 XMLStringToTime
 XMLStringToDate
 XMLStringToUTCDateTime
 XMLStringToUTCTime
 XMLDivideString
 XMLStringToBooleans

```

XMLStringToIntegers
XMLDecimalStringToReals
XMLStringToReals
XMLStringToDateTimeZ
XMLStringToTimesZ
XMLStringToDatesZ
XMLStringToDateTime
XMLStringToTimes
XMLStringToDates
XMLStringToUTCDateTime
XMLStringToUTCTimes
XMLBooleanToString
XMLIntegerToString
XMLRealToDecimalString
XMLRealToString
XMLDateTimeToString
XMLTimeToString
XMLDateToString
XMLUTCDateTimeToString
XMLUTCTimeToString
XMLUniteStrings
XMLBooleansToString
XMLIntegersToString
XMLRealsToDecimalString
XMLRealsToString
XMLDateTimesToString
XMLTimesToString
XMLDatesToString
XMLUTCDateTimeToString
XMLUTCTimesToString
XMLValidName

```

Properties

GetPrivateTools

Parameters

```
PrTools As PrivateTools
```

Description

Returns the PrivateTools object. See PrivateTools.

Example

```

Dim PrTools As PrivateTools

Set A = New Application
A.GetPrivateTools PrTools

```

PrivateToolMenuEnabled

Returns

```
Boolean
```

Description

Returns the status of the Private Tool menu item or sets the Private Tool menu item enabled or disabled. The Private Tool menu can only be selected if a Macro has executed the statement.

Note: this property became redundant. We recommend to use PrivateTools and PrivateTool objects introduced in Pythagoras 13. New functionality will only be added in the new classes.

Example

```

Dim A As Application

If A.PrivateToolMenuEnabled = False Then
    A.PrivateToolMenuEnabled= True
End If

```

PrivateToolMenuString

Returns

String

Description

Changes or returns the name of the Private Tool menu item.
The maximum length of a menu name is 32 characters.

Note: this property became redundant. We recommend to use `PrivateTools` and `PrivateTool` objects introduced in Pythagoras 13. New functionality will only be added in the new classes.

Example

```
Dim A As Application

If A.PrivateToolMenuString = "GPS Roving" Then
End If
```

PrivateToolSelected

Returns

Boolean

Description

Returns True if the Private Tool is active.
When setting, the Private Tool is activated or deactivated.

Note: this property became redundant. We recommend to use `PrivateTools` and `PrivateTool` objects introduced in Pythagoras 13. New functionality will only be added in the new classes.

Example

```
Dim A As Application
Dim GPSPos As XYZ

If A.PrivateToolSelected Then
    A.PrivateToolTargetPosition = GPSPos
    A.PrivateToolTargetEnabled = True
End If
```

PrivateToolCursor

Returns

Integer

Description

Sets the type of the cursor when the private tool is active. The cursor type remains active until either the macro library is terminated or when the property `PrivateToolCursor` is changed.
The value is any of the constants specified in section Chapter 140 - Constants, item *Cursors*.

Note: this property became redundant. We recommend to use `PrivateTools` and `PrivateTool` objects introduced in Pythagoras 13. New functionality will only be added in the new classes.

Example

```
Dim A As Application

If A.PrivateToolSelected Then
    A.PrivateToolCursor = ptCursorP1
End If
```

PrivateToolSnapping

Returns

ObjectTypes As Long

Description

Sets or returns the types of objects that Pythagoras will snap to **when the user has selected the Private Tool** before calling the OnMouseMove Event Macro. The value `ObjectTypes` specifies the types of objects (points, lines, ...) the cursor will snap to in the Private Tool Mode. So if `A.PrivateToolSnapping = ptPoint`, the cursor will only snap to points. `ObjectTypes` is defined as the sum of one or more constants of the type *ObjectTypes* defined in the section "Constants" of

this document. If `A.PrivateToolSnapping = ptAll`, the cursor will snap to all object types. If `A.PrivateToolSnapping = ptNoObjects`, the cursor will not snap to any object. Even if there is no `OnMouseMove` Event Macro, the cursor will snap only to the objects set by this property. When `ptLine` and `ptPoint` is in `ObjectTypes`, cursor will also snap to intersection points of lines. The default value when a Macro Library is loaded = `ptAll`.

Note: this property became redundant. We recommend to use `PrivateTools` and `PrivateTool` objects introduced in Pythagoras 13. New functionality will only be added in the new classes.

Example

```
Dim A As Application

set A = New Application
A.PrivateToolSnapping = ptPoint + ptLine
```

PrivateToolTargetEnabled

Returns

Boolean

Description

Returns or sets the status of the Target. If the target is enabled, a marker is displayed on the screen on the document position given by the property `PrivateToolTargetPosition`. The default target type = `ptGPSPositionFound`.

Note: this property became redundant. We recommend to use `PrivateTools` and `PrivateTool` objects introduced in Pythagoras 13. New functionality will only be added in the new classes.

Example

```
Dim A As Application

If A.PrivateToolTargetEnabled = False Then
    A.PrivateToolTargetEnabled = True
End If
```

PrivateToolTargetType

Returns

Integer

Description

Sets the type of the Target (marker) when `PrivateToolTargetEnabled = True`. The value can be either `ptGPSPositionFound` or `ptGPSSearchingPosition`.

Note: this property became redundant. We recommend to use `PrivateTools` and `PrivateTool` objects introduced in Pythagoras 13. New functionality will only be added in the new classes.

Example

```
Dim A As Application

If A.PrivateToolTargetEnabled = False Then
    A.PrivateToolTargetEnabled = True
    A.PrivateToolTargetType = ptGPSSearchingPosition
End If
```

PrivateToolTargetPosition

Returns

XYZ

Description

Sets the position of the target. The position is in local coordinates. The target is a marker on the screen, and is used to highlight a position. This could be for example the position received from GPS equipment. The display of the target has only effect if the Private Tool is selected. This is a write-only property.

Note: this property became redundant. We recommend to use `PrivateTools` and `PrivateTool` objects introduced in Pythagoras 13. New functionality will only be added in the new classes.

Example

```
Dim A As Application
Dim GPSPos As XYZ

A.PrivateToolTargetPosition = GPSPos
```

LicenceNumber

Returns

Variant

Description

This property returns the actual used license number of the Pythagoras protection key (dongle).

Example

```
Dim A As Application

set A = New Application
MsgBox CStr(A.LicenceNumber)
```

Language

Returns

Integer

Description

This property returns the language of the Pythagoras application.
For constants of languages see Chapter 140 - Constants

Example

```
Dim A As Application

set A = New Application
If A.Language = ptAppDutch Then
End If
```

Version

Returns

Integer

Description

This property returns the version of Pythagoras.
The returned value: main version * 100 + minor version.
Example: V12.05 will return the value 1205.
Every new value of Pythagoras will return a value that is larger than previous version.

Example

```
Dim A As Application

Set A = New Application
If A.Version >= 1210 Then ' Pythagoras 12.10 or larger
End If
```

VersionStr

Returns

String

Description

This property returns the version in string format of Pythagoras.

Example

```
Dim A As Application

set A = New Application
MsgBox A.VersionStr
```

ApplicationDirectory

Returns

String

Description

This property returns the directory of the Pythagoras application (Pythagoras.exe). It is advised not to place own files or to change existing files in this directory: starting from Windows Vista this directory can be write protected.

Example

```
Dim A As Application

set A = New Application
MsgBox A.ApplicationDirectory
```

UserProfileDirectory

Returns

String

Description

This property returns a full path to the personal directory of the current user. Normally return is "C:\Documents and Settings\user name\" for Windows XP and "C:\Users\user name\" for Windows Vista and subsequent operating systems where *user name* is a name of a logged in user. Standard subdirectories of this directory "Documents", "Pictures" and others are commonly used to save user's documents, pictures and other similar files.

Example

```
' Writing text file.
' For Windows XP
' "My documents" should be used instead of "Documents"
Dim A As Application

Set A = New Application
Open A.UserProfileDirectory() & "Documents\t.txt" For Output As 1
Print #1, "Hello!"
Close 1
```

PublicDirectory

Returns

String

Description

This property returns a full path to the common directories of all users. Normally return is "C:\Documents and Settings\All Users\" for Windows XP and "C:\Users\Public\" for Windows Vista and subsequent operating systems. Standard subdirectories of this directory "Documents", "Pictures" and others are commonly used to save common documents, pictures and other similar files.

Example

```
' Writing text file
Dim A As Application

Set A = New Application
Open A.PublicDirectory() & "Documents\t.txt" For Output As 1
Print #1, "Hello, all!"
Close 1
```

TemporaryDirectory

Returns

String

Description

This property returns a full path to the directory of the current user used for temporary files. Normally return is "C:\Documents and Settings\user name\Local Settings\Temp\" for Windows XP and "C:\Users\user name\AppData\Local\Temp\" for Windows Vista and

subsequent operating systems where *user name* is a name of a logged in user. Files stored to this directory are deleted when the operating system clears temporary files.

Example

```
' Writing temporary text file
Dim A As Application

Set A = New Application
Open A.TemporaryDirectory() & "t.txt" For Output As 1
Print #1, "Nothing valuable."
Close 1
```

PythagorasLocalDirectory

Returns

String

Description

This property returns a full path to the directory where work files of Pythagoras reside such as style library, macro library and others. Normally return is "C:\Documents and Settings*user name*\Local Settings\Application Data\Pythagoras\" for Windows XP and "C:\Users*user name*\AppData\Local\Pythagoras\" for Windows Vista and subsequent operating systems where *user name* is a name of a logged in user.

Example

```
' Copying macro library file
Dim A As Application

Set A = New Application
FileCopy A.PythagorasLocalDirectory() & "Pythagor.mlb", _
"F:\Pythagor.mlb"
```

CodeDescriptionFile

Returns

String

Description

This property returns the complete name of the Code Description File (CDF) that is loaded.

Example

```
Dim A As Application

set A = New Application
MsgBox A.CodeDescriptionFile
```

HiliteSelections

Returns

Boolean

Description

This property sets or resets the hiliting of the selected objects. When the user does any operation that stops hiliting, this property becomes FALSE. Many actions, also calling a macro, stops hiliting.

Example

```
Dim A As Application

set A = New Application
A.HiliteSelections = True
```

SystemClipboard

Parameters

String

Description

SystemClipboard reads text from, or writes text on the Windows clipboard. When writing, other information of the clipboard is lost.

Example

```
A.SystemClipboard = "Put this text on the clipboard"  
MsgBox A.SystemClipboard
```

SystemDate

Returns

String

Description

SystemDate returns the date. The format will depend on the settings of the date format of the Windows system. Alternative for this property : Date. See "Pythagoras VBA language elements", section "Date Functions".

Example

```
MsgBox A.SystemDate
```

AreaAnnotationFormat

Returns

Integer

Description

AreaAnnotationFormat sets or returns the format used by the Annotation Tool "Polygon Area". The value is either ptDefaultAreaFormat, ptAcres or ptHaAreCa.

Example

```
A.AreaAnnotationFormat = ptAcres
```

SystemParameters

Returns

SystemParameters

Description

Returns or sets the actual system parameters.

The system parameters are a set of variables that need to be used only in a few rare cases. In future versions of Pythagoras, new items will be added to the record structure SystemParameters.

Before modifying a system parameter, you must first read the actual parameters, and subsequently modify the field concerned. For a description of the Pythagoras data type SystemParameters see the Chapter 139 - *Pythagoras Data Types*.

Example

```
Dim A As Application  
Dim SystemPars As SystemParameters  
  
Set A = New Application  
SystemPars = A.SystemParameters      ' Read all parameters  
SystemPars.DWGExportAttribs = False  ' Change one parameter  
A.SystemParameters = SystemPars      ' Write all parameters  
                                     ' Only the parameter DWGExportAttribs will be affected.
```

IgnoreDataErrors

Returns

Boolean

Description

This property controls the behaviour of the program when an attempt is made to change a table property or a column property and the new property value contradicts to existing data in the table

If IgnoreDataErrors = False (initial setting), then an error is generated. If

IgnoreDataErrors = True, the program tries to change data or other property values correspondingly.

The default value of this property = False.

When IgnoreDataErrors = True, the following data or properties changes may take place:

a) Type of column is changed and some data cannot be converted: data becomes NULL if such values are not prohibited;

- b) Minimum or Maximum is changed and some data value contradicts this: contradicting values are changed to Minimum or Maximum correspondingly;
- c) Other column property (ObjectTypes, ReferencedColumn, PendingAction) is changed so that data becomes invalid: data becomes NULL if such values are not prohibited;
- d) Table ObjectTypes property is changed and one or more CAD objects linked with the table have a type not included in the new ObjectTypes: CAD object is unlinked from the table and the corresponding row is removed from the table.

Example

```
Dim A As Application

Set A = New Application
A.IgnoreDataErrors = True      ' No error will be generated
```

NilUUID

Returns

UUID

Description

Returns a so called NilUUID (see OpenGroup standard).

Example

```
Dim A As Application
Dim Id As UUID

Set A = New Application
Id = A.NilUUID
```

NegativeInfinity

Returns

Double

Description

This property returns the special real value $-\infty$ (negative infinity).

Example

```
Dim A As Application, X As Double
X = A.NegativeInfinity
```

PositiveInfinity

Returns

Double

Description

This property returns the special real value $+\infty$ (positive infinity).

Example

```
Dim A As Application, X As Double
X = A.PositiveInfinity
```

NaN

Returns

Double

Description

This property returns the special real value NaN (not a number).

Example

```
Dim A As Application, X As Double
X = A.NaN
```

Methods

NewToolbarButton

Parameters

TooltipText As String, MacroName As String, IconSpec As Variant, Enabled As Boolean, ButtonIndex As Long

Description

Creates a new button in the toolbar.

TooltipText is the tooltip given with this button.

MacroName : is the name of the macro to be executed when the button is clicked.

IconSpec : can be a Long value for an icon number that indicates the type of icon used (ptMacroIcon_1 .. ptMacroIcon_9, ptMacroIcon_A .. ptMacroIcon_Z, ptMacroIcon_Next, ptMacroIcon_Previous). Or it can be a string value specifying a custom icon file path. In this case an icon file can be specified by passing the full path or by the file name without path and extension. In the latter case it is a user responsibility to place the corresponding icon file with extension “.ICO” in the subdirectory “ICONS\MACROS” of Pythagoras installation directory. The icon file must be in the format of 24-bit with alpha channel.

If Enabled is set to TRUE, the button is enabled.

On return the variable ButtonIndex contains a unique identifier. This identifier can eventually be used to do enable or disable the toolbar button.

Example

```
Dim A As Application
Dim ButtonIndex As Long

set A = New Application
A.NewToolbarButton "Calculate", "MyFunction", ptMacroIcon_A, True, ButtonIndex
```

RemoveToolbarButtons

Parameters

None

Description

Removes the macro buttons in the toolbar.

Example

```
A.RemoveToolbarButtons
```

EnableToolbarButton

Parameters

ButtonIndex As Long, Enable As Boolean

Description

Enables or disables the toolbar button with given index.

Example

```
A.EnableToolbarButton ButtonIndex, FALSE
```

NewUserMenu

Parameters

MenuName As String

Description

Creates a new User Menu. MenuName is the title of the menu. The maximum number of items in the User menu is 27. The maximum number of items in a submenu is 32. The maximum length of a menu name is 32 characters.

Example

```
Dim A As Application

set A = New Application
A.NewUserMenu "My Menu"

A.NewUserMenu "My Menu"
```

RemoveUserMenu

Parameters

None

Description

Removes the User menu.

Example

```
Dim A As Application

set A = New Application
A.NewUserMenu "My Menu"
A.RemoveUserMenu
```

AppendUserMenuItem

Parameters

MenuName As String, Enabled As Boolean, Checked As Boolean, MacroName As String, MenuId As Long

Description

Appends a User Menu Item to an existing user menu.

MenuName : is the name of the User Menu Item. If MenuName is an empty string, a separator is inserted.

MacroName: is the name of the macro to be executed when the item is selected. If the macro with the given name does not exist, an error is raised unless the MenuName is an empty string.

If Enabled is set to TRUE, the item is enabled (selection possible). If Checked is set to TRUE, the menu item is checked.

On return the variable MenuId contains a unique identifier. This identifier can eventually be used to do operations on the menu item. E.g. remove or disable a menu item.

Example

```
Dim A As Application
Dim MenuId1 As LONG

set A = New Application
A.NewUserMenu "My Menu"
A.AppendUserMenuItem "Item 1", TRUE, FALSE, "MyFunction", MenuId1
```

AppendUserPopupMenuItem

Parameters

MenuName As String, Enabled As Boolean, MenuId As Long

Description

Creates a menu item with makes it possible to create submenus. On return the variable MenuId contains a unique identifier. This identifier must be used to create a submenu item. MenuId can also be used to do other operations on the menu item. E.g. remove or disable the menu item.

Example

```
Dim A As Application
Dim MenuId1 As Long, MenuId2 As Long, MenuId3 As Long

set A = New Application
A.NewUserMenu "My Menu"
A.AppendUserMenuItem "Menu item 1", TRUE, FALSE, "MacroItem1", MenuId1
A.AppendUserMenuItem "Menu item 2", TRUE, TRUE, "MacroItem2", MenuId2
A.AppendUserPopupMenuItem "Menu item 3", TRUE, MenuId3
```

AppendUserSubMenuItem

Parameters

MenuId As Long, MenuName As String, Enabled As Boolean, Checked As Boolean, MacroName As String, SubMenuId As Long

Description

Appends a menu item in a submenu. The value set in Menu sets in which submenu the item has to be placed.

MenuId : the menuId returned by AppendUserPopupMenuItem.

MenuName : the name of the sub menu. If MenuName is an empty string, a separator is inserted.

Enabled : if Enabled is TRUE, the menu item is enabled and can be selected.

Checked : if Checked is TRUE, the menu item is checked.

MacroName : the name of the macro to be executed when selecting the menu item. If the macro with the given name does not exist, an error is raised unless the MenuName is an empty string.

SubMenuId = On return this variable contains a unique identifier. This identifier can eventually be used to do operations on the submenu item. E.g. remove or disable a submenu item.

The maximum number of items in a submenu is 32.

Example

```
Dim A As Application
Dim MenuId1 As Long, MenuId2 As Long
Dim SubMenuId1 As Long, SubMenuId2 As Long

set A = New Application
A.NewUserMenu "My Menu"
A.AppendUserMenuItem "Menu Item 1", TRUE, FALSE, "MacroItem1", MenuId1
A.AppendUserPopupMenu "Menu Item 2", TRUE, MenuId2
A.AppendUserSubMenuItem MenuId2, "Submenu Item 1", TRUE, FALSE, "MacroSM1",
                                                                    SubMenuId1
A.AppendUserSubMenuItem MenuId2, "Submenu Item 2", TRUE, FALSE, "MacroSM2",
                                                                    SubMenuId2
```

RemoveUserMenuItem

Parameters

MenuId As Long

Description

Removes the specified menu item from the User Menu. MenuId has to be a value returned by AppendUserMenuItem or AppendUserPopupMenu

Example

```
Dim A As Application
Dim MenuId As Long

Set A = New Application
A.RemoveUserMenuItem MenuId
```

RemoveUserSubMenuItem

Parameters

MenuId As Long, SubMenuId As Long

Description

Removes a submenu item. MenuId and SubMenuId define the submenu to be deleted.

See also: "AppendUserSubMenuItem"

Example

```
Dim A As Application
Dim MenuId1 As Long, SubMenuId1 As Long

Set A = New Application
A.RemoveUserSubMenuItem MenuId1, SubMenuId1
```

EnableUserMenuItem

Parameters

MenuId As Long, SubMenuId As Long, Enabled As Boolean

Description

Enables or disables a menu item or a submenu item.

MenuId : the menu id returned by AppendUserMenuItem.

SubMenuId : the sub menu Id returned by AppendUserPopupMenu.

If SubMenuId = ptNilUserMenuIndex, then the MenuItem is enabled or disabled. Else the item of a submenu is disabled.

Enabled : If = TRUE, the menu item or submenu item is enabled and can be selected.

Example

```
Dim A As Application
```

```
Dim MyMenuId As Long

Set A = New Application
A.EnableUserMenuItem MyMenuId, ptNilUserMenuIndex, TRUE
```

CheckUserMenuItem

Parameters

MenuId As Long, SubMenuId As Long, Checked As Boolean

Description

Sets or removes a check mark.

MenuId : the menu id returned by AppendUserMenuItem.

SubMenuId : the sub menu Id returned by AppendUserPopupMenuItem.

If SubMenuId = ptNilUserMenuIndex, then function applies to the main menu item, else the function applies to the submenu item.

Checked : If = TRUE, the menu, or submenu item is checked.

Example

```
Dim A As Application
Dim MyMenuId As Long

Set A = New Application
A.CheckUserMenuItem MyMenuId, ptNilUserMenuIndex, TRUE
```

OpenURL

Parameters

URL As String

Description

Opens the URL in the default browser.

Example

```
Set A = New Application
A.OpenURL "www.pythagoras.net"
```

OpenDocument

Parameters

Name As String, OpenCopy As Boolean, Doc As Document, Optional
HandleDataOfSubdocuments As PtSynchronizeDictionaryData
:= ptDataRaiseError

Description

Opens the document. If OpenCopy = TRUE, then a copy of the document is opened. The document window will become the top window and the document will be displayed. If the document is already open, then it will become the top window.

A Document object, referring to the open document is returned.

If the name is empty, the open dialog box will come up.

If the name of a DXF or DWG file is given, the DXF/DWG file will be opened. The DXF / DWG dialog box will come up.

Input parameter HandleDataOfSubdocuments defines what to do when the overview document and a subdocument contain a database dictionary table with the same name and structure, but with different timestamps. Data in such tables can differ. See also description of PtSynchronizeDictionaryData enumeration. Value PtDataOfSubdocument should not be used here because it actually means that data from the last opened subdocument will be present in the dictionary table. Value ptDataDontOpenSubdocument may not be specified.

Example

```
Dim A As Application
Dim Doc As Document

Set A = New Application
A.OpenDocument "C:\Test.pyt", False, Doc
```

GetNewDocument

Parameters

Doc As Document

Description

Opens an Untitled document. The document window will become the top window.

Example

```
Dim Doc As Document

Set A = New Application
A.GetNewDocument Doc
```

PutDocumentOnTop

Parameters

Doc As Document

Description

Makes document the top window.

Example

```
Dim Doc As Document
Dim A As Application

Set A = New Application
A.PutDocumentOnTop Doc
```

GetActiveDocument

Parameters

Doc As Document

Description

Returns the active document (the document which is on top).

Example

```
Dim Doc As Document
Dim A As Application

Set A = New Application
A.GetActiveDocument Doc
```

Wait

Parameters

Milliseconds As Long

Description

Waits Milliseconds.

Example

```
Dim A As Application

Set A = New Application
A.Wait 300
```

ContinueEvent

Parameters

none

Description

When ContinueEvent is called when executing an Event Macro, and StopEvent is not called afterwards, then on leaving the Event Macro Pythagoras will still execute the event as if no Event Macro was called. When an Event Macro does not contain the StopEvent method, ContinueEvent will be taken as default.

Calling the method ContinueEvent only makes sense when executing an Event Macro.

Example

```
Dim A As Application
```

A.ContinueEvent

StopEvent

Parameters

None

Description

When StopEvent is called when executing an Event Macro, and ContinueEvent is not called afterwards, then on leaving the Event Macro Pythagoras will stop processing the event. In other words, StopEvent suppresses the standard behavior of the mouse events.

Calling the method StopEvent only makes sense when executing an Event Macro.

Example

```
Public Sub OnMouseDown()  
    Dim A As Application  
    Dim Doc As Document  
    Dim Obj As CadObject  
    Dim P As Point  
    Dim strError As String  
    Dim Sel As Selection  
    Dim T As Text  
    Dim CommentMode As Boolean  
  
    On Error GoTo ErrorHandler  
    A.GetActiveDocument Doc  
    Doc.GetObjectUnderCursor ptText + ptPoint, Obj  
    If Not (Obj is Nothing) Then  
        If (Obj.ObjectType = ptText) Then  
            Obj.getText T  
            Obj.Color = ptRed  
            A.StopEvent ' If not called, text will be selected.  
        Else  
            If (obj.ObjectType = ptPoint) And Not (T is Nothing) Then  
                Obj.GetPoint P  
                T.color = ptGreen  
                If CommentMode Then  
                    P.Comment = T.Content  
                Else  
                    P.Elevation = CDBL(T.Content)  
                End If  
                Set T = Nothing  
                A.StopEvent ' If not called, point will be selected.  
            Else  
                Beep  
            End If  
        End If  
    End If  
  
    ExitHere:  
    Set Doc = Nothing  
    Set Obj = Nothing  
    Exit Sub  
  
    ErrorHandler:  
    MsgBox Err.Description, vbCritical, strError  
End Sub 'StopEvent
```

SetEventKeys

Parameters

NrOfKeys As Integer, Keys() As Long

Description

The OnKeyDown and OnKeyUp event macros will only be called when a key in the set Keys is pressed. The value in the array is the virtual key code. The virtual keycodes, of the non-alpha-numeric keys, are predeclared constants. (See section Constants).

On Windows, the virtual keycodes for the keys A through Z are 041h - 05Ah. The keys 0 through 9 on the main clipboard have virtual keycodes 030h - 039h. The symbol keys (such as the key with the / and ?

symbols), have virtual keycodes of 128 and above, and they are often defined differently on international keyboards.

Example

```
Dim A As Application
Dim Keys() As Long

Redim Keys(1 To 2)
Keys(1) = 100
Keys(2) = 101
A.SetEventKeys 2, Keys
```

GetEventKeys

Parameters

NrOfKeys As Integer, Keys() As Long

Description

Returns the set of keys on which the OnKeyDown and OnKeyUp event macros will be called. The value in the array is the virtual key code. (See chapter "Constants", Virtual Key Codes). If any key in the set, the lower bound of the array will be = 1.

Example

```
Dim A As Application
Dim Number As Integer
Dim Keys() As Long

A.GetEventKeys Number, Keys
If Number > 0 Then
    If Keys(1) = 100 Then
        End If
    End If
End If
```

ComposeFileName

Parameters

Optional Drive As String = "",
Optional Server As String = "", Optional Shared As String = "",
Optional Path As String = "",
Optional Name As String = "", Optional Extension As Variant

Returns

Variant

Description

The method creates and returns a full name path from specified components. The components should be specified stripped as it is described in the description of SplitFileName method. You may specify Extension="" to force inclusion of a separating dot to a result string.

If input parameter values are wrong or contradictory, for example Server is empty but Shared is not, the method returns Error value.

Example

```
Dim A As Application, FileName1 As String, FileName2 As String
Dim Drive As String, Path As String, Name As String, Ext As String
Dim Ok As Boolean

Ok = A.SplitFileName (FileName1, Drive=Drive, Path=Path, _
    Name=Name, Extension=Ext)
If Ok Then
    FileName2 = A.ComposeFileName (Drive=Drive, Path=Path, _
        Name=Name&"1", Extension=Ext)
    Ok = FileName2 <> ""
End If
```

ConvertImage

Parameters

OrgFile As String, NewFile As String, GeoTiff As Boolean, Pos As XY, Scale As Double, Ok As Boolean

Description

Converts images (BMP, TIFF, JPEG) to the Pythagoras Image Format (PTIF)

OrgFile : the filename of the original image
NewFile : the filename of the converted PTIF image
GeoTiff : return parameter. Returns TRUE if the original file is a GeoTIFF file. Only if TRUE, the parameters Pos and Scale will be filled in, otherwise the values will be left unchanged.
Pos : Return parameter. Position of the top left corner of the image in the drawing.
 Only valid if parameter GeoTiff = True.
Scale : Return parameter. Scale of the image.
 Only valid if parameter GeoTiff = True.
Ok : Return parameter. If = True, the conversion was successful.

Example

```

Sub ConvertImage

Dim A As Application
Dim Doc As Document
Dim I As Image
Dim Drawing As overlay
Dim Pos As XY
Dim OrgFile As String
Dim NewFile As string
Dim GeoTiff As BOOLEAN
Dim Ok As BOOLEAN
Dim Scale As DOUBLE

Set A = New Application
A.GetActiveDocument D
D.GetOverlay ptDrawing, Drawing
Pos.x = 50
Pos.y = 50
Scale = 0.002
OrgFile = "C:\Test\Test.tif"
NewFile = "C:\test.ptif"
A.convertImage OrgFile, NewFile, GeoTiff, Pos, Scale, Ok
If Ok Then
    Drawing.CreateImage NewFile, Pos, GeoTiff, Scale, I
    I.Color = 180
    I.Layer = "Test"
    I.Displaylevel = -1
    I.Intensity = 90
    I.Inverted = TRUE
    I.SetReduce_Enlarge 150.0, 150.0
end If
End Sub
  
```

GetDocuments

Parameters

Docs As Documents

Description

Returns the set of open documents.

Example

```

Sub GetDocuments

Dim A As Application
Dim Docs As Documents

Set A = New Application
A.GetDocuments Docs
MsgBox "Nr of open documents : " & Str(Docs.NrOfDocuments)
End sub
  
```

GetSymbolLibrary

Parameters

Lib As Library

Description

Returns the System Library. This object can be used to obtain information about all symbols, line styles, hatch patterns and typical sections of the System Symbol Library.

Note : to obtain the Document Library of a particular document, use the Document method GetSymbolLibrary. (See Document object methods)

Example

```
Sub GetLibrary

Dim A As Application
Dim L As Library

set A = New Application
A.GetSymbolLibrary L

End Sub
```

GetCoordinateReferenceSystems

Parameters

CRSs As CoordinateReferenceSystems

Description

Returns the Coordinate Reference Systems (CRSs) of the Pythagoras CRS library.

Example

```
Dim A As Application
Dim CRSs As CoordinateReferenceSystems

Set A = New Application
A.GetCoordinateReferenceSystems CRSs
```

GetTiePointsSetCollection

Parameters

TPSC As TiePointsSetCollection

Description

Returns the collection of all Tie Points Sets in the Pythagoras library. (stored in .PLB file)

Example

```
Dim A As Application
Dim TPSC As TiePointsSetCollection

Set A = New Application
A.GetTiePointsSetCollection TPSC
```

GetDialogs

Parameters

None

Description

Returns the object Dialogs. See Chapter Dialogs.

Example

```
Sub GetDialogs

Dim A As Application
Dim MyDialogs As Dialogs

set A = New Application
A.GetDialogs MyDialogs
MyDialogs.RequestTextFileView "C:\Test.txt"

End Sub
```

TileWindows

Parameters

None

Description

Tiles all opened Pythagoras windows.

Example

```
Sub TileWindows

    Dim A As Application
    Dim D1 As Document, D2 As Document

    set A = New Application
    A.GetActiveDocument D1
    A.GetNewDocument D2
    A.TileWindows

End Sub
```

CascadeWindows

Parameters

None

Description

Cascades the open Pythagoras windows

Example

```
Sub CascadeWindows

    Dim A As Application
    Dim D1 As Document
    Dim D2 As Document

    set A = New Application
    A.GetActiveDocument D1
    A.GetNewDocument D2
    A.CascadeWindows

End Sub
```

AddToRecentTextFiles

Parameters

FileName As String

Description

Adds the file with name *FileName* to the list of recently opened text files.

.

Example

```
Dim A As Application

Set A = New Application
A.AddToRecentTextFiles "C:\Test.txt"
```

OpenExternalFile

Parameters

FileName As String, Ok As Boolean

Description

Opens the file *FileName*. On MS-Windows, Pythagoras will launch the same application as if the user "double clicked" on the document *FileName*.

If on return Ok = False, the file could not be opened.

Example

```
Dim A As Application
Dim Ok As Boolean

Set A = New Application
A.OpenExternalFile "C:\Test.doc", Ok
```

DownloadFile

Parameters

URL As String, FileName As String,
Optional Reload As Boolean = False

Returns

Long

Description

Downloads a file from the given URL and saves the file. The name of the file is given by parameter FileName. If Reload = True then the method tries to get from the Web a new version of the file; otherwise content of a file received earlier and saved in the system cache can be copied. If the file is downloaded and saved, then 0 is returned, otherwise returned value is an error code from the operating system.

Example

```
Dim A As Application
Dim Result As Long

Set A = New Application
Result = A.DownloadFile "FTP://pyth@ftpdrivehq.com/test.txt", "C:\Test.txt"
```

GetPathName

Parameters

FullFileName As String, PathName As String

Description

Returns a string that contains the full pathname of the file with the given name. Initially FileName must contain a file name with a complete path.

Example

```
Dim A As Application
Dim PathName As String

Set A = New Application
A.GetPathName "C:\Program Files\Pythagoras\Pythagoras.exe", PathName
MsgBox PathName ' Will display "C:\Program Files\Pythagoras\"
```

GetFileName

Parameters

FullFileName As String, FileName As String

Description

Returns a string that contains only the file name part, without the path, of the file.

Example

```
Dim A As Application
Dim FileName As String

Set A = New Application
A.GetFileName "C:\Program Files\Pythagoras\Pythagoras.exe", FileName
MsgBox FileName ' Will display "Pythagoras.exe"
```

SplitFileName

Parameters

FileName As String,
Optional Drive As String,
Optional Server As String, Optional Shared As String,
Optional Path As String,
Optional Name As String, Optional Extension As Variant

Returns

Boolean

Description

The method parses an input string as a full file name and returns separate components. Together with `ComposeFileName` method this method can be used to change a full file name. This method should not be applied to strings starting with "\\?\". If the input string cannot be parsed `False` will be returned. All file name parts are returned stripped: `Drive` without the final colon, `Server` and `Shared` without starting and final slashes, `Path` without the final slash and `Extension` without the starting dot. However starting slash will be present in `Path` when it designates a root directory. When extension is absent in `FileName` then `Extension` becomes either `Empty` or an empty string. `Empty` is stored when a separating dot is also absent. If you are not interested whether a separating dot is present then you may specify `Extension` as a string variable and will obtain an empty string in both cases.

Example

See method `ComposeFileName`.

ChangeExtension

Parameters

`FileName As String, Extension As String`

Returns

`Boolean`

Description

The method parses a file name string and then replaces a file name extension by the specified. If the input string cannot be parsed `False` will be returned.

Example

```
Dim A As Application, PythFileName As String, MyFileName As String
Dim Ok As Boolean

MyFileName = PythFileName
Ok = A.ChangeExtension (MyFileName, "dat")
```

SplitURI

Parameters

`URI As String,`
`Optional Scheme As String, Optional Authority As Variant,`
`Optional Path As String,`
`Optional Query As Variant, Optional Fragment As Variant`

Description

The method parses an input string as URI (uniform resource identifier, usually a URI is an URL) and returns separate components. Together with `ComposeURI` method this method can be used to change a URI.

All URI components are returned stripped: `Scheme` without the final colon, `Authority` without the starting slashes, `Query` without the initial question mark and `Fragment` without the starting number sign. However `Path` is returned with an initial slash.

Accordingly to URI standard this method makes difference between not defined and empty components `Authority`, `Query` and `Fragment`. A component is not defined when it is absent together with its initial or final separator; a component is empty when an only separator is present. For example a query component is not defined in URI string "http://example", but is empty in "http://example?". Corresponding parameters obtain `Empty` values for not defined components and empty string values for empty components. If you specify the corresponding parameters as string type variables then you obtain empty string values for both cases.

Example

See method `ComposeURI`.

ComposeURI

Parameters

`Optional Scheme As String = "", Optional Authority As Variant,`
`Optional Path As String = "",`
`Optional Query As Variant, Optional Fragment As Variant`

Returns

String

Description

The method creates and returns an URI from specified components. The components should be specified stripped as it is described in the description of `SplitURI` method. Not defined components should be omitted or specified as `Empty` values; empty components should be specified as empty strings.

When input parameter values are contradictory, i.e. when `Authority` and `Path` both are present and `Path` does not start with "/" then the method returns `Error` value.

Example

```
Dim A As Application, URL1 As String, URL2 As String
Dim Authority As Variant, Path As String

A.SplitURI URL1, Authority=Authority, Path=Path
URL2 = A.ComposeURI (Scheme="http", Authority=Authority, _
                    Path=Path, Query="page=5")
```

Intersection2Lines

Parameters

L1 As Line2D, L2 As Line2D, Pnt As XY, Ok As Boolean

Description

`Intersection2Lines` calculates the intersection of 2 lines. If the lines are parallel, `Ok` returns `False`, otherwise `Ok = True`.

Example

```
Dim A As Application
Dim L1 As Line2D, L2 As Line2D
Dim Pnt As XY
Dim Ok As Boolean

Set A = New Application
L1.p1.x = 10.0
L1.p1.y = 10.0
L1.p2.x = 20.0
L1.p2.y = 20.0
L2.p1.x = 10.0
L2.p1.y = 20.0
L2.p2.x = 20.0
L2.p2.y = 10.0
A.Intersection2Lines L1, L2, Pnt, Ok
```

Intersection

Parameters

O1 As CadObject, O2 As CadObject, Pnt As XY, Ok As Boolean,
Optional SecondSolution := False

Description

`Intersection` calculates the intersection of 2 objects. The objects may be a combination of the following object types : Line - Line, Line - Arc, Line - Path and Line - Curve.

If there is more then one intersection if the `CadObject` is a curve, the one closest to the midpoint of the line is returned.

In case of Line - Arc and there are 2 solutions, the optional parameter `SecondSolution` allows to return the second solution.

If the objects do not intersect, `Ok` returns `False`, otherwise `Ok = True`.

Example

```
Dim A As Application
Dim L As Line, B As Arc
Dim O1 As CadObject, O2 As CadObject
Dim Pnt1 As XY, Pnt2 As XY
Dim Ok As Boolean

L.GetCadObject O1
B.GetCadObject O2
A.Intersection O1, O2, Pnt1, Ok
```

```
A.Intersection O1, O2, Pnt2, Ok, SecondSolution:=True
```

BestLine

Parameters

```
NrOfPoints As Long, Points() As XYZ, L1 As Line3D, Ok As Boolean
```

Description

Calculates a line so that the sum of the squares of the distances of the points to the line is minimal.

Example

```
Dim A As Application
Dim L As Line3D
Dim Pnts() As XYZ
Dim Ok As Boolean

Set A = New Application
A.BestLine 5, Pnts, L, Ok
```

BestArc

Parameters

```
NrOfPoints As Long, Points() As XYZ, _
P1 As XYZ, P2 As XYZ, Center As XY, Ok As Boolean
```

Description

Calculates an arc so that the sum of the squares of the distances of the points to the arc is minimal.

Example

```
Dim A As Application
Dim P1 As XYZ, P2 As XYZ, Center As XY
Dim Pnts() As XYZ
Dim Ok As Boolean

Set A = New Application
A.BestArc 5, Pnts, P1, P2, Center, Ok
```

BestCircle

Parameters

```
NrOfPoints As Long, Points() As XYZ, _
Center As XYZ, RadiusPnt As XYZ, Ok As Boolean
```

Description

Calculates a circle so that the sum of the squares of the distances of the points to the circle is minimal.

Example

```
Dim A As Application
Dim Center As XYZ, RadiusPnt As XYZ
Dim Pnts() As XYZ
Dim Ok As Boolean

Set A = New Application
A.BestCircle 5, Pnts, Center, RadiusPnt, Ok
```

ProjectionOnLine

Parameters

```
L As Line3D, Pnt As XY
```

Returns

```
ProjPnt As XYZ
```

Description

Calculates the projection of a point on a line. The elevation of the projection point is derived from the elevation of the starting and end point of the line.

Example

```
Dim A As Application
Dim L As Line3D
Dim Pnt As XY, ProjPnt As XYZ

Set A = New Application
```

```
ProjPnt = A.ProjectionOnLine(L, Pnt)
```

OptionSupported

Parameters

Option As Long

Returns

Boolean

Description

Returns True if the option is supported.

Valid options: ptPO_RW, ptPO_Lite, ptPO_DTM, ptPO_DC, ptPO_3D, ptPO_RD, ptPO_I, ptPO_DB (GIS), ptPO_SB (Streckenberechnung)

Example

```
Dim A As Application

Set A = New Application
If A.OptionSupported(ptPO_DTM) THEN
    ' DTM option is available
End If
```

MacroOptionSupported

Parameters

Option As Long

Returns

Boolean

Description

Returns True if the macro is supported.

Valid macro's: ptMO_CM (Cadmap), ptMO_GRB (GRB)

Example

```
Dim A As Application

Set A = New Application
If A.MacroOptionSupported(ptPO_CM) THEN
    ' Cadmap is available
End If
```

ColumnValue

Parameters

Column As Variant, Value As Variant

Returns

ColumnValue

Description

This method initializes a variable of type ColumnValue.

Example

```
Dim A As Application
Dim Value As ColumnValue

Set A = New Application
Value = A.ColumnValue("Population", 100000)
```

SelectCondition

Parameters

Column As Variant, Condition As PtCompare, Value As Variant

Returns

SelectCondition

Description

This method initializes a variable of type SelectCondition

Example

```
Dim A As Application
Dim Condition As SelectCondition

Condition = A.SelectCondition("Country", ptCompareEqual, "Belgium")
```

NameCategory

Parameters

Name As String

Returns

Integer

Description

NameCategory returns one of the following values: ptStrictRegularName, ptRegularName, ptIrregularName, ptInvalidName.

- ptStrictRegularName: the safest name containing only Latin letters (A..Z), digits (0..9) and the "_" (underscore). The first character of name is a letter.
- ptRegularName: name is regular according SQL92 standard. The name may contain letters and digits of any alphabet and also the characters \$ and #. Using such name may cause problems when the database is exported and used in other database applications.
- ptIrregularName: name may be used in a Pythagoras database but it is not recommended since it may cause problems when used by other database programs.
- ptInvalidName: name may not be used in a Pythagoras database.

Example

```
Dim A As Application
Dim TableName As String

Set A = New Application
If A.NameCategory(TableName) = ptInvalidName THEN
    ' Name may not be used
End If
```

RowSetDefinition

Parameters

RowSetType As PtRowSetType,
Optional Name As Variant = Empty (String or DBQuery),
Optional ObjectTypes As Long = 0,
Optional Overlay As PtRowSetBaseOverlay = ptCurrentOverlay,
Optional SheetName As String = ""

Returns

RowSetDefinition

Description

This method initializes a variable of type RowSetDefinition. Argument values are assigned to result fields having the same names.

Example

See class DBQuery method AddDimension

MatchCondition

Parameters

Kind As PtMatchConditionKind,
Optional Expression As String = "",
Optional Distance As Double = 0.0

Returns

MatchCondition

Description

This method initializes a variable of type MatchCondition. Argument values are assigned to result fields having the same names.

Example

```
Dim A As Application
Dim Condition As MatchCondition
```

```
Condition = A.MatchCondition (ptExpression, "main.id = aux.senior")
```

WhereCondition

Parameters

```
Compare As PtCompare,  
Optional Column As Variant (PtObjectFunction or String),  
Optional Value As Variant, Optional ValueType As Byte = vbNull,  
Optional Aggregation As PtAggregateFunction = ptFunctionNil
```

Returns

```
WhereCondition
```

Description

This method initializes a variable of type WhereCondition.

When Column is of PtObjectFunction type then result field Feature is initialized to this value.

When Column is of String type then result field Feature is initialized to ptFunctionEmpty and result field Column is initialized to this value.

Column is not necessary when Aggregation = ptCount.

Values of other parameters are assigned to result fields having the same names.

Parameter Value is not necessary when Compare is ptCompareIsNull, ptCompareIsNotNull, ptCompareIsTrue or ptCompareIsFalse.

See also DBQuery.AddWhere method description about ValueType parameter usage.

Example

See class DBQuery method AddWhere

ColumnDefinition

Parameters

```
Name As String, Optional Description As String = "",  
Optional IsComputed As Boolean = False,  
Optional RowSet As String = "", Optional Expression As String = "",  
Optional Aggregation As PtAggregateFunction = ptFunctionNil,  
Optional Format As String = "", Optional Fractional As Byte = 2
```

Returns

```
ColumnDefinition
```

Description

This method initializes a variable of type ColumnDefinition. Argument values are assigned to result fields having the same names.

Explicitly or implicitly defined fields of the result value can occasionally be incomplete or contradictory. Nevertheless this method always completes successfully. An error can happen later when such definition will be used.

Example

See class Database method CreateSimpleQuery.

CreateUUID

Parameters

```
no parameters
```

Returns

```
UUID
```

Description

CreateUUID returns a new generated UUID.

For more information about UUID see <http://en.wikipedia.org/wiki/UUID>

Example

```
Dim A As Application  
Dim Id As UUID  
  
Set A = New Application  
Id = A.CreateUUID()
```

IsNilUUID

Parameters

Id As UUID

Returns

Boolean

Description

IsNilUUID returns true if the UUID = NilUUID.

Example

```
Dim A As Application
Dim Id As UUID

If A.IsNilUUID(Id) Then
End If
```

EqualUUID

Parameters

Id1 As UUID, Id2 As UUID

Returns

Boolean

Description

EqualUUID returns true if bot UUIDs are identical.

Example

```
Dim A As Application
Dim Id1 As UUID, Id2 As UUID

Set A = New Application
If A.EqualUUID(Id1, Id2) Then
    ' Id1 = Id2
End If
```

UUIDToString

Parameters

Id As UUID

Returns

String

Description

UUIDToString returns a string representation (according OpenGroup standard) of the UUID.

Note: the string is **not** enclosed in braces.

Example

```
Dim A As Application
Dim Id As UUID
Dim S As String

S = A.UUIDToString(Id)
```

StringToUUID

Parameters

S As String, Id As UUID

Returns

Boolean

Description

StringToUUID converts a string to a UUID.

The method returns False if the string could not be converted to a UUID.

Example

```
Dim A As Application
Dim Id As UUID
Dim S As String
```

```

If A.StringToUUID(S, Id) Then
    ' Conversion to UUID ok
End If

```

UUIDToDate

Parameters

Id As UUID

Returns

Date

Description

UUIDToDate returns a date extracted from the UUID. The result is meaningless if the UUID = NilUUID.

Example

```

Dim A As Application
Dim Id As UUID
Dim D As Date

D = A.UUIDToDate(Id)

```

Parse

Parameters

Expression As String, Optional Table As Variant = Empty,
Optional ResultType As Byte, Optional ErrorText As String,
Optional ErrorStart As Long, Optional ErrorEnd As Long

Returns

PtSyntaxError

Description

Parse parses Expression. The expression is a string. The syntax is the same as specified for expressions used in the Find-function and Thematic Map of Pythagoras and is described in the Pythagoras reference manual. The method Parse accepts only the international (not localized) version of the language.

Table may be of type DBTable or DBQuery. If this value is neither Empty nor Nothing then expression may contain names of columns of the given rowset. Return value is = ptSyntaxSuccess when Expression is a valid expression.

If return value = ptSyntaxSuccess then ResultType is the type of the result of the expression: vbLong, vbDouble, vbString, vbBoolean ... ptXY, ptCadObjectType. For Expression = "NULL", ResultType = vbNull is returned.

If the return value <> ptSyntaxSuccess, the 3 last parameters will contain more information about the error.

ErrorText will contain textual description of the error. ErrorStart and ErrorEnd will contain starting and ending position of the error in Expression.

Example

```

Dim A As Application
Dim Expr As Expression
Dim ErrorMessage As String
Dim ExpType As Byte, Start As Long, End As Long

If _
    A.Parse(Expr, MyTable, ExpType, ErrorMessage, Start, End) _
    = ptSyntaxSuccess _
Then
    If ExpType = vbDouble Then
        ..
    Else
        ' Inform the user that the expression type is wrong
    End If
Else
    ' Give syntax error
End If

```

TrueColorToColor

Parameters

TrueColor As RGB, Match As Boolean

Returns

Color As Integer

Description

This method converts a true color RGB value to its nearest color index.

If Match = True then the method ColorToTrueColor(Color) will return the same RGB values.

If Match = False, the method ColorToTrueColor(Color) will return an approximate value.

Example

```
Dim A As Application
Dim Color As Integer
Dim TrueColor As RGB
Dim Match As Boolean

TrueColor.Red = 25
TrueColor.Green = 190
TrueColor.Blue = 20
Color = A.TrueColorToColor(TrueColor, Match)
```

ColorToTrueColor

Parameters

Color As Integer

Returns

TrueColor As RGB

Description

This method returns the true color RGB values of a color index.

Example

```
Dim A As Application
Dim Color As Integer
Dim TrueColor As RGB
Dim Match As Boolean

TrueColor = A.ColorToTrueColor(Color)
```

GetUserDefaults

Parameters

UDs As UserDefaults

Description

This method returns an object of type UserDefaults. This object contains the list of all user defaults.

Example

```
Dim App As Application
Dim UDs As UserDefaults

Set App = New Application
App.GetUserDefaults UDs
```

GetAttributesSet

Parameters

Type As Integer, Attribs As AttributesSet

Description

This method returns the AttributesSet that corresponds with the requested type.

Type = ptSystem:AttributesSet of the System Defaults of Pythagoras.

Type = ptDefault:AttributesSet of the Current Defaults of Pythagoras.

Type = ptUserDefault:AttributesSet of the Current Defaults overwritten by the User Defaults.

If the attributes in case `Type = ptUserDefault` are used, creating a `CadObject` with the styles defined in `AttributesSet`, will give the same attributes as if the `CadObject` was created interactively in Pythagoras.

Example

```
Dim A As Application
Dim Attribs As AttributesSet

A.GetAttributesSet ptUserDefault, Attribs
```

SetAttributesSet

Parameters

Attribs As AttributesSet

Description

This method sets the given `AttributesSet` as the Current Defaults of Pythagoras. An active user default will be put inactive.

Example

```
Dim A As Application
Dim Attribs As AttributesSet

A.SetAttributesSet Attribs
```

SetVBADefaultAttributes

Parameters

Type As Integer

Description

This method defines which attributes are used when a `CadObject` is created by VBA.

When creating a new `CadObject` and Type has been set to:

`ptSystem`: the System Default attributes of Pythagoras are used.

`ptDefault`: the Current Defaults of Pythagoras are used.

`ptUserDefault`: the Current Defaults overwritten by the User Defaults are used.

When this method has not been called the default attributes correspond with `ptUserDefault`.

This setting remains only valid till a macro is terminated after termination the default attributes will be set to `ptUserDefault`. As a consequence, a VBA program that always wants to create objects using System Defaults, should always call this method before starting to create objects.

Example

```
Dim A As Application

A.SetVBADefaultAttributes ptSystem
```

IsNegativeInfinity

Parameters

Source As Double

Returns

Boolean

Description

These methods returns `True` when Source is $-\infty$ (negative infinity).

Example

```
Dim A As Application, X As Double

If Not A.IsNegativeInfinity(X) And A.IsPositiveInfinity(X) And Not A.IsNaN(X)
Then
' We are here if X is a finite real value
' . . .
End If
```

IsPositiveInfinity

Parameters

Source As Double

Returns

Boolean

Description

These methods returns True when Source is $+\infty$ (positive infinity).

Example

See Application.IsNegativeInfinity

IsInfinity

Parameters

Source As Double

Returns

Boolean

Description

These methods returns True when Source is either $-\infty$ or $+\infty$.

Example

```
Dim A As Application, X As Double

If Not A.IsInfinity(X) And Not A.IsNaN(X) Then
' We are here if X is a finite real value
' . . .
End If
```

IsNaN

Parameters

Source As Double

Returns

Boolean

Description

These methods return True when Source is any kind of not-a-number.

Example

See Application.IsInfinity

SortAll

Parameters

Values() As Variant, Optional Ascending As Boolean = True,
Optional StringSortType As Integer = ptSortAlphaNumerical

Description

This method will sort all values in the array in ascending (default) or descending order.

The allowed subtypes of value are: Integer, Long, Single, Double, String, Boolean and Date.

All values should be of the same subtype.

If the subtype of the Values = String, the optional parameter StringSortType is either ptSortAlphaNumerical or ptSortAlphabetical.

ptSortAlphaNumerical: this sorting corresponds with the way Windows sorts file names in Windows Explorer: "ABC30" > "ABC4". The string may contain a mixture of numerical and alpha parts.
ptSortAlphabetical: this is pure alphabetical sorting: "ABC30" < "ABC4".

If one of the above conditions is not satisfied, the error 1522, "An actual variant argument has improper type", will be raised.

Example

```
Dim App As Application
Dim V() As Variant

Set App = New Application
Redim V(1 To 5000)
```

```
' Fill array
App.SortAll V
```

Sort

Parameters

```
Values() As Variant, Count As Long, Optional Ascending As Boolean = True,
Optional StringSortType As Integer = ptSortAlphaNumerical
```

Description

This method will sort the first n values in ascending (default) or descending order. The number (n) is given by the parameter Count. If Count is larger than the size of Values, the error 1501: "Invalid Parameter" will be raised.

See also the method SortAll.

Example

```
Dim App As Application
Dim V() As Variant

Set App = New Application
Redim V(1 To 5000)
' Fill array
App.Sort V, 200, False
```

SortValueIndex

Parameters

```
Data() As ValueIndex, Count As Long, Optional Ascending As Boolean = True,
Optional StringSortType As Integer = ptSortAlphaNumerical
```

Description

This method will sort the first n values of array data in ascending (default) or descending order of the field value.

See also the method SortAll.

Example

```
Type AddressBook
    Name As String
    Street As String
    Number As Integer
    ZipCode As String
End Type

Dim R(1 TO 50) As AddressBook      ' Not sorted addressbook
Dim R2(1 TO 50) As AddressBook    ' Sorted addressbook
Dim D(1 TO 50) As ValueIndex      ' See Pythagoras Data Types

Set App = New Application
For i = LBound(R) to UBound(R)
    D(i).Value = R(i).Name
    D(i).Index = i
Next i

App.SortValueIndex D, 50

For i = LBound(D) to UBound(D)
    R2(i) = R(D(i).Index)
Next i
```

IncPointIdent

Parameters

```
PointIdent As String, Optional AlphaIncrement As Boolean = False
```

Description

This method increments PointIdent. This parameter is considered to be a point number.

Incrementing the value behaves as the autoincrement of the point number in Pythagoras.

If the parameter AlphaIncrement = True, passing a string that does not contain numeric characters will be alphabetically incremented. E.g. "A" will become "B".

Example

```
Sub PM_Test
    Dim A As Application
    Dim PntId As String

    Set A = New Application
    PntId = "A"
    A.IncPointIdent PntId, True    'On return PntId becomes "B"

    PntId = "A01"
    A.IncPointIdent PntId        'On return PntId becomes "A02"
End Sub
```

InitSelectionCriteria

Parameters

SelCriteria As SelectionCriteria

Description

This method initialises SelectionCriteria.

All fields of SelectionCriteria will be set to there default values which for most = binary value 0.

In VBA the binary value 0 will make booleans = 0.

Exception: SelectionCriteria.Objects = ptAll,

SelectionCriteria.PointDistance = 0.000001.

Every time a new variable SelectionCriteria is used, the VBA program should call this method.

Example

```
Sub PM_Test
    Dim A As Application
    Dim SelCriteria As SelectionCriteria

    Set A = New Application
    A.InitSelectionCriteria SelCriteria    'All variables are initialised
    SelCriteria.Objects = ptPoint
    SelCriteria.OnLayer = True
    SelCriteria.Layer = "VBA Layer"
    A.InitSelectionCriteria SelCriteria    'Will undo the above changes
End Sub
```

ActivateTool

Parameters

Tool As Integer

Description

This method sets the active tool.

Tool = ptToolSelect, ptToolZoomOut, ptToolZoomIn, ptToolFitWindow, ptToolDragWindow, ptToolPoint, ptToolFreeLine, ptDefineParallelLine, ptToolPerpendicularLine, ptToolArcTanglLine, ptToolArcTang2Lines, ptToolArc3P, ptToolArcCenterR, ptToolCircleCenterR, ptToolCircle3P, ptToolXAxis, ptToolPolygon, ptToolPath, ptToolOpenCurve, ptToolClosedCurve, ptToolEditCurve, ptToolText, ptToolRichText, ptToolObliqueText, ptToolObliqueRichText, ptToolAnSegmentLengths, ptToolAnPolygonArea, ptToolAnPolygonLengthsAndArea, ptToolAnCoordinates, ptToolAnPointNumber, ptToolAnElevation, ptToolAnXYMinusOffset, ptToolAnComment, ptToolAnHorizontalLength, ptToolAnSlopeLength, ptToolAnTotalLength, ptToolAnBearing, ptToolAnDifferenceInHeight, ptToolAnGradient, ptToolAnRadius, ptToolAnAngle, ptToolAnDeltaX, ptToolAnDeltaY, ptToolAnVolume, ptToolAlignLeft, ptToolAlignHorizontalCentered, ptToolAlignRight, ptToolAlignTop, ptToolAlignVerticalCentered, ptToolAlignBottom, ptToolAlignObject, ptToolFreeClothoid, ptToolClothoidLineCircle, ptToolClArcClTang2Lines, ptToolCutOffInsideAllObjects, ptToolCutOffInsideWithoutImages, ptToolCutOffInsideImages, ptToolCutOffInsidePointClouds, ptToolCutOffOutsideAllObjects, ptToolCutOffOutsideWithoutImages, ptToolCutOffOutsideImages, ptToolCutOffOutsidePointClouds, ptToolErasePointCloud, ptToolEraseImageArea, ptToolEraseImageAreaWhite,

```
ptToolEraseImageAreaBlack, ptToolClipPointCloud,
ptToolClipImageAreaWhite, ptToolClipImageArea,
ptToolClipImageAreaBlack, ptToolCreateParallelObject, ptToolCutLine,
ptToolDivideLine, ptToolExtendLine, ptToolJoinLines or
ptToolIntersectObjects.
```

Example

```
Sub PM_Test
    Dim A As Application

    Set A = New Application
    A.ActivateTool ptToolSelect
End Sub
```

SetStatusBarText

Parameters

```
text As String
```

Description

This method sets the text of the status bar to the given text.

Example

```
Sub PM_Test
    Dim A As Application

    Set A = New Application
    A.SetStatusBarText "Hello world!"
End Sub
```

InitVectoriseData

Parameters

```
VectData as VectoriseData
```

Description

This method initializes the vectorisation data used for the vectorisation of point clouds.

type is set to ptVectoriseSection

isBox: is false

BLN, TLN, BRN, BLF, p1 and p2: contain all zeros. This indicates there is no section box or section plane.

Example

```
Sub PM_Test
    Dim A As Application
    Dim VectData As VectoriseData

    Set A = New Application
    A.InitVectoriseData VectData
End Sub
```

XML Methods

CreateXMLDocument

Parameters

```
Name As String
```

Returns

```
XMLDocument
```

Description

Creates and returns a new empty XML document with the main element having the specified. If Name is invalid then Nothing is returned.

Example

```
Const eAcute = Chr(&HC3)+Chr(&HA8)
' Small Latin e whith acute in UTF-8
```

```

Dim A As Application, Doc As XMLDocument
Dim Region As XMLElement, Province As XMLElement

Sub AddProvince (ByVal Name As String, _
    ByVal Capital As String, ByVal Population As Long, _
    ByVal Languages As String)
    Set Province = Region.SubItems.AddElement ("province")
    Province.AddAttribute "name", Name
    Province.AddAttribute "capital", Capital
    Province.AddAttribute "population", _
        A.XMLIntegerToString(Population)
    Province.AddAttribute "language", Languages
End Sub

Sub PM_CreateBelgiumXMLFile
    Dim Comment As XMLComment
    Set A = New Application
    Set Doc = A.CreateXMLDocument ("country")
    Doc.Version = xmlv1_0
    Doc.Main.AddAttribute "name", "Belgium"
    Doc.Main.AddAttribute "capital", "Brussels"
    Doc.Main.AddAttribute "censusDate", _
        A.XMLDateToString (#2010-01-01#)
    Doc.Main.AddAttribute "population", "10839905"
    Doc.Main.AddAttribute "language", "DEU DUT FRA"
    Set Region = Doc.Main.SubItems.AddElement ("region")
    Region.AddAttribute "name", "Brussels Capital region"
    Region.AddAttribute "population", "1083538"
    Region.AddAttribute "language", "DUT FRA"
    Set Comment = _
        Region.SubItems.AddComment ( _
            "the region is not divided to provinces, and often " & _
            "is recognized as a province" _
        )
    Set Region = Doc.Main.SubItems.AddElement ("region")
    Region.AddAttribute "name", "Flemish region"
    Region.AddAttribute "capital", "Brussels"
    Region.AddAttribute "population", "6251983"
    Region.AddAttribute "language", "DUT"
    AddProvince "Antwerp" , "Antwerp", 1744862, "DUT"
    AddProvince "East Flanders" , "Ghent" , 1432326, "DUT"
    AddProvince "Flemish Brabant", "Leuven", 1076924, "DUT"
    AddProvince "Limburg" , "Hasselt", 838505, "DUT"
    AddProvince "West Flanders" , "Bruges" , 1159366, "DUT"
    Set Region = Doc.Main.SubItems.AddElement ("region")
    Region.AddAttribute "name", "Wallon region"
    Region.AddAttribute "capital", "Namur"
    Region.AddAttribute "population", "3498384"
    Region.AddAttribute "language", "DEU FRA"
    Set Comment = _
        Region.SubItems.AddComment ( _
            "the region is also called Wallonia")
    AddProvince "Hainaut" , "Mans" , 1309880, "FRA"
    AddProvince "Wallon Brabant", "Wavre" , 379515, "FRA"
    AddProvince "Li"&eAcute&"ge", "Li"&eAcute&"ge", 1067685, _
        "DEU FRA"
    AddProvince "Luxembourg", "Arlon", 269023, "FRA LTZ"
    AddProvince "Namur" , "Namur", 472281, "FRA"
    If Doc.Save ("c:\Belgium.xml", MaxLine:=68) <> xmlcSuccess Then
        MsgBox "XML document cannot be written"
    End If
End Sub
Dim A As Application

```

Created file looks as:

```

<?xml version="1.0" standalone="yes" ?>
<country
    capital="Brussels"
    censusDate="2010-01-01"
    language="DEU DUT FRA"
    name="Belgium"

```

```

    population="10839905"
  >
  <region
    language="DUT FRA"
    name="Brussels Capital region"
    population="1083538"
  >
  <!--
    the region is not divided to provinces, and often is
    recognized as a province
  -->
</region>
<region
  capital="Brussels"
  language="DUT"
  name="Flemish region"
  population="6251983"
>
<province
  capital="Antwerp"
  language="DUT"
  name="Antwerp"
  population="1744862"
/>
<province
  capital="Ghent"
  language="DUT"
  name="East Flanders"
  population="1432326"
/>
<province
  capital="Leuven"
  language="DUT"
  name="Flemish Brabant"
  population="1076924"
/>
<province
  capital="Hasselt"
  language="DUT"
  name="Limburg"
  population="838505"
/>
<province
  capital="Bruges"
  language="DUT"
  name="West Flanders"
  population="1159366"
/>
</region>
<region
  capital="Namur"
  language="DEU FRA"
  name="Wallon region"
  population="3498384"
>
  <!-- the region is also called Wallonia -->
  <province
    capital="Mans"
    language="FRA"
    name="Hainaut"
    population="1309880"
  />
  <province
    capital="Wavre"
    language="FRA"
    name="Wallon Brabant"
    population="379515"
  />
  <province
    capital="Liège"
    language="DEU FRA"

```

```

    name="Liège"
      population="1067685"
    />
  <province
    capital="Arlon"
    language="FRA LTZ"
  name="Luxembourg"
  population="269023"
  />
  <province
    capital="Namur"
    language="FRA"
    name="Namur"
    population="472281"
  />
</region>
</country>

```

ReadXMLDocument

Parameters

```

FullFilename As String,
Optional ReadComments As Boolean = False,
Optional ReadPI As Boolean = False,
Optional Completion As XMLCompletionCode,
Optional Line As Long, Optional Column As Long
Optional Reload As Boolean = False

```

Returns

XMLDocument

Description

The method opens the specified file together with possible referenced files, and returns a new XML document filled by information read from the files. If some file has not been successfully read or has wrong syntax Nothing is returned.

Parameter `FullFileName` may be either a full path to a local file or an absolute URL to the file in the Web.

Parameters `ReadComments` and `ReadPI` define whether comments and PIs correspondingly will be stored to the new document.

Parameter `Completion` will contain code of error in case of failure, or `xmlcSuccess` in case of success.

Parameters `Line` and `Column` may contain information about position where an error was encountered during reading. Numerations of lines and positions start from 1.

Parameter `Reload = True` makes the method to get new versions of the files from the Web; otherwise copies of files received earlier and saved in the system cache can be read

If an error was encountered analyzing an embedded internal or external entity or an external DTD, additional information about position of the error can be obtained calling `GetXMLErrorPosition` method.

Example

See example XMLDocument

ParseStringAsXML

Parameters

```

InputString As String,
Optional IncludeComments As Boolean = False,
Optional IncludePI As Boolean = False,
Optional Completion As XMLCompletionCode, Optional Column As Long

```

Returns

XMLDocument

Description

The method parses `InputString` as an XML textual document, creates an empty XML document, fills it by information gotten from the parsed string and returns the created document. If the string was not successfully parsed Nothing is returned.

Parameters IncludeComments, IncludePI, Completion and Column are analogs of parameters ReadComments, ReadPI, Completion and Column of method ReadXMLDocument, see its description.

Example

```
Dim A As Application, Doc As XMLDocument

Set A = New Application
Set Doc = _
    A.ParseStringAsXML("<rectangle coordinates='13 15 68 98' />")
```

GetXMLErrorPosition

Parameters

Position() As Location

Returns

Boolean

Description

This method returns additional information about error position during the last execution of ReadXMLDocument method.

If the error was found not in an embedded entity then returned value will be False and Position will be empty. Otherwise returned value will be True and Position will contains as many elements as many embedded levels were at the moment when the error was found. Field Entity of each array element will contain the following information:

- %name; for a parameter internal entity where name is a name of the entity
- &name; for a parsed internal entity where name is a name of the entity
- an absolute URL for an external entity

Example

See example XMLDocument

XMLReplaceWhites

Parameters

S As String

Description

Replaces all white characters (<SP>, <HT>, <LF> and <CR>) in S with blanks (<SP>). This action is one of the white space normalization operations described in the XML standard.

Example

```
Dim A As Application
Dim S As String

A.XMLReplaceWhites S
```

XMLCollapseWhites

Parameters

S As String

Description

Leading and trailing white characters (<SP>, <HT>, <LF> and <CR>) in S are eliminated and consecutive whites are replaced by blanks (<SP>). This action is one of the white space normalization operations described in the XML standard.

Example

```
Dim A As Application,
Dim S As String

A.XMLCollapseWhites S
```

XMLWhiteSpaceNormalize

Parameters

S As String,

```
WhiteSpace As XMLWhiteSpaceFacet
```

Description

String *S* is normalized accordingly to the XML standard and correspondingly to *WhiteSpace* value:

- *xmlPreserve*—nothing is done.
- *xmlReplace*—as *XMLReplaceWhites* method does.
- *xmlCollapse*—as *XMLCollapseWhites* method does.

Example

```
Dim A As Application
Dim S As String,
Dim Element As XElement

A.XMLWhiteSpaceNormalize S, Element.WhiteSpace
```

XMLStringToBoolean

Parameters

Expression As String

Returns

Variant (Boolean)

Description

This method converts an *Expression* value to Boolean type accordingly to the XML standard. If input value is wrong, *Error* is returned. Note that unlike standard VBA function *CBool* white space characters being both at the beginning or at the end of an input strings are ignored.

Example

```
Dim A As Application
Dim S As String, B As Boolean

B = A.XMLStringToBoolean (S)
```

XMLStringToInteger

Parameters

Expression As String

Returns

Variant (Long)

Description

This method converts an *Expression* value to Long type accordingly to the XML standard. If input value is wrong, *Error* is returned. Note that unlike standard VBA function *CLong* white space characters being both at the beginning or at the end of an input strings are ignored.

Example

```
Dim A As Application
Dim S As String, L As Long

L = A.XMLStringToInteger (S)
```

XMLDecimalStringToReal

Parameters

Expression As String

Returns

Variant (Double)

Description

This method converts an *Expression* value to Double type accordingly to the XML standard assuming XML standard type decimal for an input value (integer and optional fraction parts separated by a dot and possible white characters at the beginning and the end). If input value is wrong, *Error* is returned.

Example

```
Dim A As Application
Dim S As String, D As Double

D = A.XMLDecimalStringToReal (S)
```

XMLStringToReal

Parameters

Expression As String

Returns

Variant (Double)

Description

This method converts an Expression value to Double type accordingly to the XML standard assuming XML standard type double for an input value (scientific notation and special values accordingly to IEEE-764 standard plus possible whites at the beginning and the end). If input value is wrong, Error is returned.

Example

```
Dim A As Application
Dim S As String, D As Double

D = A.XMLDecimalStringToReal (S)
```

XMLStringToDateTime

Parameters

Expression As String

Optional Zone As Variant (Integer)

Returns

Variant (Date)

Description

These method converts an Expression value to Date type accordingly to the XML standard. If input value is wrong, Error is returned.

Returned value is local date/time as it is stored in the input string. Zone obtains value of a time zone expressed as number of minutes which the local time differs by from the UTC (positive for the East hemisphere). If zone part is not present in the input string, Zone obtains value Empty.

Example

```
Dim A As Application
Dim S As String, T As Variant, Z As Variant

T = A.XMLStringToDateTime (S,Z)
If IsError (T) Then
    MsgBox "Time is incorrect"
Else
    MsgBox "Time Is " & Hour(T) & ":" & Minute(T)
    If IsEmpty (Zone) Then
        MsgBox "Time zone is not defined"
    Else
        MsgBox "Time difference from UTC is " & Zone & " minutes"
    End If
End If
```

XMLStringToTime

Parameters

Expression As String

Optional Zone As Variant (Integer)

Returns

Variant (Date)

Description

These methodsconvert an Expression value to Date type accordingly to the XML standard. If input value is wrong, Error is returned.

Returned value is local time as it is stored in the input string. Zone obtains value of a time zone expressed as number of minutes which the local time differs by from the UTC (positive for the East hemisphere). If zone part is not present in the input string, Zone obtains value Empty.

Example

See XMLStringToDateTime

XMLStringToDate

Parameters

Expression As String
Optional Zone As Variant (Integer)

Returns

Variant (Date)

Description

These methods convert an Expression value to Date type accordingly to the XML standard. If input value is wrong, Error is returned.

Returned value is local date as it is stored in the input string. Zone obtains value of a time zone expressed as number of minutes which the local time differs by from the UTC (positive for the East hemisphere). If zone part is not present in the input string, Zone obtains value Empty.

Example

See XMLStringToDateTime

XMLStringToUTCDateTime

Parameters

Expression As String

Returns

Variant (Date)

Description

These methods convert an Expression value to Date type accordingly to the XML standard. Returned value is UTC date/time. If input value is wrong or does not contain a zone part Error is returned.

Example

```
Dim A As Application
Dim S As String, T As Variant

T = A.XMLStringToUTCDateTime (S)
If IsError (T) Then
    MsgBox "UTC time is unknown"
Else
    MsgBox "UTC time Is " & Hour(T) & ":" & Minute(T)
End If
```

XMLStringToUTCTime

Parameters

Expression As String

Returns

Variant (Date)

Description

These methods convert an Expression value to Date type accordingly to the XML standard. Returned value is UTC time. If input value is wrong or does not contain a zone part Error is returned.

Example

See XMLStringToUTCDateTime

XMLDivideString

Parameters

Expression As String, Values() As String

Description

Expression value is divided to substrings treating white characters (in sense of XML standard) as separators and result is stored to Values. The operation is a routine operation for XML array data to obtain elements of arrays. Values may become empty.

Example

```
Dim A As Application
Dim Element As XElement, Values() As String

A.XMLDivideString Element.Value, Values
```

```
MsgBox UBound(Values)+1 & " values are present"

' Note :
'      this example will not work properly when there are 0 values
```

XMLStringToBooleans

Parameters

Expression As String, Values() As Boolean

Returns

Boolean

Description

This method first of all divides Expression content to several strings as XMLDivideString method does. Then each obtained string is converted to Boolean type as XMLStringToBoolean method does. If any of conversions is impossible, False is returned and Values becomes empty.

Example

```
Dim A As Application
Dim Element As XElement, Values() As Boolean, OK As Boolean

OK = A.XMLStringToBooleans (Element.Value, Values)
```

XMLStringToIntegers

Parameters

Expression As String, Values() As Long

Returns

Boolean

Description

This method first of all divides Expression content to several strings as XMLDivideString method does; then each obtained string is converted to Longtype as XMLStringToInteger method does. If any of conversions is impossible, False is returned and Values becomes empty.

Example

```
Dim A As Application
Dim Element As XElement, Values() As Long, OK As Boolean

OK = A.XMLStringToIntegers (Element.Value, Values)
```

XMLDecimalStringToReals

Parameters

Expression As String, Values() As Double

Returns

Boolean

Description

Each of these methodsfirst of all divides Expression content to several strings as XMLDivideString method does. Then each obtained string is converted to Realtype as corresponding method XMLDecimalStringToRealdoes. If any of conversions is impossible, False is returned and Values becomes empty.

Example

```
Dim A As Application
Dim Element As XElement, Values() As Double, OK As Boolean

OK = A.XMLDecimalStringToReals (Element.Value, Values)
```

XMLStringToReals

Parameters

Expression As String, Values() As Double

Returns

Boolean

Description

This method first of all divides Expression content to several strings as XMLDivideString method does. Then each obtained string is converted to Real type as corresponding method XMLStringToReal does. If any of conversions is impossible, False is returned and Values becomes empty.

Example

```
Dim A As Application
Dim Element As XElement, Values() As Double, OK As Boolean

OK = A.XMLStringToReals (Element.Value, Values)
```

XMLStringToDateTimesZ

Parameters

Expression As String, Values() As Date,
Zones() As Variant (Integer)

Returns

Boolean

Description

Each of these methods first of all divides Expression content to several strings as XMLDivideString method does. Then each obtained string is converted to a pair of values of Date and Integer types as corresponding method XMLStringToDateTime does. If any of the conversions is impossible, False is returned and both Values and Zones become empty.

Example

```
Dim A As Application
Dim Element As XElement, Values() As Date, Zones() As Variant
Dim OK As Boolean

OK = A.XMLStringToDateTimesZ (Element.Value, Values, Zones)
```

XMLStringToTimesZ

Parameters

Expression As String, Values() As Date,
Zones() As Variant (Integer)

Returns

Boolean

Description

Each of these methods first of all divides Expression content to several strings as XMLDivideString method does. Then each obtained string is converted to a pair of values of Date and Integer types as corresponding method XMLStringToTime does. If any of the conversions is impossible, False is returned and both Values and Zones become empty.

Example

See XMLStringToDateTimesZ

XMLStringToDatesZ

Parameters

Expression As String, Values() As Date,
Zones() As Variant (Integer)

Returns

Boolean

Description

This method first of all divides Expression content to several strings as XMLDivideString method does. Then each obtained string is converted to a pair of values of Date and Integer types as corresponding method XMLStringToDate does. If any of the conversions is impossible, False is returned and both Values and Zones become empty.

Example

See XMLStringToDateTimesZ

XMLStringToDateTimes

Parameters

Expression As String, Values() As Date

Returns

Boolean

Description

The same method as XMLStringToDateTimesZbut without time zones.

Example

```
Dim A As Application
Dim Element As XElement, Values() As DateTime, OK As Boolean

OK = A.XMLStringToDateTimes (Element.Value, Values)
```

XMLStringToDates

Parameters

Expression As String, Values() As Date

Returns

Boolean

Description

The same method as XMLStringToDatesZbut without time zones.

Example

See XMLStringToDateTimes

XMLStringToTimes

Parameters

Expression As String, Values() As Date

Returns

Boolean

Description

The same method as XMLStringToTimesZbut without time zones.

Example

See XMLStringToDateTimes

XMLStringToUTCDateTimes

Parameters

Expression As String, Values() As Date

Returns

Boolean

Description

The same methods as XMLStringToDateTimes but converted to UTC values. Note that omitted time zone in one or more values is treated as error.

Example

```
Dim A As Application
Dim Element As XElement, Values() As Date, OK As Boolean

OK = A.XMLStringToUTCDateTimes (Element.Value, Values)
```

XMLStringToUTCTimes

Parameters

Expression As String, Values() As Date

Returns

Boolean

Description

The same methods as `XMLStringToDates` but converted to UTC values. Note that omitted time zone in one or more values is treated as error.

Example

See `XMLStringToUTCDateTimes`

XMLBooleanToString

Parameters

Source As Boolean

Returns

String

Description

This method returns Source value converted to string.

Note. The method differs from standard VB function `CString` function in that that this method returns values in lower case.

Example

```
Dim A As Application
Dim Element As XMLElement, Value As Boolean

Element.AddText A.XMLBooleanToString (Value)
```

XMLIntegerToString

Parameters

Source As Long

Returns

String

Description

This method returns Source value converted to string.

Note. The method does exactly the same as standard VB function `CString` function does. It is included to Pythagoras VBA only for completeness reason.

Example

```
Dim A As Application
Dim Element As XMLElement, Value As Long

Element.AddText A.XMLIntegerToString (Value)
```

XMLRealToDecimalString

Parameters

Source As Double
Optional Fractional As Integer = 6

Returns

String

Description

This method returns Source value converted to string correspondingly to XML standard type decimal with specified number of fractional digits, i.e. an optional sign, an integer part, a dot and a fractional part.

Example

```
Dim A As Application
Dim Element As XMLElement, Value As Double

Element.AddText A.XMLRealToDecimalString (Value, 4)
```

XMLRealToString

Parameters

Source As Double, Optional Fractional As Integer = 6,
Optional Exponential As Boolean = True

Returns

String

Description

This method returns `Source` value converted to string correspondingly to XML standard type double with specified number of fractional digits. This method with parameter `Exponential = False` differs from the previous one only in that that special values such as infinity are converted to special strings. When `Exponential = True` resulting string will have an exponential part (power of ten).

Example

```
Dim A As Application
Dim Element As XElement, Value As Double

Element.AddText A.XMLRealToString (Value, 5)
```

XMLDateTimeToString

Parameters

`Source As Date`, `Optional Zone As Integer`,
`Optional Fractional As Integer = 0`

Returns

String

Description

This method returns `Source` value converted to string correspondingly to the XML standard. `Fractional` defines how many fractional digits of seconds to store. If `Zone` is present it defines a time zone expressed as difference with UTC time expressed as number of minutes (positive for the East hemisphere). If `Zone` is omitted then a zone part is not stored.

Example

```
Dim A As Application
Dim Element As XElement, Value As Date

Element.AddText A.XMLDateTimeToString (Value)
```

XMLTimeToString

Parameters

`Source As Date`, `Optional Zone As Integer`,
`Optional Fractional As Integer = 0`

Returns

String

Description

This method returns `Source` value converted to string correspondingly to the XML standard. `Fractional` defines how many fractional digits of seconds to store. If `Zone` is present it defines a time zone expressed as difference with UTC time expressed as number of minutes (positive for the East hemisphere). If `Zone` is omitted then a zone part is not stored.

Example

See `XMLDateTimeToString`.

XMLDateToString

Parameters

`Source As Date`
`Optional Zone As Integer`

Returns

String

Description

This method returns `Source` value converted to string correspondingly to XML standard. `Zone` is processed in the same way as for `XMLDateTimeToString` method.

Example

```
Dim A As Application
Dim Element As XElement, Value As Date

Element.AddText A.XMLDateToString (Value)
```

XMLUTCDateTimeToString

Parameters

Source As Date, Zone As Integer, Optional Fractional As Integer = 0

Returns

String

Description

This method differs from XMLDateTimeToString method only in that that input time is rather UTC than local.

Example

```
Dim A As Application
Dim Element As XElement, Value As Date

Element.AddText A.XMLUTCDateTimeToString (Value, 60)
' store as Central Europe time
```

XMLUTCTimeToString

Parameters

Source As Date, Zone As Integer, Optional Fractional As Integer = 0

Returns

String

Description

This methods differs from XMLTimeToString method only in that input time is rather UTC than local.

Example

See XMLUTCDateTimeToString

XMLUniteStrings

Parameters

Source() As String

Returns

String

Description

This method returns all strings united together by a space character. This is a normal way to make an array value for an XML file.

Example

```
Dim A As Application
Dim Element As XElement, Values() As String

Element.AddText A.XMLUniteStrings (Values)
```

XMLBooleansToString

Parameters

Source() As Boolean

Returns

String

Description

This method converts all values in Source to strings as method XMLBooleanToString does. Then it returns a united value as XMLUniteStrings method does.

Example

```
Dim A As Application
Dim Element As XElement, Values() As Boolean

Element.AddText A.XMLBooleansToString (Values)
```

XMLIntegersToString

Parameters

Source() As Long

Returns
String

Description

This method converts all values in Source to strings as method XMLIntegerToString does. Then it returns a united value as XMLUniteStrings method does.

Example

```
Dim A As Application
Dim Element As XElement, Values() As Long

Element.AddText A.XMLIntegersToString (Values)
```

XMLRealsToDecimalString

Parameters

Source() As Double, Optional Fractional As Integer = 6

Returns
String

Description

This method converts all values in Source to strings as method XMLRealToDecimalString does. Then it returns a united value as XMLUniteStrings method does.

Example

```
Dim A As Application
Dim Element As XElement, Values() As Double

Element.AddText A.XMLRealsToDecimalString (Values, 0)
```

XMLRealsToString

Parameters

Source() As Double, Optional Fractional As Integer = 6,
Optional Exponential As Boolean = True

Returns
String

Description

This method converts all values in Source to strings as method XMLRealToString does. Then it returns a united value as XMLUniteStrings method does.

Example

```
Dim A As Application
Dim Element As XElement, Values() As Double

Element.AddText A.XMLRealsToString (Values)
```

XMLDateTimesToString and XMLTimesToString

Parameters

Source() As Date, Optional Zone As Integer,
Optional Fractional As Integer = 0

Returns
String

Description

This method converts all values in Source to strings as XMLDateTimeToString method does. Then it returns a united value as XMLUniteStrings method does.

Example

```
Dim A As Application
Dim Element As XElement, Values() As Date

Element.AddText A.XMLDateTimesToString (Values, Zone:=120)
```


XMLTimesToString

Parameters

Source() As Date, Optional Zone As Integer,
Optional Fractional As Integer = 0

Returns

String

Description

SeeXMLDateTimesToString

XMLDatesToString

Parameters

Source() As Date, Optional Zone As Integer

Returns

String

Description

This method converts all values in Source to strings as XMLDateToString method does. Then it returns a united value as XMLUniteStrings method does.

Example

```
Dim A As Application
Dim Element As XElement, Values() As Date

Element.AddText A.XMLDatesToString (Values)
```

XMLUTCDateTimesToString

Parameters

Source() As Date, Zone As Integer,
Optional Fractional As Integer = 0

Returns

String

Description

This methods converts all values in Source to strings asXMLUTCDateTimeToStringmethod does. Then it returns a united value as XMLUniteStrings method does.

Example

```
Dim A As Application
Dim Element As XElement, Values() As Date

Element.AddText A.XMLUTCDateTimesToString(Values, -480)
```

XMLUTCTimesToString

Parameters

Source() As Date, Zone As Integer,
Optional Fractional As Integer = 0

Returns

String

Description

This methods converts all values in Source to strings asXMLUTCTimeToString method does. Then it returns a united value as XMLUniteStrings method does.

Example

See XMLUTCDateTimesToString

XMLValidName

Parameters

Name As String

Returns

Boolean

Description

Returns True if Name is correct.

Example

```
Dim A As Application, Name As String

While Not A.XMLValidName(Name)
    ' Ask operator for other name
Wend
```

Chapter 3 - PrivateTool

Description

A PrivateTool object is created by the PrivateTools method AddPrivateTool.

Note: the default private tool (see PrivateTools) does always exist.

Example

```
Dim prtTools As PrivateTools
Dim ptTool As PrivateTool
App.GetPrivateTools prtTools
prtTools.AddPrivateTool ptTool, "Info.ico"
ptTool.MenuString = "Show Data"
ptTool.OnMouseDoubleClick = "ShowData"
ptTool.OnMouseDown = "HideData"
ptTool.MenuEnabled = True

ptTool.Snapping = ptLine

ptTool.TargetEnabled = True
ptTool.TargetType = ptGPSPositionFound
```

Properties

Active
Default
Tooltip
MenuEnabled
MenuString
Snapping
KeepSelection
TargetEnabled
TargetType
TargetPosition
TargetLineStyle
TargetLineWidth
TargetLineColor
TargetLine
Cursor
OnPrivateTool
OnMouseMove
OnMouseDown
OnMouseUp
OnMouseDoubleClick
OnRightMouseDown
OnRightMouseUp
OnRightMouseDoubleClick
OnMiddleMouseDown
OnMiddleMouseUp
OnMiddleMouseDoubleClick

Methods

RemoveTargetPoint
RemoveTargetLine

Properties

Active

Returns

Boolean

Description

This property returns True if the private tool is active.

Only one private tool can be active at any time.

Activating and deactivating a private tool can only be done by the user by clicking on the VBA private tools and/or Pythagoras tools.

Example

```
Dim PT As PrivTool

...

If PT.Active Then

...

End If
```

Default

Returns

Boolean

Description

This property returns True for the private tool that is managed by the Application properties PrivateToolMenuEnabled, PrivateToolMenuString etc.

Example

```
Dim A As Application
Dim PrivTools As PrivateTools
Dim PT As PrivTool

...

If PT.Default Then

...

End If
```

Tooltip

Returns

String

Description

This property sets or returns the tooltip text for the private tool's toolbar button.

Example

```
Dim A As Application
Dim PrivTools As PrivateTools
Dim PT As PrivTool

...

PT.Tooltip = "This is a tooltip"
```

MenuEnabled

Parameters

Boolean

Description

Sets the status of the Private Tool menu item: True – for enabled, False – for disabled.

The disabled private tools will remain visible but disabled in the toolbar and menu. Only enabled private tools can be made active (selected).

Example

```
Dim A As Application
Dim PTS As PrivateTools
Dim PT As PrivateTool

Set A = New Application
A.GetPrivateTools PTS
PTS.GetFirst PT
PT.MenuEnabled = True
```

MenuString

Parameters

String

Description

Sets the name of the Private Tool menu item.

The maximum length of a menu name is 32 characters.

Example

```
Dim A As Application
Dim PTS As PrivateTools
Dim PT As PrivateTool

Set A = New Application
A.GetPrivateTools PTS
PTS.GetFirst PT
PT.MenuString = "My Private Tool"
```

Snapping

Parameters

ObjectTypes As Long

Description

Sets the types of objects that Pythagoras will snap to **when the Private Tool has been selected** before calling the OnMouseMove Event Macro. The value `ObjectTypes` specifies the types of objects (points, lines, ...) the cursor will snap to in the Private Tool Mode. So if `Snapping` returns `ptPoint`, the cursor will only snap to points. `ObjectTypes` is defined as the sum of one or more constants of the type *ObjectTypes* defined in the section "Constants" of this document. If `Snapping` returns `ptAll`, the cursor will snap to all object types.

Even if there is no OnMouseMove Event Macro, the cursor will snap only to the objects set by this property.

When `ptLine` and `ptPoint` is in `ObjectTypes`, cursor will also snap to intersection points of lines.

Example

```
Dim A As Application
Dim PTS As PrivateTools
Dim PT As PrivateTool

Set A = New Application
A.GetPrivateTools PTS
PTS.GetFirst PT
PT.Snapping = ptPoint+ptLine
```

KeepSelection

Parameters

Boolean

Description

Defines if the selection is cleared when the private tool is activated. The default value of this property `FALSE` (selection is removed automatically).

Example

```
Dim A As Application
Dim PTS As PrivateTools
Dim PT As PrivateTool

Set A = New Application
A.GetPrivateTools PTS
PTS.GetFirst PT

PT.KeepSelection = TRUE 'keeps the selected objects
```

TargetEnabled

Parameters

Boolean

Description

Sets the status of the targets. Two targets exist: a target point and a target line.

A target point can be created by the method `TargetPosition`. A target point will show a marker on screen at the given position if `TargetEnabled` is `TRLLineUE`.

A target line can be created by the method `TargetLine`. A marker line will be shown between the specified positions if `TargetEnabled` is `TRUE`.

Example

```
Dim A As Application
Dim PTS As PrivateTools
Dim PT As PrivateTool

Set A = New Application
A.GetPrivateTools PTS
PTS.GetFirst PT

PT.TargetEnabled = TRUE
```

TargetType

Parameters

Integer

Description

Sets the type of the target point (marker) when the private tool is enabled. The value can be either `ptGPSPositionFound` or `ptGPSSearchingPosition`.

Example

```
Dim A As Application
Dim PTS As PrivateTools
Dim PT As PrivateTool

Set A = New Application
A.GetPrivateTools PTS
PTS.GetFirst PT

PT.TargetEnabled TRUE
PT.TargetType = ptGPSSearchingPosition
```

TargetPosition

Parameters

XYZ

Description

Sets the position of the target point. The position is in local coordinates. The target is a marker on the screen, and is used to highlight a position. This could be for example the position received from GPS equipment. The display of the target has only effect if the private tool is selected.

Example

```
Dim A As Application
Dim PTS As PrivateTools
Dim PT As PrivateTool
Dim Pos As XYZ

Set A = New Application
A.GetPrivateTools PTS
PTS.GetFirst PT
PTS.SetPrivateToolActive PT

PT.TargetPosition = Pos
```

TargetLineStyle

Parameters

Style

Description

Sets the linetype of the target line (marker line) when the private tool is enabled. The value can be either `ptSolid`, `ptDash`, `ptDot`, `ptDashDot`, `ptDashDotDot` or `ptHiddenLine`.

Example

```
Dim A As Application
Dim PTS As PrivateTools
Dim PT As PrivateTool
Dim P1 As XYZ, P2 As XYZ
Dim Col As RGB

Col.Red = 255
```

```

Col.Blue = 0
Col.Green = 0

Set A = New Application
A.GetPrivateTools PTS
PTS.GetFirst PT

PT.TargetLineStyle ptSolid
PT.TargetLineWidth = 10
PT.TargetLineColor = Col
PT.TargetLine P1, P2

```

TargetLineWidth

Parameters

Integer

Description

Sets the width of the target line (marker line). The line width must be between 0.1 (0.01mm) and 25.5 (2.55 mm). To be used in combination with TargetLineStyle = ptSolid.

Example

See TargetLineStyle

TargetLineColor

Parameters

RGB

Description

Sets the color (RGB) of the target line (marker line).

Example

See TargetLineStyle

Cursor

Parameters

variant

Description

Sets the type of the cursor when the private tool is active. The cursor type remains active until either the macro library is terminated or when the private tool Cursor is changed.

When the value is an integer, it can be any of the constants specified in section Chapter 140 - Constants, item *Cursors*.

You can use custom cursor for your private tool by specifying the .cur file with your designed cursor

Example

```

Dim PT As PrivateTool

If PT.Active Then
    PT.Cursor = ptCursorP1
End If

If PT.Active Then
    PT.Cursor = "C:\Mycursors\Example.cur"
End If

```

OnPrivateTool, OnMouseMove, OnMouseDown, OnMouseUp, OnMouseDoubleClick, OnRightMouseDown, OnRightMouseUp, OnRightMouseDoubleClick, OnMiddleMouseDown, OnMiddleMouseUp, OnMiddleMouseDoubleClick

Parameters

EventSubName As String

Description

Sets the corresponding event macro (a VBA Sub) **for a non-default-private tool**. The event macro with the specified name must be defined in the macro library. This macro will be called each time when the

event occurs for the private tool being in selected state. In order to disable the event processing the string "" can be specified.

See also Chapter 1, section Events for more information about the type of events and how events are handled by Pythagoras and passed to the event macro.

Example

```
Dim A As Application
Dim PrivTools As PrivateTools
Dim PT As PrivTool

Set A = New Application
A.GetPrivateTools PrivTools
PrivTools.GetFirst PT
PrivTools.GetNext PT

PT.OnPrivateTool = "MyOnPrivateTool"
```

Methods

TargetLine

Parameters

P1 As XYZ, P2 As XYZ

Description

Sets the target line start end points.

Example

See TargetLineStyle

RemoveTargetPoint

Description

Removes the target point.

Example

```
Dim A As Application
Dim PrivTools As PrivateTools
Dim PT As PrivTool

Set A = New Application
A.GetPrivateTools PrivTools
PrivTools.GetFirst PT
PT.RemoveTargetPoint
```

RemoveTargetLine

Description

Removes the target line.

Example

```
Dim A As Application
Dim PrivTools As PrivateTools
Dim PT As PrivTool

Set A = New Application
A.GetPrivateTools PrivTools
PrivTools.GetFirst PT
PT.RemoveTargetLine
```


Chapter 4 - PrivateTools

Description

The PrivateTools object allows creating and manipulating several Private Tool objects. It represents a set of private tools in Pythagoras. The PrivateTools object is created by the Application method GetPrivateTools.

Pythagoras supports up to 31 private tools.

Note : the PrivateTools object contains always the private tool whose properties are accessible through the Application properties PrivateToolMenuEnabled, PrivateToolMenuString etc.

Example

```
Dim A As Application
Dim PrivTools As PrivateTools
Dim PT As PrivTool

Set A = New Application
A.GetPrivateTools PrivTools
PrivTools.GetFirst PT
If PT.Default Then      ' The set of private tools contains always
                        ' (as first) the "Default" private tool.
    MsgBox "This is the default private tool"
End If
```

Properties

None

Methods

```
GetActivePrivateTool
SetPrivateToolActive
GetFirst
GetNext
AddPrivateTool
RemovePrivateTool
```

Methods

GetFirst

Parameters

PrTool As PrivateTool

Description

Returns the first private tool. The first private tool is always the default private tool (accessible through Application).

Example

See introduction.

GetNext

Parameters

PrTool As PrivateTool

Description

Returns the next private tool. If there is no next private tool, Nothing is returned.

Example

```
Dim A As Application
Dim PTS As PrivateTools
Dim PT As PrivateTool

Set A = New Application
A.GetPrivateTools PTS
PTS.GetFirst PT
PTS.GetNext PT
```

GetActivePrivateTool

Parameters

PrTool As PrivateTool

Description

Returns the active (selected) private tool if any. If no private tool is active Nothing is returned.

Example

```
Dim A As Application
Dim PTS As PrivateTools
Dim PT As PrivateTool

Set A = New Application
A.GetPrivateTools PTS
...
PTS.GetActivePrivateTool PT
```

SetPrivateToolActive

Parameters

PrTool As PrivateTool

Description

Sets the argument private tool to be active (selected). The previous active private tool, if any, will be deactivated (deselected)

Example

```
Dim A As Application
Dim PTS As PrivateTools
Dim PT As PrivateTool

Set A = New Application
A.GetPrivateTools PTS
...
PTS.SetActivePrivateTool PT
```

AddPrivateTool

Parameters

PrTool As PrivateTool, Optional Icon As Variant := ptDefaultIcon

Description

Adds a new private tool. The new private tool will be added to the menu *Tools* and an icon will be added in the *Macro Tools* toolbar. By default, the new private tool will be enabled.

The parameter *PrTool* is the new created *PrivateTool* object.

The optional parameter *Icon* defines the icon that is displayed in the toolbar.

Normally the value will be either:

(1) the full path name of a custom icon file (extension .ICO) e.g.

"C:\MyIcons\MyCustomIcon.ico" or

(2) the name of an icon in the folder <PythagorasDirectory>\Macros\.

Alternatively the parameter *Icon* may be one of the following values:

ptIconDefault, ptIconRed, ptIconOrange, ptIconYellow, ptIconGreen,
ptIconBlue, ptIconIndigo, ptIconViolet,

These icons represent the default icon (see below) and icons with the colors: red, orange, yellow, green, blue, indigo and violet.

When the parameter is not given, the icons will have a sequential number (1, 2, ... A, B, ...).

Example

```
Dim A As Application
Dim PTS As PrivateTools
Dim PT As PrivateTool

Set A = New Application
A.GetPrivateTools PTS
...
PTS.AddPrivateTool PT, "C:\MyIcons\MyCustomIcon2.ico"
PT.MenuString = "My Tool 2"
```

```

...
PTS.AddPrivateTool PT, "MyCustomIcon1.ico"
PT.MenuString = "My Tool 1"
...
PTS.AddPrivateTool PT, ptIconBlue
PT.MenuString = "My Tool 3"

```

RemovePrivateTool

Parameters

PrTool As PrivateTool

Description

Removes the private tool. The default (standard) private tool cannot be removed.

Example

```

Dim A As Application
Dim PTS As PrivateTools
Dim PT As PrivateTool

Set A = New Application
A.GetPrivateTools PTS
...
PTS.RemovePrivateTool PT

```

Chapter 5 - Document

Description

A Document object is created by the Application methods `OpenDocument` and `GetNewDocument`. A reference to an existing Pythagoras document can be obtained with the method `GetActiveDocument`.

Example

```
Dim A As Application
Dim Doc1 As Document, Doc2 As Document

Set A = New Application
A.GetNewDocument Doc1 ' Untitled document becomes the top window
A.OpenDocument "C:\Test.pyt", False, Doc2 ' Test.pyt will be opened
```

Properties

`ReadOnly`
`NewGroupId`
`Name`
`SnappedObjectPosition`
`EventCursorPosInDoc`
`EventCursorPosInWindow`
`Info`
`Extent`
`FolderExternalFile`
`ActiveOverlay`
`Scale`
`Database`
`LockedDatabase`
`DifferentDatabaseLock`
`NrOfTables`
`NrOfAttributeTables`
`NrOfDictionaryTables`
`NrOfViews`
`NextPointIdent`

Methods

`Save`
`SaveAs`
`PrintToPDF`
`CloseDoc`
`NewState`
`EndState`
`ClearUndo`
`UpdateScreen`
`RedrawDocument`
`GetOverlay`
`GetSelection`
`GetFlaggedObjects`
`GetObjectUnderCursor`
`ObjectReferenceToCadObject`
`GetKeyboardEventData`
`Paste`
`ImportShape`
`ImportShapeRegion`
`ImportDXF - ImportDWG`
`ImportDataCollector`
`GetActiveCoordinateSystem`
`TransformXYTo`
`TransformXYFrom`
`TransformPossible`
`Center`
`Zoom`
`ZoomRectangle`
`FitInWindow`
`ShowAll`
`GetSheets`
`GetLayers`
`GetLayerGroups`
`GetSymbolLibrary`

```

GetWindow
GetTerrainModels
GetSubDocuments
FirstTable
NextTable
TableByName
CreateTable
ImportDBF
DropTable
CopyTableSkeleton
CopyTable
FirstDBView
NextDBView
DBViewByName
CreateSimpleQuery
CreateAggregationQuery
DropQuery
CopyQuery
CopyDatabaseSkeleton
CopyDatabase
PurgeDatabase
DropDatabase
LockDatabase
UnLockDatabase
ActualizeQueries
ShowDataViewer
GetThematicMaps
GetViews
PointIdentUsed
Vectorise
GetCameras

```

Properties

ReadOnly

Returns

Boolean

Description

This property returns true if the document is read-only. The user can have only read-only access due to settings in the file system, or because already another user has opened the document for read-write.

Example

```

Dim A As Application
Dim Doc As Document

Set A = New Application
A.GetActiveDocument Doc
If Doc.ReadOnly then
    \...
End If

```

NewGroupId

Returns

Long

Description

This property returns a free Group identifier. Maximum 65535 group identifiers (0 .. 65534) can be used in a document. If no group identifier free, -1 is returned.

If a document contains subdocument, this property returns a new group identifier in the main document. Existing group identifiers in subdocuments will not be taken into account. As a consequence objects in different subdocuments can have the same Group identifier although they will not belong to the same group.

This property is obsolete should be replaced by Overlay method CreateCompound.

Example

```

Dim A As Application
Dim Doc As Document

```

```
Dim Group As Long

Set A = New Application
A.GetNewDocument Doc
Group = Doc.NewGroupId
```

Name

Returns
String

Description

This property returns the filename of the document. E.g. "C:\Test.pyt". If the document contains subdocuments, the property will always return the name of the main document.

Example

```
Dim FileName As String
FileName = Doc.Name
```

SnappedObjectPosition

Returns
XYZ

Description

This property returns the XYZ coordinates of the *adjusted* position of the cursor in the active document. Adjusted in this context means that not the real coordinates of the cursor are returned, but the coordinates of the object the cursor has snapped to. These coordinates will always be identical to the coordinates shown in the control panel.

This property may only be used when processing a mouse event macro. E.g. OnMouseMove.

Example

```
Dim A As Application
Dim D As Document
Dim Obj As CadObject
Dim L As Line
Dim Pos As XYZ

Set A = New Application
A.GetActiveDocument D
D.GetObjectUnderCursor ptLine, Obj
If not (Obj is nothing) Then
    Obj.GetLine L
    Pos = D.SnappedObjectPosition ' returns point on the line near cursor
End if
```

EventCursorPosInDoc

Returns
XY

Description

This property returns the XY coordinates of the position of the cursor in the active document. This property may only be used when processing a mouse event macro. E.g. OnMouseMove.

Example

```
Dim A As Application
Dim Doc As Document
Dim P As XY
Dim Obj As CadObject
Dim T As Text

Set A = New Application
A.GetActiveDocument Doc
Doc.GetObjectUnderCursor ptText, Obj
If not (Obj is nothing) Then
    Obj.GetText T
    P = Doc.EventCursorPosInDoc
    T.Position = P
End If
```

EventCursorPosInWindow

Returns

WindowCoord

Description

This property returns the coordinates of the position of the cursor in the window. This property may only be used when processing a mouse event macro. E.g. OnMouseMove.

Example

```
Sub OnMouseUp

Dim A As Application
Dim D As Document

Set A = New Application
A.GetActiveDocument D
MsgBox "X : " & D.EventCursorPosInWindow.X & _
      ", Y : " & D.EventCursorPosInWindow.Y

End Sub
```

Info

Returns

DocInfo

Description

This property sets or returns the drawing information.

Example

```
Dim A As Application
Dim D As Document
Dim Txt As DocInfo

Set A = New Application
A.GetActiveDocument D
If Not (D is Nothing) then
    Txt.Info1 = "First Name"
    Txt.Info2 = "Last Name"
    Txt.Info3 = "ZIP code"
    Txt.Info4 = "City"
    Txt.Info5 = "Country"
    D.Info = Txt
end if
```

Extent

Returns

Rectangle

Description

This property returns the extent (the enclosing rectangle) of the document in local coordinates. The extent includes the main document and all non-sleeping subdocuments.

Example

```
Dim A As Application
Dim D As Document
Dim R As Rectangle

Set A = New Application
A.GetActiveDocument D
If Not (D is Nothing) then
    R = D.Extent
end if
```

ActiveOverlay

Returns

Overlay As Integer

Description

ActiveOverlay returns or sets the active overlay. The value is either ptPaper or ptDrawing.

Example

```
Dim A As Application
Dim D As Document

set A = New Application
A.GetActiveDocument D
If Not (D is Nothing) then
    D.ActiveOverlay = ptPaper
end if
```

Scale

Returns

Scale As Double

Description

Scale returns the default scale of the active document.

Example

```
Dim A As Application
Dim D As Document

set A = New Application
A.GetActiveDocument D
If Not (D is Nothing) then
    MsgBox CSTR(D.Scale)
end if
```

ZoomFactor

Parameters

ZoomFactor As Double

Description

ZoomFactor : returns the actual zoomfactor. When ZoomFactor = 1.0, the document is displayed at the right scale.

Example

```
Sub PM_ZoomFactor

Dim A As Application
Dim D As Document

Set A = New Application
A.GetActiveDocument D
If NOT D is nothing Then
    MsgBox CStr(D.ZoomFactor)
end if
End sub
```

ActiveSheet

Returns

S As Sheet

Description

ActiveSheet sets or returns the active sheet.

The object Sheets provides methods GetActive and SetActive that have the same function.

Example

```
Dim A As Application
Dim D As Document
Dim S As Sheet

set A = New Application
A.GetActiveDocument D
If Not (D is Nothing) then
    Set S = D.ActiveSheet
End If
```

Database

Returns

Database

Description

This property returns the document database.

Example

```
Dim Doc As Document, DB As Database

Set DB = Document.Database
```

LockedDatabase, DifferentDatabaseLocks, NrOfTables, NrOfAttributeTables, NrOfDictionaryTables, NrOfViews

These properties are duplicates of properties in Database class, ex. `Document.NumberOfTables` call is the same as `Document.Database.NumberOfTables`.

NextPointIdent

Returns

String

Description

Returns the next point number according to the current document settings or sets (modifies) the next point number in the current document settings.

When the point number is set, the following rules apply:

- 1) The string is empty: point numbering will be switched off. (Points will not get a point number)
- 2) The string contains no numeric characters: point numbers will increment alphabetically.
E.g. A, B, C, ...
- 3) The string contains numeric characters: the last numeric part will be incremented.
E.g. A01, A02, A03, ...

Example

```
Dim D As Document
Dim pointNumber As String

pointNumber = D.NextPointIdent
\
D.NextPointIdent = "S01"    'Next point numbers will become S01, S02, S03, ...
```

Methods

Save

Parameters

None

Description

Saves the document if any changes were done. If the requested document was a new *Untitled* document, the user will be prompted to give the name of the document.

Example

```
Doc As Document

Doc.Save
```

SaveAs

Parameters

FileName As String, DocVersion As Integer, Optional AskAttributes As Boolean

Description

Saves the current document with the path and filename in `FileName`. If the parameter `FileName` is empty, then a dialog box will come up prompting the user to enter the filename.
`DocVersion` : ptV11 : file is saved in Pythagoras 11 format.

DocVersion : ptV12 : file is saved in Pythagoras 12 format.
 DocVersion : ptLastVersion(-1) : file is saved in the most recent format.
 DocVersion : ptDXF13 : file is saved in AutoCAD 13 DXF format.
 DocVersion : ptDXF14 : file is saved in AutoCAD 14 DXF format.
 DocVersion : ptDXF2000 : file is saved in AutoCAD 2000 DXF format.
 DocVersion : ptDXF2004 : file is saved in AutoCAD 2004 DXF format.
 DocVersion : ptDXF2007 : file is saved in AutoCAD 2007 DXF format.
 DocVersion : ptDXF2010 : file is saved in AutoCAD 2010 DXF format.
 DocVersion : ptDWG13 : file is saved in AutoCAD 13 DWG format.
 DocVersion : ptDWG14 : file is saved in AutoCAD 14 DWG format.
 DocVersion : ptDWG2000 : file is saved in AutoCAD 2000 DWG format.
 DocVersion : ptDWG2004 : file is saved in AutoCAD 2004 DWG format.
 DocVersion : ptDWG2007 : file is saved in AutoCAD 2007 DWG format.
 DocVersion : ptDWG2010 : file is saved in AutoCAD 2010 DWG format.
 DocVersion : ptGRBFile2D: file is saved in GRB DXF 2D format.
 DocVersion : ptGRBFile2_5D: file is saved in GRB DXF 2.5D format.
 DocVersion : ptGRBFile3D: file is saved in GRB DXF 3D format.

If the parameter `AskAttributes = False`, the dialog box that is normally displayed when a DXF or DWG format is chosen, is not shown and the default values, or the previously entered values in this dialog box are used.

If the document can not be saved in the requested format, the run-time error 1028 is raised.

Example

```

Doc As Document

Doc.SaveAs "C:\Test.pyt", ptLastVersion

DWG & SHAPE :
- units : control panel,
- prefix : preferences
- CRS : control panel.
```

PrintToPDF

Parameters

Filename, Optional OpenPDFFile As Boolean

Description

Prints all sheets of the document into a single pdf file with the specified `Filename`.

When `OpenPDFFile` is TRUE the file is opened automatically in the default PDF viewer.

Example

```

Set App = New Application
App.GetActiveDocument Doc

Doc.PrintToPDF("C:\Temp\pyth.pdf")
```

CloseDoc

Parameters

AddToRecentDocList As Boolean, SaveBeforeClose As Integer, FileName As String, DocVersion As Integer

Description

Closes the document. If `AddToRecentDocList = True` then the file name is added to the list of recently used documents in the "File" menu. The value of `SaveBeforeClose` is only significant if changes are made to the document and the document is not yet saved.

`SaveBeforeClose = ptDontSave` : Ignore changes

`SaveBeforeClose = ptAskUser` : Displays dialog box requesting if changes need to be saved.

`SaveBeforeClose = ptSaveAs` : Saves changes before closing document. `Name = FileName`.

If the document can not be saved in the requested format, the run-time error 1028 is raised.

`Filename` : Must contain the file name if `SaveBeforeClose = ptSaveAs`.

`DocVersion` : See `SaveAs` method.

The document that contains the active macro can not be closed. An error is raised.

Example

```
Dim Doc As Document
Dim Name As String

Doc.CloseDoc True, ptAskUser, Name, ptLastVersion
```

NewState

Parameters

UndoText As String, RedoText As String

Description

NewState starts a new state in Pythagoras so that all operations done on a document between NewState and EndState can be "Undone" by the user. UndoText and RedoText are the texts that will appear in the Undo and Redo Edit menus. When a VB program contains no NewState statement(s), Pythagoras will automatically add a new state for each document that is changed by the VB program. When the program in the following example is executed, the user will have the possibility to undo the creation of the 2nd point and subsequently undo the creation of the first point.

Example

```
Sub NewState
    Dim A As Application
    Dim Doc As Document
    Dim Ovl As Overlay
    Dim P As Point
    Dim Position As XYZ

    Set A= New Application
    A.GetActiveDocument Doc
    Doc.GetOverlay ptDrawing, Ovl
    Position.X=10
    Position.Y=10
    Position.Z=0
    Doc.NewState "Undo Point1", "Redo Point1"
    Ovl.CreatePoint Position, P          ' Create point in Drawing overlay
    Doc.EndState True
    Position.X=20
    Position.Y=20
    Doc.NewState "Undo Point2", "Redo Point2"
    Ovl.CreatePoint Position, P          ' Create a second point
    Doc.EndState True
End Sub
```

EndState

Parameters

KeepChanges As Boolean

Description

This method terminates the state initiated by NewState. Normally the parameter KeepChanges will be = True. If the parameter KeepChanges = False then all changes made on the document between NewState and EndState will be removed. This feature is useful to undo changes when an operation cannot be successfully terminated. EndState refreshes the screen to reflect all changes made between NewState and EndState.

Example

```
Dim A As Application
Dim Doc As Document

On Error GoTo ErrorHandler

' processing & creating objects.
' If an error occurs, the code after the label ErrorHandler will restore
' the document to the original state

Exit Sub

ErrorHandler :
```

```
Doc.EndState False      ' Restores the document to the previous state.
Doc.UpdateScreen
```

ClearUndo

Parameters

None

Description

This method clears all undo states in Pythagoras. The method may be called any time. Other statements that change the document may follow. However the screen will be updated immediately when ClearUndo is called.

Example

```
Dim Doc As Document

Doc.NewState "Undo Action 1", "Redo Action 1"

' Update some objects ...

Doc.EndState True
Doc.ClearUndo      ' User will be unable to undo all previous actions.

Doc.NewState "Undo Action 2", "Redo Action 2"

' Update some objects ...

Doc.EndState True  ' User will now be able to undo only the last action.
```

UpdateScreen

Parameters

None

Description

UpdateScreen will refresh the screen so that all changes made to a document after NewState will be reflected on the screen. UpdateScreen must not be done after calling EndState. EndState updates the screen. UpdateScreen has to be used to make partial changes visible to the user. If UpdateScreen is used to reflect the progress of the macro, e.g. when creating many objects, then displaying the updates will be much faster if NewState is called after each UpdateScreen. Of course, undo of all operations done by the macro will not be possible. The method UpdateScreen is generally faster compared with RedrawDocument (see below).

Example

```
Doc.UpdateScreen
```

RedrawDocument

Parameters

None

Description

RedrawDocument will redraw the complete document. Normally this function should not be required (See UpdateScreen).

Example

```
Doc.RedrawDocument
```

GetOverlay

Parameters

```
OverlayType As Integer, Ovl As Overlay,
```

Description

GetOverlay returns a reference to either the Paper or Drawing Overlay.
 OverlayType = ptPaper : Return Paper Overlay.
 OverlayType = ptDrawing : Return Drawing Overlay.

Example

```
Dim A As Application
Dim Doc As Document
```

```

Dim Position As XYZ
Dim NewPoint As Point
Dim Drawing As Overlay

Set A= New Application
A.GetActiveDocument Doc
Doc.GetOverlay ptDrawing, Drawing
Drawing.CreatePoint Position, NewPoint      ' Creates point in Drawing Overlay
NewPoint.Color = ptRed

```

GetSelection

Parameters

SelectedObjects As Selection

Description

GetSelection returns the set of selected objects. If no objects are selected, the set is empty. All objects in the set are either in the Drawing or in the Paper overlay.

Example

```

Dim SelectedObjects As Selection
Dim Doc As Document

Doc.GetSelection SelectedObjects

```

GetFlaggedObjects

Parameters

FlaggedObjs As FlaggedObjects

Description

GetFlaggedObjects returns the set of flagged objects. If no objects are flagged, the set is empty. All objects in the set are either in the Drawing or in the Paper overlay. Only VBA can set objects flagged. Objects are set flagged if an operation on multiple flagged objects has to be done at a later state in the program.

Example

```

Dim FlaggedObjs As FlaggedObjects
Dim Doc As Document

Doc.GetFlaggedObjects FlaggedObjs

```

GetObjectUnderCursor

Parameters

ObjectTypes As Long, Object As CADObject

Description

GetObjectUnderCursor returns the object that is near the cursor on the screen. (Snapped to the object). If no object is snapped, Object = nothing. The input parameter ObjectTypes specifies the types of objects (points, lines, ...) the cursor will snap to. So if ObjectTypes = ptPoint, the cursor will only snap to points, and only a CADObject of type Point can be returned. ObjectTypes is defined as the sum of one or more constants of the type *ObjectTypes* defined in the section "Constants" of this document. If ObjectTypes = ptAll, the cursor will snap to all object types besides ptCompound. Since new object types may be defined in future versions of Pythagoras, use the constant ptAll. When a new object type is added, ptAll will include the new object types.

Example

```

Public Sub DocumentOnMouseDoubleClick()

Dim A As Application
Dim D As Document
Dim Obj As CadObject

Set A = New Application
A.GetActiveDocument D
D.GetObjectUnderCursor ptPoint + ptLine, Obj
If Not (Obj is Nothing) Then
    Obj.Color = ptGreen
    A.StopEvent

```

End If

ObjectReferenceToCadObject

Parameters

ObjectReference As Long, Subdoc As Subdocument, Object As CADObject

Description

ObjectReferenceToCadObject returns the object in the the subdocument SubDoc with the given ObjectReference. If no object is found, an empty object is returned.

See also the CADObject property ObjectReference.

Example

```
Dim A As Application
Dim D As Document
Dim SubDocs As SubDocuments
Dim SubDoc As SubDocument
Dim ObjRef As Long
Dim Obj As CADOBJECT

Set A = New Application
A.GetActiveDocument D
D.GetSubDocuments SubDocs
SubDocs.GetMain SubDoc
D.ObjectReferenceToCadObject ObjRef, SubDoc, Obj
if Not (Obj is Nothing) then
    Obj.Color = ptGreen
end if
```

GetKeyboardEventData

Parameters

VirtualKeyCode As Long, Character As Variant

Description

Returns the Virtual Keycode and the character value of the keyboard key pressed.

This method may only be used in a keyboard event macro. If not an error will be raised.

VirtualKeyCode is a code assigned to a keyboard key and it does not depend on the current keyboard layout, see

[http://msdn.microsoft.com/en-us/library/windows/desktop/dd375731\(v=vs.85\).aspx](http://msdn.microsoft.com/en-us/library/windows/desktop/dd375731(v=vs.85).aspx)

Virtual keycode can be used to test on special keys (e.g. F1..F12).

Parameter Character normally should be a string variable, a Unicode character corresponding to the pressed key is returned, returned value depends on the current keyboard layout (QWERTY, AZERTY, Cyrillic etc.). For compatibility with old versions if this parameter is of Byte type, then ASCII code is returned.

Example

```
Sub OnKeyDown

    Dim A As Application, D As Document
    Dim VKey As Long, C As String

    Set A = New Application
    A.GetActiveDocument D
    D.GetKeyboardEventData VKey, C
    MsgBox C ' Character pressed is shown

End Sub
```

Paste

Parameters

CS As CoordinateSystem, Operations As Integer, Scale As Double

Description

This method will paste the content of the clipboard in the drawing. The coordinates on the clipboard are considered to be relative to the selected Coordinate System. The data on the clipboard is scaled with the given scale factor (parameter Scale). When the coordinate system is not initialized, (empty object), then the data is pasted relative to the active coordinate system or projection system.

Parameter Operations : is the sum of 1 or more of the following constants :

- ptPasteNormal: the method Paste behaves identically as the menu function Paste.
- ptMirrorX : mirror around X-axis of CS
- ptMirrorY : mirror around Y-axis of CS
- ptZToY : the Z-value becomes Y and Y-value becomes Z. (Tipping over data).
- ptClearPointNumber : During pasting, the point numbers of the points pasted in the drawing are cleared.
- ptAdjustSketch : If points with identical point numbers are found in the drawing, the coordinates of the drawing are retained, and the pasted data is adjusted to fit with the existing points.

Example

```
Sub DoPaste

Dim A As Application
Dim D As Document
Dim CS As CoordinateSystem
Dim CSEmpty As CoordinateSystem
Dim O As Overlay
Dim Pnt As XY
Dim Origin As XYZ

set A = New Application
A.GetActiveDocument D
If Not (D is Nothing) then
    D.Paste CSEmpty, ptPasteNormal, 1.0
    D.GetOverlay ptDrawing, O
    Origin.x = 1.0
    Origin.y = 1.0
    Origin.z = 2.0
    Pnt.x = 3.0
    Pnt.y = 3.0
    O.CreateTempCoordinateSystem Origin, Pnt, CS
    D.Paste CS, ptMirrorX + ptMirrorY, 1.0
end If
End sub
```

ImportShape

Parameters

Optional Name As String = "", Optional AskAttributes As Boolean = True

Description

This method will import one or multiple shape files in the drawing. When no parameters are given, the method behaves in the same way as if the Pythagoras menu item "File", "Import", "Shape" was selected. The parameter name contains none, 1 or multiple files. If this parameter is passed, the dialog box requesting the user to select shape files is not displayed. If the name contains one or more files, the files are imported. If the pathname is omitted, the files should be in the same directory as the previous shape import. If the parameter AskAttributes = False, the dialog box that is normally displayed when one or multiple shape files are imported is not shown and the default values, or the previously entered values in this dialog box are used.

Example

```
Sub DoImportShape

Dim A As Application
Dim D As Document
Dim O As Overlay

set A = New Application
A.GetActiveDocument D
If Not (D is Nothing) then
    D.ImportShape "C:\MyFiles\File1.shp*File2.shp*File3.shp"
    ' Imports 3 files. All files are in directory C:\MyFiles
    ' Dialog box requesting attributes (layer, styles, ...) will be displayed.

    D.ImportShape "File4.shp", False
    ' Imports C:\MyFiles\File4.shp. Same attributes as used above will be used.
end If
```


End sub

ImportShapeRegion

Parameters

Region = Rectangle, Optional Name As String = "",
Optional AskAttributes As Boolean = True

Description

The method is similar to the ImportShape method. The difference is that with this method only the Shape objects are imported that are partially or completely inside the specified Region Rectangle. Note that some extra Shape objects lying very close to the borders of Region could be imported too. The borders of Region can be specified as Application.NegativeInfinity or Application.PositiveInfinity (see example below).

Example

```
Dim A As Application, D As Document, R As Rectangle

Set A = New Application
A.GetActiveDocument D
If Not D Is Nothing Then
    R.x1 = 0.0
    R.x2 = A.PositiveInfinity
    R.y1 = A.NegativeInfinity
    R.y2 = A.PositiveInfinity
    D.ImportShapeRegion R, "File4.shp"
    ' Imports C:\MyFiles\File4.shp. Attributes will be asked.
    ' Only objects having x > 0 are imported.
End If
```

ImportDWG (ImportDXF)

Parameters

Optional Name As String = "", Optional Scale as Double = <Scale of Document>,
Optional Blocks as Long = ptAsSymbols, Optional Hatching as Long = ptAsSymbols,
Optional AskAttributes as Boolean = FALSE, Optional Log as String = "",
Optional Units as Integer = ptM, Optional Geopus As Boolean
Optional AskSHX As Boolean

Description

Imports a DXF/DWG file in the document.

Name : the full filename of the DXF/DWG file.

If Name = "", request file open dialog will be opened.

Scale : the scale used to import the DXF/DWG texts, symbols, lins styles and polygon hatching.

The default is the scale of the active drawing.

Blocks : defines how to import blocks. Values: ptAsPoints, ptAsSymbols or ptUnpacked.

Hatching : defines how to import polygon hatchings.

Values: ptNoImport, ptAsSymbols or ptUnpacked)

AskAttributes : Show a dialog allowing users to manually choose parameters.

Log : returns a detailed log of the import results.

Units: the units used to import the DXF/DWG. The default units is ptM. Values: ptMM, ptCM, ptM, ptKM, ptInch, ptFeet, ptMiles or ptKeepOriginalUnit.

Geopus : If this parameter is TRUE, all points & blocks with GEOPUS_PUNTEN metadata (xdata) will use the elevation, point id, code and setup point of the metadata. These last 2 values will show in that order in the comment field separated by "-". If the value is missing in the metadata, it will show as "/" instead.

Any other metadata besides GEOPUS_PUNTEN is ignored.

AskSHX : Indicates if SHX files should be asked or not.

Example

```
Dim A As Application
Dim Doc as Document
Dim MyLog as String

set A = New Application
A.GetActiveDocument Doc
If NOT(Doc is nothing) Then
```

```

Doc.ImportDWG Scale:=1/250.0, AskAttributes:=TRUE, Log:=MyLog
If Len(MyLog)>0 Then
    MsgBox MyLog
End If
End If

```

ImportDataCollector

Parameters

FileName As String, ShowErrors As Boolean

Description

ImportDataCollector will import the file FileName. This method simulates the command *Import, Data Collector*.

When ShowError = True, error messages are shown if there are any. For example, if the file with the given name does not exist the error "File with given name not found" is raised. Independent of ShowError if the import file contains errors, the editor will open.

The file FileName should have a structure that corresponds to the format required for the active (configured) data collector. When Geocoding is switched off, the attributes of the points will correspond with the actual default settings.

Example

```

Dim A As Application
Dim Doc As Document
Dim ImportFile As String

Set A = New Application
A.GetActiveDocument Doc
If Not Doc Is Nothing Then
    Doc.ImportDataCollector ImportFile, False
End If

```

GetActiveCoordinateSystem

Parameters

CS As CoordinateSystem

Description

Gets the active coordinate system. This method returns Nothing, if the active coordinate system = Profile.

Example

```

Dim A As Application
Dim Doc As Document
Dim CS As CoordinateSystem

Set A = New Application
A.GetActiveDocument Doc
If Not (Doc Is Nothing) Then
    Doc.GetActiveCoordinateSystem CS
    If CS Is Nothing Then
        Exit Sub
    End If
    If CS.Overlay = ptDrawing Then
        MsgBox "CS in Local = " & CS.Name
    Else
        MsgBox "CS in Page = " & CS.Name
    End If
End If

```

GetCoordinateReferenceSystem

Parameters

CRS As CoordinateReferenceSystem

Description

Gets the coordinate reference system of the document. This method returns Nothing if the coordinate reference system is not defined.

Example

```

Dim A As Application

```

```

Dim Doc As Document
Dim CRS As CoordinateReferenceSystem

Set A = New Application
A.GetActiveDocument Doc
If Not (Doc Is Nothing) Then
    Doc.GetCoordinateReferenceSystem CRS
    If CRS Is Nothing Then
        MsgBox "CRS = Unknown"
    Else
        MsgBox "CRS = " & CRS.Name
    End If
End If

```

SetCoordinateReferenceSystem

Parameters

CRS As CoordinateReferenceSystem

Description

This method will assign the coordinate reference system to the local CS of the document. Afterwards, if the document is on top, the control panel will hold the previous coordinate reference system if it is compatible with the new coordinate reference system.

If an empty parameter is passed, the previous coordinate reference system of the document is removed.

Example

```

Dim A As Application
Dim Doc As Document
Dim CRSs As CoordinateReferenceSystems
Dim CRS As CoordinateReferenceSystem

Set A = New Application
A.GetActiveDocument Doc
A.GetCoordinateReferenceSystems CRSs
CRSs.GetCoordinateReferenceSystem"Belgium/Lambert72", CRS

If Not (Doc Is Nothing) And Not (CRS Is Nothing) Then
    Doc.SetCoordinateReferenceSystem CRS
End If

```

TransformXYTo

Parameters

DestinationCRS As CoordinateReferenceSystem,
Pnt As XY

Returns

XY

Description

TransformXYTo transforms the coordinates Pnt from the CRS of the document to the DestinationCRS taking into account tie points and treatment of residuals that are defined in the document for transformation to the destination CRS.

For geographical systems, XY is used to return the latitude, longitude values. These angle values must be passed in degrees decimal. Sequence: x = latitude, y = longitude. For projection systems the (x,y) coordinates are passed/returned in meter.

Example

```

Dim A As Application
Dim D As Document
Dim DestinationCRS As CoordinateReferenceSystem
Dim Pnt As XY
Dim LatLong As XY

set A = New Application

A.GetActiveDocument D
CRSs.GetCoordinateReferenceSystem "WGS84", DestinationCRS
Pnt.x = 123456.256

```

```

Pnt.y = 258321.128
If D.TransformPossible(DestinationCRS) then
    LatLong = D.TransformXYTo(DestinationCRS, Pnt)
End if

```

TransformXYFrom

Parameters

```

SourceCRS As CoordinateReferenceSystem,
Pnt As XY
DocPnt As XY

```

Returns

```

XY

```

Description

TransformXYFrom transforms the coordinates Pnt from the DestinationCRS to the CRS of the document taking into account tie points and treatment of residuals that are defined in the document for transformation to the destination CRS.

For geographical systems, XY is used to return the latitude, longitude values. These angle values must be passed in degrees decimal. Sequence: x = latitude, y = longitude. For projection systems the (x,y) coordinates are passed/returned in meter.

Example

```

Dim A As Application
Dim D As Document
Dim SourceCRS As CoordinateReferenceSystem
Dim Pnt As XY
Dim DocPnt As XY

set A = New Application

A.GetActiveDocument D
CRSs.GetCoordinateReferenceSystem "WGS84", DestinationCRS
Pnt.x = 51.3146424 'Latitude
Pnt.y = 4.8776564 'Longitude
If D.TransformPossible(DestinationCRS) then
    DocPnt = D.TransformXYFrom(DestinationCRS, Pnt)
End if

```

TransformPossible

Parameters

```

CRS As CoordinateReferenceSystem

```

Returns

```

Boolean

```

Description

TransformPossible returns True if the transformation to/from CRS is possible.

Example

See TransformXYTo and TransformXYFrom

Center

Parameters

```

P As XY

```

Description

Redraws the document on the screen. Point P of the document will become the center of the window. The display scale will remain unchanged.

Example

```

Sub PM_Center

Dim A As Application
Dim D As Document
Dim p As XY

set A = New Application
A.GetActiveDocument D

```

```

If NOT D is nothing Then
    P.x = 20.0
    P.y = 10.0
    D.Center P
end If
End sub

```

Zoom

Parameters

P As XY, Scale As Double

Description

Redraws the document on the screen. Point P of the document will become the center of the window.
Scale : defines the scale at which the drawing will be displayed.

Example

```

Sub PM_Zoom

Dim A As Application
Dim D As Document
Dim p As XY

set A = New Application
A.GetActiveDocument D
If NOT D is nothing Then
    P.x = 20.0
    P.y = 10.0
    D.Zoom P, 1/125
end if
End sub

```

ZoomRectangle

Parameters

Window As Rectangle

Description

ZoomRectangle will change the content of the document window so that the objects (points, lines, ...) within the rectangle defined by the parameter Window will become visible.
Window : a rectangle with the coordinates of the bottom left and top right corners of a rectangle in document coordinates.

Example

```

Sub PM_ZoomRectangle

Dim A As Application
Dim D As Document
Dim R As Rectangle

set A = New Application
A.GetActiveDocument D
If NOT D is nothing Then
    R.x1 = -10.0
    R.y1 = 10.0
    R.x2 = 10.0
    R.y2 = -10.0
    D.ZoomRectangle R
end if
End sub

```

FitInWindow

Parameters

None

Description

This method will display the complete drawing on the screen. The parts of the drawing falling outside the page boundary will not be displayed.
Behavior is identical as the menu selection : View, Fit In Window.

Example

```
Dim D As Document

D.FitInWindow
```

ShowAll

Parameters

None

Description

This method will display the complete drawing on the screen. If the drawing fits on the page, the page position will be moved so that the drawing is centered relative to the page. If the drawing does not fit on the page, the page is hidden and all objects will become visible on the screen.

Example

```
Dim D As Document

D.ShowAll
```

GetSheets

Parameters

Shts As Sheets

Description

GetSheets returns the set of sheets in the document.

Example

```
Dim A As Application
Dim D As Document
Dim Shts As Sheets

Set A = New Application
A.GetActiveDocument D
D.GetSheets Shts
```

GetLayers

Parameters

Ls As Layers

Description

GetLayers returns the set of layers in the document.

Example

```
Sub PM_ChangeAttribsAllLayers

Dim A As Application
Dim D As Document
Dim G As LayerGroup
Dim Ls As Layers
Dim L As Layer

set A = New Application
A.GetActiveDocument D
If Not (D is Nothing) then
    D.GetLayers Ls
    Ls.SetProtected False
    Ls.SetVisible False
    Ls.SetWallpaper True
    Ls.GetFirst L
    If not L is nothing then
        L.SetMinimumAndMaximumScale 1/1000, 1/500
        set L = nothing
    end if
end if
End sub
```

GetLayerGroups

Parameters

Group As LayerGroup

Description

GetLayerGroups returns a set containing all layer groups. Remember that in the Pythagoras Layer dialog box, you can create layers and layer groups. A layer group may contain 0, 1 or more layers.

Example

```
Sub PM_DeleteAllLayergroups
Dim A As Application
Dim D As Document
Dim G As LayerGroup
Dim LGs As LayerGroups

Set A = New Application
A.GetActiveDocument D
If Not (D is Nothing) then
    D.GetLayerGroups LGs
    MsgBox "Nbr of LayerGroups : " & Str(LGs.NrOfGroups)
    Do While LGs.NrOfGroups > 0
        LGs.GetFirst G
        LGs.DeleteGroup G
    Loop
    MsgBox "Nbr of LayerGroups : " & Str(LGs.NrOfGroups)
end if
End sub
```

GetSymbolLibrary

Parameters

DocLibrary As Library

Description

GetSymbolLibrary returns the Document Library of the document on which the method is applied. This object can be used to obtain information about all symbols, line styles, hatch patterns and typical sections of the Document Symbol Library.

Note : to get information about the System Library, use the Application method GetSymbolLibrary.

Example

```
Dim A As Application
Dim D As Document
Dim L As Library
Dim PS As Symbols

set A = New Application
A.GetActiveDocument D
If Not (D is Nothing) then
    D.GetSymbolLibrary L
    L.GetPointSymbols PS
    MsgBox "Nr of Document Point symbols : " & str(PS.NrOfSymbols)
    A.GetSymbolLibrary L
    L.GetPointSymbols PS
    MsgBox "Nr of Library Point symbols : " & str(PS.NrOfSymbols)
End If
```

GetWindow

Parameters

MyWindow As Window

Description

GetWindow returns the Window object of the document on which the method is applied.

Example

```
Dim A As Application
Dim D As Document
Dim P As XY
Dim W As Window

set A = New Application
```

```

A.GetActiveDocument D
If NOT D is nothing Then
    P.x = 20.0
    P.y = 10.0
    D.Zoom P, 1/125
    A.Wait 2000
    D.GetWindow W
    W.VertScrollThumb = 1
    W.HorScrollThumb = 99
end if

```

GetTerrainModels

Parameters

DTMs As TerrainModels

Description

GetTerrainModels returns the set of Terrain Models (DTM's) in the document. If there are no DTM's, the set is empty.

Example

```

Dim A As Application
Dim D As Document
Dim DTMs As TerrainModels

set A = New Application
A.GetActiveDocument D
If NOT D is nothing Then
    D.GetTerrainModels DTMs
end if

```

GetSubDocuments

Parameters

SubDocs As SubDocuments

Description

GetSubDocuments returns the set of subdocuments in the document. There is at least 1 subdocument (the main document) in the set.

Example

```

Dim A As Application
Dim D As Document
Dim SubDocs As SubDocuments

set A = New Application
A.GetActiveDocument D
If NOT D is nothing Then
    D.GetSubDocuments SubDocs
end if

```

GetTiePointsSetCollection

Parameters

TPSC As TiePointsSetCollection

Description

Returns the collection of all Tie Points Sets in the Pythagoras document. Only the sets that are not in the library (see same method on Application) are returned, even if a set is used in the document.

Example

```

Dim Doc As Document
Dim TPSC As TiePointsSetCollection

Doc.GetTiePointsSetCollection TPSC

```


FirstTable, NextTable, TableByName, CreateTable, ImportDFB, DropTable, CopyTableSkeleton, CopyTable, FirstDBView, NextDBView, DBViewByName, CreateSimpleQuery, CreateAggregationQuery, DropQuery, CopyQuery, CopyDatabaseSkeleton, CopyDatabase, PurgeDatabase, DropDatabase, LockDatabase, UnlockDatabase, ActualizeQueries

Duplicates of methods in Database class, see their descriptions. Ex. `Document.DropDatabase` call is the same as `Document.Database.DropDatabase`.

ShowDataViewer

Parameters

`Names() As String`

Description

Opens the Data Viewer and shows all tables that are in `Names`. The document will be put on top if it was not so.

Example

```
Dim A As Application
Dim D As Document
Dim Names() As String

Set A = New Application
A.GetActiveDocument D
If Not (D Is Nothing) then
    Redim Names(1 To 1)
    Names(1) = "*Polygons"
    D.ShowDataViewer Names
End If
```

GetThematicMaps

Parameters

`TMs As ThematicMaps`

Description

Returns the Thematic Maps of the document.

Example

```
Dim A As Application
Dim D As Document
Dim TMs As ThematicMaps
Dim TM As ThematicMap

Set A = New Application
A.GetActiveDocument D
If Not (D Is Nothing) then
    D.GetThematicMaps TMs
    TMs.GetFirst TM
    Do While Not (TM Is Nothing)
        TMs.GetNext TM
    Loop
End If
```

GetViews

Parameters

`Vws As Views`

Description

Returns the Views of the document.

Example

```
Dim A As Application
Dim D As Document
Dim Vws As Views
Dim V As View

Set A = New Application
A.GetActiveDocument D
If Not (D Is Nothing) then
    D.GetViews Vws
```

```

        Vws.GetFirst V
        Do While Not (V Is Nothing)
            Vws.GetNext V
        Loop
    End If

```

PointIdentUsed

Parameters

PointIdent As String

Returns

Boolean

Description

Returns True if the PointIdent is already used in the document.

Example

```

Dim A As Application
Dim D As Document
Dim PntId As String

A.GetActiveDocument D
PntId = D.NextPointIdent
If D.PointIdentUsed(PntId) then
    A.IncPointIdent PntId
End If

```

Vectorise

Parameters

VectData As VectoriseData, lines() As Line3D, nbLines As Integer

Description

This method vectorises the point cloud of the document.

VectData can be initialised with the Application method InitVectoriseData. Initially there is no section box or section plane set.

VectData.type: must be either ptVectoriseSection, ptVectorise3DEdges or ptVectorise3DLines.

VectData.isBox: If this parameter is True then the points in BLN, TLN, BRN and BLF indicate the section box. The point cloud points in this box are considered for the vectorisation. If this parameter is False, the points in p1 and p2 indicate two parallel planes which form the section plane. The point cloud points inside these two planes are considered for the vectorisation.

When no section box or section plane is required these points should contain all zeros.

For ptVectoriseSection, there needs to be a section box or section plane, because one of the planes is used to vectorise on. When there is a section plane, the plane containing the first point, is the one to vectorise on. When there is a section box, the face of the box with the biggest area is chosen. From these two planes, the plane which contains three of the given points is vectorised on.

Example

```

Dim A As Application
Dim D As Document
Dim VectData As VectoriseData
Dim lines() As Line3D
Dim nbLines As Integer, I As Integer
Dim drawing As Overlay
Dim p1 as Point
Dim p2 as Point
Dim l As Line

Set A = New Application
A.GetActiveDocument D
D.GetOverlay ptDrawing, drawing
A.InitVectoriseData VectData
VectData.type = ptVectorise3DLines
D.Vectorise VectData, lines, nbLines

For i = 1 TO nbLines
    drawing.CreatePoint lines(i).p1, p1

```

```

        drawing.CreatePoint lines(i).p2, p2
        drawing.CreateLine p1, p2, l
    Next i

```

GetCameras

Parameters

Cams As Cameras

Description

Returns the 3D cameras stored into the document.

Example

```

Dim A As Application
Dim D As Document
Dim Cams As Cameras
Dim Cam As Camera

Set A = New Application
A.GetActiveDocument D
If Not (D Is Nothing) then
    D.GetCameras Cams
    Cams.GetFirst Cam
    Do While Not (Cam Is Nothing)
        Cams.GetNext Cam
    Loop
End If

```

Chapter 6 - Documents

Description

A Documents object is a set of all open documents in Pythagoras. The Documents object is created by the Application method GetDocuments.

Example

```
' Macro saves all open documents
Dim A As Application
Dim Docs As Documents
Dim Doc As Document
Dim i As Long

Set A = New Application
A.GetDocuments Docs
Docs.GetFirst Doc
For i = 1 To Docs.NrOfDocuments
    Doc.Save
    Docs.GetNext Doc
Next i
```

Properties

NrOfDocuments

Methods

GetFirst
GetNext

Properties

NrOfDocuments

Returns

Long

Description

This property returns the number of open documents.

Example

See introduction.

Methods

GetFirst

Parameters

Doc As Document

Description

Returns the first open document.

Example

See introduction.

GetNext

Parameters

Doc As Document

Description

Returns the next open document.

Example

See introduction.

Chapter 7 - SubDocument

Description

A SubDocument object can be obtained from either a CadObject by using the method GetSubdocument or from SubDocuments by using the methods GetFirst, GetNext, AddSubDocument, GetActive.

Example

```
Dim P As Point
Dim SubDoc As SubDocument

P.GetSubdocument SubDoc ' Returns the subdocument containing point P.
```

Properties

Name
Active
Main
Status
NameRelative
DifferentDatabaseLock

Methods

GetDocument

Properties

Name

Returns

String

Description

This property returns the filename of the subdocument. E.g. "C:\Topo5X3.pyt".

Example

```
Dim FileName As String
Dim SubDoc As SubDocument

FileName = SubDoc.Name
```

Active

Returns

BOOLEAN

Description

This property returns true if the subdocument is the active one.

Example

```
Dim SubDoc As SubDocument

If SubDoc.Active Then
    MsgBox "The document " & SubDoc.Name & " is the active subdocument"
End If
```

Main

Returns

BOOLEAN

Description

This property returns True if the subdocument is the main document.

Example

```
Dim SubDoc As SubDocument

If SubDoc.Main Then
    MsgBox "The document " & SubDoc.Name & " is the main document"
End If
```

Status

Returns

Integer

Description

This property sets or returns the status of a subdocument. The status of a subdocument is `ptSleepingSubDocument`, `ptWallpaperSubDocument`, `ptVisibleSubDocument`, or `ptEditableSubDocument`.

WARNING: the undo-system of the document is cleared when changing the status.

Example

```
Dim SubDoc As SubDocument

If SubDoc.Status = ptSleepingSubDocument Then
    SubDoc.Status = ptEditableSubDocument
End If
```

NameRelative

Returns

Boolean

Description

This property sets or returns the way the name of a subdocument is stored in the main document. Relative means that only part of the name (eg. Utilities/WaterNet.pyt) is stored. This is only possible if the subdocument is in the folder or a nested folder of the main document. When moving a document with its subdocuments to another drive or folder, the reference to subdocuments remains valid.

Example

```
Dim SubDoc As SubDocument

If SubDoc.NameRelative = False Then
    SubDoc.NameRelative = True
End If
```

DifferentDatabaseLock

Returns

Boolean

Description

This property returns true if one of the following conditions is met:

1. Database in the main document is locked, but database in the subdocument is not locked
2. Database in the main document is unlocked, but database in the subdocument is locked.
3. Databases in the main document and in the subdocument are locked with a different name or password.

Example

```
Dim SubDoc As SubDocument

If SubDoc.DifferentDatabaseLock Then
    ' Give warning
End If
```

Methods

GetDocument

Parameters

Doc As Document

Description

Returns the document that contains this and all other subdocuments.

Example

```
Dim Doc As Document
Dim SubDoc As SubDocument

SubDoc.GetDocument Doc
```

Chapter 8 - SubDocuments

Description

A Subdocuments object is a set of all subdocuments in a Pythagoras document. The SubDocuments object is created by the Document method GetSubDocuments.

Example

```
Dim A As Application
Dim SubDocs As SubDocuments
Dim SubDoc As SubDocument
Dim Doc As Document
Dim i As Long

set A = New Application
A.GetActiveDocument Doc
Doc.GetSubDocuments SubDocs
SubDocs.GetFirst SubDoc
For i = 1 To SubDocs.NrOfSubDocuments - 1
    SubDocs.GetNext SubDoc
Next i
```

Properties

NrOfSubDocuments

Methods

GetFirst
GetNext
AddSubDocument
RemoveSubDocument
GetMain
GetActive
SetActive
GetDocument

Properties

NrOfSubDocuments

Returns

Long

Description

This property returns the number of subdocuments in a document.

Example

See introduction.

Methods

GetFirst

Parameters

SubDoc As SubDocument

Description

Returns the first subdocument.

Example

See introduction.

GetNext

Parameters

SubDoc As SubDocument

Description

Returns the next subdocument.

Example

See introduction.

AddSubDocument

Parameters

Name As String, State As Integer, SubDoc As SubDocument,
Optional HandleDataOfSubdocuments As PtSynchronizeDictionaryData
:= ptDataRaiseError

Description

Adds the file with name equal to Name to a document. The initial state of the document is ptSleepingSubDocument, ptWallpaperSubDocument, ptVisibleSubDocument, or ptEditableSubDocument. A reference to the new subdocument is returned in the parameter SubDoc.

If Name is not a complete pathname, the file with name PathNameOverview & Name becomes the subdocument. The subdocument will by default be absolute. If a relative name is required, the SubDocument property NameRelate has to be set.

Input parameter HandleDataOfSubdocument defines what to do when the overview document and the subdocument to add contain a database dictionary table with the same name and structure, but with different datastamps. Data in such tables can differ. See also description of PtSynchronizeDictionaryData enumeration. When value PtDataDontOpenSubdocument is specified it is possible to test if the subdocument has been added by checking ifSubDocisNothing.

WARNING: the undo-system of the document is cleared.

Example

```
Dim A As Application
Dim SubDocs As SubDocuments
Dim SubDoc As SubDocument
Dim Doc As Document

set A = New Application
A.GetActiveDocument Doc
Doc.GetSubDocuments SubDocs
SubDocs.AddSubDocument "C:\Topo4x3.pyt", ptSleepingSubDocument, SubDoc
```

RemoveSubDocument

Parameters

SubDoc As SubDocument

Description

Removes a Subdocument.

WARNING: the undo-system of the document is cleared.

Example

```
Dim A As Application
Dim SubDocs As SubDocuments
Dim SubDoc As SubDocument
Dim Doc As Document

set A = New Application
A.GetActiveDocument Doc
Doc.GetSubDocuments SubDocs
' ....
SubDocs.RemoveSubDocument SubDoc
```

' Get a SubDoc
' Removes the subdocument.

GetMain

Parameters

SubDoc As SubDocument

Description

Returns the main subdocument.

Example

```
Dim A As Application
Dim SubDocs As SubDocuments
```



```

Dim SubDoc As SubDocument
Dim Doc As Document

set A = New Application
A.GetActiveDocument Doc
Doc.GetSubDocuments SubDocs
SubDocs.GetMain SubDoc

```

GetActive

Parameters

SubDoc As SubDocument

Description

Returns the active subdocument.

Example

```

Dim A As Application
Dim SubDocs As SubDocuments
Dim SubDoc As SubDocument
Dim Doc As Document

set A = New Application
A.GetActiveDocument Doc
Doc.GetSubDocuments SubDocs
SubDocs.GetActive SubDoc

```

SetActive

Parameters

SubDoc As SubDocument

Description

Sets a subdocument active.

Example

```

Dim A As Application
Dim SubDocs As SubDocuments
Dim SubDoc As SubDocument
Dim Doc As Document

set A = New Application
A.GetActiveDocument Doc
Doc.GetSubDocuments SubDocs
' ... ' Obtain a subdocument
SubDocs.SetActive SubDoc

```

GetDocument

Parameters

Doc As Document

Description

Returns the document that contains this and all other subdocuments.

Example

```

Dim Doc As Document
Dim SubDocs As SubDocuments

SubDocs.GetDocument Doc

```

Chapter 9 - View

Description

A View object can be obtained from Views by using the methods GetFirst, GetNext, GetView, GetActive.

Example

```
Dim Vs As Views
Dim V As View

Vs.GetActive V           ' Returns the active view.
```

Properties

Name
Active
Visible

Properties

Name

Returns

String

Description

This property returns or sets the name of the view.

Example

```
Dim V As View

MsgBox "The name of this view is " & V.Name
```

Active

Returns

Boolean

Description

Returns True if a view is the active view and False – otherwise.

Example

```
Dim V As View

If V.Active Then
    MsgBox "The View " & V.Name & " is active"
End If
```

Visible

Returns

Visibility As Boolean

Description

Returns or sets the visibility of polygon patterns, texts, point numbers, point comments, point elevations, special line styles, and images. Possible values: ptPointNumbers, ptComments, ptPointElevations, ptAreas, ptTexts, ptImages, ptSpecialLineStyles, ptPatterns.

Example

```
Dim A As Application
Dim D As Document
Dim Vs As Views
Dim V As View

set A = New Application
A.GetActiveDocument D
If Not (D is Nothing) Then
    D.GetViews Vs
    Vs.GetActive V
```

```
        V.Visible(ptTexts) = True  
End If
```

Chapter 10 - Views

Description

A Views object is a set of all views of a document. The Views object is created by the Document method GetViews.

Example

```
'Find View named as "Main" and make it active
Dim A As Application
Dim D As Document
Dim Vs As Views
Dim V As View

set A = New Application
A.GetActiveDocument D
If Not (D is Nothing) then
    D.GetViews Vs
    Vs.GetView "Main", V
    If Not (V Is Nothing) Then
        V.Active = True
    End IF
End If
End If
```

Properties

NrOfViews

Methods

GetFirst
GetNext
GetView
GetActive
SetActive

Properties

NrOfViews

Returns

Long

Description

This property returns the number of views.

Example

```
Dim A As Application
Dim D As Document
Dim Vs As Views
Dim N As Long

set A = New Application
A.GetActiveDocument D
If Not (D is Nothing) then
    N = Vs.NrOfViews
End If
```

Methods

GetFirst

Parameters

V As View

Description

Returns the first view.

Example

```
Dim A As Application
Dim D As Document
```

```

Dim Vs As Views
Dim V As View

set A = New Application
A.GetActiveDocument D
If Not (D Is Nothing) then
    D.GetViews Vs
    Vs.GetFirst V
Do While Not(V Is Nothing)
V.Visible(ptAreas) = False
    Vs.GetNext V
Loop
End If

```

GetNext

Parameters

V As View

Description

Returns the next View.

Example

See GetFirst.

GetView

Parameters

Name As String
V As View

Description

Returns the view with a name specified by the argument string or Nothing if there is no such a view.

Example

```

Dim A As Application
Dim Vs As Views
Dim V As View
Dim Doc As Document

set A = New Application
A.GetActiveDocument Doc
Doc.GetViews Vs
Vs.GetView "Main", V

```

GetActive

Parameters

V As View

Description

Returns the active view.

Example

```

Dim A As Application
Dim Vs As Views
Dim V As View
Dim Doc As Document

set A = New Application
A.GetActiveDocument Doc
Doc.GetViews Vs
Vs.GetActive V

```

SetActive

Parameters

v As View

Description

Sets a view active.

Example

```
Dim A As Application
Dim Vs As Views
Dim V As View
Dim Doc As Document

set A = New Application
A.GetActiveDocument Doc
Doc.GetViews Vs
Vs.GetView "Main", V
Vs.SetActive V
```

Chapter 11 - ThematicMap

Description

A ThematicMap object can be obtained from ThematicMaps by using the methods GetFirst, GetNext.

Example

```
Dim TMs As ThematicMaps
Dim TM As ThematicMap
```

```
TMs.GetFirst TM ' Returns the first thematic map.
```

Properties

Name
Active

Properties

Name

Returns

String

Description

This property returns or sets the name of the thematic map.

Example

```
Dim TM As ThematicMap
```

```
MsgBox "The name of this thematic map is " & TM.Name
```

Active

Returns

Boolean

Description

This property returns or sets the status (active / inactive) of the thematic map.

Example

```
Dim TM As ThematicMap
```

```
TM.Active = False
```

Chapter 12 - ThematicMaps

Description

A ThematicMaps object is a set of all thematic maps of a document. The ThematicMaps object is created by the Document method GetThematicMaps.

Example

```
'Find Thematic named "Customers" and make it active
Dim A As Application
Dim D As Document
Dim TMs As ThematicMaps
Dim TM As ThematicMap

set A = New Application
A.GetActiveDocument D
If Not (D is Nothing) then
    D.GetThematicMaps TMs
    TMs.GetFirst TM
    Do While Not (TM Is Nothing)
        If TM.Name = "Customers" Then
            TM.Active = True
        End If
        TMs.GetNext TM
    Loop
End If
```

Properties

NrOfThematicMaps

Methods

GetFirst
GetNext

Properties

NrOfThematicMaps

Returns

Long

Description

This property returns the number of thematic maps.

Example

```
Dim TMs As ThematicMaps
Dim TM As ThematicMap

MsgBox TMs.NrOfThematicMaps
```

Methods

GetFirst

Parameters

TM As ThematicMap

Description

Returns the first Thematic Map.

Example

```
Dim A As Application
Dim D As Document
Dim TMs As ThematicMaps
Dim TM As ThematicMap

set A = New Application
A.GetActiveDocument D
If Not (D is Nothing) then
```



```
        D.GetThematicMaps TMs
        TMs.GetFirst TM
    Do While Not(TM Is Nothing)
        TMs.GetNext TM
    Loop
End If
```

GetNext

Parameters

TM As ThematicMap

Description

Returns the next Thematic Map.

Example

See GetFirst.

Chapter 13 - Sheet

Description

A Sheet object can be obtained from Sheets by using the methods GetFirst, GetNext, GetSheet, GetActive.

Example

```
Dim Shts As Sheets
Dim Sht As Sheet
```

```
Sheets.GetActive Sht ' Returns the active sheet.
```

Properties

Name
Active
PaperWidth
PaperHeight
Margins
ScaleFactor
Scale
Position
PageAngle
Floating

Methods

GetAllObjects
CreatePoint
CreateLine
CreateArc
CreateCircle
CreateText
CreateRichText
CreateClothoid
CreateHorizontalCurve
CreateCurve
CreatePolygon
CreatePath
CreateImage
InsertImage
CreateCompound
Duplicate
PrintOut
PrintToPDF
SetPrintArea
ClearPrintArea

Properties

Name

Returns

String

Description

This property returns or sets the name of the sheet.

Example

```
Dim Sht As Sheet
```

```
MsgBox "The name of this sheet is " & Sht.Name
```

Active

Returns

Boolean

Description

Returns the activity status of the sheet.

Example

```
Dim Sht As Sheet

If Sht.Active Then
    MsgBox "The sheet " & Sht.Name & " is active"
End If
```

PaperWidth

Returns

Width As Double

Description

Returns the width of the selected paper size.

Example

```
Dim A As Application
Dim D As Document
Dim Shts As Sheets
Dim Sht As Sheet

set A = New Application
A.GetActiveDocument D
If Not (D is Nothing) Then
    D.GetSheets Shts
    Shts.GetActive Sht
    MsgBox "PaperDiminsions:" & vbCrLf & _
        "W = " & Cstr(Sht.PaperWidth) & " H = " & Cstr(Sht.PaperHeight)
End If
```

PaperHeight

Returns

Height As Double

Description

Returns the height of the selected paper size.

Example

See PaperWidth

Margins

Returns

Rect As Rectangle

Description

Returns the printable rectangle. The printable rectangle depends on the selected paper size and on the selected printer. P1 is bottom left coordinate and P2 is top right coordinate of printable area.

Rect.p1.x = left margin

Rect.p1.y = bottom margin

Rect.p2.x = PaperWidth - right margin

Rect.p2.y = PaperHeight - top margin

Example

```
Dim A As Application
Dim D As Document
Dim Shts As Sheets
Dim Sht As Sheet
Dim Rect As Rectangle

set A = New Application
A.GetActiveDocument D
If Not (D is Nothing) Then
    D.GetSheets Shts
    Shts.GetActive Sht
    Rect = Sht.Margins
    MsgBox Cstr(Rect.p1.x) & " " & Cstr(Rect.p1.y) & " " & _
        Cstr(Sht.PaperWidth - Rect.p2.x) & " " & Cstr(Sht.PaperHeight - Rect.p2.y)
End If
```

ScaleFactor

Returns

ScaleFactor As Double

Description

Returns or sets the scale percentage of the sheet. The scale factor is not the same as scale. When the ScaleFactor = 1.0, the sheet will be printed at the given scale. When ScaleFactor <> 1.0, the Scale will change accordingly but also all elements of the drawing that are expressed in paper dimensions (texts, linewidth, symbols having paper dimensions, ...) will be scaled with ScaleFactor. The effect is the same as optical zooming. If only the scale is changed (see next property), paper dimensions (text size, line width, ...) remain unchanged.

Example

```
Dim A As Application
Dim D As Document
Dim Shts As Sheets
Dim Sht As Sheet

set A = New Application
A.GetActiveDocument D
If Not (D is Nothing) Then
    D.GetSheets Shts
    Shts.GetActive Sht
    MsgBox Cstr(Sht.ScaleFactor)
End If
```

Scale

Returns

Scale As Double

Description

Returns or sets the scale of the sheet. See also ScaleFactor.

Example

```
Dim A As Application
Dim D As Document
Dim Shts As Sheets
Dim Sht As Sheet

set A = New Application
A.GetActiveDocument D
If Not (D is Nothing) Then
    D.GetSheets Shts
    Shts.GetActive Sht
    Sht.Scale = 0.001           \ Scale = 1/1000
End If
```

Position

Returns

Pos As XY

Description

Returns or sets the position of the sheet (the coordinates of the point (0, 0) in the local coordinate system relative to the bottom left of the sheet). The positions of the other coordinates depend on the angle of the sheet and on Scale and ScaleFactor.

Example

```
Dim A As Application
Dim D As Document
Dim Shts As Sheets
Dim Sht As Sheet
Dim Pos As XY

set A = New Application
A.GetActiveDocument D
If Not (D is Nothing) Then
    D.GetSheets Shts
    Shts.GetActive Sht
```

```

        Pos = Sht.Position
        Pos.X = 0.05
        Pos.Y = 0.025
        Sht.Position = Pos
    End If
' (0.0, 0.0) is printed at
' position (0.05, 0.025) of the page.

```

PageAngle

Returns

Angle As Double

Description

Returns or sets the angle of the sheet.

Example

```

Dim A As Application
Dim D As Document
Dim Shts As Sheets
Dim Sht As Sheet
Dim Angle As Double

set A = New Application
A.GetActiveDocument D
If Not (D is Nothing) Then
    D.GetSheets Shts
    Shts.GetActive Sht
    Angle = Sht.PageAngle
End If

```

Floating

Returns

Boolean

Description

Returns or sets the floating status of the sheet.

Example

```

Dim Sht As Sheet

If Sht.Floating Then
    MsgBox "The sheet " & Sht.Name & " is floating"
End If

```

Methods

The methods CreatePoint, CreateLine, CreateArc, CreateCircle, CreateText, CreateRichText, CreateClothoid, CreateHorizontalCurve, CreateCurve, CreatePolygon, CreatePath, CreateImage, CreateCompound: create objects in overlay Paper in the sheet. The parameters are identical to the parameters of methods of the same name of the object Overlay.

GetAllObjects

Parameters

All As AllObjects, Optional Wallpaper As Boolean = True

Description

Returns a set containing all objects in the sheet of a document. This set will always contain all objects in the sheet even if after calling GetAllObjects new objects are added or objects are deleted. Alternatively the Overlay method GetAllObjects can be used with the optional parameter Sheet. The optional parameter Wallpaper defines whether objects of wallpaper layers are included. If this parameter is True then objects of all layers are included. Otherwise only objects of not wallpaper layers are included.

Example

```

Dim A As Application
Dim Doc As Document
Dim PaperWorld As AllObjects

```

```

Dim Sht As Sheet
Dim Shts As Sheets

Set A= New Application
A.GetActiveDocument Doc
Doc.GetSheets Shts
Shts.GetActive Sht
Sht.GetAllObjects PaperWorld
PaperWorld.Color = ptRed

```

CreatePoint

Parameters

Position As XYZ, P As Point

Description

Creates a point in overlay Paper in the sheet. The object P referring to the created point is returned. The style and attributes of the new created point correspond with the defaults set by the user.

Example

See example Overlay.CreatePoint

CreateLine

Parameters

P1 As Point, P2 As Point, L As Line

Description

Creates a line (in overlay Paper in the sheet) from point P1 to point P2. P1 and P2 are point objects. These points must have been created before calling CreateLine and must be in the same overlay as the new line. The object L referring to the created line is returned.

Example

See example Overlay.CreateLine

CreateArc

Parameters

P1 As Point, P2 As Point, Center As XY, NewArc As Arc

Description

Creates an arc in either Paper (active sheet) or Drawing Overlay. The direction of the arc is counter clockwise from point P1 to point P2. These points must have been created before calling CreateArc and must be in the same overlay as the new arc. The object A referring to the created arc is returned. If the distance of P1 and P2 is not the same, then the error 1055 is raised.

Example

See example Overlay.CreateArc

CreateCircle

Parameters

Center As XYZ, CirclePoint As XYZ, NewCircle As Circle

Description

Creates a circle in Paper overlay of the sheet.
Center : The centre of the circle.
CirclePoint : A point on the circumference of the circle.
The object NewCircle referring to the created circle is returned.

Example

See example Overlay.CreateCircle

CreateText

Parameters

Position As XY, Text As String, T As Text

Description

Creates the text with content `Text` at `Position` in Paper overlay of the sheet. The object `T` referring to the created text is returned. The length of `Text` may be maximum 255 characters. A new line in the text is indicated with a CR (013x) or CRLF.

Example

See example `Overlay.CreateText`

CreateRichText

Parameters

`Position` As `XYZ`, `Height` As `Double`, `Width` As `Double`, `Text` As `String`, `RT` As `RichText`

Description

Creates the `RichText` object with content `Text` at `Position` in Paper overlay of the sheet. The object `RT` referring to the created `RichText` is returned. A new line in the text is indicated with `vbCrLf`.

Example

See example `Overlay.CreateRichText`

CreateClothoid

Parameters

`P1` As `XYZ`, `P2` As `XYZ`, `RStart` As `Double`, `REnd` As `Double`, `K` As `Double`, `Mirror` As `Boolean`, `Cl` As `Clothoid`

Description

Creates a clothoid (transition curve) in Paper overlay of the sheet. The object `Cl` referring to the created clothoid is returned.

`P1` : Starting point

`P2` : End point

`RStart` : Radius at starting point

`REnd` : Radius at end point

`K` : K-Factor of the clothoid

`Mirror` : If `False` the clothoid is in counter-clockwise direction (`P1` - `P2`). If `True` clockwise.

If the radius of the starting or end point is infinite, the value `ptMaxDouble` must be passed.

Note : The parameters must be specified very accurately. If the resulting clothoid can not be calculated, run-time error 1056 will be generated.

Example

See example `Overlay.CreateClothoid`

CreateHorizontalCurve

Parameters

`Points()` As `XYZ`, `Z` As `Double`, `C` As `Curve`

Description

Creates a polyline through `Points` with line segments with elevation `Z` in Paper overlay of the sheet. The object `C` referring to the created polyline is returned. The parameter `Points` must be a dynamic array. A fixed array is not accepted.

Example

See example `Overlay.CreateHorizontalCurve`

CreateCurve

Parameters

`Points()` As `XYZ`, `C` As `Curve`

Description

Creates a polyline through `Points` with line segments in Paper overlay of the sheet. The object `C` referring to the created polyline is returned. The parameter `Points` must be a dynamic array. A fixed array is not accepted.

Example

See example `Overlay.CreateCurve`

CreatePolygon

Parameters

`Objs()` As `CadObject`, `Poly` As `Polygon`

Description

Creates a polygon in Paper overlay of the sheet. The dynamic array `Objs`, must contain the elements that make up the polygon and should have the same dimension. (Use `Redim`). If the polygon contains only one element, this must be a circle. In all other cases, the first element of `Objs` must be a `CadObject` of type `Point`. The following elements can be points, arcs or clothoids. A polygon must be closed, this means the last element must end in the first element. When the last element is an arc or a clothoid, then the end point of that arc or clothoid must be the same as the first point of the polygon.

Example :

A - B : Line

B - C : Arc

C - D : Arc

D - E : Line

E - A : Arc

Then `Objs` must contain (Point A, Arc BC, Arc CD, Point E, Arc EA).

The object `Poly` referring to the created polygon is returned.

Example

See example `Overlay.CreatePolygon`

CreatePath

Parameters

`Objs()` As `CadObject`, `P` As `Path`

Description

This method creates a path. The dynamic array `Objs`, must contain the elements that make up the path and should have the same dimension. (Use `Redim`). The rules regarding the type and sequence of elements in `Objs` is the same as for a polygon, except for the rule that the starting point and end point must be the same.

The object `P` referring to the created path is returned.

Example

See example `CreatePolygon`.

CreateImage

Parameters

`Name` As `String`, `Pos` As `XY`, `MyImage` As `Image`

Description

Creates a Pythagoras image object referring to an existing Pythagoras image file (*.ptif or *.img file). If such file does not exist, but an image file of world-known format exists (BMP, JPEG, TIFF, etc), then the Application method `ConvertImage` has to be used first in order to create a PTIF file.

`Name` = name of the Pythagoras image file.

`MyImage` = On return : the image Object.

Example

```
Dim Sht As Sheet
```

```
Dim Pos As XY
```

```
Dim I As Image
```

```
Pos.x = 0.01
```

```
Pos.y = 0.15
```

```
Sht.CreateImage "C:\MyImage.ptif", Pos, I
```

InsertImage

Parameters

`Name` As `String`, `Pos` As `XY`, `MyImage` As `Image`

Description

Creates a Pythagoras image object enclosed (embedded) in the document.

Name = name of a BMP, TIFF, JPG or Pythagoras image file.

Pos = the coordinates of the top-left corner, in paper dimensions, of the image.

MyImage = On return : the image Object.

Example

```
Dim I As Image
Dim Sht As Sheet
Dim Pos As XY

Pos.x = 0.01
Pos.y = 0.15
Sht.InsertImage "C:\Test\Test.tif", Pos, I
```

CreateCompound

Parameters

Container As Compound

Description

Creates a Pythagoras compound object. The compound will contain no elements. Use the

CadObject/Point/Line/... property Container to add one object at a time to the compound.

If a compound is created in a macro, and no object is included into this compound then the compound can be silently deleted after the macro execution completion.

Example

See example Overlay.CreateCompound

Duplicate

Returns

Sheet

Description

Duplicates the sheet and returns a new sheet. All sheet properties are duplicated except the sheet name.

WARNING: the undo-system of the document is cleared.

Example

```
Dim A As Application
Dim Sht As Sheet, NewSht As Sheet
Dim Doc As Document

set A = New Application
A.GetActiveDocument Doc
Set Sht = D.ActiveSheet
Set NewSht = Sht.Duplicate
```

PrintOut

Parameters

Copies As Long, Status As Long

Description

Prints one or several Copies of the sheet on the printer specified for the sheet. The sheet may not be

floating and all printer settings must match with the attached printer(s). The status of printing is returned in the parameter Status. The printing is successful if Status = ptOk. Otherwise, ptNotOk is returned in

Status.

Example

```
Dim Sht As Sheet
Dim Status As Long

If Not Sht.Floating Then
    Sht.PrintOut 1, Status
    If Status <> 0 Then
        MsgBox "Can't print the sheet."
    End If
End If
```

PrintToPDF

Parameters

Filename As String, Status As Long, Optional OpenPDFFile As Boolean

Description

Prints the sheet to PDF with the specified `Filename`. The sheet may not be floating. The status of printing is returned in the parameter `Status`. The printing is successful if `Status = ptOk`. Otherwise, `ptNotOk` is returned in `Status`.

When `OpenPDFFile` is `TRUE` the file is opened automatically in the default PDF viewer.

Example

```
Set App = New Application
App.GetActiveDocument Doc
Set actSheet = Doc.ActiveSheet

actSheet.PrintToPDF "C:\Temp\pyth_sheet.pdf", status
```

SetPrintArea

Parameters

NrOfObjects As Long, Objects() As CadObject

Description

Sets the print area of the sheet equal to the polygon formed by the array of `CadObjects`. It is required that the list of objects contains only points, lines, arcs or clothoids, and that these objects can be connected to a polygon in the order of the array.

Example

```
Dim App As Application
Dim Doc As Document
Dim Sht As Sheet
Dim AllObjs As AllObjects
Dim Obj As CadObject
Dim Poly As Polygon
Dim PolyObjs() As CadObject

Set App = New Application
App.GetActiveDocument Doc
Set Sht = Doc.ActiveSheet
Sht.GetAllObjects AllObjs
If AllObjs.Count(ptPolygon) > 0 Then
    AllObjs.GetFirst ptPolygon, Obj
    Obj.GetPolygon Poly
    Redim PolyObjs(1 To Poly.NrOfObjects)
    Poly.GetPolygonObjects PolyObjs
    Sht.SetPrintArea Poly.NrOfObjects, PolyObjs
Else
    MsgBox("No polygon")
End If
```

ClearPrintArea

Description

Clears the print area of the sheet.

Example

```
Dim App As Application
Dim Doc As Document
Dim Sht As Sheet

Set App = New Application
App.GetActiveDocument Doc
Set Sht = Doc.ActiveSheet
If Not Sht.Floating Then
    Sht.ClearPrintArea
End If
```

Chapter 14 - Sheets

Description

A `Sheets` object is a set of all sheets of a document. The `Sheets` object is created by the `Document` method `GetSheets`.

Example

```
'Find sheet named as "Overall" and make it floating
Dim A As Application
Dim D As Document
Dim Shts As Sheets
Dim Sht As Sheet

set A = New Application
A.GetActiveDocument D
If Not (D Is Nothing) then
    D.GetSheets Shts
    Shts.GetSheet "Overall", Sht
    If Not (Sht Is Nothing) Then
        Sht.Floating = True
    End IF
End If
```

Properties

`NrOfSheets`

Methods

`GetFirst`
`GetNext`
`GetSheet`
`GetActive`
`SetActive`

Properties

NrOfSheets

Returns

`Long`

Description

This property returns the number of sheets.

Example

```
Dim A As Application
Dim D As Document
Dim Shts As Sheets
Dim N As Long

set A = New Application
A.GetActiveDocument D
If Not (D Is Nothing) then
    N = Shts.NrOfSheets
End If
```

Methods

GetFirst

Parameters

`Sht As Sheet`

Description

Returns the first sheet.

Example

```
Dim A As Application
Dim D As Document
```

```

Dim Shts As Sheets
Dim Sht As Sheet

set A = New Application
A.GetActiveDocument D
If Not (D Is Nothing) then
    D.GetSheets Shts
    Shts.GetFirst Sht
Do While Not(Sht Is Nothing)
    Sht.ScaleFactor = 150.0
    Shts.GetNext Sht
Loop
End If

```

GetNext

Parameters

Sht As Sheet

Description

Returns the next sheet.

Example

See GetFirst.

GetSheet

Parameters

SheetName As String, Sht As Sheet

Description

Returns the sheet with the given name. If the sheet with the given name is not found, then on return Sht = Nothing.

Example

```

Dim Shts As Sheets
Dim S As Sheet

Shts.GetSheet "Valley", Sht
If Not (Sht Is Nothing) Then
    Sht.ScaleFactor = 100.0
End If

```

GetActive

Parameters

Sht As Sheet

Description

Returns the active sheet.

The Document property ActiveSheet can be used for the same purpose.

Example

```

Dim A As Application
Dim Shts As Sheets
Dim Sht As Sheet
Dim Doc As Document

set A = New Application
A.GetActiveDocument Doc
Doc.GetSheets Shts
Shts.GetActive Sht

```

SetActive

Parameters

Sht As Sheet

Description

Sets a sheet active.

The Document property ActiveSheet can be used for the same purpose.

Example

```
Dim A As Application
Dim Shts As Sheets
Dim Sht As Sheet
Dim Doc As Document
```

```
set A = New Application
A.GetActiveDocument Doc
Doc.GetSheets Shts
'...
Shts.SetActive Sht
```

' Obtain a sheet

Chapter 15 - Overlay

Description

Objects (points, lines, arcs, ...) in a Pythagoras document are created in either the Paper or Drawing Overlay.

Example

```
Dim A As Application
Dim Doc As Document
Dim Ovl As Overlay

Set A = New Application
A.GetNewDocument Doc ' Untitled document becomes the top window
Doc.GetOverlay ptDrawing, Ovl ' Returns an object referring to the Drawing
                              Overlay of document Doc.
```

Properties

OverlayType

Methods

```
CreatePoint
CreateLine
CreateArc
CreateCircle
CreateText
CreateRichText
CreateClothoid
CreateHorizontalCurve
CreateCurve
CreatePolygon
CreatePath
CreatePathElement
CreatePathElement3D
CreateImage
InsertImage
CreateCompound
GetAllObjects
GetCoordinateSystems
CreateTempCoordinateSystem
CreateUserCoordinateSystem
DeleteCoordinateSystem
Clip
XYToPoint
```

Properties

OverlayType

Returns

Integer

Description

This property returns the type of the overlay. This type can be ptPaper or ptDrawing.

Example

```
Dim A As Application
Dim D As Document
Dim Ovl As Overlay

set A = New Application
A.GetActiveDocument D
D.GetOverlay ptDrawing, Ovl
If Ovl.OverlayType = ptDrawing Then
    MsgBox "Correct"
Else
    MsgBox "Incorrect"
End If
```

Methods

If objects are created in Paper overlay using the methods *Create_Obj* (Point, line, ...), these objects will be created in the active Sheet. When you want to create objects in a specific sheet, you need to use the methods of the Class Sheet.

CreatePoint

Parameters

Position As XYZ, P As Point

Description

Creates a point in either Paper (active sheet) or Drawing Overlay. The object P referring to the created point is returned. The style and attributes of the new created point correspond with the defaults set by the user.

Example

```
Dim A As Application
Dim Doc As Document
Dim Position As XYZ
Dim NewPoint As Point
Dim Drawing As Overlay

Set A= New Application
A.GetActiveDocument Doc
Doc.GetOverlay ptDrawing, Drawing
Position.X = 10.0
Position.Y = 25.2
Position.Z = 10.0
Drawing.CreatePoint Position, NewPoint      ' Creates point in Drawing Overlay
NewPoint.Color = ptRed
```

CreateLine

Parameters

P1 As Point, P2 As Point, L As Line

Description

Creates a line from point P1 to point P2 in either Paper (active sheet) or Drawing Overlay. P1 and P2 are point objects. These points must have been created before calling *CreateLine* and must be in the same overlay as the new line. The object L referring to the created line is returned.

The style and attributes of the new created line correspond with the defaults set by the user.

Example

```
Dim A As Application
Dim Doc As Document
Dim P1 As Point, P2 As Point
Dim NewLine As Line
Dim Drawing As Overlay
Dim Pos As XYZ

Set A = New Application
A.GetActiveDocument Doc
Doc.GetOverlay ptDrawing, Drawing
Pos.X=10
Pos.Y=10
Pos.Z=0
Doc.NewState "Undo Point1", "Redo Point1"
Drawing.CreatePoint Pos, P1      ' Create point in Drawing overlay
Doc.EndState True
Pos.X=20
Pos.Y=20
Doc.NewState "Undo Point2", "Redo Point2"
Drawing.CreatePoint Pos, P2      ' Create a second point
Doc.EndState True
Drawing.CreateLine P1, P2, NewLine
```

CreateArc

Parameters

P1 As Point, P2 As Point, Center As XY, NewArc As Arc

Description

Creates an arc in either Paper (active sheet) or Drawing Overlay. The direction of the arc is counter clockwise from point P1 to point P2. These points must have been created before calling CreateArc and must be in the same overlay as the new arc. The object A referring to the created arc is returned. If the distance of P1 and P2 is not the same, then the error 54 is raised.

The style and attributes of the new created arc correspond with the defaults set by the user.

Example

```
Dim A As Application
Dim Doc As Document
Dim P1 As Point, P2 As Point
Dim Pos1 As XYZ, Pos2 As XYZ
Dim Center As XY
Dim NewArc As Arc
Dim Drawing As Overlay

Set A= New Application
A.GetActiveDocument Doc
Doc.GetOverlay ptDrawing, Drawing
Pos1.x = 10.0
Pos1.y = 10.0
Pos1.z = 0.0
Drawing.CreatePoint Pos1, P1
Pos2 = Pos1
Pos2.x = -10.0
Drawing.CreatePoint Pos2, P2
Center.x = 0.0
Center.y = 0.0
Drawing.CreateArc P1, P2, Center, NewArc
```

CreateCircle

Parameters

Center As XYZ, CirclePoint As XYZ, NewCircle As Circle

Description

Creates a circle in either Paper (active sheet) or Drawing Overlay.

Center : The centre of the circle.

CirclePoint : A point on the circumference of the circle.

The object NewCircle referring to the created circle is returned.

The style and attributes of the new created circle correspond with the defaults set by the user.

Example

```
Dim A As Application
Dim Doc As Document
Dim Center As XYZ, Pos2 As XYZ
Dim P1 As point
Dim NewCircle As Circle
Dim Drawing As Overlay

Set A= New Application
A.GetActiveDocument Doc
Doc.GetOverlay ptDrawing, Drawing
Pos2.x = 10.0
Pos2.y = 10.0
Pos2.z = 0.0
Center.x = 5.0
Center.y = 5.0
Drawing.CreateCircle Center, Pos2, NewCircle
' Creates circle in Drawing Overlay
```

CreateText

Parameters

Position As XY, Text As String, T As Text

Description

Creates the text with content `Text` at `Position` in either Paper (active sheet) or Drawing Overlay. The object `T` referring to the created text is returned. The length of `Text` may be maximum 255 characters. A new line in the text is indicated with a CR (013x) or CRLF.

The style and attributes of the new created text correspond with the defaults set by the user.

Example

```
Dim A As Application
Dim Doc As Document
Dim Position As XY
Dim S As String
Dim T As Text
Dim Drawing As Overlay

Set A= New Application
A.GetActiveDocument Doc
Doc.GetOverlay ptDrawing, Drawing
Position.x = 10.0
Position.y = 10.0
S = "Line 1" & VbCrLf & "Line 2"
Drawing.CreateText Position, S, T ' Creates a text in the Drawing Overlay
```

CreateRichText

Parameters

`Position` As XY, `Height` As Double, `Width` As Double, `Text` As String, `RT` As RichText

Description

Creates the RichText object with content `Text` at `Position` in either Paper (active sheet) or Drawing Overlay. The object `RT` referring to the created rich text is returned. A new line in the text is indicated with `vbCrLf`.

The style and attributes of the new created rich text correspond with the defaults set by the user.

Example

```
Dim A As Application
Dim Doc As Document
Dim Position As XY
Dim S As String
Dim RT As RichText
Dim Drawing As Overlay

Set A= New Application
A.GetActiveDocument Doc
Doc.GetOverlay ptDrawing, Drawing
Position.x = 10.0
Position.y = 10.0
S = "Line 1" & VbCrLf & "Line 2"
Drawing.CreateRichText Position, 18., 50., S, RT ' Creates a text in the
' Drawing Overlay
```

CreateClothoid

Parameters

`P1` As XYZ, `P2` As XYZ, `RStart` As Double, `REnd` As Double, `K` As Double, `Mirror` As Boolean, `C1` As Clothoid

Description

Creates a clothoid (transition curve) in either Paper (active sheet) or Drawing Overlay. The object `C1` referring to the created clothoid is returned.

`P1` : Starting point

`P2` : End point

`RStart` : Radius at starting point

`REnd` : Radius at end point

`K` : K-Factor of the clothoid

`Mirror` : If False the clothoid is in counter-clockwise direction (`P1` - `P2`). If True clockwise.

If the radius of the starting or end point is infinite, the value `ptMaxDouble` must be passed.

The style and attributes of the new created clothoid correspond with the defaults set by the user.

Note : The parameters must be specified very accurately. If the resulting clothoid can not be calculated, run-time error 1056 will be generated.

Example

```
Dim A As Application
Dim Doc As Document
Dim P1 As XYZ, P2 As XYZ
Dim Cl As Clothoid
Dim Drawing As Overlay

Set A= New Application
A.GetActiveDocument Doc
Doc.GetOverlay ptDrawing, Drawing
P1.x = 0.0
P1.y = 0.0
P1.z = 0.0
P2.x = 63.348
P2.y = 6.777
P2.z = 0.0
Drawing.CreateClothoid P1, P2, ptMaxDouble, 100.0, 80.0, False, Cl
```

CreateHorizontalCurve

Parameters

Points() As XY, Z As Double, C As Curve

Description

Creates a polyline through **Points** with line segments with elevation **Z** in either Paper (active sheet) or Drawing Overlay. The object **C** referring to the created polyline is returned. The parameter **Points** must be a dynamic array. A fixed array is not accepted.

The style and attributes of the new created polyline correspond with the defaults set by the user.

Example

```
Dim A As Application
Dim Doc As Document
Dim Points() As XY ' Must be a Dynamic Array
Dim C As Curve
Dim Drawing As Overlay

Set A= New Application
A.GetActiveDocument Doc
Doc.GetOverlay ptDrawing, Drawing
Redim Points(0 To 3)
Points(0).x = 0.0
Points(0).y = 0.0
Points(1).x = 10.0
Points(1).y = 0.0
Points(2).x = 10.0
Points(2).y = 10.0
Points(3) = Points(0)
Drawing.CreateHorizontalCurve Points, 0.0, C ' Creates curve in Drawing Overlay
C.Color = ptRed
```

CreateCurve

Parameters

Points() As XYZ, C As Curve

Description

Creates a polyline through **Points** with line segments in either Paper (active sheet) or Drawing Overlay. The object **C** referring to the created polyline is returned. The parameter **Points** must be a dynamic array. A fixed array is not accepted.

The style and attributes of the new created polyline correspond with the defaults set by the user.

Example

```
Dim A As Application
Dim Doc As Document
Dim Points() As XYZ ' Must be a Dynamic Array
Dim C As Curve
Dim Drawing As Overlay
```

```

Set A = New Application
A.GetActiveDocument Doc
Doc.GetOverlay ptDrawing, Drawing
Redim Points(0 To 3)
Points(0).x = 0.0
Points(0).y = 0.0
Points(0).z = 1.0
Points(1).x = 10.0
Points(1).y = 0.0
Points(1).z = 2.0
Points(2).x = 10.0
Points(2).y = 10.0
Points(2).z = 3.0
Points(3) = Points(0)
Drawing.CreateCurve Points, C
C.Color = ptBlue

```

CreatePolygon

Parameters

Objs() As CadObject, Poly As Polygon

Description

Creates a polygon in either Paper (active sheet) or Drawing Overlay. The dynamic array Objs, must contain the elements that make up the polygon and should have the same dimension. (Use Redim). If the polygon contains only one element, this must be a circle. In all other cases, the first element of Objs must be a CadObject of type Point. The following elements can be points, arcs or clothoids. A polygon must be closed, this means the last element must end in the first element. When the last element is an arc or a clothoid, then the end point of that arc or clothoid must be the same as the first point of the polygon.

Example :

A - B : Line

B - C : Arc

C - D : Arc

D - E : Line

E - A : Arc

Then Objs must contain (Point A, Arc BC, Arc CD, Point E, Arc EA).

The object Poly referring to the created polygon is returned.

The style and attributes of the new created polygon correspond with the defaults set by the user.

Example

```

Sub PM_CreatePolygon

Dim A As Application
Dim Doc As Document
Dim Points() As XYZ           ' Must be a Dynamic Array
Dim Objs() As Point
Dim CadObjs() As CADObject
Dim Poly As Polygon
Dim Drawing As Overlay
Dim i As Integer
Dim PolyStyle As Style, PointStyle As Style

Set A= New Application
A.GetActiveDocument Doc
Doc.GetOverlay ptDrawing, Drawing
Redim Points(0 To 4)
Points(0).x = 0.0
Points(0).y = 0.0
Points(0).z = 0.0
Points(1).x = 10.0
Points(1).y = 0.0
Points(1).z = 0.0
Points(2).x = 10.0
Points(2).y = 10.0
Points(2).z = 0.0
Points(3).x = 0.0
Points(3).y = 10.0
Points(3).z = 0.0

```

```

Points(4) = Points(0)
Redim Objs(0 To 4)
Redim CadObjs(0 To 4)
PointStyle.Symbol = FALSE
PointStyle.Group = ""
PointStyle.item = ""
PointStyle.StyleId = ptHiddenPoint
For i = 0 To 4
    Drawing.CreatePoint Points(i), Objs(i)
    Objs(i).PointStyle = PointStyle
    Objs(i).GetCadObject CadObjs(i)
Next i

Drawing.CreatePolygon CadObjs, Poly
PolyStyle.Symbol = FALSE
PolyStyle.Group = ""
PolyStyle.item = ""
PolyStyle.StyleId = ptSolidFill
Poly.Pattern = PolyStyle
Poly.Color = 211

End Sub

```

CreatePath

Parameters

Objs() As CadObject, P As Path

Description

This method creates a path in either Paper (active sheet) or Drawing Overlay. The dynamic array *Objs*, must contain the elements that make up the path and should have the same dimension. (Use *Redim*). The rules regarding the type and sequence of elements in *Objs* is the same as for a polygon, except for the rule that the starting point and end point must be the same. The object *P* referring to the created path is returned.

Example

See example *CreatePolygon*.

CreatePathElement

Parameters

PathElem as PathElement, Elevation as Double, Obj as CadObj

Description

This method creates a cadobject either Paper (active sheet) or Drawing Overlay. The type of cadobject correspond with the *ElementType* of the *PathElement*. The elevation of the *PathElement* is overruled by the elevation parameter in this method.

Example

See example *CreatePathElement3D*.

CreatePathElement3D

Parameters

PathElem as PathElement, Obj as CadObj

Description

This method creates a cadobject either Paper (active sheet) or Drawing Overlay. The type of cadobject correspond with the *ElementType* of the *PathElement*.

Example

```

Dim A As Application
Dim Doc As Document
Dim P1 As XYZ
Dim P2 as XYZ
Dim PElem as PathElement
Dim PL as PathLine

Set A= New Application
A.GetActiveDocument Doc

```

```

Doc.GetOverlay ptDrawing, Drawing
Set GM = New GeoMath
P1.x = 0.0
P1.y = 0.0
P1.z = 0.0
P2.x = 10.0
P2.y = 0.0
P2.z = 0.0
Set PL = GM.NewPathLine(P1, P2)
PL.GetPathElement PElem
Drawing.CreatPathElement3D PElem, Obj
If Obj.ObjectType = ptLine Then
    MsgBox "Line is created"
End if

```

CreteImage

Parameters

Name As String, Pos As XY, ScaleDependent As Boolean, Scale As Double, MyImage As Image

Description

Creates a Pythagoras image object referring to an existing Pythagoras image file (.PTIF or .IMG-file). If such file does not exist, but an image file of world-known format exists (BMP, JPEG, TIFF, etc), than the Application method ConvertImage has to be used first in order to create a PTIF file.

Name = name of the Pythagoras image file.

Pos = the coordinates of the top-left corner of the image in the drawing.

ScaleDependent = must be set = TRUE when the size of the image, in paper dimensions, will vary with the scale of the drawing. (e.g. when importing multiple georeferenced images).

If the size of the image must remain fixed when the scale of the drawing is changed, the parameter ScaleDependent must be set = FALSE.

Scale = The scale of the raster file in case the parameter ScaleDependent = True.

MyImage = On return : the image Object.

Example

See example ConvertImage

InsertImage

Parameters

Name As String, Georeference As Boolean, Pos As XY, ScaleDependent As Boolean, Scale As Double, MyImage As Image

Description

Creates a Pythagoras image object enclosed (embedded) in the document.

Name = name of a BMP, TIFF, JPG or Pythagoras image file.

GeoReference = if True, the file should be either a GeoTiff or a World file (TFW, JGW, ...) should be present. If not, an error will be raised and no image is returned.

If an image is georeferenced, the parameters Pos, ScaleDependent, and Scale will not be used.

Pos = the coordinates of the top-left corner of the image in the drawing.

ScaleDependent = must be set = True when the size of the image, in paper dimensions, will vary with the scale of the drawing. (e.g. when importing multiple georeferenced images).

If the size of the image must remain fixed when the scale of the drawing is changed, the parameter ScaleDependent must be set = FALSE.

Scale = The scale of the raster file in case the parameter ScaleDependent = True.

MyImage = On return : the image Object.

Example

```

Dim A As Application
Dim Doc As Document
Dim I As Image
Dim Drawing As overlay
Dim Pos As XY

Set A = New Application
A.GetActiveDocument D
D.GetOverlay ptDrawing, Drawing
Drawing.InsertImage "C:\Test\Test.tif", True, Pos, True, Scale, I

```

CreateCompound

Parameters

Container As Compound

Description

Creates a Pythagoras compound object. The compound will contain no elements. Use the CadObject/Point/Line/... property Container to add one object at a time to the compound. If a compound is created in a macro, and no object is included into this compound then the compound can be silently deleted after the macro execution completion.

Example

```
Dim C As Compound
Dim O As Overlay
Dim P As Point

O.CreateCompound C
Set P.Container = C          ' Point is added to compound C
' See also example at start of chapter Compound
```

GetAllObjects

Parameters

All As AllObjects, Optional Sht As Sheet, Optional Wallpaper As Boolean = True

Description

Returns a set containing all objects in the overlay of a document. This set will always contain all objects in the overlay even if after calling GetAllObjects new objects are added or objects are deleted.

The optional parameter Sht is by default Nothing. If Sht <> Nothing AllObjects refers to all objects in the given Sheet. If the overlay is ptDrawing the parameter Sht must be omitted.

Alternatively the Sheet method GetAllObjects can be used.

The optional parameter Wallpaper defines whether objects of wallpaper layers and wallpaper subdocuments are included. If this parameter is True then objects of all layers of all visible subdocuments are included. Otherwise only objects of not wallpaper layers of both editable and snappable subdocuments are included.

Example

```
Dim A As Application
Dim Doc As Document
Dim PaperWorld As AllObjects
Dim PaperOverlay As Overlay
Dim Sht As Sheet
Dim Shts As Sheets

Set A = New Application
A.GetActiveDocument Doc
Doc.GetSheets Shts
Shts.GetActive Sht
Doc.GetOverlay ptPaper, PaperOverlay
PaperOverlay.GetAllObjects PaperWorld, Sht
PaperWorld.Color = ptRed
```

GetCoordinateSystems

Parameters

CSs As CoordinateSystems

Description

Gets the set of coordinate systems in the active overlay.

Example

```
Sub PM_GetCoordinateSystems

Dim A As Application
Dim D As Document
Dim O As Overlay
Dim CSs As CoordinateSystems

set A = New Application
A.GetActiveDocument D
```

```

If Not (D is Nothing) then
    D.GetOverlay ptDrawing, O
    O.GetCoordinateSystems CSs
    MsgBox "Nr Of CoordinateSystems: " & Str(CSs.NrOfCoordinateSystems)
end if
End sub

```

CreateTempCoordinateSystem

Parameters

Origin As XYZ, Orientation As XY, CS As CoordinateSystem

Description

Creates a Temporary Coordinate System in either the Paper or the Drawing Overlay. The object CS referring to the new created Coordinate System is returned.

Origin = the origin of the new temporary Coordinate System.

Orientation = the second point that gives the orientation of the new temporary Coordinate System.

Example

```

Sub PM_CreateTempCoordinateSystem

Dim A As Application
Dim D As Document
Dim CS As CoordinateSystem
Dim O As Overlay
Dim Orientation As XY
Dim Origin As XYZ

set A = New Application
A.GetActiveDocument D
If Not (D is Nothing) then
    D.GetOverlay ptDrawing, O
    Origin.x = 10.0
    Origin.y = 10.0
    Origin.z = 5.0
    Orientation.x = 20.0
    Orientation.y = 20.0
    O.CreateTempCoordinateSystem Origin, Orientation, CS
    CS.SetActive
end if
End sub

```

CreateUserCoordinateSystem

Parameters

Origin As Point, Orientation As Point, CS As CoordinateSystem

Description

Creates a User Coordinate System in either the Paper or the Drawing Overlay. The object CS referring to the new created Coordinate System is returned.

Origin = the origin of the new User Coordinate System

Orientation = the second point that gives the orientation of the new User Coordinate System

Example

```

Dim O As Overlay
Dim CS As CoordinateSystem
Dim Origin As Point
Dim Orientation As Point

O.CreateUserCoordinateSystem Origin, Orientation, CS

```

DeleteCoordinateSystem

Parameters

CS As CoordinateSystem

Description

Deletes a Coordinate System in either the Paper or the Drawing Overlay. The object CS refers to the Coordinate System that will be deleted. Only temporary and user Coordinate Systems can be deleted.

Example

```
Dim O As Overlay
Dim CS As CoordinateSystem

O.DeleteCoordinateSystem CS
```

Clip

Parameters

```
Poly As Polygon, RemoveOutside As Boolean, Types As Long, _
RemoveImagesOnTheOutside As Boolean
```

Description

Will remove all objects of type in *Types* which are inside or outside the polygon *Poly*.

RemoveOutside: if *True*, removes the objects outside the polygon. If *False*, removes the objects inside the polygon.

Types: is the sum of the combination of one or more of the following constants:

ptPoint, *ptLine*, *ptArc*, *ptClothoid*, *ptText*, *ptRichText*, *ptImage*,
ptCurve, *ptPath*, *ptPolygon*, *ptCircle*, *ptCoordinateSystem*.

RemoveImagesOnTheOutside: if *True*, images completely outside the polygon will be removed if *ptImage* in *Types*.

Protected objects will not be clipped.

Invisible objects will be clipped.

Due to technical reasons, the *Clip* method automatically inserts an extra state.

Example

```
Dim A As Application
Dim Doc As Document
Dim Sel As Selection
Dim DrawingOverlay As Overlay
Dim Obj As CadObject
Dim Poly As Polygon

Set A = New Application
A.GetActiveDocument Doc
Doc.GetOverlay ptDrawing, DrawingOverlay
Doc.GetSelection Sel
If Sel.Count(ptPolygon) = 1 Then
    Sel.GetFirst ptPolygon, Obj
    Obj.GetPolygon Poly
    DrawingOverlay.Clip Poly, True, ptAll, False
End If
```

XYToPoint

Parameters

```
Position As XY
```

Returns

```
P As Point
```

Description

Returns an existing point that has the requested coordinates. If no point exists with these coordinates, Nothing is returned.

This method is only supported for drawing overlay (*ptDrawing*).

Example

```
Dim A As Application
Dim Doc As Document
Dim Pos As XY
Dim P As Point
Dim Drawing As Overlay

Set A = New Application
A.GetActiveDocument Doc
Doc.GetOverlay ptDrawing, Drawing
Pos.x = 10.0
Pos.y = 10.0
Set P = Drawing.XYToPoint(Pos)
```


Chapter 16 - CadObject

Description

A CadObject object is a generic object type that encompasses the object types Point, Line, Arc, Circle, Clothoid, Curve, Text, RichText, Image, Polygon, Path, Road, Compound and CoordinateSystem. Several properties and methods accept parameters of type CadObject, e.g. the Document method GetObjectUnderCursor. It is obvious that some functions can only return a generic object type.

When the type of a CadObject is known, the CadObject can be converted to its specific object type. The methods GetPoint, GetLine, etc. do not create a new object, but they assign the reference to the object of a specific type. Each CadObject of type ptCoordinateSystem can be converted to a CoordinateSystem, but the inverse is not always the case. The coordinate systems ptLocal, ptPage, ptGlobal and ptTemp can not be converted to a CadObject.

Example

```
Dim A As Application
Dim Doc As Document
Dim Obj As CadObject
Dim P As Point
Dim Z As Double

Set A = New Application
A.GetActiveDocument Doc
Doc.GetObjectUnderCursor ptPoint, Obj ' Returns generic object type Obj
If not (Obj is nothing) Then
    If Obj.ObjectType = ptPoint Then ' Converts generic object type Obj
        Obj.GetPoint P ' to a specific object type P

        If P.Elevation > 0.0 Then
            ' ...
        End If
    End If
End If
```

Properties

ObjectType
Color
TrueColor
Comment
Link
DisplayLevel
GroupId
Container
Layer
Overlay
CanBeDeleted
CanChangeOverlay
ObjectReference
Selected
Flagged
Data
Sheet

All of the above properties but ObjectType are also properties of each specific object type.

Methods

GetPoint
GetLine
GetArc
GetCircle
GetClothoid
GetCurve
GetText
GetRichText
GetPolygon
GetSolid
GetImage
GetPath

```

GetCoordinateSystem
GetCompound
GetAnnotation
GetDocument
GetSubDocument
GetReferencingObjects
SetData
DeleteData
ConvertToPolyline
SetLink
GetQueryData

```

The methods `GetDocument` and `GetSubDocument` are also methods of each specific object type.

Properties

ObjectType

Returns

Long

Description

This property returns the specific type of the object. Valid return values : `ptPoint`, `ptLine`, `ptArc`, `ptClothoid`, `ptText`, `ptRichText`, `ptImage`, `ptCurve`, `ptPath`, `ptPolygon`, `ptCircle`, `ptCoordinateSystem`, `ptRoad`, `ptCompound`, `ptAnnotation`, `pt3DSolid`

Example

```

Dim Obj As CadObject
Dim P As Point

If Obj.ObjectType = ptPoint Then
    Obj.GetPoint P
End If

```

Comment

Returns

String

Description

This property sets or returns the comment (object information) of the `CadObject`. Setting the property for compound sets the specified comment for all its elements.

Example

```

Dim Obj1 As CadObject, Obj2 As CadObject

Obj2.Comment = Obj1.Comment

```

Link

Returns

String

Description

This property sets or returns the external link of the `CadObject`. For setting this property it is recommended to use the method `SetLink`.

The name of the link may be a full or partial path name. If a partial name, the name is relative either to the Folder External File or the the directory of the document containing the link.

Pythagoras will detect if the link is a file, a folder or a URL. Pythagoras does not verify if the file, folder or URL exists.

Example

```

Dim Obj As CadObject

Obj.Link = "MyPicture.jpg"
Obj.SetLine "MyPicture.jpg", False ' Better use this rather than Obj.Link = ...

```

Color

Returns

Integer

Description

This property sets or returns the color of the CadObject. The color is a sequence number in the standard palette of 256 colors. When the true color of an object is not in the palette, then the nearest color is returned.

When applying the property to object types which have no color, then an error is raised. Applying the property on an Annotation will set both the color of the text and the symbol and will return the color of the text of the annotation.

Setting the property for compound sets the specified color for all its elements where color makes sense.

Example

```
Dim Obj1 As CadObject, Obj2 As CadObject

Obj2.Color = Obj1.Color
```

TrueColor

Returns

RGB

Description

This property sets or returns the color (RGB) of the CadObject. When applying the property to object types which have no color, then an error is raised. This property may not be applied to a rich text. Applying the property on Annotation will set or return the color of the text of the annotation.

Setting the property for compound sets the specified color for all its elements where color makes sense.

Example

```
Dim Obj1 As CadObject
Dim RGBColor As RGB

RGBColor.Red = 120
RGBColor.Green = 250
RGBColor.Blue = 65
Obj1.TrueColor = RGBColor
```

DisplayLevel

Returns

Integer

Description

This property sets or returns the display level of the CadObject.

Setting the property for compound sets the specified display level for all its elements.

Example

```
Dim Obj1 As CadObject, Obj2 As CadObject

Obj2.DisplayLevel = Obj1.DisplayLevel
```

GroupId

Returns

Integer

Description

This property sets or returns the Group to which the CadObject belongs. Because the method NewGroupId returns a free GroupId of the main document without taking into account subdocuments, care must be taken when working with subdocuments. Two objects that lie in a different subdocument could have the same GroupId even when intentionally they don't belong to the same group.

All CADObjects must have the same overlay and belong to the same subdocument. If CADObjects are in ptPaper, then they must be in the same sheet.

This property is obsolete. Use the property Container in stead.

Example

```
Dim Obj1 As CadObject, Obj2 As CadObject

If Obj1.GroupId <> ptNoGroup Then
```

```

Obj2.GroupId = Obj1.GroupId
End If

```

Container

Returns

Compound

Description

This property sets or returns the Compound object containing this CadObject. If applied to a Compound object it returns Nothing. When the CadObject is not an element of a compound, the property returns Nothing.

When assigning a Compound to a CadObject, the CadObject will become a member of the Compound. Assigning to Nothing will exclude the CadObject from the compound it belonged to.

Example

```

Dim Obj1 As CadObject, Obj2 As CadObject

If Obj1.Container Is Obj2.Container Then
    If Obj1.Container Is Nothing Then
        ' Obj1 and Obj2 belong to the same compound
        Set Obj2.Container = Nothing
    End If
End If

```

Layer

Returns

String

Description

This property sets or returns the Layer of the object.

IfLayer = "~SYSTEM~" (name of the System Layer), then the object, when it's a point, arc or clothoid, will be put in the System Layer. Trying to put other objects in the System Layer will have no effect.

For more information about System Layer see chapter Layers.

Setting the property for compound moves all its elements to the specified layer/

Example

```

Dim Obj1 As CadObject, Obj2 As CadObject

Obj2.Layer = Obj1.Layer

```

Overlay

Returns

Integer

Description

This property returns the Overlay (ptDrawing or ptPaper) of the object.

Example

```

Dim Obj As CadObject
Dim O As Overlay

If Obj.Overlay = ptPaper Then
    'Call Sub(parameters)
End If

```

CanBeDeleted

Returns

Boolean

Description

This property returns True if the object can be deleted. In certain cases, e.g. layer of object is protected, an object can not be deleted.

Example

```

Dim Obj As CadObject

If Obj.CanBeDeleted Then

```

```

        Obj.Flagged = True
    End If

```

CanChangeOverlay

Returns

Boolean

Description

This property returns True if the overlay of the object can be changed. The overlay of an individual object can only be changed if the object does not refer to another object, and if it not referred to by another object. The property cannot be applied to a compound.

Example

```

Dim Obj As CadObject

If Obj.CanChangeOverlay Then
    Obj.Flagged = True
End If

```

ObjectReference

Returns

Long

Description

This property returns the Object Reference of the Object. Each CadObject of a document has a unique Object Reference. The Object Reference remains unchanged over the lifetime of a document. A database could save Object References to find specific CadObject of a document.

See also the Document method `ObjectReferenceToCadObject`.

When a document contains subdocuments, the object reference in the respective subdocument is returned. This means that Object References in the set of subdocuments are not unique. As a consequence, calling `ObjectReferenceToCadObject` using the result of this call could return another CadObject.

Example

```

Dim Obj As CadObject
Dim ObjRef As Long

ObjRef = Obj.ObjectReference

```

Selected

Returns

Boolean

Description

This property sets an object selected or returns the selection status of the object. Setting the property for compounds does not change the flag values for its elements. To change the flag values for all compound elements, use property `Compound.AllSelected`.

Example

```

Dim Obj As CadObject

Obj.Selected = True

```

Flagged

Returns

Boolean

Description

This property sets an object Flagged or returns the the status of the Flagged bit of the object. Setting the property for compounds does not change values of Flagged bits for its elements. To change flag values for all compound elements, use property `Compound.AllFlagged`.

For more information about the Flag, see the object type `FlaggedObjects`.

Example

```

Dim Obj As CadObject

Obj.Flagged = True

```

Data

Returns

DBRow

Description

This property returns a DBRowobject if the CAD object is linked to a table.

Returns Nothing if the CAD object is not linked to a database table.

If the CADObject is an element of a Compound, the data linked to the Compound is returned. Elements of a Compound do not have own data.

Example

```
Dim Obj As CadObject
Dim Row As DBRow

Set Row = Obj.Data
```

Sheet

Returns

Sheet

Description

This property returns a Sheet object if the CAD object is in paper overlay.

Returns Nothing if the CAD object is in overlay ptDrawing.

Example

```
Dim Obj As CadObject
Dim S As Sheet

S = Obj.Sheet
```

Methods

CadObjects can not be assigned to objects of a specific type.

So the statement :

```
Set P = Obj
```

is not valid. Instead, the statements GetPoint, GetLine, etc. need to be used.

GetPoint

Parameters

P As Point

Description

GetPoint sets the reference to the object in a variable of type Point. Note that the reference is copied, not the object.

Example

```
Dim Obj As CadObject
Dim P As Point

Obj.GetPoint P           ' Same as : Set P = Obj
```

GetLine

Parameters

L As Line

Description

GetLine sets the reference to the object in a variable of type Line. Note that the reference is copied, not the object.

Example

```
Dim Obj As CadObject
Dim L As Line

Obj.GetLine L           ' Same as : Set L = Obj
```

GetArc

Parameters

A As Arc

Description

GetArc sets the reference to the object in a variable of type Arc. Note that the reference is copied, not the object.

Example

```
Dim Obj As CadObject
Dim A As Arc
```

```
Obj.GetArc A           ' Same as : Set A = Obj
```

GetCircle

Parameters

C As Circle

Description

GetCircle sets the reference to the object in a variable of type Circle. Note that the reference is copied, not the object.

Example

```
Dim Obj As CadObject
Dim C As Circle
```

```
Obj.GetCircle C        ' Same as : Set C = Obj
```

GetClothoid

Parameters

C As Clothoid

Description

GetClothoid sets the reference to the object in a variable of type Clothoid. Note that the reference is copied, not the object.

Example

```
Dim Obj As CadObject
Dim C As Clothoid
```

```
Obj.GetClothoid C      ' Same as : Set c = Obj
```

GetCurve

Parameters

A As Curve

Description

GetCurve sets the reference to the object in a variable of type Curve. Note that the reference is copied, not the object.

Example

```
Dim Obj As CadObject
Dim C As Curve
```

```
Obj.GetCurve C         ' Same as : Set C = Obj
```

GetText

Parameters

T As Text

Description

GetText sets the reference to the object in a variable of type Text. Note that the reference is copied, not the object.

Example

```
Dim Obj As CadObject
Dim T As Text
```



```
Obj.GetText T ' Same as : Set T = Obj
```

GetRichText

Parameters

```
RT As RichText
```

Description

GetRichText sets the reference to the object in a variable of type RichText. Note that the reference is copied, not the object.

Example

```
Dim Obj As CadObject  
Dim RT As RichText
```

```
Obj.GetRichText RT ' Same as : Set RT = Obj
```

GetPolygon

Parameters

```
P As Polygon
```

Description

GetPolygon sets the reference to the object in a variable of type Polygon. Note that the reference is copied, not the object.

Example

```
Dim Obj As CadObject  
Dim P As Polygon
```

```
Obj.GetPolygon P ' Same as : Set P = Obj
```

GetSolid

Parameters

```
S As Solid
```

Description

GetSolid sets the reference to the object in a variable of type Solid. Note that the reference is copied, not the object.

Example

```
Dim Obj As CadObject  
Dim S As Solid
```

```
Obj.GetSolid S ' Same as : Set S = Obj
```

GetImage

Parameters

```
I As Image
```

Description

GetImage sets the reference to the object in a variable of type Image. Note that the reference is copied, not the object.

Example

```
Dim Obj As CadObject  
Dim I As Image
```

```
Obj.GetImage I ' Same as : Set I = Obj
```

GetPath

Parameters

```
P As Path
```

Description

GetPath sets the reference to the object in a variable of type Path. Note that the reference is copied, not the object.

Example

```
Dim Obj As CadObject
Dim P As Path

Obj.GetPath P           ' Same as : Set P = Obj
```

GetCoordinateSystem

Parameters

CS As CoordinateSystem

Description

GetCoordinateSystem returns an object of type CoordinateSystem that corresponds with the CadObject. Note that a Coordinate System is not a CadObject.

Example

```
Dim Obj As CadObject
Dim CS As CoordinateSystem

Obj.GetCoordinateSystem CS
```

GetCompound

Parameters

Container As Compound

Description

GetCompound returns an object of type Compound that corresponds with the CadObject. Note that the reference is copied, not the object.

Example

```
Dim Obj As CadObject
Dim C As Compound

Obj.GetCompound C
```

GetAnnotation

Parameters

An As Annotation

Description

GetAnnotation returns an object of type Annotation that corresponds with the CadObject. Note that the reference is copied, not the object.

Example

```
Dim Obj As CadObject
Dim An As Annotation

Obj.GetAnnotation An
```

GetDocument

Parameters

Doc As Document

Description

GetDocument returns the document that contains the CadObject.

Example

```
Dim Obj As CadObject
Dim Doc As Document

Obj.GetDocument Doc
```

GetSubDocument

Parameters

SubDoc As SubDocument

Description

GetSubDocument returns the Subdocument that contains the CadObject.

Example

```
Dim Obj As CadObject
Dim SubDoc As SubDocument

Obj.GetSubDocument SubDoc
```

GetReferencingObjects

Parameters

Count As Integer, RefObjs() As CadObject

Description

GetReferencingObjects returns a list of all the objects that are referencing the CadObject.

Referencing objects can be lines, arcs, polygons and paths. For a example a point can referencie one or more lines. It does not work the other way around: you cannot find the start- and endpoint of a line.

Example

```
Dim Obj As CadObject
Dim Count As Integer
Dim RefObjs() As CadObject

Obj.GetReferencingObjects Count, RefObjs
If Count > 0 Then
    If RefObjs(1).ObjType = ptLine Then
        ' ...
    End If
Else
    ' no objects referencing to this Object
End If
```

SetData

Parameters

Table As DBTable, Data() As ColumnValue

Description

SetData inserts a new row into the table and links the row with the Cad Object.

Table must refer to a database attribute table that is in the same document as the CAD Object.

Data: all columns that are contained in Data are used to initialise the corresponding columns in the table.

Other columns will get the default values or remain empty. See also DBTable AddColumn.

If the CAD object is already linked to the given table, then the method does the same as the DBRow class

Update method. When the CAD object is already linked to another table, an error will be raised.

If the CAD object is an element of a Compound, SetData is applied to the compound. Elements of a Compound do not have own data.

Example

```
Dim Obj As CadObject
Dim D As Document
Dim T As DBTable
Dim Data() As ColumnValue

Set T = D.TableByName("Countries")
Obj.SetData T, Data
```

DeleteData

Parameters

None

Description

If an object has data, DeleteData deletes the linked row from the table.

If the CAD object is an element of a Compound, the row linked to the compound is deleted. Elements of a Compound do not have own data.

Example

```
Dim Obj As CadObject

Obj.DeleteData
```

ConvertToPolyline

Parameters

NrOfPoints As Long, Points() As XYZ

Description

Converts a CadObject that represents an arc, a clothoid, a circle, a path, or a polygon to a polyline. Coordinates and the number of polyline points are specified by output arguments Points() and NrOfPoints.

Example

```
Dim P As Polygon
Dim Pnts() As XYZ
Dim N As Long

P.ConvertToPolylineN, Pnts
```

SetLink

Parameters

Link As String, URL_Link As Boolean

Description

Sets the external link of the CadObject.

The name of the link may be a full or partial path name. If a partial name, the name is relative either to the Folder External File or the directory of the document containing the link. Pythagoras does not verify if the file or URL exists.

Example

```
Dim Obj As CadObject

Obj.SetLink "MyPicture.jpg", False ' Better use this rather than Obj.Link = ...
```

GetQueryData

Parameters

Query As DBQuery

Returns

DBRow

Description

If Query is valid and CAD object linked (see class DBQuery corresponding properties) and contains a row linked to the object, this row is returned. Otherwise Nothing is returned.

Example

```
Dim Obj As CadObject
Dim Q As Query
Dim Row As DBRow

Set Row = Obj.GetQueryData (Q)
```

CreateAnnotation

Parameters

AnKind As AnnotationKind, An As Annotation,
Optional ElemType As ElementType = ptObject,
Optional Index As Integer = 0

Description

Creates an annotation of AnKind for the CadObject. The object An referring to the created annotation is returned. The style and attributes of the new created annotation correspond with the defaults set by the user.

The given AnKind can be one of the following:

```
AnKind = ptAn_TotalLength (= 1)
AnKind = ptAn_Coordinate (= 2)
AnKind = ptAn_Elevation (= 3)
AnKind = ptAn_PointId (= 4)
AnKind = ptAn_Area (= 5)
AnKind = ptAn_LineLength (= 6)
```

AnKind = ptAn_Comment (= 7)
 AnKind = ptAn_Bearing (= 8)
 AnKind = ptAn_SlopeDistance (= 9)
 AnKind = ptAn_Dz (= 10)
 AnKind = ptAn_Gradient (= 11)
 AnKind = ptAn_Angle (= 12)
 AnKind = ptAn_Radius (= 13)
 AnKind = ptAn_DeltaX (= 14)
 AnKind = ptAn_DeltaY (= 15)
 AnKind = ptAn_Volume (= 16)

If the CadObject can't have an annotation or cannot have the given kind of annotation an error is raised.

The optional AnSpec can be one of the following:

ElemType = ptObject (= 0)
 ElemType = ptVertex (= 1)
 ElemType = ptSegment (= 2)

For now ptVertex and ptSegment are only possible on curve objects, which will add an annotation on the Indexth vertex or segment respectively, starting from 0.

If the combination of kind, specifier and index is not possible an error is raised.

The following combinations are possible:

CadObject type	AnnotationKind	ElemType	Extra info
Point	ptObject	ptAn_Coordinate, ptAn_Elevation, ptAn_PointId, ptAn_Comment	
Line	ptObject	ptAn_TotalLength, ptAn_LineLength, ptAn_Comment, ptAn_Bearing, ptAn_SlopeDistance, ptAn_Dz, ptAn_Gradient, ptAn_DeltaX, ptAn_DeltaY	
Arc	ptObject	ptAn_TotalLength, ptAn_LineLength, ptAn_Comment, ptAn_SlopeDistance, ptAn_Dz, ptAn_Gradient, ptAn_Angle, ptAn_Radius	
Circle	ptObject	ptAn_Radius, ptAn_Comment	
Curve	ptObject	ptAn_TotalLength, ptAn_Elevation	ptAn_Elevation is only possible on horizontal curve objects
	ptVertex	ptAn_Coordinate, ptAn_Elevation	
	ptSegment	ptAn_TotalLength, ptAn_LineLength, ptAn_Comment, ptAn_Bearing, ptAn_SlopeDistance, ptAn_Dz, ptAn_Gradient, ptAn_Angle, ptAn_Radius, ptAn_DeltaX, ptAn_DeltaY	ptAn_DeltaX and ptAn_DeltaY are only possible on line segments. ptAn_Angle and ptAn_Radius are only possible on arc segments.

Clothoid	ptObject	ptAn_TotalLength, ptAn_LineLength, ptAn_Comment, ptAn_SlopeDistance, ptAn_Dz, ptAn_Gradient	
Path	ptObject	ptAn_TotalLength, ptAn_Comment	
Polygon	ptObject	ptAn_Area, ptAn_Comment	An annotation is created for each element of the polygon. The object An contains the last created annotation.
		ptAn_LineLength, ptAn_Bearing, ptAn_SlopeDistance, ptAn_Dz, ptAn_Gradient	
	ptSegment	ptAn_LineLength, ptAn_Bearing, ptAn_SlopeDistance, ptAn_Dz, ptAn_Gradient	
Road	ptObject	ptAn_LineLength, ptAn_Comment, ptAn_Dz, ptAn_Gradient	
Solid	ptObject	ptAn_Volume, ptAn_Comment	

Example

```

Dim A As Application
Dim Doc As Document
Dim Position As XYZ
Dim NewPoint As Point
Dim Drawing As Overlay
Dim Obj As CadObject
Dim An As Annotation

Set A= New Application
A.GetActiveDocument Doc
Doc.GetOverlay ptDrawing, Drawing
Position.X = 10.0
Position.Y = 25.2
Position.Z = 10.0
Drawing.CreatePoint Position, NewPoint      ' Creates point in Drawing Overlay
NewPoint.GetCadObject Obj
Obj.CreateAnnotation ptAn_Coordinate, An
An.Color = ptRed

```

Chapter 17 - Point

Description

A Point is a CadObject. A new Point object can be created with the Overlay method CreatePoint. A reference to an existing Pythagoras Point can be obtained with e.g. the method GetPoint.

Example

```
Dim A As Application
Dim Doc As Document
Dim Pnt As XYZ
Dim P As Point
Dim Drawing As Overlay

Set A = New Application
A.GetActiveDocument Doc
Doc.GetOverlay ptDrawing, Drawing
Drawing.CreatePoint Pnt, P
```

Properties

Color
Comment
Link
DisplayLevel
GroupId
Layer
Overlay
CanBeDeleted
CanChangeOverlay
ObjectReference
Selected
Flagged
Container
Data
PointIdent
Coordinates
Elevation
ZoomFactor
PointStyle

Methods

GetDocument
GetSubDocument
SetData
DeleteData
GetCadObject
GetOrientation
SetOrientation
GetScaleFactor
SetScaleFactor

Properties

For a description of the properties and methods Color, TrueColor, Comment, Link, DisplayLevel, GroupId, Layer, Overlay, CanBeDeleted, CanChangeOverlay, ObjectReference, Selected, Flagged, Container, Data, GetQueryData see object type CadObject.

PointIdent

Returns

String

Description

This property sets or returns the point number.

Example

```
Dim P As Point
```

```
Dim O As Overlay
Dim Pnt As XYZ

O.CreatePoint Pnt, P
P.PointIdent = "1"
```

Coordinates

Returns
XYZ

Description

This property returns the coordinates of the point. For changing the coordinates, the method `Move` (see `FlaggedObjects`) must be used.

Example

```
Dim P As Point
Dim O As Overlay
Dim Pnt As XYZ

O.CreatePoint Pnt, P
Pnt = P.Coordinates
```

Elevation

Returns
Double

Description

This property sets or returns the elevation of the point.

Example

```
Dim P As Point
Dim O As Overlay
Dim Pnt As XYZ

O.CreatePoint Pnt, P
P.Elevation = 2.0
```

ZoomFactor

Returns
Double

Description

This property sets or returns the Zoomfactor of a symbol.

Example

```
Dim P As Point
Dim O As Overlay
Dim Pnt As XYZ

O.CreatePoint Pnt, P
P.ZoomFactor = 0.5
```

PointStyle

Returns
Style

Description

This property sets or returns the style of the point.

Example

```
Dim P As Point, P2 As Point
Dim S As Style, S2 As Style

S.Symbol = False
S.Group = ""
S.item = ""
S.StyleId = ptHiddenPoint
P.PointStyle = S
```



```

S2.Symbol = True
S2.Group = "Examples"
S2.Item = "manhole"
P2.PointStyle = S2

```

Methods

CadObjects can not be assigned to objects of a specific type.

So the statement :

```
Set P = Obj
```

is not valid. In stead, the statements GetPoint, GetLine, etc. need to be used.

For a description of the methods GetDocument, GetSubDocument, SetData, DeleteData see object type CadObject.

GetCadObject

Parameters

```
Obj As CadObject
```

Description

GetCadObject sets the reference to the Point in a variable of type CadObject. Note that the reference is copied, not the object.

Example

```

Dim Obj As CadObject
Dim P As Point

P.GetCadObject Obj          ' Same as : Set Obj = P

```

GetOrientation

Parameters

```
Oblique As Boolean, Angle As Double
```

Description

GetOrientation returns the angle of the point if the parameter Oblique returns True. The variable Oblique returns False if the symbol representing the point remains horizontal. In that case the returned angle is also equal to zero.

Example

```

Dim P As Point
Dim Oblique As Boolean
Dim Angle As Double

P.GetOrientation Oblique, Angle

```

SetOrientation

Parameters

```
Oblique As Boolean, Angle As Double
```

Description

SetOrientation set the angle of the point if the point is oblique. The variable Oblique must be set False if the symbol representing the point must remain horizontal. In that case the value of the parameter angle is ignored.

Example

```

Const pi = 3.1416
Dim P As Point

P.SetOrientation True, 90.0/360.0*pi

```

GetScaleFactor

Parameters

```
ScaleX As Double, ScaleY As Double
```

Description

GetScaleFactor returns both the factors (X and Y) by which the symbol is scaled. The scale factors are expressed in %. One or both scale factors may be negative.

Example

```
Dim P As Point
Dim ScaleX As Double, ScaleY As Double

P.GetScaleFactor ScaleX, ScaleY
```

SetScaleFactor

Parameters

ScaleX As Double, ScaleY As Double

Description

SetScaleFactor sets the new absolute scale factors (X and Y) of the symbol. So the old scale factors have no effect on the result. The scale factors are expressed in %. A scale factor may be negative. This means that the symbol is mirrored around the corresponding axis.

Example

```
Dim P As Point

P.SetScaleFactor -100.0, 100.0      ' Set new scale factors to -100%, 100%
```

Chapter 18 - Line

Description

A Line is a CadObject. A new Line object can be created with the Overlay method CreateLine. A reference to an existing Pythagoras Line can be obtained e.g. with the method GetLine. Since a line defined by VBA is a Pythagoras line, it is defined by 2 Point Objects.

Example

```
Dim A As Application
Dim Doc As Document
Dim Pnt1 As XYZ, Pnt2 As XYZ
Dim P1 As Point, P2 As Point
Dim L As Line
Dim Drawing As Overlay

Set A = New Application
A.GetActiveDocument Doc
Doc.GetOverlay ptDrawing, Drawing
Drawing.CreatePoint Pnt1, P1
Drawing.CreatePoint Pnt2, P2
Drawing.CreateLine P1, P2, L
```

Properties

Color
Comment
Link
DisplayLevel
GroupId
Layer
Overlay
CanBeDeleted
CanChangeOverlay
ObjectReference
Selected
Flagged
Container
Data
LineStyle
Scale
Width
Arrows
LineLength
Bearing
DeltaZ
SlopeLength

Methods

GetDocument
GetSubDocument
SetData
DeleteData
GetCadObject
GetBeginPoint
GetEndPoint
CDZtoXYZ
XYZtoCDZ
Invert

Properties

For a description of the properties and methods Color, TrueColor, Comment, Link, DisplayLevel, GroupId, Layer, Overlay, CanBeDeleted, CanChangeOverlay, ObjectReference, Selected, Flagged, Container, Data, GetQueryData see object type CadObject.

LineStyle

Returns
Style

Description
This property sets or returns the style of the line.

Example

```
Dim L1 As Line, L2 As Line
Dim S As Style, S2 As Style

S.Symbol = True
S.Group = "Example"
S.item = "Wall"
L1.LineStyle = S
S2 = L1.LineStyle
L2.LineStyle = S2
```

Scale

Returns
Double

Description
This property sets or returns the scale of the line style. When setting, and the new scale is negative, then the scale of the line will become, or remain negative. Scaling is only applicable to scalable line styles.

Example

```
Dim L1 As Line, L2 As Line

L1.Scale = 75.0
L2.Scale = L1.Scale
```

Width

Returns
Double

Description
This property sets or returns the width of the line. The line width must be between 0.1 (0.01 mm) and 25.5 (2.55 mm).

Example

```
Dim L1 As Line, L2 As Line

L1.Width = 5.0
L2.Width = L1.Width
```

Arrows

Returns
Integer

Description
This property sets arrows at the end of the line, or returns a value indicating which ends of the line has arrows. The arrows parameter has the following valid values : ptNoArrows, ptArrowP1, ptArrowP2, ptArrowP1P2.

Example

```
Dim L1 As Line, L2 As Line

L1.Arrows = ptArrowP1P2
L2.Arrows = L1.Arrows
```

LineLength

Returns
Double

Description
This property returns the horizontal length of the line.

Example

```
Dim L As Line

If L.LineLength > 100.0 Then
End If
```

Bearing

Returns

Double

Description

This property returns the bearing of the line. The bearing is the angle of the line relative to the north axis and clockwise.

Example

```
Dim L As Line
Dim Bearing As Double

Bearing = L.Bearing
```

DeltaZ

Returns

Double

Description

This property returns the difference in height between the starting point and the end point of the line.

Example

```
Dim L As Line
Dim dZ As Double

dZ = L.DeltaZ
```

SlopeLength

Returns

Double

Description

This property returns the absolute length (3D) of the line.

Example

```
Dim L As Line
Dim Slope As Double

Slope = L.SlopeLength
```

Methods

CadObjects can not be assigned to objects of a specific type.

So the statement :

```
Set L = Obj
```

is not valid. In stead, the statements GetPoint, GetLine, etc. need to be used.

For a description of the methods GetDocument, GetSubDocument, SetData, DeleteData see object type CadObject.

GetCadObject

Parameters

Obj As CadObject

Description

GetCadObject sets the reference to the Line in a variable of type CadObject. Note that the reference is copied, not the object.

Example

```
Dim Obj As CadObject
Dim L As Line
```

L.GetCadObject Obj ' Same as : Set Obj = L

GetBeginPoint

Parameters

BeginPoint As Point

Description

GetBeginPoint returns the first point of the line..

Example

```
Dim P As Point
Dim L As Line

L.GetBeginPoint P
```

GetEndPoint

Parameters

EndPoint As Point

Description

GetEndPoint returns the second point of the line.

Example

```
Dim P As Point
Dim L As Line

L.GetEndPoint P
```

CDZtoXYZ

Parameters

Chainage As Double, Distance As Double, DeltaZ As Double, _
Pnt As XY, Z As Double

Description

CDZtoXYZ converts the values Chainage, Distance and DeltaZ, to coordinates. On entry DeltaZ is the elevation relative to the line.

Example

See example for Path

XYZtoCDZ

Parameters

Pnt As XYZ, Chainage As Double, Distance As Double, DeltaZ As Double

Description

XYZtoCDZ converts the coordinates of a point to the values Chainage, Distance and DeltaZ. On return DeltaZ will be the elevation relative to the projected point the line.

Example

See CDZtoXYZ.

Invert

Parameters

Pnt As XYZ, Chainage As Double, Distance As Double, DeltaZ As Double

Description

The method swaps begin and end points of the line. In most cases the order of points is irrelevant, however see example below.

Example

Style Arrows shows direction of lower altitude (ie waterflow).

```
Dim L As Line, Arrows As Style

If L.DeltaZ > 0 Then
```

```
        L.Invert  
End If  
L.LineStyle = Arrows
```

Chapter 19 - Arc

Description

An Arc is a CadObject. A new Arc can be created with the Overlay method CreateArc. A reference to an existing Pythagoras Arc can be obtained e.g. with the method GetArc.

Example

See method CreateArc of the object type Overlay.

Properties

Color
Comment
Link
DisplayLevel
GroupId
Layer
Overlay
CanBeDeleted
CanChangeOverlay
ObjectReference
Selected
Flagged
Container
Data
LineStyle
Scale
Width
Arrows
ArcLength
Radius
DeltaZ
SlopeLength
ArcAperture
Center
Center3D

Methods

GetDocument
GetSubDocument
SetData
DeleteData
GetCadObject
GetBeginPoint
GetEndPoint
CDZtoXYZ
XYZtoCDZ

Properties

For a description of the properties and methods Color, TrueColor, Comment, Link, DisplayLevel, GroupId, Layer, Overlay, CanBeDeleted, CanChangeOverlay, ObjectReference, Selected, Flagged, Data, GetQueryData see object type CadObject.

For a description of the properties Style, Scale, Width, Arrows, and of the methods GetBeginPoint, GetEndPoint, Container, see object type Line.

ArcLength

Returns

Double

Description

This property returns the horizontal length of the arc.

Example

```
Dim MyArc As Arc

If MyArc.ArcLength > 100.0 Then
End If
```


Radius

Returns

Double

Description

This property returns the radius of the arc.

Example

```
Dim MyArc As Arc
Dim Radius As Double

Radius = MyArc.Radius
```

DeltaZ

Returns

Double

Description

This property returns the difference in height between the starting point and the end point of the arc.

Example

```
Dim MyArc As Line
Dim dZ As Double

dZ = MyArc.DeltaZ
```

SlopeLength

Returns

Double

Description

This property returns the absolute length (3D) of the arc.

Example

```
Dim MyArc As Arc
Dim Slope As Double

Slope = MyArc.SlopeLength
```

ArcAperture

Returns

Double

Description

This property returns the angle at the center of the arc in radians.

Example

```
Dim MyArc As Arc
Dim Angle As Double

Angle = MyArc.ArcAperture
```

Center

Returns

XY

Description

This property returns the coordinate of the center of the arc.

Example

```
Dim MyArc As Arc
Dim Center As XY

Center = MyArc.Center
```

Center3D

Returns

XYZ

Description

This property returns the coordinate of the center of the arc.

Example

```
Dim MyArc As Arc
Dim Center As XYZ

Center = MyArc.Center
```

Methods

CadObjects can not be assigned to objects of a specific type.

So the statement :

```
Set L = Obj
```

is not valid. In stead, the statements GetPoint, GetLine, etc. need to be used.

For a description of the methods GetDocument, GetSubDocument, SetData, DeleteData see object type CadObject.

GetCadObject

Parameters

Obj As CadObject

Description

GetCadObject sets the reference to the Line in a variable of type CadObject. Note that the reference is copied, not the object.

Example

```
Dim Obj As CadObject
Dim MyArc As Arc

MyArc.GetCadObject Obj      ' Same as : Set Obj = MyArc
```

GetBeginPoint

Parameters

BeginPoint As Point

Description

GetBeginPoint returns the first point of the arc.

Example

```
Dim P1 As Point, P2 As Point
Dim A As Arc

A.GetBeginPoint P1
A.GetEndPoint P2
```

GetEndPoint

Parameters

EndPoint As Point

Description

GetEndPoint returns the second point of the arc.

Example

See GetBeginPoint

CDZtoXYZ

Parameters

Chainage As Double, Distance As Double, DeltaZ As Double, _
Pnt As XY, Z As Double

Description

CDZtoXYZ converts the values Chainage, Distance and DeltaZ, to coordinates. On entry DeltaZ is the elevation relative to the arc.

Example

See example for Path

XYZtoCDZ

Parameters

Pnt As XYZ, Chainage As Double, Distance As Double, DeltaZ As Double

Description

XYZtoCDZ converts the coordinates of a point to the values Chainage, Distance and DeltaZ. On return DeltaZ will be the elevation of the projected point the arc.

Example

See CDZtoXYZ.

Chapter 20 - Circle

Description

A Circle is a CadObject. A new Circle can be created with the Overlay method CreateCircle. A reference to an existing Pythagoras Circle can be obtained e.g. with the method GetCircle.

Example

See method CreateCircle of the object type Overlay

Properties

Color
Comment
Link
DisplayLevel
GroupId
Layer
Overlay
CanBeDeleted
CanChangeOverlay
ObjectReference
Selected
Flagged
Container
Data
LineStyle
Scale
Width
Radius
Center
RadiusPnt

Methods

GetDocument
GetSubDocument
SetData
DeleteData
GetCadObject

Properties

For a description of the properties and methods Color, TrueColor, Comment, Link, DisplayLevel, GroupId, Layer, Overlay, CanBeDeleted, CanChangeOverlay, ObjectReference, Selected, Flagged, Container, Data, GetQueryData see object type CadObject.

For a description of the properties LineStyle, Scale, Width see object type Line.

Radius

Returns

Double

Description

This property returns the radius of the circle.

Example

```
Dim MyCircle As Circle
Dim Radius As Double

Radius = MyCircle.Radius
```

Center

Returns

XYZ

Description

This property returns the coordinate (3D) of the center of the circle .

Example

```
Dim MyCircle As Circle
Dim Center As XYZ

Center = MyCircle.Center
```

RadiusPnt

Returns

XYZ

Description

This property returns the coordinate (3D) of the radius point of the circle. The radius point is a point on the circle with the largest Z.

Example

```
Dim MyCircle As Circle
Dim RadiusPnt As XYZ

RadiusPnt = MyCircle.RadiusPnt
```

Methods

CadObjects can not be assigned to objects of a specific type.

So the statement :

```
Set L = Obj
```

is not valid. In stead, the statements GetPoint, GetLine, etc. need to be used.

For a description of the methods GetDocument, GetSubDocument, SetData, DeleteData see object type CadObject.

GetCadObject

Parameters

Obj As CadObject

Description

GetCadObject sets the reference to the Line in a variable of type CadObject. Note that the reference is copied, not the object.

Example

```
Dim Obj As CadObject
Dim MyCircle As Circle

MyCircle.GetCadObject Obj           ' Same as : Set Obj = MyCircle
```

Chapter 21 - Curve

Description

A Curve is a `CadObject`. A new Curve can be created with the Overlay method `CreateCurve`. A reference to an existing Pythagoras Curve can be obtained e.g. with the method `GetCurve`. A Curve is called a polyline in Pythagoras since v16.

Example

See method `CreateCurve` of the object type `Overlay`.

Properties

`Color`
`Comment`
`Link`
`DisplayLevel`
`GroupId`
`Layer`
`Overlay`
`CanBeDeleted`
`CanChangeOverlay`
`ObjectReference`
`Selected`
`Flagged`
`Container`
`Data`
`LineStyle`
`Scale`
`Width`
`NrOfPoints`
`NrOfSegments`
`Curvature`
`MaxRounding`
`ThreeD`
`Elevation`
`StartTangent`
`EndTangent`

Methods

`GetDocument`
`GetSubDocument`
`SetData`
`DeleteData`
`GetCadObject`
`GetCurvePoints`
`GetCurveSegments`
`GetBearingAt`

Properties

For a description of the properties and methods `Color`, `TrueColor`, `Comment`, `Link`, `DisplayLevel`, `GroupId`, `Layer`, `Overlay`, `CanBeDeleted`, `CanChangeOverlay`, `ObjectReference`, `Selected`, `Flagged`, `Container`, `Data`, `GetQueryData` see object type `CadObject`.

For a description of the properties `Style`, `Scale`, `Width` see object type `Line`.

NrOfPoints

Returns

`Integer`

Description

This property returns the number of control points of the curve .

Example

```
Dim C As Curve
Dim Count As Integer

Count = C.NrOfPoints
```

NrOfSegments

Returns

Integer

Description

This property returns the number of segments in the curve.

Example

```
Dim C As Curve
Dim Count as Integer

Count = C.NrOfSegments
```

Curvature (obsolete)

Returns

Integer

Description

Curvature is a property of the v15 Curve object.

This property returns the curvature of the curve.

0 = the curve contains only line segments.

5 = the curve contains only curve (Bezier) segments.

-1 = other cases.

The property also sets the curvature of the curve, if the curve contains only line segments or only curve (Bezier) segments. The value needs to be in the range [0, 10]. 0 are line segments, 5 are normal curve (Bezier) segments and the other values differ in curvature like the v15 Curve.

Example

```
Dim C As Curve

C.Curvature = 5
```

MaxRounding (obsolete)

Returns

Boolean

Description

It returns true since v16 polylines always have max rounding. The variable doesn't set the value anymore since v16.

Example

```
Dim C As Curve
Dim hasMaxRounding As Boolean

hasMaxRounding = C.MaxRounding 'Always true
```

ThreeD

Returns

Boolean

Description

This property returns a Boolean which indicates if the curve is 3D. A curve is 3D if not all control points of the curve have the same elevation.

Example

```
Dim C As Curve
Dim Z As Double

If C.ThreeD = False Then
    Z = C.Elevation
End If
```

Elevation

Returns

Double

Description

This property sets or returns the elevation of a horizontal curve. A curve is horizontal if all control points of the curve have the same elevation. (Is not 3D). If the curve is not horizontal the value `ptMaxDouble` is returned.

Example

```
Dim C As Curve
Dim Z As Double

If C.ThreeD = False Then
    Z = C.Elevation
End If
```

StartTangent (obsolete)

Returns

XY

Description

The property doesn't set anything since v16. It returns a point which gives the tangent direction at the starting point of the curve. If no tangent point or the first segment is not a Bezier segment, both coordinates are equal to `ptMaxDouble`.

Example

```
Dim C As Curve
Dim P As XY

P = C.StartTangent
```

EndTangent (obsolete)

Returns

XY

Description

The property doesn't set anything since v16. It returns a point which gives the tangent direction at the end point of the curve. If no tangent point or the last segment is not a Bezier segment, both coordinates are equal to `ptMaxDouble`.

Example

```
Dim C As Curve
Dim P As XY

P = C.EndTangent
```

Methods

CadObjects can not be assigned to objects of a specific type.

So the statement :

```
Set C = Obj
```

is not valid. In stead, the statements `GetPoint`, `GetCurve`, etc. need to be used.

For a description of the methods `GetDocument`, `GetSubDocument`, `SetData`, `DeleteData` see object type `CadObject`.

GetCadObject

Parameters

Obj As CadObject

Description

`GetCadObject` sets the reference to the Curve in a variable of type `CadObject`. Note that the reference is copied, not the object.

Example

```
Dim Obj As CadObject
Dim MyCurve As Curve

MyCurve.GetCadObject Obj          ' Same as : Set Obj = MyCurve
```


GetCurvePoints

Parameters

Points() As XYZ

Description

GetCurvePoints returns the control points of the curve. The dimension of the array should be large enough to store the points before calling this method.

Example

```
Dim Points() As XYZ
Dim MyCurve As Curve
Dim NrOfPoints As Integer

NrOfPoints = MyCurve.NrOfPoints
Redim Points(1 To NrOfPoints)
MyCurve.GetCurvePoints Points
```

GetCurveSegments

Parameters

Segments() as CurveSegment

Description

Returns the segments of the curve in an array.

Example

```
Dim MyCurve As Curve
Dim Count As Integer
Dim NewPos As XYZ

Count = MyCurve.NrOfSegments()
redim Segments(Count)
MyCurve.GetCurveSegments Segments
```

GetBearingAt

Parameters

Pos As XY

Description

This methods calculates the bearing of the curve at the given position. If the position does not lie on the curve the projected position is used instead.

The bearing is calculated as the angle relative to the north axis and clockwise.

Example

```
Dim C As Curve
Dim Bearing As Double
Dim Pos As XY

C.GetBearingAt(Pos, Bearing)
```

Chapter 22 - Clothoid

Description

A Clothoid is a `CadObject`. A new Clothoid can be created with the `Overlay` method `CreateClothoid`. A reference to an existing Pythagoras Clothoid can be obtained e.g. with the method `GetClothoid`.

Example

See method `CreateClothoid` of the object type `Overlay`.

Properties

`Color`
`Comment`
`Link`
`DisplayLevel`
`GroupId`
`Layer`
`Overlay`
`CanBeDeleted`
`CanChangeOverlay`
`ObjectReference`
`Selected`
`Flagged`
`Container`
`Data`
`LineStyle`
`Scale`
`Width`
`Arrows`
`Length`
`DeltaZ`
`SlopeLength`
`ClothoidConstant` (or `K`)
`StartRadius`
`EndRadius`
`Mirrored`
`StartCenter`
`EndCenter`

Methods

`GetDocument`
`GetSubDocument`
`SetData`
`DeleteData`
`GetCadObject`
`GetBeginPoint`
`GetEndPoint`
`CDZtoXYZ`
`XYZtoCDZ`

Properties

For a description of the properties `Color`, `TrueColor`, `Comment`, `Link`, `DisplayLevel`, `GroupId`, `Layer`, `Overlay`, `CanBeDeleted`, `CanChangeOverlay`, `ObjectReference`, `Selected`, `Flagged`, `Container`, `Data`, `GetQueryData` see object type `CadObject`.

For a description of the properties `Style`, `Scale`, `Width`, `Arrows`, and of the methods `GetBeginPoint`, `GetEndPoint`, see object type `Line`.

Length

Returns

`Double`

Description

This property returns the horizontal length of the clothoid.

Example

```
Dim C As Clothoid

MsgBox "Length spiral = " & C.Length
```

DeltaZ

Returns

Double

Description

This property returns the difference in height between the starting point and the end point of the clothoid.

Example

```
Dim C As Clothoid
Dim dZ As Double

dZ = C.DeltaZ
```

SlopeLength

Returns

Double

Description

This property returns the absolute length (3D) of the clothoid.

Example

```
Dim C As Clothoid
Dim Slope As Double

Slope = C.SlopeLength
```

K (or ClothoidConstant)

Returns

Double

Description

This property returns the rate of change of curvature also called the K-factor of the spiral curve (clothoid).

Example

```
Dim C As Clothoid
Dim K As Double

K = C.K
```

StartRadius

Returns

Double

Description

This property returns the radius at the starting point of the clothoid. If the radius is infinite, ptMaxDouble is returned.

Example

```
Dim C As Clothoid
Dim Radius1 As Double, Radius2 As Double

Radius1 = C.StartRadius
Radius2 = C.EndRadius
```

EndRadius

Returns

Double

Description

This property returns the radius at the end point of the clothoid. If the radius is infinite, ptMaxDouble is returned.

Example

```
Dim C As Clothoid
Dim Radius1 As Double, Radius2 As Double

Radius1 = C.StartRadius
Radius2 = C.EndRadius
```

Mirrored

Returns

Boolean

Description

This property returns the direction of the clothoid. If `False` the clothoid is in counter-clockwise direction (P1 - P2). If `True` clockwise.

Example

```
Dim C As Clothoid
Dim Radius1 As Double, Radius2 As Double

Radius1 = C.StartRadius
Radius2 = C.EndRadius
```

StartCenter

Returns

XY

Description

This property returns the coordinate of the center of the clothoid at the starting point.

Example

```
Dim C As Clothoid
Dim Center1 As XY, Center2 As XY

Center1 = C.StartCenter
Center2 = C.EndCenter
```

EndCenter

Returns

XY

Description

This property returns the coordinate of the center of the clothoid at the end point .

Example

```
Dim C As Clothoid
Dim Center1 As XY, Center2 As XY

Center1 = C.StartCenter
Center2 = C.EndCenter
```

Methods

CadObjects can not be assigned to objects of a specific type.

So the statement :

```
Set C = Obj
```

is not valid. In stead, the statements `GetPoint`, `GetClothoid`, etc. need to be used.

For a description of the methods `GetDocument`, `GetSubDocument`, `SetData`, `DeleteData` see object type `CadObject`.

GetCadObject

Parameters

Obj As `CadObject`

Description

`GetCadObject` sets the reference to the Clothoid in a variable of type `CadObject`. Note that the reference is copied, not the object.

Example

```
Dim Obj As CadObject
Dim C As Clothoid

C.GetCadObject Obj          ' Same as : Set Obj = C
```

GetBeginPoint

Parameters

BeginPoint As Point

Description

GetBeginPoint returns the first point of the clothoid.

Example

```
Dim P1 As Point, P2 As Point
Dim C As Clothoid

C.GetBeginPoint P1
C.GetEndPoint P2
```

GetEndPoint

Parameters

EndPoint As Point

Description

GetEndPoint returns the second point of the clothoid.

Example

See GetBeginPoint

CDZtoXYZ

Parameters

Chainage As Double, Distance As Double, DeltaZ As Double, _
Pnt As XY, Z As Double

Description

CDZtoXYZ converts the values Chainage, Distance and DeltaZ, to coordinates. On entry DeltaZ is the elevation relative to the clothoid.

Example

See example for Path

XYZtoCDZ

Parameters

Pnt As XYZ, Chainage As Double, Distance As Double, DeltaZ As Double

Description

XYZtoCDZ converts the coordinates of a point to the values Chainage, Distance and DeltaZ. On return DeltaZ will be the elevation of the projected point the clothoid.

Example

See CDZtoXYZ.

Chapter 23 - Text

Description

A Text is a CadObject. A new Text can be created with the Overlay method CreateText. A reference to an existing Pythagoras Text can be obtained e.g. with the method GetText.

Example

See method CreateText of the object type Overlay.

Properties

Color
Link
DisplayLevel
GroupId
Layer
Overlay
CanBeDeleted
CanChangeOverlay
ObjectReference
Selected
Flagged
Container
Data
Position
Content
Orientation
Angle
Style
Weight
PointSize
HorizontalAlignment
VerticalAlignment
Underline
Border
LineSpacing
Opaque
NrOfLines

Methods

GetDocument
GetSubDocument
GetCadObject
SetData
DeleteData
GetAngle
GetContent
GetFontName
GetFont
SetFont

Properties

For a description of the properties and methods Color, TrueColor, Link, DisplayLevel, GroupId, Layer, Overlay, CanBeDeleted, CanChangeOverlay, ObjectReference, Selected, Flagged, Container, Data, GetQueryData see object type CadObject.

Position

Returns

XY

Description

This property sets or returns the insertion point of the text.
Insertion point : a text is aligned relative to its insertion point.

Example

```
Dim T As Text  
Dim Pos As XY
```

```
Pos.X = 20.0
Pos.Y = 30.0
T.Position = Pos
```

Content

Returns
String

Description

This property sets or returns the content of the text. To obtain the text, the alternative method `GetContent` returns the text, with the option to replace the placeholders with their respective values.

Example

```
Dim T As Text

T.Content = "Text created with VBA" & vbCrLf & "Line 2 of the text"
```

Orientation

Returns
Integer

Description

This property sets or returns the orientation of the text.

`Orientation = ptTextHorizontal`: The text will always appear horizontal independent of page orientation or angle.

`Orientation = ptTextOrientationReadable`: The angle of the displayed or printed texts will change if the text would come upside down. If the angle of the texts is in 2nd or 3rd quadrant, the texts are turned over 180 degrees.

`Orientation = ptTextOrientationAbsolute`: The texts are always displayed at the given angle.

Warning : when setting the angle with the `Angle` property, the orientation will be set to `ptTextOrientationAbsolute` if it was `ptTextHorizontal` otherwise the orientation is kept.

Example

```
Dim T As Text

T.Angle = 1.3
T.Orientation = ptTextOrientationReadable
```

Angle

Returns
Double

Description

This property sets or returns the angle of the text. When writing, `TextAngle` will set the orientation to `ptTextOrientationAbsolute`, if the orientation was horizontal (even if the angle = 0.0). If the orientation was not horizontal, the orientation is kept. The angle is expressed in radians.

Example

```
Dim T As Text

T.Angle = 1.3
T.Orientation = ptTextOrientationReadable
```

Style

Returns
Integer

Description

This property sets or returns the style of the text.

`Style = ptUpright` : Normal

`Style = ptItalic` : Italic

Example

```
Dim T As Text
```

```
T.Style = ptUpright
```

Weight

Returns

Integer

Description

This property sets or returns the weight of the text.

Weight = ptLight : Light

Weight = ptMedium : Normal

Weight = ptBold : Bold

Weight = ptExtraBold : Extra Bold

Not all fonts can be represented Light or Extra Bold.

Example

```
Dim T As Text
```

```
T.Weight = ptBold
```

PointSize

Returns

Integer

Description

This property sets or returns the point size of the text. Value of PointSize must be between 1 and 1000.

Example

```
Dim T As Text
```

```
T.PointSize = 20
```

HorizontalAlignment

Returns

Integer

Description

This property sets or returns the horizontal alignment of the text.

Alignment = ptAlignLeft : left aligned.

Alignment = ptAlignCenter : centered.

Alignment = ptAlignRight : right aligned.

Example

```
Dim T As Text
```

```
T.HorizontalAlignment = ptAlignRight
```

VerticalAlignment

Returns

Integer

Description

This property sets or returns the vertical alignment of the text.

Alignment = ptAlignBottom: bottom aligned.

Alignment = ptAlignCenter : centered.

Alignment = ptAlignTop : top aligned.

Example

```
Dim T As Text
```

```
T.VerticalAlignment = ptAlignCenter
```

Underline

Returns

Integer

Description

This property sets or returns the underlining of the text.

Underline = ptUnderlineNone : no underlining.

Underline = ptUnderlineSingle : single underlining.

Underline = ptUnderlineDouble : double underlining.

Underline = ptUnderlineDashed : underlining dashed line.

Example

```
Dim T As Text
```

```
T.Underline = ptUnderlineSingle
```

Border

Returns

Integer

Description

This property sets or returns the border of the text.

Border = ptBorderNone : no border.

Border = ptBorderSingle : single border.

Border = ptBorderDouble : double border.

Example

```
Dim T As Text
```

```
T.Border = ptBorderSingle
```

LineSpacing

Returns

Integer

Description

This property sets or returns the line spacing of the text.

LineSpacing = ptSpacingSmall : small line spacing.

LineSpacing = ptSpacingNormal : normal line spacing.

LineSpacing = ptSpacingLarge : large line spacing.

Example

```
Dim T As Text
```

```
T.LineSpacing = ptSpacingSmall
```

Opaque

Returns

Boolean

Description

This property sets or returns the transparency of the text.

If = True : Opaque

Example

```
Dim T As Text
```

```
T.Opaque = True
```

NrOfLines

Returns

Integer

Description

This property returns the number of lines in the text content.

Example

```
Dim T As Text
```

```

If T.NrOfLines > 1 then
    MsgBox "Multiline text"
End if

```

Methods

CadObjects can not be assigned to objects of a specific type.

So the statement :

```
Set T = Obj
```

is not valid. In stead, the statements GetPoint, GetText, etc. need to be used.

For a description of the methods GetDocument, GetSubDocument, SetData, DeleteData see object type CadObject.

GetCadObject

Parameters

```
Obj As CadObject
```

Description

GetCadObject sets the reference to the Text in a variable of type CadObject. Note that the reference is copied, not the object.

Example

```
Dim Obj As CadObject
Dim T As Text
```

```
T.GetCadObject Obj ' Same as : Set Obj = T
```

GetAngle

Parameters

```
AngleAsDisplayed As Boolean, Angle As Double
```

Description

GetAngle returns the angle of the text. If the orientation of the text = ptTextHorizontal, then the returned angles is always = 0.0.

If the input parameter AngleAsDisplayed = True, then the corrected angle (+180°) will be returned if the orientation of the text = ptTextOrientationReadable, and the text without correction would have been displayed upside down. The angle is expressed in radians.

Example

```
Dim T As Text
Dim Angle As Double
```

```
T.GetAngle True, Angle
```

GetContent

Parameters

```
ReplacePlaceHolders As Boolean, Content As String
```

Description

GetContent returns the content of the text. If the Boolean ReplacePlaceHolders = True, then the variables in the text (^D, ^S, ^F) will be replaced by their actual values. (Date, Scale, File name).

Example

```
Dim T As Text
Dim Content As String
```

```
T.GetContent True, Content
```

GetFontName

Parameters

```
FontName As String
```

Description

Name of font is returned.

Example

```
Dim T As Text, FontName As String

T.GetFontName FontName
```

GetFont

Parameters

Font As Variant, Optional IsDingbat As Boolean

Description

If text is painted in Pythagoras vector font then Font is set to ptPythagorasVectorFont (integer value). Otherwise name of the font is stored. IsDingbat variable is set to True for dingbat fonts (Windings, Webdings etc.) or to False otherwise.

Example

```
Dim T As Text, FontName

T.GetFont FontName
```

GetFont (obsolete format)

Parameters

FontName As String, CodePage As Integer

Description

This format remains in VBA only to execute code written for VBA version ≤ 13 . Name of font is returned. CodePage variable is set to ptSymbol when the font is a dingbat font or to ptWestern otherwise.

SetFont

Parameters

Font As Variant, Optional IsDingbat As Boolean

Description

Font should be either ptPythagorasVectorFont or a name of a font. The text will be painted in this font. Specified name of the font should contain only characters with codes < 128 (basic ASCII). When the specified font is not installed in the current instance of Windows then value of IsDingbat will be used to set the corresponding property of the text object. If the font is installed in Windows then the value will be ignored and the corresponding property will be set correspondingly to properties of the font itself.

Example

```
Dim T As Text

T.SetFont "Arial", False
```

SetFont (obsolete format)

Parameters

FontName As String, CodePage As Integer

Description

This format remains in VBA only to execute code written for VBA version ≤ 13 . The text will be painted in a font with this name. When the specified font is not installed in the current instance of Windows then value of CodePage will be used to set the corresponding property of the text object: CodePage=ptSymbol defines the text object to contain dingbat characters. If the font is installed in Windows then CodePage value will be ignored and the corresponding property of the text object will be set correspondingly to properties of the font itself.

Chapter 24 - RichText

Description

A RichText is a CadObject. A new RichText can be created with the Overlay method CreateRichText. A reference to an existing Pythagoras Rich Text can be obtained e.g. with the method GetRichText.

Example

```
Dim Obj As CadObject
Dim RT As RichText

Obj.GetRichText RT
RT.Angle = 0
```

Properties

Link
DisplayLevel
GroupId
Layer
Overlay
CanBeDeleted
CanChangeOverlay
ObjectReference
Selected
Flagged
Container
Data
Position
Content
Orientation
Angle
HorizontalAlignment
VerticalAlignment
Opaque
NrOfLines

Methods

GetDocument
GetSubDocument
SetData
DeleteData
GetCadObject
GetAngle

Properties

For a description of the properties and methods Link, DisplayLevel, GroupId, Layer, Overlay, CanBeDeleted, CanChangeOverlay, ObjectReference, Selected, Flagged, Container, Data, GetQueryData see object type CadObject.

Position

See object type Text.

Content

Returns

String

Description

This property returns the text content of A RichText object as a string.

Example

```
Dim RT As RichText
Dim Str As String

Str = T.Content
```

Orientation

See object type `Text`.

Angle

See object type `Text`.

HorizontalAlignment

See object type `Text`.

VerticalAlignment

See object type `Text`.

Opaque

See object type `Text`.

Border

See object type `Text`.

NrOfLines

See object type `Text`.

Methods

`CadObjects` can not be assigned to objects of a specific type.

So the statement :

```
Set RT = Obj
```

is not valid. In stead, the statements `GetPoint`, `GetRichText`, etc. need to be used.

For a description of the methods `GetDocument`, `GetSubDocument`, `SetData`, `DeleteData` see object type `CadObject`.

GetCadObject*Parameters*

`Obj As CadObject`

Description

`GetCadObject` sets the reference to the `RichText` in a variable of type `CadObject`. Note that the reference is copied, not the object.

Example

```
Dim Obj As CadObject  
Dim RT As RichText
```

```
RT.GetCadObject Obj          ' Same as : Set Obj = RT
```

GetAngle

See object type `Text`.

Chapter 25 - Path

Description

A Path is a CadObject. A new Path can be created with the Overlay method CreatePath. A reference to an existing Pythagoras Path can be obtained e.g. with the method GetPath.

Example

See method CreatePath of the object type Overlay

Properties

Color
Comment
Link
DisplayLevel
GroupId
Layer
Overlay
CanBeDeleted
CanChangeOverlay
ObjectReference
Selected
Flagged
Container
Data
NrOfObjects
NrOfElements
TotalLength
TotalSlopeLength
DeltaZ
StartingChainage

Methods

GetDocument
GetSubDocument
SetData
DeleteData
GetCadObject
GetPathObjects
GetPathElements
CDZtoXYZ
XYZtoCDZ

Properties

For a description of the properties and methods Comment, Link, DisplayLevel, GroupId, Layer, Overlay, CanBeDeleted, CanChangeOverlay, ObjectReference, Selected, Flagged, Container, Data, GetQueryData see object type CadObject.

NrOfObjects

Returns

Integer

Description

This property returns the number of objects (points, arcs or clothoids) which make up the path.

Example

```
Dim MyPath As Path
Dim NrOfObjects As Double

NrOfObjects = MyPath.NrOfObjects
```

NrOfElements

Returns

Integer

Description

This property returns the number of elements (lines, arcs or clothoids) which make up the path. This property can be used when working with objects of the class `PathElement`.

Example

```
Dim MyPath As Path
Dim NrOfElements As Double

NrOfObjects = MyPath.NrOfElements
```

TotalLength

Returns

Double

Description

`TotalLength` returns the sum of the horizontal lengths of all objects that make up the path.

Example

```
Dim MyPath As Path
Dim L As Double

L = MyPath.TotalLength
```

TotalSlopeLength

Returns

Double

Description

`TotalSlopeLength` returns the sum of the slope lengths of all objects that make up the path.

Example

```
Dim MyPath As Path
Dim L As Double

L = MyPath.TotalSlopeLength
```

DeltaZ

Returns

Double

Description

`DeltaZ` returns the difference in height between the starting point and the end point of the path.

Example

```
Dim MyPath As Path
Dim DeltaZ As Double

DeltaZ = MyPath.DeltaZ
```

StartingChainage

Returns

Double

Description

`StartingChainage` sets or returns the chainage at the starting point of the path.

Example

```
Dim MyPath As Path
Dim StartingChainage As Double

StartingChainage = MyPath.StartingChainage
```

Methods

`CadObjects` can not be assigned to objects of a specific type.

So the statement :

```
Set L = Obj
```

is not valid. In stead, the statements `GetPoint`, `GetLine`, etc. need to be used.

For a description of the methods `GetDocument`, `GetSubDocument`, `SetData`, `DeleteData` see object type `CadObject`.

GetCadObject

Parameters

`Obj As CadObject`

Description

`GetCadObject` sets the reference to the Path in a variable of type `CadObject`. Note that the reference is copied, not the object.

Example

```
Dim Obj As CadObject
Dim MyPath As Path

MyPath.GetCadObject Obj          ' Same as : Set Obj = MyPath
```

GetPathObjects

Parameters

`Objects() As CadObject`

Description

`GetPathObjects` returns an array of variables of type `CadObject`. This array contains all objects of the path (Point, Arc, Clothoid) in the same sequence as they appear in the path. On return, the dimension of the array is always (1 To `NrOfObjects`).

Example

```
Dim A As Application
Dim D As Document
Dim Sel As Selection
Dim Counter As Long
Dim NrPoints As Long, NrArcs As Long, NrClothoids As Long
Dim Obj As CadObject
Dim Objs() As CadObject
Dim Pth As Path

Set A= New Application
A.GetActiveDocument D
D.GetSelection Sel
If Sel.Count(ptPath) = 1 then
    Sel.GetFirst ptPath, Obj
    Obj.GetPath Pth
    Redim Objs(1 To Pth.NrOfObjects)
    Pth.GetPathObjects Objs
    For Counter = 1 TO Pth.NrOfObjects
        Select case Objs(Counter).ObjectType
            case ptPoint
                NrPoints = NrPoints + 1
            case ptArc
                NrArcs = NrArcs + 1
            case ptClothoid
                NrClothoids = NrClothoids + 1
            case else
                end select
    Next counter
    MsgBox "Points : " & NrPoints & VBCRLF & "Arcs : " & NrArcs & VBCRLF & _
        "Clothoids : " & NrClothoids
Else
    MsgBox "Select 1 path"
End If
```

GetPathElements

Parameters

`Elements () As PathElement`

Description

GetPathElements returns an array of variables of type PathElement. This array contains all objects of the path (Line, Arc, Clothoid) in the same sequence as they appear in the path.
On return, the dimension of the array is always (1 To NrofElements).

Example

```
Dim A As Application
Dim D As Document
Dim Sel As Selection
Dim Counter As Long
Dim NrLines As Long, NrArcs As Long, NrClothoids As Long
Dim Obj As CadObject
Dim Elems() As PathElement
Dim Pth As Path

Set A = New Application
A.GetActiveDocument D
D.GetSelection Sel
If Sel.Count(ptPath) = 1 then
    Sel.GetFirst ptPath, Obj
    Obj.GetPath Pth
    Redim Elems(1 To Pth.NrofElements)
    Pth.GetPathElements Elems
    For Counter = 1 TO Pth.NrofElements
        Select case Elems (Counter).ElementType
            case ptLine
                NrLines = NrLines + 1
            case ptArc
                NrArcs = NrArcs + 1
            case ptClothoid
                NrClothoids = NrClothoids + 1
            case else
            end select
    Next counter
    MsgBox "Lines : " & NrLines & VBCRLF & "Arcs : " & NrArcs & VBCRLF &
    "Clothoids : " & NrClothoids
Else
    MsgBox "Select 1 path"
End If
```

CDZtoXYZ

Parameters

Chainage As Double, Distance As Double, DeltaZ As Double, _
Pnt As XY, Z As Double

Description

CDZtoXYZ converts the values Chainage, Distance and DeltaZ, to coordinates. On entry DeltaZ must be the elevation relative to the path.

The returned elevation will be:

$Z = \text{DeltaZ} + z\text{Path}(\text{Chainage})$. The starting chainage of the path will be taken into account.

Example

```
Dim A As Application
Dim Doc As Document
Dim Sel As Selection
Dim Obj As CadObject
Dim Pa As Path
Dim Pnt As XY
Dim Pnt3D As XYZ
Dim C As Double, D As Double, Z As Double

Set A = New Application
A.GetActiveDocument Doc
Doc.GetSelection Sel
If Sel.Count(ptPath) = 1 Then
    Sel.GetFirst ptPath, Obj
    Obj.GetPath Pa
    Pa.CDZtoXYZ 70.0, 5.0, 1.5, Pnt, Z
    MsgBox "X = " & Format(Pnt.x, "###.000") & vbNewLine & _
```

```

        "Y = " & Format(Pnt.Y, "###.000") & vbNewLine & _
        "Z = " & Format(Z, "###.000") & vbNewLine
Pnt3D.x = Pnt.x
Pnt3D.y = Pnt.y
Pnt3D.z = Z
Pa.XYZtoCDZ Pnt3D, C, D, Z
MsgBox "C = " & Format(C, "###.000") & vbNewLine & _
        "D = " & Format(D, "###.000") & vbNewLine & _
        "Z = " & Format(Z, "###.000") & vbNewLine
End If

```

XYZtoCDZ

Parameters

Pnt As XYZ, Chainage As Double, Distance As Double, DeltaZ As Double

Description

XYZtoCDZ converts the coordinates of a point to the values Chainage, Distance and DeltaZ. On return DeltaZ will be the elevation relative to the path. The starting chainage of the path will be taken into account.

Example

See CDStoXYZ

Chapter 26 - Road

Description

A Road is a `CadObject`. Currently there is no VBA interface to create a Road. A reference to an existing Pythagoras Road can be obtained e.g. with the method `GetRoad`.

Example

```
Dim Obj As CadObject
Dim R As Road

Obj.GetRoad R
```

Properties

`Color`
`Comment`
`Link`
`DisplayLevel`
`GroupId`
`Layer`
`Overlay`
`CanBeDeleted`
`CanChangeOverlay`
`ObjectReference`
`Selected`
`Flagged`
`Data`
`TotalLength`
`StartingChainage`

Methods

`GetDocument`
`GetSubDocument`
`GetColor`
`SetColor`
`SetData`
`DeleteData`
`GetCadObject`
`GetProfileData`
`GetHorIPData`
`GetSuperElevationData`
`GetRD_CrossSectionPositions`
`GetRD_ReferenceObjects`
`GetCenterLine`
`CDZtoXYZ`
`XYZtoCDZ,`
`GetCrossSectionData`

Properties

For a description of the properties and methods `Comment`, `Link`, `DisplayLevel`, `GroupId`, `Layer`, `Overlay`, `CanBeDeleted`, `CanChangeOverlay`, `ObjectReference`, `Selected`, `Flagged`, `Data`, `GetQueryData` see object type `CadObject`.

Color

Returns

Integer

Description

This property sets or returns the color of the center line of the road. See details in the description of the property *Color* in the chapter *CadObject*.

TotalLength

Returns

Double

Description

TotalLength returns the sum of the lengths of all objects that make up the road.

Example

```
Dim MyRoad As Road
Dim L As Double

L = MyRoad.TotalLength
```

StartingChainage

Returns

Double

Description

StartingChainage sets or returns the chainage at the starting point of the road.

Example

```
Dim MyRoad As Road
Dim StartingChainage As Double

StartingChainage = MyRoad.StartingChainage
```

Methods

CadObjects can not be assigned to objects of a Road type.

So the statement :

```
Set R = Obj
```

is not valid. Instead, the statement GetRoad needs to be used.

For a description of the methods GetRoad, SetData, DeleteData see object type CadObject.

GetCadObject

Parameters

Obj As CadObject

Description

GetCadObject sets the reference to the Road in a variable of type CadObject. Note that the reference is copied, not the object.

Example

```
Dim Obj As CadObject
Dim MyRoad As Road

MyRoad.GetCadObject Obj           ' Same as : Set Obj = MyRoad
```

GetColor

Parameters

AttributeType As Integer

Returns

Integer

Description

This method returns the color of the specified attribute of the Road. The attribute is specified by one of constants ptCenterLineAttr, ptSplitPointAttr, ptReferencePointAttr, and ptSlopeInterceptAttr. The color is a sequence number in the standard palette of 256 colors. When the true color of an object is not in the palette, then the nearest color is returned.

Example

```
Dim MyRoad As Road
Dim SplitColor As Integer

SplitColor = MyRoad.GetColor(ptSplitPointAttr)
```

SetColor

Parameters

AttributeType As Integer, ColorValue As Integer

Description

This method sets the color of the specified attribute of the Road. The attribute is specified by one of constants `ptCenterLineAttr`, `ptSplitPointAttr`, `ptReferencePointAttr`, and `ptSlopeInterceptAttr`. The color is a sequence number in the standard palette of 256 colors.

Example

```
Dim MyRoad As Road
Dim SplitColor As Integer

MyRoad.SetColor ptSplitPointAttr, SplitColor
```

GetProfileData

Parameters

`NumberOfIPs` As Long, `IPs()` As ProfileElement

Description

`GetProfileData` returns the data of the profile of the Road.

If the road has no profile, then `NumberOfIPs` will be = 0. The starting chainage of the road is taken into account.

Example

```
Dim A As Application
Dim Doc As Document
Dim Sel As Selection
Dim Obj As CadObject
Dim oRoad As Road
Dim IPs() As ProfileElement
Dim NrOfIPs As Long, i As Long
Dim MyDialogs As Dialogs
Dim FileName As String
Dim BCx As String, BCy As String, ECx As String, ECy As String

Redim IPs(1 To 5)
Set A = New Application
A.GetActiveDocument Doc
If Not (Doc Is Nothing) Then
    Doc.GetSelection Sel
    If Sel.Count(ptRoad) = 1 Then
        A.GetDialogs MyDialogs
        Open "ProfileReport.txt" For Output As #1
        Sel.GetFirst ptRoad, Obj
        Obj.GetRoad oRoad
        oRoad.GetProfileData NrOfIPs, IPs
        Print #1, "BC x"; Tab(8); "z"; Tab(16); "EC x"; Tab(24); "z"
        For i = 1 To NrOfIPs
            BCx = Format(IPs(i).BC.x, "###.00")
            BCy = Format(IPs(i).BC.y, "###.00")
            ECx = Format(IPs(i).EC.x, "###.00")
            ECy = Format(IPs(i).EC.y, "###.00")
            Print #1, BCx; Tab(8); BCy; Tab(16); ECx; Tab(24); ECy
        Next i
        Close #1
        MyDialogs.RequestTextFileView "ProfileReport.txt"
    End If
End If
```

GetHorizontalAlignmentData

Parameters

`NumberOfIPs` As Long, `IPs()` As HorizontalTransition

Description

`GetHorizontalAlignmentData` returns the data of the horizontal alignment of the Road.

Example

```
Dim A As Application
Dim Doc As Document
Dim Sel As Selection
Dim Obj As CadObject
Dim oRoad As Road
```

```

Dim IPs() As HorizontalTransition
Dim NrOfIPs As Long

Set A = New Application
A.GetActiveDocument Doc
If Not (Doc Is Nothing) Then
    Doc.GetSelection Sel
    If Sel.Count(ptRoad) = 1 Then
        Sel.GetFirst ptRoad, Obj
        Obj.GetRoad oRoad
        oRoad.GetHorizontalAlignmentData NrOfIPs, IPs
    End If
End If

```

GetSuperElevationData

Parameters

NumberOfIPs As Long, IPs() As SuperElevationData

Description

GetSuperElevationData returns the data of the superelevation data of the Road.

Example

```

Dim A As Application
Dim Doc As Document
Dim Sel As Selection
Dim Obj As CadObject
Dim oRoad As Road
Dim IPs() As SuperElevationData
Dim NrOfIPs As Long

Set A = New Application
A.GetActiveDocument Doc
If Not (Doc Is Nothing) Then
    Doc.GetSelection Sel
    If Sel.Count(ptRoad) = 1 Then
        Sel.GetFirst ptRoad, Obj
        Obj.GetRoad oRoad
        oRoad.GetSuperElevationData NrOfIPs, IPs
    End If
End If

```

GetRD_CrossSectionPositions

Parameters

NumberOfCrossSections As Long, Positions() As Double

Description

GetRD_CrossSectionPositions returns the chainage of each cross section (also called marker) that is defined in road design mode.

If the road has no cross section markers, then NumberOfCrossSections will be = 0. The starting chainage of the road is taken into account.

Example

```

Dim A As Application
Dim Doc As Document
Dim Sel As Selection
Dim Obj As CadObject
Dim oRoad As Road
Dim NrOfXSections As Long
Dim Positions() As Double

Redim Positions(1 To 5)
Set A = New Application
A.GetActiveDocument Doc
If Not (Doc Is Nothing) Then
    Doc.GetSelection Sel
    If Sel.Count(ptRoad) = 1 Then
        Sel.GetFirst ptRoad, Obj
        Obj.GetRoad oRoad
        oRoad.GetRD_CrossSectionPositions NrOfXSections, Positions
    End If
End If

```

```

        End If
    End If

```

GetRD_ReferenceObjects

Parameters

NumberOfReferences As Long, References() As CadObject

Description

GetRD_ReferenceObjects returns all the referenced objects that are displayed in the profile window. If the road has no profile, then NumberOfReferences will be = 0.

Example

```

Dim A As Application
Dim Doc As Document
Dim Sel As Selection
Dim Obj As CadObject
Dim oRoad As Road
Dim NumberOfReferences As Long
Dim References() As CadObject

Redim References(1 To 5)
Set A = New Application
A.GetActiveDocument Doc
If Not (Doc Is Nothing) Then
    Doc.GetSelection Sel
    If Sel.Count(ptRoad) = 1 Then
        Sel.GetFirst ptRoad, Obj
        Obj.GetRoad oRoad
        oRoad.GetRD_ReferenceObjects NumberOfReferences, References
    End If
End If

```

GetCenterLine

Parameters

NumberOfElements As Long, Elements() As PathElement

Description

GetCenterLine gets the center line of the road as an array of values of the type PathElement. Each element represents a line element: Line, Arc, or Clothoid. Objects in the array are in the same sequence as the corresponding elements of the center line of the road.

Example

```

Dim A As Application
Dim Doc As Document
Dim Sel As Selection
Dim NrOfPs As Long
Dim Ps As PathElement
Dim MyDialogs As Dialogs
Dim R As Road
Dim Obj As CadObject
Dim i As Integer
Dim Str As String

Redim Ps(1 To 20)
Set A = New Application
A.GetActiveDocument Doc
If Not (Doc Is Nothing) Then
    Doc.GetSelection Sel
    If Sel.Count(ptRoad) = 1 Then
        Sel.GetFirst ptRoad, Obj
        Obj.GetRoad R
        A.GetDialogs MyDialogs
        Open "CenterLineReport.txt" For Output As #1
        R.GetCenterLine NrOfPs, Ps
        For i = 1 To NrOfPs
            Select Case Ps(i).ElementType
            Case ptLine
                Str = "Line: (" & _
                    & CStr(Ps(i).Line.BeginPoint.x) & ", " & _

```

```

        & CStr(Ps(i).Line.BeginPoint.y) & "), (" _
        & CStr(Ps(i).Line.EndPoint.x) & "," _
        & CStr(Ps(i).Line.EndPoint.y) & ") "
    Case ptArc
        Str = "Arc: (" & CStr(Ps(i).Arc.Center.x) & "," & _
            CStr(Ps(i).Arc.Center.y) & ") "
    Case ptClothoid
        Str = "Clothoid..."
    End Select
    Print #1, Str
Next i
Close #1
MyDialogs.RequestTextFileView "CenterLineReport.txt"
End If
End If

```

CDZtoXYZ

Parameters

Chainage As Double, Distance As Double, DeltaZ As Double, _
Pnt As XY, Z As Double

Description

CDZtoXYZ converts the values Chainage, Distance and DeltaZ, to coordinates. On entry DeltaZ must be the elevation relative to the profile. The starting chainage of the road will be taken into account.

The returned elevation will be:

$Z = \text{DeltaZ} + \text{zProfile}(\text{Chainage})$.

Example

```

Dim A As Application
Dim Doc As Document
Dim Sel As Selection
Dim Obj As CadObject
Dim R As Road
Dim Pnt As XY
Dim Pnt3D As XYZ
Dim C As Double, D As Double, Z As Double

Set A = New Application
A.GetActiveDocument Doc
Doc.GetSelection Sel
If Sel.Count(ptRoad) = 1 Then
    Sel.GetFirst ptRoad, Obj
    Obj.GetRoad R
    R.CDZtoXYZ 70.0, 5.0, 1.5, Pnt, Z
    MsgBox "X = " & Format(Pnt.x, "###.000") & vbNewLine & _
        "Y = " & Format(Pnt.y, "###.000") & vbNewLine & _
        "Z = " & Format(Z, "###.000") & vbNewLine
    Pnt3D.x = Pnt.x
    Pnt3D.y = Pnt.y
    Pnt3D.z = Z
    R.XYZtoCDZ Pnt3D, C, D, Z
    MsgBox "C = " & Format(C, "###.000") & vbNewLine & _
        "D = " & Format(D, "###.000") & vbNewLine & _
        "Z = " & Format(Z, "###.000") & vbNewLine
End If

```

XYZtoCDZ

Parameters

Pnt As XYZ, Chainage As Double, Distance As Double, DeltaZ As Double

Description

XYZtoCDZ converts the coordinates of a point to the values Chainage, Distance and DeltaZ. On return DeltaZ will be the elevation relative to the profile. The starting chainage of the road will be taken into account.

Example

See CDZtoXYZ.

GetCrossSectionData

Parameters

Chainage As Double, NumberOfElements As Long, Elements() As GeoObject,
NumberOfReferences As Long, References() As XY

Description

GetCrossSectionData returns an adjusted cross section symbol at a specific chainage as an array of GeoObjects and its reference points as XY coordinates.

Example

```
Dim A As Application
Dim Doc As Document
Dim Sel As Selection
Dim Obj As CadObject
Dim R As Road
Dim Chainage As Double
Dim NrElements As Long
Dim Elements() As GeoObject
Dim NrRefPnts As Long
Dim RefPnts() As XY
Dim i As Long
Dim Element As GeoObject
Dim RefPnt As XY

Set A = New Application
A.GetActiveDocument Doc
Doc.GetSelection Sel
If Sel.Count(ptRoad) = 1Then
    Sel.GetFirst ptRoad, Obj
    Obj.GetRoad R
    Chainage = 25.0;
    R.GetXrossSectionData Chainage, NrElements, Elements, NrRefPnts, RefPnts
    For i = 1 To NrElements
        Element = Element(i)
    Next i
    For i = 1 To NrRefPnts
        RefPnt = RefPnts(i)
    Next i
End If
```

Chapter 27 - Polygon

Description

A Polygon is a CadObject. A new Polygon can be created with the Overlay method CreatePolygon. A reference to an existing Pythagoras Polygon can be obtained e.g. with the method GetPolygon.

Example

See method CreatePolygon of the object type Overlay

Properties

Color
Comment
Link
DisplayLevel
GroupId
Layer
Overlay
CanBeDeleted
CanChangeOverlay
ObjectReference
Selected
Flagged
Container
Data
NrOfObjects
Pattern
Border
Opaque
BackgroundColor
BackgroundPattern
Scale
Area
Perimeter
TotalSlopeLength

Methods

GetDocument
GetSubDocument
SetData
DeleteData
GetCadObject
GetPolygonObjects
AlignPolygon
GetPolygonAlignment
PointInsidePolygon

Properties

For a description of the properties and methods Color, TrueColor, Comment, Link, DisplayLevel, GroupId, Layer, Overlay, CanBeDeleted, CanChangeOverlay, ObjectReference, Selected, Flagged, Container, Data, GetQueryData see object type CadObject.

NrOfObjects

Returns

Integer

Description

This property returns the number of objects (points, arcs or clothoids) which make up the polygon. The first point is included also as last point. So a square polygon consists of 5 objects.

Example

```
Dim Poly As Polygon
Dim NrOfObjects As Double

NrOfObjects = Poly.NrOfObjects
```

Pattern

Returns

Style

Description

The property `Pattern` sets or returns the hatching of the polygon. See also the constants `Polygon : StyleId`

Example

```
Dim Poly As Polygon
Dim NewStyle As Style

With NewStyle
    .Symbol = True
    .Group = "Test"
    .Item = "Horizontal"
End With
Poly.Pattern = NewStyle
```

Border

Returns

Integer

Description

Sets or returns the border of the polygon.

`Border = ptNoBorder`: The polygon is filled / will be filled normal (not only a border).

`Border = ptBorder1mm .. ptBorder10mm`: The width of the border of the polygon is / will be from 1 mm up to 10 mm depending on the value of the parameter.

`Border = ptBorder1mmOpen .. ptBorder10mmOpen`: The width of the border of the polygons is / will be from 1 mm up to 10 mm depending on the value of the parameter. The last leg of the polygon is left open.

Example

```
Dim Poly1 As Polygon, Poly2 As Polygon

Poly1.Border = ptBorder5mm
Poly2.Border = Poly1.Border
```

Opaque

Returns

Boolean

Description

Sets or returns the attribute `Opaque` of the Polygon. If `Opaque = True`, then the polygon will be opaque.

Example

```
Dim Poly1 As Polygon, Poly2 As Polygon

Poly1.Border = ptBorder5mm
Poly2.Opaque = Poly1.Opaque
```

BackgroundColor

Returns

Integer

Description

Sets or returns the background color of the Polygon. The background color corresponds to a color number of the systems color palette. Setting the background color has only effect if the type of background is set.

Example

```
Dim Poly As Polygon

Poly.BackgroundPattern = ptGray50
Poly.BackgroundColor = 81
```

BackgroundPattern

Returns

Integer

Description

Sets or returns the background pattern of the Polygon. The background pattern corresponds to a number indicating a specific polygon pattern. (See Polygon : styleId). Only the values `ptNoFill`, `ptSolidFill`, `ptGrayxx` (xx = 75, 50, 25 or 12) are valid background patterns.

Example

```
Poly.BackgroundPattern = ptGray12
```

Scale

Returns

Double

Description

Sets or returns the scale of the hatching pattern of the Polygon. The scale is expressed in %.

Example

```
Dim Poly1 As Polygon, Poly2 As Polygon

Poly1.Scale = 150.0
Poly2.Scale = Poly1.Scale
```

Area

Returns

Double

Description

Returns the area of the Polygon.

Example

```
Dim Poly As Polygon

MsgBox "Area = " & Poly.Area & " m2"
```

Perimeter

Returns

Double

Description

Returns the perimeter of the Polygon.

Example

```
Dim Poly As Polygon

MsgBox "Perimeter = " & Poly.Perimeter & " m"
```

TotalSlopeLength

Returns

Double

Description

Returns the sum of the slope lengths of all elements of the Polygon.

Example

```
Dim Poly As Polygon

MsgBox "Perimeter = " & Poly.Perimeter & " m"
```

Centroid

Returns

XY

Description

Returns the centroid (centre of gravity) of the Polygon.

Example

```
Dim Poly As Polygon
Dim C As XY

C = Poly.Centroid
```

Methods

CadObjects can not be assigned to objects of a specific type.

So the statement :

```
Set L = Obj
```

is not valid. In stead, the statements GetPoint, GetLine, etc. need to be used.

For a description of the methods GetDocument, GetSubDocument, SetData, DeleteData see object type CadObject.

GetCadObject

Parameters

```
Obj As CadObject
```

Description

GetCadObject sets the reference to the Line in a variable of type CadObject. Note that the reference is copied, not the object.

Example

```
Dim Obj As CadObject
Dim MyCircle As Circle

MyCircle.GetCadObject Obj           ' Same as : Set Obj = MyCircle
```

GetPolygonObjects

Parameters

```
Objects() As CadObject
```

Description

GetPolygonObjects returns an array of variables of type CadObject. This array contains all objects of the polygon (Point, Arc, Clothoid) in the same sequence as they appear in the polygon. On return, the dimension of the array is always (1 To NrofObjects).

Example

```
Dim A As Application
Dim D As Document
Dim Sel As Selection
Dim Counter As Long
Dim NrPoints As Long, NrArcs As Long, NrClothoids As Long
Dim Obj As CadObject
Dim Objs() As CadObject
Dim Poly As Polygon

Set A= New Application
A.GetActiveDocument D
D.GetSelection Sel
If Sel.Count(ptPolygon) = 1 then
    Sel.GetFirst ptPolygon, Obj
    Obj.GetPolygon Poly
    Redim Objs(1 To Poly.NrofObjects)
    Poly.GetPolygonObjects Objs
    For Counter = 1 TO Poly.NrofObjects
        Select case Objs(Counter).ObjectType
            case ptPoint
                NrPoints = NrPoints + 1
            case ptArc
                NrArcs = NrArcs + 1
            case ptClothoid
                NrClothoids = NrClothoids + 1
            case else
            end select
    Next Counter
End If
```

```

        Next counter
        MsgBox "Points : " & NrPoints & VBCRLF & "Arcs : " & NrArcs & VBCRLF & _
            "Clothoids : " & NrClothoids

    Else
        MsgBox "Select 1 Polygon"
    End If

```

GetPolygonPathElements

Parameters

Elements() As PathElement

Description

GetPolygonPathElements returns an array of variables of type PathElement. This array contains all path elements of the polygon (Line, Arc, Clothoid) in the same sequence as they appear in the polygon. On return, the dimension of the array is always starts from 1 To the number of Path elements that form the polygon. The method can not be used on a polygon that is formed by one circle.

Example

```

Dim A As Application
Dim D As Document
Dim Sel As Selection
Dim Counter As Long
Dim NrLines As Long, NrArcs As Long, NrClothoids As Long
Dim Obj As CadObject
Dim Elements() As PathElement
Dim Poly As Polygon

Set A= New Application
A.GetActiveDocument D
D.GetSelection Sel
If Sel.Count(ptPolygon) = 1 then
    Sel.GetFirst ptPolygon, Obj
    Obj.GetPolygon Poly
    Poly.GetPolygonPathElements Elements
    For Counter = 1 TO UBound(Elements)
        Select case Elements(Counter).ElementType
            case ptLine
                NrLines = NrLines + 1
            case ptArc
                NrArcs = NrArcs + 1
            case ptClothoid
                NrClothoids = NrClothoids + 1
            case else
            end select
    Next counter
    MsgBox "Lines : " & NrLines & VBCRLF & "Arcs : " & NrArcs & VBCRLF & _
        "Clothoids : " & NrClothoids

Else
    MsgBox "Select 1 Polygon"
End If

```

AlignPolygon

Parameters

Horizontal As Boolean, Origin As XY, Angle As Double

Description

Aligns the hatching of the polygon.

If Horizontal = True, then the hatching will become horizontal to the page and have an absolute origin which is the same for all unaligned hatchings. In this case, the other parameters are ignored.

If Horizontal = False, then the origin of the hatching is defined by the parameter Origin, and the angle by the parameter Angle.

Example

```

Const pi = 3.1416
Dim Org As XY
Dim Poly As Polygon

```

```

Org.X = 10.0
Org.Y = 10.0
Poly.AlignPolygon False, Org, 45.0/180.0*pi ' hatching 45 deg.

```

GetPolygonAlignment

Parameters

Horizontal As Boolean, Origin As XY, Angle As Double

Description

This method returns the alignment of the hatching of the polygon.
For more details of the parameters see method AlignPolygon.

Example

```

Dim Org As XY
Dim Angle As Double
Dim Poly As Polygon
Dim Horizontal As Boolean

Org.X = 10.0
Org.Y = 10.0
Poly.GetPolygonAlignment Horizontal, Org, Angle

```

PointInsidePolygon

Parameters

Pos As XY

Description

This method returns one of the following values: ptInside, ptOutside, ptOnEdge, ptOnVertex.

Example

```

Dim Pos As XY
Dim Angle As Double
Dim Poly As Polygon
Dim Horizontal As Boolean

Pos.X = 10.0
Pos.Y = 10.0
If Poly.PointInsidePolygon(Pos) = ptInside Then
End If

```

Chapter 28 - Solid

Description

A Solid is a CadObject. A reference to an existing Pythagoras Solid can be obtained e.g. with the method GetSolid.

Properties

Color
Comment
Link
DisplayLevel
Layer
Overlay
CanBeDeleted
CanChangeOverlay
ObjectReference
Selected
Flagged
Container
Opaque
Intensity
Volume

Methods

GetDocument
GetSubDocument
SetData
DeleteData
GetCadObject
GetBottomPnts
GetTopPnts
GetBottomIdents
GetTopIdents

Properties

For a description of the properties and methods Color, TrueColor, Comment, Link, DisplayLevel, Layer, Overlay, CanBeDeleted, CanChangeOverlay, ObjectReference, Selected, Flagged, Container see object type CadObject.

Opaque

Returns

Boolean

Description

Sets or returns the attribute Opaque of the Solid. If Opaque = True, then the solid will be opaque.

Example

```
Dim S As Solid

S.Opaque = True
```

Intensity

Returns

Integer

Description

This property sets or returns the intensity at which the solid is displayed and printed. The intensity must be a value between 1 and 100. Intensity values less than 100 will make the solid partially transparent.

Example

```
Dim S As Solid

S.Intensity = 80
```


Volume

Returns

Double

Description

Returns the volume of the Solid.

Example

```
Dim S As Solid

MsgBox "Volume = " & S.Volume & " m3"
```

Methods

CadObjects can not be assigned to objects of a specific type.

So the statement :

```
Set S = Obj
```

is not valid. In stead, the statements GetPoint, GetLine, etc. need to be used.

For a description of the methods GetDocument, GetSubDocument, SetData, DeleteData see object type CadObject.

GetCadObject

Parameters

Obj As CadObject

Description

GetCadObject sets the reference to the Solid in a variable of type CadObject. Note that the reference is copied, not the object.

Example

```
Dim Obj As CadObject
Dim MySolid As Solid

MySolid.GetCadObject Obj           ' Same as : Set Obj = MySolid
```

GetBottomPnts

Parameters

nrPnts As Integer
pnts() As XYZ

Description

Returns the bottom points of the solid. The first point is not repeated as the last point.

Example

```
Dim A As Application
Dim Doc As Document
Dim allObjs As AllObjects
Dim DrawingOverlay As Overlay
Dim Object As CADObject
Dim s As Solid
Dim nrPnts As Integer
Dim pnts() As XYZ
Dim i As Integer

Set A = New Application
A.GetActiveDocument Doc
Doc.GetOverlay ptDrawing, DrawingOverlay
DrawingOverlay.GetAllObjects allObjs
allObjs.GetFirst pt3DSolid, Object

If not Object Is Nothing Then
    Object.GetSolid s
    s.GetBottomPnts nrPnts, pnts
    For i = 1 To nrPnts
        MsgBox pnts(i).x + " " + pnts(i).y + " " + pnts(i).z
    Next i
End if
```

GetTopPnts

Parameters

nrPnts As Integer
pnts() As XYZ

Description

Returns the top points of the solid. The first point is not repeated as the last point.

Example

```
Dim A As Application
Dim Doc As Document
Dim allObjs As AllObjects
Dim DrawingOverlay As Overlay
Dim Object As CADObject
Dim s As Solid
Dim nrPnts As Integer
Dim pnts() As XYZ
Dim i As Integer

Set A = New Application
A.GetActiveDocument Doc
Doc.GetOverlay ptDrawing, DrawingOverlay
DrawingOverlay.GetAllObjects allObjs
allObjs.GetFirst pt3DSolid, Object

If not Object Is Nothing Then
    Object.GetSolid s
    s.GetTopPnts nrPnts, pnts
    For i = 1 To nrPnts
        MsgBox pnts(i).x + " " + pnts(i).y + " " + pnts(i).z
    Next i
End if
```

GetBottomIdents

Parameters

nrIdents As Integer
idents() As String

Description

Returns the idents of the bottom points of the solid. The points have the same order as in GetBottomPnts. The first point is not repeated as the last point.

Example

```
Dim A As Application
Dim Doc As Document
Dim allObjs As AllObjects
Dim DrawingOverlay As Overlay
Dim Object As CADObject
Dim s As Solid
Dim nrPnts As Integer
Dim pnts() As XYZ
Dim i As Integer
Dim nrIdents As Integer
Dim idents() As String

Set A = New Application
A.GetActiveDocument Doc
Doc.GetOverlay ptDrawing, DrawingOverlay
DrawingOverlay.GetAllObjects allObjs
allObjs.GetFirst pt3DSolid, Object

If not Object Is Nothing Then
    Object.GetSolid s
    s.GetBottomPnts nrPnts, pnts
    s.GetBottomIdents nrIdents, idents
    For i = 1 To nrPnts
        MsgBox idents(i) + " " + pnts(i).x + " " + pnts(i).y + " " + pnts(i).z
    Next i
End if
```

GetTopIdents

Parameters

nrIdents As Integer
idents() As String

Description

Returns the idents of the top points of the solid. The points have the same order as in GetTopPnts. The first point is not repeated as the last point.

Example

```
Dim A As Application
Dim Doc As Document
Dim allObjs As AllObjects
Dim DrawingOverlay As Overlay
Dim Object As CADObject
Dim s As Solid
Dim nrPnts As Integer
Dim pnts() As XYZ
Dim i As Integer
Dim nrIdents As Integer
Dim idents() As String

Set A = New Application
A.GetActiveDocument Doc
Doc.GetOverlay ptDrawing, DrawingOverlay
DrawingOverlay.GetAllObjects allObjs
allObjs.GetFirst pt3DSolid, Object

If not Object Is Nothing Then
    Object.GetSolid s
    s.GetTopPnts nrPnts, pnts
    s.GetTopIdents nrIdents, idents
    For i = 1 To nrPnts
        MsgBox idents(i) + " " + pnts(i).x + " " + pnts(i).y + " " + pnts(i).z
    Next i
End If
```


Chapter 29 - Image

Description

An Image is a `CadObject`. A new Image object can be created with the `Overlay` method `CreateImage`. A reference to an existing Pythagoras Image can be obtained with e.g. the method `GetImage`.

Example

See `ConvertImage`

Properties

`Color`
`Comment`
`Link`
`DisplayLevel`
`GroupId`
`Layer`
`Overlay`
`CanBeDeleted`
`CanChangeOverlay`
`ObjectReference`
`Selected`
`Flagged`
`Container`
`Data`
`Monochrome`
`ColorDepth`
`Width`
`Height`
`DPMx`
`DPMY`
`WidthInPixels`
`HeightInPixels`
`Position`
`FileName`
`NameRelative`
`Intensity`
`Inverted`
`Transparent`

Methods

`GetDocument`
`GetSubDocument`
`SetData`
`DeleteData`
`GetCadObject`
`SetReduce_Enlarge`
`GetReduce_Enlarge`
`SetScale`
`GetScale`
`GetOrientation`
`SetOrientation`
`PixelRect`
`PositionRect`
`ImageData`
`ImageDataSize`

Properties

For a description of the properties and methods `Color`, `TrueColor`, `Comment`, `Link`, `DisplayLevel`, `GroupId`, `Layer`, `Overlay`, `CanBeDeleted`, `CanChangeOverlay`, `ObjectReference`, `Selected`, `Flagged`, `Container`, `Data`, `GetQueryData` see object type `CadObject`.

Monochrome

Returns

`Boolean`

Description

This property returns True if the image is monochrome.

Example

```
Dim I As Image

If I.Monochrome = True Then
    I.Inverted = True
Else
    ' Do Something
End If
```

ColorDepth

Returns

Integer

Description

This property returns one of the following values : ptMonochrome, ptPalet256, ptTrueColor or ptQuadColor. For images created by Pythagoras v21 and higher always ptQuadColor is returned.

Example

```
Dim I As Image

If I.ColorDepth = ptTrueColor Then
    'Do Something
End If
```

Width & Height

Returns

Double

Description

These properties return the width / height of the original unscaled image in paper dimensions. This value is calculated by Pythagoras based on the DPI and on the number of pixels of the image.

Example

```
Dim I As Image
Dim W As Double
Dim H As Double

W = I.Width
H = I.Height
```

DPMx & DPMY

Returns

Double

Description

These properties return the resolution at which the image was scanned expressed in dots (pixels) per meter.

Example

```
Dim I As Image
Dim DPMx As Double
Dim DPMY As Double

DPMx = I.DPMx
DPMY = I.DPMY
```

WidthInPixels & HeightInPixels

Returns

Long

Description

These properties return the number of pixels of the image.

Example

```
Dim I As Image
```

```

Dim x As Long
Dim y As Long

x = I.WidthInPixels
y = I.HeightInPixels

```

Position

Returns
XY

Description

This property returns the position of the top-left corner of the image.
Possible clipping is not taken into account. If required see method PositionRect.

Example

```

Dim I As Image
Dim Pnt As XY

Pnt = I.Position

```

FileName

Returns
String

Description

This property returns the name of the .ptif (or .img) file as stored in the document. If the name refers to an .img file, the name can contain either the full path name, or a relative name.
If the image is stored in the Pythagoras document, this property returns the short name (without path) of the original image file.

Example

```

Dim I As Image
Dim Name As String

Name = I.FileName

```

NameRelative

Returns
Boolean

Description

This property returns True if the name of the .ptif (or .img) file is not a full path name.
When setting this property to True, the full path name will be made relative.

Example

```

Dim I As Image
Dim Name As String

If I.NameRelative = False Then
    I.NameRelative = True
End If

```

Intensity

Returns
Integer

Description

This property sets or returns the intensity at which the image is displayed and printed.
The intensity must be a value between 1 and 100. Intensity values less than 100 will make the whole image partially transparent.

Example

```

Dim I As Image

I.Intensity = 80

```


Inverted

Returns

Boolean

Description

This property sets or returns the *invert image* attribute. This property may only be used on monochrome images.

Example

```
Dim I As Image

If I.Monochrome = True Then
    I.Inverted = True
End If
```

Transparent

Returns

Boolean

Description

This property sets or returns the attribute that controls transparency of some parts of the image. When this attribute is set, then pixels of pure white color are treated as transparent.

This attribute has no effect on images where ColorDepth=ptQuadColor because transparency of every pixel is stored inside the pixel itself.

Example

```
Dim I As Image

If I.Transparent = True Then
    I.Transparent = False
End If
```

Methods

CadObjects can not be assigned to objects of a specific type.

So the statement :

```
Set P = Obj
```

is not valid. In stead, the statements GetPoint, GetLine, etc. need to be used.

For a description of the methods GetDocument, GetSubDocument, SetData, DeleteData see object type CadObject.

GetCadObject

Parameters

Obj As CadObject

Description

GetCadObject sets the reference to the Image in a variable of type CadObject. Note that the reference is copied, not the object.

Example

```
Dim Obj As CadObject
Dim I As Image

I.GetCadObject Obj ' Same as : Set Obj = I
```

SetReduce_Enlarge

Parameters

ScaleX As Double, ScaleY As Double

Description

SetReduce_Enlarge sets the new scale factors (X and Y) of the image. So the old scale factors have no effect on the result. The scale factors are expressed in %.

Example

```
Dim I As Image
```

```
I.SetReduce_Enlarge 150.0, 150.0
```

GetReduce_Enlarge

Parameters

ScaleX As Double, ScaleY As Double

Description

GetReduce_Enlarge returns both the factors (X and Y) by which the image is scaled. The scale factors are expressed in %.

Example

```
Dim I As Image
Dim ScaleX As Double, ScaleY As Double

I.SetReduce_Enlarge ScaleX, ScaleY
```

SetScale

Parameters

ScaleDependent As Boolean, Scale As Double

Description

SetScale sets the real scale of the image, e.g. at 1/10000, if the parameter ScaleDependent is set True. If the parameter ScaleDependent is False, the scale is ignored and the size of the image when printed will depend on the DPM, the number of pixels and on the Enlarge_Reduce factors.

Example

```
Dim I As Image
Dim ScaleDependent As Boolean
Dim Scale As Double

I.GetScale ScaleDependent, Scale
```

GetScale

Parameters

ScaleDependent As Boolean, Scale As Double

Description

GetScale returns the real scale of the image, if the parameter ScaleDependent is returned True. If the parameter ScaleDependent returns False, the returned scale = 0.0.

Example

```
Dim I As Image
Dim ScaleDendent As Boolean
Dim Scale As Double

I.GetScale ScaleDependent, Scale
```

GetOrientation

Parameters

Oblique As Boolean, Angle As Double

Description

GetOrientation returns the angle of the image if the parameter Oblique returns True. The variable Oblique returns False if the image is horizontal. In that case the returned angle is also equal to zero.

Example

```
Dim I As Image
Dim Oblique As Boolean, Angle As Double

I.GetOrientation Oblique, Angle
```

SetOrientation

Parameters

Oblique As Boolean, Angle As Double

Description

SetOrientation sets the angle of the image if the image is oblique. The parameter Oblique must be set False if the image must remain horizontal. In that case the value of the parameter angle is ignored.

Example

```
Const pi = 3.1416
Dim I As Image

I.SetOrientation True, 90.0/360.0 * pi
```

PixelRect

Parameters

Clipped As Boolean

Returns

Rectangle

Description

This property returns the rectangle {1.0, 1.0, WidthInPixels, HeightInPixels}.

If the parameter Clipped is true, the rectangle of the clipped image as seen by the user is returned. So {1.0, 1.0, Width, Height} where Width and Height are the number of pixels of the clipped image.

Example

```
Dim I As Image
Dim Rect As Rectangle

Rect = I.PixelRect
```

PositionRect

Parameters

Clipped As Boolean

Returns

Rectangle

Description

This property returns in the position of the top-left and bottom-right pixels of the image in the drawing. This property is mainly intended for obtaining the position of georeferenced imaged. For other images a rectangle is returned anyway.

For images that are not scale dependent (see methods SetScale and GetScale), the property will return the rectangle of the image on the active sheet.

If the parameter Clipped is true, the rectangle of the clipped image as seen by the user is returned.

Example

```
Dim I As Image
Dim Rect As Rectangle
Dim Clipped As Boolean

Clipped = True
Rect = I.PositionRect(Clipped)
```

ImageData

Parameters

ColorDepth As Integer, Rect As Rectangle,
UpsideDown As Boolean, ApplyImageOperations As Boolean,
Data() As Byte

Description

ImageData returns the raw bytes of the image. The parameter ColorDepth may be either ptTrueColor or ptQuadColor, the first value may not be applied to images with ColorDepth=ptQuadColor. If this parameter is ptTrueColor, then Data will contain three bytes per pixel (RGB). Otherwise four (RGBA).

The parameter Rect specifies the part of the image that must be read.

The value of x1 and y1 must be ≥ 0 . If x2 or y2 of the rectangle is $\geq$ than respectively the width or height of the image, the data is padded with white.

If `UpsideDown = True`, the bottom line will be first, and the top line will be last.

If `ApplyImageOperations` is `True`, the image as it visible on the screen will be returned. In that case the parameter `Rect` is applied to the clipped image. In other words, point (1.0, 1.0) is the top left point of the clipped image.

Example

```
Dim I As Image
Dim ImageData() As Byte

Redim ImageData(1 To I.ImageData(ptQuadColor, False, I.PixelRect(False)))
I.ImageData ptQuadColor, I.PixelRect(False), False, False, ImageData
```

ImageDataSize

Parameters

`ColorDepth` As Integer, `Clipped` As Boolean, `Rect` As Rectangle

Returns

`Size` As Long

Description

`ImageDataSize` returns the number of bytes required to retrieve the part `Rect` at the given `ColorDepth` the image. The parameters `ColorDepth` and `Rect` must be the same as used by subsequent call to `ImageData`.

If the parameter `Clipped` is true, the size required by the clipped image as seen by the user is returned. See also method `ImageData` parameter `ApplyImageOperations`.

Example

See `ImageData`.

Chapter 30 - Compound

Description

A Compound is a CadObject, that groups a number of other Cad objects. A Compound can be created with the Overlay method CreateCompound. Only the Cad object types Point, Line, Arc, Circle, Curve, Clothoid, Text, Richtext, Path, Polygon, Image, Annotation and Solid can be elements of a compound.

All objects in a Compound must be in the same Overlay and in the same Subdocument. If the objects are in the overlay ptPaper, the objects must be in the same Sheet.

Example

```
Dim A As Application
Dim Doc As Document
Dim Drawing As Overlay
Dim Position As XYZ
Dim C As Compound
Dim P1 As Point
Dim P2 As Point
Dim P3 As Point

Set A = New Application
A.GetActiveDocument Doc
Doc.GetOverlay ptDrawing, Drawing
Position.X = -1.
Position.Y = 0.
Position.Z = 0.
Drawing.CreatePoint Position, P1
Position.X = 1.
Drawing.CreatePoint Position, P2
Position.X = 0.
Position.Y = 2.
Drawing.CreatePoint Position, P3
Drawing.CreateCompound C
Set P1.Container = C          ' Add point P1 to compound C
MsgBox "A compound contains " & C.NrOfElements & " elements"
Set P2.Container = C          ' Add two more points to compound C
Set P3.Container = C
MsgBox "A compound contains " & C.NrOfElements & " elements"
Set P1.Container = Nothing    ' Exclude point P1 from
MsgBox "A compound contains " & C.NrOfElements & " elements"
```

Properties

Comment
Color
TrueColor
DisplayLevel
Layer
Overlay
CanBeDeleted
ObjectReference
Selected
Flagged
AllFlagged
AllSelected
Data
Sheet
NrOfElements

Methods

GetCadObject
GetDocument
GetSubDocument
SetData
DeleteData
SetLink
GetQueryData
GetElements

Properties

Properties `Comment`, `Color`, `TrueColor`, `DisplayLevel` and `Layer` are read-only. For their description and description of properties `Overlay`, `CanBeDeleted`, `ObjectReference`, `Selected`, `Flagged`, `Data` and `Sheet` see object type `CadObject`.

AllFlagged

Value

Flag As Boolean

Description

Set Flagged bit of the compound and of all its elements.

Example

```
Dim Comp As Compound

Comp.AllFlagged = True
' Comp and all its elements obtain Flagged bit = True
```

AllSelected

Value

Selected As Boolean

Description

Set selection status of the compound and of all its elements.

Example

```
Dim Comp As Compound

Comp.AllSelected = False ' Comp and all its elements become
unselected
```

NrOfElements

Returns

Integer

Description

This property returns the number of elements of a compound.

Example

```
Dim C As Compound

MsgBox "Nr elements in compound = " & CStr(C.NrOfElements)
```

Methods

`CadObjects` can not be assigned to objects of a specific type.

So the statement :

```
Set C = Obj
```

is not valid. Instead, the statements `GetPoint`, `GetLine`, etc. need to be used.

For description of methods `GetDocument`, `GetSubDocument`, `SetData`, `DeleteData`, `SetLink` and `GetQueryData` see object type `CadObject`.

GetCadObject

Parameters

Obj As `CadObject`

Description

`GetCadObject` sets the reference to the compound in a variable of type `CadObject`. Note that the reference is copied, not the object.

Example

```
Dim Obj As CadObject
Dim Comp As Compound
```

```
Comp.GetCadObject Obj ' Same as : Set Obj = Comp
```

GetElements

Parameters

```
Objs() As CadObject
```

Description

GetElements returns the list of all members (CadObjects) in the compound. If there are no elements in the compound, an empty array is returned.

Example

```
Dim Objs() As CadObject  
Dim C As Compound  
  
C.GetElements Objs
```

Chapter 31 - Annotation

Description

An Annotation is a `CadObject`. A reference to an existing Pythagoras Annotation can be obtained with the method `GetAnnotation`.

Example

See method `GetAnnotation` of the object type `CadObject`.

Properties

`Link`
`Color`
`TrueColor`
`DisplayLevel`
`GroupId`
`Container`
`Layer`
`Overlay`
`CanBeDeleted`
`CanChangeOverlay`
`ObjectReference`
`Selected`
`Flagged`
`TextPosition`
`TextContent`
`TextResult`
`TextOrientation`
`TextAngle`
`TextStyle`
`TextWeight`
`TextPointSize`
`TextHorizontalAlignment`
`TextVerticalAlignment`
`TextUnderline`
`TextBorder`
`TextLineSpacing`
`TextOpaque`
`TextFreeze`
`TextColor`
`AnnotationType`
`SymbolType`
`SymbolColor`
`SymbolScale`
`NrOfLines`

Methods

`GetDocument`
`GetSubDocument`
`GetCadObject`
`SetData`
`DeleteData`
`GetTextAngle`
`GetTextFont`
`SetTextFont`
`GetReferencedObjects`
`GetSymbolStyle`
`SetSymbolStyle`

Properties

For a description of the properties and methods `Link`, `Color`, `TrueColor`, `DisplayLevel`, `GroupId`, `Container`, `Layer`, `Overlay`, `CanBeDeleted`, `CanChangeOverlay`, `ObjectReference`, `Selected`, `Flagged`, see object type `CadObject`.

TextPosition

Returns

`XY`

Description

This property sets or returns the insertion point of the text.
Insertion point : a text is aligned relative to its insertion point.

Example

```
Dim An As Annotation
Dim Pos As XY

Pos.X = 20.0
Pos.Y = 30.0
An.TextPosition = Pos
```

TextContent

Returns

String

Description

This property sets or returns the content of the text.

Example

```
Dim An As Annotation

An.TextContent = "-^LineLength-"
```

TextResult

Returns

String

Description

This property returns the content of the text, but the placeholders (e.g. ^LineLength) are replaced by their values.

Example

```
Dim An As Annotation

MsgBox(An.TextResult)
```

TextOrientation

Returns

Integer

Description

This property sets or returns the orientation of the text.

TextOrientation = ptTextHorizontal: The text will always appear horizontal independent of page orientation or angle.

TextOrientation = ptTextOrientationReadable: The angle of the displayed or printed texts will change if the text would come upside down. If the angle of the texts is in 2nd or 3rd quadrant, the texts are turned over 180 degrees.

TextOrientation = ptTextOrientationAbsolute: The texts are always displayed at the given angle.

The possible text orientations depend on the annotation type. A pointer annotation can have the three possible text orientations. The other annotation types can only have ptTextOrientationReadable and ptTextOrientationAbsolute. When these are set to ptTextHorizontal, the request will be ignored.

See also AnnotationType.

Example

```
Dim An As Annotation

An.TextAngle = 1.3
An.TextOrientation = ptTextOrientationReadable
```

TextAngle

Returns

Double

Description

This property returns the angle of the text. The angle is expressed in radians.

Example

```
Dim An As Annotation

MsgBox (An.TextAngle)
```

TextStyle

Returns

Integer

Description

This property sets or returns the style of the text.

TextStyle = ptUpright : Normal

TextStyle = ptItalic : Italic

Example

```
Dim An As Annotation

An.TextStyle = ptUpright
```

TextWeight

Returns

Integer

Description

This property sets or returns the weight of the text.

TextWeight = ptLight : Light

TextWeight = ptMedium : Normal

TextWeight = ptBold : Bold

TextWeight = ptExtraBold : Extra Bold

Not all fonts can be represented Light or Extra Bold.

Example

```
Dim An As Annotation

An.TextWeight = ptBold
```

TextPointSize

Returns

Integer

Description

This property sets or returns the point size of the text. Value of PointSize must be between 1 and 1000.

Example

```
Dim An As Annotation

An.TextPointSize = 20
```

TextHorizontalAlignment

Returns

Integer

Description

This property sets or returns the horizontal alignment of the text.

TextAlignment = ptAlignLeft : left aligned.

TextAlignment = ptAlignCenter : centered.

TextAlignment = ptAlignRight : right aligned.

Example

```
Dim An As Annotation

An.TextHorizontalAlignment = ptAlignRight
```

TextVerticalAlignment

Returns

Integer

Description

This property returns the vertical alignment of the text. This is always centered.
TextAlignment = ptAlignCenter : centered.

Example

```
Dim An As Annotation

MsgBox (An.TextVerticalAlignment)
```

TextUnderline

Returns

Integer

Description

This property sets or returns the underlining of the text.
TextUnderline = ptUnderlineNone : no underlining.
TextUnderline = ptUnderlineSingle : single underlining.
TextUnderline = ptUnderlineDouble : double underlining.
TextUnderline = ptUnderlineDashed : underlining dashed line.

Example

```
Dim An As Annotation

An.TextUnderline = ptUnderlineSingle
```

TextBorder

Returns

Integer

Description

This property sets or returns the border of the text.
TextBorder = ptBorderNone : no border.
TextBorder = ptBorderSingle : single border.
TextBorder = ptBorderDouble : double border.

Example

```
Dim An As Annotation

An.TextBorder = ptBorderSingle
```

TextLineSpacing

Returns

Integer

Description

This property sets or returns the line spacing of the text.
LineSpacing = ptSpacingSmall : small line spacing.
LineSpacing = ptSpacingNormal : normal line spacing.
LineSpacing = ptSpacingLarge : large line spacing.

Example

```
Dim An As Annotation

An.TextLineSpacing = ptSpacingSmall
```

TextOpaque

Returns

Boolean

Description

This property sets or returns the transparency of the text.

If = True : Opaque

Example

```
Dim An As Annotation

An.Opaque = True
```

TextFreeze

Returns

Boolean

Description

This property sets or returns the possibility to change the text.

If = True : TextFreeze

Example

```
Dim An As Annotation

An.TextFreeze = True
```

TextColor

Returns

Integer

Description

This property sets and returns the color of the text of the annotation. The color is a sequence number in the standard palette of 256 colors. When the true color of an object is not in the palette, then the nearest color is returned.

The color of both the text and symbol of the annotation, can be set with the inherited property `Color` of `CadObject`. As return value this property will give the color of the text of the annotation.

Example

```
Dim An As Annotation

An.SymbolColor = ptRed
```

AnnotationType

Returns

Integer

Description

This property returns the type of the annotation.

```
AnnotationType = ptPointerAnnotation (= 1).
AnnotationType = ptRangeAnnotation (= 2).
AnnotationType = ptAngleAnnotation (= 3).
AnnotationType = ptRadiusAnnotation (= 4).
AnnotationType = ptDeltaXAnnotation (= 5).
AnnotationType = ptDeltaYAnnotation (= 6).
```

Example

```
Dim An As Annotation

If An.AnnotationType = ptPointerAnnotation Then
    ...
End If
```

SymbolType

Returns

Integer

Description

This property returns the type of the symbol.

SymbolType = ptPointerSymbol.

SymbolType = ptRangeSymbol.

Example

```
Dim An As Annotation

If An.SymbolType = ptPointerAnnotation Then
    ...
End If
```

SymbolColor

Returns

Integer

Description

This property sets and returns the color of the symbol. The color is a sequence number in the standard palette of 256 colors. When the true color of an object is not in the palette, then the nearest color is returned.

The color of both the text and symbol of the annotation, can be set with the inherited property Color of CadObject. As return value this property will give the color of the text of the annotation.

Example

```
Dim An As Annotation

An.SymbolColor = ptRed
```

SymbolScale

Returns

Double

Description

This property sets and returns the scale factor by which the symbol is scaled. The scale factor is expressed in %. The scale factor may be negative. This means that the symbol is mirrored around the corresponding axis.

Example

```
Dim An As Annotation

An.SymbolScale = 90
```

NrofLines

Returns

Integer

Description

This property returns the number of lines of the text.

Example

```
Dim An As Annotation

If An.NrofLines > 1 Then
    MsgBox "Multiline annotation text"
End If
```

Methods

CadObjects can not be assigned to objects of a specific type.
So the statement :

Set An = Obj
is not valid. In stead, the statements GetPoint, GetAnnotation, etc. need to be used.
For a description of the methods GetDocument, GetSubDocument, SetData, DeleteData
see object type CadObject.

GetCadObject

Parameters

Obj As CadObject

Description

GetCadObject sets the reference to the Annotation in a variable of type CadObject. Note that the reference is copied, not the object.

Example

```
Dim Obj As CadObject
Dim An As Annotation

An.GetCadObject Obj          ' Same as : Set Obj = An
```

GetTextAngle

Parameters

AngleAsDisplayed As Boolean, Angle As Double

Description

GetAngle returns the angle of the text. If the orientation of the text = ptTextHorizontal, then the returned angle is always = 0.0.

If the input parameter AngleAsDisplayed = True, then the corrected angle (+180°) will be returned if the orientation of the text = ptTextOrientationReadable, and the text without correction would have been displayed upside down. The angle is expressed in radians.

Example

```
Dim An As Annotation
Dim Angle As Double

An.GetTextAngle True, Angle
```

GetTextFont

Parameters

FontName As String, CodePage As Integer

Description

GetTextFont returns the Font name and the codepage of the text.

Example

```
Dim An As Annotation
Dim FontName As String
Dim CodePage As Integer

An.GetTextFont FontName, CodePage
```

SetTextFont

Parameters

Font As Variant, Optional IsDingbat As Boolean

Description

Font should be either ptPythagorasVectorFont or a name of a font installed in Windows. All text objects in the set will be painted in this font. Specified name of the font should contain only characters with codes < 128 (basic ASCII). Concerning IsDingbat parameter see description of Text.SetFont method.

Example

```
Dim Sel As SelectedObjects

Sel.SetTextFont "Arial"
```

SetTextFont (obsolete format)

Parameters

FontName As String, CodePage As Integer

Description

This format remains in VBA only to execute code written for VBA version ≤ 13 . All text objects in the set will be painted in a font with this name. Concerning CodePage parameter see description of Text.SetFont method (obsolete format).

GetReferencedObjects

Parameters

RefObj1, RefObj2 As CadObject

Description

GetReferencedObjects returns the referenced objects of the annotation.

Example

```
Dim An As Annotation
Dim RefObj1 As CadObject
Dim RefObj2 As CadObject

An.GetReferencedObjects RefObj1, RefObj2
```

GetSymbolStyle

Parameters

SymbolStyle As Style

Description

This property returns the style of the symbol of the annotation.

Example

```
Dim An As Annotation
Dim S As Style

An.GetSymbolStyle S
```

SetSymbolStyle

Parameters

SymbolStyle As Style, SymbolType As Integer

Description

This property sets the style of the symbol of the annotation. The SymbolStyle.StyleId should belong to the SymbolType.

SymbolType = ptPointerSymbol.

SymbolType = ptRangeSymbol.

SymbolStyle.StyleId can be

ptRangeOnlyText: range text only.

ptExternalRange: range external.

ptFullRange: range simple.

ptFullRangeDotted: range dotted.

ptFullRangeSlashes: range with slashes.

ptFullRangeVertSlashes: range with vertical slashes.

ptFullRangeExtensions: range with extensions.

ptFullRangePullUp: range with dotted extensions.

ptIntermittedRange: intermitted range simple.

ptIntermittedRangeDotted: intermitted range dotted.

ptIntermittedRangeSlashes: intermitted range with slashes.

ptIntermittedRangeVertSlashes: intermitted range with vertical slashes.

ptIntermittedRangeExtensions: intermitted range with extensions.

ptIntermittedRangePullUp: intermitted range with dotted extensions.

ptPointerTextOnly: pointer text only.
 ptPointerOnlyTextOnObject: pointer text only centered.
 ptSimplePointerLeftOfText: pointer to vertical side.
 ptSimplePointerUnderText: pointer to horizontal side.
 ptPointerToCorner: pointer to corner.
 ptPointerBroken: pointer broken.
 ptPointerBrokenToVerticalSide: pointer broken to vertical side.
 ptPointerWithVerticalSlash: pointer with vertical slash.
 ptSimplePointerUnderlinedText: pointer with underline.
 ptRadiusPointer: pointer radius.
 ptCirclePointer: pointer with circle.
 ptPointerSideWaysText: pointer side ways text
 ptPointerManHoleBroken: pointer manhole broken
 ptPointerManHoleBrokenToVerticalSide: pointer manhole broken to vertical side

Example

```

Dim An As Annotation
Dim S As Style

S.Symbol = FALSE
S.Group = ""
S.item = ""
S.StyleId = ptCirclePointer
An.SetSymbolStyle S, ptPointerSymbol
  
```


Chapter 32 - CoordinateSystem

Description

A CoordinateSystem is either a CadObject, in which case it is a user defined coordinate system or the coordinate system Local, Global, Page or Temporary coordinate system.

A CoordinateSystem can be obtained with the Document method GetActiveCoordinateSystem, with the CadObject method GetCoordinateSystem, or by running through the object CoordinateSystems (GetFirst and GetNext).

Example

```
Dim A As Application
Dim Doc As Document
Dim CS As CoordinateSystem

Set A = New Application
A.GetActiveDocument Doc
Doc.GetActiveCoordinateSystem CS
MsgBox CS.Name
```

Properties

Overlay
TypeOfCS
Name
Origin
ZOfOrigin
XAxisPnt
YaxisPnt
TransformationParameters

Methods

SetActive

Properties

Overlay

Returns

Integer

Description

This property returns the type of overlay of the coordinate system. The value is either ptDrawing or ptPaper.

Example

```
Dim CS As CoordinateSystem

If CS.Overlay = ptPaper Then
End If
```

TypeOfCS

Returns

Integer

Description

This property returns one of the following values : ptLocalCS, ptUserCS, ptGlobalCS, ptPageCS, ptTempCS.

Example

```
Dim CS As CoordinateSystem

If CS.TypeOfCS = ptUserCS Then
    MsgBox "User coordinate system is active"
End If
```

Name

Returns

String

Description

This property returns the name of the coordinate system.

Example

```
Dim A As Application
Dim Doc As Document
Dim CS As CoordinateSystem

Set A = New Application
A.GetActiveDocument Doc
Doc.GetActiveCoordinateSystem CS
MsgBox CS.Name
```

Origin

Returns

XY

Description

This property returns the origin (x, y) of the coordinate system.

Example

```
Dim CS As CoordinateSystem
Dim Pnt As XY

Pnt = CS.Origin
```

ZOfOrigin

Returns

Double

Description

Returns the elevation of the Coordinate system.

Example

```
Dim CS As CoordinateSystem
Dim Z As Double

Z = CS.ZOfOrigin
```

XAxisPnt

Returns

XY

Description

This property returns a point on the X-axis of the coordinate system.

Example

```
Dim CS As CoordinateSystem
Dim Pnt As XY

Pnt = CS.XAxisPnt
```

YAxisPnt

Returns

XY

Description

This property returns a point on the Y-axis of the coordinate system.

Example

```
Dim CS As CoordinateSystem
Dim Pnt As XY

Pnt = CS.YAxisPnt
```

TransformationParameters

Returns

Transformation

Description

This property returns the transformation parameters. These transformation parameters can directly be used to convert local coordinates to coordinates in the coordinate system or vice versa. For more information about the data type and its use, see [Pythagoras Data Type Transformation](#), and the object `Units`.

Example

```
Dim CS As CoordinateSystem
Dim TrPars As Transformation
Dim U As Units

TrPars = CS.TransformationParameters
Set U = New Units
U.SetTransformation TrPars
```

Methods

SetActive

Parameters

CS As CoordinateSystem

Description

SetActive makes the coordinate system the active one.

Example

```
Dim CS As CoordinateSystem

CS.SetActive
```

Chapter 33 - CoordinateSystems

Description

CoordinateSystems is the set of all coordinate systems. This Object allows enumerating all coordinate systems in either the paper or drawing overlay. CoordinateSystems is obtained from the object Overlay by using the method GetCoordinateSystems.

Example

```
Dim A As Application
Dim Doc As Document
Dim Ovl As Overlay
Dim CSs As CoordinateSystems
Dim CS As CoordinateSystem
Dim i As Integer

Set A = New Application
A.GetActiveDocument Doc
Doc.GetOverlay ptDrawing, Ovl
Ovl.GetCoordinateSystems CSs
CSs.GetFirst CS
For i = 1 To CSs.NrOfCoordinateSystems
    MsgBox CS.Name
    CSs.GetNext CS
Next i
```

Properties

NrOfCoordinateSystems

Methods

GetFirst
GetNext

Properties

NrOfCoordinateSystems

Returns

Integer

Description

This property returns the number of coordinate systems in the set.

Example

See introduction.

Methods

GetFirst

Parameters

CS As CoordinateSystem

Description

GetFirst returns the first coordinate system in the set.

Example

See introduction.

GetNext

Parameters

CS As CoordinateSystem

Description

GetNext returns the following coordinate system in the set.

Example

See introduction.

Chapter 34 - CoordinateReferenceSystem

Description

A CoordinateReferenceSystem is either a geographical reference system (e.g. WGS84) or a specific projection system applied to a geographical reference system (e.g. Belgian Lambert 72). The CoordinateReferenceSystem can be obtained from CoordinateReferenceSystems using the methods GetFirst, GetNext, GetCoordinateReferenceSystem.

Example

```
Dim A As Application
Dim CRSs As CoordinateReferenceSystems
Dim CRS As CoordinateReferenceSystem

Set A = New Application
A.GetCoordinateReferenceSystems CRSs
CRSs.GetCoordinateReferenceSystem "UTM Z31", CRS
```

Properties

Name
LocalName
ShortName
CRSType

Methods

TransformXYTo
TransformToPossible

Properties

Name

Returns

String

Description

This property returns the name that identifies the CoordinateReferenceSystem. Each CRS has a unique name independent of the language version of Pythagoras.

Example

```
Dim CRS As CoordinateReferenceSystem

MsgBox CRS.Name
```

LocalName

Returns

String

Description

This property returns the localised name of the CoordinateReferenceSystem. The local name depends on the language version of Pythagoras. The local name is used in dialog boxes in Pythagoras.

Example

```
Dim CRS As CoordinateReferenceSystem

MsgBox CRS.LocalName
```

ShortName

Returns

String

Description

This property returns the short name of the CoordinateReferenceSystem. The short name depends on the language version of Pythagoras. The short name is used in the control panel.

Example

```
Dim CRS As CoordinateReferenceSystem
```

MsgBox CRS.ShortName

CRSType

Returns

Integer

Description

This property returns one of the following values:

ptProjection
ptGeographical

This property can be used to find out if a Coordinate Reference System is either only a Geographical Coordinate System returning Latitude,Longitude or a Projection System based on a Geographical Coordinate System.

Example

```
Dim CRS As CoordinateReferenceSystem

If CRS.CRSType = ptGeographical Then
    MsgBox "Please use Latitude/Longitude"
End If
```

Enabled

Returns

Boolean

Description

This property can be used to see if a CRS is enabled in the Preferences of Pythagoras.

Example

```
Dim A As Application
Dim CRSs As CoordinateReferenceSystems
Dim CRS As CoordinateReferenceSystem

Set A = New Application
A.GetCoordinateReferenceSystems CRSs

CRSs.GetFirst CRS
Do While Not (CRS Is Nothing)
    If CRS.Enabled Then
        MsgBox CRS.Name
    End If
    CRSs.GetNext CRS
Loop
```

Methods

TransformXYTo

Parameters

DestinationCRS As CoordinateReferenceSystem,
Pnt As XY,
Optional Transformation As HelmertTransformation,
Optional SmoothResiduals As Integer,
Optional TiePointSetName As String

Returns

XY

Description

TransformXYTo transforms the coordinates Pnt from the current CRS to the DestinationCRS.

For geographical systems, XY is used to pass/return the latitude, longitude values. These angle values must be passed in degrees decimal. Sequence: x = latitude, y = longitude. For projection systems the (x,y) coordinates are passed/returned in meter.

If TiepointSetName is the name of a tie point set in the library (not in the document!), and if its coordinate systems match, this tie point set is used to correct the transformation results.

Example

```
Dim A As Application
Dim CRSs As CoordinateReferenceSystems
Dim SourceCRS As CoordinateReferenceSystem
Dim DestinationCRS As CoordinateReferenceSystem
Dim Pnt As XY
Dim UTMCoord As XY

set A = New Application

A.GetCoordinateSystems CRSs
CRSs.GetCoordinateReferenceSystem "WGS84", SourceCRS
CRSs.GetCoordinateReferenceSystem "UTM N Z31", DestinationCRS
Pnt.x = 51.3146424 'Latitude
Pnt.y = 4.8776564 'Longitude
If SourceCRS.TransformToPossible(DestinationCRS) then
    UTMCoord = SourceCRS.TransformXYTo(DestinationCRS, Pnt)
End if
```

TransformToPossible

Parameters

CRS As CoordinateReferenceSystem

Returns

Boolean

Description

TransformToPossible returns True if the transformation to CRS is possible.

Example

See TransformXYTo

Chapter 35 - CoordinateReferenceSystems

Description

CoordinateReferenceSystems is the set of all Coordinate Reference Systems. This Object allows enumerating all coordinate reference systems. CoordinateReferenceSystems is obtained from the object Application by using the method GetCoordinateReferenceSystems.

Example

```
Dim A As Application
Dim CRSs As CoordinateReferenceSystems
Dim CRS As CoordinateReferenceSystem

Set A = New Application
A.GetCoordinateReferenceSystems CRSs
CRSs.GetFirst CRS
Do While Not (CRS is Nothing)
    MsgBox CRS.Name & vbCrLf & CRS.LocalName
    CRSs.GetNext CRS
Loop
```

Properties

NrOfCoordinateReferenceSystems

Methods

GetFirst
GetNext
GetCoordinateReferenceSystem

Properties

NrOfCoordinateReferenceSystems

Returns

Long

Description

This property returns the total number of Coordinate Reference Systems.

Example

```
Dim A As Application
Dim CRSs As CoordinateReferenceSystems

Set A = New Application
A.GetCoordinateReferenceSystems CRSs
MsgBox CRSs.NrOfCoordinateReferenceSystems
```

Methods

GetFirst

Parameters

CRS As CoordinateReferenceSystem

Description

GetFirst returns the first coordinate reference system.

Example

See introduction.

GetNext

Parameters

CRS As CoordinateReferenceSystem

Description

GetNext returns the next coordinate reference system.

Example

See introduction.

GetCoordinateReferenceSystem

Parameters

Name As String
CRS As CoordinateReferenceSystem

Description

GetCoordinateReferenceSystem returns the coordinate reference system with the given name.
If a CRS with the given name does not exist, the return value CRS will be equal to Nothing.

Example

```
Dim A As Application
Dim CRSs As CoordinateReferenceSystems
Dim CRS As CoordinateReferenceSystem

Set A = New Application
A.GetCoordinateReferenceSystems CRSs
CRSs.GetCoordinateReferenceSystem "UTM N Z32", CRS
If CRS Is Nothing Then
    MsgBox "CRS does not exist"
End If
```

Chapter 36 - TiePointsSetCollection

Description

TiePointsSetCollection is the collection of all TiePointsSets in either a document or in the system Library. This Object allows enumerating allTiePointsSets. TiePointsSetCollection is obtained from the object Documentor from Application by using the method GetTiePointsSetCollection.

Example

```
Dim A As Application
Dim TPC As TiePointsSetCollection
Dim TPS As TiePointsSet

Set A = New Application
A.GetTiePointsSetCollection TPC
TPC.GetFirst TPS
Do While Not (TPS is Nothing)
    MsgBox TPS.Name
    TPC.GetNext TPS
Loop
```

Properties

Count

Methods

GetFirst
GetNext
GetTiePointsSet

Properties

Count

Returns

Integer

Description

This property returns the number of Tie Points Sets in the collection.

Example

```
Dim Doc As Document
Dim TPC As TiePointsSetCollection

Doc.GetTiePointsSetCollection TPC
MsgBox TPC.Count
```

Methods

GetFirst

Parameters

TPS As TiePointsSet

Description

GetFirst returns the first set of tie points in TiePointsCollection.

Example

See introduction.

GetNext

Parameters

TPS As TiePointsSet

Description

GetNext returns the next set of tie points in the collection of Tie Point Set.

Example

See introduction.

GetTiePointsSet

Parameters

Name As String
TPS As TiePointsSet

Description

GetTiePointSet returns the set of tie points with the given name. If a set with the given name does not exist, the return value will be equal to Nothing.

Example

```
Dim A As Application
Dim TPC As TiePointsSetCollection
Dim TPS As TiePointsSet

Set A = New Application
A.GetTiePointsSetCollection TPC
TPS = TPC.GetTiePointsSet "Test1", TPS
If TPS Is Nothing Then
    MsgBox "Tie points set does not exist"
End If
```

Chapter 37 - TiePointsSet

Description

A TiePointsSet is set of pairs of coordinates in different projection systems for the same geographical point. Such set can be used for improvement the accuracy of transformation between the different projection systems. The TiePointsSet can be obtained from TiePointsCollection using the methods GetFirst, GetNext.

Example

```
Dim A As Application
Dim TPC As TiePointsSetCollection
Dim TPS As TiePointsSet

Set A = New Application
A.GetTiePointsSetCollection TPC
TPC.GetTiePointsSet "Test1", TPS
If TPS Is Nothing Then
    MsgBox "Tie points set does not exist"
End If
```

Properties

Name
NrOfElements
CRSFrom
CRSTo

Methods

GetFirstElement
GetNextElement

Properties

Name

Returns

String

Description

This property returns the name that identifies the TiePointsSet.

Example

```
Dim TPS As TiePointsSet

MsgBox TPS.Name
```

NrOfElements

Returns

Integer

Description

This property returns the number of points (pairs) in the TiePointsSet.

Example

```
Dim TPS As TiePointsSet

MsgBox TPS.NrOfElements
```

CRSFrom

Returns

CoordinateReferenceSystem

Description

This property return theCoordinate reference system of the first point of each pair in the TiePointsSet.

Example

```
Dim TPS As TiePointsSet
Dim CRS As CoordinateReferenceSystem
```

```
Set CRS = TPS.CRSFrom
MsgBox CRS.Name
```

CRSTo

Returns

CoordinateReferenceSystem

Description

This property return theCoordinate reference system of the second point of each pair in the TiePointsSet.

Example

```
Dim TPS As TiePointsSet
Dim CRS As CoordinateReferenceSystem

Set CRS = TPS.CRSTo
MsgBox CRS.Name
```

Methods

GetFirstElement

Parameters

None

Returns

Ident As String, P1 As XY, P2 As XY

Description

Returns the first element of the tie points set.

Example

```
Dim TPS As TiePointsSet
Dim Ident As String
Dim P1 As XY
Dim P2 As XY

TPS.GetFirstElement Ident, P1, P2
Do While Not (Ident = "")
MsgBox Ident & " " & CStr(P1.x) & " " & CStr(P1.y)
TPS.GetNextElement Ident, P1, P2
Loop
```

GetNextElement

Parameters

None

Returns

Ident As String, P1 As XY, P2 As XY

Description

Returns the next element of the tie point set

Example

See GetFirstElement

Chapter 38 - Selection, FlaggedObjects, AllObjects

Description

Selection and FlaggedObjects are created by using the Document methods GetSelection, and GetFlaggedObjects. AllObjects is created by using the Overlay method GetAllObjects, Selection refers to the currently selected objects of the corresponding document. It does not necessarily refer to the selected objects in the active document.

Notes :

1. Per document, **only one** of each of the above sets can exist. Any other instance will refer to the same set.
2. The Selection object remains up-to-date with the situation on the screen. This means a.o. that when a Selection Object is defined globally and the user changes the selection between 2 macros, the Selection Object will reflect the new status.
3. The AllObjectsSelection remains up-to-date. (See note 2)
4. Flagged objects can only be accessed by Pythagoras VBA programs. They are useful to do operations on multiple objects with one statement. All Flagged objects need to be in the same overlay. Setting the first object flagged, will determine the overlay of the set. Setting an object of another overlay flagged will be ignored.

Example

```
Dim A As Application
Dim Doc1 As Document
Dim SelectionDoc1 As Selection
Dim FlaggedObjectsDoc1 As FlaggedObjects
Dim MyDrawing As AllObjects
Dim DrawingOverlay As Overlay

Set A = New Application
A.GetActiveDocument Doc1
Doc1.GetSelection SelectionDoc1
Doc1.GetFlaggedObjects FlaggedObjectsDoc1
Doc1.GetOverlay ptDrawing, DrawingOverlay
DrawingOverlay.GetAllObjects MyDrawing
```

Properties

Overlay
Sheet
Wallpaper
NrOfObjects
NrOfWholeObjects
Color
TrueColor
ImageIntensity
Comment
DisplayLevel
Layer
PointStyle
LineStyle
LineWidth
LineArrows
TextStyle
TextWeight
TextPointSize
TextHorizontalAlignment
TextVerticalAlignment
TextUnderline
TextBorder
TextLineSpacing
TextOpaque
TextOrientation
TextAngle
CurveCurvature
CurveMaxRounding
PolygonPattern
PolygonBorder
PolygonOpaque
PolygonBackgroundColor

PolygonBackgroundPattern
SolidOpaque
SolidIntensity
CanChangeOverlay
Container
AnnotationTextStyle
AnnotationTextWeight
AnnotationTextPointSize
AnnotationTextHorizontalAlignment
AnnotationAnnotationTextUnderline
AnnotationTextBorder
AnnotationTextLineSpacing
AnnotationTextOpaque
AnnotationTextOrientation
AnnotationTextColor
AnnotationSymbolColor
AnnotationSymbolScale

Methods

Count
GetFirst
GetNext
ChangeOverlay
Copy
Cut
Clear
Move
Rotate
Group
Ungroup
Unpack
DeleteData
AddAnnotation
AdjustDisplayLevel
SetZ
ScalePoints
ScaleLines
ScaleTexts
ScalePolygons
AlignPolygons
ConvertToPolylines
ConvertObjectsToPolylines
GetFirstPoint
GetNextPoint
IntersectingLines
Intersections
UnderShoot
OverShoot
SetTextFont
SetAnnotationTextFont
SetAnnotationSymbolStyle
Transform
ClearSet
MakeSet
MakeSetDB
MakeSetTextual
Toggle
Include
Exclude
InSet
ApplyThematics
GetThematics
RemoveThematics
GetTerrainModel

Properties

Overlay

Returns

OverlayType As Integer

Description

This property returns the overlay in which the objects of the set are located.

ptPaper if the Set refers to the Paper Overlay.

ptDrawing if the Set refers to the Drawing Overlay.

Example

```
Dim A As Application
Dim Doc As Document
Dim MyDrawing As AllObjects
Dim DrawingOverlay As Overlay
Dim OverlayType As Integer

Set A = New Application
A.GetActiveDocument Doc
Doc.GetOverlay ptDrawing, DrawingOverlay
DrawingOverlay.GetAllObjects MyDrawing
OverlayType = MyDrawing.Overlay
```

Sheet

Value

Sheet

Description

This property may be applied only to objects of AllObjects class.

The property returns a sheet the set contains objects of. If the set was defined for a whole overlay the property returns Nothing.

Example

```
Dim All As AllObjects, Sh As Sheet

Set Sh = All.Sht
```

Wallpaper

Value

Boolean

Description

Returns False if the object of class AllObjects was defined not to contain objects of wallpaper layers and subdocuments. Being applied to an object of class Selection or FlaggedObjects this property returns True.

Example

```
Dim All As AllObjects, WP As Boolean

WP = All.Wallpaper
```

NrOfObjects

Returns

Long

Description

Returns the number of elementary objects in the set. The compounds are not included, but the elements of the compounds are included.

Example

```
Dim A As Application
Dim Doc As Document
Dim MyDrawing As AllObjects
Dim DrawingOverlay As Overlay
Dim OverlayType As Integer
```

```

Set A = New Application
A.GetActiveDocument Doc
Doc.GetOverlay ptDrawing, DrawingOverlay
DrawingOverlay.GetAllObjects MyDrawing
MsgBox "Number of objects in Drawing Overlay : " & Str(MyDrawing.NrOfObjects)

```

NrOfWholeObjects

Returns

Long

Description

Returns the number of elementary plus compound objects in the set. So the elements of a compound are not included. In other words: the result is the sum of the objects not being a member of a compound plus the number of compounds.

Example

```

Dim A As Application
Dim Doc As Document
Dim MyDrawing As AllObjects
Dim DrawingOverlay As Overlay
Dim OverlayType As Integer
Dim Count As Long

Set A = New Application
A.GetActiveDocument Doc
Doc.GetOverlay ptDrawing, DrawingOverlay
DrawingOverlay.GetAllObjects MyDrawing
Count = MyDrawing.NrOfWholeObjects

```

Color

Parameters

Color As Integer

Description

Sets the color (one of the 255 predefined palette colors) of the objects in the set. The color is only changed for those object types for which the property color is applicable. When applied to Rich Texts, all characters in the rich text block will get the given color. When applied to Annotations, both the texts as the symbol will get the given color.

Example

```

Dim A As Application
Dim Doc As Document
Dim MyDrawing As AllObjects
Dim DrawingOverlay As Overlay
Dim OverlayType As Integer

Set A = New Application
A.GetActiveDocument Doc
Doc.GetOverlay ptDrawing, DrawingOverlay
DrawingOverlay.GetAllObjects MyDrawing
MyDrawing.Color = ptBlue

```

TrueColor

Parameters

Color As RGB

Description

Sets the color (given as RGB value) of the objects in the set. The color is only changed for those object types for which the property color is applicable. When applied to Rich Texts, all characters in the rich text block will get the given color.

Example

```

Dim Obj1 As CadObject
Dim RGBColor As RGB

RGBColor.Red = 120
RGBColor.Green = 250
RGBColor.Blue = 65

```

```
Obj1.TrueColor = RGBColor
```

ImageIntensity

Parameters

Intensity As Integer

Description

Sets the intensity of all the images in the set.

Example

```
MyDrawing.ImageIntensity = 50
```

Comment

Parameters

Comment As String

Description

Sets the comment of all objects in the set to Comment.

Example

```
Dim A As Application
Dim Doc As Document
Dim SelectedObjects As Selection

Set A = New Application
A.GetActiveDocument Doc
Doc.GetSelection SelectedObjects
SelectedObjects.Comment = "ABCD"
```

DisplayLevel

Parameters

Level As Integer

Description

Sets the display level of all objects in the set to Level.
The value of DisplayLevel must be in the range -10 to 10.

Example

```
Dim A As Application
Dim Doc As Document
Dim SelectedObjects As Selection

Set A = New Application
A.GetActiveDocument Doc
Doc.GetSelection SelectedObjects
SelectedObjects.DisplayLevel = 2
```

Layer

Parameters

LayerName As String

Description

Sets the Layer of all objects in the set to LayerName. If the layer does not exist, a layer with the given name will be created. This property has no effect on Compound objects.

IfLayerName = "~SYSTEM~" (name of the System Layer), then the object, when it's a point, arc or clothoid, will be put in the System Layer. Trying to put other objects in the System Layer will have no effect.

For more information about System Layer see chapter Layers.

Example

```
Dim A As Application
Dim Doc As Document
Dim SelectedObjects As Selection

Set A = New Application
A.GetActiveDocument Doc
Doc.GetSelection SelectedObjects
SelectedObjects.Layer = "Selections"
```

PointStyle

Parameters

NewStyle As Style

Description

Sets the Style of all Points in the set to NewStyle.

Example

```
Dim A As Application
Dim Doc As Document
Dim SelectedObjects As Selection
Dim NewStyle As Style

Set A = New Application
A.GetActiveDocument Doc
Doc.GetSelection SelectedObjects
With NewStyle
    .Symbol = True
    .Group = "Example"
    .Item = "Doorstep"
End With
SelectedObjects.PointStyle = NewStyle
```

LineStyle

Parameters

NewStyle As Style

Description

Sets the Style of all Line type objects (Lines, Arcs, Circles, Spirals, Curves) in the set to NewStyle.

Example

```
Dim A As Application
Dim Doc As Document
Dim SelectedObjects As Selection
Dim NewStyle As Style

Set A = New Application
A.GetActiveDocument Doc
Doc.GetSelection SelectedObjects
With NewStyle
    .Symbol = True
    .Group = "Example"
    .Item = "Wall"
End With
SelectedObjects.LineStyle = NewStyle
```

LineWidth

Parameters

Width As Double

Description

Sets the Width of all Line type objects (Lines, Arcs, Circles, Curves, ..) in the set to Width.
Width: between 0.1 and 25.5. 1.0 = 0.1 mm, 20.0 = 2.0 mm.

Example

```
Dim A As Application
Dim Doc As Document
Dim SelectedObjects As Selection

Set A = New Application
A.GetActiveDocument Doc
Doc.GetSelection SelectedObjects
SelectedObjects.LineWidth = 2.0 ' Sets Line Width = 0.2 mm.
```

LineArrows

Parameters

On As Boolean

Description

Sets or resets the line arrows at the end points of Lines, Arcs and Spirals in the set. Arrows of an object will only be set if the corresponding end points of the objects are also in the set. If On = TRUE then the arrow is set. This function behaves identically as its interactive counterpart.

Example

```
Dim A As Application
Dim Doc As Document
Dim SelectedObjects As Selection

Set A = New Application
A.GetActiveDocument Doc
Doc.GetSelection SelectedObjects
SelectedObjects.LineArrows = TRUE
```

TextStyle

Parameters

Style As Integer

Description

Sets the Style of all texts in the set to Style.
Style = ptUpright (= 1): Normal
Style = ptItalic (= 2): Italic
This property is not applicable to Rich Text objects.

Example

```
Dim A As Application
Dim Doc As Document
Dim SelectedObjects As Selection

Set A = New Application
A.GetActiveDocument Doc
Doc.GetSelection SelectedObjects
SelectedObjects.TextStyle = ptItalic ' Sets all selected text Italic
```

TextWeight

Parameters

Weight As Integer

Description

Sets the Weight of all texts in the set to Weight.
Weight = ptLight (= 0): Light
Weight = ptMedium (= 1): Normal
Weight = ptBold (= 2): Bold
Weight = ptExtraBold (= 3): Extra Bold
Not all fonts can be represented Light or Extra Bold.
This property is not applicable to Rich Text objects.

Example

```
Dim A As Application
Dim Doc As Document
Dim SelectedObjects As Selection

Set A = New Application
A.GetActiveDocument Doc
Doc.GetSelection SelectedObjects
SelectedObjects.TextWeight = ptBold ' Sets all selected text Bold
```

TextPointSize

Parameters

PointSizeAs Integer

Description

Sets the point size of all texts in the set to PointSize.
Value of PointSize must be between 1 and 1000.
This property is not applicable to Rich Text objects.

Example

```
Dim A As Application
Dim Doc As Document
Dim SelectedObjects As Selection

Set A = New Application
A.GetActiveDocument Doc
Doc.GetSelection SelectedObjects
SelectedObjects.TextPointSize = 20           ' Sets all selected 20 point
```

TextHorizontalAlignment

Parameters

Alignment As Integer

Description

Sets the horizontal alignment of all texts in the set to Alignment.

Alignment = ptAlignLeft(= 0): left aligned.

Alignment = ptAlignCenter(= 1): centered.

Alignment = ptAlignRight (= 2): right aligned.

This property is not applicable to Rich Text objects.

Example

```
Dim A As Application
Dim Doc As Document
Dim SelectedObjects As Selection

Set A = New Application
A.GetActiveDocument Doc
Doc.GetSelection SelectedObjects
SelectedObjects.TextHorizontalAlignment = ptAlignCenter
```

TextVerticalAlignment

Parameters

Alignment As Integer

Description

Sets the vertical alignment of all texts in the set to Alignment.

Alignment = ptAlignBottom(= 0) : bottom aligned.

Alignment = ptAlignCenter(= 1) : centered.

Alignment = ptAlignTop (= 2) : top aligned.

This property is not applicable to Rich Text objects.

Example

```
Dim A As Application
Dim Doc As Document
Dim SelectedObjects As Selection

Set A = New Application
A.GetActiveDocument Doc
Doc.GetSelection SelectedObjects
SelectedObjects.TextVerticalAlignment = ptAlignTop
```

TextUnderline

Parameters

Underline As Integer

Description

Sets the underlining of all texts in the set to Underline.

Underline = ptUnderlineNone (= 0) : no underlining.

Underline = ptUnderlineSingle (= 1) : single underlining.

Underline = ptUnderlineDouble (= 2) : double underlining.

Underline = ptUnderlineDashed (= 3) : underlining dashed line.

This property is not applicable to Rich Text objects.

Example

```
Dim A As Application
```

```

Dim Doc As Document
Dim SelectedObjects As Selection

Set A = New Application
A.GetActiveDocument Doc
Doc.GetSelection SelectedObjects
SelectedObjects.TextUnderline = ptUnderlineSingle

```

TextBorder

Parameters

Border As Integer

Description

Sets the border of all texts in the set toBorder.

Border = ptBorderNone (= 0) : no border.

Border = ptBorderSingle (= 1) : single border.

Border = ptBorderDouble (= 2) : double border.

This property is not applicable to Rich Text objects.

Example

```

Dim A As Application
Dim Doc As Document
Dim SelectedObjects As Selection

Set A = New Application
A.GetActiveDocument Doc
Doc.GetSelection SelectedObjects
SelectedObjects.TextBorder = ptBorderSingle

```

TextLineSpacing

Parameters

LineSpacing As Integer

Description

Sets the spacing of all texts in the set toLineSpacing.

LineSpacing = ptSpacingSmall (= 0) : small line spacing.

LineSpacing = ptSpacingNormal (= 1) : normal line spacing.

LineSpacing = ptSpacingLarge (= 2) : large line spacing.

This property is not applicable to Rich Text objects.

Example

```

Dim A As Application
Dim Doc As Document
Dim SelectedObjects As Selection

Set A = New Application
A.GetActiveDocument Doc
Doc.GetSelection SelectedObjects
SelectedObjects.TextLineSpacing = ptSpacingLarge

```

TextOpaque

Parameters

Opaque As Boolean

Description

Sets all texts in the set, also rich texts, either Opaque or Transparent.

Opaque = True : Opaque.

Opaque = False : Transparent.

Example

```

Dim A As Application
Dim Doc As Document
Dim SelectedObjects As Selection

Set A = New Application
A.GetActiveDocument Doc
Doc.GetSelection SelectedObjects

```

```
SelectedObjects.TextOpaque = True
```

TextOrientation

Parameters

Orientation As Integer

Description

Sets the orientation of all texts in the set, also rich texts, to Orientation.

Orientation = ptTextHorizontal (= 0): The texts will always appear horizontal independent of page orientation or angle.

Orientation = ptTextOrientationReadable (= 1) : The angle of the displayed or printed texts will change if the text would come upside down. If the angle of the texts is in 2nd or 3rd quadrant, the texts are turned over 180 degrees.

Orientation = ptTextOrientationAbsolute (= 2) : The texts are always displays at the given angle.

Example

```
Dim A As Application
Dim Doc As Document
Dim SelectedObjects As Selection

Set A = New Application
A.GetActiveDocument Doc
Doc.GetSelection SelectedObjects
SelectedObjects.TextAngle = 1.3
SelectedObjects.TextOrientation = ptTextOrientationReadable
```

TextAngle

Parameters

Angle As Double

Description

Sets the angle of all texts in the set, also rich texts, to Angle. TextAngle will set the orientation to ptTextOrientationAbsolute, if the orientation was horizontal (even if the angle = 0.0). If the orientation was not horizontal, the orientation is kept. The angle is expressed in radians.

See also TextOrientation

Example

See TextOrientation

CurveCurvature (obsolete)

Parameters

Curvature As Integer

Description

Sets the curvature of curves in the set with all line segments or all curve (Bezier) segments to Curvature. The curvature needs to be in the range [0, 10]. 0 are line segments, 5 are normal curve (Bezier) segments and the other values differ in curvature like the v15 Curve.

Example

```
Dim A As Application
Dim Doc As Document
Dim SelectedObjects As Selection

Set A = New Application
A.GetActiveDocument Doc
Doc.GetSelection SelectedObjects
SelectedObjects.CurveCurvature = 2
```

CurveMaxRounding (obsolete)

Parameters

MaxRounding As Boolean

Description

Doesn't set the max rounding since v16.

Example

```
Dim A As Application
Dim Doc As Document
Dim SelectedObjects As Selection

Set A = New Application
A.GetActiveDocument Doc
Doc.GetSelection SelectedObjects
SelectedObjects.CurveMaxRounding = 2
```

PolygonPattern

Parameters

Pattern As Style

Description

Sets the hatching of all Polygons in the set. For . Pattern corresponds to a number indicating a specific polygon pattern. (See Polygon : styleId)..

Example

```
Dim SelectedObjects As Selection
Dim NewStyle As Style

With NewStyle
    .Symbol = True
    .Group = "Test"
    .Item = "Horizontal"
End With
SelectedObjects.PolygonPattern = NewStyle
```

PolygonBorder

Parameters

Border As Integer

Description

Sets the border of all polygons in the set, to Border.

Border = ptNoBorder: The polygons will be filled normal (not only a border).

Border = ptBorder1mm .. ptBorder10mm: The width of the border of the polygons will be from 1 mm up to 10 mm depending on the value of the parameter.

Border = ptBorder1mmOpen .. ptBorder10mmOpen: The width of the border of the polygons will be from 1 mm up to 10 mm depending on the value of the parameter. The last leg of the polygon is left open.

Example

```
Dim A As Application
Dim Doc As Document
Dim SelectedObjects As Selection

Set A = New Application
A.GetActiveDocument Doc
Doc.GetSelection SelectedObjects
SelectedObjects.PolygonBorder = ptBorder5mm
```

PolygonOpaque

Parameters

Opaque As Boolean

Description

Sets the attribute Opaque of all Polygons in the set. If Opaque = True, then the polygons will be opaque.

Example

```
SelectedObjects.PolygonOpaque = True
```

PolygonBackgroundColor

Parameters

BGColor As Integer

Description

Sets the background color of all Polygons in the set. `BGColor` corresponds to a color number of the systems color palette. Setting the background color has only effect if the type of background is set.

Example

```
SelectedObjects.PolygonBackgroundPattern = ptGray50
SelectedObjects.PolygonBackgroundColor = 81
```

PolygonBackgroundPattern

Parameters

`BGPattern` As Integer

Description

Sets the background pattern of all Polygons in the set. `BGPattern` corresponds to a number indicating a specific polygon pattern. (See `Polygon : styleId`). Only the values `ptNoFill`, `ptSolidFill`, `ptGrayxx` (`xx = 75, 50, 25` or `12`) are valid background patterns.

Example

```
SelectedObjects.PolygonBackgroundPattern = ptSolidFill
```

SolidOpaque

Parameters

`Opaque` As Boolean

Description

Sets the attribute `Opaque` of all solids in the set. If `Opaque = True`, then the solids will be opaque.

Example

```
SelectedObjects.SolidOpaque = True
```

SolidIntensity

Parameters

`Intensity` As Integer

Description

Sets the intensity of all the solids in the set.

Example

```
MyDrawing.SolidIntensity = 50
```

CanChangeOverlay

Returns

`CanChange` As Boolean

Description

Returns `TRUE` if all objects in the set can be moved from paper overlay to drawing overlay or vice versa. It is not always possible to change the overlay of an object. A line refers to 2 points. The line can only be moved to another overlay together with the 2 points.

Example

```
Dim A As Application
Dim Doc As Document
Dim MyDrawing As AllObjects
Dim DrawingOverlay As Overlay

Set A = New Application
A.GetActiveDocument Doc
Doc.GetOverlay ptDrawing, DrawingOverlay
DrawingOverlay.GetAllObjects MyDrawing
If MyDrawing.CanChangeOverlay Then
    MyDrawing.ChangeOverlay ' Will move objects to Paper overlay
End If
```

Container

Returns

Compound

Description

All members of the set will become a member of the Compound. Assigning to `Nothing` will exclude the `CadObjects` from the compound they belonged to.

All objects in a `Container` must be in the same `Overlay` and in the same `Subdocument`. If objects are in the overlay `ptPaper`, the objects must be in the same `Sheet`.

`Container` property is only applicable to `Selected` and `Flagged` objects.

Example

```
Dim A As Application
Dim Doc As Document
Dim Flagged As FlaggedObjects
Dim C As Compound

Set A = New Application
A.GetActiveDocument Doc
Doc.GetFlaggedObjects Flagged
Set Flagged.Container = C ' All flagged objects will become member of compound.
```

AnnotationTextStyle

Parameters

`Style` As Integer

Description

Sets the `Style` of the texts of all annotations in the set to `Style`.

`Style = ptUpright (= 1): Normal`

`Style = ptItalic (= 2): Italic`

Example

```
Dim A As Application
Dim Doc As Document
Dim SelectedObjects As Selection

Set A = New Application
A.GetActiveDocument Doc
Doc.GetSelection SelectedObjects
SelectedObjects.AnnotationTextStyle = ptItalic ' Sets all selected
annotations Italic
```

AnnotationTextWeight

Parameters

`Weight` As Integer

Description

Sets the `Weight` of the texts of all annotations in the set to `Weight`.

`Weight = ptLight (= 0): Light`

`Weight = ptMedium (= 1): Normal`

`Weight = ptBold (= 2): Bold`

`Weight = ptExtraBold (= 3): Extra Bold`

Not all fonts can be represented `Light` or `Extra Bold`.

Example

```
Dim A As Application
Dim Doc As Document
Dim SelectedObjects As Selection

Set A = New Application
A.GetActiveDocument Doc
Doc.GetSelection SelectedObjects
SelectedObjects.AnnotationTextWeight = ptBold ' Sets all selected
annotations Bold
```

AnnotationTextPointSize

Parameters

`PointSizeAs` Integer

Description

Sets the point size of the texts of all annotations in the set to `PointSize`.
Value of `PointSize` must be between 1 and 1000.

Example

```
Dim A As Application
Dim Doc As Document
Dim SelectedObjects As Selection

Set A = New Application
A.GetActiveDocument Doc
Doc.GetSelection SelectedObjects
SelectedObjects.AnnotationTextPointSize = 20 ' Sets all selected
annotations to 20 point
```

AnnotationTextHorizontalAlignment

Parameters

`Alignment` As Integer

Description

Sets the horizontal alignment of the texts of all annotations in the set to `Alignment`.
`Alignment = ptAlignLeft (= 0)`: left aligned.
`Alignment = ptAlignCenter (= 1)`: centered.
`Alignment = ptAlignRight (= 2)`: right aligned.

Example

```
Dim A As Application
Dim Doc As Document
Dim SelectedObjects As Selection

Set A = New Application
A.GetActiveDocument Doc
Doc.GetSelection SelectedObjects
SelectedObjects.AnnotationTextHorizontalAlignment = ptAlignCenter
```

AnnotationTextUnderline

Parameters

`Underline` As Integer

Description

Sets the underlining of the texts of all annotations in the set to `Underline`.
`Underline = ptUnderlineNone (= 0)`: no underlining.
`Underline = ptUnderlineSingle (= 1)`: single underlining.
`Underline = ptUnderlineDouble (= 2)`: double underlining.
`Underline = ptUnderlineDashed (= 3)`: underlining dashed line.

Example

```
Dim A As Application
Dim Doc As Document
Dim SelectedObjects As Selection

Set A = New Application
A.GetActiveDocument Doc
Doc.GetSelection SelectedObjects
SelectedObjects.AnnotationTextUnderline = ptUnderlineSingle
```

AnnotationTextBorder

Parameters

`Border` As Integer

Description

Sets the border of the texts of all annotations in the set to `Border`.
`Border = ptBorderNone (= 0)`: no border.
`Border = ptBorderSingle (= 1)`: single border.
`Border = ptBorderDouble (= 2)`: double border.
This property is not applicable to Rich Text objects.

Example

```
Dim A As Application
Dim Doc As Document
Dim SelectedObjects As Selection

Set A = New Application
A.GetActiveDocument Doc
Doc.GetSelection SelectedObjects
SelectedObjects.AnnotationTextBorder = ptBorderSingle
```

AnnotationTextLineSpacing

Parameters

LineSpacing As Integer

Description

Sets the spacing of the texts of all annotations in the set to LineSpacing.

LineSpacing = ptSpacingSmall (= 0) : small line spacing.

LineSpacing = ptSpacingNormal (= 1) : normal line spacing.

LineSpacing = ptSpacingLarge (= 2) : large line spacing.

This property is not applicable to Rich Text objects.

Example

```
Dim A As Application
Dim Doc As Document
Dim SelectedObjects As Selection

Set A = New Application
A.GetActiveDocument Doc
Doc.GetSelection SelectedObjects
SelectedObjects.AnnotationTextLineSpacing = ptSpacingLarge
```

AnnotationTextOpaque

Parameters

Opaque As Boolean

Description

Sets the texts of all annotations in the set, either Opaque or Transparent.

Opaque = True : Opaque.

Opaque = False : Transparent.

Example

```
Dim A As Application
Dim Doc As Document
Dim SelectedObjects As Selection

Set A = New Application
A.GetActiveDocument Doc
Doc.GetSelection SelectedObjects
SelectedObjects.AnnotationTextOpaque = True
```

AnnotationTextOrientation

Parameters

Orientation As Integer

Description

Sets the orientation of the texts of all annotations in the set to Orientation.

Orientation = ptTextHorizontal (= 0): The texts will always appear horizontal independent of page orientation or angle.

Orientation = ptTextOrientationReadable (= 1) : The angle of the displayed or printed texts will change if the text would come upside down. If the angle of the texts is in 2nd or 3rd quadrant, the texts are turned over 180 degrees.

Orientation = ptTextOrientationAbsolute (= 2) : The texts are always displayed at the given angle.

The possible text orientations depend on the annotation type. A pointer annotation can have the three possible text orientations. The other annotation types can only have ptTextOrientationReadable

and `ptTextOrientationAbsolute`. When these are set to `ptTextHorizontal`, the request will be ignored.

See also `AnnotationType`.

Example

```
Dim A As Application
Dim Doc As Document
Dim SelectedObjects As Selection

Set A = New Application
A.GetActiveDocument Doc
Doc.GetSelection SelectedObjects
SelectedObjects.AnnotationTextAngle = 1.3
SelectedObjects.AnnotationTextOrientation = ptTextOrientationReadable
```

AnnotationTextColor

Parameters

Color As Integer

Description

Sets the color (one of the 255 predefined palette colors) of the annotation texts in the set.

Example

```
Dim A As Application
Dim Doc As Document
Dim MyDrawing As AllObjects
Dim DrawingOverlay As Overlay

Set A = New Application
A.GetActiveDocument Doc
Doc.GetOverlay ptDrawing, DrawingOverlay
DrawingOverlay.GetAllObjects MyDrawing
MyDrawing.AnnotationTextColor = ptBlue
```

AnnotationSymbolColor

Parameters

Color As Integer

Description

Sets the color (one of the 255 predefined palette colors) of the annotation symbols in the set.

Example

```
Dim A As Application
Dim Doc As Document
Dim MyDrawing As AllObjects
Dim DrawingOverlay As Overlay

Set A = New Application
A.GetActiveDocument Doc
Doc.GetOverlay ptDrawing, DrawingOverlay
DrawingOverlay.GetAllObjects MyDrawing
MyDrawing.AnnotationSymbolColor = ptBlue
```

AnnotationSymbolScale

Parameters

Scale As Double

Description

Sets the scale of the annotation symbols in the set.

Example

```
Dim A As Application
Dim Doc As Document
Dim MyDrawing As AllObjects
Dim DrawingOverlay As Overlay

Set A = New Application
A.GetActiveDocument Doc
```

```
Doc.GetOverlay ptDrawing, DrawingOverlay
DrawingOverlay.GetAllObjects MyDrawing
MyDrawing.AnnotationSymbolScale = 50
```

Methods

Count

Returns

Number As Long

Parameters

Types As Long

Description

Returns the number of CadObjects in the set whose type is in Types .

Types : is the sum of the combination of any of the following constants :

ptPoint, ptLine, ptArc, ptClothoid, ptText, ptRichText, ptImage,
ptCurve, ptPath, ptPolygon, ptCircle, ptCoordinateSystem, ptCompound,
ptAnnotation, pt3DSolid.

If you need to count all object types, the constant ptAll should be used.

Example

```
Dim A As Application
Dim Doc As Document
Dim DrawingOverlay As Overlay
Dim Object As CadObject
Dim MyDrawing As AllObjects
Dim Number As Long, Counter As Long
Dim Cnt As Long
Dim P As Point

Set A = New Application
A.GetActiveDocument Doc
Doc.GetOverlay ptDrawing, DrawingOverlay
DrawingOverlay.GetAllObjects MyDrawing
Number = MyDrawing.Count(ptPoint)
Cnt = 0
MyDrawing.GetFirst ptPoint, Object
For Counter = 1 TO Number
    If Object Is Nothing Then
        ' No more objects
    Else
        Object.GetPoint P
        Cnt = Cnt + 1
    End If
    MyDrawing.GetNext Object
Next Counter
MsgBox "Nr of points = " & Str(Cnt)
```

GetFirst

Parameters

Types As Long, Object As CadObject, Optional OnlyWholeObjects As Boolean

Description

Returns the first CadObject in the set whose type is in Types.

Types : is the sum of the combination of any of the following constants :

ptPoint, ptLine, ptArc, ptClothoid, ptText, ptRichText, ptImage,
ptCurve, ptPath, ptPolygon, ptCircle, ptCoordinateSystem, ptCompound,
ptAnnotation, pt3DSolid. The value ptAll is the sum of all object types.

OnlyWholeObjects: if False (default value), the first object, and with GetNext the following object in Types is returned. If True, only elementary objects (not being a member of a compound) and compounds, if they are in the set, are returned.

The program should check if Object <> Nothing.

Example

```
Dim A As Application
Dim Doc As Document
Dim Object As CadObject
Dim SelectedObjects As Selection

Set A = New Application
A.GetActiveDocument Doc
Doc.GetSelection SelectedObjects
SelectedObjects.GetFirst ptLine + ptArc + ptCircle, Object
' The returned Object is either a Line, Arc or Circle.
If Object Is Nothing Then
    MsgBox "No Line, Arc or Circle Selected"
End If
SelectedObjects.GetFirst ptAll, Object, True
' Returns either an elementary element of a compound
```

GetNext

Parameters

Object As CadObject

Description

GetNext must be called after GetFirst. GetNext returns the next CadObject in the set, whose type is in the types specified by GetFirst. This method will normally be called in a loop until all objects are investigated or until the desired result is reached. When running through the selected objects, including new objects is **not** allowed. Correct working of the GetNext loop is not guaranteed in that case. Resetting selections is allowed, and this operation will directly change the Selection set. The same restriction applies to flagged objects.

The program should check if Object <> Nothing when returning from GetNext. No error is raised if GetNext is called after retrieving the last object from the set.

Example

See example Count

ChangeOverlay

Parameters

Optional ForceFreezeAnnotations As Boolean

Description

Changes the overlay of all objects of the set. If the objects are in the Paper overlay, they will be moved to the drawing overlay.

If they are in the drawing overlay, they will be moved to the Paper overlay. The operation will not be done when the overlay of one or more of the objects cannot be changed.

If the document contains multiple sheets, the method will only affect the active sheet.

See also CanChangeOverlay.

When this method is applied to Selection or FlaggedObjects then the corresponding set becomes empty after the operation.

When ForceFreezeAnnotations is True the text of the annotation is frozen before the change is done.

Example

```
Dim A As Application
Dim Doc As Document
Dim MyDrawing As AllObjects
Dim DrawingOverlay As Overlay

Set A = New Application
A.GetActiveDocument Doc
Doc.GetOverlay ptDrawing, DrawingOverlay
DrawingOverlay.GetAllObjects MyDrawing
If MyDrawing.CanChangeOverlay Then
    MyDrawing.ChangeOverlay True ' Will move objects to Paper overlay and
the text of the annotations is kept
End If
```


Copy

Parameters

PythagorasClipboard As Boolean

Description

Copies all objects of the set to either the Pythagoras clipboard or the system (Mac or Windows) clipboard. When PythagorasClipboard = TRUE then the objects will be copied the Pythagoras clipboard.

Example

```
Dim A As Application
Dim Doc As Document
Dim Object As CadObject
Dim SelectedObjects As Selection

Set A = New Application
A.GetActiveDocument Doc
Doc.GetSelection SelectedObjects
SelectedObjects.Copy False      ' Will copy all selected objects to the system
                                ' clipboard
```

Cut

Parameters

None

Description

All objects of the set are deleted from the document and copied to the Pythagoras clipboard.

Example

```
Dim A As Application
Dim Doc As Document
Dim Object As CadObject
Dim SelectedObjects As Selection

Set A = New Application
A.GetActiveDocument Doc
Doc.GetSelection SelectedObjects
SelectedObjects.Cut
```

Clear

Parameters

None

Description

All objects of the set are deleted from the document.

Example

```
Dim SelectedObjects As Selection

SelectedObjects.Clear
```

Move

Parameters

Delta As XYZ

Description

Moves all objects of the set using the vector Delta. Care must be taken if some objects refer to other which are not in the set, or objects outside the set refer to objects in the set. The method CanChangeOverlay could be used to check if such references exist.

Example

```
Dim A As Application
Dim Doc As Document
Dim Object As CadObject
Dim SelectedObjects As Selection
Dim Vector As XYZ

Set A = New Application
A.GetActiveDocument Doc
```

```

Doc.GetSelection SelectedObjects
Vector.x = 10.0
Vector.y = 20.0
Vector.z = 1.0
SelectedObjects.Move Vector

```

Rotate

Parameters

Center As XY, Angle As Double

Description

Rotates all objects of the set around point Center. The Angle is counter clockwise in radians. After the operation all rotated objects occur to be in the active subdocument. That is why this method should not be applied to sets containing objects lying outside the active subdocument.

Example

```

Dim A As Application
Dim Doc As Document
Dim Object As CadObject
Dim SelectedObjects As Selection
Dim Center As XY

Set A = New Application
A.GetActiveDocument Doc
Doc.GetSelection SelectedObjects
Center.x = 10.0
Center.y = 20.0
SelectedObjects.Rotate Center, 3.1418

```

Group

Parameters

Optional NewCompound As Compound

Description

Makes a new Compound containing all objects of the set. If the set contains Cad objects of multiple subdocuments, one compound will be made per subdocument. if the set contains Cad Objects of multiple subdocuments then NewCompound Is Nothing.

If this method is applied on AllObjects in overlay ptPaper, then AllObjects should refer to one specific Sheet.

Example

```

Dim A As Application
Dim Doc As Document
Dim Object As CadObject
Dim SelectedObjects As Selection
Dim C As Compound

Set A = New Application
A.GetActiveDocument Doc
Doc.GetSelection SelectedObjects
SelectedObjects.Group
' or
SelectedObjects.Group C ' Is Nothing if multiple subdocuments

```

Ungroup

Parameters

None

Description

Removes all objects of the set from the Compound they belong to. So the objects of the set will not belong to any Compound after this operation.

Example

```

Dim A As Application
Dim Doc As Document
Dim Object As CadObject
Dim SelectedObjects As Selection

```

```

Set A = New Application
A.GetActiveDocument Doc
Doc.GetSelection SelectedObjects
SelectedObjects.Ungroup

```

Unpack

Parameters
None

Description
Unpacks all objects in the set.

Example

```

Dim A As Application
Dim Doc As Document
Dim Object As CadObject
Dim SelectedObjects As Selection

Set A = New Application
A.GetActiveDocument Doc
Doc.GetSelection SelectedObjects
SelectedObjects.Unpack

```

DeleteData

Parameters
None

Description
Deletes the data of all objects in the set.

Example

```

Dim A As Application
Dim Doc As Document
Dim Object As CadObject
Dim SelectedObjects As Selection

Set A = New Application
A.GetActiveDocument Doc
Doc.GetSelection SelectedObjects
SelectedObjects.DeleteData

```

AddAnnotation

Parameters
MyUnits As Units, AnnotationType As Integer

Description
This method creates annotations as simple text objects.

This method will be renewed to create annotation objects instead. We will add an annotation of type AnnotationType, the units and other parameters regarding the annotation are given by MyUnits. (See Object Class Units)

Example

```

Dim A As Application
Dim Doc As Document
Dim Object As CadObject
Dim SelectedObjects As Selection
Dim MyUnits As Units

Set A = New Application
Set MyUnits = New Units
MyUnits.DecimalPlacesForLengths = 2
A.GetActiveDocument Doc
Doc.GetSelection SelectedObjects
SelectedObjects.AddAnnotation MyUnits, ptAn_LineLength

```

AdjustDisplayLevel

Parameters

Delta As Integer

Description

Adds the value given by parameter `Delta` to the display level of all objects in the set. When the adjusted display level would exceed the limiting value of a display level (-10 .. +10), then an error will be raised. The value of `Delta` must be between -10 and +10.

Example

```
Dim A As Application
Dim Doc As Document
Dim SelectedObjects As Selection

Set A = New Application
A.GetActiveDocument Doc
Doc.GetSelection SelectedObjects
SelectedObjects.AdjustDisplayLevel -3
```

SetZ

Parameters

Height As Double

Description

Sets the Z value of the objects in the set to the specified value.

Example

```
Dim SelectedObjects As Selection

SelectedObjects.SetZ 34.7
```

ScalePoints

Parameters

AbsoluteScale As Boolean, ScaleX As Double, ScaleY As Double

Description

Changes the scale all the symbols in the set. If the parameter `AbsoluteScale = True`, then `ScaleX` and `ScaleY` denote the scale to which the symbols will be set. If `AbsoluteScale = False`, then `ScaleX` and `ScaleY` will be applied to the current scale factor. The scale factors are expressed in %. So a value of 200 means that the symbols will be scaled with a factor 2.

Attention : in the case of `AbsoluteScale = True` and a scale factor is positive, the sign of the corresponding previous scale of each point in the set will be retained. If the scale factor is negative, the sign of the corresponding previous scale of each point in the set will be inverted.

Example

```
SelectedObjects.ScalePoints False, 50.0, 100.0
```

ScaleLines

Parameters

AbsoluteScale As Boolean, ScaleFactor As Double

Description

Changes the scale of all the lines in the set. If the parameter `AbsoluteScale = True`, then `ScaleFactor` denotes the scale to which the style of the lines will be set. If `AbsoluteScale = False`, then `ScaleFactor` will be applied to the current scale factor.

It is obvious that only lines with a scalable line style will be affected. The scale factor is expressed in %.

Attention : in the case of `AbsoluteScale = True` and the scale factor is positive, the sign of the previous scale of each line in the set will be retained. If the scale factor is negative, the sign of the previous scale of each line in the set will be inverted.

Example

```
SelectedObjects.ScaleLines True, 50.0
```

ScaleTexts

Parameters

ScaleFactor As Double

Description

Scales the texts in the set by a given scale factor. Since the size of texts are discrete values between 1 and 1000. The result will be rounded to the nearest cardinal value.

Example

```
SelectedObjects.ScaleTexts 200.0      ' Point size X 2
```

ScalePolygons

Parameters

AbsoluteScale As Boolean, ScaleFactor As Double

Description

Changes the scale of all the polygons in the set. If the parameter AbsoluteScale = True, then ScaleFactor denotes the scale to which the style of the hatching will be set. If AbsoluteScale = False, then ScaleFactor will be applied to the current scale factor. It is obvious that only polygons with a scalable hatching will be affected. The scale factor is expressed in %.

Example

```
SelectedObjects.ScalePolygons False, 150.0
```

AlignPolygons

Parameters

Horizontal As Boolean, Origin As XY, Angle As Double

Description

Aligns the hatching of all polygons in the set.

If Horizontal = True, then the hatchings will become horizontal to the page and have an absolute origin which is the same for all unaligned hatchings. In this case, the other parameters are ignored.

If Horizontal = False, then the origin of the hatchings is defined by the parameter Origin, and the angle by the parameter Angle.

Example

```
Const pi = 3.1416
Dim Org As XY

Org.X = 10.0
Org.Y = 10.0
SelectedObjects.AlignPolygons False, Org, 45.0/180.0*pi ' hatching 45 deg.
```

ConvertToPolylines

Parameters

NrOfPolylines as Long, Polylines() as Curve

Description

Converts all lines of the set to polylines. The original lines are deleted and removed from the set.

NrOfPolylines indicates the number of polylines that were created. The new polylines are returned in the Polylines() array parameter.

Example

```
SelectedObjects.ConvertToPolylines NrOfPolyLines, Polylines
```

ConvertObjectsToPolylines (obsolete)

Parameters

Types As ObjectTypes

Description

Converts all lines of the set to polylines. The original lines are deleted and removed from the set.

Types must be equal to ptLine.

Example

```
SelectedObjects.ConvertObjectsToPolylines ptLine
```

GetFirstPoint

Parameters

Center As XY, Radius As Double, Points As Point

Description

Returns the first point that lies within the circle with the given center and radius. The first point is not necessarily the point that is nearest to the center. Points are returned in random order.

Example

See GetNextPoint

GetNextPoint

Parameters

Point As Point

Description

Returns the next point within the circle.

Example

```
Dim A As Application
Dim Pnt As Point
Dim Center As XY
Dim Flagged As FlaggedObjects

Set A = New Application
A.GetActiveDocument Doc
Doc.GetFlaggedObjects Flagged
Center.x = 10.0
Center.y = 10.0
Flagged.GetFirstPoint Center, 1.5, Pnt
If Pnt Is Nothing Then
    MsgBox "No Point found"
End If
Do While Pnt <> Nothing
    Flagged.GetNextPoint Pnt
Loop
```

IntersectingLines

Parameters

IntersectType As Long, NrOfLines As Long, Lines() As LinePair,
Optional OnlyVisibleLayers As Boolean = False

Description

Makes a list of the lines, arcs, circles, polygons and curves (containing only line segments) that one or other way intersect. Overshoot conditions are not taken into account.

IntersectType : only line pairs that satisfy the condition will be returned.

ptPartiallyOverlapping: if 2 lines partially overlap,

ptIdenticalLines: if 2 lines are completely identical,

ptIntersectingLines: if 2 lines intersect. Parallel lines and lines with a common end point are excluded.

ptCommonEndPoint: if 2 lines have a common end point. Does not include partially overlapping lines that have a common end point.

On return :

NrOfLines : the number of line pairs in the array Lines

Lines : all the line pairs that intersect. Maximum 100 line pairs are returned. Indexing starts from 1.

Example

```
Dim A As Application
Dim Doc As Document
Dim Lines() As LinePair
Dim NrOfLines As Long
Dim Flagged As FlaggedObjects

Set A = New Application
A.GetActiveDocument Doc
Doc.GetFlaggedObjects Flagged
```

```
Flagged.IntersectingLines ptIntersectingLines, NrOfLines, Lines, _
                                OnlyVisibleLayers = True
```

Intersections

Parameters

```
MaxDs As Double,
NrOfLines As Long, Lines() As LinePair,
Optional OnlyVisibleLayers As Boolean = False
```

Description

Makes a list of objects (line, arc, circle, curve or polygon) that intersect with each other. Overshooting objects, (overshoot condition is defined by parameter MaxDS), will not be included in the list.

MaxDs : value defining the the overshoot condition.

On return :

NrOfLines : the number of line pairs in the array Lines

Lines : all the line pairs (lines and arcs) that have the undershoot error.

Example

```
Dim A As Application
Dim Lines() As LinePair
Dim NrOfLines As Long
Dim Flagged As FlaggedObjects

Set A = New Application
A.GetActiveDocument Doc
Doc.GetFlaggedObjects Flagged
Flagged.Intersections 0.05, NrOfLines, Lines, OnlyVisibleLayers=True
```

UnderShoot

Parameters

```
MinDs As Double, MaxDs As Double,
NrOfLines As Long, Lines() As LinePair
Optional OnlyVisibleLayers As Boolean = False
```

Description

Makes a list of lines that have undershoot error.

MinDs and MaxDs : values defining the the overshoot conditions.

On return :

NrOfLines : the number of line pairs in the array Lines

Lines : all the line pairs (lines and arcs) that have the undershoot error.

Example

```
Dim A As Application
Dim Lines() As LinePair
Dim NrOfLines As Long
Dim Flagged As FlaggedObjects

Set A = New Application
A.GetActiveDocument Doc
Doc.GetFlaggedObjects Flagged
Flagged.Undershoot 0.01, 0.05, NrOfLines, Lines, OnlyVisibleLayers:=True
```

OverShoot

Parameters

```
MinDs As Double, MaxDs As Double,
NrOfLines As Long, Lines() As LinePair
Optional OnlyVisibleLayers As Boolean = False
```

Description

See UnderShoot

Example

See UnderShoot

SetTextFont

Parameters

FontName As String, CodePage As Integer

Description

Changes the font of all objects of the set.

Example

```
SelectedObjects.SetTextFont "Arial", ptWestern
```

SetAnnotationTextFont

Parameters

FontName As String, CodePage As Integer

Description

Changes the font of the texts of all annotations in the set.

Example

```
SelectedObjects.SetAnnotationTextFont "Arial", ptWestern
```

SetAnnotationSymbolStyle

Parameters

SymbolStyle As Style, SymbolType As Integer

Description

Sets the style of the symbols of all annotation in the set. The SymbolStyle.StyleId should belong to the corresponding SymbolType. Annotations which symbols cannot be changed to the given SymbolType are omitted.

SymbolType = ptPointerSymbol.

SymbolType = ptRangeSymbol.

SymbolStyle.StyleId can be

ptRangeOnlyText: range text only.

ptExternalRange: range external.

ptFullRange: range simple.

ptFullRangeDotted: range dotted.

ptFullRangeSlashes: range with slashes.

ptFullRangeVertSlashes: range with vertical slashes.

ptFullRangeExtensions: range with extensions.

ptFullRangePullUp: range with dotted extensions.

ptIntermittedRange: intermitted range simple.

ptIntermittedRangeDotted: intermitted range dotted.

ptIntermittedRangeSlashes: intermitted range with slashes.

ptIntermittedRangeVertSlashes: intermitted range with vertical slashes.

ptIntermittedRangeExtensions: intermitted range with extensions.

ptIntermittedRangePullUp: intermitted range with dotted extensions.

ptPointerTextOnly: pointer text only.

ptPointerOnlyTextOnObject: pointer text only centered.

ptSimplePointerLeftOfText: pointer to vertical side.

ptSimplePointerUnderText: pointer to horizontal side.

ptPointerToCorner: pointer to corner.

ptPointerBroken: pointer broken.

ptPointerBrokenToVerticalSide: pointer broken to vertical side.

ptPointerWithVerticalSlash: pointer with vertical slash.

ptSimplePointerUnderlinedText: pointer with underline.

ptRadiusPointer: pointer radius.

ptCirclePointer: pointer with circle.

ptPointerSideWaysText: pointer side ways text

ptPointerManHoleBroken: pointer manhole broken

ptPointerManHoleBrokenToVerticalSide: pointer manhole broken to vertical side

Example

```
Dim A As Application
Dim Doc As Document
Dim MyDrawing As AllObjects
Dim DrawingOverlay As Overlay
Dim S As Style

Set A = New Application
A.GetActiveDocument Doc
Doc.GetOverlay ptDrawing, DrawingOverlay
DrawingOverlay.GetAllObjects MyDrawing

S.Symbol = FALSE
S.Group = ""
S.item = ""
S.StyleId = ptCirclePointer
MyDrawing.SetAnnotationSymbolStyle S, ptPointerAnnotation
```

Transform

Parameters

Transformation As HelmertTransformation, SmoothResiduals As Boolean,
OptionalSmoothOutType As Integer = ptAllDistances

Description

Transforms all objects of the set.

The parameter Transformation must have been calculated using the HelmertTransformation method Calculate. If the boolean SmoothOutResiduals = True, then the pass points which were used to calculate the transformation, will be moved so that the coordinates will be exactly the same as the target coordinates. In that case, the drawing (only the elements of the set) will be distorted.

SmoothOutType: used to define the type of interpolation.

ptNNDistance: only the residuals of the Natural Neighbours are used. The interpolation is based on the distance (1/s) to the Natural Neighbours (subset of the Tie Points).

ptNNArea: only the residuals of the Natural Neighbours are used. The interpolation is based on a method where the change of area in the Voronoi Tessellation defines the weight.

ptAllDistances: the residuals of all Tie Points are taken into account. The weight is inverse proportional with the square of the distance to the Tie Points. (1/s²)

ptInTrianglePlane: the residuals of the 3 points of the triangle in the delauney triangulation is used to calculate the weight.

See also object HelmertTransformation.

Example

```
Dim SelectedObjects As Selection
Dim Tr As HelmertTransformation

SelectedObjects.Transform Tr, True, ptNNArea
```

ClearSet

Parameters

None

Description

Makes the set empty. This method can not be applied to AllObjects.

Example

```
Dim A As Application
Dim Doc As Document
Dim Flagged As FlaggedObjects

Set A = New Application
A.GetActiveDocument Doc
Doc.GetFlaggedObjects Flagged
Flagged.ClearSet
```

MakeSet

Parameters

OverlayType As Integer, SelCriteria As SelectionCriteria,
Optional Table As DBTable = Nothing

Description

Initializes a set containing objects in the overlay OverlayType (either ptDrawing or ptPaper). The parameter SelCriteria defines which objects will be included in the set. **Only objects of visible layers will be included.**

If Table <>Nothing then only objects that are linked to the given table are included in the set.

This method can not be applied to AllObjects.

SelectionCriteria should be initialised with the Application method

InitSelectionCriteria.

The ptCompareAboutEqual parameter for the SelectionCriteria.CommentSelection property cannot be used in this method MakeSet.

Example

```
Dim A As Application
Dim Doc As Document
Dim Flagged As FlaggedObjects
Dim SelCriteria As SelectionCriteria

Set A = New Application
A.GetActiveDocument Doc
Doc.GetFlaggedObjects Flagged
Flagged.ClearSet
A.InitSelectionCriteria SelCriteria 'Inits all fields to default values.
With SelCriteria
    .SelectionType = ptNewSelection
    .OnColor = True
    .Color = ptRed
    .Wallpaper = False
    .OnSubdocument = False
End With
Flagged.MakeSet ptDrawing, SelCriteria
' Will put all red in the set
Flagged.Cut ' Delete & put on Pythagoras clipboard.
```

MakeSetDB

Parameters

OverlayType As Integer, Criteria As SelectionCriteria,
Table As DBTable, DBCriteria() As SelectCondition

Description

This method is an extension of MakeSet. The parameter DBCriteria defines additional conditions for the data linked with a CadObject. The array DBCriteria cannot be empty since an array in VBA must contain at least one element. OverlayType value must be ptLocal always otherwise an empty set will be made.

All conditions must be valid. If not an error is generated.

SelectionCriteria can be initialised with the Application method

InitSelectionCriteria.

The ptCompareAboutEqual parameter for the SelectionCriteria.CommentSelection property cannot be used in this method MakeSetDB.

DBCriteria : for more information see the Pythagoras Data Type SelectCondition.

Example

```
Dim A As Application
Dim Doc As Document
Dim Flagged As FlaggedObjects
Dim SelCriteria As SelectionCriteria
Dim DBCriteria() As SelectCondition
Dim Countries As DBTable

Set A = New Application
A.GetActiveDocument Doc
Doc.GetFlaggedObjects Flagged
Flagged.ClearSet
```

```

A.InitSelectionCriteria SelCriteria 'Inits all fields to default values.
SelCriteria.SelectionType = ptNewSelection
SelCriteria.Objects = ptPolygon
Redim DBCriteria(0)
DBCriteria(0) = A.SelectCondition("Population", ptCompareGreaterEqual,
10000000)
Flagged.MakeSetDB ptDrawing, SelCriteria, Countries, DBCriteria
' Polygons linked to table Countries with Population >= 10 million.

```

MakeSetTextual

Parameters

OverlayType As Integer, Criterion As String, Optional SelectionType As Integer
= ptNewSelection, Optional Table As DBTable = Nothing,

Description

This method is a variant of MakeSet, where the selection criteria are defined in textual form.

Criterion: a string that must describe a Boolean expression. Eg. "NrChildren > 2". See also *Application* method *Parse*.

The ptCompareAboutEqual parameter for the SelectionCriteria.CommentSelection property cannot be used in this method MakeSetTextual.

Table: if Not Table = Nothing, then the expression may contain field names of the given table.

Note: to avoid errors in the syntax, it may be usefull to call the Application method Parse to check validity of the parameter Criterion.

Example

```

Dim A As Application
Dim Doc As Document
Dim Flagged As FlaggedObjects
Dim Families As DBTable

Set A = New Application
A.GetActiveDocument Doc
Doc.GetFlaggedObjects Flagged
Flagged.ClearSet
Flagged.MakeSetTextual ptDrawing, "NrChildren > 2", ptNewSelection, Families

```

Toggle

Parameters

Obj As CADObject

Description

If Obj is in the set, it is removed from the set. If Obj is not in the set, Obj is added to the set.

This method can not be applied to AllObjects.

Example

```

Dim A As Application
Dim Doc As Document
Dim Obj As CADObject
Dim Flagged As FlaggedObjects

Set A = New Application
A.GetActiveDocument Doc
Doc.GetFlaggedObjects Flagged
Flagged.ClearSet
Flagged.Toggle Obj

```

Include

Parameters

Obj As CADObject

Description

If Obj is not in the set, it is included.

This method can not be applied to AllObjects.

Example

```

Dim A As Application
Dim Doc As Document

```

```

Dim Obj As CADObject
Dim Flagged As FlaggedObjects

Set A = New Application
A.GetActiveDocument Doc
Doc.GetFlaggedObjects Flagged
Flagged.ClearSet
Flagged.Include Obj

```

Exclude

Parameters

Obj As CADObject

Description

If Obj is in the set, it is excluded from the set.
This method can not be applied to AllObjects.

Example

```

Dim A As Application
Dim Doc As Document
Dim Obj As CADObject
Dim Sel As Selection

Set A = New Application
A.GetActiveDocument Doc
Doc.GetSelection Sel
If Sel.InSet(Obj) Then
    Obj.Comment = "ABC"
    Sel.Exclude Obj
End If

```

InSet

Parameters

Obj As CADObject

Description

Returns True if Obj is in the set.
This method may not be applied to AllObjects.

Example

See Exclude

ApplyThematics

Parameters

Attribs as AttributesSet

Description

Applies a thematic representation defined by AttributesSet to all the objects in the set.
This method can only be applied to Selection or to AllObjects, but the objects have to exist on the same sheet. Calling this method for AllObjects removes a thematic defined for the same overlay even if it was defined for a set with different value of Wallpaper.
This method will not automatically update the screen when the macro is terminated. If the macro does not change the screen setting, e.g. zooming, the Document method RedrawDocument must be called.
The AnnotationAttributes used to apply the thematics on annotations must have as kind ptAn_GeneralAnnotation.

Example

```

Dim A As Application
Dim Doc As Document
Dim Ovl as Overlay
Dim All As AllObjects
Dim Attribs as AttributesSet
Dim AnAttribs As AnnotationAttributes
Dim Red As RGB

Set A = New Application
A.GetActiveDocument Doc

```

```

Doc.GetOverlay ptDrawing, Ovl
Ovl.GetAllObjects All
Set Attribs = New AttributesSet
Set AnAttribs = New AnnotationAttributes
AnAttribs.HasTextColor = True
Red.Red = 255
AnAttribs.TextTrueColor = Red
Attribs.SetAnnotationAttributes ptAn_GeneralAnnotation, AnAttribs
All.ApplyThematics Attribs

```

GetThematics

Parameters

Attribs as AttributesSet

Description

Returns the thematic representation that has been applied using ApplyThematics.
This method cannot be applied neither to Selection nor to AllObjects limited to a sheet.
The AnnotationAttributes used to apply the thematics on annotations has as kind ptAn_GeneralAnnotation in Attribs.

Example

```

Dim A As Application
Dim Doc As Document
Dim Ovl as Overlay
Dim All As AllObjects
Dim Attribs as AttributesSet

Set A = New Application
A.GetActiveDocument Doc
Doc.GetOverlay ptDrawing, Ovl
Ovl.GetAllObjects All
All.GetThematics Attribs

```

RemoveThematics

Description

Removes the thematic representation that has been applied to the set. This method cannot be applied neither to Selection nor to AllObjects limited to a sheet.
This method will not automatically update the screen when the macro is terminated. If the macro does not change the screen setting, e.g. zooming, the Document method RedrawDocument must be called.

Example

```

Dim A As Application
Dim Doc As Document
Dim Ovl as Overlay
Dim All As AllObjects
Dim Attribs as AttributesSet

Set A = New Application
A.GetActiveDocument Doc
Doc.GetOverlay ptDrawing, Ovl
Ovl.GetAllObjects All
All.RemoveThematics

```

ExportShape

Parameters

```

Optional Path As String := ""
Optional AskAttribures As Boolean := False
Optional ExportType As Integer := ptShape2D
Optional ExportObjInfo As Boolean := True
Optional ExportInvisiblePoints as Boolean := False
Optional PrefixType as Integer := 0
Optional ExportByLayer As Boolean := False

TBD
tPrefixType = (tNoPrefix, tShortPrefix, tFullPrefix);

```

Description

This method exports all objects in the set to a shape file.

Path needs to be an existing path, if Path = "", a request folder dialog will be opened.

AskAttributes: if = True, a dialog box comes up requesting export parameters. The parameters passed by VBA become the defaults in the dialog.

ExportType defines how you want to export coordinates. Possible values: ptShape2D, ptShape2DM, ptShape3D.

When ExportObjInfo is TRUE table data is exported.

ExportInvisiblePoints defines if you want to include invisible points in the exported files.

When ExportByLayer is TRUE a separate file will be generated per layer.

Export is done in current CRS and in meters.

Objects of the page overlay are not exported even when they have the corresponding flag set. When this method is applied to AllObjects then all objects of the local overlay are exported independently of how the set is defined: for which overlay and sheet and with which value of Wallpaper parameter.

Example

```
Dim A as Application
Dim D as Document
Dim O as Overlay
Dim All as AllObjects

Set A = New Application
A.GetActiveDocument D
If Not (D is Nothing) Then
    D.GetOverlay ptDrawing, O
    O.GetAllObjects All
    All.ExportShape AskAttributes:=TRUE, ExportType:=ptShape2DM, _
        ExportInvisiblePoints:=TRUE
End If
```

GetTerrainModel

Parameters

Name As String,
ReduceParams As DTMRReduce,
TM As TerrainModel

Description

Creates a terrain model with specified Name based on the object set. ReduceParams indicate if and how reduction should be applied.

Only points, lines and polygons are considered with the following restrictions:

If more than one polygon is in the set then all polygons are ignored;

If exactly one polygon is in the set it will be the boundary of the terrain model;

If no polygon is selected there should be points in the set to make up the terrain model;

If there are lines, arcs, circles or clothoids in the object set they will be used as breaklines.

Example

```
Dim A As Application
Dim Doc As Document
Dim SelectedObjects As Selection
Dim Red As DTMRReduce
TM As TerrainModel

Set A = New Application
A.GetActiveDocument Doc
Doc.GetSelection SelectedObjects
Red.Reduce = False
SelectedObjects.GetTerrainModel "SampleTerrainModel", Red, TM
```

Chapter 39 - TerrainModel

Description

The `TerrainModel` object corresponds with an existing Digital Terrain Model in a Pythagoras drawing. This object can be obtained from the set of terrain models, or created via the object `TerrainModels`. See also the object type `TerrainModels`.

Example

```
Dim A As Application
Dim Doc As Document
Dim DTMs As TerrainModels
Dim DTM1 As TerrainModel
Dim Height As Double
Dim Pos As XY

Set A = New Application
A.GetActiveDocument Doc
If NOT Doc Is Nothing Then
    Doc.GetTerrainModels DTMs
    If DTMs.NrOfTerrainModels > 0 Then
        DTMs.FindTerrainModel "Existing Situation", DTM1
        If DTM1 Is Nothing Then
            MsgBox "DTM : 'Existing Situation' does not exist."
        Else
            Pos.x = 30.0
            Pos.y = 20.0
            Height = DTM1.Elevation(Pos)
            If Height = ptNilElevation Then
                MsgBox "Point outside DTM"
            Else
                MsgBox "Height at 30.0, 20.0 = " & CSTR(Height)
            End If
        End If
    End If
End If
end if
```

Properties

Name
zMin
zMax
NrOfTriangles
NrOfVertexes
EnclosingRectangle
Visible

Methods

Boundary
Elevation
AdjustElevationOfVertex
AdjustAllElevations
Slope
GetTriangle
GetFirstTriangle
GetNextTriangle
GetFirstVertex
GetNextVertex
CrossSection
CrossSectionPnts
CalculateVolume
Duplicate
AddVertex
RemoveVertex
MoveVertex
AddTriangle
RemoveTriangle
SwapDiagonal

Properties

Name

Returns
String

Description
Returns the name of the Digital Terrain Model.

Example

```
Dim DTM As TerrainModel

MsgBox "The name of this DTM is " & DTM.Name
```

zMin

Value
Double

Description
Returns the height of the lowest point of the DTM.

Example

```
Dim MinHeight As Double, MaxHeight As Double
Dim DTM As TerrainModel

MinHeight = DTM.zMin
MaxHeight = DTM.zMax
```

zMax

Value
Double

Description
Returns the height of the highest point of the DTM.

Example

```
Dim MinHeight As Double, MaxHeight As Double
Dim DTM As TerrainModel

MinHeight = DTM.zMin
MaxHeight = DTM.zMax
```

NrOfTriangles

Value
Long

Description
Returns the number of triangles of the DTM.

Example

```
Dim DTM As TerrainModel
Dim Count As Long

Count = DTM.NrOfTriangles
```

NrOfVertexes

Value
Long

Description
Returns the number of vertexes of the DTM.

Example

```
Dim DTM As TerrainModel
Dim Count As Long

Count = DTM.NrOfVertexes
```


EnclosingRectangle

Value

Rectangle

Description

Returns the enclosing rectangle of the DTM.

Example

```
Dim DTM As TerrainModel
Dim R As Rectangle

R = DTM.EnclosingRectangle      ' R.p1 = minimum, R.p2 = maximum
```

Visible

Value

Boolean

Description

Returns TRUE if the TerrainModel is visible.

Example

```
Dim DTM As TerrainModel

If DTM.Visible Then
    MsgBox "The DTM is visible."
End If
```

Methods

Boundary

Parameters

NrOfPoints As Long, Points() As XY

Description

Returns the 2D boundary of the terrain model. The number of points and the list of points is returned. For terrains that have holes and terrains that consist of different islands all contours are included in the point list in a random order.

Note: the index of the point list starts from zero. The list does not have to be initialized before this method is called.

Example

See example of CalculateVolume

Elevation

Parameters

Position As XY

Description

Returns the height at Position in the DTM. If the point lies outside the DTM, then ptNilElevation is returned.

Example

```
Dim Height As Double
Dim Pos As XY
Dim DTM1 As TerrainModel

Pos.x = 30.0
Pos.y = 20.0
Height = DTM1.Elevation(Pos)
If Height <> ptNilElevation Then
    'Do Something
End If
```

AdjustElevationOfVertex

Parameters

Vertex As XY

Description

This method adjusts the elevation of the `Vertex` at the specified position. If the position does not match with a vertex, nothing is happening.

Example

```
Dim V As XY
Dim DTM1 As TerrainModel

V.x = 30.0
V.y = 20.0
DTM1.AdjustElevationOfVertex V, 12.34567
```

AdjustAllElevations

Parameters

dZ As Double

Description

This method adjusts the elevation of all vertexes in the DTM with a height difference equal to `dZ`.

Example

```
Dim DTM1 As TerrainModel

DTM1.AdjustAllElevations 1.7
```

Slope

Parameters

Position As XY, Gradient As Double, Direction As Double

Description

Returns the gradient and the downward direction of the slope at `Position` in the DTM. If the point lies outside the DTM, then both values will be = 0. The gradient is tangent of the angle of the slope (dS/dZ).

Example

```
Dim Pos As XY
Dim DTM As TerrainModel
Dim Gradient As Double, Direction As Double

Pos.x = 30.0
Pos.y = 20.0
DTM.Slope Pos, Gradient, Direction
```

GetTriangle

Parameters

Pos As XY, T As Triangle

Description

Returns the triangle of a `TerrainModel`. The projection of the point with coordinates (x, y) will be inside, or on the edge, of the triangle. If `Pos` is outside the Digital Terrain Model, `Nothing` is returned

Example

```
Dim x As Double, y As Double
Dim Pos As XY
Dim DTM As TerrainModel
Dim T As Triangle

Pos.x = 30.0
Pos.y = 20.0
DTM.GetTriangle Pos, T
```

GetFirstTriangle

Parameters

T As Triangle

Description

Returns the first triangle of a TerrainModel. GetFirstTriangle must be called before calling GetNextTriangle.

Example

```
Dim DTM As TerrainModel
Dim P1 As XYZ, P2 As XYZ, P3 As XYZ
Dim T As Triangle

DTM.GetFirstTriangle T
Do While Not(T Is Nothing)
    T.Vertexes P1, P2, P3
    DTM.GetNextTriangle T
Loop
```

GetNextTriangle

Parameters

T As Triangle

Description

Returns the next triangle of a TerrainModel. May only be called after call to GetFirstTriangle.

Example

See GetFirstTriangle

GetFirstVertex

Parameters

Vertex As XYZ

Description

Returns the first vertex of a TerrainModel. GetFirstVertex must be called before calling GetNextVertex.

Example

```
Dim DTM As TerrainModel
Dim V As XYZ
Dim I As Long

I = 1
DTM.GetFirstVertex V
MsgBox "(" & V.X & ", " & V.Y & ", " & V.Z & ")"
Do While I <= DTM.NrOfVertexes Do
    MsgBox "(" & V.X & ", " & V.Y & ", " & V.Z & ")"
    DTM.GetNextVertex V
    I = I + 1
End
```

GetNextVertex

Parameters

Vertex As XYZ

Description

Returns the next vertex of a TerrainModel. May only be called after call to GetFirstVertex.

Example

See GetFirstVertex

CrossSection

Parameters

Obj As CadObject, NrOfXSPoints As Long, XS_Points() As XY

Description

Returns the cross section through the DTM at the positions defined by the CadObject Obj. Obj is either a Line, an Arc, a Clothoid, a Path, a Road or a Polygon. When the object is a line or an arc, the starting point corresponds with the first point of the line or arc. On return XS_Points contains the station (chainage) and the height of all points of intersection of the object with the DTM in increasing order. The parameter NrOfXSPoints contains the number of points in the dynamic array XS_Points.

Points of the cross section that are outside the DTM have the elevation `ptNilElevation`. The segment consisting of a pair or cross section points (`XS_Points(n)`, `XS_Points(n+1)`) where either point `n` or point `n+1` has the elevation `ptNilElevation`, lies outside the DTM.

The array `XS_Points` should be large enough to contain all points of the cross section. If not, an error is raised. If `Obj` is a path, the starting chainage of the path is taken into account.

Example

```
Sub PM_GetCrossSection

Dim A As Application
Dim Doc As Document, Doc2 As Document
Dim DTMs As TerrainModels
Dim DTM1 As TerrainModel
Dim Pth As Path
Dim Obj As CadObject
Dim NrofXSPoints As Long
Dim XS_Points() As XY
Dim SelectedObjects As Selection
Dim i As Long
Dim DrawingXS As Overlay
Dim Pnt As XYZ
Dim P As Point

Set A = New Application
A.GetActiveDocument Doc
If NOT Doc Is Nothing Then
    Doc.GetTerrainModels DTMs
    If DTMs.NrofTerrainModels > 0 Then
        DTMs.FindTerrainModel "Existing Situation", DTM1
        If DTM1 Is Nothing Then
            MsgBox "DTM : 'Existing Situation' does not exist."
        Else
            Doc.GetSelection SelectedObjects
            If SelectedObjects.NrofObjects = 1 Then
                SelectedObjects.GetFirst ptPath, Obj
                If Not (Obj Is Nothing) Then
                    Redim XS_Points(1 To 100000)
                    DTM1.CrossSection Obj, NrofXSPoints, XS_Points
                    A.GetNewDocument Doc2
                    Doc2.GetOverlay ptDrawing, DrawingXS
                    For i = 1 To NrofXSPoints
                        Pnt.x = XS_Points(i).x
                        Pnt.y = XS_Points(i).y
                        Pnt.z = 0.0
                        DrawingXS.CreatePoint Pnt, P
                    Next i
                End If
            Else
                MsgBox "Select a path"
            End If
        End If
    End If
End If
end if

End Sub
```

CrossSectionPnts

Parameters

`NrofPoints` As Long, `Points()` As XY, `NrofXSPoints` As Long, `XS_Points()` As XY

Description

The method `CrossSectionPnts` is very similar to `CrossSection`. The only difference is the definition of the position where to calculate the cross section. In stead of passing a `CadObject`, an array of 2D points defines the position where to calculate the cross section.

Returns the cross section through the DTM at the positions defined by the array `Points`. `NrofPoints` is the number of points in this array. On return `XS_Points` contains the station (chainage) and the height of all points of intersection of the object with the DTM in increasing order. The parameter `NrofXSPoints` contains the number of points in the dynamic array `XS_Points`.

Points of the cross section that are outside the DTM have the elevation `ptNilElevation`. The segment consisting of a pair or cross section points (`XS_Points(n)`, `XS_Points(n+1)`) where either point `n` or point `n+1` has the elevation `ptNilElevation`, lies outside the DTM. The array `XS_Points` should be large enough to contain all points of the cross section. If not, an error is raised.

Example

```
Dim Pnts() As XY
Dim NrofXSPoints As Long
Dim XS_Points() As XY
Dim DTM As TerrainModel

DTM.CrossSectionPnts 2, Pnts, NrofXSPoints, XS_Points
```

CalculateVolume

Parameters

```
Params As VolumeCalculationDTM
ShowProgress As Boolean
```

Returns

```
Result As VolumeCalculationResult
```

Description

Calculates the volume of the terrain model based on the next input parameters:

- `Params.Elevation`: this value defines the horizontal reference level to use in the computation
- `Params.UseDistance`: if this parameter is `True` the `NrofCrossSections` will be deducted from the `Distance`, otherwise it is the other way around.
- `Params.NrofCrossSections`: the number of cross sections to use. This value should be at least 3 or higher. It needs to be defined only when `UseDistance` is `False`.
- `Params.Distance`: the (positive) distance between two cross sections. It needs to be defined only when `UseDistance` is `True`.
Next two parameters indicate the boundary to use. If `NrofBoundaryPoints` is zero the boundary of the entire terrain model will be used.
- `Params.NrofBoundaryPoints`: the number of boundary points
- `Params.BoundaryPoints`: indicates the boundary polygon for the volume calculation. The index of this list of points starts from 0. The last boundary point should be equal to the first boundary point. The coordinates of the points should be passed in local coordinates.
- `ShowProgress`: indicates if a progress should be shown during the calculation or not.

The result of the calculation holds the details of the calculated volume, see the description of the `VolumeCalculationResult` type.

Please note that all input and output parameters should be passed in meters. The calculation will use the active coordinate system to determine the cross sections.

Example

```
Dim App As Application
Dim Doc As Document
Dim LMs As TerrainModels
Dim LM1 As TerrainModel
Dim LM2 As TerrainModel

Dim params As VolumeCalculationDTM
Dim result As VolumeCalculationResult

Set App = New Application
App.GetActiveDocument Doc
Doc.GetTerrainModels LMs
If LMs.NrofTerrainModels > 0 Then
    LMs.GetFirst LM1
    params.Elevation = 0.0
    params.UseDistance = False
    params.NrofCrossSections = 100
    LM1.Boundary params.nrofBoundaryPoints, params.BoundaryPoints 'should give
same result as with nrofBoundaryPoints = 0
    result = LM1.CalculateVolume(params, True)
```

```

        MsgBox Pos=" & result.VolumePlus & " / Neg=" & result.VolumeMin
    End If

```

Duplicate

Parameters

```

Name As String
TM As TerrainModel

```

Description

With this method a user can create a duplicate of an existing terrain model. A unique Name should be passed to identify the new terrain model. If a terrain model with the same name already exists in the document an error is generated.

Example

```

Dim DTM1 As TerrainModel
Dim DTM2 As TerrainModel

DTM1.Duplicate "DuplicatedTM", DTM2

```

AddVertex

Parameters

```

Vertex As XYZ

```

Description

The specified Vertex is added to a TerrainModel. The triangles are recalculated to take this new vertex into account.

Example

```

Dim DTM As TerrainModel
Dim V As XYZ

V.X = -9.0
V.Y = -2.4
V.Z = 10.5
DTM.AddVertex V

```

RemoveVertex

Parameters

```

Vertex As XY

```

Description

The specified Vertex is removed from the TerrainModel. When the specified position does not correspond with a vertex, nothing happens. Otherwise the triangles in the DTM are recalculated.

Example

```

Dim DTM As TerrainModel
Dim V As XY

V.X = -9.0
V.Y = -2.4
DTM.RemoveVertex V

```

MoveVertex

Parameters

```

Vertex As XY
Position As XY

```

Description

The specified Vertex is moved to a new position in a TerrainModel. When the specified original vertex position does not correspond with a vertex, nothing happens. Otherwise the triangles in the DTM are recalculated. The new vertex position should be inside the triangle where the original vertex position was.

Example

```

Dim DTM As TerrainModel
Dim V As Vertex, V1 As Vertex

V.X = -9.0

```

```
V.Y = -2.4  
V1.X = -10.0  
V1.Y = -1.4  
DTM.MoveVertex V, V1
```

AddTriangle

Parameters

```
P1 As XYZ  
P2 As XYZ  
P3 As XYZ  
T As Triangle
```

Description

The triangle specified with 3 points is added to a `TerrainModel`. The created triangle is returned .

Example

```
Dim P1 As XYZ  
Dim P2 As XYZ  
Dim P3 As XYZ  
Dim T As Triangle  
  
DTM.AddTriangle P1, P2, P3, T
```

RemoveTriangle

Parameters

```
T As Triangle
```

Description

The specified triangle is removed from a `TerrainModel`.

Example

```
Dim DTM As TerrainModel  
  
DTM.RemoveTriangle T
```

SwapDiagonal

Parameters

```
T1 As Triangle  
T2 As Triangle
```

Description

The diagonal between the two given triangles is swapped. Both triangles should share an edge to make the operation possible.

Example

```
Dim T1 As Triangle  
Dim T2 As Triangle  
  
DTM.SwapDiagonal T1, T2
```

Chapter 40 - TerrainModels

Description

The object `TerrainModels` refers to the set of all Terrain Models in a Pythagoras drawing. An object of this type refers always to the actual situation. So it will never contain DTM's which are deleted. This object can not be created, but can be obtained with the Document method `GetTerrainModels`.

Example

See object type `TerrainModel`

Properties

`NrOfTerrainModels`

Methods

`GetFirst`
`GetNext`
`FindTerrainModel`
`NewTerrainModel`
`DeleteTerrainModel`
`CalculateVolume`

Properties

NrOfTerrainModels

Returns

Long

Description

Returns the number of the Digital Terrain Models in the document.

Example

```
Dim DTMs As TerrainModels

MsgBox "Number of DTM's = " & DTMs.NrOfTerrainModels
```

Methods

GetFirst

Parameters

DTM As `TerrainModel`, Optional `VisibleOnly` as Boolean

Description

Returns the first DTM in the drawing. `GetFirst` must be called before calling `GetNext`.
If `VisibleOnly` is `TRUE`, only visible `TerrainModels` will be returned.
If no visible `TerrainModels` exist, nothing will be returned.

Example

```
Dim DTMs As TerrainModels
Dim DTM1 As TerrainModel
Dim DTM2 As TerrainModel

DTMs.GetFirst DTM1
DTMs.GetFirst DTM2, VisibleOnly := True
```

GetNext

Parameters

DTM As `TerrainModel`, Optional `VisibleOnly` as Boolean

Description

Returns the next DTM in the set.
If `VisibleOnly` is `TRUE`, only visible `TerrainModels` will be returned.
If no visible `TerrainModels` exist, nothing will be returned.

Example

```
Dim DTMs As TerrainModels
Dim DTM As TerrainModel

DTMs.GetFirst DTM
While Not DTM Is Nothing
    DTMs.GetNext DTM
Loop
DTMs.GetFirst DTM, VisibleOnly := True
While Not DTM Is Nothing
    DTMs.GetNext DTM, VisibleOnly := True
Loop
```

FindTerrainModel

Parameters

Name As String, DTM As TerrainModel

Description

Returns the DTM with the requested name. If no DTM with this name exists, then DTM = Nothing.

Example

```
Dim DTMs As TerrainModels
Dim DTM1 As TerrainModel

If DTMs.NrOfTerrainModels >= 1 Then
    DTMs.FindTerrainModel "Ground Works", DTM1
End If
```

NewTerrainModel

Parameters

Name As String,
Reduce As DTMRReduce,
ShowProgress As Boolean,
NrOfPoints As Long,
Points As XYZ(),
NrBoundaryPoints As Long,
BoundaryPoints() As XYZ

Returns

TerrainModel

Description

Creates a terrain model with the requested parameters:

Name indicates the name of the new DTM.

Reduce specifies if and how reduction should be applied during the calculation.

If ShowProgress is true progress bar will be shown during the calculation.

NrOfPoints and Points define the 3D points that are used. This list should hold at least three different positions.

NrOfBoundaryPoints and BoundaryPoints gives you the possibility to use a boundary to for the calculation. Only points that are contained in this boundary polygon will be used. When this list is empty all positions in Points will be used in the calculation.

TBD: Possibility to use break lines.

Note: Indexes of the lists are starting from zero.

Example

```
Dim DTMs As TerrainModels
Dim ps() As XYZ
Dim bs() As XYZ
Dim Red As DTMRReduce
TM As TerrainModel

Set tm = DTMs.NewTerrainModel("NewDTM", Red, False, 3, ps, 4, bs)
```

DeleteTerrainModel

Parameters

DTM As TerrainModel

Description

Deletes the DTM. On return DTM = Nothing.

Example

```
Dim DTMs As TerrainModels
Dim DTM As TerrainModel

DTMs.DeleteTerrainModel DTM
```

CalculateVolume

Parameters

Params As VolumeCalculationDTMs
ShowProgress As Boolean

Returns

Result As VolumeCalculationResult

Description

Calculates the volume of the difference of two terrain models.

- Params.LMName1: the name of the first terrain model.
- Params.LMName2: the name of the second terrain model.
- Params.UseDistance: if this parameter is True the NrOfCrossSections will be deducted from the Distance, otherwise it is the other way around.
- Params.NrOfCrossSections: the number of cross sections to use. This value should be at least 3 or higher. It needs to be defined only when UseDistance is False.
- Params.Distance: the (positive) distance between two cross sections. It needs to be defined only when UseDistance is True.
Next two parameters indicate the boundary to use. If NrOfBoundaryPoints is zero the boundary of the entire terrain model will be used.
- Params.NrOfBoundaryPoints: the number of boundary points
- Params.BoundaryPoints: indicates the boundary polygon for the volume calculation. The index of this list of points starts from 0. The last boundary point should be equal to the first boundary point. The coordinates of the points should be passed in local coordinates.
- ShowProgress: indicates if a progress should be shown during the calculation or not.

The result of the calculation holds the details of the calculated volume, see the description of the VolumeCalculationResult type.

Please note that all input and output parameters should be passed in meters. The calculation will use the active coordinate system to determine the cross sections.

Example

```
Dim App As Application
Dim Doc As Document
Dim LMs As TerrainModels
Dim LM1 As TerrainModel
Dim LM2 As TerrainModel

Dim params As VolumeCalculationDTMs
Dim result As VolumeCalculationResult

Set App = New Application
App.GetActiveDocument Doc
Doc.GetTerrainModels LMs
If LMs.NrOfTerrainModels > 0 Then
    LMs.GetFirst LM1
    LMs.GetNext LM2
    params.LMName1 = LM1.Name
    params.LMName2 = LM2.Name
    params.UseDistance = False
    params.NrOfCrossSections = 100
    GetBoundary params.nrofBoundaryPoints, params.BoundaryPoints 'a specific
    procedure to define the boundary
    result = LMs.CalculateVolume(params, True)

    MsgBox "Accuracy=" & result.Accuracy & " / Pos=" & result.VolumePlus & " /
    Neg=" & result.VolumeMin & " / Extrapolated=" & result.Extrapolated & " /
```

```
PosExtr=" & result.ExtrapolatedVolumePlus & " / NegExtr=" &  
result.ExtrapolatedVolumeMin  
End If
```

Chapter 41 - Triangle

Description

The **Triangle** object corresponds with an existing triangle of a Digital Terrain Model in a Pythagoras drawing. This object can not be created, but a reference to an existing Triangle can be obtained from a **TerrainModel** with the methods **GetFirstTriangle**, **GetNextTriangle** or **GetTriangle**.

Example

```
Dim A As Application
Dim Doc As Document
Dim DTMs As TerrainModels
Dim DTM1 As TerrainModel
Dim T As Triangle
Dim Pos As XY
Dim Height As Double

Set A = New Application
A.GetActiveDocument Doc
If Not Doc Is Nothing Then
    Doc.GetTerrainModels DTMs
    If DTMs.NrOfTerrainModels > 0 Then
        DTMs.FindTerrainModel "Existing Situation", DTM1
        If DTM1 Is Nothing Then
            MsgBox "DTM : 'Existing Situation' does not exist."
        Else
            Pos.x = 30.0
            Pos.y = 20.0
            DTM1.GetTriangle Pos, T
            If T Is Nothing Then
                MsgBox "Point (30.0, 20.0) is outside DTM"
            Else
                MsgBox "Point (30.0, 20.0) is inside DTM"
            End If
        End If
    End If
End If
```

Methods

Adjacent
Vertexes

Methods

Adjacent

Parameters

T1 As Triangle, T2 As Triangle, T3 As Triangle

Description

Returns the adjacent triangles. The adjacent triangles are enumerated in counterclockwise order and in correspondence with triangle's vertexes (see **Vertexes**). T1 is adjacent to the edge P1-P2, T2 is adjacent to the edge P2-P3, and T3 is adjacent to the edge P3-P1.

Example

```
Dim T As Triangle, T1 As Triangle, T2 As Triangle, T3 As Triangle
Dim P1 As XYZ, P2 As XYZ, P3 As XYZ

T.Adjacent T1, T2, T3
T3.Vertexes P1, P2, P3
```

Vertexes

Parameters

P1 As XYZ, P2 As XYZ, P3 As XYZ

Description

Returns the vertexes of a triangle. The vertexes are enumerated in counterclockwise order and in correspondence with the adjacent triangles (see **Adjacent**).

Example

```
Dim T As Triangle, T1 As Triangle, T2 As Triangle, T3 As Triangle
Dim P1 As XYZ, P2 As XYZ, P3 As XYZ

T.Adjacent T1, T2, T3
T3.Vertexes P1, P2, P3
```

Chapter 42 - Database

Description

An object of this class describes a database, being either a document database or a so-called Pythagoras database.

Pythagoras databases are databases existing out of Pythagoras documents. They are created by VBA Set New statement.

Example

```
Dim D As Database
Set D = New Database
' Do something with created database
D.CloseDatabase
```

Properties

Document
LockedDatabase
DifferentDatabaseLocks
NrOfTables
NrOfAttributeTables
NrOfDictionaryTables
NrOfDBViews

Methods

FirstTable
NextTable
TableByName
CreateTable
ImportDBF
DropTable
CopyTableSkeleton
CopyTable
FirstDBView
NextDBView
DBViewByName
CreateSimpleQuery
CreateAggregationQuery
DropQuery
CopyQuery
CopyDatabaseSkeleton
CopyDatabase
PurgeDatabase
DropDatabase
CloseDatabase
NewState
EndState
LockDatabase
UnlockDatabase
ActualizeQueries

Properties

Document

Returns

Document

Description

This property returns the document the database belongs to. If the database belongs to no document Nothing is returned.

Example

```
Dim D As Database
If D.Document Is Nothing Then
    ' If this database is not a document one
    ' Do something
End If
```

LockedDatabase

Returns

Boolean

Description

Returns true if the database is locked.

Example

```
Dim D As Database

If D.LockedDatabase then
    MsgBox "Database is locked"
End If
```

DifferentDatabaseLocks

Returns

Boolean

Description

Returns true if the database belongs to a document and the document contains subdocuments and the locking status (Locked/Unlocked) or, owner name and/or password, of the databases in the main document / subdocuments is not the same.

If main document and subdocuments have different locking, the database always remains locked. See also Subdocument method DifferentDatabaseLock.

Being applied to Pythagoras databases this property always returns False.

Example

```
Dim D As Database

If D.DifferentDatabaseLocks then
    MsgBox "Not all subdocuments are locked in the same way"
End If
```

NrOfTables

Returns

Integer

Description

Returns the number of tables in the database.

Example

```
Dim D As Database
Dim NrTables As Integer

NrTables = D.NrOfTables
```

NrOfAttributeTables

Returns

Integer

Description

Returns the number of attribute tables in the database. Being applied to Pythagoras database this property always returns 0.

Example

```
Dim D As Database
Dim NrTables As Integer

NrTables = D.NrOfAttributeTables
```

NrOfDictionaryTables

Returns

Integer

Description

Returns the number of dictionary tables in the database. Being applied to Pythagoras database this property is the same as `NrOfTables`.

Example

```
Dim D As Database
Dim NrTables As Integer

NrTables = D.NrOfDictionaryTables
```

NrOfDBViews

Returns

Integer

Description

Returns the number of views in the database.

Example

```
Dim D As Database
Dim NrViews As Integer

NrViews = D.NrOfViews
```

Methods

FirstTable

Parameters

None

Returns

Table As DBTable

Description

`FirstTable` returns the first table of the database. Returns `Nothing` if the database contains no table.

Example

```
Dim D As Database
Dim T As DBTable
Dim S As String

S = "List of tables:"
Set T = D.FirstTable()
While Not T Is Nothing
    S = S & T.Name & " "
    Set T = D.NextTable()
WEnd
```

NextTable

Parameters

None

Returns

Table As DBTable

Description

`NextTable` returns the next table of the database. Returns `Nothing` if the database contains no more tables.

Example

See example `FirstTable`

TableByName

Parameters

Name As String

Returns

Table As DBTable

Description

TableByName returns the table with the given name. Returns Nothing if the database contains no table with such name.

Example

```
Dim D As Database
Dim T As DBTable

Set T = D.TableByName("Countries")
```

CreateTable

Parameters

TableType As Byte, Name As String, Optional Description As String,
Optional ObjectTypes As Long

Returns

DBTable

Description

CreateTable creates a new table in the database. The table will contain no fields. For document databases access permission is initialized to ptFullAccess.

This method will be normally be used to create a table. Subsequently columns need to be added. See DBTable method AddColumn.

TableType : either ptDictionary = table not linked to CadObjects, or ptAttrTable (only in document databases) = table linked to CadObjects.

Name : the name of the new table. The name must be valid. See also Application property NameCategory.

Description : description of the table. By default the string is empty.

ObjectTypes : Pythagoras object types that may link to this table. Eg: ptPoint + ptPolygon.

Note: a coordinate system can not be linked to a table.

This parameter should be omitted or be equal to 0 if Type = ptDictionary.

If TableType = ptAttrTable then the parameter must be given and must be different from 0.

Example

```
Dim D As Database
Dim T As DBTable

Set D = New Database
Set T = D.CreateTable (ptDictionary, "Countries")
```

ImportDBF

Parameters

FileName As String, Optional TableName As String = "",
Optional CodePage As Long = 0,
Optional Completion As PtImportCompletion

Returns

DBTable

Description

ImportDBF reads a DBF file and lay down read data to a new dictionary table in the database. The new table will be an exact copy of the file, i.e. it will contains the same number of columns with the same names and same data types and it will contain the same number of rows with same data.

FileName: a full path to DBF file, name extension may be omitted.

TableName : the name of the new table. If it is omitted or is an empty string then pure name of the file will be taken as a name of the new table.

CodePage defines Window code page used to convert string data to Unicode during import. It should be equal to a code page, string data are in the file in.

If CodePage = 0 (default) then the code page indicated inside the file (in so-called code page mark field) is used; if code page mark = 0 or indicates an unsupported encoding then the code page installed as default in Windows (Control Panel→Regional and Language Options→Advanced) is used

If CodePage is positive then code page with this number is used. If the number is illegal then the further algorithm is the same as for CodePage = 0 (see above).

Negative `CodePage` defines a default code page. If the code page mark field defines some supported code page, it is used. Otherwise the code page with number equal to an absolute value of the parameter is used. If the parameter value is also illegal then the default Windows code page is used. `Completion` is filled by the completion code (see also `PtImportCompletion` description). String values are right-trimmed during import. Column of type `N` with decimal digits number = 0 are converted to columns of type `vbLong`, `vbDouble` or `vbString` dependently of actual data being in these columns. In case of success the created database table is returned otherwise `Nothing` is returned. The current version of the method has the following limitations:

- type of file: `dBase` or `FoxBase`
- data encryption is not supported
- types of fields: `C`, `D`, `F`, `L`, `M`, and `N`
- field flags ignored, binary data is not supported

Example

```
Dim D As Database
Dim T As DBTable

Set D = New Database
Set T = D.ImportDBF ("C:\Data\Population", CodePage:=-1252)
If T Is Nothing Then
    MsgBox ("Import of C:\Data\Population.dbf Failed")
End If
' String data from file are converted correspondingly
' to the code page mark value.
' If code page mark = 0
' then Western Europe Windows encoding is assumed.
```

DropTable

Parameters

Table As DBTable

Description

This method removes the table from the database. All data are lost.

Example

```
Dim D As Database
Dim T As DBTable

D.DropTable T
```

CopyTableSkeleton

Parameters

Table As DBTable, Optional WithReferences As Boolean = False

Returns

DBTable

Description

This method creates and returns a new, but empty, table in the current database with the same properties as Table. Table is assumed to belong to another database.

If `WithReferences` = False then columns with attribute `ReferencedColumn` being not `Nothing` in the source table will get the attribute `ReferencedColumn` = `Nothing` in the new created table.

If `WithReferences` = True then for each column with attribute `ReferencedColumn` <> `Nothing` an attempt is made to find in the current database a table and its column with the same names as in the source database. If all references are found, then the attributes `ReferencedColumn` are copied, otherwise an error is raised and the table skeleton is not copied.

Example

```
Dim D1 As Database, D2 As Database
Dim T1 As DBTable, T2 As DBTable

Set T1 = D1.TableByName("Countries")
Set T2 = D2.CopyTableSkeleton(T1, WithReferences:=True)
```

CopyTable

Parameters

Table As DBTable, Optional WithReferences As Boolean = False

Returns

Table As DBTable

Description

This method creates and returns a new table in the current database with the same properties as Table. Additionally it copies all rows. Table must be a dictionary table. Table is assumed to belong to another database.

WithReferences: see CopyTableSkeleton.

After copying, the new table may contain pending references, i.e. references to other tables having no matches. These references may be corrected by calling the PurgeDatabase method.

Example

```
Dim A As Application
Dim D1 As Database, D2 As Database
Dim T1 As DBTable, T2 As DBTable

Set T1 = D1.TableByName("Countries")
Set T2 = D2.CopyTable(T1, WithReferences:=True)
```

FirstDBView

Parameters

None

Returns

DBQuery

Description

FirstDBView returns the first view of the database. Returns Nothing if the database contains no view.

Example

```
Dim D As Database
Dim S As String
Dim V As DBQuery

S = "List of views:"
Set V = D.FirstDBView()
While Not V Is Nothing
    S = S & V.Name & " "
    Set V = D.NextDBView()
WEnd
```

NextDBView

Parameters

None

Returns

DBQuery

Description

NextDBView returns the next view of the database. Returns Nothing if the database contains no more views.

Example

See example FirstDBView

DBViewByName

Parameters

Name As String

Returns

DBQuery

Description

DBViewByName returns a view the given name. Returns Nothing if the database contains no such view.

Example

```
Dim D As Database
Dim V As DBQuery

Set V = D.DBViewByName("PopulationDensity")
```

CreateSimpleQuery

Parameters

```
Base As RowSetDefinition, Optional Name As String = "",
Optional Description As String = "",
Optional Where As String = "",
Optional Preferences As Units = Nothing,
```

Returns

DBQuery

Description

CreateSimpleQuery creates a new simple query in the database. The query will contain no non-textual where conditions and no columns. New query will be either a view or a temporary query; see description of class DBQuery for discussion of DB views and temporary queries.

This method will be normally used to create a simple query using VBA. Subsequently non-textual where conditions and columns can be added. See class DBQuery methods AddWhere and AddColumn.

Base defines the main base rowset (a table, a set of CAD objects or another query). A temporary query cannot be a base of a view. When defining a new DB view and Base.RowSetType=ptObjectSet, then Base.ObjectTypes should have only a value compatible with one of system tables, i.e. one of the following values: ptAllDBLinkable,

ptLine+ptArc+ptCircle+ptCurve+ptPath+ptClothoid+ptRoad, ptPont, ptLine, ptArc, ptCircle, ptClothoid, ptCurve, ptPolygon, ptText+ptRichText, ptRoad, ptPath, pt3DSolid.

Name : the name of the new view. The name must be valid or empty. See also Application property NameCategory. If the name is empty then a temporary query is created otherwise a DB view is created.

Description : description of the query.

Where : starting value of property Where; see its description.

Preferences : Preferences used to calculate query values. By default the current preferences of the session are used. See also class DBQuery property Preferences.

Unlike CreateTable method this and next methods almost always create a new query. But depending on query definition and presence in the database of referenced tables or queries the created query can be valid or not. See also class DBQuery property Validity.

Example

```
Dim D As Database
Dim RowSet As RowSetDefinition
Dim V As DBQuery

RowSet.RowSetType = ptTable
RowSet.Name = "Countries"
Set V = D.CreateSimpleQuery (RowSet, "CountriesView")
```

CreateAggregationQuery

Parameters

```
MainBase As RowSetDefinition, AuxBase As RowSetDefinition,
Match As MatchCondition,
Optional Name As String = "", Optional Description As String = "",
Optional Where As String = "",
Optional Preferences As Units = Nothing
```

Returns

DBQuery

Description

CreateAggregationQuery creates a new database aggregation query in the database. The query will contain no non-textual where conditions and no columns. New query will be either a view or a temporary query; see description of class DBQuery for discussion of DB views and temporary queries.

This method will be normally be used to create an aggregation query using VBA. Subsequently non-textual where conditions and columns can be added. See DBQuery methods AddWhere and AddColumn.

MainBase and AuxBase define the base rowsets; restrictions on possible values are the same as for Base parameter of CreateSimpleQuery method.

Match : a condition defining which pairs of rows of both bases correspond, see class DBQuery property Match. Name, Description, Where and Preferences have the same meanings as described in the previous method.

Example

```
Dim A As Application, D As Database
Dim Main As RowSetDefinition, Aux As RowSetDefinition
Dim V As DBQuery

Main.RowSetType = ptTable
Main.Name = "Countries"
Aux.RowSetType = ptTable
Aux.Name = "Cities"
Set V = _
    D.CreateAggregationQuery (Main, Aux, _
        A.MatchCondition (PtConditionDistance, Distance:=0.0), _
        "CountriesWithCities")
```

DropQuery

Parameters

Query As DBQuery

Description

This method removes the query from the database.

Example

```
Dim D As Database
Dim Q As DBQuery
```

```
D.DropQuery Q
```

CopyQuery

Parameters

Query As DBQuery

Returns

DBQuery

Description

This method creates and returns a new query in this database with the same definition as Query.

Query as a rule belongs to another database. Note that a DB view cannot be copied to the same database (a name is repeated!). When a temporary query referencing other temporary queries is copied to another database, such references are lost.

Example

```
Dim A As Application
Dim Doc1 As Document, Doc2 As Document
Dim V1 As DBQuery, V2 As DBQuery

Set V1 = Doc1.QueryByName("CountriesWithCities")
Set V2 = Doc2.CopyQuery(V1)
```

CopyDatabaseSkeleton

Parameters

Source As Variant, Optional WithViews As Boolean = True

Description

CopyDatabaseSkeleton deletes all tables and queries from the database and then creates new tables with the same properties as the tables in Source. Source may be either Document or Database. If one of the databases does not belong to a document then attribute tables are not copied. The new tables are empty.

If WithViews = True then all views are also copied from Source.

Example

```
Dim D1 As Database, D2 As Document
```

D1.CopyDatabaseSkeleton D2

CopyDatabase

Parameters

Source As Variant, Optional WithViews As Boolean = True

Description

CopyDatabase deletes all tables and queries from the database and then creates new tables with the same properties. Source may be either Document or Database. If one of the databases does not belong to a document then attribute tables are not copied. The new attribute tables are empty. Data of directory tables are copied

If WithViews = True then all views are also copied from Source.

Example

```
Dim D1 As Database, D2 As Document
```

```
D1.CopyDatabase D2
```

PurgeDatabase

Parameters

None

Description

PurgeDatabase removes all pending references. This means that all columns that contain references :

(1) DataType=vbObject

(2) Not ReferencedColumn Is Nothing and containing a value (not being Null)

are investigated.

ptPendingLeave : leaves pending references untouched.

ptPendingNull : pending references are set = Null.

ptPendingDelete : pending references are removed.

This method eliminates all pending references. A not NULL value is considered to be a pending reference when one of the following conditions are met :

1) DataType=vbObject for this column and there is no CAD object with such identifier;

2) Not ReferencedColumn Is Nothing and there is no row with such value in the referenced column.

All pending references in columns where PendingAction=ptPendingNull are changed to Null.

All rows with pending references in columns where PendingAction=ptPendingDelete are deleted.

See also DBTableColumn properties ReferencedColumn and PendingAction.

Example

```
Dim D As Database
```

```
D.PurgeDatabase
```

DropDatabase

Parameters

None

Description

DropDatabase deletes all tables and queries except system tables. All data are lost.

Note: to eliminate a Pythagoras database from memory rather CloseDatabase then this method is to be used.

Example

```
Dim D As Database
```

```
D.DropDatabase
```

CloseDatabase

Parameters

None

Description

This method is ignored being applied to a document database. Otherwise it deletes the database from the memory. On return the database object becomes `Nothing`. This method is also called implicitly when the last reference to the Pythagoras database is cleared. Here a reference is understood as either a direct reference—an object of class `Database` or an indirect reference—an object of another class (`DBTable`, `DDBRow` etc.).

Example

```
Dim DB As Database

Set DB = New Database
' Do something with Database
DB.CloseDatabase
```

An example of implicit call of the method is the following:

```
Dim DB As Database

Set DB = New Database
' Do something with Database
Set DB = Nothing
' if no other variable references the database
' it is deleted from the memory.
```

NewState

Parameters

None

Description

`NewState` starts a new state in Pythagoras database so that all operations done on a database after `NewState` call can be undone by `EndState False` call. When a VB program contains no `Database.NewState` method call(s), Pythagoras will automatically add a new state for each Pythagoras database that is changed by the VB program. This call cannot be applied to a document database.

Example

See example `EndState`.

EndState

Parameters

`KeepChanges` As Boolean

Description

This method terminates the state initiated by `NewState`. If the parameter `KeepChanges = False` then all changes made on the document between `NewState` and `EndState` will be removed. This feature is useful to undo changes when an operation cannot be successfully terminated. `EndState True` has an only effect that any subsequent change of the database will open a new state. This call cannot be applied to a document database.

Example

```
Dim DB As Database

On Error GoTo ErrorHandler
DB.NewState

' processing & creating database objects.
' If an error occurs, the code after the label ErrorHandler will restore
' the database to the original state

Exit Sub

ErrorHandler :
Doc.EndState False      ' Restores the database to the previous state.
```

LockDatabase

Parameters

Owner As String, Password As String

Returns

Boolean

Description

LockDatabase locks the database. The Owner and Password are remembered and then are to be stored to the file. They are required to unlock the database. Return is True if the database becomes locked. Return is False if the database is already locked. The method can be applied to Pythagoras database but does not seem to be useful.

When the database is locked database structure cannot be changed, either tables or views. Particularly, DBTable.Access property cannot be changed. This method combined with may DBTable.Access property be used to protect database structure and data from occasional change out of the macro.

Example

```
Dim D As Database, Ok As Boolean, Table As DBTable

Sub ProcessPersons
    Ok = D.Unlock ("Admin", "v1fG9ty")
    On Error GoTo LErr
    Set Table = D.TableByName ("Person")
    Table.Access = ptReadWrite
    ' process data
    Table.Access = ptReadOnly
    Ok = D.Lock ("Admin", "v1fG9ty")
    Exit Sub
LErr:
    Table.Access = ptReadOnly
    Ok = D.Lock ("Admin", "v1fG9ty")
    Err.Raise Err.Number, Err.Source, Err.Description, _
        Err.HelpFile, Err.HelpContext
End Sub
```

UnlockDatabase

Parameters

Owner As String, Password As String

Returns

Boolean

Description

UnlockDatabase unlocks the database if the Owner and Password match with the values defined in LockDatabase method. Return is True if either the database was locked and Owner and Password are correct, or the database was unlocked.

Example

See example LockDatabase

ActualizeQueries

Parameters

None

Description

This method recalculates some internal structures in the database to guarantee query data be actual.

Data returned by some query depends on table data and for document databases also from CAD object properties. To accelerate getting query data some internal structures are built. For reason of speed these internal structures are not recalculated each time when table or object data changes. This means that if the program goes through rows of a query and in parallel changes table data or object properties then some gotten query row data could be wrong. A call of this method guarantees that the program will get correct query data at least until next table data or object property change.

Note that query data is actualized also in the following cases:

- When subdocument is added, deleted, is awaked up or goes to sleep
- When one of the following methods is called: Document.NewState, Document.EndState, Database.NewState or Database.EndState.

- When structure of the database changes.

Note also that as a result of query data actualization position in a query may be lost, and the program may get error 2090 when trying to get next query row.

Example

```
Dim Q As DBQuery, Row As DBRow, Id As Long, P As Point

P.PointIdent = "23874"
' This method call may change values in query Q
Q.Database.ActualizeQueries
Row = Q.RowById(Id) ' Reread a row
```

Chapter 43 - DBTable

Description

The DBTable object corresponds with a database table in a Pythagoras drawing. There are 2 types of tables :

- (1) Attribute table : each row in an attribute table linked to one CadObject.

Example : table *Countries* containing fields *Population*, *Area*, *Language*. Polygons that represent countries link to the rows of the table.

- (2) Dictionary table : a table that is not linked or not directly linked to CadObjects.

Example : table *Languages* containing fields *Name*, *Description*. Table contains a list of all languages with a description.

Attribute tables and dictionaries can refer to dictionary tables. Attribute tables can also refer to CAD objects. See DBTableColumn class ReferencedColumn property description.

Documents containing subdocuments:

When reading subdocuments a conflict can occur that there are 2 or more tables with the same name.

1. Tables with the same name and different structure: this is not allowed. A subdocument will remain sleeping.

2. Attribute tables with same name and identical structure: the tables are joined together (Union), if structure of the table changes, the table is updated in all subdocuments.

3. Dictionary tables with same name and identical structure but different data: this situation is not allowed. A subdocument will remain sleeping.

4. Dictionary tables with same name and identical structure and data: a table in subdocument is ignored. If the structure or data of the table change, the table is updated in all subdocuments.

Example

Option MSCompatible

```
Dim A As Application
Dim Doc As Document
Dim Country As DBTable, Place As DBTable
Dim CountryCode As DBTableColumn
Dim Column As DBTableColumn
Dim Row As DBRow
Dim Values() As ColumnValue
Dim Coord As XYZ
Dim Ov As Overlay
Dim P As Point
Dim Brussels As Point

Set A = New Application
A.GetNewDocument Doc

Doc.GetOverlay ptDrawing, Ov

'Create Table
Set Country = Doc.CreateTable(ptDictionary, "Country")
Set Place = Doc.CreateTable(ptAttrTable, "Place", ObjectTypes:=ptPoint)

'Add Columns
Set CountryCode = Country.AddColumn("Code", vbString, Indexed:=True, _
    MayBeNull:=False)
Set Column = Country.AddColumn("Name", vbString)
Set Column = Place.AddColumn("Name", vbString)
Column.Indexed = True
Set Column = Place.AddColumn("Country", vbString, _
    ReferencedColumn:=CountryCode, MayBeNull:=False)
Column.Description = "Country code by ISO 3166"
Set Column = Place.AddColumn("Population", vbLong)
Column.Minimum = 1000
Column.Maximum = 1000000000

' Insert countries
Redim Values(0 To 1)
Values(0) = A.ColumnValue("Code", "B")
Values(1) = A.ColumnValue("Name", "Belgium")
```

```

Country.InsertRow Values

' Insert places: both points and table rows
Redim Values(0 To 2)
Coord.X = 0.
Coord.Y = 7.
Coord.Z = 0.
Ov.CreatePoint Coord, Brussels
Values(0) = A.ColumnValue("Name", "Brussels")
Values(1) = A.ColumnValue("Country", "B")
Values(2) = A.ColumnValue("Population", 1000000)
Brussels.SetData Place, Values

```

Properties

```

Document
Name
Description
TableType
ObjectTypes
Access
NrOfColumns
NrOfRows
DataStamp
Database

```

Methods

```

Identical
GetColumn
AddColumn
DropColumn
InsertRow
FirstRow      (obsolete)
NextRow       (obsolete)
LastRow       (obsolete)
PreviousRow   (obsolete)
GetRows
RowById
DeleteRow
FirstSubDocument
NextSubDocument
GetSubDocuments
Clear
FastClear
CopyData

```

Properties

Document

Returns

```
Doc As Document
```

Description

Returns a reference to the document the table belongs to.

Example

```

Dim T As DBTable
Dim D As Document

Set D = T.Document

```

Name

Returns

```
String
```

Description

Sets or returns the name of the table. Table names in Pythagoras are not case sensitive.

Example

```
Dim T As DBTable
```

```
T.Name = "Countries"
MsgBox T.Name
```

Description

Returns
String

Description
Sets or returns the description of a table.

Example

```
Dim T As DBTable

T.Description = "Statistical data of all countries"
```

TableType

Returns
Byte

Description
Returns the type of the table. The return value is either ptAttrTable or ptDictionary.

Example

```
Dim T As DBTable

If T.TableType = ptAttrTable Then
    'Do Something
End If
```

ObjectTypes

Returns
Long

Description
Returns or sets the CAD Object types that may be linked to this table. ObjectTypes must be equal to 0 if Type = ptDictionary. If Type = ptAttrTable then ObjectTypes may not be equal to 0. Its value is either ptPoint, ptLine, ptArc, ptClothoid, ptText, ptRichText, ptImage, ptCurve, ptPath, ptPolygon, ptCircle, ptRoad, pt3DSolid or the sum of any combination of these constants. The constant ptAllDBLinkable is equal to the sum of all the above object types. To test if the table can be linked to one or more object types, the logical operator And, may be used. See example.

Example

```
Dim T1 As DBTable, T2 As DBTable, T3 As DBTable
Dim D As Document

T1.ObjectTypes = ptAllDBLinkable
T2.ObjectTypes = ptLine + ptArc + ptClothoid + ptCurve
T3.ObjectTypes = T2.ObjectTypes
If (T1.ObjectTypes And ptImage) <> 0 Then
    MsgBox "Images may be linked to table"
End If
```

Access

Returns
Byte

Description
Returns or sets the access rights of the table. A table can have the following access rights:

- ptNoAccess The data is read and write protected. Read, write and delete are not permitted.
- ptReadOnly The data may only be read.
- ptReadWrite The data may be read, written or deleted.
- ptFullAccess No restrictions.

Only when `Access = ptFullAccess`, the structure and properties of the table may be changed. In all the other cases the structure and properties of the table may not be changed.

This property in combination with the `Document` property `LockedDatabase` can be used to control the way a database can be accessed. If the user does not know the owner and password of locked database, only a VBA macro that temporarily unlocks the database can control the way the user reads and modifies the data. The macro can have forms to control these actions.

Example

```
Dim T As DBTable

T.Access = ptReadOnly
```

NrOfColumns

Returns

Long

Description

Returns the number of columns (fields) in the table.

Example

```
Dim T As DBTable
Dim NrColumns As Long

NrColumns = T.NrOfColumns
```

NrOfRows

Returns

Long

Description

Returns the number of rows in the table.

Example

```
Dim T As DBTable
Dim Size As Long

Size = T.NrOfRows
```

DataStamp

Returns

UUID

Description

Returns a unique value that reflects the state of all data (all rows) of a dictionary table. This property can be used to check if the contents of dictionary tables in 2 documents are identical. The structure of the table (names of columns, etc) is not included in the `DataStamp`, only the content. To compare the structure, the method `Identical` should be used.

For attribute tables this option is not supported. `NilUUID` is always returned.

Note : Equality of data stamp guarantees that the data of 2 tables is identical. However two identical tables, e.g. when they have been filled separately, may have different data stamps.

Example

```
Dim A As Application
Dim T1 As DBTable, T2 As DBTable

If A.EqualUUID(T1.DataStamp, T2.DataStamp) Then
    ' Tables T1 and T2 contain the same data.
End If
```

Database

Returns

Database

Description

Returns a reference to the database the table belongs to.

Example

```
Dim T As DBTable
Dim D As Database

Set D = T.Database
```

Methods

Identical

Parameters

Table As DBTable, Optional WithReferences As Boolean = False

Returns

Boolean

Description

Identical returns True if this table and Table have identical structures. This means that the names and all other table properties, the number of columns, their names and properties are equal.

IfWithReferences = False, then the references to other columns are not compared.

IfWithReferences = True, then the references to other columns are compared. Column references are equal if the names of the tables and columns they refer to are equal.

Example

```
Dim T1 As DBTable, T2 As DBTable

If T1.Identical(T2, True) Then
    ' Tables inclusive the references are identical
End If
```

GetColumn

Parameters

Column As Variant (Subtype of Column is either Integer or String)

Returns

DBTableColumn

Description

GetColumn returns the column specified by parameter Column.

Column : contains either the sequence number or the name of the column.

Example

```
Dim T As DBTable
Dim Column As DBTableColumn
Dim ColumnName As Variant

ColumnName = "Population"
Set Column = T.GetColumn(ColumnName)
```

AddColumn

Parameters

Name As String, DataType As Byte,
Optional Description As String,
Optional ObjectTypes As Long,
Optional Display As Boolean = True,
Optional MyBeNull As Boolean = True,
Optional Changeable As Boolean = True,
Optional Minimum As Variant = Null,
Optional Maximum As Variant = Null,
Optional Default As Variant = Null,
Optional Fractional As Byte = 6,
Optional ReferencedColumn As DBTableColumn = Nothing,
Optional PendingAction As Byte = ptPendingLeave,
Optional Indexed As Boolean = False
Optional Unique As Boolean = False

Returns

DBTableColumn

Description

AddColumn adds a new column to the table with the specified properties. The new column becomes the last one.

Name: the name of the column.

Type: one of the following values : vbBoolean, vbDate, ptDateOnly, ptTime, vbDouble, vbLong, vbString, vbObject.

Description: when empty string ("") or not specified, Description = Name.

ObjectTypes: has only a meaning if DataType = vbObject.

If the parameter DataType = vbObject and the ObjectTypes = 0 then ObjectTypes will be changed to ptAllDBLinkable. For more information see DBTableColumn property

ObjectTypes.

If DataType = vbObject then Indexed and Unique must be both = FALSE.

If DataType = vbDouble then Unique must be both = FALSE.

For the meaning of the other parameters see the DBTableColumn properties.

Example

```
Dim D As Document
Dim A As Application
Dim T As DBTable
Dim C1 As DBTableColumn, C2 As DBTableColumn, C3 As DBTableColumn

Set A = New Application
A.GetActiveDocument D
Set T = D.CreateTable(ptAttrTable, "Countries", ObjectTypes:=ptPolygon)
Set C1 = T.AddColumn("Name", vbString, Description:="Name of country",_
                    Indexed:=True)
Set C2 = T.AddColumn("Population", vbLong)
Set C3 = T.AddColumn("Area", vbLong, Indexed:=True)
```

DropColumn

Parameters

Column As DBTableColumn

Description

DropColumn deletes the column from the table. All data of this column are lost.

On return Column = Nothing.

Example

```
Dim T As DBTable
Dim C As DBTableColumn

T.DropColumn C ' On return C = Nothing
```

InsertRow

Parameters

Data() As ColumnValue, Optional Id As Long, Optional Row As DBRow

Returns

DBRow

Description

InsertRow inserts a new row into a dictionary table. This method may not be used for an attribute table.

Id: output parameter. Contains on return the assigned row identifier.

Row: output parameter. Contains on return the new row.

Example

```
Dim T As DBTable
Dim Data() As ColumnValue
Dim Id As Long

Redim Data(1 To 3)

' Initialise "Data"

T.InsertRow Data, Id
```

FirstRow

Parameters

Optional SubDoc As SubDocument = Nothing,
Optional ByColumn As Variant = Null,
Optional From As Variant = Null

Returns

DBRow

Description

FirstRow returns the first row of a table. When the table is empty, Nothing is returned.
SubDoc: if specified and not Nothing, then only rows of the specified subdocument will be returned. If SubDoc = Nothing, the rows of all non-sleeping documents will be taken into account.
ByColumn: if specified and not Null, then the rows are considered to be sorted by this column. The specified column must be indexed (see DBTableColumn class property Indexed).
The column is specified either by name or by sequence number. First column = sequence number 0.
From: if specified and not Null, then the first row with value in the specified column >= From will be returned.
A call to FirstRow will normally be followed by a series of calls to NextRow.
Note: The DBTablemethod FirstRow is obsolete. We recommend to use the DBRows methodFirstRow. The new methods allow concurrent and nested access to rows of a table. For more information see DBRows.

Example

```
Dim T As DBTable
Dim Row As DBRow

Set Row = T.FirstRow(ByColumn:=2)
While Not Row Is Nothing
    ' Treat row
    Set Row = T.NextRow
Loop
```

NextRow

Parameters

None

Returns

DBRow

Description

NextRow returns the next row of the table. When there is no next row, Nothing is returned. A sequence of calls to NextRow must have been preceded by a call to FirstRow.
NextRow takes into account all parameters specified when FirstRow was called.
Note: The DBTablemethod NextRow is obsolete. We recommend to use the DBRows methodNextRow. The new methods allow concurrent and nested access to rows of a table. For more information see DBRows.

Example

```
Dim T As DBTable
Dim Row As DBRow

Set Row = T.NextRow
```

LastRow

Parameters

Optional ByColumn As Variant

Returns

DBRow

Description

LastRow returns the last row of a table. When there the table is empty, Nothing is returned.
A call to LastRow will normally be followed by a series of calls to PreviousRow.
ByColumn: see FirstRow description.

Note: The DBTable method LastRow is obsolete. We recommend to use the DBRows method LastRow. The new methods allow concurrent and nested access to rows of a table. For more information see DBRows.

Example

```
Dim T As DBTable
Dim Row As DBRow

Set Row = T.LastRow(ByColumn:="Area")
While Not Row Is Nothing
    ' Treat rows : descending order by "Area"
    Set Row = T.PreviousRow
Loop
```

PreviousRow

Parameters

None

Returns

DBRow

Description

PreviousRow returns the previous row of the table. When there is no previous row, Nothing is returned. A sequence of calls to PreviousRow must have been preceded by a call to LastRow.

PreviousRow takes into account ByColumn parameter specified when LastRow was called.

Note: The DBTable method PreviousRow is obsolete. We recommend to use the DBRows method PreviousRow. The new methods allow concurrent and nested access to rows of a table. For more information see DBRows.

Example

```
Dim T As DBTable
Dim Row As DBRow

Set Row = T.PreviousRow
```

GetRows

Parameters

Optional Subdoc As Subdocument = Nothing

Returns

DBRows

Description

GetRows returns the set of rows of the table. The order of the rows depends on the internal structure of the table. As a consequence, no logic in the ordering may be assumed.

Subdoc: if specified, then only the rows of the subdocument are in the set. If Subdoc = Nothing the rows of all non-sleeping subdocuments are included.

Example

```
Dim T As DBTable
Dim Rows As DBRows
Dim Row As DBRow

Set Rows = T.GetRows()
Set Row = Rows.FirstRow()
Do While Not Row Is Nothing
    ' Treat row
    Set Row = Rows.NextRow()
Loop
```

RowById

Parameters

RowId As Long

Returns

DBRow

Description

RowById returns the row identified by RowId of the dictionary table. This method may not be called for attribute tables. When there is no row with the given RowId, Nothing is returned.

Example

```
Dim T As DBTable
Dim Row As DBRow

Set Row = T.RowById(3)
```

DeleteRow

Parameters

Row As DBRow

Description

DeleteRow deletes the given row of the dictionary table. This method may **not** be called for attribute tables. Instead the CadObject method DeleteData must be used.

Example

```
Dim T As DBTable
Dim Row As DBRow

T.DeleteRow Row
```

FirstSubDocument

Parameters

None

Returns

SubDocument

Description

FirstSubDocument returns the first subdocument the table belongs to.

Note: The DBTablemethod FirstSubDocument is obsolete. We recommend to use the DBSubDocuments methodFirstSubDocument. The new methods allow concurrent and nested access to subdocuments of a table. For more information see DBSubDocuments.

For a table of a Pythagoras database (not linked to a document) this method will return Nothing.

Example

```
Dim T As DBTable
Dim SubDoc As SubDocument

Set SubDoc = T.FirstSubDocument
```

NextSubDocument

Parameters

None

Returns

SubDocument

Description

NextSubDocument returns the next subdocument this table belongs to.

Returns Nothing if the previous returned subdocument was the last one.

Note: The DBTablemethod NextSubDocument is obsolete. We recommend to use the DBSubDocuments methodNextSubDocument. The new methods allow concurrent and nested access to subdocuments of a table. For more information see DBSubDocuments.

For a table of a Pythagoras database (not linked to a document) this method will return Nothing.

Example

```
Dim T As DBTable
Dim SubDoc As SubDocument

Set SubDoc = T.NextSubDocument
```

GetSubDocuments

Parameters

None

Returns

DBSubDocuments

Description

GetSubDocuments returns the set of subdocuments the table belongs to.

For a table of a Pythagoras database (not linked to a document) this method will return Nothing.

Example

```
Dim T As DBTable
Dim SubDocs As DBSubdocuments
Dim SubDoc As SubDocument

Set SubDocs = T.GetSubDocuments
Set SubDoc = Subdocs.FirstSubDocument
```

Clear

Parameters

None

Description

Clear deletes all rows from the table.

Example

```
Dim T As DBTable

T.Clear
```

FastClear

Parameters

None

Description

FastClear does the same as Clear, but also clears the history and therefore this operation can not be undone. This method is faster.

Example

```
Dim T As DBTable

T.FastClear
```

CopyData

Parameters

Source As DBTable

Description

CopyData deletes all rows from the current (destination) table and copies all rows from Source table to the current table. This method may only be applied to dictionary tables of identical structure.

Before copying, the data of both tables are checked on equality (using the data stamps). If data are equal no actual copying is done.

Example

```
Dim T As DBTable
Dim TCopy As DBTable

TCopy.CopyData T
```

Chapter 44 - DBTableColumn

Description

The DBTableColumn object defines one column of a table.

Example

See example DBTable.

Properties

Table
SeqOrder
Name
Description
DataType
ObjectTypes
Display
MayBeNull
Changeable
Minimum
Maximum
Default
Fractional
ReferencedColumn
PendingAction
Indexed
Format

Methods

Move
GetRow
GetRows

Properties

Table

Returns

DBTable

Description

Returns the database table that contains this column.

Example

```
Dim T As DBTable
Dim Column As DBTableColumn

Set T = Column.Table
```

SeqOrder

Returns

Integer

Description

Returns the number of the column. Unlike SQL, the columns are numbered starting from 0.

Example

```
Dim Nr As Integer
Dim Column As DBTableColumn

Nr = Column.SeqOrder
```

Name

Returns

String

Description

Returns or sets the name of the column. The name of a column may not include spaces or control characters. Column names are not case sensitive. Eg names "PYTHAGORAS" and "Pythagoras" are considered equal.

Example

```
Dim Name As String
Dim Column As DBTableColumn

Name = Column.Name
```

Description

Returns

String

Description

Returns or sets the description of the column of the table.

Example

```
Dim Column1 As DBTableColumn, Column2 As DBTableColumn

Column2.Description = Column1.Description
```

DataType

Returns

Byte

Description

Returns or sets the type of the data in the column.

The **DataType** is one of the following values : **vbBoolean**, **vbDate**, **ptDateOnly**, **ptTime**, **vbDouble**, **vbLong**, **vbString**, **vbObject**. The type **vbObject** defines a column that references to a Cad Object in the drawing.

When the type of a column is changed, the data of all rows is converted.

The column properties **Minimum**, **Maximum** and **Default** are also converted to the new type.

When the table is not empty and the conversion of the data of the column or of the properties **Minimum**, **Maximum** and **Default** is not possible for one or more rows, an error will be generated.

Minimum and **Maximum** may be set to **NULL** when the new type is **vbString** or **vbObject**.

When the type is set to **vbObject**, the property **ObjectTypes** becomes **ptAllDBLinkable**. When the type was **vbObject** and is changed to another value, the property **ObjectTypes** becomes 0.

A column of **DataType = vbObject** may only contain references to Cad Objects in the same document. Regarding pending references see also the **Document** method **PurgeDatabase** and the **DBTableColumn** property **PendingAction**.

The property **DataType** of columns that are **Indexed** or with property **MayBeNull = False** can not be set to **vbObject**.

The property **DataType** of columns that are **Unique** can not be set to **vbDouble** or to **vbObject**.

Example

```
Dim Column As DBTableColumn

Column.DataType = vbObject
```

ObjectTypes

Returns

Long

Description

The property **ObjectTypes** sets or returns the type of CAD Objects this column may refer to.

This property has only a meaning if **Type = vbObject**. Else **ObjectTypes must be= 0**.

The value is one of the following constants : **ptAllDBLinkable**, **ptPoint**, **ptLine**, **ptArc**, **ptClothoid**, **ptText**, **ptRichText**, **ptImage**, **ptCurve**, **ptPath**, **ptPolygon**, **ptCircle**, **ptRoad**, **pt3DSolid** or the sum of any of the above values.

Before setting or modifying **ObjectTypes**, the property **Type** must be changed if required. If the new value of **ObjectTypes** conflicts with the actual value of **Type**, an error will be generated.

Example

```
Dim Column As DBTableColumn

Column.DataType = vbObject           ' Column refers to CadObject
Column.ObjectTypes = ptPoint + ptLine ' Only Points and Lines
```

Display

Returns

Boolean

Description

Sets or returns a flag that indicates if this column has to be displayed on the screen in a number of cases. This property is only meaningful for attribute tables.

Example

```
Dim Column As DBTableColumn

Column.Display = True
```

MaybeNull

Returns

Boolean

Description

Sets or returns a flag indicating if the value in the column may be Null. A value = Null means that no value is assigned. NULL is different from all other possible values including an empty string. This property cannot be set to False when DataType = vbObject.

Example

```
Dim Column As DBTableColumn

Column.MaybeNull = False ' This column must have a value.
```

Changeable

Returns

Boolean

Description

Sets or returns a flag indicating if the data (stored in the rows) may be changed.

Example

```
Dim Column As DBTableColumn

Column.Changeable = False ' The data may not be changed.
```

Minimum

Returns

Variant

Description

Sets or returns the minimum value of this column.

If Minimum = NULL, there is no minimum value. This property is always = NULL when DataType = vbString or DataType = vbObject.

Example

```
Dim Column As DBTableColumn

Column.DataType = vbLong
Column.Minimum = -10
Column.Maximum = +10
Column.Default = 0
```

Maximum

Returns

Variant

Description

Sets or returns the maximum value of this column.

If `Maximum = NULL`, there is no maximum value. This property is always = `NULL` when `DataType = vbString` or `Type = vbObject`.

Example

See property `Minimum`

Default

Returns

Variant

Description

Sets or returns the default value for this column.

If `Default = NULL`, there is no default value.

Example

See property `Minimum`

Fractional

Returns

Byte

Description

Sets or returns the number of fractional decimal places for columns of `DataType = vbDouble`.

The property only affects the visual representation of the value of the column in dialog boxes of Pythagoras and sets the number of decimals when exporting data (DBF and Shape).

Example

```
Dim Column As DBTableColumn
```

```
Column.Fractional = 2           ' 123.456787 will be displayed as 123.46
```

ReferencedColumn

Returns

DBTableColumn

Description

`ReferencedColumn` defines the column of a table which is referenced by this column. The column on which this property is applied is called the referencing column. The column to which is referenced is called the referenced column. In the same respect the words referencing table and referenced table are used.

The referenced table must be a dictionary table. References to attribute tables are not allowed.

Type of referencing and referenced columns must be identical. Values in the referencing column are references. A value in the referencing column may be `NULL` which means that there is no reference.

Otherwise they are treated as references to the rows of the referenced table, where values in the referenced column are the same.

The referencing and referenced table must be in the same document. When a document is composed of subdocuments, referencing is allowed between subdocuments. If a reference is made to a dictionary table of another subdocument, then a copy of the referenced table is silently added to the referencing document.

Only column with `Unique=True` can be referenced.

Example

```
Dim Column As DBTableColumn
```

```
Dim RefColumn As DBTableColumn
```

```
Set Column.ReferencedColumn = RefColumn
```

```
' Reading
```

```
If Column.ReferencedColumn Is Nothing Then
```

```
    ' No referenced column
```

```
End If
```

PendingAction

Returns

Byte

Description

PendingAction returns or sets the type of action to be taken when the Document method PurgeDatabase is executed and the referencing column has pending references. PendingAction must have one of the following values : ptPendingLeave, ptPendingNull or PtPendingDelete. For detailed information what action is taken see the Document method PurgeDatabase.

Example

```
Dim Column As DBTableColumn

Column.PendingAction = ptPendingNull
```

Indexed

Returns

DBTable

Description

Sets or returns the indication if this column is indexed. When a column is indexed, a special structure called an index is build for the column. The index is used to find rows by values in the column faster. The DBTable method GetFirstRow with the ByColumn argument can only be applied to an indexed column. When setting Indexed = False then property Unique is automatically set to False. Value of this property cannot be set to False when the column is referenced and cannot be set to True when DataType = vbObjects.

Example

```
Dim Column As DBTableColumn

Column.Indexed = True
```

Unique

Returns

Boolean

Description

Sets or returns the indication if the column values (unless = Null) are unique for all rows. Only columns with the property Unique = True can be referenced from other columns. When setting Unique = True the property Indexed also changes automatically to True. Value of this property cannot be set to False when the column is referenced and cannot be set to True when DataType = vbObject or when Table.TableType = ptAttrTable.

Example

```
Dim Column As DBTableColumn

Column.Unique = True
```

Format

Returns

String

Description

Sets or returns a format string defining how data of type vbDate, ptOnlyDate or ptTime should be formatted during conversion to text. For columns of other types a value of the property is senseless.

Example

```
Dim Column As DBTableColumn

Column.Format = "yyyy-MM-dd"
```


Methods

Move

Parameters

N As Integer

Description

Move changes the sequence number (SeqOrder property) of the column. The sequence number is incremented by N. If the result becomes less than 0 or greater or equal to the number of columns, the sequence number will get the minimum (0) or maximal (number of columns - 1) value. It is evident that the sequence number of other columns is adapted accordingly.

Example

```
Dim Column As DBTableColumn

Column.Move -1          ' Decrement sequence number by 1.
```

GetRow

Parameters

Key As Variant

Returns

DBRow

Description

The method can only be applied to a column with property Unique = True. The method returns the row that contains the requested value (Key). The no row in the table with this value Nothing is returned.

Example

```
Dim Column As DBTableColumn
Dim Row As DBRow

Set Row = Column.GetRow("Belgium")
If not Row is Nothing Then
    ' Do something with the row (as long as you don't split up in 2 rows)
End If
```

GetRows

Parameters

None

Returns

DBRows

Description

The method returns a set of all rows of the table ordered by the values in the column. The method can only be applied to a column with Indexed = True. Rows with Null values are not included in the set.

Example

```
Dim Column As DBTableColumn
Dim Rows As DBRows
Dim Row As DBRow

Set Rows = Column.GetRows()
Set Row = Rows.FirstRow
Do While Not Row Is Nothing
    ' Treat row
    Set Row = Rows.NextRow
Loop
```

Chapter 45 - DBQuery

Description

An object of this class describes a DB query. DB queries can be one of the two following types: DB view and temporary queries.

DB views have different names and are stable objects, i.e. are retained in the database until either a view is explicitly dropped or the whole database is dropped.

Temporary queries can be created only via VBA code, they have no name. They are dropped in the same cases as DB views and additionally when the last reference to them is lost, in particular when a VBA library is unloaded.

Example

```
Dim D As Database
Dim Q As DBQuery

Set Q = _
    D.CreateSimpleQuery ( _
        A.RowSetDefinition (ptTable, Name="Employees"), Employees2)
```

Properties

Database
QueryType
MainBaseDefinition
AuxBaseDefiniton
Name
Description
Where
Preferences
Match
Validity
ErrorIndex
SyntaxError
ErrorStart
ErrorEnd
NrOfWhereConditions
NrOfColumns
CADObjectLinked
ObjectTypes
Overlay
Sheet

Methods

GetWhere
AddWhere
AlterWhere
MoveWhere
DropWhere
AddColumn
GetColumn
DropColumn
GetRows
RowById

Properties

Database

Returns

Database

Description

Returns a reference to a database the query belongs to.

Example

```
Dim Q As DBQuery
Dim D As Database

Set D = Q.Database
```

QueryType

Returns
PtQueryType

Description
Returns a type of the query.

Example

```
Dim Q As DBQuery

If Q.QueryType = ptQueryExtension Then
    ' do something
End If
```

MainBaseDefinition, AuxBaseDefinition

Returns
RowSetDefinition

Description
Returns or sets definition of the main or of the auxiliary base rowset correspondingly.
AuxBaseDefinition property may be applied only to an aggregation query.

Example

```
Dim Q As DBQuery, Def As RowSetDefinition

' Changing name of main base rowset (a table or a query)

Def = Q.MainBaseDefinition
Def.Name "Provinces"
Q.MainBaseDefinition = Def
```

Name

Returns
String

Description
Returns or sets a name of the view. If the query is not a view then this property cannot be written and returns an empty string when read. Reading the property could be used to define if this query is a DB view.

Example

```
Dim V As DBQuery

V.Name = "View5"
```

Description

Returns
String

Description
Returns or sets the description of the query.

Example

```
Dim V As DBQuery

V.Description = "City statistics by countries"
```

Where

Returns
String

Description
Returns or sets a textual where condition. This condition is used additionally to other where conditions added by AddWhere method. An empty value means that the query contains no textual where condition.

Example

```
Dim Q As DBQuery

Q.Where = "Name = ""PS360"""
```

Preferences

Returns

Units

Description

Sets or returns preferences used in this query. Preferences are used mainly when calculating expressions. Returned value subproperty assignment has no effect (see example below).

Example

```
' We want to change item separator to semicolon
' (as used in Europe)
Dim Q As DBQuery
Dim Pref As Units

Pref = Q.Preferences
Pref.ItemSeparator = Asc(";")
Q.Preferences = Pref

' Note that the following code has no effect :

Q.Preferences.ItemSeparator = Asc(";")
```

Match

Returns

MatchCondition

Description

Sets or returns a match condition for the aggregation query. This property cannot be applied either to a simple query.

Example

```
Dim Q As DBQuery
Dim M As MatchCondition

M.Kind = ptDistance
M.Distance = 100
Q.Match = M
```

Validity

Returns

PtQueryValidity

Description

Returns information whether the query is valid (Validity=ptQueryValid) and if not what is an error. A query is valid when all its definitions are valid. This requirement means also that definitions of where conditions, dimensions and columns are valid. Data can be obtained only from valid queries. Additional information on the error is returned by properties `ErrorIndex`, `SyntaxError`, `ErrorStart` and `ErrorEnd`.

Example

```
Dim Q As DBQuery

' Check if the Q is valid and print message if not.
' An example contains some error codes that cannot occur in
' 13th version.
' They will be possible in 14th version of Pythagoras.

Case ptQueryValid ' Do nothing
Case ptQueryCircularQueries
    MsgBox "Circular reference of queries"
Case ptQueryBadAliasName
    MsgBox CStr(Q.ErrorIndex) &
        "th dimension has illegal name"
Case ptQueryAliasNameRepeated
    MsgBox "In " & CStr(Q.ErrorIndex) & _
        "th dimension alias name is repeated"
Case ptQueryBadViewName
    MsgBox "View name is invalid"
```

```

Case ptQueryNoReferencedQuery
    Select Case Q.QueryType
        Case ptQueryExtension
            MsgBox _
                "Reference to a query that " & _
                "does not exist in " & _
                CStr(Q.ErrorIndex) & "th dimension"
        Case ptQueryAggregation
            MsgBox _
                "Reference to a query that does not exist"
    End Select
Case ptQueryNoReferencedTable
    Select Case Q.QueryType
        Case ptQueryExtension
            MsgBox _
                "Reference to a table that " & _
                "does not exist in " & _
                CStr(Q.ErrorIndex) & "th dimension"
        Case ptQueryAggregation
            MsgBox _
                "Reference to a table that does not exist"
    End Select
Case ptQueryNoReferencedSheet
    MsgBox "Reference to a sheet that does not exist"
Case ptQueryBadLeftMatch
    MsgBox "Syntax error " & CStr(Q.SyntaxError) & _
        " in left match expression of " & _
        CStr(Q.ErrorIndex) & "th dimension, " & _ "positions " & _
        CStr(Q.ErrorStart+1) & "-" & _
        CStr(Q.ErrorEnd)
Case ptQueryNoRightMatch
    MsgBox "Match column not found in " & _
        CStr(Q.ErrorIndex) & "th dimension"
Case ptQueryNotUniqueMatch
    MsgBox "Match column is not unique in " & _
        CStr(Q.ErrorIndex) & "th dimension"
Case ptQueryNoReferencedAlias
    MsgBox CStr(Q.ErrorIndex) & "th " & _
        "where condition references not existing base"
Case ptQueryNoReferencedColumn
    MsgBox CStr(Q.ErrorIndex) & "th " & _
        "where condition references not existing column"
Case ptQueryBadCondition
    MsgBox CStr(Q.ErrorIndex) & "th " & _
        "where condition contains inconsistency"
Case ptQueryBadWhereExpression
    MsgBox "Syntax error " & CStr(Q.SyntaxError) & _
        " in where expression, positions " & _
        CStr(Q.ErrorStart) & "-" & CStr(Q.ErrorEnd)
Case ptQueryBadColumnExpression
    MsgBox "Syntax error " & CStr(Q.SyntaxError) & _
        " in expression for " & CStr(Q.ErrorIndex) & _
        "th column, positions " & _
        CStr(Q.ErrorStart) & "-" & CStr(Q.ErrorEnd)
Case ptQueryNoMirroredColumn
    MsgBox CStr(Q.ErrorIndex) & "th column definition " & _
        "references not existing column"
Case ptQueryBadAggregationMatch
    MsgBox "Incorrect match expression for " & _
        "an aggregation query"
Case ptQueryBadSpatialAggregation
    MsgBox "An attempt to build a spatial aggregation " & _
        "when one of base rowsets is not an attribute one"
Case ptQueryBadAggregateType
    MsgBox "In " & CStr(Q.ErrorIndex) & _
        "th column aggregate function is applied to " & _
        "not proper data type"
Case ptQueryBadColumnName
    MsgBox CStr(Q.ErrorIndex) & "th column has invalid name"
Case ptQueryColumnNameRepeated
    MsgBox "In " & CStr(Q.ErrorIndex) & _

```

```

        "th column name is repeated"
Case ptQueryNoReferencedSheet
    MsgBox "Reference to a sheet that does not exist"
Case ptQueryReferencesInvalidQuery
    Select Case Q.QueryType
        Case ptQueryExtension
            MsgBox _
                "Reference to an invalid query in " & _
                CStr(Q.ErrorIndex) & "th dimension"
        Case ptQueryAggregation
            MsgBox "Reference to an invalid query"
    End Select
Case ptQueryNoReferencedQuery
    Select Case Q.QueryType
        Case ptQueryExtension
            MsgBox _
                "Reference to a temporary query that " & _
                "has been deleted in " & _
                CStr(Q.ErrorIndex) & "th dimension"
        Case ptQueryAggregation
            MsgBox _
                "Reference to a temporary query that " & _
                "has been deleted"
    End Select
Case ptQueryViewReferencesTempQuery
    Select Case Q.QueryType
        Case ptQueryExtension
            MsgBox _
                "View base is a temporary query in " & _
                CStr(Q.ErrorIndex) & "th dimension"
        Case ptQueryAggregation
            MsgBox "View base is a temporary query"
    End Select
End Select

```

ErrorIndex

Returns

Integer

Description

Returns additional information about an error in query definition. Returned value has one of the following meanings:

- When Validity is `ptQueryNoReferencedAlias`, `ptQueryNoReferencedColumn` or `ptQueryBadCondition`—sequential number of where condition which is erratic.
- When Validity is `ptQueryBadColumnExpression`, `ptQueryNoMirroredColumn`, `ptQueryBadAggregateType` or `ptQueryBadColumnName`—sequential number of column with error.
- Otherwise no meaningful information is in the field.

Example

See property Validity example.

SyntaxError

Returns

PtSyntaxError

Description

Returns additional information about an error in query definition. Returned value is an error code issued by the expression compiler during compilation of a textual expression. Returned value is meaningful only when Validity is one of `ptQueryBadLeftMatch`, `ptQueryBadWhereExpression` or `ptQueryBadColumnExpression`.

Example

See property Validity example.

ErrorStart, ErrorEnd

Returns

Long

Description

Returns additional information about an error in query definition. Returned values are starting and final positions (numerated from 0) of a token in an expression string where the compiler found an error described by `SyntaxError` property. Returned values are meaningful only when `Validity` is one of `ptQueryBadLeftMatch`, `ptBadQueryWhereExpression` or `ptBadQueryColumnExpression`.

Example

See property `Validity` example.

NrOfWhereConditions

Returns

Integer

Description

Returns number of where conditions defined for this query. This number does not include a textual where condition. Where conditions can be accessed using methods `GetWhere`, `AddWhere`, `AlterWhere` and `DropWhere`.

Example

```
Q As DBQuery
Dim NrWhere As Integer

NrWhere = Q.NrOfWhereConditions
```

NrOfColumns

Returns

Integer

Description

Returns number of columns of the query. Column definitions can be accessed using methods `GetColumnDefinition`, `AddColumn`, `AlterColumn`, `DropColumn` and `GetColumn`.

Example

```
Q As DBQuery
Dim NrDim As Integer

NrDim = Q.NrOfDimensions
```

CadObjectLinked

Returns

Boolean

Description

Returns `True` when rows of this query are linked to CAD objects. The property may be applied only to a valid query. Returned value is defined by the main base rowset definition.

Example

```
Dim Q As DBQuery

If Q.CadObjectLinked Then
    ' do something
End If
```

ObjectTypes

Returns

Long

Description

Returns the CAD object types that may be linked to this query. This property may be applied only to a valid query. Returned value is actually object types allowed for the main base rowset. 0 is returned when the query is not CAD object linked.

Example

```
Dim Q As DBQuery

If Q.Valid And (Q.ObjectTypes And ptPoint) <> 0 Then
    ' If points can be linked to the view then do something
End If
```

Overlay

Returns

Overlay

Description

Returns an overlay CAD objects linked to the query belong to. The property may be applied only to a valid query. Being applied to a not CAD linked query it returns Nothing.

Example

```
Dim Q As DBQuery, Ov As Overlay

If Q.Overlay Is Ov Then
    ' If Q contains objects of Ov then do something
End If
```

Sheet

Returns

Sheet

Description

Returns a sheet CAD objects linked to the query belong to. The property may be applied only to a valid query. Being applied to a not CAD object linked query it returns Nothing. Also Nothing is returned when the query is applied to the local overlay.

Example

```
Dim Q As DBQuery, Sh As Sheet

If Q.Sheet Is Sh Then
    ' If Q contains objects of Sh then do something
End If
```

Methods

GetWhere

Parameters

Index As Integer

Returns

WhereCondition

Description

GetWhere returns the where condition specified by Index. Conditions are indexed starting from 0. Condition with Compare=ptCompareNil is returned if there is no such where condition.

Example

```
Dim I As Integer, N As Integer
Dim Q As DBQuery
Dim Conditions() As WhereCondition
N = Q.NrOfWhere
If N > 0 Then
    ReDim Conditions(N-1)
    For I = 0 To N-1
        Conditions(I) = Q.GetWhere(I)
    Next
End If
```

AddWhere

Parameters

Where As WhereCondition

Returns

Integer

Description

AddWhere adds new where condition, which becomes the last one. The index of the added condition is returned.

Where.ValueType is meaningful only in such combinations:

- Where.Value is of type vbDate and Where.ValueType is equal either to ptDateOnly or ptTime
- Where.Value is of one of integer types and Where.ValueType=ptCadObjectType

In these cases Value of the new where condition will have type defined by Where.ValueType.

Otherwise Where.ValueType is ignored.

Example

```
Dim A As Application
Dim N As Integer
Dim Q As DBQuery

N = _
    Q.AddWhere ( _
        A.WhereCondition (ptCompareGreaterEqual, "Population", _
            Value:=1000000) _
    )
```

AlterWhere

Parameters

Index As Integer,
Where As WhereCondition

Returns

None

Description

AlterWhere changes the where condition specified by Index. DataType of an argument must correspond to type of a query. An error is raised when there is no such where condition.

Where.ValueType is treated in the same way as by AddWhere method.

Example

```
Dim N As Integer
Dim Q As DBQuery
Dim Condition As WhereCondition

Condition = Q.GetWhere(2)
Condition.Value = 100000
Q.AlterWhere 2, Condition
```

MoveWhere

Parameters

Index As Integer, NewIndex As Integer

Returns

None

Description

MoveWhere changes order of where conditions. Where condition with index = Index is moved to have an index = NewIndex. If NewIndex is too small or too big, then the condition becomes correspondingly 0th or the last. Relative order of other where conditions does not change.

Example

```
Dim Q As DBQuery

Q.MoveDimension 1, 0 ' Exchange 0th and 1st where conditions
```

DropWhere

Parameters

Index As Integer

Returns

None

Description

DropWhere deletes the where condition specified by **Index**. Next conditions are moved left. An error is raised when there is no such where condition.

Example

```
Dim Q As DBQuery

While Q.NrOfWhere > 0
    Q.DropWhere 0
WEnd
```

AddColumn

Parameters

Column As **ColumnDefinition**

Returns

Integer

Description

AddColumn adds new column to the query, the new column becomes the last one. The index of the added column is returned.

Example

```
Dim A As Application
Dim N As Integer
Dim Q As DBQuery

N = _
    Q.AddColumn ( _
        A.ColumnDefinition ("Density", IsComputed:=True, _
            Expression:="Population/Area() ") _
    )
```

GetColumn

Parameters

Index As Variant (Integer or String)

Returns

DBQueryColumn

Description

GetColumn returns a column found by an index or a name. Nothing is returned when there is no such column.

Example

```
Dim I As Integer, N As Integer
Dim Q As DBQuery
Dim Columns() As DBQueryColumn
N = Q.NrOfColumns
If Q.Valid And N > 0 Then
    ReDim Columns(N-1)
    For I = 0 To N-1
        Columns(I) = Q.GetColumn(I)
    Next
End If
```

DropColumn

Parameters

Index As Variant (Integer or String)

Returns

None

Description

DropColumn deletes column found by an index or by a name. Next columns are moved left. An error is raised when there is no column with such index. Nothing is done when there is no column with such name.

Example

```
Dim Q As DBQuery

While Q.NrOfColumns > 0
    Q.DropColumn 0
WEnd
```

GetRows

Parameter

Optional SubDoc As SubDocument = Nothing

Returns

DBRows

Description

GetRows returns a set of rows of the query ordered by some internally defined order.

SubDoc: if specified and not Nothing, then only rows of the specified subdocument are included. If

SubDoc = Nothing, the rows of all non-sleeping documents are included.

Example

```
Dim Q As Query
Dim Rows As DBRows
Dim Row As DBRow
Set Rows = Q.GetRows()
Set Row = Rows.FirstRow()
While Not Row Is Nothing
    ' Treat row
    Set Row = Rows.NextRow()
WEnd
```

RowById

Parameters

RowId As Long

Returns

DBRow

Description

RowById returns a row identified by RowId of the query. This method may be called only for valid, not CAD object linked queries (see corresponding properties). When there is no row with the given RowId, Nothing is returned.

Example

```
Dim Q As DBQuery
Dim Row As DBRow

Set Row = Q.RowById(3)
```

Chapter 46 - DBQueryColumn

Description

An object of this class represents a column of a query.

Unlike other database objects (classes DBTable, DBTableColumn, DBQuery, DBRow) query columns have no unique identifier, therefore a query column is internally identified by query unique identifier and sequential number of the column. Therefore, if query columns have been inserted, moved or deleted, it can happen that an object of this class will point to another column of the same query having the same sequential number.

Example

```
Dim Q As DBQuery, C As DBQueryColumn

Set C = Q.GetColumn (0)
```

Properties

Query
SeqOrder
Definition
Name
Description
DataType
Format
Fractional

Methods

Move

Properties

Properties Name, Description, Format and Fractional are similar as the corresponding properties of DBTableColumn class; see their descriptions.

Query

Returns

DBQuery

Description

Returns a query that contains this column.

Example

```
Dim Column As DBQueryColumn

If Column.Query.Valid Then
    ' do something
End If
```

SeqOrder

Returns

Integer

Description

Returns a sequential number of the column in a query. Columns are numbered starting from 0.

Example

```
Dim Nr As Integer
Dim Column As DBQueryColumn

Nr = Column.SeqNumber
```

Definition

Returns

ColumnDefinition

Description

Definition returns or sets definition of the column.

Example

```
Dim Column As DBQueryColumn
Dim D As ColumnDefinition

D = Column.Definition
D.Aggregation = ptFunctionMax
Column.Definition = D
' Note that the following example does not work :
Column.Definition.Aggregation = ptFunctionMax
```

DataType

Returns

Byte

Description

Returns data type of the column.

If the query is valid then data type of the column is returned which may be: vbBoolean, vbDate, ptDateOnly, ptTime, vbDouble, vbLong, vbString, vbObject, ptType. Otherwise vbError is returned.

Example

```
Dim Column As DBQueryColumn

If Column.DataType = vbString Then
    ' do something
End If
```

Methods

Move

Parameters

N As Integer

Returns

None

Description

Move changes the sequence number (SeqOrder property) of the column. The sequence number is incremented by N. If the result becomes less than 0 or greater or equal to the number of columns, the sequence number will get the minimum (0) or maximal (number of columns - 1) value.

It is evident that the sequence number of other columns is adapted accordingly.

Example

```
Dim Column As DBQueryColumn

Column.Move 32000 ' Move column "Weight" to the end
```

Chapter 47 - DBRow

Description

The DBRow object refers to one row of a rowset. A row of a table is set of data that may be read, written, changed or deleted. A row of a query is calculated when it is read, therefore it cannot be written, changed or deleted.

Example

See example DBTable

Properties

Table
Query
LinkedObject
Id
Overlay
Sheet

Methods

GetAll
GetValue
SetValue
GetStringValue
Update
Identical

Properties

Table

Returns

DBTable

Description

Returns the table this row belongs to. If this row belongs to a query, Nothing is returned.

Example

```
Dim T As DBTable
Dim R As DBRow

Set T = R.Table
```

Query

Returns

DBQuery

Description

Returns a query the row belongs to. If this row belongs to a table, Nothing is returned.

Example

```
Dim Row As DBRow

If Not Row.Query Is Nothing And Row.Query.Name = "PolesA" Then
    ' Do something
```

LinkedObject

Returns

CadObject

Description

Returns the CadObject that is linked to the row. This property is only applicable to rows of rowsets linked to CAD objects, i.e. to when the rowset is either an attribute table or a set of objects or a query with CADObjectLinked=True.

Example

```
Dim Obj As CadObject
Dim R As DBRow
```

```
Set Obj = R.LinkedObject
```

Id

Returns

Long

Description

Returns the unique row identifier. This property is only applicable to rows of a rowsets not linked to CAD objects. Each row in a dictionary table has a unique identifier that is assigned when the row is created. The unique identifier of a row never changes. Rows of not CAD linked objects queries are assumed to have the same identifiers as rows of the main base rowset they are built from.

Example

```
Dim R As DBRow
Dim RowId As Long

RowId = R.Id
```

Overlay

Returns

Overlay

Description

Returns an overlay the row belongs to.

For a row of a dictionary table returned value is `Nothing`. For a row of an attribute table the local overlay of the document the table belongs to is returned.

For a row of a query returned value is the same as for `DBQuery.Overlay` property.

For Pythagoras databases returned value is `Nothing` always.

Example

```
Dim Row As DBRow, Ov As Overlay

If Row.Overlay Is Ov Then
    ' If row belongs to Ov then do something
End If
```

Sheet

Returns

Sheet

Description

Returns a sheet the row belongs to.

For a row of a table `Nothing` is returned.

For a row of a query returned value is the same as for `DBQuery.Sheet` property.

For Pythagoras databases returned value is `Nothing` always.

Example

```
Dim Row As DBRow, Sh As Sheet

If Row.Sheet Is Sh Then
    ' If row belongs to Sh then do something
End If
```

Methods

GetAll

Parameters

`Data()` As ColumnValue, Optional `ReturnNames` As Boolean = False

Description

This method returns all data from a row. The data is stored in an array of `ColumnValues`. When the optional parameter `ReturnNames = True`, then the `Column`-fields will contain the names of the columns. Else the field contains the sequence number of the column.

Example

```
Dim Data() As ColumnValue
```

```
Dim R As DBRow
```

```
R.GetAll Data          ' Column fields of Data contain sequence number.
```

```
R.GetAll Data, True    ' Column fields of Data contain the name of the column.
```

GetValue

Parameters

Column As Variant

Returns

Value As Variant

Description

The method `GetValue` returns the value of the specified column.

The parameter `Column` is either the sequence number of the column (0 .. n-1) or the name of the column.

Example

```
Dim R As DBRow
```

```
Dim Population As Long
```

```
Population = R.GetValue("Population")
```

SetValue

Parameters

Column As Variant, Value As Variant

Description

The method `SetValue` changes the value of the specified column.

The parameter `Column` is either the sequence number of the column (0 .. n-1) or the name of the column.

This method can be applied only to a table row.

Example

```
Dim R As DBRow
```

```
R.SetValue 1, 11450000
```

GetStringValue

Parameters

Column As Variant

Returns

String

Description

This method differs from `GetValue` method only in that a value is converted to string. Conversion from date/time types is done using `Format` property of the column. Conversion from floating point type is done with number of decimal digits defined by `Fractional` property of the column.

Example

```
Dim Row As DBRow
```

```
MsgBox "Deadline is " & Row.GetStringValue("DeadLine")
```

Update

Parameters

Data() As ColumnValue

Description

The method `Update` changes the values of the specified columns.

This method can be applied only to a table row.

Example

```
Dim R As DBRow
```

```
Dim Data() As ColumnValue
```

```
R.Update Data
```


Identical

Parameters

Row As DBRow

Description

Identical compares two rows. The method returns true if all columns contain the same value.

This method may only be applied on rows of tables that have an identical structure (see DBTable method Identical).

Example

```
Dim R1 As DBRow, R2 As DBRow

If R1.Identical(R2) Then
    ' Content of R1 and R2 are identical
End If
```

Chapter 48 - DBRows

Description

The DBRows object refers to a set of rows of a rowset. The set of rows of a table may be ordered by an indexed column.

The DBRows object is created by method GetRows of one of classes: DBTable, DBTableColumn or DBQuery.

Example

See example DBTableGetRows.

Properties

Table
Query
Subdocument
ByColumn

Methods

FirstRow
NextRow
LastRow
PreviousRow

Properties

Table

Returns

DBTable

Description

Returns the table these rows belongs to. If these rows belong to a query then Nothing is returned.

Example

```
Dim T As DBTable
Dim Rows As DBRows

Set T = Rows.Table
```

Query

Returns

DBQuery

Description

Returns the query these rows belong to. If these rows belong to a table then Nothing is returned.

Example

```
Dim Q As Query
Dim Rows As DBRows

Set Q = Rows.Query
```

Subdocument

Returns

Subdocument

Description

Returns the subdocument specified when the object was created or Nothing.

Example

```
Dim SubDoc As SubDocument
Dim Rows As DBRows

Set SubDoc = Rows.Subdocument
```

ByColumn

Returns

DBTableColumn

Description

This property returns the table column used to create the DBRows object or Nothing.

Example

```
Dim Column As DBTableColumn
Dim Rows As DBRows

Set Column = Rows.ByColumn
```

Methods

FirstRow

Parameters

Optional From As Variant = Null

Returns

DBRow

Description

This method returns the first row, or the requested row (if parameter From <> Null), of the set.

The order of the rows in the set has been defined during creation of the set.

From: if specified and not Null, then the first row in the set with value in the specified column (see DBTableColumn method GetRows) >= From will be returned.

Example

```
Dim Column As DBTableColumn
Dim Rows As DBRows
Dim Row As DBRow

Set Rows = Column.GetRows()

' return all rows starting from 100
Set Row = Rows.FirstRow(100)
Do While Not Row Is Nothing
    ' Treat row
    Set Row = Rows.NextRow()
Loop
```

NextRow

Parameters

None

Returns

DBRow

Description

This method returns the next row of the set.

Example

```
Dim Column As DBTableColumn
Dim Rows As DBRows
Dim Row As DBRow

Set Rows = Column.GetRows()
Set Row = Rows.FirstRow("P") ' return row starting from "P"
Do While Not Row Is Nothing
    ' Treat row
    Set Row = Rows.NextRow()
Loop
```

LastRow

Parameters

None

Returns

DBRow

Description

This method returns the last row of the set. When the set is empty, Nothing is returned.

A call to LastRow will normally be followed by a series of calls to PreviousRow. The method can not be applied to sets created by DBTableColumn method GetRows with as SubDoc argument <>Null.

Example

```
Dim T As DBTable
Dim Rows As DBRows
Dim Row As DBRow

Set Rows = T.GetRows()
Set Row = Rows.LastRow
Do While Not Row Is Nothing
    ' Treat row
    Set Row = Rows.PreviousRow()
Loop
```

PreviousRow

Parameters

None

Returns

DBRow

Description

This method returns the previous row of the set. When there is no previous row, Nothing is returned.

Example

```
Dim Column As DBTableColumn
Dim Rows As DBRows
Dim Row As DBRow

Set Rows = Column.GetRows()
Set Row = Rows.LastRow
Do While Not Row Is Nothing
    ' Treat row
    Set Row = Rows.PreviousRow()
Loop
```

Chapter 49 - DBSubDocuments

Description

The DBSubDocuments object is a set of all subdocuments that make up a database table.

Example

```
Dim A As Application
Dim SubDocs As DBSubDocuments
Dim SubDoc As SubDocument
Dim Doc As Document
Dim Table As DBTable
Dim i As Long

Set A = New Application
A.GetActiveDocument Doc
Set Table = Doc.TableByName("Pales")
Set SubDocs = Table.GetSubDocuments()
Set SubDoc = SubDocs.FirstSubDocument
For i = 1 To SubDocs.NrOfSubDocuments - 1
    Set SubDoc = SubDocs.NextSubDocument
Next i
```

Properties

Table
NrOfSubDocuments

Methods

FirstSubDocument
NextSubDocument

Properties

Table

Returns

DBTable

Description

Returns the table the DBSubDocument object belongs to.

Example

```
Dim T As DBTable
Dim SubDocs As DBSubDocuments

Set T = SubDocs.Table
```

NrOfSubDocuments

Returns

Long

Description

This property returns the number of subdocuments in the set.

Example

```
Dim SubDocs As DBSubDocuments

Nr = SubDocs.NrOfSubDocuments
```

Methods

FirstSubDocument

Parameters

None

Returns

SubDocument

Description

This method returns first SubDocument of the set.

Example

See example at introduction of the class DBSubDocuments

FirstSubDocument

Parameters

None

Returns

SubDocument

Description

This method returns the next SubDocument of the set. This method must have been preceded by FirstSubDocument.

Example

See example at introduction of the class DBSubDocuments

Chapter 50 - Layer

Description

The `Layer` object corresponds with a Layer in a Pythagoras drawing. See also the object type `Layers`.

Example

```
Dim A As Application
Dim Doc As Document
Dim Ls As Layers
Dim L As Layer
Dim i As Integer
Dim LayerNames() As String

Set A = New Application
A.GetActiveDocument Doc
If NOT Doc is nothing Then
    Doc.GetLayers Ls
    ReDim LayerNames(1 To Ls.NrOfLayers)
    Ls.GetFirst L
    For i = 1 To Ls.NrOfLayers
        LayerNames(i) = L.Name
        Ls.GetNext L
    Next i
End if
```

Properties

Name
Protected
Visible
Wallpaper
MinimumScale
MaximumScale
CanBeDeleted

Methods

SetMinimumAndMaximumScale

Properties

Name

Returns

String

Description

Returns or sets the name of the layer.

Example

```
Dim L As Layer

MsgBox "The name of this Layer is " & L.Name
```

Protected

Value

Boolean

Description

Returns or sets the protection flag of the layer.

Example

```
Dim L1 As Layer, L2 As Layer

L1.Protected = L2.Protected
```

Visible

Value

Boolean

Description

Returns or sets the visible flag of the layer. The active layer can not be set invisible.

Example

```
Dim L1 As Layer, L2 As Layer

L1.Visible = L2.Visible
```

Wallpaper

Value

Boolean

Description

Returns or sets the wallpaper flag of the layer (formerly SnappingDisabled)

Example

```
Dim L1 As Layer, L2 As Layer

L1.Wallpaper = L2.Wallpaper
```

MinimumScale

Value

Double

Description

Returns the scale that is the lower boundary at which the layer should be displayed.

Example

```
Dim L As Layer
Dim Minimum As Double

Minimum = L.MinimumScale
```

MaximumScale

Value

Double

Description

Returns the scale that is the upper boundary for displaying the layer.

Example

```
Dim L As Layer
Dim Maximum As Double

Maximum = L.MaximumScale
```

CanBeDeleted

Value

Boolean

Description

Returns True if the layer can be deleted. A layer can only be deleted if there are no objects in it.

Example

```
Dim L As Layer
Dim Ls As Layers

If L.CanBeDeleted Then
    Ls.DeleteLayer L
End If
```

Methods

SetMinimumAndMaximumScale

Parameters

MinimumScale As Double, MaximumScale As Double

Description

Sets the minimum scale and maximum scale at which the layers should be displayed.

MinimumScale must be less than MaximumScale.

MinimumScale = ptMinDouble if the layer has to be displayed at any scale up to MaximumScale.

MaximumScale = ptMaxDouble if the layer has to be displayed starting from MinimumScale.

Example

```
Dim L As Layer
```

```
L.SetMinimumAndMaximumScale 1/1000.0, ptMaxDouble
```

Chapter 51 - Layers

Description

An instance of an object `Layers` refers to all layers in a Pythagoras drawing. This object type allows obtaining all layers of a document, but also to create and to delete layers.

Pythagoras uses internally a layer called *System Layer*. Objects, currently only points, lines and clothoids, that are in the System Layer are not visible to the user. In Pythagoras objects in the System Layer are automatically created in some cases, e.g. when creating a polygon. Objects in the System Layer will automatically be deleted by Pythagoras if possible.

The System Layer does not appear in dialog boxes.

An instance of an object `Layers` does not include the System Layer! As a consequence an object `Layer` will never refer to the System Layer.

Be careful putting objects in the System Layer. When there is no reference to those objects, Pythagoras may delete them.

Example

See `Layer`

Properties

`NrOfLayers`
`Protected`
`Visible`
`Wallpaper`

Methods

`GetFirst`
`GetNext`
`GetActiveLayer`
`SetActiveLayer`
`GetLayer`
`NewLayer`
`DeleteLayer`
`DeleteEmptyLayers`

Properties

NrOfLayers

Returns

`Long`

Description

Returns the number of layers in the set.

Example

```
Dim Ls As Layers
```

```
MsgBox "Number of layers in this document = " & Ls.NrOfLayers
```

Protected

Value

`Boolean`

Description

Sets or resets the protected flag for all layers in the group.

Example

```
Dim LG As LayerGroup
```

```
LG.Protected = True
```

Visible

Value

`Boolean`

Description

Sets or resets the visible flag for all layers in the group.

Example

```
Dim LG As LayerGroup

LG.Visible = True
```

Wallpaper

Value

Boolean

Description

Sets or resets the wallpaper flag for all layers in the group.

Example

```
Dim LG As LayerGroup

LG.Wallpaper = True
```

Methods

GetFirst

Parameters

L As Layer

Description

Returns the first layer of the drawing.

Example

```
Dim Ls As Layers
Dim L As Layer

Ls.GetFirst L
MsgBox "First Layer = " & L.Name
```

For a more elaborate example see the introduction on the object Layer.

GetNext

Parameters

L As Layer

Description

Returns the next layer in the set.

Example

```
Dim Ls As Layers
Dim L1 As Layer, L2 As Layer

Ls.GetFirst L1
MsgBox "First Layer = " & L1.Name
Ls.GetNext L2
MsgBox "Second Layer = " & L2.Name
```

For a more elaborate example see the introduction on the object Layer.

GetActiveLayer

Parameters

L As Layer, Optional Exists As Boolean

Description

Returns the active layer of the document if it exists. Whether or not the active layer exists, is returned in the optional parameter Exists.

Example

```
Dim Ls As Layers
Dim L As Layer
Dim Exists As Boolean
```

```

Ls.GetActiveLayer L, Exists
If Exists Then
    MsgBox "Active Layer = " & L.Name
Else
    MsgBox "The active layer doesn't exist."
End If

```

SetActiveLayer

Parameters

L As Layer

Description

Makes L the active layer. This operation has no effect if the document containing the layer L is not the top window.

Example

```

Dim Ls As Layers
Dim L As Layer

Ls.SetActiveLayer L

```

GetLayer

Parameters

LayerName As Name, L As Layer

Description

Returns the layer with the given name. If the layer with the given name is not found, then on return L = Nothing.

If the LayerName = "~SYSTEM~" (name of the System Layer), then on return L = Nothing.

Example

```

Dim Ls As Layers
Dim L As Layer

Ls.GetLayer "Buildings", L
If Not (L Is Nothing) Then
    L.Visible = True
End If

```

NewLayer

Parameters

LayerName As Name, L As Layer

Description

Creates a new layer with the given name. If the layer with the given name already exists, then the parameter L will refer to the existing layer. So NewLayer may be used in stead of GetLayer.

If the LayerName = "~SYSTEM~" (name of the System Layer), then on return L = Nothing.

Example

```

Dim Ls As Layers
Dim L As Layer

Ls.NewLayer "Buildings", L
L.Protected = True

```

DeleteLayer

Parameters

L As Layer

Description

Deletes the layer from the document if the layer contains no objects and the layer is not the active layer.

WARNING: the undo-system of the document is cleared.

Example

```

Dim Ls As Layers
Dim L As Layer

```

```
Ls.GetLayer "Buildings", L
If Not (L Is Nothing) Then
    Ls.DeleteLayer L
End If
```

DeleteEmptyLayers

Parameters

None

Description

Deletes all empty layers from the document. The active layer can not be deleted.

WARNING: the undo-system of the document is cleared.

Example

```
Dim Ls As Layers

Ls.DeleteEmptyLayers
```

SetProtected, SetVisible, SetSnappingDisabled

Obsolete (replaced by corresponding properties)

Chapter 52 - LayerGroup

Description

The LayerGroup object corresponds with a layer group in a Pythagoras drawing. See also the object type LayerGroups.

Example

```
Dim A As Application
Dim Doc As Document
Dim LGs As LayerGroups
Dim LG As LayerGroup
Dim L As Layer
Dim i As Integer, j As Integer
Dim GroupNames() As String

Set A = New Application
A.GetActiveDocument Doc
If NOT Doc is nothing Then
    Doc.GetLayerGroups LGs
    If LGs.NrOfGroups > 0 Then
        ReDim GroupNames(1 To LGs.NrOfGroups)
        LGs.GetFirst LG
        For i = 1 To LGs.NrOfGroups
            GroupNames(i) = LG.Name
            For j = 1 To LG.NrOfLayers
                MsgBox "Group : " & LG.Name & "    Layer : " & L.Name
            Next j
            LGs.GetNext LG
        Next i
    End If
End if
```

Properties

Name
NrOfLayers

Methods

GetFirst
GetNext
SetProtected
SetVisible
SetSnappingDisabled
SetMinimumAndMaximumScale
AddLayer
RemoveLayer

Properties

Name

Returns
String

Description

Returns or sets the name of the group.

Example

```
Dim LG As LayerGroup

MsgBox "The name of this group was " & LG.Name
LG.Name = "Utilities"
MsgBox "The name of this group is " & LG.Name
```

NrOfLayers

Value
Long

Description

Returns the number of layers in the group.

Example

```
Dim LG As LayerGroup
Dim Nr As Long

Nr = LG.NrOfLayers
```

Methods

GetFirst

Parameters

L As Layer

Description

Returns the first layer of the group.

Example

```
Dim LG As LayerGroup
Dim L As Layer

LG.GetFirst L
MsgBox "First Layer = " & L.Name
```

For a more elaborate example see the introduction on the object LayerGroup.

GetNext

Parameters

L As Layer

Description

Returns the next layer in the group.

Example

```
Dim LG As LayerGroup
Dim L1 As Layer, L2 As Layer

LG.GetFirst L1
MsgBox "First Layer in group = " & L1.Name
LG.GetNext L2
MsgBox "Second Layer in group = " & L2.Name
```

For a more elaborate example see the introduction on the object LayerGroup.

SetProtected

Value

Protected As Boolean

Description

Sets or resets the protection flag of all layers in the group.

Example

```
Dim LG As LayerGroup

LG.SetProtected True
```

SetVisible

Value

Visible As Boolean

Description

Sets or resets the visible flag of all layers in the group.

Example

```
Dim LG As LayerGroup

LG.SetVisible True
```

SetSnappingDisabled

Value

SnappingDisabled As Boolean

Description

Sets or resets the snapping disabled flag of all layers in the group.

Example

```
Dim LG As LayerGroup

LG.SetSnappingDisabled True
```

SetMinimumAndMaximumScale

Parameters

MinimumScale As Double, MaximumScale As Double

Description

Sets the minimum scale and maximum scale at which all layers of the group should be displayed.

MinimumScale **must be less than** MaximumScale.

MinimumScale = ptMinDouble if the layer has to be displayed at any scale up to MaximumScale.

MaximumScale = ptMaxDouble if the layer has to be displayed starting from MinimumScale.

Example

```
Dim LG As LayerGroup

LG.SetMinimumAndMaximumScale 1/500.0, ptMaxDouble
```

AddLayer

Parameters

L As Layer

Description

Adds a layer to the group.

Example

```
Dim LGs As LayerGroup
Dim L As Layer

LGs.AddLayer L
```

RemoveLayer

Parameters

L As Layer

Description

Removes the layer from the group.

Example

```
Dim LG As LayerGroup
Dim L As Layer

LG.RemoveLayer L
```


Chapter 53 - LayerGroups

Description

An instance of an object `LayerGroups` refers to all layer groups in a Pythagoras drawing. See also the object type `LayerGroup`.

Example

See example `LayerGroup`

Properties

`NrOfGroups`

Methods

`GetFirst`
`GetNext`
`NewGroup`
`DeleteGroup`
`SetActive`
`GetActive`

Properties

NrOfGroups

Value

`Long`

Description

Returns the number of layer groups in a document.

Example

```
Dim LGs As LayerGroups
Dim Nr As Long

Nr = LGs.NrOfGroups
```

Methods

GetFirst

Parameters

`LG As LayerGroup`

Description

Returns the first group.

Example

```
Dim LGs As LayerGroups
Dim LG As LayerGroup

LGs.GetFirst LG
MsgBox "First Group = " & LG.Name
```

For a more elaborate example see the introduction on the object `LayerGroup`.

GetNext

Parameters

`L As Layer`

Description

Returns the next layer in the group.

Example

```
Dim LGs As LayerGroups
Dim LG1 As LayerGroup, LG2 As LayerGroup

LGs.GetFirst LG1
MsgBox "First group = " & LG1.Name
```

```

LGS.GetNext LG2
MsgBox "Second group = " & LG2.Name

```

For a more elaborate example see the introduction on the object LayerGroup.

NewGroup

Parameters

```
GroupName As String, LG As LayerGroup
```

Description

Creates a new group with the name GroupName. On return, LG refers to the new group.

Example

```

Dim LGS As LayerGroups
Dim LG As LayerGroup

LGS.NewGroup "Utilities", LG

```

DeleteGroup

Parameters

```
LG As LayerGroup
```

Description

Deletes the group.

Example

```

Dim LGS As LayerGroups
Dim LG As LayerGroup

LGS.DeleteGroup LG

```

SetActive

Parameters

```
LG As LayerGroup
```

Description

Sets the group LG active. This means that only layers of this group will show up in pop-up menus in the control panel and in dialog boxes. If LG = Nothing, no group will be active.

Example

```

Dim LGS As LayerGroups
Dim LG As LayerGroup

LGS.SetActive LG

```

GetActive

Parameters

```
LG As LayerGroup
```

Description

Returns the active layer group. If LG = Nothing, no group is active.

Example

```

Dim LGS As LayerGroups
Dim LG As LayerGroup

LGS.GetActive LG
If LG Is Nothing Then
    MsgBox "No Layer Group is active"
Else
    MsgBox LG.Name & " is the active Layer Group"
End If

```

Chapter 54 - Symbol

Description

The Symbol object is a generic object type that refers to either a PointSymbol, a LineSymbol, a Pattern or a SectionSymbol. These object types have a number of properties and methods common.

See also the object types Symbols, PointSymbol, LineSymbol, Pattern, SectionSymbol, Symbols, SymbolGroups and Library. Be careful changing the symbol library. These operations can not be undone!

Example

```
' This program enumerates all point symbols in the library.
Dim A As Application
Dim Lib As Library
Dim Syms As Symbols
Dim S As Symbol
Dim Ps As PointSymbol
Dim i As Long
Dim AlwaysHorizontal As Boolean

Set A = New Application
A.GetSymbolLibrary Lib
Lib.GetPointSymbols Syms
If Not (Syms Is Nothing) Then
    Syms.GetFirstSymbol S
    For i = 1 To Syms.NrOfSymbols
        MsgBox "Group : " & S.Group & VBCRLF & "Item : " & S.Item
        If S.TypeOfSymbol = ptPointSymbol Then
            S.GetPointSymbol Ps
            AlwaysHorizontal = Not Ps.Rotatable
        End If
        Syms.GetNextSymbol S
    Next i
End If
```

Properties

TypeOfSymbol
Class
Group
Item
FixedColor
ScaleDependent
CanBeRemoved

Methods

GetDocument
GetPointSymbol
GetLineSymbol
GetPattern
GetSectionSymbol

Properties

TypeOfSymbol

Returns

Integer

Description

Returns the type of the symbol. The return value can be ptPointSymbol, ptLineSymbol, ptPattern or ptSectionSymbol.

Example

```
Dim S As Symbol

If S.TypeOfSymbol = ptPattern Then
    BlaBla
End If
```

Class

Returns

Integer

Description

Returns the class of the symbol. A symbol is either a Document or a Library symbol. The return value is either `ptDocumentSymbol` or `ptLibrarySymbol`.

Example

```
Dim S As Symbol

If S.Class = ptDocumentSymbol Then
    BlaBla
End If
```

Group

Returns

String

Description

Returns or sets the name of the group to which of the symbol belongs.

Example

```
Dim S As Symbol

S.Group = "Buildings"      ' Changes the group name part of the symbol name.
```

Item

Returns

String

Description

Returns or sets the item name (name within the group), of the symbol.

Example

```
Dim S As Symbol

S.Item = "House"          ' Changes the 2nd part of the symbol name.
```

FixedColor

Returns

Boolean

Description

Returns True if the color of one or more of the elements of the symbol is fixed. This means that changing the color of an instance of this symbol has only effect on elements of the symbol without fixed color.

Example

```
Dim S As Symbol
Dim Pnt As Point

If S.FixedColor = False Then
    Pnt.Color = ptBlue
End If
```

ScaleDependent

Returns

Boolean

Description

Returns True if the size of the point symbol, the line style or the pattern will depend on the scale of the drawing. The property returns False if the symbol has a fixed size defined in paper dimensions.

Example

```
Dim S As Symbol
Dim B As Boolean

B = S.ScaleDependent
```

CanBeRemoved

Returns

Boolean

Description

Indicates if the symbol can be removed from the library. A symbol can not be removed if it is used in any open document.

Example

```
Dim S As Symbol
Dim Lib As Library

If S.CanBeRemoved Then
    Lib.Remove S
End If
```

Methods

GetDocument

Parameters

Doc As Document

Description

Returns the document object that contains the symbol. For a library symbol Doc = Nothing.

Example

```
Dim Doc As Document
Dim S As Symbol
Dim Lib As Library

S.GetDocument Doc
If (Doc Is Nothing) And S.CanBeRemoved Then
    Lib.Remove S
End If
```

GetPointSymbol

Parameters

PntSymbol As PointSymbol

Description

GetPointSymbol sets the reference to the symbol in a variable of type PointSymbol. Note that the reference is copied, not the symbol.

Example

```
Dim S As Symbol
Dim PntSymbol As PointSymbol

S.GetPointSymbol PntSymbol
```

GetLineSymbol

Parameters

LnSymbol As LineSymbol

Description

GetLineSymbol sets the reference to the symbol in a variable of type LineSymbol. Note that the reference is copied, not the symbol itself.

Example

```
Dim S As Symbol
Dim LnSymbol As LineSymbol

S.GetLineSymbol LnSymbol
```

GetPattern

Parameters

Pat As Pattern

Description

GetPattern sets the reference to the symbol in a variable of type Pattern. Note that the reference is copied, not the symbol itself.

Example

```
Dim S As Symbol
Dim Pat As Pattern

S.GetPattern Pat
```

GetSectionSymbol

Parameters

S As SectionSymbol

Description

GetSectionSymbol sets the reference to the symbol in a variable of type SectionSymbol. Note that the reference is copied, not the symbol itself.

Example

```
Dim S As Symbol
Dim Section As SectionSymbol

S.GetSectionSymbol Section
```

Chapter 55 - PointSymbol

Description

A PointSymbol is a Symbol. A point symbol can not be created, but a reference to an existing symbol can be obtained with the method GetPointSymbol.

Example

```
Dim A As Application
Dim Lib As Library
Dim Syms As Symbols
Dim S As Symbol
Dim Ps As PointSymbol
Dim i As Long
Dim AlwaysHorizontal As Boolean

Set A = New Application
A.GetSymbolLibrary Lib
Lib.GetPointSymbols Syms
If Not (Syms Is Nothing) Then
    Syms.GetFirstSymbol S
    For i = 1 To Syms.NrOfSymbols
        If S.TypeOfSymbol = ptPointSymbol Then
            S.GetPointSymbol Ps
            AlwaysHorizontal = Not Ps.Rotatable
        End If
        Syms.GetNextSymbol S
    Next i
End If
```

Properties

TypeOfSymbol
Class
Group
Item
FixedColor
ScaleDependent
CanBeRemoved
Rotatable
Limits

Methods

GetDocument

Properties

For a description of the properties TypeOfSymbol, Class, Group, Item, FixedColor, ScaleDependent, CanBeRemoved see object type Symbol.

Rotatable

Returns

Boolean

Description

Returns True if the symbol can be rotated.

Example

```
Dim S As PointSymbol

If S.Rotatable Then
    BlaBla
End If
```

Limits

Returns

ObliqueRectangle

Description

Returns the enclosing rectangle of the symbol. The dimensions are relative to the hotspot of the symbol. The dimensions of a scale dependent symbol are real values (e.g. 1m. x 1m.), the dimensions of scale independent symbols are expressed in paper dimensions. (e.g. 3 mm. x 5 mm).

Example

```
Dim R As ObliqueRectangle
Dim S As PointSymbol

R = S.Limits
```

Methods

For a description of the method `GetDocument` see object type `Symbol`.

Chapter 56 - LineSymbol

Description

A `LineSymbol` is a `Symbol`. A line symbol can not be created, but a reference to an existing line style can be obtained with the method `GetLineSymbol`.

Example

See `PointSymbol`

Properties

`TypeOfSymbol`
`Class`
`Group`
`Item`
`FixedColor`
`ScaleDependent`
`CanBeRemoved`
`Width`
`Length`

Methods

`GetDocument`

Properties

For a description of the properties `TypeOfSymbol`, `Class`, `Group`, `Item`, `FixedColor`, `ScaleDependent`, `CanBeRemoved` see object type `Symbol`.

Width

Returns

`Double`

Description

Returns the maximum width of the line style. The width of a scale dependent line style is the real width (e.g. 0.5 m); the width of a scale independent line style is expressed in paper dimensions. (e.g. 3 mm.).

Example

```
Dim S As LineSymbol
Dim W As Double, L As Double

W = S.Width
L = S.Length
```

Length

Returns

`Double`

Description

Returns the length of a segment of the line style. The length of a scale dependent line style is the real length (e.g. 15.0 m), the length of a scale independent lines style is expressed in paper dimensions. (e.g. 5 mm.). Warning: some line styles have a `Length = ptMaxDouble`.

Example

See `Width`

Methods

For a description of the method `GetDocument` see object type `Symbol`.

Chapter 57 - Pattern

Description

A Pattern is a Symbol. A pattern can not be created, but a reference to an existing pattern can be obtained with the method `GetPattern`.

Example

See `PointSymbol`

Properties

`TypeOfSymbol`
`Class`
`Group`
`Item`
`FixedColor`
`ScaleDependent`
`CanBeRemoved`

Methods

`GetDocument`

Properties

For a description of the properties `TypeOfSymbol`, `Class`, `Group`, `Item`, `FixedColor`, `ScaleDependent`, `CanBeRemoved` see object type `Symbol`.

Methods

For a description of the method `GetDocument` see object type `Symbol`.

Chapter 58 - SectionSymbol

Description

A SectionSymbol is a Symbol that refers to a typical section used for road design. A SectionSymbol can not be created, but a reference to an existing typical section can be obtained with the method GetSectionSymbol.

Example

See PointSymbol

Properties

TypeOfSymbol
Class
Group
Item
FixedColor
ScaleDependent
CanBeRemoved

Methods

GetDocument

Properties

For a description of the properties TypeOfSymbol, Class, Group, Item, FixedColor, ScaleDependent, CanBeRemoved see object type Symbol.

Methods

For a description of the method GetDocument see object type Symbol.

Chapter 59 - SymbolGroup

Description

The SymbolGroup object refers to the set of symbols of the same type (Pattern, Line Style, ...) and having the same group name. A SymbolGroup can be obtained from an object Symbols e.g. with the method GetGroup.

Example

```
' This program enumerates all point symbols groups in the library.
Dim A As Application
Dim Lib As Library
Dim Syms As Symbols
Dim G As SymbolGroup
Dim Ps As PointSymbol
Dim i As Long
Dim AlwaysHorizontal As Boolean

Set A = New Application
A.GetSymbolLibrary Lib
Lib.GetPointSymbols Syms
If Not (Syms Is Nothing) Then
    Syms.GetFirstGroup G
    For i = 1 To Syms.NrOfGroups
        MsgBox "Group : " & G.Group
        Syms.GetNextGroup G
    Next i
End If
```

Properties

TypeOfSymbol
Class
Group
NrOfSymbols

Methods

GetDocument
GetFirst
GetNext

Properties

TypeOfSymbol

Returns

Integer

Description

Returns the type of the symbols in the group. The return value can be ptPointSymbol, ptLineSymbol, ptPattern or ptSectionSymbol.

Example

```
Dim G As SymbolGroup

If G.TypeOfSymbol = ptPattern Then
    BlaBla
End If
```

Class

Returns

Integer

Description

Returns the class of the symbols in the group. The symbols of the group are either Document or Library symbols. The return value is either ptDocumentSymbol or ptLibrarySymbol.

Example

```
Dim G As SymbolGroup
```

```
If G.Class = ptDocumentSymbolThen
    BlaBla
End If
```

Group

Returns

String

Description

Returns or sets the name of the group.

Example

```
Dim G As SymbolGroup

G.Group = "Buildings"      ' Changes the group name.
```

NrOfSymbols

Returns

Long

Description

Returns the number of symbols in the group.

Example

```
Dim G As SymbolGroup
Dim S As Symbol
Dim Lib As Library
Dim i As Long

For i = 1 To G.NrOfSymbols
    If i = 1 Then
        G.GetFirst S
    Else
        G.GetNext S
    End If
Next i
```

Methods

GetDocument

Parameters

Doc As Document

Description

If the group was obtained from a document symbol library, then this method returns the document object that contains the group. For a library symbol Doc = Nothing.

Example

```
Dim Doc As Document
Dim G As SymbolGroup

G.GetDocument Doc
```

GetFirst

Parameters

Sym As Symbol

Description

GetFirst returns the first symbol of the group.

Example

See Property NrOfSymbols

GetNext

Parameters

Sym As Symbol

Description

`GetNext` returns the next symbol of the group.

Example

See Property `NrOfSymbols`

Chapter 60 - Symbols

Description

The `Symbols` object refers to the complete set of symbols of the same type in a document library or in the system library. A `Symbols` object can be obtained from the `Library` object with the methods `GetPointSymbols`, `GetLineSymbols`, `GetPatterns` and `GetSectionSymbols`.

Example

```
' This program enumerates all point symbols in the library.
Dim A As Application
Dim Lib As Library
Dim Syms As Symbols
Dim G As Symbol
Dim Ps As PointSymbol
Dim i As Long
Dim AlwaysHorizontal As Boolean

Set A = New Application
A.GetSymbolLibrary Lib
Lib.GetPointSymbols Syms
If Not (Syms Is Nothing) Then
    Syms.GetFirstSymbol G
    For i = 1 To Syms.NrOfSymbols
        Enumerate G 'a specific procedure to enumerate a symbol
        Syms.GetNextSymbol G
    Next i
End If
```

Properties

- `TypeOfSymbol`
- `Class`
- `NrOfSymbols`
- `NrOfGroups`

Methods

- `GetDocument`
- `GetFirstSymbol`
- `GetNextSymbol`
- `GetSymbol`
- `GetFirstGroup`
- `GetNextGroup`
- `GetGroup`
- `RemoveAllUnusedSymbols`

Properties

TypeOfSymbol

Returns

Integer

Description

Returns the type of the symbols in the set. The return value can be `ptPointSymbol`, `ptLineSymbol`, `ptPattern` or `ptSectionSymbol`.

Example

```
Dim Syms As Symbols

If Syms.TypeOfSymbol = ptPattern Then
    BlaBla
End If
```

Class

Returns

Integer

Description

Returns the class of the symbols in the set. The symbols of the set are either Document or System Library symbols. The return value is either `ptDocumentSymbol` or `ptLibrarySymbol`.

Example

```
Dim Syms As Symbols

If Syms.Class = ptDocumentSymbolThen
    BlaBla
End If
```

NrOfSymbols

Returns

Long

Description

Returns the number of symbols in the set.

Example

```
Dim Syms As Symbols
Dim S As Symbol
Dim i As Long

Syms.GetFirstSymbol S
For i = 1 To Syms.NrOfSymbols
    Syms.GetNextSymbol S
Next i
```

NrOfGroups

Returns

Long

Description

Returns the number of groups in the set.

Example

```
Dim Syms As Symbols
Dim G As SymbolGroup

Syms.GetFirstGroup G
For i = 1 To Syms.NrOfGroups
    Syms.GetNextGroup G
Next i
```

Methods

GetDocument

Parameters

Doc As Document

Description

If the group was obtained from a document symbol library, then this method returns the document object that contains the group. For a library symbol `Doc = Nothing`.

Example

```
Dim Doc As Document
Dim Syms As Symbols

Syms.GetDocument Doc
```

GetFirstSymbol

Parameters

Sym As Symbol

Description

`GetFirstSymbol` returns the first symbol of the set.

Example

See Property `NrOfSymbols`

GetNextSymbol

Parameters

`Sym As Symbol`

Description

`GetNextSymbol` returns the next symbol of the set.

Example

See Property `NrOfSymbols`

GetSymbol

Parameters

`Group As String, Item As String, Sym As Symbol`

Description

`GetSymbol` returns the symbol with the given name. If the symbol is not found, `Sym = Nothing` is returned.

Example

```
Dim Syms As Symbols
```

```
Dim S As Symbol
```

```
Syms.GetSymbol "Buildings", "Concrete", S
```

GetFirstGroup

Parameters

`G As SymbolGroup`

Description

`GetFirstGroup` returns the first group of the set.

Example

See Property `NrOfGroups`

GetNextGroup

Parameters

`G As SymbolGroup`

Description

`GetNextGroup` returns the next group of the set.

Example

See Property `NrOfGroups`

GetGroup

Parameters

`GroupName As String, G As SymbolGroup`

Description

`GetGroup` returns the group with the name given by the parameter `GroupName`. If the group is not found, `G = Nothing` is returned.

Example

```
Dim Syms As Symbols
```

```
Dim G As SymbolGroup
```

```
Syms.GetGroup "Buildings", G
```

RemoveAllUnusedSymbols

Parameters

Description

`RemoveAllUnusedSymbols` removes all symbols of the set which are not used in the document.

Example

```
Dim Syms As Symbols
```

```
Syms.RemoveAllUnusedSymbols
```

Chapter 61 - Library

Description

The `Library` object refers to the complete set of symbols of in a document or in the system library. A `Library` object can be obtained from the `Document` object with the method `GetSymbolLibrary`, or from the `Application` object with the method `GetSymbolLibrary`.

Example

```
Dim A As Application
Dim Lib As Library
Dim PSymbols As Symbols, LSymbols As Symbols, _
    HSymbols As Symbols, SSymbols As Symbols

Set A = New Application
A.GetSymbolLibrary Lib
Lib.GetPointSymbols PSymbols
Lib.GetLineSymbols LSymbols
Lib.GetPatterns HSymbols
Lib.GetSectionSymbols SSymbols
```

Properties

`Class`

Methods

`GetDocument`
`GetPointSymbols`
`GetLineSymbols`
`GetPatterns`
`GetSectionSymbols`
`Add`
`Remove`
`CreateSymbol`
`RemoveAllUnusedSymbols`

Properties

Class

Returns

`Integer`

Description

Returns the class of the symbols in the library. The symbols of the set are either `Document` or `System Library` symbols. The return value is either `ptDocumentSymbol` or `ptLibrarySymbol`.

Example

```
Dim Lib As Library

If Lib.Class = ptDocumentSymbolThen
    BlaBla
End If
```

Methods

GetDocument

Parameters

`Doc As Document`

Description

This method returns the document object this library belongs to. For the system library (`Class = ptLibrarySymbol`) : `Doc = Nothing`.

Example

```
Dim Doc As Document
Dim Lib As Library

Lib.GetDocument Doc
```

GetPointSymbols

Parameters

Syms As Symbols

Description

GetPointSymbols returns the set of point symbols in the library.

Example

```
Dim Syms As Symbols  
Dim Lib As Library
```

```
Lib.GetPointSymbols Syms
```

GetLineSymbols

Parameters

Syms As Symbols

Description

GetLineSymbols returns the set of line styles in the library.

Example

```
Dim Syms As Symbols  
Dim Lib As Library
```

```
Lib.GetLineSymbols Syms
```

GetPatterns

Parameters

Syms As Symbols

Description

GetPatterns returns the set of patterns in the library.

Example

```
Dim Syms As Symbols  
Dim Lib As Library
```

```
Lib.GetPatterns Syms
```

GetSectionSymbols

Parameters

Syms As Symbols

Description

GetSectionSymbols returns the set of section symbols in the library.

Example

```
Dim Syms As Symbols  
Dim Lib As Library
```

```
Lib.GetSectionSymbols Syms
```

Add

Parameters

S As Symbol

Description

The method Add moves a document symbol to the system library.

Example

```
Dim S As Symbol  
Dim Lib As Library
```

```
Lib.Add S
```

Remove

Parameters

S As Symbol

Description

The method `Remove` removes a symbol from the library. If the symbol can not be removed because it is used in an open document, an error will be raised.

WARNING: the undo-system of the document is cleared.

Example

```
Dim S As Symbol
Dim Lib As Library

Lib.Remove S
```

CreateSymbol

Parameters

Container As SymbolContainer, Group As String, Name As String,
ScaleDependent As Boolean, Rotatable As Boolean, NorthSymbol As Boolean,
NewSymbol As Symbol

Description

This method creates the `Symbol` object from the argument `SymbolContainer`. `ParametersGroup`, `Name`, `ScaleDependent`, `Rotatable`, and `NorthSymbol` specify attributes of the symbol being created. If the argument `ScaleDependent` is `True` then all dimensions are considered as local dimensions, otherwise – as paper dimensions. The generic symbol object `NewSymbol` referring to the created symbol is returned. If the symbol can't be created the value of `Nothing` is returned. All dimensions in `SymbolContainer` must be supplied in meters.

Example

```
Dim Sym As Symbol
Dim Lib As Library
Dim Sc As SymbolContainer

Lib.CreateSymbol Sc, "Samples", "SampleSym1", False, True, False, Sym
```

RemoveAllUnusedSymbols

Parameters

None

Description

`RemoveAllUnusedSymbols` removes all symbols of the set that are not used in the library.

WARNING: the undo-system of the document is cleared.

Example

```
Dim Lib As Library

Lib.RemoveAllUnusedSymbols
```

Chapter 62 - SymbolContainer

Description

A SymbolContainer object is a container object to collect the information about symbol elements. It is supposed that a symbol container will be used afterwards to create a symbol. Methods of SymbolContainer return True if success and False — otherwise.

Example

See example for the Library method CreateSymbol.

Methods

AddLine
AddColoredLine
AddArc
AddColoredArc
AddCircle
AddColoredCircle
AddText
AddColoredText
AddPolygon
AddColoredPolygon

Methods

AddLine

Parameters

P1 As XY, P2 As XY, Width As Double

Returns

Boolean

Description

Adds a non-colored line from point P1 to point P2.

Example

```
Dim Sc As SymbolContainer
Dim p1 As XY
Dim p2 As XY
Dim Ok As Boolean

Set Sc = New SymbolContainer
p1.X = 1.0
p1.Y = 2.0
p2.X = 5.0
p2.Y = 6.0
Ok = Sc.AddLine(p1, p2, 1)
```

AddColoredLine

Parameters

P1 As XY, P2 As XY, Width As Double, color As RGB

Returns

Boolean

Description

Adds a line with the specified color from point P1 to point P2.

Example

```
Dim Sc As SymbolContainer
Dim p1 As XY
Dim p2 As XY
Dim color As RGB
Dim Ok As Boolean

Set Sc = New SymbolContainer
p1.X = 1.0
p1.Y = 2.0
```

```

p2.X = 5.0
p2.Y = 6.0
color.Red = 120
color.Green = 250
color.Blue = 65
Ok = Sc.AddColoredLine(p1, p2, 1, color)

```

AddArc

Parameters

Center As XY, Radius As Double, Angle1 As Double, Angle2 As Double,
Width As Double

Returns

Boolean

Description

Adds a non-colored arc with the specified center and radius from the angle Angle1 to the angle Angle2 counterclockwise.

Example

```

Dim Sc As SymbolContainer
Dim c As XY
Dim Ok As Boolean

Set Sc = New SymbolContainer
c.X = 5.0
c.Y = 6.0
Ok = Sc.AddArc(c, 1, 1.5, 2.2, 0.2)

```

AddColoredArc

Parameters

Center As XY, Radius As Double, Angle1 As Double, Angle2 As Double,
Width As Double, color As RGB

Returns

Boolean

Description

Adds an arc with the specified center, radius, and color from the angle Angle1 to the angle Angle2 counterclockwise.

Example

```

Dim Sc As SymbolContainer
Dim c As XY
Dim color As RGB
Dim Ok As Boolean

Set Sc = New SymbolContainer
c.X = 5.0
c.Y = 6.0
color.Red = 0
color.Green = 255
color.Blue = 0
Ok = Sc.AddColoredArc(c, 1, 1.5, 2.2, 0.2, color)

```

AddCircle

Parameters

Center As XY, Radius As Double, Width As Double, Filling As Integer

Returns

Boolean

Description

Adds a non-colored circle with the specified center and radius. Only the values ptNoFill, ptSolidFill, ptGrayxx (xx = 75, 50, 25 or 12) are valid for Filling.

Example

```

Dim Sc As SymbolContainer
Dim c As XY

```

```

Set Sc = New SymbolContainer
c.X = 5.0
c.Y = 6.0
Sc.AddCircle(c, 0.7, 1, ptSolidFill)           ` Will be filled with a symbol
color

```

AddColoredCircle

Parameters

Center As XY, Radius As Double, Width As Double, Filling As Integer,
color As RGB

Returns

Boolean

Description

Adds a non-colored circle with the specified center, radius, and color . Only the values ptNoFill, ptSolidFill, ptGrayxx (xx = 75, 50, 25 or 12) are valid for Filling.

Example

```

Dim Sc As SymbolContainer
Dim c As XY
Dim color As RGB
Dim Ok As Boolean

Set Sc = New SymbolContainer
c.X = 5.0
c.Y = 6.0
color.Red = 0
color.Green = 0
color.Blue = 255
Ok = Sc.AddColoredCircle(c, 0.7, 1, ptGray25, color) ` Filled with 25% of blue

```

AddText

Parameters

Position As XY, Text As String, Height As Double, Weight As Integer,
Style As Integer

Returns

Boolean

Description

Adds a non-colored text with content Text at position Position . Value of Height must be specified in meters.

Example

```

Dim Sc As SymbolContainer
Dim p As XY
Dim Ok As Boolean

Set Sc = New SymbolContainer
p.X = 5.0
p.Y = 6.0
Ok = Sc.AddText(p, "Just a text", 0.004, ptMedium, ptItalic)

```

AddColoredText

Parameters

Position As XY, Text As String, Height As Double, Weight As Integer,
Style As Integer , color As RGB

Returns

Boolean

Description

Adds text with content Text at position Position and with the specified color . Value of Height must be specified in meters.

Example

```

Dim Sc As SymbolContainer

```



```

Dim p As XY
Dim color As RGB
Dim Ok As Boolean

Set Sc = New SymbolContainer
p.X = 5.0
p.Y = 6.0
color.Red = 255
color.Green = 0
color.Blue = 0
Ok = Sc.AddColoredText(p, "Just a text", 0.004, ptLight, ptUpright, color)

```

AddPolygon

Parameters

Points() As XY, Width As Double, Filling As Integer

Returns

Boolean

Description

Adds a non-colored polygon. Polygon is composed of straight lines connecting adjacent points specified by the argument `Points`. A polygon must be closed, this means the last point must end in the first one. If not, symbol container will add the missing end point. A polygon must be non-empty, so the minimal size of the points array is equal to 4. Only the values `ptNoFill`, `ptSolidFill`, `ptGrayxx` (`xx` = 75, 50, 25 or 12) are valid for `Filling`. `Width` is a value between 0.0 and 2.0. If the value = 0.0, the lines of the polygon will not become part of the symbol. In that case, the `Filling` must be different from `ptNoFill`.

Example

```

Dim Sc As SymbolContainer
Dim points() As XY
Dim Ok As Boolean

Set Sc = New SymbolContainer
Redim points(1 To 4)
Points(1).X = 5.0
Points(1).Y = 6.0
Points(2).X = 5.0
Points(2).Y = 8.0
Points(3).X = 8.0
Points(3).Y = 8.0
Points(4).X = 5.0
Points(4).Y = 6.0
Ok = Sc.AddPolygon(points, 0.7, ptGray25) ' Will be filled with a symbol
color

```

AddColoredPolygon

Parameters

Points() As XY, Width As Double, Filling As Integer, color As RGB

Returns

Boolean

Description

Adds a non-colored polygon. Polygon is composed from straight lines connecting adjacent points specified by the argument `Points`. A polygon must be closed, this means the last point must end in the first one. If not, symbol container will add the missing end point. A polygon must be non-empty, so the minimal size of the points array is equal to 4. Only the values `ptNoFill`, `ptSolidFill`, `ptGrayxx` (`xx` = 75, 50, 25 or 12) are valid for `Filling`. The lines of the polygon will have the same color as the filling. `Width` is a value between 0.0 and 2.0. If the value = 0.0, the lines of the polygon will not become part of the symbol. In that case, the `Filling` must be different from `ptNoFill`.

Example

```

Dim Sc As SymbolContainer
Dim points() As XY
Dim color As RGB
Dim Ok As Boolean

Set Sc = New SymbolContainer
color.Red = 0

```

```
color.Green = 255
color.Blue = 0
Redim points(1 To 4)
Points(1).X = 5.0
Points(1).Y = 6.0
Points(2).X = 5.0
Points(2).Y = 8.0
Points(3).X = 8.0
Points(3).Y = 8.0
Points(4).X = 5.0
Points(4).Y = 6.0
Ok = Sc.AddColoredPolygon(points, 0.7, ptGray25, color)  ' Filled with 25% of
green
```

Chapter 63 - UserDefaults

Description

A UserDefaults object contains the actual list of all user defaults that are accessible to the Pythagoras user. The UserDefaults object, and each UserDefault obtained using the methods GetFirst and GetNext, remain dynamically linked to the actual settings in Pythagoras. A deleted User Default in Pythagoras will make the corresponding VBA Object invalid.

Example

```
Dim App As Application
Dim UDs As UserDefaults
Dim UD As UserDefault
Dim Attribs as AttributesSet
Dim i As Long

Set App = New Application
App.GetUserDefaults UDs
For i = 1 To UDs.NrUserDefaults
    If i = 1 Then
        UDs.GetFirst UD
    Else
        UDs.GetNext UD
    End If
    Set Attribs = UD.AttributesSet
    MsgBox UD.Name
Next i
```

Properties

NrUserDefaults

Methods

GetFirst
GetNext
SetActive
GetActive
RemoveUserDefault
AddUserDefault

Properties

NrUserDefaults

Returns

Long

Description

This property returns the number of user defaults.

Example

See introduction.

Methods

GetFirst

Parameters

UD As UserDefault

Description

Returns the first User Default

Example

```
Dim App As Application
Dim UDs As UserDefaults
Dim UD As UserDefault
Dim i As Long

Set App = New Application
```

```

App.GetUserDefaults UDs
For i = 1 To UDs.NrUserDefaults
    If i = 1 Then
        UDs.GetFirst UD
    Else
        UDs.GetNext UD
    End If
Next i

```

GetNext

Parameters

UD As UserDefault

Description

Returns the next User Default.

Example

See GetFirst

SetActive

Parameters

UserDefault

Description

This method sets the active user default in Pythagoras. Setting Active to Nothing, will set the active User Default in Pythagoras to None.

Example

```

Dim App As Application
Dim UDs As UserDefaults
Dim UD As UserDefault
Dim Attribs as AttributesSet

Set App = New Application
App.GetUserDefaults UDs
' Get a UD that you want to set active
UDs.SetActive UD

```

GetActive

Parameters

UserDefault

Description

This property returns the active user default in Pythagoras. The method returns Nothing if no UserDefault is active.

Example

```

Dim App As Application
Dim UDs As UserDefaults
Dim UD As UserDefault
Dim Attribs as AttributesSet

Set App = New Application
App.GetUserDefaults UDs
UDs.GetActive UD
If Not UD Is Nothing Then
    Attribs = UD.AttributesSet
End If

```

RemoveUserDefault

Parameters

UD As UserDefault

Description

The method `RemoveUserDefault` removes a user default. If the user default can not be removed, an error will be raised.

Example

```
Dim UDs As UserDefaults
Dim UD As UserDefault

UDs.RemoveUserDefault UD
```

AddUserDefault

Parameters

Attribs As AttributesSet, Group As String, Name As String, UD As UserDefault

Description

The method AddUserDefault adds a new user default. If a user default with the given name already exists, an error will be raised. The AnnotationAttributes belonging to ptAn_GeneralAnnotation, is not considered as default value.

The method returns a UserDefault object.

Example

```
Dim UDs As UserDefaults
Dim UD As UserDefault

UDs.AddUserDefault Attribs, Group, Name, UD
```

Chapter 64 - UserDefault

Description

A UserDefault object contains all information of a user default. The UserDefault object obtained from UserDefaults remains dynamically linked to the actual settings in Pythagoras. A deleted User Default in Pythagoras will make the corresponding VBA Object invalid.

Example

```
Dim App As Application
Dim UDs As UserDefaults
Dim UD As UserDefault
Dim Attribs as AttributesSet
Dim LayerName As String
Dim Group As String
Dim Name As String

Set App = New Application
App.GetUserDefaults UDs
UDs.GetActive UD
If Not UD Is Nothing Then
    Attribs = UD.AttributesSet
    If Attribs.HasLayerName Then
        LayerName = Attribs.LayerName
    End If
    MsgBox "The user default " & Group & ", " & Name & "uses layer: " LayerName
End If
```

Properties

Group
Name
Active
AttributesSet

Methods

GetFullName

Properties

Group

Returns

String

Description

This property returns the Group the User Default belongs to.

Example

```
Dim UD As UserDefault
Dim Group As String
Dim Name As String

Group = UD.Group
```

Name

Returns

String

Description

This property returns the Name, within the group, of the User Default.

Example

```
Dim UD As UserDefault
Dim Name As String

Name = UD.Name
```

Active

Returns

Boolean

Description

Returns True if the User Default is active.

When setting the User Default to True, the user default becomes the active one. Setting Active to False, will set the active User Default in Pythagoras to None.

Example

```
Dim UD As UserDefault

UD.Active = True
```

AttributesSet

Returns

AttributesSet

Description

Returns an AttributesSet whose settings correspond with the settings in the UserDefault. The AnnotationAttributes belonging to the ptAn_GeneralAnnotation kind, doesn't have a default value.

Example

```
Dim UD As UserDefault
Dim Attribs As AttributesSet

Set Attribs = UD.AttributesSet
```

Methods

GetFullName

Parameters

Group As String, Name As String

Description

Returns the Group and Name of the User Default

Example

```
Dim UD As UserDefault
Dim Group As String
Dim Name As String

UD.GetFullName Group, Name
```

Chapter 65 - AttributesSet

Description

The AttributesSet is a container for the 6 objects PointAttributes, LineAttributes, PolygonAttributes, SolidAttributes, TextAttributes and ImageAttributes, and for each annotation kind the respective AnnotationAttributes, together with a common property for Color, Layer Name, Display Level and Comment. These common attributes apply to all object types unless the corresponding attribute is assigned to a specific object type.

Example

```
Dim App as Application
Dim Doc as Document
Dim Ovl as Overlay
Dim All as AllObjects
Dim Attribs as AttributesSet
Dim Pattribs as PointAttributes
Dim Lattribs as LineAttributes
Dim PolyAttribs as PolygonAttributes
Dim Sattribs as SolidAttributes
Dim TextAttribs as TextAttributes
Dim Iattribs as ImageAttributes
Dim Blue as RGB

Blue.Blue = 255
Set App = New Application
App.GetActivedocument Doc
Doc.GetOverlay ptDrawing, Ovl
Ovl.GetAllObjects All
Set Attribs = New AttributesSet
Set Pattribs = Attribs.PointAttributes
Set Lattribs = Attribs.LineAttributes
Set PolyAttribs = Attribs.PolygonAttributes
Set Sattribs = Attribs.SolidAttributes
Set TextAttribs = Attribs.TextAttributes
Set Iattribs = Attribs.ImageAttributes
Attribs.TrueColor = Blue
Attribs.HasColor = True
Attribs.Layer = "Buildings"
Attribs.HasLayer = True
All.ApplyThematics Attribs
```

Properties

```
PointAttributes
LineAttributes
PolygonAttributes
SolidAttributes
TextAttributes
ImageAttributes
TrueColor
HasColor
Layer
HasLayer
DisplayLevel
HasDisplayLevel
Comment
HasComment
PointConfirmation
HasPointConfirmation
Copy
```

Methods

```
GetAnnotationAttributes
SetAnnotationAttributes
```


Properties

PointAttributes

Returns

PointAttributes

Description

This property returns or sets the PointAttributes.

If this is set to nothing, it will reset the PointAttributes to the default values.

Example

```
Dim Attribs As AttributesSet
Dim Pattribs As PointAttributes

Set Pattribs = Attribs.PointAttributes
Set Attribs.PointAttributes = Pattribs
```

LineAttributes

Returns

LineAttributes

Description

This property returns or sets the LineAttributes.

If this is set to nothing, it will reset the LineAttributes to the default values.

Example

```
Dim Attribs As AttributesSet
Dim Lattribs As LineAttributes

Set Lattribs = Attribs.LineAttributes
Set Attribs.LineAttributes = Lattribs
```

PolygonAttributes

Returns

PolygonAttributes

Description

This property returns or sets the PolygonAttributes.

If this is set to nothing, it will reset the PolygonAttributes to the default values.

Example

```
Dim Attribs As AttributesSet
Dim Polyattribs As PolygonAttributes

Set Polyattribs = Attribs.PolygonAttributes
Set Attribs.PolygonAttributes = PolyAttribs
```

SolidAttributes

Returns

SolidAttributes

Description

This property returns or sets the SolidAttributes.

If this is set to nothing, it will reset the SolidAttributes to the default values.

Example

```
Dim Attribs As AttributesSet
Dim Sattribs As SolidAttributes

Set Sattribs = Attribs.SolidAttributes
Set Attribs.SolidAttributes = Sattribs
```

TextAttributes

Returns

TextAttributes

Description

This property returns or sets the `TextAttributes`.

If this is set to nothing, it will reset the `TextAttributes` to the default values.

Example

```
Dim Attribs As AttributesSet
Dim TextAttribs As TextAttributes

Set TextAttribs = Attribs.TextAttributes
Set Attribs.TextAttributes = TextAttribs
```

ImageAttributes

Returns

`ImageAttributes`

Description

This property returns or sets the `ImageAttributes`.

If this is set to nothing, it will reset the `ImageAttributes` to the default values.

Example

```
Dim Attribs As AttributesSet
Dim Iattribs As ImageAttributes

Set Iattribs = Attribs.ImageAttributes
Set Attribs.ImageAttributes = Iattribs
```

TrueColor

Returns

`RGB`

Description

This property returns or sets the color (RGB) of the `AttributesSet`.

Example

```
Dim Attribs As AttributesSet
Dim Red as RGB

Red.Red = 255
Set Attribs = New AttributesSet
Attribs.TrueColor = Red
```

HasColor

Returns

`Boolean`

Description

This property returns or sets a boolean that enables or disables the color of the `AttributesSet`.

Example

```
Dim Attribs As AttributesSet
Dim Red as RGB

Red.Red = 255
Set Attribs = New AttributesSet
Attribs.TrueColor = Red
Attribs.HasColor = True
```

Layer

Returns

`String`

Description

This property returns or sets the `Layer` of the `AttributesSet`.

Example

```
Dim Attribs As AttributesSet

Set Attribs = New AttributesSet
```

```
Attribs.Layer = "Buildings"
```

HasLayer

Returns

Boolean

Description

This property returns or sets a boolean that enables or disables the layer of the AttributesSet.

Example

```
Dim Attribs As AttributesSet
Dim Layer As String

Set Attribs = New AttributesSet
Attribs.Layer = Layer
Attribs.HasLayer = True
```

DisplayLevel

Returns

Integer

Description

This property returns or sets the display level of the AttributesSet.

Example

```
Dim Attribs As AttributesSet

Set Attribs = New AttributesSet
Attribs.DisplayLevel = -5
Attribs.HasDisplayLevel = True
```

HasDisplayLevel

Returns

Boolean

Description

This property returns or sets a boolean that enables or disables the display level of the AttributesSet.

Example

```
Dim Attribs As AttributesSet
Dim Red as RGB

Set Attribs = New AttributesSet
Attribs.HasDisplayLevel = False
```

Comment

Returns

String

Description

This property returns or sets the comment (object information) of the AttributesSet.

Example

```
Dim Attribs As AttributesSet

Set Attribs = New AttributesSet
Attribs.Comment = "DH"
Attribs.HasComment = True
```

HasComment

Returns

Boolean

Description

This property returns or sets a boolean that enables or disables the comment of the AttributesSet.

Example

```
Dim Attribs As AttributesSet
```

```
Set Attribs = New AttributesSet
Attribs.HasComment = False
```

PointConfirmation

Returns
Boolean

Description
This property returns or sets Confirmation of the AttributesSet.

Example

```
Dim Attribs As AttributesSet

Set Attribs = New AttributesSet
Attribs.PointConfirmation = False
```

HasPointConfirmation

Returns
Boolean

Description
This property returns or sets a boolean that enables or disables that coordinates/distances/... have to be confirmed.

Example

```
Dim Attribs As AttributesSet

Set Attribs = New AttributesSet
Attribs.PointConfirmation = False
Attribs.HasPointConfirmation = True
```

Copy

Returns
Attribs as AttributesSet

Description
This property returns a copy of the AttributesSet

Example

```
Dim Attribs as AttributesSet
Dim Attribs2 as AttributesSet

Set Attribs = New AttributesSet
Set Attribs2 = Attribs.Copy
```

Methods

GetAnnotationAttributes

Parameters
AnKind As AnnotationKind, AnAttribs As AnnotationAttributes

Description
Returns the AnnotationAttributes of the specified annotation kind. This type can be the following:

```
AnKind = ptAn_TotalLength (= 1)
AnKind = ptAn_Coordinate (= 2)
AnKind = ptAn_Elevation (= 3)
AnKind = ptAn_PointId (= 4)
AnKind = ptAn_Area (= 5)
AnKind = ptAn_LineLength (= 6)
AnKind = ptAn_Comment (= 7)
AnKind = ptAn_Bearing (= 8)
AnKind = ptAn_SlopeDistance (= 9)
AnKind = ptAn_Dz (= 10)
AnKind = ptAn_Gradient (= 11)
```

```

AnKind = ptAn_Angle (= 12)
AnKind = ptAn_Radius (= 13)
AnKind = ptAn_DeltaX (= 14)
AnKind = ptAn_DeltaY (= 15)
AnKind = ptAn_Volume (= 16)
AnKind = ptAn_GeneralAnnotation (= 17) : This type is useful for the Selection, FlaggedObject
and AllObject methods ApplyThematis, GetThematis and RemoveThematis.

```

Example

```

Dim Attribs as AttributesSet
Dim AnAttribs as AnnotationAttributes

Set Attribs = New AttributesSet
Attribs.GetAnnotationAttributes ptAn_GeneralAnnotation, AnAttribs

```

SetAnnotationAttributes

Parameters

```

AnKind As AnnotationKind, AnAttribs As AnnotationAttributes

```

Description

Sets the AnnotationAttributes of the specified annotation kind. This type can be the following:

```

AnKind = ptAn_TotalLength (= 1)
AnKind = ptAn_Coordinate (= 2)
AnKind = ptAn_Elevation (= 3)
AnKind = ptAn_PointId (= 4)
AnKind = ptAn_Area (= 5)
AnKind = ptAn_LineLength (= 6)
AnKind = ptAn_Comment (= 7)
AnKind = ptAn_Bearing (= 8)
AnKind = ptAn_SlopeDistance (= 9)
AnKind = ptAn_Dz (= 10)
AnKind = ptAn_Gradient (= 11)
AnKind = ptAn_Angle (= 12)
AnKind = ptAn_Radius (= 13)
AnKind = ptAn_DeltaX (= 14)
AnKind = ptAn_DeltaY (= 15)
AnKind = ptAn_GeneralAnnotation (= 16) : This type is useful for the Selection, FlaggedObject
and AllObject methods ApplyThematis, GetThematis and RemoveThematis.

```

Example

```

Dim Attribs as AttributesSet
Dim AnAttribs as AnnotationAttributes

Set Attribs = New AttributesSet
Set AnAttribs = New AnnotationAttributes
Attribs.SetAnnotationAttributes ptAn_GeneralAnnotation, AnAttribs

```

Chapter 66 - PointAttributes

Description

PointAttributes is a collection of visual attributes for point objects, with an option to enable or disable each attribute. It can be a part of AttributesSet, but works individually as well.

Example

```
Dim Attribs as AttributesSet
Dim Pattribs as PointAttributes

Set Attribs = New AttributesSet
Set Pattribs = Attribs.PointAttributes
```

Properties

PointStyle
HasStyle
TrueColor
HasColor
Layer
HasLayer
DisplayLevel
HasDisplayLevel
Comment
HasComment
PointIdent
HasPointIdent
ZoomFactor
HasZoomFactor
HasScaleFactor
Copy

Methods

GetScaleFactor
SetScaleFactor

Properties

PointStyle

Returns

Style

Description

This property returns or sets the style of PointAttributes.

Example

```
Dim Pattribs As PointAttributes
Dim Pstyle as Style

Set Pattribs = New PointAttributes
Pattribs.PointStyle = Pstyle
```

HasStyle

Returns

Boolean

Description

This property returns or sets a boolean that enables or disables the style of PointAttributes.

Example

```
Dim Pattribs As PointAttributes
Dim Pstyle as Style

Set Pattribs = New PointAttributes
Pattribs.HasStyle = True
Pattribs.PointStyle = Pstyle
```

TrueColor

Returns

RGB

Description

This property returns or sets the color (RGB) of PointAttributes.

Example

```
Dim Pattribs As PointAttributes
Dim Red as RGB

Red.Red = 255
Set Pattribs = New PointAttributes
Pattribs.TrueColor = Red
```

HasColor

Returns

Boolean

Description

This property returns or sets a boolean that enables or disables the color of PointAttributes.

Example

```
Dim Pattribs As PointAttributes
Dim Red as RGB

Red.Red = 255
Set Pattribs = New PointAttributes
Pattribs.TrueColor = Red
Pattribs.HasColor = TRUE
```

Layer

Returns

String

Description

This property returns or sets the Layer of PointAttributes.

Example

```
Dim Pattribs As PointAttributes

Set PAttribs = New PointAttributes
Pattribs.Layer = "Buildings"
```

HasLayer

Returns

Boolean

Description

This property returns or sets a boolean that enables or disables the layer of PointAttributes.

Example

```
Dim Pattribs As PointAttributes

Set PAttribs = New PointAttributes
Pattribs.Layer = "Buildings"
Pattribs.HasLayer = True
```

DisplayLevel

Returns

Integer

Description

This property returns or sets the display level of the PointAttributes.

Example

```
Dim Pattribs As PointAttributes
```

```
Set PAttribs = New PointAttributes
PAttribs.DisplayLevel = -5
PAttribs.HasDisplayLevel = True
```

HasDisplayLevel

Returns

Boolean

Description

This property returns or sets a boolean that enables or disables the display level of the PointAttributes.

Example

```
Dim PAttribs As PointAttributes

Set PAttribs = New PointAttributes
PAttribs.HasDisplayLevel = False
```

Comment

Returns

String

Description

This property returns or sets the comment (object information) of thePointAttributes.

Example

```
Dim PAttribs As PointAttributes

Set PAttribs = New PointAttributes
PAttribs.Comment = "DH"
PAttribs.HasComment = True
```

HasComment

Returns

Boolean

Description

This property returns or sets a boolean that enables or disables the comment of the PointAttributes.

Example

```
Dim PAttribs As PointAttributes

Set PAttribs = New PointAttributes
PAttribs.HasComment = False
```

PointIdent

Returns

Boolean

Description

This property returns or sets the indication if a new created point will get the next point number or no point number.

Example

```
Dim PAttribs As PointAttributes

Set PAttribs = New PointAttributes
PAttribs.PointIdent = False
```

HasPointIdent

Returns

Boolean

Description

This property returns or sets a boolean that enables or disables thethe PointIdent of PointAttributes.

Example

```
Dim Pattribs As PointAttributes

Set Pattribs = New PointAttributes
Pattribs.PointIdent = False
Pattribs.HasPointIdent = True
```

ZoomFactor

Returns

Double

Description

This property returns or sets the ZoomFactor of PointAttributes.

Example

```
Dim Pattribs As PointAttributes

Set Pattribs = New PointAttributes
Pattribs.ZoomFactor = 0.6
```

HasZoomFactor

Returns

Boolean

Description

This property returns or sets a boolean that enables or disables the zoomfactor of PointAttributes.

Example

```
Dim Pattribs As PointAttributes

Set Pattribs = New PointAttributes
Pattribs.ZoomFactor = 0.6
Pattribs.HasZoomFactor = TRUE
```

HasScaleFactor

Returns

Boolean

Description

This property returns or sets a boolean that enables or disables the ScaleFactor of PointAttributes.

Example

```
Dim Pattribs As PointAttributes

Set Pattribs = New PointAttributes
Pattribs.SetScaleFactor 0.5, 0.5
Pattribs.HasScaleFactor = TRUE
```

Copy

Returns

PointAttributes

Description

This property returns a copy of PointAttributes.

Example

```
Dim Pattribs As PointAttributes
Dim Pattribs2 As PointAttributes

Set Pattribs = New PointAttributes
Set Pattribs2 = Pattribs.Copy
```

Methods

GetScaleFactor

Parameters

ScaleX As Double, ScaleY As Double

Description

GetScaleFactor returns both the factors (X and Y) by which the symbol is scaled. The scale factors are expressed in %. One or both scale factors may be negative.

Example

```
Dim Pattribs As PointAttributes
Dim ScaleX as Double, ScaleY as Double

Set Pattribs = New PointAttributes
GetScaleFactor ScaleX, ScaleY
```

SetScaleFactor

Parameters

ScaleX As Double, ScaleY As Double

Description

SetScaleFactor sets the scale factors(X and Y) of the PointAttributes.

Example

```
Dim Pattribs As PointAttributes

Set Pattribs = New PointAttributes
SetScaleFactor -100.0, 100.0
```

Chapter 67 - LineAttributes

Description

LineAttributes is a collection of visual attributes for line objects, with an option to enable or disable each attribute. It can be a part of AttributesSet, but works individually as well.

Example

```
Dim Attribs as AttributesSet
Dim Lattribs as LineAttributes

Set Attribs = New AttributesSet
Set Lattribs = Attribs.LineAttributes
```

Properties

LineStyle
HasStyle
TrueColor
HasColor
Layer
HasLayer
DisplayLevel
HasDisplayLevel
Comment
HasComment
HiddenEndPoints
HasHiddenEndPoints
Arrows
HasArrows
Width
HasWidth
Scale
HasScale
Curvature
FullRounding
HasCurvatureAndFullRounding
Copy

Properties

LineStyle

Returns

Style

Description

This property returns or sets the style of LineAttributes.

Example

```
Dim Lattribs As LineAttributes
Dim Lstyle as Style

Set Lattribs = New LineAttributes
Lattribs.LineStyle = Lstyle
```

HasStyle

Returns

Boolean

Description

This property returns or sets a boolean that enables or disables the Style of LineAttributes.

Example

```
Dim Lattribs As LineAttributes
Dim Lstyle as Style

Set Lattribs = New LineAttributes
Lattribs.LineStyle = Lstyle
Lattribs.HasStyle = TRUE
```

TrueColor

Returns
RGB

Description
This property returns or sets the color (RGB) of LineAttributes.

Example

```
Dim Lattribs As LineAttributes
Dim Red as RGB

Red.Red = 255
Set Lattribs = New LineAttributes
Lattribs.TrueColor = Red
```

HasColor

Returns
Boolean

Description
This property returns or sets a boolean that enables or disables the color of LineAttributes.

Example

```
Dim Lattribs As LineAttributes
Dim Red as RGB

Red.Red = 255
Set Lattribs = New LineAttributes
Lattribs.TrueColor = Red
Lattribs.HasColor = TRUE
```

Layer

Returns
String

Description
This property returns or sets the Layer of LineAttributes.

Example

```
Dim Lattribs As LineAttributes

Set Lattribs = New LineAttributes
Lattribs.Layer = "Buildings"
```

HasLayer

Returns
Boolean

Description
This property returns or sets a boolean that enables or disables the layer of LineAttributes.

Example

```
Dim Lattribs As LineAttributes

Set Lattribs = New LineAttributes
Lattribs.Layer = "Buildings"
Lattribs.HasLayer = True
```

DisplayLevel

Returns
Integer

Description
This property returns or sets the display level of the LineAttributes.

Example

```
Dim Lattribs As LineAttributes
```

```
Set Lattribs = New LineAttributes
Lattribs.DisplayLevel = -5
Lattribs.HasDisplayLevel = True
```

HasDisplayLevel

Returns

Boolean

Description

This property returns or sets a boolean that enables or disables the display level of the LineAttributes.

Example

```
Dim Lattribs As LineAttributes

Set Lattribs = New LineAttributes
Lattribs.HasDisplayLevel = False
```

Comment

Returns

String

Description

This property returns or sets the comment (object information) of theLineAttributes.

Example

```
Dim Lattribs As LineAttributes

Set Lattribs = New LineAttributes
Lattribs.Comment = "DH"
Lattribs.HasComment = True
```

HasComment

Returns

Boolean

Description

This property returns or sets a boolean that enables or disables the comment of the LineAttributes.

Example

```
Dim Lattribs As LineAttributes

Set Lattribs = New LineAttributes
Lattribs.HasComment = False
```

HiddenEndPoints

Returns

Boolean

Description

This property returns or sets a boolean that indicates if the style of the end points of new lines that don't refer to existing points will be hidden. In other words, if the PointStyle of those points will be = ptHiddenPoint.

Example

```
Dim Lattribs As LineAttributes

Set Lattribs = New LineAttributes
Lattribs.HiddenEndPoints = True
```

HasHiddenEndPoints

Returns

Boolean

Description

This property returns or sets a boolean that enables or disables the HiddenEndpointsproperty inLineAttributes.

Example

```
Dim Lattribs As LineAttributes

Set Lattribs = New LineAttributes
Lattribs.HasComment = False
```

Arrows

Returns

Integer

Description

This property returns or sets a value of `LineAttributes` indicating which ends of a line has arrows. The arrows parameter has the following valid values: `ptNoArrows`, `ptArrowP1`, `ptArrowP2`, `ptArrowP1P2`.

Example

```
Dim Lattribs As LineAttributes

Set Lattribs = New LineAttributes
Lattribs.Arrows = ptArrowP1P2
```

HasArrows

Returns

Boolean

Description

This property returns or sets a boolean that enables or disables arrows in `LineAttributes`

Example

```
Dim Lattribs As LineAttributes

Set Lattribs = New LineAttributes
Lattribs.Arrows = ptArrowP1P2
Lattribs.HasArrows = TRUE
```

Width

Returns

Double

Description

This property sets or returns the Width of the `LineAttributes`. The line width must be between 0.1 (0.01 mm) and 25.5 (2.55 mm).

Example

```
Dim Lattribs As LineAttributes

Set Lattribs = New LineAttributes
Lattribs.Width = 5.0
```

HasWidth

Returns

Boolean

Description

This property sets or returns the boolean that enables or disables the width of `LineAttributes`.

Example

```
Dim Lattribs As LineAttributes

Set Lattribs = New LineAttributes
Lattribs.Width = 5.0
Lattribs.HasWidth = TRUE
```

Scale

Returns

Double

Description

This property sets or returns the scale of `LineAttributes`. Scaling is only applicable to scalable line styles.

Example

```
Dim Lattributes As LineAttributes

Set Lattributes = New LineAttributes
Lattributes.Scale = 120.0
```

HasScale

Returns

Boolean

Description

This property sets or returns the boolean that enables or disables the scale of `LineAttributes`.

Example

```
Dim Lattributes As LineAttributes

Set Lattributes = New LineAttributes
Lattributes.Scale = 120.0
Lattributes.HasScale = TRUE
```

Curvature (obsolete)

Returns

Integer

Description

This property returns always 0 and doesn't set the value since v16.

Example

```
Dim Lattribs As LineAttributes

Set Lattribs = New LineAttributes
Lattributes.Curvature = 6 'Does'nt do anything
```

FullRounding (obsolete)

Returns

Integer

Description

This property returns always true and doesn't set the value since v16.

Example

```
Dim Lattribs As LineAttributes

Set Lattribs = New LineAttributes
Lattributes.FullRounding = True 'Does'nt do anything
```

HasCurvatureAndFullRounding (obsolete)

Returns

Integer

Description

This property returns always false and doesn't set the value since v16.

Example

```
Dim Lattribs As LineAttributes

Set Lattribs = New LineAttributes
Lattributes.FullRounding = True 'Does'nt do anything
```

Copy

Returns

`LineAttributes`

Description

This property returns a copy of LineAttributes.

Example

```
Dim Lattribs As LineAttributes
Dim Lattribs2 As LineAttributes

Set Lattribs = New LineAttributes
Set Lattribs2 = Lattribs.Copy
```


Chapter 68 - PolygonAttributes

Description

PolygonAttributes is a collection of visual attributes for polygon objects, with an option to enable or disable each attribute. It can be a part of AttributesSet, but works individually as well.

Example

```
Dim Attribs as AttributesSet
Dim PolyAttribs as PolygonAttributes

Set Attribs = New AttributesSet
Set PolyAttribs = Attribs.PolygonAttributes
```

Properties

Pattern
HasPattern
Border
HasBorder
TrueColor
HasColor
Layer
HasLayer
DisplayLevel
HasDisplayLevel
Comment
HasComment
BGPattern
HasBGPattern
Scale
HasScale
BGColor
Opaque
HasOpaque
Copy

Properties

Pattern

Returns

Style

Description

This property returns or sets the pattern of PolygonAttributes.

Example

```
Dim PolyAttribs As PolygonAttributes
Dim PolyStyle as Style

Set PolyAttribs = New PolygonAttributes
Polyattribs.Pattern = PolyStyle
```

HasPattern

Returns

Boolean

Description

This property returns or sets a boolean that enables or disables the pattern of PolygonAttributes.

Example

```
Dim PolyAttribs As PolygonAttributes
Dim PolyStyle as Style

Set Polyattribs = New PolygonAttributes
Polyattribs.Pattern = PolyStyle
PolyAttribs.HasPattern = TRUE
```

Border

Returns

Integer

Description

This property returns or sets the border of PolygonAttributes.

ptNoBorder = The polygon is filled / will be filled normal (not only a border).

ptBorder1mm .. ptBorder10mm = The width of the border of the polygon is / will be from 1 mm up to 10 mm depending on the value of the parameter.

ptBorder1mmOpen .. ptBorder10mmOpen = The width of the border of the polygons is / will be from 1 mm up to 10 mm depending on the value of the parameter. The last leg of the polygon is left open.

Example

```
Dim PolyAttribs As PolygonAttributes

Set PolyAttribs = New PolygonAttributes
PolyAttribs.Border = ptBorder3mm
```

HasBorder

Returns

Boolean

Description

This property returns or sets a boolean that enables or disables the border of PolygonAttributes.

Example

```
Dim PolyAttribs As PolygonAttributes

Set PolyAttribs = New PolygonAttributes
PolyAttribs.Border = ptBorder3mm
PolyAttribs.HasBorder = TRUE
```

TrueColor

Returns

RGB

Description

This property returns or sets the color (RGB) of PolygonAttributes.

Example

```
Dim PolyAttribs As PolygonAttributes
Dim Red as RGB

Red.Red = 255
Set PolyAttribs = New PolygonAttributes
PolyAttribs.TrueColor = Red
```

HasColor

Returns

Boolean

Description

This property returns or sets a boolean that enables or disables the color of PolygonAttributes.

Example

```
Dim PolyAttribs As PolygonAttributes
Dim Red as RGB

Red.Red = 255
Set PolyAttribs = New PolygonAttributes
PolyAttribs.TrueColor = Red
PolyAttribs.HasColor = TRUE
```

Layer

Returns

String

Description

This property returns or sets the Layer of PolygonAttributes.

Example

```
Dim PolyAttribs As PolygonAttributes

Set PolyAttribs = New PolygonAttributes
PolyAttribs.Layer = "Buildings"
```

HasLayer

Returns

Boolean

Description

This property returns or sets a boolean that enables or disables the layer of PolygonAttributes.

Example

```
Dim PolyAttribs As PolygonAttributes

Set PolyAttribs = New PolygonAttributes
PolyAttribs.Layer = "Buildings"
PolyAttribs.HasLayer = True
```

DisplayLevel

Returns

Integer

Description

This property returns or sets the display level of the PolygonAttributes.

Example

```
Dim PolyAttribs As PolygonAttributes

Set PolyAttribs = New PolygonAttributes
PolyAttribs.DisplayLevel = -5
PolyAttribs.HasDisplayLevel = True
```

HasDisplayLevel

Returns

Boolean

Description

This property returns or sets a boolean that enables or disables the display level of the PolygonAttributes.

Example

```
Dim PolyAttribs As PolygonAttributes

Set PolyAttribs = New PolygonAttributes
PolyAttribs.HasDisplayLevel = False
```

Comment

Returns

String

Description

This property returns or sets the comment (object information) of the PolygonAttributes.

Example

```
Dim PolyAttribs As PolygonAttributes

Set PolyAttribs = New PolygonAttributes
PolyAttribs.Comment = "DH"
PolyAttribs.HasComment = True
```

HasComment

Returns

Boolean

Description

This property returns or sets a boolean that enables or disables the comment of the PolygonAttributes.

Example

```
Dim PolyAttribs As PolygonAttributes

Set PolyAttribs = New PolygonAttributes
PolyAttribs.HasComment = False
```

BGPattern

Returns

Style

Description

This property returns or sets the background pattern of PolygonAttributes.

Example

```
Dim PolyAttribs As PolygonAttributes
Dim PolyStyle as Style

Set PolyAttribs = New PolygonAttributes
PolyAttribs.BGPattern = PolyStyle
```

BGColor

Returns

RGB

Description

This property returns or sets the background color (RGB) of PolygonAttributes.

Example

```
Dim PolyAttribs As PolygonAttributes
Dim Red as RGB

Red.Red = 255
Set PolyAttribs = New PolygonAttributes
PolyAttribs.BGColor = Red
```

HasBGPattern

Returns

Boolean

Description

This property returns or sets a boolean that enables or disables the background pattern and background color of PolygonAttributes.

Example

```
Dim PolyAttribs As PolygonAttributes
Dim PolyStyle as Style

Set PolyAttribs = New PolygonAttributes
PolyAttribs.BGPattern = PolyStyle
PolyAttribs.HasBGPattern = TRUE
```

Scale

Returns

Double

Description

Sets or returns the scale of PolygonAttributes. The scale is expressed in %.

Example

```
Dim PolyAttribs As PolygonAttributes
```

```
Set PolyAttribs = New PolygonAttributes
Polyattribs.Scale = 150.0
```

HasScale

Returns
Boolean

Description
This property sets or returns the boolean that enables or disables the scale of PolygonAttributes.

Example

```
Dim PolyAttribs As PolygonAttributes

Set PolyAttribs = New PolygonAttributes
PolyAttribs.Scale = 150.0
PolyAttribs.HasScale = TRUE
```

Opaque

Returns
Boolean

Description
Sets or returns the attribute Opaque of the PolygonAttributes.

Example

```
Dim PolyAttribs As PolygonAttributes

Set PolyAttribs = New AttributesSet
PolyAttribs.Opaque = TRUE
```

HasOpaque

Returns
Boolean

Description
Sets or returns a boolean that enables or disables the attribute Opaque of the PolygonAttributes.

Example

```
Dim PolyAttribs As PolygonAttributes

Set PolyAttribs = New AttributesSet
PolyAttribs.Opaque = TRUE
PolyAttribs.HasOpaque = TRUE
```

Copy

Returns
PolygonAttributes

Description
This property returns a copy of PolygonAttributes.

Example

```
Dim PolyAttribs as PolygonAttributes
Dim PolyAttribs2 as PolygonAttributes

Set PolyAttribs = New PolygonAttributes
Set PolyAttribs2 = Polyattribs.Copy
```

Chapter 69 - SolidAttributes

Description

SolidAttributes is a collection of visual attributes for solid objects, with an option to enable or disable each attribute. It can be a part of AttributesSet, but works individually as well.

Example

```
Dim Attribs as AttributesSet
Dim SAttribs as SolidAttributes

Set Attribs = New AttributesSet
Set SAttribs = Attribs.SolidAttributes
```

Properties

TrueColor
HasColor
Layer
HasLayer
DisplayLevel
HasDisplayLevel
Comment
HasComment
Opaque
HasOpaque
Intensity
HasIntensity
Copy

Properties

TrueColor

Returns

RGB

Description

This property returns or sets the color (RGB) of SolidAttributes.

Example

```
Dim SAttribs As SolidAttributes
Dim Red as RGB

Red.Red = 255
Set SAttribs = New SolidAttributes
SAttribs.TrueColor = Red
```

HasColor

Returns

Boolean

Description

This property returns or sets a boolean that enables or disables the color of SolidAttributes.

Example

```
Dim SAttribs As SolidAttributes
Dim Red as RGB

Red.Red = 255
Set SAttribs = New SolidAttributes
SAttribs.TrueColor = Red
SAttribs.HasColor = TRUE
```

Layer

Returns

String

Description

This property returns or sets the Layer of SolidAttributes.

Example

```
Dim SAttribs As SolidAttributes

Set SAttribs = New SolidAttributes
SAttribs.Layer = "Buildings"
```

HasLayer

Returns

Boolean

Description

This property returns or sets a boolean that enables or disables the layer of SolidAttributes.

Example

```
Dim SAttribs As SolidAttributes

Set SAttribs = New SolidAttributes
SAttribs.Layer = "Buildings"
SAttribs.HasLayer = True
```

DisplayLevel

Returns

Integer

Description

This property returns or sets the display level of the SolidAttributes.

Example

```
Dim SAttribs As SolidAttributes

Set SAttribs = New SolidAttributes
SAttribs.DisplayLevel = -5
SAttribs.HasDisplayLevel = True
```

HasDisplayLevel

Returns

Boolean

Description

This property returns or sets a boolean that enables or disables the display level of the SolidAttributes.

Example

```
Dim SAttribs As SolidAttributes

Set SAttribs = New SolidAttributes
SAttribs.HasDisplayLevel = False
```

Comment

Returns

String

Description

This property returns or sets the comment (object information) of SolidAttributes.

Example

```
Dim SAttribs As SolidAttributes

Set SAttribs = New SolidAttributes
SAttribs.Comment = "DH"
SAttribs.HasComment = True
```

HasComment

Returns

Boolean

Description

This property returns or sets a boolean that enables or disables the comment of the SolidAttributes.

Example

```
Dim SAttribs As SolidAttributes

Set SAttribs = New SolidAttributes
SAttribs.HasComment = False
```

Opaque

Returns

Boolean

Description

Sets or returns the attribute Opaque of the SolidAttributes.

Example

```
Dim SAttribs As SolidAttributes

Set SAttribs = New AttributesSet
SAttribs.Opaque = TRUE
```

HasOpaque

Returns

Boolean

Description

Sets or returns a boolean that enables or disables the attribute Opaque of the SolidAttributes.

Example

```
Dim SAttribs As SolidAttributes

Set SAttribs = New AttributesSet
SAttribs.Opaque = TRUE
SAttribs.HasOpaque = TRUE
```

Intensity

Returns

Integer

Description

Sets or returns the attribute Intensity of the SolidAttributes.

The intensity must be a value between 1 and 100. Intensity values less than 100 will make the solid partially transparent.

Example

```
Dim SAttribs As SolidAttributes

Set SAttribs = New AttributesSet
SAttribs.Intensity = 50
```

HasIntensity

Returns

Boolean

Description

Sets or returns a boolean that enables or disables the attribute Intensity of the SolidAttributes.

Example

```
Dim SAttribs As SolidAttributes

Set SAttribs = New AttributesSet
```



```
SAttribs.Intensity = 50  
SAttribs.HasIntensity = TRUE
```

Copy

Returns

SolidAttributes

Description

This property returns a copy of SolidAttributes.

Example

```
Dim SAttribs as SolidAttributes  
Dim SAttribs2 as SolidAttributes  
  
Set SAttribs = New SolidAttributes  
Set SAttribs2 = SAttribs.Copy
```

Chapter 70 - TextAttributes

Description

TextAttributes is a collection of visual attributes for text objects, with an option to enable or disable each attribute. It can be a part of AttributesSet, but works individually as well.

Example

```
Dim Attribs as AttributesSet
Dim TAttribs as TextAttributes

Set Attribs = New AttributesSet
Set TAttribs = Attribs.TextAttributes
```

Properties

Content
Style
HasStyle
PointSize
HasPointSize
Underline
HasUnderline
Border
HasBorder
Weight
HasWeight
LineSpacing
HasLineSpacing
HorizontalAlignment
HasHorizontalAlignment
VerticalAlignment
HasVerticalAlignment
Orientation
HasOrientation
Opaque
HasOpaque
TrueColor
HasColor
Layer
HasLayer
DisplayLevel
HasDisplayLevel
HasFont
Copy

Methods

GetFontName
GetFont
SetFont

Properties

Content

Returns

String

Description

This property sets or returns the content of the text. The content may be an empty string.

Example

```
Dim TAttribs As TextAttributes
TAttribs.Content = "Cable 4x10"
```

Style

Returns

Integer

Description

This property returns or sets the style of the text in the `TextAttributes`.

`Style = ptUpright : Normal`

`Style = ptItalic : Italic`

Example

```
Dim TAttribs As TextAttributes
Dim Style as Integer
```

```
Set TAttribs = New TextAttributes
TAttribs.Style = ptItalic
```

HasStyle

Returns

Boolean

Description

This property returns or sets a boolean that enables or disables the style of the text in `TextAttributes`.

Example

```
Dim TAttribs As TextAttributes
Dim Tstyle as Style
```

```
Set TAttribs = New TextAttributes
TAttribs.Style = ptItalic
TAttribs.HasStyle = True
```

PointSize

Returns

Integer

Description

This property sets or returns the point size of the text. Value of `PointSize` must be between 1 and 1000.

Example

```
Dim TAttribs As TextAttributes

TAttribs.PointSize = 20
```

HasPointSize

Returns

Boolean

Description

This property returns or sets a boolean that enables or disables the pointsize property of the text.

Example

```
Dim TAttribs As TextAttributes

Set TAttribs = New TextAttributes
TAttribs.PointSize = 20
TAttribs.HasPointSize = TRUE
```

Underline

Returns

Integer

Description

This property sets or returns the underlining of the text.

`Underline = ptUnderlineNone : no underlining.`

`Underline = ptUnderlineSingle : single underlining.`

`Underline = ptUnderlineDouble : double underlining.`

`Underline = ptUnderlineDashed : underlining dashed line.`

Example

```
Dim TAttribs As TextAttributes
```

```
TAttribs.Underline = ptUnderlineNone
```

HasUnderline

Returns

Boolean

Description

This property returns or sets a boolean that enables or disables the Underline property of the text.

Example

```
Dim TAttribs As TextAttributes

TAttribs.Underline = ptUnderlineNone
TAttribs.HasUnderline = True
```

Border

Returns

Integer

Description

This property sets or returns the border of the text.

Border = ptBorderNone : no border.

Border = ptBorderSingle : single border.

Border = ptBorderDouble : double border.

Example

```
Dim TAttribs As TextAttributes

TAttribs.Border = ptBorderSingle
```

HasBorder

Returns

Boolean

Description

This property returns or sets a boolean that enables or disables the Border property of the text.

Example

```
Dim TAttribs As TextAttributes

TAttribs.Border = ptBorderSingle
TAttribs.HasBorder = True
```

Weight

Returns

Integer

Description

This property sets or returns the weight of the text.

Weight = ptLight : Light

Weight = ptMedium : Normal

Weight = ptBold : Bold

Weight = ptExtraBold : Extra Bold

Not all fonts can be represented Light or Extra Bold.

Example

```
Dim TAttribs As TextAttributes

TAttribs.Weight = ptBold
```

HasWeight

Returns

Boolean

Description

This property returns or sets a boolean that enables or disables the Weight property of the text.

Example

```
Dim TAttribs As TextAttributes

TAttribs.Weight = ptBold
TAttribs.HasWeight = True
```

LineSpacing

Returns

Integer

Description

This property sets or returns the line spacing of the text.

LineSpacing = ptSpacingSmall : small line spacing.

LineSpacing = ptSpacingNormal : normal line spacing.

LineSpacing = ptSpacingLarge : large line spacing.

Example

```
Dim TAttribs As TextAttributes

TAttribs.LineSpacing = ptSpacingNormal
```

HasLineSpacing

Returns

Boolean

Description

This property returns or sets a boolean that enables or disables the LineSpacing property of the text.

Example

```
Dim TAttribs As TextAttributes

TAttribs.LineSpacing = ptSpacingNormal
TAttribs.HasLineSpacing = True
```

HorizontalAlignment

Returns

Integer

Description

This property sets or returns the horizontal alignment of the text.

Alignment = ptAlignLeft : left aligned.

Alignment = ptAlignCenter : centered.

Alignment = ptAlignRight : right aligned.

Example

```
Dim TAttribs As TextAttributes

TAttribs.HorizontalAlignment = ptAlignRight
```

HasHorizontalAlignment

Returns

Boolean

Description

This property returns or sets a boolean that enables or disables the HorizontalAlignment property of the text.

Example

```
Dim TAttribs As TextAttributes

TAttribs.HorizontalAlignment = ptAlignCenter
TAttribs.HasHorizontalAlignment = True
```

VerticalAlignment

Returns

Integer

Description

This property sets or returns the vertical alignment of the text.

Alignment = ptAlignBottom: bottom aligned.

Alignment = ptAlignCenter: centered.

Alignment = ptAlignTop: top aligned.

Example

```
Dim TAttribs As TextAttributes

TAttribs.VerticalAlignment = ptAlignCenter
TAttribs.HasVerticalAlignment = True
```

HasVerticalAlignment

Returns

Boolean

Description

This property returns or sets a boolean that enables or disables the VerticalAlignment property of the text.

Example

```
Dim TAttribs As TextAttributes

TAttribs.VerticalAlignment = ptAlignBottom
TAttribs.HasVerticalAlignment = True
```

Orientation

Returns

Integer

Description

This property sets or returns the orientation of the text.

Orientation = ptTextHorizontal: The text will always appear horizontal independent of page orientation or angle.

Orientation = ptTextOrientationReadable: The angle of the displayed or printed texts will change if the text would come upside down. If the angle of the texts is in 2nd or 3rd quadrant, the texts are turned over 180 degrees.

Orientation = ptTextOrientationAbsolute: The texts are always displayed at the given angle.

Example

```
Dim TAttribs As TextAttributes

TAttribs.Orientation = ptTextHorizontal
TAttribs.HasOrientation = True
```

HasOrientation

Returns

Boolean

Description

This property returns or sets a boolean that enables or disables the Orientation property of the text.

Example

```
Dim TAttribs As TextAttributes

TAttribs.Orientation = ptTextHorizontal
TAttribs.HasOrientation = True
```

Opaque

Returns

Integer

Description

This property sets or returns the transparency of the text.

If = True : Opaque

False : Transparant

Example

```
Dim TAttribs As TextAttributes

TAttribs.Opaque = False
```

HasOpaque

Returns

Boolean

Description

This property returns or sets a boolean that enables or disables the Opaque property of the text.

Example

```
Dim TAttribs As TextAttributes

TAttribs.Opaque = False
TAttribs.HasOpaque = True
```

TrueColor

Returns

RGB

Description

This property returns or sets the color (RGB) of TextAttributes.

Example

```
Dim TAttribs As TextAttributes
Dim Red as RGB

Red.Red = 255
Set TAttribs = New TextAttributes
TAttribs.TrueColor = Red
```

HasColor

Returns

Boolean

Description

This property returns or sets a boolean that enables or disables the color of TextAttributes.

Example

```
Dim TAttribs As TextAttributes
Dim Red as RGB

Red.Red = 255
Set TAttribs = New TextAttributes
TAttribs.TrueColor = Red
TAttribs.HasColor = True
```

Layer

Returns

String

Description

This property returns or sets the Layer of TextAttributes.

Example

```
Dim TAttribs As TextAttributes

Set TAttribs = New TextAttributes
TAttribs.Layer = "Buildings"
```

HasLayer

Returns

Boolean

Description

This property returns or sets a boolean that enables or disables the layer of `TextAttributes`.

Example

```
Dim TAttribs As TextAttributes

Set TAttribs = New TextAttributes
TAttribs.Layer = "Buildings"
TAttribs.HasLayer = True
```

DisplayLevel

Returns

Integer

Description

This property returns or sets the display level of the `TextAttributes`.

Example

```
Dim TAttribs As TextAttributes

Set TAttribs = New TextAttributes
TAttribs.DisplayLevel = -5
TAttribs.HasDisplayLevel = True
```

HasDisplayLevel

Returns

Boolean

Description

This property returns or sets a boolean that enables or disables the display level of the `TextAttributes`.

Example

```
Dim TAttribs As TextAttributes

Set TAttribs = New TextAttributes
TAttribs.HasDisplayLevel = False
```

HasFont

Returns

Boolean

Description

This property returns or sets a boolean that enables or disables the font of the `TextAttributes`.

Example

```
Dim TAttribs As TextAttributes

Set TAttribs = New TextAttributes
TAttribs.HasFont = False
```

Copy

Returns

`TextAttributes`

Description

This property returns a copy of `TextAttributes`.

Example

```
Dim TAttribs As TextAttributes
Dim TAttribs2 As TextAttributes

Set TAttribs = New TextAttributes
Set TAttribs2 = TAttribs.Copy
```


Methods

GetFontName

Parameters

FontName As String

Description

Name of font is returned.

Example

```
Dim T As Text, FontName As String

T.GetFontName FontName
```

GetFont

Parameters

Font As Variant, Optional IsDingbat As Boolean

Description

If text attributes references Pythagoras vector font then ptPythagorasVectorFont (integer value) is returned. Otherwise name of the font is returned. Concerning IsDingbat parameter see description of Text.GetFont method.

Example

```
Dim TAttribs As TextAttributes, FontName

TAttribs.GetFont FontName
```

GetFont (obsolete format)

Parameters

FontName As String, CodePage As Integer

Description

This format remains in VBA only to execute code written for VBA version ≤ 13 . Name of font is returned. Concerning CodePage parameter see description of Text.GetFont method (obsolete format).

SetFont

Parameters

Font As Variant, Optional IsDingbat As Boolean

Description

Font should be either ptPythagorasVectorFont or a name of a font installed in Windows. The text attributes object will reference this font. Specified name of the font should contain only characters with codes < 128 (basic ASCII). Concerning IsDingbat parameter see description of Text.SetFont method.

Example

```
Dim TAttribs As TextAttributes

TAttribs.SetFont "Arial"
```

SetFont (obsolete format)

Parameters

FontName As String, CodePage As Integer

Description

This format remains in VBA only to execute code written for VBA version ≤ 13 . The text attributes object will reference a font with this name. Concerning CodePage parameter see description of Text.GetFont method (obsolete format).

Chapter 71 - ImageAttributes

Description

ImageAttributes is a collection of visual attributes for Images, with an option to enable or disable each attribute. It can be a part of AttributesSet, but works individually as well.

Example

```
Dim Attribs as AttributesSet
Dim Iattribs as ImageAttributes

Set Attribs = New AttributesSet
Set Iattribs = Attribs.ImageAttributes
```

Properties

TrueColor
HasColor
Layer
HasLayer
DisplayLevel
HasDisplayLevel
Comment
HasComment
Copy

Properties

TrueColor

Returns

RGB

Description

This property returns or sets the color (RGB) of ImageAttributes.

Example

```
Dim Iattribs As ImageAttributes
Dim Red as RGB

Red.Red = 255
Set Iattribs = New ImageAttributes
Iattribs.TrueColor = Red
```

HasColor

Returns

Boolean

Description

This property returns or sets a boolean that enables or disables the color of ImageAttributes.

Example

```
Dim Iattribs As ImageAttributes
Dim Red as RGB

Red.Red = 255
Set Iattribs = New ImageAttributes
Iattribs.TrueColor = Red
Iattribs.HasColor = TRUE
```

Layer

Returns

String

Description

This property returns or sets the Layer of ImageAttributes.

Example

```
Dim Iattribs As ImageAttributes
```

```
Set IAttribs = New ImageAttributes
IAttribs.Layer = "Buildings"
```

HasLayer

Returns
Boolean

Description
This property returns or sets a boolean that enables or disables the layer of ImageAttributes.

Example

```
Dim IAttribs As ImageAttributes

Set IAttribs = New ImageAttributes
IAttribs.Layer = "Buildings"
IAttribs.HasLayer = True
```

DisplayLevel

Returns
Integer

Description
This property returns or sets the display level of the ImageAttributes.

Example

```
Dim IAttribs As ImageAttributes

Set IAttribs = New ImageAttributes
IAttribs.DisplayLevel = -5
IAttribs.HasDisplayLevel = True
```

HasDisplayLevel

Returns
Boolean

Description
This property returns or sets a boolean that enables or disables the display level of the ImageAttributes.

Example

```
Dim IAttribs As ImageAttributes

Set IAttribs = New ImageAttributes
IAttribs.HasDisplayLevel = False
```

Comment

Returns
String

Description
This property returns or sets the comment (object information) of the ImageAttributes.

Example

```
Dim IAttribs As ImageAttributes

Set IAttribs = New ImageAttributes
IAttribs.Comment = "DH"
IAttribs.HasComment = True
```

HasComment

Returns
Boolean

Description
This property returns or sets a boolean that enables or disables the comment of the ImageAttributes.

Example

```
Dim IAttribs As ImageAttributes
```

```
Set IAttribs = New ImageAttributes
IAttribs.HasComment = False
```

Copy

Returns

ImageAttributes

Description

This property returns a copy of ImageAttributes.

Example

```
Dim IAttribs as ImageAttributes
Dim IAttribs2 as ImageAttributes

Set IAttribs = New ImageAttributes
Set IAttribs2 = IAttribs.Copy
```

Chapter 72 - AnnotationAttributes

Description

AnnotationAttributes is a collection of visual attributes for annotation objects, with an option to enable or disable each attribute. It can be a part of AttributesSet, but works individually as well.

Example

```
Dim Attribs as AttributesSet
Dim AnAttribs as AnnotationAttributes

Set Attribs = New AttributesSet
Attribs.GetAnnotationAttributes ptAn_Comment AnAttribs
```

Properties

TextContent
TextStyle
HasTextStyle
TextPointSize
HasTextPointSize
TextUnderline
HasTextUnderline
TextBorder
HasTextBorder
TextWeight
HasTextWeight
TextLineSpacing
HasTextLineSpacing
TextHorizontalAlignment
HasTextHorizontalAlignment
TextOrientation
HasTextOrientation
TextOpaque
HasTextOpaque
TextTrueColor
HasTextColor
FreezeText
HasFreezeText
HasTextFont
SymbolType
HasSymbolType
SymbolStyle
HasSymbolStyle
SymbolTrueColor
HasSymbolColor
SymbolScale
HasSymbolScale
FlipRange
HasFlipRange
PointerQuadrant
HasPointerQuadrant
Layer
HasLayer
DisplayLevel
HasDisplayLevel
UnitsAndDecimals
HasUnitsAndDecimals
UnitsCoord
HasUnitsCoord
UnitsLength
HasUnitsLength
UnitsArea
HasUnitsArea
UnitsAngle
HasUnitsAngle
DecCoord
HasDecCoord
DecLength
HasDecLength
DecArea

```
HasDecArea
DecAngle
HasDecAngle
FormatArea
HasFormatArea
Copy
```

Methods

```
GetTextFont
SetTextFont
```

Properties

TextContent

Returns

String

Description

This property sets or returns the content of the annotation. The content may be an empty string.

Example

```
Dim AnAttribs As AnnotationAttributes
AnAttribs.Content = "-^LineLength-"
```

TextStyle

Returns

Integer

Description

This property returns or sets the textstyle of the annotation in the AnnotationAttributes.

Style = ptUpright : Normal

Style = ptItalic : Italic

Example

```
Dim AnAttribs As AnnotationAttributes
Dim Style as Integer

Set AnAttribs = New AnnotationAttributes
AnAttribs.TextStyle = ptItalic
```

HasTextStyle

Returns

Boolean

Description

This property returns or sets a boolean that enables or disables the textstyle of the annotation in AnnotationAttributes.

Example

```
Dim AnAttribs As AnnotationAttributes

Set AnAttribs = New AnnotationAttributes
AnAttribs.TextStyle = ptItalic
AnAttribs.HasTextStyle = True
```

TextPointSize

Returns

Integer

Description

This property sets or returns the text point size of the annotation. Value of PointSize must be between 1 and 1000.

Example

```
Dim AnAttribs As AnnotationAttributes

AnAttribs.TextPointSize = 20
```

HasTextPointSize

Returns

Boolean

Description

This property returns or sets a boolean that enables or disables the text point size property of the annotation.

Example

```
Dim AnAttribs As AnnotationAttributes

Set AnAttribs = New AnnotationAttributes
AnAttribs.TextPointSize = 20
AnAttribs.HasTextPointSize = TRUE
```

TextUnderline

Returns

Integer

Description

This property sets or returns the underlining of the text of the annotation.

Underline = ptUnderlineNone : no underlining.

Underline = ptUnderlineSingle : single underlining.

Underline = ptUnderlineDouble : double underlining.

Underline = ptUnderlineDashed : underlining dashed line.

Example

```
Dim AnAttribs As AnnotationAttributes

AnAttribs.TextUnderline = ptUnderlineNone
```

HasTextUnderline

Returns

Boolean

Description

This property returns or sets a boolean that enables or disables the Underline property of the text of the annotation.

Example

```
Dim AnAttribs As AnnotationAttributes

AnAttribs.TextUnderline = ptUnderlineNone
AnAttribs.HasTextUnderline = True
```

TextBorder

Returns

Integer

Description

This property sets or returns the border of the text of the annotation.

Border = ptBorderNone : no border.

Border = ptBorderSingle : single border.

Border = ptBorderDouble : double border.

Example

```
Dim AnAttribs As AnnotationAttributes

AnAttribs.TextBorder = ptBorderSingle
```

HasBorder

Returns

Boolean

Description

This property returns or sets a boolean that enables or disables the Border property of the text of the annotation.

Example

```
Dim AnAttribs As AnnotationAttributes

AnAttribs.TextBorder = ptBorderSingle
AnAttribs.HasTextBorder = True
```

TextWeight

Returns

Integer

Description

This property sets or returns the weight of the text of the annotation.

Weight = ptLight : Light

Weight = ptMedium : Normal

Weight = ptBold : Bold

Weight = ptExtraBold : Extra Bold

Not all fonts can be represented Light or Extra Bold.

Example

```
Dim AnAttribs As AnnotationAttributes

AnAttribs.TextWeight = ptBold
```

HasTextWeight

Returns

Boolean

Description

This property returns or sets a boolean that enables or disables the Weight property of the text of the annotation.

Example

```
Dim AnAttribs As AnnotationAttributes

AnAttribs.TextWeight = ptBold
AnAttribs.HasTextWeight = True
```

TextLineSpacing

Returns

Integer

Description

This property sets or returns the line spacing of the text of the annotation.

LineSpacing = ptSpacingSmall : small line spacing.

LineSpacing = ptSpacingNormal : normal line spacing.

LineSpacing = ptSpacingLarge : large line spacing.

Example

```
Dim AnAttribs As AnnotationAttributes

AnAttribs.TextLineSpacing = ptSpacingNormal
```

HasTextLineSpacing

Returns

Boolean

Description

This property returns or sets a boolean that enables or disables the TextLineSpacing property of the annotation .

Example

```
Dim AnAttribs As AnnotationAttributes
```



```
AnAttribs.TextLineSpacing = ptSpacingNormal
AnAttribs.HasTextLineSpacing = True
```

TextHorizontalAlignment

Returns

Integer

Description

This property sets or returns the horizontal alignment of the text of the annotation.

Alignment = ptAlignLeft : left aligned.

Alignment = ptAlignCenter : centered.

Alignment = ptAlignRight : right aligned.

Example

```
Dim AnAttribs As AnnotationAttributes

AnAttribs.TextHorizontalAlignment = ptAlignRight
```

HasTextHorizontalAlignment

Returns

Boolean

Description

This property returns or sets a boolean that enables or disables the TextHorizontalAlignment property of the annotation.

Example

```
Dim AnAttribs As AnnotationAttributes

AnAttribs.TextHorizontalAlignment = ptAlignCenter
AnAttribs.HasTextHorizontalAlignment = True
```

TextOrientation

Returns

Integer

Description

This property sets or returns the orientation of the text of the annotation.

Orientation = ptTextHorizontal: The text will always appear horizontal independent of page orientation or angle. Pure pointer annotations are always horizontal.

Orientation = ptTextOrientationReadable: The angle of the displayed or printed texts will change if the text would come upside down. If the angle of the texts is in 2nd or 3rd quadrant, the texts are turned over 180 degrees.

Orientation = ptTextOrientationAbsolute: The texts are always displayed at the given angle.

Example

```
Dim AnAttribs As AnnotationAttributes

AnAttribs.TextOrientation = ptTextHorizontal
AnAttribs.HasTextOrientation = True
```

HasTextOrientation

Returns

Boolean

Description

This property returns or sets a boolean that enables or disables the TextOrientation property of the text.

Example

```
Dim AnAttribs As AnnotationAttributes

AnAttribs.TextOrientation = ptTextHorizontal
AnAttribs.HasTextOrientation = True
```

TextOpaque

Returns

Boolean

Description

This property sets or returns the transparency of the text of the annotation.

If = True : Opaque
False : Transparant

Example

```
Dim AnAttribs As AnnotationAttributes

AnAttribs.TextOpaque = False
```

HasTextOpaque

Returns

Boolean

Description

This property returns or sets a boolean that enables or disables the TextOpaque property of the annotation.

Example

```
Dim AnAttribs As AnnotationAttributes

AnAttribs.TextOpaque = False
AnAttribs.HasTextOpaque = True
```

TextTrueColor

Returns

RGB

Description

This property returns or sets the text color (RGB) of AnnotationAttributes.

Example

```
Dim AnAttribs As AnnotationAttributes
Dim Red as RGB

Red.Red = 255
Set AnAttribs = New AnnotationAttributes
AnAttribs.TextTrueColor = Red
```

HasTextColor

Returns

Boolean

Description

This property returns or sets a boolean that enables or disables the text color of AnnotationAttributes.

Example

```
Dim AnAttribs As AnnotationAttributes
Dim Red as RGB

Red.Red = 255
Set AnAttribs = New AnnotationAttributes
AnAttribs.TextTrueColor = Red
AnAttribs.HasTextColor = True
```

FreezeText

Returns

Boolean

Description

This property sets or returns the possibility to alter the text of the annotation.

Example

```
Dim AnAttribs As AnnotationAttributes
```

```
AnAttribs.FreezeText = False
```

HasFreezeText

Returns

Boolean

Description

This property returns or sets a boolean that enables or disables the FreezeText property of the annotation.

Example

```
Dim AnAttribs As AnnotationAttributes

AnAttribs.FreezeText = False
AnAttribs.HasFreezeText = True
```

HasTextFont

Returns

Boolean

Description

This property returns or sets a boolean that enables or disables the TextFont property of the annotation.

Example

```
Dim AnAttribs As AnnotationAttributes

AnAttribs.HasTextFont = True
```

SymbolType

Returns

Integer

Description

This property sets or returns the symbol type of the annotation.

```
SymbolType = ptPointerSymbol
SymbolType = ptRangeSymbol
```

Example

```
Dim AnAttribs As AnnotationAttributes

AnAttribs.SymbolType = ptPointerSymbol
```

HasSymbolType

Returns

Boolean

Description

This property returns or sets a boolean that enables or disables the SymbolType property of the annotation.

Example

```
Dim AnAttribs As AnnotationAttributes

AnAttribs.SymbolType = ptPointerSymbol
AnAttribs.HasSymbolType = True
```

SymbolStyle

Returns

Integer

Description

This property sets or returns the symbol style of the annotation.

```
SymbolStyle = ptRangeOnlyText: range text only.
SymbolStyle = ptExternalRange: range external.
SymbolStyle = ptFullRange: range simple.
SymbolStyle = ptFullRangeDotted: range dotted.
```

```

SymbolStyle = ptFullRangeSlashes: range with slashes.
SymbolStyle = ptFullRangeVertSlashes: range with vertical slashes.
SymbolStyle = ptFullRangeExtensions: range with extensions.
SymbolStyle = ptFullRangePullUp: range with dotted extensions.
SymbolStyle = ptIntermittedRange: intermitted range simple.
SymbolStyle = ptIntermittedRangeDotted: intermitted range dotted.
SymbolStyle = ptIntermittedRangeSlashes: intermitted range with slashes.
SymbolStyle = ptIntermittedRangeVertSlashes: intermitted range with vertical slashes.
SymbolStyle = ptIntermittedRangeExtensions: intermitted range with extensions.
SymbolStyle = ptIntermittedRangePullUp: intermitted range with dotted extensions.

SymbolStyle = ptPointerTextOnly: pointer text only.
SymbolStyle = ptPointerOnlyTextOnObject: pointer text only centered.
SymbolStyle = ptSimplePointerLeftOfText: pointer to vertical side.
SymbolStyle = ptSimplePointerUnderText: pointer to horizontal side.
SymbolStyle = ptPointerToCorner: pointer to corner.
SymbolStyle = ptPointerBroken: pointer broken.
SymbolStyle = ptPointerBrokenToVerticalSide: pointer broken to vertical side.
SymbolStyle = ptPointerWithVerticalSlash: pointer with vertical slash.
SymbolStyle = ptSimplePointerUnderlinedText: pointer with underline.
SymbolStyle = ptRadiusPointer: pointer radius.
SymbolStyle = ptCirclePointer: pointer with circle.
SymbolStyle = ptPointerSideWaysText: pointer side ways text
SymbolStyle = ptPointerManHoleBroken: pointer manhole broken
SymbolStyle = ptPointerManHoleBrokenToVerticalSide: pointer manhole broken to
vertical side

```

Example

```

Dim AnAttribs As AnnotationAttributes

AnAttribs.SymbolStyle = ptRangeOnlyText

```

HasSymbolStyle

Returns

Boolean

Description

This property returns or sets a boolean that enables or disables the SymbolStyle property of the annotation.

Example

```

Dim AnAttribs As AnnotationAttributes

AnAttribs.SymbolStyle = ptRangeOnlyText
AnAttribs.HasSymbolStyle = True

```

SymbolTrueColor

Returns

RGB

Description

This property returns or sets the symbol color (RGB) of AnnotationAttributes.

Example

```

Dim AnAttribs As AnnotationAttributes
Dim Red as RGB

Red.Red = 255
Set AnAttribs = New AnnotationAttributes
AnAttribs.SymbolTrueColor = Red

```

HasSymbolColor

Returns

Boolean

Description

This property returns or sets a boolean that enables or disables the symbol color of AnnotationAttributes.

Example

```
Dim AnAttribs As AnnotationAttributes
Dim Red as RGB

Red.Red = 255
Set AnAttribs = New AnnotationAttributes
AnAttribs.SymbolTrueColor = Red
AnAttribs.HasSymbolColor = True
```

SymbolScale

Returns

Double

Description

This property sets or returns the symbol scale of the annotation.

Example

```
Dim AnAttribs As AnnotationAttributes

AnAttribs.SymbolScale = 90.0
```

HasSymbolScale

Returns

Boolean

Description

This property returns or sets a boolean that enables or disables the SymbolScale property of the annotation.

Example

```
Dim AnAttribs As AnnotationAttributes

AnAttribs.SymbolScale = 90.0
AnAttribs.HasSymbolScale = True
```

FlipRange

Returns

Boolean

Description

This property sets or returns the FlipRange of the annotation.

Example

```
Dim AnAttribs As AnnotationAttributes

AnAttribs.FlipRange = True
```

HasFlipRange

Returns

Boolean

Description

This property returns or sets a boolean that enables or disables the FlipRange property of the annotation.

Example

```
Dim AnAttribs As AnnotationAttributes

AnAttribs.FlipRange = True
AnAttribs.HasFlipRange = True
```

PointerQuadrant

Returns

Integer

Description

This property sets or returns the PointerQuadrant of the annotation. This value has to be in the interval [1 - 4].

Example

```
Dim AnAttribs As AnnotationAttributes

AnAttribs.PointerQuadrant = 2
```

HasPointerQuadrant

Returns

Boolean

Description

This property returns or sets a boolean that enables or disables the PointerQuadrant property of the annotation.

Example

```
Dim AnAttribs As AnnotationAttributes

AnAttribs.PointerQuadrant = 2
AnAttribs.HasPointerQuadrant = True
```

Layer

Returns

String

Description

This property returns or sets the Layer of AnnotationAttributes.

Example

```
Dim AnAttribs As AnnotationAttributes

Set AnAttribs = New AnnotationAttributes
AnAttribs.Layer = "Buildings"
```

HasLayer

Returns

Boolean

Description

This property returns or sets a boolean that enables or disables the layer of AnnotationAttributes.

Example

```
Dim AnAttribs As AnnotationAttributes

Set AnAttribs = New AnnotationAttributes
AnAttribs.Layer = "Buildings"
AnAttribs.HasLayer = True
```

DisplayLevel

Returns

Integer

Description

This property returns or sets the display level of the AnnotationAttributes.

Example

```
Dim AnAttribs As AnnotationAttributes

Set AnAttribs = New AnnotationAttributes
AnAttribs.DisplayLevel = -5
AnAttribs.HasDisplayLevel = True
```

HasDisplayLevel

Returns

Boolean

Description

This property returns or sets a boolean that enables or disables the display level of the AnnotationAttributes.

Example

```
Dim AnAttribs As AnnotationAttributes

Set AnAttribs = New AnnotationAttributes
AnAttribs.HasDisplayLevel = False
```

UnitsAndDecimals

Returns

Boolean

Description

This property returns or sets if the UnitsAndDecimals are specified for the AnnotationAttributes.

Example

```
Dim AnAttribs As AnnotationAttributes

Set AnAttribs = New AnnotationAttributes
AnAttribs.UnitsAndDecimals = True
AnAttribs.HasUnitsAndDecimals = True
```

HasUnitsAndDecimals

Returns

Boolean

Description

This property returns or sets a boolean that enables or disables the UnitsAndDecimals of the AnnotationAttributes.

Example

```
Dim AnAttribs As AnnotationAttributes

Set AnAttribs = New AnnotationAttributes
AnAttribs.HasUnitsAndDecimals = False
```

UnitsCoord

Returns

Integer

Description

This property returns or sets the units of coordinates of the AnnotationAttributes.
UnitsCoord = ptMM, ptCM, ptDM, ptM, ptDAM, ptHM, ptKM, ptInch, ptMiles, ptFeet or ptUSfeet.

Example

```
Dim AnAttribs As AnnotationAttributes

Set AnAttribs = New AnnotationAttributes
AnAttribs.UnitsCoord = ptCM
AnAttribs.HasUnitsCoord = True
```

HasUnitsCoord

Returns

Boolean

Description

This property returns or sets a boolean that enables or disables the units coordinates of the AnnotationAttributes.

Example

```
Dim AnAttribs As AnnotationAttributes

Set AnAttribs = New AnnotationAttributes
```

```
AnAttribs.HasUnitsCoord = False
```

UnitsLength

Returns

Integer

Description

This property returns or sets the units of length of the AnnotationAttributes.
UnitsLength = ptMM, ptCM, ptDM, ptM, ptDAM, ptHM, ptKM, ptInch, ptMiles, ptFeet or ptUSfeet.

Example

```
Dim AnAttribs As AnnotationAttributes

Set AnAttribs = New AnnotationAttributes
AnAttribs.UnitsLength = ptCM
AnAttribs.HasUnitsLength = True
```

HasUnitsLength

Returns

Boolean

Description

This property returns or sets a boolean that enables or disables the units length of the AnnotationAttributes.

Example

```
Dim AnAttribs As AnnotationAttributes

Set AnAttribs = New AnnotationAttributes
AnAttribs.HasUnitsLength = False
```

UnitsArea

Returns

Integer

Description

This property returns or sets the units of area of the AnnotationAttributes.
UnitsArea = ptMM, ptCM, ptDM, ptM, ptDAM, ptHM, ptKM, ptInch, ptMiles, ptFeet or ptUSfeet.

Example

```
Dim AnAttribs As AnnotationAttributes

Set AnAttribs = New AnnotationAttributes
AnAttribs.UnitsArea = ptM
AnAttribs.HasUnitsArea = True
```

HasUnitsArea

Returns

Boolean

Description

This property returns or sets a boolean that enables or disables the units area of the AnnotationAttributes.

Example

```
Dim AnAttribs As AnnotationAttributes

Set AnAttribs = New AnnotationAttributes
AnAttribs.HasUnitsArea = False
```


UnitsAngle

Returns

Integer

Description

This property returns or sets the units of Angle of the AnnotationAttributes.

UnitsAngle = ptGRAD, ptDEGdec, ptDEGMinSec, ptMIL, ptSURVEY or ptRADIAN.

Example

```
Dim AnAttribs As AnnotationAttributes

Set AnAttribs = New AnnotationAttributes
AnAttribs.UnitsAngle = ptGRAD
AnAttribs.HasUnitsAngle = True
```

HasUnitsAngle

Returns

Boolean

Description

This property returns or sets a boolean that enables or disables the units angle of the AnnotationAttributes.

Example

```
Dim AnAttribs As AnnotationAttributes

Set AnAttribs = New AnnotationAttributes
AnAttribs.HasUnitsAngle = False
```

DecCoord

Returns

Integer

Description

This property returns or sets the decimals of coordinates of the AnnotationAttributes.

DecCoord =[0..5].

Example

```
Dim AnAttribs As AnnotationAttributes

Set AnAttribs = New AnnotationAttributes
AnAttribs.DecCoord = 2
AnAttribs.HasDecCoord = True
```

HasDecCoord

Returns

Boolean

Description

This property returns or sets a boolean that enables or disables the decimals of coordinates of the AnnotationAttributes.

Example

```
Dim AnAttribs As AnnotationAttributes

Set AnAttribs = New AnnotationAttributes
AnAttribs.HasDecCoord = False
```

DecLength

Returns

Integer

Description

This property returns or sets the decimals of length of the AnnotationAttributes.

```
DecLength =[0..5].
```

Example

```
Dim AnAttribs As AnnotationAttributes

Set AnAttribs = New AnnotationAttributes
AnAttribs.DecLength = 2
AnAttribs.HasDecLength = True
```

HasDecLength

Returns

Boolean

Description

This property returns or sets a boolean that enables or disables the decimals of Length of the AnnotationAttributes.

Example

```
Dim AnAttribs As AnnotationAttributes

Set AnAttribs = New AnnotationAttributes
AnAttribs.HasDecLength = False
```

DecArea

Returns

Integer

Description

This property returns or sets the decimals of area of the AnnotationAttributes.
DecArea =[0..5].

Example

```
Dim AnAttribs As AnnotationAttributes

Set AnAttribs = New AnnotationAttributes
AnAttribs.DecArea = 2
AnAttribs.HasDecArea = True
```

HasDecArea

Returns

Boolean

Description

This property returns or sets a boolean that enables or disables the decimals of area of the AnnotationAttributes.

Example

```
Dim AnAttribs As AnnotationAttributes

Set AnAttribs = New AnnotationAttributes
AnAttribs.HasDecArea = False
```

DecAngle

Returns

Integer

Description

This property returns or sets the decimals of angle of the AnnotationAttributes.
DecAngle =[0..5].

Example

```
Dim AnAttribs As AnnotationAttributes

Set AnAttribs = New AnnotationAttributes
AnAttribs.DecAngle = 2
AnAttribs.HasDecAngle = True
```

HasDecAngle

Returns

Boolean

Description

This property returns or sets a boolean that enables or disables the decimals of angle of the AnnotationAttributes.

Example

```
Dim AnAttribs As AnnotationAttributes

Set AnAttribs = New AnnotationAttributes
AnAttribs.HasDecAngle = False
```

FormatArea

Returns

Integer

Description

This property returns or sets the format of area of the AnnotationAttributes.
FormatArea = ptDefaultAreaFormat, ptAcres or ptHaAreCa.

Example

```
Dim AnAttribs As AnnotationAttributes

Set AnAttribs = New AnnotationAttributes
AnAttribs.FormatArea = ptDefaultAreaFormat
AnAttribs.HasFormatArea = True
```

HasFormatArea

Returns

Boolean

Description

This property returns or sets a boolean that enables or disables the format of area of the AnnotationAttributes.

Example

```
Dim AnAttribs As AnnotationAttributes

Set AnAttribs = New AnnotationAttributes
AnAttribs.HasFormatArea = False
```

Copy

Returns

TextAttributes

Description

This property returns a copy of AnnotationAttributes.

Example

```
Dim AnAttribs As AnnotationAttributes
Dim AnAttribs2 As AnnotationAttributes

Set AnAttribs = New TextAttributes
Set AnAttribs2 = AnAttribs.Copy
```

Methods

GetTextFont

Parameters

FontName As String, CodePage As Integer

Description

GetTextFont returns the Font name and de codepage of the text in AnnotationAttributes.

Example

```
Dim AnAttribs As AnnotationAttributes
Dim FontName As String
Dim CodePage As Integer

AnAttribs.GetTextFont FontName, CodePage
```

SetTextFont

Parameters

FontName As String, CodePage As Integer

Description

SetFont sets the Font name and de codepage of the text in AnnotationAttributes.

Example

```
Dim AnAttribs As AnnotationAttributes

AnAttribs.SetTextFont "Arial", ptWestern
```


Chapter 73 - Stack

Description

A Stack is a well known data type (LIFO). The Stack may mix variants of the following subtypes: Integer, Long, Single, Double, String, Date

Example

```
Dim S As Stack
Dim V As Variant

Set S = New Stack
S.Push 5.0
S.Push "Value : "
' ...
MsgBox S.Pop() & CStr(S.Pop())
```

Properties

NrOfItems

Methods

Push
Pop
Clear

Properties

NrOfItems

Returns

Long

Description

This property returns the number of items on the stack.

Example

```
Dim S As Stack

Set S = New Stack
S.Push "Item 1"
MsgBox CStr(S.NrOfItems)
```

Methods

Push

Parameters

Value As Variant

Description

This method will push the value on the stack. The stack may contains variants of different subtypes.

Example

```
Dim S As Stack
Dim V As Variant

Set S = New Stack
S.Push 1.25
V = "This is a string"
S.Push V
```

Pop

Return Parameter

Value As Variant

Description

This method will pop the value from the stack.

Example

```
Dim S As Stack
Dim V As Variant

V = S.Pop()
```

Clear*Description*

This method will remove all items from the stack.

Example

```
Dim S As Stack

S.Clear
```

Chapter 74 - Queue

Description

A Queue is a well known data type (FIFO). The Queue may mix variants of the following subtypes: Integer, Long, Single, Double, String, Date.

Example

```
Dim Q As Queue

Set Q = New Queue
Q.PutItem 5.0
Q.PutItem " m"
' ...
MsgBox CStr(S.GetItem()) & Q.GetItem()
```

Properties

NrOfItems

Methods

PutItem
GetItem
Clear

Properties

NrOfItems

Returns

Long

Description

This property returns the number of items in the queue.

Example

```
Dim Q As Queue

Set Q = New Queue
Q.PutItem "Item 1"
MsgBox CStr(Q.NrOfItems)
```

Methods

PutItem

Parameters

Value As Variant

Description

This method will put the value on the queue. The queue may contain variants of different subtypes.

Example

```
Dim Q As Queue
Dim V As Variant

Set Q = New Queue
Q.PutItem 1.25
V = "This is a string"
Q.PutItem V
```

GetItem

Return Parameter

Value As Variant

Description

This method will get the first value from the queue.

Example

```
Dim Q As Queue
```



```
Dim V As Variant
```

```
V = Q.GetItem()
```

Clear

Description

This method will remove all items from the queue.

Example

```
Dim Q As Queue
```

```
Q.Clear
```

Chapter 75 - Tree

Description

A Tree is the well known data structure *binary tree*.

The elements of the tree are of type `ValueIndex`, a record with the first field (Value) being a Variant and the second field (Index) a Long (see chapter: **Pythagoras Data Types**).

The field `Value` contains the key, while the `Index` may be used to point to data that is related to the key. The pair (Value, Index) must be unique. So even if you don't need an index, and you want duplicate values in the tree, you will need to make Index unique for identical Values.

The subtypes of Value may be numeric (mix of variants of the following subtypes: Integer, Long, Single and Double), String, Date, or Boolean. All elements of the tree should have the same type (numeric, String, Date or Boolean).

Example

```
Dim T As Tree
Dim V As ValueIndex

Set T = New Tree

V.Value = "Fred"
T.Insert V
V.Value = "Annie"
T.Insert V
'...
V = T.GetFirst("Annie")
V = T.GetNext
```

Properties

`NrOfItems`

Methods

`Insert`
`Delete`
`Clear`
`GetFirst`
`GetNext`

Properties

NrOfItems

Returns

Long

Description

This property returns the number of items in the Tree.

Example

```
Dim T As Tree
Dim Count As Long

Count = T.NrOfItems
```

Methods

Insert

Parameters

Value As ValueIndex

Description

This method will insert the data in the tree. The data will be sorted by value in the Value field of ValueIndex.

The first value inserted in the tree defines the sub type.

Types may not be mixed. If mixed, an error will be raised.
Duplicate keys are allowed. However the combination Value, Index must be unique.

Example

```
Dim T As Tree
Dim Data As ValueIndex

Set T = New Tree
Data.Value = "Pythagoras"
Data.Index = 1
T.Insert Data
```

Delete

Parameters

Data As ValueIndex

Description

This method will delete the data from the tree. If the data is not found, an error will be raised.

Example

```
Dim T As Tree
Dim Data As ValueIndex

Set T = New Tree
T.Insert Data
T.Delete Data
```

Clear

Description

This method removes all items from the Tree.

Example

```
Dim T As Tree

T.Clear
```

GetFirst

Parameters

Key As Variant

Return Parameter

Data As ValueIndex

Description

This method will get the first element from the tree with value $\geq$ Key. If the data is not found Empty value is returned in the Value part of ValueIndex.

Example

```
Dim T As Tree
Dim Data As ValueIndex
Dim Key As Variant

Set T = New Tree
T.Insert Data
Data = T.GetFirst(Key)
```

GetNext

Return Parameter

Data As ValueIndex

Description

This method will get the next element from the tree. If no more data, Empty value is returned in the Value part of ValueIndex.

Example

```
Dim T As Tree
Dim Data As ValueIndex
```

```
Set T = New Tree
'Insert data

Data = T.GetFirst("Jozefien")
'....
Data = T.GetNext
```

Chapter 76 - Window

Description

The Window object corresponds with the window of a Pythagoras document.

Example

```
Dim A As Application
Dim Doc As Document
Dim W As Window

Set A = New Application
A.GetActiveDocument Doc
Doc.GetWindow W
W.Maximize
```

Properties

HasHorScrollBar
HasVertScrollBar
Width
Height
Position
ScreenPosition
HorScrollThumb
VertScrollThumb

Methods

Minimize
Maximize
Restore
SetWindowSize

Properties

HasHorScrollBar

Returns

Boolean

Description

Returns True if the horizontal scroll bar is active.

Example

```
Dim W As Window

If W.HasHorScrollBar Then
    W.HorScrollThumb = 50
End If
```

HasVertScrollBar

Returns

Boolean

Description

Returns True if the vertical scroll bar is active.

Example

```
Dim W As Window

If W.HasVertScrollBar Then
    W.VertScrollThumb = 50
End If
```

Width

Returns

Long

Description

Returns the width, in pixels, of the window.

Example

```
Dim W As Window
Dim WidthInPixels As Long

WidthInPixels = W.Width
```

Height

Returns

Long

Description

Returns the height, in pixels, of the window.

Example

```
Dim W As Window
Dim HeightInPixels As Long

HeightInPixels = W.Height
```

Position

Returns

WindowCoord

Description

Returns or sets the position of the window relative to the frame window of Pythagoras.

Example

```
Dim A As Application
Dim D As Document
Dim W As Window
Dim P As WindowCoord

set A = New Application
A.GetActiveDocument D

If NOT D is nothing Then
    D.GetWindow W
    p = W.Position
    MsgBox "Position : " & str(P.x) & " , " & str(P.y)
    P.x = P.x + 24
    P.y = P.y - 30
    W.Position = P
    W.SetWindowSize W.Width\2, W.Height\2
end if
```

ScreenPosition

Returns

WindowCoord

Description

Returns the position of the window relative to the top left position of the screen.

Example

```
Dim A As Application
Dim D As Document
Dim W As Window
Dim P As WindowCoord

set A = New Application
A.GetActiveDocument D

If NOT D is nothing Then
    D.GetWindow W
    p = W.ScreenPosition
    MsgBox "Position : " & str(P.x) & " , " & str(P.y)
end if
```

HorScrollThumb

Returns

Integer

Description

Returns or sets the position of the thumbs in the horizontal scroll bar. This value must be between 0 and 100. When reading HorScrollThumb, the value can differ from the value that has been set by 1 unit.

Example

```
Dim W As Window

If W.HasHorScrollBar Then
    W.HorScrollThumb = 50
    If W.HorScrollThumb <> 50 Then
        ' This may happen. Value should be between 49 and 51
    End If
End If
```

VertScrollThumb

Returns

Integer

Description

Returns or sets the position of the thumbs in the vertical scroll bar. This value must be between 0 and 100. When reading VerScrollThumb, the value can differ from the value that has been set by 1 unit.

Example

```
Dim W As Window

If W.HasVertScrollBar Then
    W.VertScrollThumb = 50
End If
```

Methods

Minimize

Parameters

None

Description

Minimizes the window. On Windows, this method will behave identically as if the operation was done by clicking on the minimize control of the window.

Example

```
Dim W As Window

W.Minimize
```

Maximize

Parameters

None

Description

Maximizes the window. This method will behave identically as if the operation was done by clicking on the maximize control of the window.

Example

```
Dim W As Window

W.Maximize
```

Restore

Parameters

None

Description

Restores the window to the previous size.

Example

```
Dim W As Window
```

```
W.Restore
```

SetWindowSize

Parameters

Horizontal As Long, Vertical As Long

Description

Sets the size of the window. The size is expressed in pixels.

Example

```
Dim W As Window
```

```
W.SetWindowSize 500, 300
```


Chapter 77 - HTMLWindow

Description

The HTMLWindow object is a separate window that can be used to display HTML content .

Example

```
Sub PM_Test

    Dim W As HTMLWindow
    Dim width as Long
    Dim height as Long
    Dim title as String

    Set W = New HTMLWindow

    width = W.Width
    height = W.Height
    title = W.Title

    MsgBox("Make sure you have some HTML file at D:\test\vba\HTMLWindow\test.html")

    W.DisplayHTML("D:\test\vba\HTMLWindow\test.html")

    W.Show

    MsgBox("HTML window shown with default dimensions and caption.")

    W.Width = 333
    W.Height = 555
    W.Title = "Test file VBA"

    MsgBox("HTML window 333 by 555 pixels with caption: Test file VBA")

    W.Hide

    MsgBox("Hidden")

    W.Show

    MsgBox("Reshown and will stay")

    MsgBox("Title: "+W.Title)
    MsgBox("Width: "+W.Width)
    MsgBox("Height: "+W.Height)
    MsgBox("End of test")

End Sub
```

Properties

Width
Height
Title

Methods

DisplayHTML
Show
Hide

Properties

Width

Returns

Long

Description

Sets or returns the width (in pixels) of the HTML Window

Example
See higher

Height

Returns
Long

Description
Sets or returns the height (in pixels) of the HTML Window

Example
See higher

Title

Returns
String

Description
Sets or returns the title of the HTML Window

Example
See higher

Methods

DisplayHTML

Parameters
Filename As String

Description
Will invoke the HTML window for the given path to a valid HTML file.

Example
See higher.

Show

Parameters
None

Description
Shows the HTML window. This method has no effect without a call to DisplayHTML.

Example
See higher.

Hide

Parameters
None

Description
Hides the HTML window. This method has no effect without a call to DisplayHTML.

Example
See higher.

Chapter 78 - PathElement

Description

A PathElement object is a generic object type that encompasses the element types PathLine, PathArc and PathClothoid. PathElement objects are used for two purposes. First they are intended to represent elements of a road center line in a convenient manner. And secondly each element of a Pathor Polygon can be represented as a PathElement and used in the GeoPath or GeoPolygon class. With this class you can do calculations on paths and polygons. Only circle polygon are not supported in this way. The PathElement object is three dimensional.

When the type of a PathElement is known, the PathElement can be converted to its specific element type. The properties Line, Arc and Clothoid return the object of a specific type.

Example

See example for the Road method GetCenterLine.

Properties

ElementType
Sequence
Length
BeginPoint
EndPoint
BeginPoint3D
EndPoint3D
Line
Arc
Clothoid

Properties

ElementType

Returns

Long

Description

This property returns the specific type of the element. Valid return values : ptLine, ptArc, ptClothoid.

Example

```
Dim Elem As PathElement
Dim L As PathLine

If Elem.ElementType = ptLine Then
    L = Elem.Line
End If
```

Sequence

Returns

Long

Description

This property returns the path sequence of begin and end points of the element. Valid return values : ptP1P2, ptP2P1.

Example

```
Dim Elem As PathElement
Dim Seq As Long
Dim S As String

If Elem.ElementType = ptArc Then
    If Elem.Sequence = ptP1P2 Then
        S = "forward"
    Else
        S = "backward"
    End If
    MsgBox "Arc in "& S &" direction"
```

```
End If
```

Length

Returns

Double

Description

This property returns the length of the path element.

Example

```
Dim Elem As PathElement

If Elem.Length > 100.0 Then
    . . .
End If
```

BeginPoint

Returns

XY

Description

This property returns the first point of the path element.

Example

```
Dim Elem As PathElement
Dim P1 As XY

P1 = Elem.BeginPoint
```

EndPoint

Returns

XY

Description

This property returns the second point of the path element.

Example

```
Dim Elem As PathElement
Dim P1 As XY

P1 = Elem.EndPoint
```

BeginPoint3D

Returns

XYZ

Description

This property returns the first point of the path element.

Example

```
Dim Elem As PathElement
Dim P1 As XYZ

P1 = Elem.BeginPoint3D
```

EndPoint3D

Returns

XYZ

Description

This property returns the second point of the path element.

Example

```
Dim Elem As PathElement
Dim P1 As XYZ

P1 = Elem.EndPoint
```

Line

Returns

PathLine

Description

Returns the reference to the PathLine object, if this object has the type ptLine, otherwise returns Nothing.

Example

```
Dim I As PathElement
Dim L As PathLine

Set L = I.Line
```

Arc

Returns

PathArc

Description

Returns the reference to the PathArc object, if this object has the type ptArc, otherwise returns Nothing.

Example

```
Dim E As PathElement
Dim A As PathArc

Set A = E.Arc
```

Clothoid

Returns

PathClothoid

Description

Returns the reference to the PathClothoid object, if this object has the type ptClothoid, otherwise returns Nothing.

Example

```
Dim E As PathElement
Dim C As PathClothoid

Set C = E.Clothoid
```

Chapter 79 - PathLine

Description

A PathLine is a PathElement. A new PathLine object can be created with the PathElement property Line.

Example

```
Dim P As PathElement
Dim PL As PathLine

...
If P.ElementType = ptLine Then
    Set PL = P.Line
    MsgBox "Begin point is (" & CStr(PL.BeginPoint.x) & _
        ", " & CStr(PL.BeginPoint.y) & ") "
End If
```

Properties

Sequence
Length
BeginPoint
EndPoint

Method

GetPathElement

Properties

For a description of the properties Sequence, Length, BeginPoint, EndPoint see object type PathElement.

Methods

GetPathElement

Returns

PathElement

Description

This method returns the PathElement for this PathLine.

Example

```
Dim MyLine As PathLien
Dim MyPE as PathElement

MyLine.GetPathElement MyPE
```


Chapter 80 - PathArc

Description

A PathArc is a PathElement. A new PathArc object can be created with the PathElement property Arc.

Example

```
Dim P As PathElement
Dim PA As PathArc

...
If P.ElementType = ptArc Then
    Set PA = P.Arc
    MsgBox "Center is (" & CStr(PA.Center.x) & _
        ", " & CStr(PA.Center.y) & ") "
End If
```

Properties

Sequence
Length
BeginPoint
EndPoint
BeginPoint3D
EndPoint3D
Radius
Center
Center3D

Method

GetPathElement

Properties

For a description of the properties Sequence, Length, BeginPoint, EndPoint, BeginPoint3D, EndPoint3D see object type PathElement.

Radius

Returns

Double

Description

This property returns the radius of the arc in radians.

Example

```
Dim MyArc As PathArc
Dim Radius As Double

Radius = MyArc.Radius
```

Center

Returns

XY

Description

This property returns the coordinate of the center of the arc.

Example

```
Dim MyArc As PathArc
Dim C As XY

C = MyArc.Center
```

Center3D

Returns

XYZ

Description

This property returns the coordinate of the center of the arc.

Example

```
Dim MyArc As PathArc
Dim C As XYZ

C = MyArc.Center3D
```

Methods

GetPathElement

Returns

PathElement

Description

This method returns the PathElement for this PathArc.

Example

```
Dim MyArc As PathArc
Dim MyPE as PathElement

MyArc.GetPathElement MyPE
```

Chapter 81 - PathClothoid

Description

A PathClothoid is a PathElement. A new PathClothoid object can be created with the PathElement property Clothoid.

Example

```
Dim P As PathElement
Dim PC As PathClothoid

...
If P.ElementType = ptClothoid Then
    Set PC = P.Clothoid
    MsgBox "Center is (" & CStr(PC.Center.x) & _
        ", " & CStr(PC.Center.y) & ") "
End If
```

Properties

Sequence
Length
BeginPoint
BeginPoint3D
EndPoint
EndPoint3D
ClothoidConstant
StartRadius
EndRadius
StartAngle
StartCenter
EndCenter
Mirror

Method

GetPathElement

Properties

For a description of the properties Sequence, Length, BeginPoint, EndPoint, BeginPoint3D, EndPoint3D see object type PathElement.

ClothoidConstant

Returns

Double

Description

This property returns the rate of change of curvature also called the K-factor of the spiral curve (clothoid).

Example

```
Dim PC As PathClothoid
Dim K As Double

K = PC.ClothoidConstant
```

StartRadius

Returns

Double

Description

This property returns the radius at the starting point of the clothoid. If the radius is infinite, ptMaxDouble is returned.

Example

```
Dim PC As PathClothoid
Dim Radius1 As Double, Radius2 As Double
```

```
Radius1 = PC.StartRadius  
Radius2 = PC.EndRadius
```

EndRadius

Returns
Double

Description
This property returns the radius at the end point of the clothoid. If the radius is infinite, ptMaxDouble is returned.

Example
Dim PC As PathClothoid
Dim Radius1 As Double, Radius2 As Double

Radius1 = PC.StartRadius
Radius2 = PC.EndRadius

StartAngle

Returns
Double

Description
This property returns the angle of the tangent line at the start point of the clothoid.

Example
Dim PC As PathClothoid
Dim Fi As Double, Radius2 As Double

Fi = PC.StartAngle

StartCenter

Returns
XY

Description
This property returns the coordinate of the center of the clothoid at the starting point.

Example
Dim PC As PathClothoid
Dim Center1 As XY, Center2 As XY

Center1 = PC.StartCenter
Center2 = PC.EndCenter

EndCenter

Returns
XY

Description
This property returns the coordinate of the center of the clothoid at the end point .

Example
Dim PC As PathClothoid
Dim Center1 As XY, Center2 As XY

Center1 = PC.StartCenter
Center2 = PC.EndCenter

Mirror

Returns
Boolean

Description
This property returns the direction of the clothoid. If False the clothoid is in counter-clockwise direction (P1 - P2). If True clockwise.

Example

```
Dim PC As PathClothoid
Dim Direction As Boolean

Direction = PC.Mirror
```

Methods

GetPathElement

Returns

PathElement

Description

This method returns the PathElement for this PathClothoid.

Example

```
Dim MyCl As PathClothoid
Dim MyPE as PathElement

MyCl.GetPathElement MyPE
```

Chapter 82 - CurveSegment

Description

A `CurveSegment` is a generic object type that encompasses the object types `CurveLine`, `CurveArc` and `CurveBezier`. It is intended to represent a single segment on a curve.

If the segment type is known you can use `GetCurveLine`, `GetCurveArc`, `GetCurveBezier` to convert to its specific segment type. These do not create a new object, but they assign the reference to the segment of a specific type.

Example

```
Dim App As Application
Dim Doc As Document
Dim Sel As Selection
Dim Obj As CadObject
Dim Cu As Curve
Dim Segments() As CurveSegment
Dim NrofSegments As Integer
Dim i As Integer
Dim CurveL As CurveLine
Dim CurveA As CurveArc
Dim CurveB As CurveBezier

Set App = New application
App.GetActiveDocument Doc
Doc.GetSelection Sel
If Sel.Count(ptCurve) > 0 Then
    Sel.GetFirst ptCurve, Obj
    Obj.GetCurve Cu
    NrofSegments = Cu.NrofSegments()
    redim Segments(NrofSegments)
    Cu.GetCurveSegments Segments
    For i = 1 to NrofSegments
        Select Case Segments(i).SegmentType
            Case ptCurveLine
                Segments(i).GetCurveLine CurveL
            Case ptCurveArc
                Segments(i).GetCurveArc CurveA
            Case ptCurveBezier
                Segments(i).GetCurveBezier CurveB
        End select
    Next i
End If
```

Properties

SegmentType
P1
P2
Length
SlopeLength

Methods

MoveP1
MoveP2
GetCurveLine
GetCurveArc
GetCurveBezier

Properties

SegmentType

Returns

Long

Description

This property returns the specific type of the segment. Valid return values : prCurveLine, ptCurveArc, ptCurveBezier

Example

```
Dim Cu As Curve
Dim CurveL As CurveLine
Dim CurveA As CurveArc
Dim CurveB As CurveBezier

Select Case Cu.SegmentType
Case ptCurveLine
    Cu.GetCurveLine CurveL
Case ptCurveArc
    Cu.GetCurveArc CurveA
Case ptCurveBezier
    Cu.GetCurveBezier CurveB
End select
```

P1 & P2

Returns

XYZ

Description

This property returns the position of P1 or P2 on the segment

Example

```
Dim Cu As Curve
Dim Pos1 As XYZ
Dim Pos2 As XYZ

Pos1 = Cu.P1
Pos2 = Cu.P2
```

Length

Returns

Double

Description

This property returns the length of the segment.

Example

```
Dim Cu As Curve
Dim L As Double

L = Cu.Length
```

SlopeLength

Returns

Double

Description

This property returns the slopelenhth of the segment.

Example

```
Dim Cu As Curve
Dim L As Double

L = Cu.SlopeLength
```

Methods

MoveP1 & MoveP2

Parameters

Pos As XYZ

Description

Moves P1 or P2 of the `CurveSegment` to the new position and redraws .

Example

```
Dim CuSegment As CurveSegment
Dim NewPos1 As XYZ
Dim NewPos2 As XYZ

CuSegment.MoveP1 NewPos1
CuSegment.MoveP2 NewPos2
```

GetCurveLine

Parameters

CurveL As CurveLine

Description

`GetCurveLine` sets the reference to the `CurveSegment` in a variable of type `CurveLine`. Note that the reference is copied, not the object.

Example

```
Dim CuSegment As CurveSegment
Dim CurveL As CurveLine

CuSegment.GetCurveLine CurveL
```

GetCurveArc

Parameters

CurveA As CurveArc

Description

`GetCurveArc` sets the reference to the `CurveSegment` in a variable of type `CurveArc`. Note that the reference is copied, not the object.

Example

```
Dim CuSegment As CurveSegment
Dim CurveA As CurveArc

CuSegment.GetCurveArc CurveA
```

GetCurveBezier

Parameters

CurveB As CurveBezier

Description

`GetCurveBezier` sets the reference to the `CurveSegment` in a variable of type `CurveBezier`. Note that the reference is copied, not the object.

Example

```
Dim CuSegment As CurveSegment
Dim CurveB As CurveBezier
```

`CuSegment.GetCurveBezier CurveB`

Chapter 83 - CurveLine

Description

A `CurveLine` is a specific type of `CurveSegment` intended to represent a single line segment on a curve.

Example

```
Dim Segment As CurveSegment
Dim CurveL As CurveLine

If Segment.SegmentType = ptCurveLine Then
    Segment.GetCurveLine CurveL
End If
```

Properties

`SegmentType`
`P1`
`P2`
`Length`
`SlopeLength`

Methods

`MoveP1`
`MoveP2`

Properties

SegmentType

See object `CurveSegment`

P1

See object `CurveSegment`

P2

See object `CurveSegment`

Length

See object `CurveSegment`

SlopeLength

See object `CurveSegment`

Methods

MoveP1

See object `CurveSegment`

MoveP2

See object `CurveSegment`

Chapter 84 - CurveArc

Description

A `CurveArc` is a specific type of `CurveSegment` intended to represent a single arc segment on a curve.

Example

```
Dim Segment As CurveSegment
Dim CurveA As CurveArc

If Segment.SegmentType = ptCurveArc Then
    Segment.GetCurveArc CurveA
End If
```

Properties

`SegmentType`
`P1`
`P2`
`ArcP`
`Length`
`SlopeLength`

Methods

`MoveP1`
`MoveP2`
`MoveArcP`

Properties

SegmentType

See object `CurveSegment`

P1

See object `CurveSegment`

P2

See object `CurveSegment`

ArcP

Returns

XYZ

Description

This property returns the position of P1 or P2 on the segment

Example

```
Dim CurveA As CurveArc
Dim Pos As XYZ

Pos = CurveA.ArcP
```

Length

See object `CurveSegment`

SlopeLength

See object `CurveSegment`

Methods

MoveP1

See object CurveSegment

MoveP2

See object CurveSegment

MoveArcP

Parameters

Pos as XYZ

Description

Moves ArcP of the CurveSegment to the new position and redraws .

Example

```
Dim CurveA As CurveArc
Dim NewPos As XYZ

CuSegment.MoveArcP NewPos
```

Chapter 85 - CurveBezier

Description

A `CurveBezier` is a specific type of `CurveSegment` intended to represent a single bezier segment on a curve.

Example

```
Dim Segment As CurveSegment
Dim CurveB As CurveBezier

If Segment.SegmentType = ptCurveBezier Then
    Segment.GetCurveBezier CurveB
End If
```

Properties

`SegmentType`
`P1`
`P2`
`Length`
`SlopeLength`

Methods

`MoveP1`
`MoveP2`

Properties

SegmentType

See object `CurveSegment`

P1

See object `CurveSegment`

P2

See object `CurveSegment`

Length

See object `CurveSegment`

SlopeLength

See object `CurveSegment`

Methods

MoveP1

See object `CurveSegment`

MoveP2

See object `CurveSegment`

Chapter 86 - GeoMath

Description

A GeoMath object is a collection of properties and methods that allow: (1) to get information about geometric objects, (2) do calculations with geometric objects and (3) get relations between them.

Example

```
Dim GM As GeoMath

Set GM = New GeoMath
```

Methods

```
Distance, dS
dZ
Projection
Intersections
CalculateIntersectionPoints
CalculateCenterPoints
CalculateCenterArc
NewPoint
NewLine
NewArc
NewCircle
NewCircle3P
NewPolygon
NewPolygonByPathElements
NewPath
NewCurve
NewClothoid
NewPathLine
NewPathArc
NewPathClothoid
```

Methods

Distance (or dS)

Parameters

```
O1 As GeoObject, O2 As GeoObject
```

Returns

```
Double
```

Description

This method returns the 2D distance between two GeoObjects (as if the objects have $Z = 0.0$).

- Points: distance between both points.
- Point and Line: the perpendicular distance between the Point and the Line. The line is considered to have indefinite length.
- Point and Arc: the distance between Point and Arc is the same as between Point and Circle with same center and radius.
- Point and Circle: if the Point is outside the circle, the distance is positive, if inside, the distance is negative.
- Point and Polygon: if the Point is outside the polygon, the distance is positive, if inside, the distance is negative. If the polygon contains holes and the point is in a hole, the point is outside, and as a consequence is positive
- Point and Curve: the closest distance to the Curve. If there is no projection on the Curve, the distance is the closest distance to start or end vertex of the Curve.
- Line and Line: is zero if lines are not parallel, else the distance between both lines.
- Line and Arc: the distance between Line and Arc is the same as between Line and Circle with same center and radius.
- Line and Circle: is zero if line and circle intersect, else the shortest distance between the line and the circle.
- Line and Polygon: if the line intersects the polygon, the distance is zero, else it is the shortest distance of a polygon vertex and the Line.
- Arc and Arc: same as Circle - Circle
- Arc and Circle: same as Circle - Circle

- Circle and Circle: is zero if both circles intersect. If one circle lies in the other, the distance is negative.
 - Arc and Polygon: same as Circle - Polygon
 - Circle and Polygon: if the circle intersects the polygon, the distance is zero, else it is the shortest distance between the circle and the polygon. If the circle is inside the polygon, or the polygon lies in the circle, the distance is negative.
 - Polygon and Polygon: the shortest distance between both polygons. If both polygons intersect the distance is zero. If one polygon is inside the other, the distance is negative.
- All other combinations result in an exception.

Example

```
Dim GM As GeoMath
Dim O1 As GeoObject, O2 As GeoObject
Dim D As Double

D = GM.dS(O1, O2)
```

dZ NOT IMPLEMENTED

Parameters

O1 As GeoObject, O2 As GeoObject

Returns

Double

Description

This method returns the difference in height between two GeoObjects. The value is always positive.

- Points: height difference between both points.
 - Point and Line / Arc / Circle: height difference between point and its projection point (see method Projection).
 - Point and Curve / Polygon: zero if Curve or Polygon are not horizontal. Else the difference in height.
 - Line and Line: if parallel 2D (not 3D): zero, if both lines are parallel (3D): the difference in height, else the dZ of the points on the line that have the same XY as the intersection.
 - Line and Arc: the dZ between Line and Arc is the same as between Line and Circle with same center and radius.
 - Line and Circle: is zero if line is not horizontal or has same elevation as circle. If line is horizontal, the difference in height.
 - Line and Polygon: the difference in height if both Line and Polygon are horizontal. Else zero.
 - Arc / Circle / Polygon / Curve and Arc / Circle / Polygon / Curve: difference in height if objects are horizontal. Else zero.
- All other combinations result in an exception.

Example

```
Dim GM As GeoMath
Dim O1 As GeoObject, O2 As GeoObject
Dim dZ As Double

dZ = GM.dZ(O1, O2)
```

Projection

Parameters

OfObj As GeoObject, OnObj As GeoObject, OnObject As Boolean

Returns

XYZ

Description

This method returns the coordinates of the projection of a point on an object.

Example

```
Dim GM As GeoMath
Dim O1 as GeoObject, O2 as GeoObject
Dim OnObject As Boolean
Dim Pnt As XYZ

Pnt = GM.Projection(O1, O2, OnObject)
```

Intersections

Parameters

O1 As GeoObject, O2 As GeoObject, NrofPoints As Long, Pnts() As XY

Description

This method returns the intersections of two objects. The intersection points will be listed in `Pnts` starting from index 0.

If both objects don't intersect `NrofPoints` will be zero.

For `GeoPath` only the intersection with a `GeoPoint` is supported. When the combination of the objects is not supported an `InvalidParameter` error will be thrown.

Example

```
Dim GM As GeoMath
Dim O1 As GeoObject, O2 As GeoObject
Dim NrofPoints As Long
Dim Pnts() As XY

GM.Intersections O1, O2, NrofPoints, Pnts
```

CalculateIntersectionPoints

Parameters

P1 As GeoPoint, R1 As Double, P2 As GeoPoint, R2 As Double,
NrofIntersections : Integer, I1 As GeoPoint, I2 As GeoPoint

Description

This method returns the intersection of the circles formed by P1, R1 and P2, R2.

The point I1 will be left of the line P1-P2, the point I2 will be right of the line.

There is one solution if $R1 + R2 = \text{Distance}(P1, P2)$.

Example

```
Dim GM As GeoMath
Dim P1 As GeoPoint, P2 As GeoPoint
Dim NrofIntersections As Integer
Dim I1 As GeoPoint, I2 As GeoPoint

GM.CalculateIntersectionPoints P1, R1, P2, R2, NrofIntersections, I1, I2
```

CalculateCenterArc

Parameters

P1 As GeoPoint, P2 As GeoPoint, R As Double, C As GeoPoint

Description

This method calculates the center point of an arc (or circle) defined by two points P1 and P2 and the radius.

If $R < 0.0$, C will be left of the line P1-P2, else C will be right of the line.

The result is identical to the intersection of the circles formed by P1, R and P2, R.

If there is no solution, the error 1577 will be raised.

Example

```
Dim GM As GeoMath
Dim P1 As GeoPoint, P2 As GeoPoint
Dim R As Double
Dim C As GeoPoint

P1.X = 10.0
P1.Y = 10.0
P2.X = 20.0
P2.Y = 20.0

GM.CalculateCenterArc P1, P2, -10.0, C ' C = 10.0, 20.0
```

NewPoint

Parameters

Pnt As XYZ

Returns

GeoPoint

Description

This method creates a `GeoPoint`.

Example

```
Dim GM As GeoMath
Dim PO As GeoPoint
Dim Pnt As XYZ

Set PO = GM.NewPoint(Pnt)
' alternative
Set PO = New GeoPoint
PO.XYZ = Pnt
```

NewLine

Parameters

Pnt1 As XYZ, Pnt2 As XYZ

Returns

`GeoLine`

Description

This method creates a `GeoLine`.

Example

```
Dim GM As GeoMath
Dim L As GeoLine
Dim Pnt1 As XYZ, Pnt2 As XYZ

Set L = GM.NewLine(Pnt1, Pnt2)
```

NewArc

Parameters

C As XYZ, Pnt1 As XYZ, Pnt2 As XYZ

Returns

`GeoArc`

Description

This method creates a `GeoArc`. If the arc is not horizontal, the object is spatially an ellipse. In plane view the object is an arc.

Meaning of parameters:

C: center of the arc,

Pnt1: starting point of the arc. The radius = `Distance(C, Pnt1)`.

Pnt2: end point of the arc.

If `Distance(C, Pnt2) <> radius`, internally Pnt2 will be adapted.

The arc is counter-clockwise from Pnt1-Pnt2. The elevation of the arc is defined by the elevation of points Pnt1 and Pnt2 and taking into account that the object spatially is an ellipse.

Example

```
Dim GM As GeoMath
Dim A As GeoArc
Dim C As XYZ, Pnt1 As XYZ, Pnt2 As XYZ

Set A = GM.NewArc(C, Pnt1, Pnt2)
```

NewCircle

Parameters

C As XYZ, Pnt As XYZ

Returns

`GeoCircle`

Description

This method creates a `GeoCircle`. If the circle is not horizontal, the object is spatially an ellipse. In plane view the object is a circle.

C: center of the circle,
Pnt: point on the circle,

Example

```
Dim GM As GeoMath
Dim C As GeoCircle
Dim Center As XYZ, Pnt As XYZ

Set C = GM.NewCircle(C, Pnt)
```

NewCircle3P

Parameters

Pnt1 As XYZ, Pnt2 As XYZ, Pnt3 As XYZ

Returns

GeoCircle

Description

This method creates a `GeoCircle`. If the circle is not horizontal, the object is spatially an ellipse. In plane view the object is a circle.

Pnt1: first point on the circle,
Pnt2: second point on the circle,
Pnt3: third point on the circle

Example

```
Dim GM As GeoMath
Dim C As GeoCircle
Dim Pnt1 As XYZ, Pnt2 As XYZ, Pnt3 As XYZ

Set C = GM.NewCircle3P(Pnt1, Pnt2, Pnt3)
```

NewPolygon

Parameters

Pnts() As XYZ

Returns

GeoPolygon

Description

This method creates a `GeoPolygon`. The number of points in the polygon is defined by the dimension of `Pnts`. The minimum dimension is 3 (triangle).

The first point and the last point of the polygon may be the same. In this case the polygon will be considered as having one point less.

Example

```
Dim GM As GeoMath
Dim Poly As GeoPolygon
Dim Pnts() As XYZ

Redim Pnts(1 To 3)
Pnts(1)
Set GM = New GeoMath
Set Poly = GM.NewPolygon(Pnts)
```

NewPolygonByPathElements

Parameters

Elements() As PathElement

Returns

GeoPolygon

Description

This method creates a `GeoPolygon`. The number of path elements in the polygon is defined by the dimension of `Elements`. The path elements should be connected: the last point of a path element should be equal to the first element of the next path element. The last point of the last path element must be equal to the first point of the first path element to close the polygon correctly. You can use the methods `NewPathLine`, `NewPathArc` and `NewPathClothoid` to fill the array of `PathElement`.

Example

```
Dim GM As GeoMath
Dim Poly As GeoPolygon
Dim Elements() As PathElement

Redim Elements(1 To 3)
Elements(1).
Set GM = New GeoMath
Set Poly = GM.NewPolygonByPathElements(Elements)
```

NewPath

Parameters

Elements() As PathElement

Returns

GeoPath

Description

This method creates a GeoPath. The number of path elements in the path is defined by the dimension of Elements.

The path elements should be connected: the last point of a path element should be equal to the first element of the next path element.

Example

```
Dim GM As GeoMath
Dim Path As GeoPath
Dim Elements () As PathElement

PathElement.
Set Path = GM.NewPath(Elements)
```

NewCurve

Parameters

Pnts() As XYZ, Optional Curvature As Integer (0), Optional MaxRounding As Boolean (TRUE)

Returns

GeoCurve

Description

This method creates a GeoCurve. The number of points of the Curve is defined by the dimension of Pnts. The Curvature is from the v15 Cuve object and has range [0, 10]. A curvature of 0, makes line segments between the point, 5 makes the standard curve (Bezier) segments. The other values also make curve (Bezier) segments but with a different curvature. Since v16 is the MaxRounding always true.

Example

```
Dim GM As GeoMath
Dim S As GeoCurve
Dim Pnts() As XYZ

Set S = GM.NewCurve(Pnts)
```

NewClothoid

Parameters

P1 As XYZ, P2 As XYZ,
RStart As Double, REnd As Double, K As Double,
Mirror As Boolean

Returns

GeoClothoid

Description

This method creates a GeoClothoid.

P1 : Starting point

P2 : End point

RStart : Radius at starting point

REnd : Radius at end point

K : K-Factor of the clothoid

Mirror : If False the clothoid is in counter-clockwise direction (P1 - P2). If True clockwise.

If the radius of the starting or end point is infinite, the value ptMaxDouble must be passed.

Note : The parameters must be specified very accurately. If the resulting clothoid can not be calculated, run-time error 1056 will be generated.

Example

```
Dim GM As GeoMath
Dim Cl As GeoClothoid
Dim P1 As XYZ, P2 As XYZ

P1.x = 0.0
P1.y = 0.0
P1.z = 0.0
P2.x = 63.348
P2.y = 6.777
P2.z = 0.0
Set Cl = GM.NewClothoid P1, P2, ptMaxDouble, 100.0, 80.0, False
```

NewPathLine

Parameters

P1 As XYZ, P2 As XYZ

Returns

PathLine

Description

This method creates a PathElement of the type PathLine.

Example

```
Dim GM As GeoMath
Dim MyPathLine As PathLine
Dim P1 as XYZ, P2 as XYZ

Set GM = New GeoMath
P1.x = 0.0
P1.y = 0.0
P1.z = 0.0
P2.x = 10.0
P2.y = 15.0
P2.z = 0.0
Set MyPathLine = GM.NewPathLine (P1, P2)
```

NewPathArc

Parameters

P1 As XYZ, P2 As XYZ, Center As XYZ
Sequence As Long

Returns

PathArc

Description

This method creates a PathElement of the type PathArc.

Example

```
Dim GM As GeoMath
Dim MyPathArc As PathArc
Dim P1 as XYZ, P2 as XYZ, Center As XYZ
Dim Sequence as Long

Set GM = New GeoMath
Center.x = 0.0
Center.y = 0.0
Center.z = 0.0
P1.x = 10.0
P1.y = 0.0
P1.z = 0.0
```

```

P2.x = 0.0
P2.y = 10.0
P2.z = 0.0
Sequence = ptP1P2
Set MyPathArc = GM.NewPathArc (Center, P1, P2, Sequence)

```

NewPathClothoid

Parameters

```

P1 As XYZ, P2 As XYZ,
RStart as Double, REnd as Double, K as Double
Mirror as Boolean,
Sequence As Long

```

Returns

```

PathClothoid

```

Description

This method creates a PathElement of the type PathClothoid.

Example

```

Dim GM As GeoMath
Dim MyPathClothoid As PathClothoid
Dim P1 as XYZ, P2 as XYZ
Dim Sequence as Long

Set GM = New GeoMath
P1.x = 0.0
P1.y = 0.0
P1.z = 0.0
P2.x = 10.0
P2.y = 15.0
P2.z = 0.0
Sequence = ptP1P2
Set MyPathArc = GM.NewPathClothoid (P1, P2, ptMaxDouble, 100.0, 80.0, False,
Sequence)

```

Chapter 87 - GeoObject

Description

A GeoObject is a generic object type that encompasses the object types `PointObject`, `LineObject`, `ArcObject`, `CircleObject`, `CurveObject`, `ClothoidObject`, and `PolygonObject`.

Properties

`Horizontal`
`Z`
`ObjectType`
`HasColor`
`GetColor`

Methods

`GetPoint`
`GetLine`
`GetArc`
`GetCircle`
`GetClothoid`
`GetCurve`
`GetPolygon`
`GetPath`
`Copy`

Properties

Horizontal

Returns

`Boolean`

Description

This property returns `True` if all points of the `GeoObject` have the same elevation. This property returns always true for a point.

Example

```
Dim GM As GeoMath
Dim O1 As GeoObject, O2 As GeoObject
Dim Dz As Double

If O1.Horizontal And O2.Horizontal Then
    Dz = O1.Z - O2.Z
End If
```

Z

Returns

`Double`

Description

This method returns the height of an horizontal object. If the object is not horizontal, the method raises an exception.

Example

```
Dim GM As GeoMath
Dim O As GeoObject
Dim Z As Double

Z = O.Z
```

ObjectType

Returns

`Integer`

Description

This property returns the specific type of the `GeoObject`. Valid return values : `ptPoint`, `ptLine`, `ptArc`, `ptCircle`, `ptCurve`, `ptPolygon`, `ptClothoid`.

Example

```
Dim GM As GeoMath
Dim O As GeoObject

If O.Type = ptLine Then
End If
```

HasColor

Returns

Boolean

Description

This property returns `TRUE` if the `GeoObject` has a color defined.

Example

```
Dim GM As GeoMath
Dim O As GeoObject
Dim HasColor As Boolean
Dim Color As RGB

If O.HasColor Then
    Color = O.Color
End If
```

Color

Returns

RGB

Description

This property sets or returns the color of the `GeoObject`.

Example

```
Dim GM As GeoMath
Dim O As GeoObject
Dim HasColor As Boolean
Dim NewColor As RGB

With NewColor
    .Red = 255
    .Green = 0
    .Blue = 0
End With
O.Color = NewColor
```

Methods

`GeoObjects` can not be assigned to objects of a specific type.

So the statement :

```
Set P = Obj
```

is not valid. Instead, the statements `GetPoint`, `GetLine`, etc. need to be used.

GetPoint

Parameters

P As `GeoPoint`

Description

`GetPoint` sets the reference to the object in a variable of type `GeoPoint`. Note that the reference is copied, not the object.

Example

```
Dim Obj As GeoObject
Dim P As GeoPoint
```

```
Set P = Obj.GetPoint
```

GetLine

Parameters

```
L As GeoLine
```

Description

GetLine sets the reference to the object in a variable of type GeoLine. Note that the reference is copied, not the object.

Example

```
Dim Obj As GeoObject
Dim L As GeoLine

Set L = Obj.GetLine
```

GetArc

Parameters

```
A As GeoArc
```

Description

GetArc sets the reference to the object in a variable of type GeoArc. Note that the reference is copied, not the object.

Example

```
Dim Obj As GeoObject
Dim A As GeoArc

Set A = Obj.GetArc
```

GetCircle

Parameters

```
C As GeoCircle
```

Description

GetCircle sets the reference to the object in a variable of type GeoCircle. Note that the reference is copied, not the object.

Example

```
Dim Obj As GeoObject
Dim C As GeoCircle

Set C = Obj.GetCircle
```

GetClothoid

Parameters

```
C As GeoClothoid
```

Description

GetClothoid sets the reference to the object in a variable of type GeoClothoid. Note that the reference is copied, not the object.

Example

```
Dim Obj As GeoObject
Dim C As GeoClothoid

Set C = Obj.GetClothoid
```

GetCurve

Parameters

```
A As Curve
```

Description

GetCurve sets the reference to the object in a variable of type GeoCurve. Note that the reference is copied, not the object.

Example

```
Dim Obj As GeoObject
Dim C As GeoCurve

C = Obj.GetCurve
```

GetPolygon

Parameters

P As GeoPolygon

Description

GetPolygon sets the reference to the object in a variable of type GeoPolygon. Note that the reference is copied, not the object.

Example

```
Dim Obj As GeoObject
Dim P As GeoPolygon

Obj.GetPolygon P      ' Same as : Set P = Obj
```

GetPath

Parameters

P As GeoPath

Description

GetPath sets the reference to the object in a variable of type GetPath. Note that the reference is copied, not the object.

Example

```
Dim Obj As GeoObject
Dim P As GeoPath

Set P = Obj
```

Copy

Parameters

None

Returns

NewObj As GeoObject

Description

This method returns a copy of the object.

Example

```
Dim GM As GeoMath
Dim O1 As GeoObject, O2 As GeoObject

Set O2 = O1.Copy()
```


Chapter 88 - GeoPoint

Description

A GeoPoint is a GeoObject. A new GeoPoint can be created with the GeoMath method NewPoint or simple by using Set GP = New GeoPoint

Example

```
Dim GM As GeoMath
Dim PntObj As GeoPoint
Dim Pnt As XYZ

Set GM = New GeoMath
Pnt.x = 10.0
Pnt.y = 5.0
Pnt.z = 1.0
Set PntObj = GM.NewPoint(Pnt)
' or
Set PntObj = New GeoPoint
PntObj.XYZ = Pnt
```

Properties

XYZ
XY
X
Y
Z

Methods

GetGeoObject

Properties

XYZ

Returns

XYZ

Description

This property sets or returns the 3D coordinates of the point object .

Example

```
Dim P As GeoPoint
Dim Pnt As XYZ

Pnt = P.XYZ
```

XY

Returns

XY

Description

This property sets or returns the 2D coordinates of the point object .

Example

```
Dim P As GeoPoint
Dim Pnt As XY

Pnt = P.XY
```

X

Returns

Double

Description

This property sets or returns the X coordinate of the point object .

Example

```
Dim P As GeoPoint
```

```
P.X = 5.0
```

Y

Returns

Double

Description

This property sets or returns the Y coordinate of the point object .

Example

```
Dim P As GeoPoint
```

```
P.X = 10.0
```

Z

Returns

Double

Description

This property sets or returns the Z coordinate of the point object .

Example

```
Dim P As GeoPoint
```

```
P.Z = 2.5
```

Methods

GetGeoObject

Parameters

PO As GeoPoint

Returns

GeoObject

Description

GetGeoObject sets the reference to the GeoPoint in a variable of type GeoObject. Note that the reference is copied, not the object.

Example

```
Dim Obj As GeoObject
```

```
Dim P As GeoPoint
```

```
Set Obj = P.GetGeoObject() ' Same as : Set Obj = P
```

Chapter 89 - GeoLine

Description

A GeoLine is a GeoObject. A new GeoLine can be created with the GeoMath method NewLine or simple by using Set GP = New GeoLine

Example

```
Dim GM As GeoMath
Dim LineObj As GeoLine
Dim Pnt1 As XYZ, Pnt2 As XYZ

Set GM = New GeoMath
Pnt1.x = 10.0
Pnt1.y = 5.0
Pnt1.z = 1.0
Pnt2 = Pnt1
Pnt2.x = 20.0
Set LineObj = GM.NewLine(Pnt1, Pnt2)
' or
Set LineObj = New GeoLine
LineObj. = Pnt
```

Properties

Horizontal
Z
P1
P2
Bearing
Angle

Methods

GetGeoObject

Properties

For a description of the properties Horizontal, Zsee object type GeoObject.

P1 and P2

Returns

XYZ

Description

These properties set or return the coordinates of the line .

Example

```
Dim L As GeoLine
Dim P1 As XYZ, P2 As XYZ

P1 = L.P1
L.P2 = P2
```

Bearing

Returns

Double

Description

These properties returns the bearing angle of the line. The bearing is the angle of the line relative to the north axis and clockwise.

Example

```
Dim L As GeoLine
Dim MyBearing as Double

MyBearing = L.Bearing
```

Angle

Returns

Double

Description

These properties returns the angle of the line. The angle of the line is defined relative to the east axis and clockwise.

Example

```
Dim L As GeoLine
Dim MyAngle as Double

MyAngle = L.Angle
```

Methods

GetGeoObject

Parameters

Line As GeoLine

Returns

GeoObject

Description

GetGeoObject sets the reference to the GeoLine in a variable of type GeoObject. Note that the reference is copied, not the object.

Example

```
Dim Obj As GeoObject
Dim L As GeoLine

Set Obj = L.GetGeoObject()           ' Same as : Set Obj = L
```

Chapter 90 - GeoArc

Description

A GeoArc is a GeoObject. A new GeoArc can be created with the GeoMath method NewArc or simple by using Set GA = New GeoArc. If the elevation of Pnt1 and Pnt are not equal, the arc is part of an ellipse (spatially). The projection in the X-Y plane is a circular arc. The elevation of the arc is defined by the plane of the ellipse. The elevation of the center does not affect the elevation of the arc.

Example

```
Dim GM As GeoMath
Dim ArcObj As GeoArc
Dim C As XYZ, Pnt1 As XYZ, Pnt2 As XYZ

Set GM = New GeoMath
Pnt1.x = 10.0
Pnt1.y = 0.0
Pnt1.z = 0.0
Pnt2 = Pnt1
Pnt2.x = 0.0
Pnt2.y = 10.0
Set ArcObj = GM.NewArc(C, Pnt1, Pnt2)
' or
Set ArcObj = New GeoArc
ArcObj.C = C
ArcObj.P1 = Pnt1
ArcObj.P2 = Pnt2
```

Properties

Horizontal
Z
C
P1
P2

Methods

GetGeoObject

Properties

For a description of the properties Horizontal, Zsee object type GeoObject.

C

Returns

XYZ

Description

This property sets or returns the center of the arc .

Example

```
Dim A As GeoArc
Dim C As XYZ

C = A.C
```

P1

Returns

XYZ

Description

This property sets or returns the starting point of the arc .

Example

```
Dim A As GeoArc
Dim P1 As XYZ, P2 As XYZ

P1 = A.P1
A.P2 = P2
```

P2

Returns

XYZ

Description

This property sets or returns the end point of the arc .

Example

See P1

Radius

Returns

Double

Description

This property returns the radius (2D) of the arc .

Example

```
Dim A As GeoArc
```

```
Dim R As Double
```

```
R = A.Radius
```

Methods

GetGeoObject

Parameters

PO As GeoArc

Returns

GeoObject

Description

GetGeoObject sets the reference to the GeoPoint in a variable of type GeoObject. Note that the reference is copied, not the object.

Example

```
Dim Obj As GeoObject
```

```
Dim A As GeoArc
```

```
Set Obj = A.GetGeoObject() ' Same as : Set Obj = A
```

Chapter 91 - GeoCircle

Description

A GeoCircle is a GeoObject. A new GeoCircle is created with the GeoMath method NewCircle or simple by using Set GC = New GeoCircle. A GeoCircle is defined by its center and a second point. If the elevation of C and P are not equal, the object is an ellipse (spatially). The projection in the X-Y plane is a circle. The elevation of the circle is defined by the plane of the ellipse.

Example

```
Dim GM As GeoMath
Dim CircleObj As GeoCircle
Dim C As XYZ, P As XYZ, Pnt As XYZ

Set GM = New GeoMath
C.x = 0.0
C.y = 0.0
C.z = 0.0
Pnt = C
Pnt.y = 10.0
Set CircleObj = GM.NewCircle(C, Pnt)
' or
Set CircleObj = New GeoCircle
CircleObj.C = C
CircleObj.P = Pnt
```

Properties

Horizontal
Z
C
P

Methods

GetGeoObject

Properties

For a description of the properties Horizontal, Z see object type GeoObject.

C

Returns

XYZ

Description

This property sets or returns the center of the circle .

Example

```
Dim Circle As GeoCircle
Dim C As XYZ

C = Circle.C
```

P

Returns

XYZ

Description

This property sets or returns the radius point of the circle .

Example

```
Dim Circle As GeoCircle
Dim P As XYZ

P = Circle.P
```

Radius

Returns

Double

Description

This property returns the radius (2D) of the circle .

Example

```
Dim Circle As GeoCircle
Dim R As Double

R = Circle.Radius
```

Methods

GetGeoObject

Parameters

PO As GeoArc

Returns

GeoObject

Description

GetGeoObject sets the reference to the GeoPoint in a variable of type GeoObject. Note that the reference is copied, not the object.

Example

```
Dim Obj As GeoObject
Dim C As GeoCircle

Set Obj = C.GetGeoObject()           ' Same as : Set Obj = C
```


Chapter 92 - GeoClothoid

Description

A GeoClothoid is a GeoObject. A new GeoClothoid is created with the GeoMath method NewClothoid or simple by using Set GC = New GeoClothoid.

Example

```
Dim GM As GeoMath
Dim ClothoidObj As GeoClothoid
Dim P1 As XYZ, P2 As XYZ, R1 As Double, R2 As Double, K As Double
Dim Mirror As Boolean, Fi As Double

Set GM = New GeoMath
P1.x = 0.0
P1.y = 0.0
P1.z = 0.0
P2.x = 100.0
P2.y = 100.0
P2.z = 0.0
R1 = ptMaxDouble
R2 = 1000.0
K = 50
Mirror = True
Fi = TBD
Set ClothoidObj = GM.NewClothoid(P1, P2, R1, R2, K, Mirror, Fi)
```

Properties

Horizontal
Z
P1
P2
R1
R2
K
Fi
Mirror

Methods

GetGeoObject

Properties

For a description of the properties Horizontal, Z see object type GeoObject.

P1

Returns

XYZ

Description

This property returns the starting point of the Clothoid .

Example

```
Dim Cl As GeoClothoid
Dim P1 As XYZ

P1 = Cl.P1
```

P2

Returns

XYZ

Description

This property returns the 2nd point of the Clothoid .

Example

```
Dim Cl As GeoClothoid
Dim P2 As XYZ
```

```
P2 = Cl.P2
```

R1

Returns

Double

Description

This property returns the radius at the starting point (P1) of the clothoid. If the radius is infinite, ptMaxDouble is returned.

Example

```
Dim Cl As GeoClothoid
Dim R1 As Double

R1 = Cl.R1
```

R2

Returns

Double

Description

This property returns the radius at the end point (P2) of the Clothoid . If the radius is infinite, ptMaxDouble is returned.

Example

```
Dim Cl As GeoClothoid
Dim R2 As Double

R2 = Cl.R2
```

K

Returns

Double

Description

This property returns the rate of change of curvature also called the K-factor of the spiral curve (clothoid) .

Example

```
Dim Cl As GeoClothoid

Cl.K = 50.0
```

Mirror

Returns

Boolean

Description

This property returns the direction of the clothoid. If False the clothoid is in counter-clockwise direction (P1 - P2). If True clockwise.

Example

```
Dim Cl As GeoClothoid
Dim Direction As Boolean

Direction = Cl.Mirror
```

Methods

GetGeoObject

Parameters

PO As GeoArc

Returns

GeoObject

Description

GetGeoObject sets the reference to the GeoPoint in a variable of type GeoObject. Note that the reference is copied, not the object.

Example

```
Dim Obj As GeoObject  
Dim C As GeoClothoid
```

```
Set Obj = C.GetGeoObject()           ' Same as : Set Obj = C
```

Chapter 93 - GeoCurve

Description

A GeoCurve is a GeoObject. A new GeoCurve is created with the GeoMath method NewCurve or simple by using Set GC = New GeoCurve.

It represents a polyline, but for now it is not possible to differentiate between the different segment types.

Example

```
Dim GM As GeoMath
Dim CurveObj As GeoCurve

Set GM = New GeoMath
Redim Pnts(1 To 3)
' Init Pnts
Set CurveObj = GM.NewCurve(Pnts)
' or
Set CurveObj = New GeoCurve
CurveObj.SetPoints Pnts
```

Properties

Horizontal
Z
NrOfPoints
Curvature
MaxRounding
ThreeD
Elevation
StartTangent
EndTangent

Methods

SetPoints
GetPoints
GetGeoObject

Properties

For a description of the properties Horizontal, Z see object type GeoObject.

NrOfPoints

Returns

Long

Description

This property returns the number of point of the Curve .

Example

```
Dim C As GeoCurve

MsgBox C.NrOfPoints
```

Curvature (obsolete)

Returns

Integer

Description

Curvature is a property of the v15 Curve object.

This property returns the curvature of the curve.

0 = the curve contains only line segments.

5 = the curve contains only curve (Bezier) segments.

-1 = other cases.

The property also sets the curvature of the curve, if the curve contains only line segments or only curve (Bezier) segments. The value needs to be in the range [0, 10]. 0 are line segments, 5 are normal curve (Bezier) segments and the other values differ in curvature like the v15 Curve object.

Example

```
Dim C As GeoCurve
```

```
C.Curvature = 5
```

MaxRounding (obsolete)

Returns

Boolean

Description

It returns true since v16 polylines always have max rounding. The variable doesn't set the value since v16.

Example

```
Dim C As GeoCurve
Dim hasMaxRounding As Boolean

hasMaxRounding = C.Curvature 'Always true
```

ThreeD

Returns

Boolean

Description

This property returns a Boolean which indicates if the curve is 3D. A curve is 3D if not all points of the curve have the same elevation.

Example

```
Dim C As GeoCurve
Dim Z As Double

If C.ThreeD = False Then
    Z = C.Elevation
End If
```

Elevation

Returns

Double

Description

This property sets or returns the elevation of a horizontal curve. A curve is horizontal if all points of the curve have the same elevation. (Is not 3D). If the curve is not horizontal the value ptMaxDouble is returned.

Example

```
Dim C As GeoCurve
Dim Z As Double

If C.ThreeD = False Then
    Z = C.Elevation
End If
```

StartTangent (obsolete)

Returns

XY

Description

The property doesn't set anything since v16. It returns a point which gives the tangent direction at the starting point of the curve. If no tangent point or the first segment is not a Bezier segment, both coordinates are equal to ptMaxDouble.

Example

```
Dim C As GeoCurve
Dim P As XY

P = C.StartTangent
```

EndTangent (obsolete)

Returns

XY

Description

The property doesn't set anything since v16. It returns a point which gives the tangent direction at the end point of the curve. If no tangent point or the last segment is not a Bezier segment, both coordinates are equal to ptMaxDouble.

Example

```
Dim C As GeoCurve
Dim P As XY

P = C.EndTangent
```

Methods

SetPoints

Parameters

Pnts() As XYZ

Description

SetPoints sets or changes the points of the curve. It will create a curve with line segments

Example

```
Dim C As GeoCurve
Dim Pnts() As XYZ

Redim Pnts(1 To 4)
' Fill in Pnts
C.SetPoints Pnts
```

GetPoints

Parameters

Pnts() As XYZ

Description

GetPoints returns the points of the curve.

Example

```
Dim C As GeoCurve
Dim Pnts() As XYZ

Redim Pnts(1 To C.NrOfPoints)
C.GetPoints Pnts
```

GetGeoObject

Parameters

C As GeoCurve

Returns

GeoObject

Description

GetGeoObject sets the reference to the GeoCurve in a variable of type GeoObject. Note that the reference is copied, not the object.

Example

```
Dim Obj As GeoObject
Dim C As GeoCurve

Set Obj = C.GetGeoObject() ' Same as : Set Obj = C
```

Chapter 94 - GeoPolygon

Description

A GeoPolygon is a GeoObject. A new GeoPolygon is created with the GeoMath method NewPolygon or simple by using Set GC = New GeoPolygon.

Example

```
Dim GM As GeoMath
Dim Poly As GeoPolygon
Dim Pnts() As XYZ

Set GM = New GeoMath
Redim Pnts(1 To 5)
'Fill Pnts
Set Poly = GM.NewPolygon(Pnts)
```

Properties

Horizontal
Z
NrOfPoints
NrOfPolygonElements
Area
Perimeter
Centroid
Pattern

Methods

SetPoints
GetPoints
GetGeoObject
PointInPolygon

Properties

For a description of the properties Horizontal, Zsee object type GeoObject.

NrOfPoints

Returns

Long

Description

This property returns the number of point of the Polygon .

Example

```
Dim Poly As GeoPolygon

MsgBox Poly.NrOfPoints
```

NrOfPolygonElements

Returns

Long

Description

This property returns the number of path elements of the Polygon .

Example

```
Dim Poly As GeoPolygon

MsgBox Poly.NrOfPolygonElements
```

Area

Returns

Double

Description

This property returns the number of point of the Polygon .

Example

```
Dim Poly As GeoPolygon

MsgBox Poly.NrOfPoints
```

Perimeter

Returns

Double

Description

This property returns the perimeter (2D) of the Polygon .

Example

```
Dim Poly As GeoPolygon
Dim P As Double

P = Poly.Perimeter
```

Centroid

Returns

XY

Description

This property returns the centroid of the Polygon .

Example

```
Dim Poly As GeoPolygon
Dim Center As XY

Center = Poly.Centroid
```

Pattern

Returns

Style

Description

This property sets or returns the hatching of the polygon. See also the constant `Polygon:StyleId`. Only the default patterns are supported.

Example

```
Dim Poly As GeoPolygon
Dim O As GeoObject
Dim Poly As GeoPolygon
Dim NewStyle As Style

O.GetPolygon Poly
With NewStyle
    .Symbol = False
    .StyleId = ptSolidFill
End With
Poly.Pattern = NewStyle
```

Methods

SetPoints

Parameters

Pnts() As XYZ

Description

SetPoints sets or changes the points of the polygon.

Example

```
Dim Poly As GeoPolygon
Dim Pnts() As XYZ

Redim Pnts(1 To 4)
```



```
' Fill in Pnts
Poly.SetPoints Pnts
```

GetPoints

Parameters

```
Pnts() As XYZ
```

Description

GetPoints returns the points of the polygon.

Example

```
Dim Poly As GeoPolygon
Dim Pnts() As XYZ

Redim Pnts(1 To Poly.NrOfPoints)
Poly.GetPoints Pnts
```

SetElements

Parameters

```
Elements() As PathElement
```

Description

SetElements sets or changes the path elements that form the polygon. The Elements array should start on index 1.

Example

```
Dim Poly As GeoPolygon
Dim Elements() As PathElement

Redim Elements(1 To 4)
' Fill in Elements
Poly.SetElements Elements
```

GetElements

Parameters

```
Elements() As PathElement
```

Description

GetElements returns the path elements of the polygon.

Example

```
Dim Poly As GeoPolygon
Dim Elements() As PathElement

Redim Elements (1 To Poly. NrOfPolygonElements)
Poly.GetElements Elements
```

GetGeoObject

Parameters

```
Poly As GeoPolygon
```

Returns

```
GeoObject
```

Description

GetGeoObject sets the reference to the GeoPolygon in a variable of type GeoObject. Note that the reference is copied, not the object.

Example

```
Dim Obj As GeoObject
Dim Poly As GeoPolygon

Set Obj = Poly.GetGeoObject() ' Same as : Set Obj = Poly
```

PointInPolygon

Parameters

Pnt As GeoPoint

Returns

long

Description

PointInPolygon returns the relative position of the point towards the polygon. The returned result can be ptInside, ptOutside, ptOnVertex or ptOnEdge

Example

```
Dim Poly As GeoPolygon
Dim Pnt As GeoPoint

If Poly.PointInPolygon(Pnt) = ptInside Then
    MsgBox " Point is really inside "
End If
```

ClipLine

Parameters

MyLine as GeoLine
KeepInside as Boolean
Clipped as Boolean
NrOfParts as Long
Parts() as GeoLine

Returns

Boolean

Description

ClipLine returns the result of the clip operation of this line with this polygon. If the clip operation was successful, the method return True. The Clipped parameter returns if the operation has changed the GeoLine.. If Clipped = True and any part of line is still left after the clipping operation, the NrOfParts and the Parts array return the resulting line parts as GeoLine

Example

```
Dim Poly As GeoPolygon
Dim Line As GeoLine
Dim Clipped As Boolean
Dim NrOfParts as Long
Dim Parts() as GeoLine

If Poly.ClipLine(Line, True, Clipped, NrOfParts, Parts) Then
    If Clipped Then
        If NrOfParts = 0 Then
            MsgBox "Line is completely clipped"
        Else
            MsgBox "Part of line is still inside the polygon"
        End If
    End If
End If
```

ClipArc

Parameters

MyArc as GeoArc
KeepInside as Boolean
Clipped as Boolean
NrOfParts as Long
Parts() as GeoArc

Returns

Boolean

Description

ClipArc returns the result of the clip operation of this arc with this polygon. If the clip operation was successful, the method return True. The Clipped parameter returns if the operation has changed the GeoArc.. If Clipped = True and any part of arc is still left after the clipping operation, the NrOfParts and the Parts array return the resulting arc parts as GeoArc

Example

```
Dim Poly As GeoPolygon
Dim Arc As GeoArc
Dim Clipped As Boolean
Dim NrOfParts as Long
Dim Parts() as GeoArc

If Poly.ClipArc(Arc, True, Clipped, NrOfParts, Parts) Then
    If Clipped Then
        If NrOfParts = 0 Then
            MsgBox "Arc is completely clipped"
        Else
            'Arc is partially clipped
        End If
    End If
End If
```

```

        MsgBox "Part of arc is still inside the polygon"
    End IF
End If
End If

```

ClipCircle

Parameters

```

MyCircle as GeoCircle
KeepInside as Boolean
Clipped as Boolean
NrOfParts as Long
Parts() as GeoArc

```

Returns

```

Boolean

```

Description

ClipCircle returns the result of the clip operation of this circle with this polygon. If the clip operation was successful, the method return True. The Clipped parameter returns if the operation has changed the GeoCircle.. If Clipped = True and any part of circle is still left after the clipping operation, the NrOfParts and the Parts array return the resulting arc parts as GeoArc

Example

```

Dim Poly As GeoPolygon
Dim Circle As GeoCircle
Dim Clipped As Boolean
Dim NrOfParts as Long
Dim Parts() as GeoArc

If Poly.ClipCircle(Circle, True, Clipped, NrOfParts, Parts) Then
    If Clipped Then
        If NrOfParts = 0 Then
            MsgBox "Circle is completely clipped"
        Else
            MsgBox "Part of Circle is still inside the polygon"
        End IF
    End If
End If

```

ClipCurve

Parameters

```

MyCurve as GeoCurve
KeepInside as Boolean
Clipped as Boolean
NrOfParts as Long
Parts() as GeoCurve

```

Returns

```

Boolean

```

Description

ClipCurve returns the result of the clip operation of this curve with this polygon. If the clip operation was successful, the method return True. The Clipped parameter returns if the operation has changed the GeoCurve. If Clipped = True and any part of curve is still left after the clipping operation, the NrOfParts and the Parts array return the resulting curve parts as GeoCurve

Example

```

Dim Poly As GeoPolygon
Dim Curve As GeoCurve
Dim Clipped As Boolean
Dim NrOfParts as Long
Dim Parts() as GeoCurve

If Poly.ClipCurve(Curve, True, Clipped, NrOfParts, Parts) Then
    If Clipped Then

```

```

        If NrOfParts = 0 Then
            MsgBox "Curve is completely clipped"
        Else
            MsgBox "Part of Curve is still inside the polygon"
        End IF
    End If
End If

```

ClipPolygon

Parameters

```

MyPolygon as GeoPolygon
KeepInside as Boolean
Clipped as Boolean
NrOfParts as Long
Parts() as GeoPolygon

```

Returns

```

Boolean

```

Description

ClipPolygon returns the result of the clip operation of MyPolygon with this polygon. If the clip operation was successful, the method return True. The Clipped parameter returns if the operation has changed the MyPolygon GeoPolygon. If Clipped = True and any part of polygon is still left after the clipping operation, the NrOfParts and the Parts array return the resulting polygon parts as GeoPolygon

Example

```

Dim Poly As GeoPolygon
Dim MyPolygon As GeoPolygon
Dim Clipped As Boolean
Dim NrOfParts as Long
Dim Parts() as GeoPolygon

```

```

If Poly.ClipPolygon(MyPolygon, True, Clipped, NrOfParts, Parts) Then
    If Clipped Then
        If NrOfParts = 0 Then
            MsgBox "MyPolygon is completely clipped"
        Else
            MsgBox "Part of MyPolygon is still inside the polygon"
        End IF
    End If
End If

```

Chapter 95 - GeoPath

Description

A GeoPath is a GeoObject. A new GeoPath is created with the GeoMath method NewPath or simply by using Set GC = New GeoPath.

Example

```
Dim A As Application
Dim D As Document
Dim Sel As Selection
Dim Obj As CadObject
Dim Pth As Path
Dim GeoP As GeoPath
Dim Elems() As PathElement
Dim GM As GeoMath

' Get path elements from CadObject
Set A= New Application
A.GetActiveDocument D
D.GetSelection Sel
If Sel.Count(ptPath) >= 1 Then
    Sel.GetFirst ptPath, Obj
    Obj.GetPath Pth
    Redim Elems(1 To Pth.NrOfElements)
    Pth.GetPathElements Elems

    ' Create new geopath
    Set GM = New GeoMath
    Set GeoP = GM.NewPath(Elems)
    MsgBox "it works"
End If
```

Properties

NrOfElements
Length

Methods

SetElements
GetElements
GetParallel
GetGeoObject
CDToXY
XYToCD

Properties

NrOfElements

Returns

Long

Description

This property returns the number of elements of the Path .

Example

```
Dim GeoP As GeoPath

MsgBox GeoP.NrOfElements
```

Length

Returns

Double

Description

This property returns the length of the Path .

Example

See SetElements

Methods

SetElements

Parameters

Elements() As PathElement

Description

SetElements sets or changes the elements of the path. The Elements array should start on index 1.

Example

```
Dim A As Application
Dim A As Application
Dim D As Document
Dim Sel As Selection
Dim Obj As CadObject
Dim Pth As Path
Dim GeoP As GeoPath
Dim GeoP2 As GeoPath
Dim Elems() As PathElement
Dim GM As GeoMath

Set GM = New GeoMath
Set A= New Application
A.GetActiveDocument D
D.GetSelection Sel
If Sel.Count(ptPath) >= 2 Then

    'define first GeoPath
    Sel.GetFirst ptPath, Obj
    Obj.GetPath Pth
    Redim Elems(1 To Pth.NrOfElements)
    Pth.GetPathElements Elems
    Set GeoP = GM.NewPath(Elems)

    'define second GeoPath
    Sel.GetNext Obj
    Obj.GetPath Pth
    Redim Elems(1 To Pth.NrOfElements)
    Pth.GetPathElements Elems
    Set GeoP2 = GM.NewPath(Elems)
    MsgBox CStr(GeoP2.Length)

    'reset second GeoPath
    GeoP.GetElements Elems
    GeoP2.SetElements Elems
    MsgBox CStr(GeoP2.Length)
End If
```

GetElements

Parameters

Elements() As PathElement

Description

GetElements returns the elements of the path. Indexes in the Elements array are starting from 1.

Example

See SetElements

GetParallel

Parameters

dS As LongReal, NrOfElements As Long, Elements() As PathElement

Description

GetParallel returns the elements of the parallel path on distance dS. When dS is positive the procedure will calculate the parallel on the right side of the path, negative distance will return parallel elements on the left side of the path. Indexes in the Elements array are starting from 1.

Example

```
Dim A As Application
Dim D As Document
Dim Sel As Selection
Dim Obj As CadObject
Dim Pth As Path
Dim GeoP As GeoPath
Dim Elems() As PathElement
Dim i As Long, Count As Long
Dim PE As PathElement
Dim GM As GeoMath
Dim P1 As Point, P2 As Point
Dim L As Line
Dim Pos As XYZ
Dim Drawing As Overlay

Set GM = New GeoMath
Set A= New Application
A.GetActiveDocument D
D.GetOverlay ptDrawing, Drawing
D.GetSelection Sel

If Sel.Count(ptPath) >= 1 Then
    'define GeoPath
    Sel.GetFirst ptPath, Obj
    Obj.GetPath Pth
    Redim Elems(1 To Pth.NrOfElements)
    Pth.GetPathElements Elems
    Set GeoP = GM.NewPath(Elems)

    'calculate parallel
    GeoP.GetParallel 10.0, Count, Elems
    For i = LBound(Elems) To UBound(Elems)
        Pos.X = Elems(i).BeginPoint.X
        Pos.Y = Elems(i).BeginPoint.Y
        Drawing.CreatePoint Pos, P1
        Pos.X = Elems(i).EndPoint.X
        Pos.Y = Elems(i).EndPoint.Y
        Drawing.CreatePoint Pos, P2
        Drawing.CreateLine P1, P2, L
    Next i
End If
```

GetGeoObject

Parameters

Path As GeoPath

Returns

GeoObject

Description

GetGeoObject sets the reference to the GeoPath in a variable of type GeoObject. Note that the reference is copied, not the object.

Example

```
Dim Obj As GeoObject
Dim GeoP As GeoPath

Set Obj = GeoP.GetGeoObject()           ' Same as : Set Obj = GeoP
```

CDToXY

Parameters

Chainage As Double, Distance As Double, Pnt As XY

Description

CDToXY converts the values Chainage and Distance to coordinates.

Example

```
Dim A As Application
Dim Doc As Document
```



```

Dim GeoP As GeoPath
Dim Pnt as XY

GeoP.CDToXY 70.0, 5.0, Pnt
MsgBox "X = " & Format(Pnt.x, "###.000") & vbNewLine & _
      "Y = " & Format(Pnt.Y, "###.000") & vbNewLine

```

XYToCD

Parameters

Pnt As XY, Chainage As Double, Distance As Double, PointOnPath as Boolean

Description

XYtoCD converts the coordinates of a point to the values Chainage, Distance. PointOnPath returns if the point still projects on the GeoPath.

Example

See CD2XY

Chapter 96 - MatrixOperations

Description

A `MatrixOperations` object is a collection of methods to do operations with or between two dimensional VBA Arrays.

In this chapter we are calling a two dimensional array of `Double` values a matrix.

In the methods of this chapter the input matrices can be one dimensional or two dimensional arrays. There is no restriction on lower bound or upper bound of these arrays.

However, all output matrices will be created as two dimensional arrays. The lower bounds of these matrices will start from 1. See next example for illustration.

Example

```
Dim Mat As MatrixOperations
Dim A() As Double
Dim B() As Double
Dim AB() As Double

Set Mat = New MatrixOperations
Redim A(0 To 2, 0 To 2)
Redim B(1 To 3)
A(0,0) = 1
A(0,1) = 1
A(0,2) = 1
A(1,0) = 1
A(1,1) = 1
A(1,2) = 1
A(2,0) = 1
A(2,1) = 1
A(2,2) = 1
B(1) = 5
B(2) = 10
B(3) = 20

Mat.Multiply A, B, AB
MsgBox "x1 = " & CStr(AB(1,1)) & ", x2 = " & CStr(AB(2,1)) & ", x3 = " & CStr(AB(3,1))
```

Methods

```
NewMatrix
NewIdentityMatrix
IsEqual
Add
Multiply
Transpose
Inverse
Solve
Determinant
```

Methods

NewMatrix

Parameters

m As Integer, n As Integer, A() As Double

Description

This method creates in A a double array with all elements equal to zero.

The bounds of the matrix A will be defined as [1..m, 1..n].

Example

```
Dim Mat As MatrixOperations
Dim A() As Double
Dim i As Integer, j As Integer

Set Mat = New MatrixOperations
Mat.NewMatrix 3, 3, A
For i = 1 To 3
```

```

        For j = 1 To 3
            If A(i,j) <> 0 Then
                MsgBox "Initialisation failed"
            End If
        Next j
    Next i

```

NewIdentityMatrix

Parameters

n As Integer, A() As Double

Description

This method creates in Aa double array representing the identity matrix of dimension n. The bounds of the matrix A will be defined as [1..n, 1..n].

Example

```

Dim Mat As MatrixOperations
Dim A() As Double
Dim i As Integer
Dim j As Integer

Set Mat = New MatrixOperations
Mat.NewIdentityMatrix 3, A
For i = 1 To 3
    For j = 1 To 3
        If i <> j Then
            If A(i,j) <> 0 Then
                MsgBox "Identity matrix incorrect"
            End If
        Else
            If A(i,j) <> 1 Then
                MsgBox "Identity matrix incorrect "
            End If
        End If
    Next j
Next i

```

IsEqual

Parameters

A As Array() Of Double, B As Array() Of Double , Optional Tolerance As Real := 0.

Returns

Equal As Boolean

Description

This method returns True if all elements of A and B are equal. Both matrices should have the same dimensions. If not, False is returned. If positive value of Tolerance is specified then matrices with corresponding elements differing not more than by this value are judged to be equal.

Example

```

Dim Mat As MatrixOperations
Dim A() As Double
Dim B() As Double

Set Mat = New MatrixOperations
Mat.NewIdentityMatrix 3, A
Mat.NewIdentityMatrix 3, B

If Not Mat.IsEqual(A, B) Then
    MsgBox "IsEqual failed"
End If

```

Add

Parameters

A As Array() Of Double, B As Array() Of Double, C As Array() Of Double

Description

This method adds the matrices A and B and stores the result into C.

The dimensions of A and B should be the same. If we note the dimensions of A and B as m by n, then the bounds of C will be defined as [1..m, 1..n]

Example

```
Dim Mat As MatrixOperations
Dim A() As Double
Dim B() As Double
Dim C() As Double
Dim i As Integer, j As Integer

Set Mat = New MatrixOperations
Mat.NewIdentityMatrix 3, A
Mat.NewIdentityMatrix 3, B
Mat.Add A, B, C
For i = 1 To 3
    For j = 1 To 3
        If I <> j Then
            If C(i,j) <> 0 Then
                MsgBox "Sum was incorrect"
            End If
        Else
            If C(i,j) <> 2 Then
                MsgBox "Sum was incorrect"
            End If
        End If
    Next j
Next i
```

Multiply

Parameters

A As Array() Of Double, B As Array() Of Double, C As Array() Of Double

Description

This method multiplies the matrices A and B and stores the result into C. The number of columns in A should match the number of rows in B.

If A holds m rows and n columns, if B holds o rows and p columns, the bounds of C will be defined as [1..m, 1..p].

Example

See example at the beginning of the chapter

Transpose

Parameters

A As Array() Of Double, AT As Array() Of Double

Description

This method defines the transposed matrix of A in AT.

The element AT[j, i] will be equal to the element A[i, j].

If A holds m rows and n columns the bounds of AT will be defined as [1..n, 1..m].

Example

```
Dim Mat As MatrixOperations
Dim A() As Double
Dim AT() As Double
Dim i As Integer, j As Integer

Set Mat = New MatrixOperations
Redim A(1 To 3, 1 To 2)

A(1,1) = 5
A(1,2) = 6
A(2,1) = 8
```

```

A(2,2) = 3
A(3,1) = 5
A(3,2) = 8

Mat.Transpose A, AT
For i = 1 To 3
    For j = 1 To 2
        If A(i,j) <> AT(j,i) Then
            MsgBox "transpose failed"
        End If
    Next j
Next i

```

Inverse

Parameters

A As Array() Of Double, AInverse As Array() Of Double

Returns

Ok As Boolean

Description

This method defines the inverse matrix of A in AInverse. The calculation of the inverse matrix is only possible when the array is square (n by n) and when the array is regular. The parameter Ok will indicate if the inverse calculation was successful.

If A holds m rows and n columns the bounds of AInverse will be defined as [1..m, 1..n].

Example

```

Dim Mat As MatrixOperations
Dim A() As Double
Dim AInv() As Double
Dim Prod() As Double
Dim I() As Double
Dim Ok As Boolean

Set Mat = New MatrixOperations
Redim A(1 To 3, 1 To 3)

A(1,1) = 1
A(1,2) = 3
A(1,3) = 1
A(2,1) = 1
A(2,2) = 1
A(2,3) = 2
A(3,1) = 2
A(3,2) = 3
A(3,3) = 4

Ok = Mat.Inverse(A, AInv)
If Not Ok Then
    MsgBox "Inverse calculation impossible"
Else
    Mat.NewIdentityMatrix 3, I
    Mat.Multiply A, AInv, Prod
    If Not Mat.IsEqual(I, Prod) Then
        MsgBox "Inverse calculation is not correct"
    Else
        MsgBox "Inverse calculation calculated succesfully"
    End If
End If

```

Solve

Parameters

A As Array() Of Double, B As Array() Of Double, X As Array() Of Double

Returns

Ok As Boolean

Description

This methods solves the linear equation $A \cdot X = B$.

A solution will only be found under certain conditions: We need the matrix A to be regular and with dimensions n by n (square matrix). We only support matrices B of dimension n (by 1).

The matrix x will be a matrix of dimensions [1..n, 1..1].

The return parameter Ok will be True when a solution was found, otherwise this method returns False.

Example

```
Dim Mat As MatrixOperations
Dim A() As Double
Dim B() As Double
Dim X() As Double
Dim Ok as Boolean

Set Mat = New MatrixOperations
Redim A(1 To 3, 1 To 3)
Redim B(1 To 3)
A(1,1) = 1
A(1,2) = 2
A(1,3) = -1
A(2,1) = 2
A(2,2) = -1
A(2,3) = 3
A(3,1) = 7
A(3,2) = -3
A(3,3) = -2
B(1) = 11
B(2) = 7
B(3) = 2

Ok = Mat.Solve(A, B, X)
If Not Ok Then
    MsgBox "Equations were not solved"
Else
    MsgBox "x1 = " & CStr(X(1,1)) & ", x2 = " & CStr(X(2,1)) & ", x3 = " &
    CStr(X(3,1))
End If
```

Determinant

Parameters

A As Array() Of Double

Returns

Determinant As Double

Description

This method calculates the determinant of A. The calculation of the determinant is only possible when the array is square (n by n).

Example

```
Dim Mat As MatrixOperations
Dim A() As Double
Dim det As Double

Set Mat = New MatrixOperations
Redim A(1 To 3, 1 To 3)

A(1,1) = 1
A(1,2) = 3
A(1,3) = 1
A(2,1) = 1
A(2,2) = 1
A(2,3) = 2
A(3,1) = 2
A(3,2) = 3
A(3,3) = 4

Det = Mat.Determinant(A)
MsgBox CStr(Det) ' should be -1
```

Chapter 97 - HelmertTransformation

Description

The `HelmertTransformation` object assist in executing a Helmert Transformation on a drawing or part of a drawing. The method `Calculate` will calculate the parameters used by the Helmert Transformation based on `n` pairs of points. These pairs are called pass points. The instance of a `HelmertTransformation` object retains not only the transformation parameters, but also the coordinates of the pairs of points on which the parameters where calculated. This allows to adjust the drawing after transformation. Adjustment (smoothing out residuals) in this context means that each object will be moved. The closure error (residuals) for the pass points will become 0. See also the method `Transform` which can be applied to `Selection`, `FlaggedObjects` and `AllObjects`.

Example

```
Dim H As HelmertTransformation
Dim FromPnts() As XY, ToPnts() As XY
Dim Ok As Boolean
Dim NrPoints As Long
Dim All As AllObjects
Dim Doc As Document
Dim Drawing As Overlay

Set A = New Application
Set H = New HelmertTransformation

A.GetActiveDocument Doc
Doc.GetOverlay ptDrawing, Drawing
Drawing.GetAllObjects All

GetNumberOfPoints NrPoints      'a specific local procedure to get nr of points
Redim FromPnts(1 To NrPoints)
Redim ToPnts(1 To NrPoints)
ReadPoints FromPnts, ToPnts     'a specific local procedure to read points

H.Calculate NrPoints, FromPnts, ToPnts, Ok
If Ok Then
    All.Transform H, True
End If
```

Properties

`NumberOfPassPoints`
`TransformationParameters`
`StandardDeviation`

Methods

`Calculate`
`GetResiduals`
`GetDelaunayTriangulation`
`Transform`

Properties

NumberOfPassPoints

Returns

Long

Description

This property returns the number of pass points used to calculate the transformation parameters.

Example

```
Dim H As HelmertTransformation
Dim Errors() As Long

Redim Errors(1 To H.NumberOfPassPoints)
```

TransformationParameters

Returns

TransformationParameters

Description

This property returns the transformation parameters. (See Pythagoras Types)

Example

```
Dim H As HelmertTransformation
Dim TPs As TransformationParameters

TPs = H.TransformationParameters
MsgBox "Angle = " & Format(H.TransformationParameters.RotationAngle, "#.###")
MsgBox "Scale = " & Format(H.TransformationParameters.ScaleFactor, "#.####")
MsgBox "dX = " & Format(H.TransformationParameters.dX, "#.###")
MsgBox "dY = " & Format(H.TransformationParameters.dY, "#.###")
```

StandardDeviation

Returns

Double

Description

This property returns the standard deviation of the transformation.

Example

```
Dim H As HelmertTransformation

MsgBox Format(H.StandardDeviation, "0.###")
```

Methods

Calculate

Parameters

NumberOfPassPoints As Long, From() As XY, To() As XY, Ok As Boolean

Description

Calculates the transformation parameters and keeps the result and the original pass points in the object.

Example

```
Dim H As HelmertTransformation
Dim FromPnts() As XY, ToPnts() As XY
Dim Ok As Boolean
Dim NrPoints As Long
Dim All As AllObjects

Set H = New HelmertTransformation
InitialiseData 'a specific procedure to initialize data
H.Calculate NrPoints, FromPnts, ToPnts, Ok
If Ok Then
    All.Transform H, True
End If
```

GetResiduals

Parameters

Residuals() As XY

Description

Returns the residuals of the pass points if the Helmert Transformation would be applied.
On return, the lower bound index will always be equal to 1.

Example

```
Dim H As HelmertTransformation
Dim FromPnts() As XY, ToPnts() As XY, Residuals() As XY
Dim Ok As Boolean
Dim NrPoints As Long, i As Long
Dim R As XY
```



```

InitialiseData    'a specific procedure to initialize data
H.Calculate NrPoints, FromPnts, ToPnts, Ok
If Ok Then
    H.GetResiduals Residuals
    For i = 1 To NrPoints
        R.x = R.x + Residuals(i).x
        R.y = R.y + Residuals(i).y
    Next i
End If

```

GetDelaunayTriangulation

Parameters

Triangulation As DelaunayTriangulation

Description

A Delaunay Triangulation based on the passpoints and with residuals as associated values is calculated and returned. The Delaunay Triangulation is saved together with the object so that methods using the Helmert Transformation may use the Delaunay Triangulation for interpolation purposes. See [CoordinateReferenceSystems](#) method [TransformXYTo](#) and the method [Transform](#) of [Selection](#), [FlaggedObjects](#) and [AllObjects](#).

Example

```

Dim H As HelmertTransformation
Dim T As DelaunayTriangulation

H.GetDelaunayTriangulation T

```

Transform

Parameters

Pnt As XY, Optional SmoothResiduals As Integer

Returns

XY

Description

Returns the transformed coordinate. For the meaning of optional parameter [SmoothResiduals](#) see [Selection](#), [FlaggedObjects](#), [AllObjects](#) method [Transform](#).

Example

```

Dim H As HelmertTransformation
Dim P As Point, Pt As Point

Pt = H.Transform(Pt, ptNNArea)

```

Chapter 98 - DelaunayTriangulation

Description

The DelaunayTriangulation object calculates a Delaunay Triangulation of a set of points and provides tools to do some calculations using the Triangulation.

Each vertex of a triangle may have n values. Where n may be 0 or any other higher value.

For one instance of a DelaunayTriangulation all values must have the same dimension.

A 1-dimensional value could represent an elevation of a point, a 2-dimensional value could for example indicate the residuals of passpoints. See method GetResiduals of the object

HelmertTransformation.

A DelaunayTriangulation object could be used in combination with a HelmertTransformation, CoordinateReferenceSystems and TiePoints to Transform data from one Coordinate Reference System to another using a set of Tie Points (pass points) and natural neighbours interpolation to smooth out the Residuals.

See also the method Transform which can be applied to Selection, FlaggedObjects and AllObjects to transform a set of points to use Natural Neighbours interpolation.

Example

```
Dim DT As DelaunayTriangulation
Dim Points() As XYV
Dim NrPoints As Long
Dim T As Triangle

Set DT = New DelaunayTriangulation

GetNumberOfPoints NrPoints      'a specific procedure to get nr of points
Redim Points(1 To NrPoints)
ReadPoints Points               'a specific procedure to read points and
                               'corresponding real values

DT.Initialise NrPoints, Points

DT.GetFirst T
Do While T Is Not Nothing
    . . .                      ' triangle processing (drawing for example)
    DT.GetNext T
Loop
```

Properties

NumberOfPoints
NumberOfTriangles

Methods

Initialise
GetFirst
GetNext
GetTriangleVertexes
NaturalNeighbours
PointInTriangulation
Interpolation

Properties

NumberOfPoints

Returns

Long

Description

This property returns the number of points used to create Delaunay triangulation.

Example

```
Dim DT As DelaunayTriangulation
Dim EndPoints() As Long
```

```
Redim EndPoints(1 To DT.NumberOfPoints)
```

NumberOfTriangles

Returns

Long

Description

This property returns the number of triangles.

Example

```
Dim DT As DelaunayTriangulation
Dim NrTriangles

NrTriangles = DT.NumberOfTriangles
```

Methods

Initialise

Parameters

NumberOfPoints As Long, Points() As XYV

Description

Creates a Delanay Triangulation with Points as basic set of points. The list of points should start from index 0.

This method can raise the following errors:

- Insufficient number of points or all points are collinear
- There are 2 or more points with the same coordinates
- Triangulation can not be calculated.

Example

See example above

GetFirst

Parameters

Triangle As Triangle

Description

Returns first triangle of the Delanay Triangulation.

Example

See example above

GetNext

Parameters

Triangle As Triangle

Description

Returns next triangle of the Delanay triangulation.

Example

See example above

GetTriangleVertexes

Parameters

P As XY, Vertexes() As XY

Description

Returns the 3 vertexes of the triangle containing point P.

Example

```
Dim DT As DelaunayTriangulation
Dim Points() As XYV
Dim NrPoints As Long
Dim Pnt As XY
```

```

Dim Vertexes() As XYV

Set DT = New DelaunayTriangulation
DT.Initialise NrPoints, Points
Pnt.X = 36.0
Pnt.Y = 1.0
DT.GetTriangleVertexes Pnt, Vertexes

```

NaturalNeighbours

Parameters

P As XY, Neighbours() As XYV

Description

Returns natural neighbours of the given point P.

Example

```

Dim DT As DelaunayTriangulation
Dim Points() As XYV
Dim NrPoints As Long
Dim Neighbours() As XYV
Dim P As XY

Set DT = New DelaunayTriangulation

GetNumberOfPoints NrPoints      'a specific procedure to get nr of points
Redim Points(1 To NrPoints)
ReadPoints Points               'a specific procedure to read points and
                                'corresponding real values

P.x = 10.0
P.y = 18.4
DT.Initialise NrPoints, Points
DT.NaturalNeighbours P, Neighbours

```

PointInTriangulation

Parameters

P As XY

Returns

Boolean

Description

Returns True if P is inside the triangulation.

Example

```

Dim DT As DelaunayTriangulation
Dim P As XY

P.x = 2568368.819
P.y = 5678220.129
If DT.PointInTriangulation(P) Then
    MsgBox "Point is in triangulation"
Else
    MsgBox "Point is out of triangulation"
End If

```

Interpolation

Parameters

P As XY, InterpolationType As Integer, Res() As Double

Description

Returns interpolated value for the given point P. The type of interpolation may be: ptInTrianglePlane, ptNNArea, ptNNDistance, ptAllDistances
 If no solution every dimension of Res = ptNilValue.
 The result list starts from index 0.

Example

```

Dim DT As DelaunayTriangulation
Dim Points() As PointVal

```

```

Dim NrPoints As Long
Dim Res() As Double
Dim P As XY

Set DT = New DelaunayTriangulation

GetNumberOfPoints NrPoints      'a specific procedure to get nr of points
Redim Points(1 To NrPoints)
ReadPoints Points                'a specific procedure to read points and
                                'corresponding real values

P.x = 10.0
P.y = 18.4
DT.Initialise NrPoints, Points
DT.Interpolation P, ptNNArea, Res

```

Chapter 99 - Camera

Description

A Camera object can be obtained from `Cameras` by using the methods `GetFirst`, `GetNext`, `GetCamera`.

Example

```
Dim Cs As Cameras
Dim C As Camera

Cs.GetFirst C ' Returns the first stored camera.
```

Properties

Name
Position
Direction
Projection

Properties

Name

Returns

String

Description

This property returns or sets the name of the camera.

Example

```
Dim C As Camera
MsgBox "The name of this camera is " & C.Name
```

Position

Returns

XYZ

Description

This property sets or returns coordinates of a camera's position.

Example

```
Dim C As Camera
Dim P As XYZ

P = C.Position
MsgBox "The Camera is at point (" & P.X & ", " & P.Y & ", " & P.Z & ")"
```

Direction

Returns

XYZ

Description

This property sets or returns coordinates of a camera's direction.

Example

```
Dim C As Camera
Dim P As XYZ

P = C.Direction
MsgBox "The Camera looks in direction (" & P.X & ", " & P.Y & ", " & P.Z & ")"
```

Projection

Returns

ptOrthogonal or ptPerspective

Description

This property sets or returns a camera's projection.

Example

```
Dim C As Camera
If C.Projection = ptPerspective Then
    MsgBox "The Camera uses Perspective projection"
Else
    MsgBox "The Camera uses Orthogonal projection"
End If
```

Chapter 100 - Cameras

Description

A Cameras object is a set of all stored cameras of a document. The Cameras object is created by the Document method GetCameras.

Example

```
Dim A As Application
Dim D As Document
Dim Cs As Cameras
Dim C As Camera

set A = New Application
A.GetActiveDocument D
If Not (D is Nothing) Then
    D.GetCameras Cs
    Cs.GetCamera "Main", C
End If
```

Properties

NrOfCameras

Methods

GetFirst
GetNext
GetCamera
AddCamera
DeleteCamera

Properties

NrOfCameras

Returns

Long

Description

This property returns the number of stored Cameras in document.

Example

```
Dim A As Application
Dim D As Document
Dim Cs As Cameras
Dim N As Long

Det A = New Application
A.GetActiveDocument D
If Not (D is Nothing) Then
    D.GetCameras Cs
    MsgBox "Document contains " + Cs.NrOfCameras + "stored cameras"
End If
```

Methods

GetFirst

Parameters

C As Camera

Description

Returns the first Camera in a list of document's stored cameras.

Example

```
Dim A As Application
Dim D As Document
```



```

Dim Cs As Cameras
Dim C As Camera

set A = New Application
A.GetActiveDocument D
If Not (D Is Nothing) then
    D.GetCameras Cs
    Cs.GetFirst C
    Do While Not(C Is Nothing)
        MsgBox "Document contains camera " + C.Name
        Cs.GetNext C
    Loop
End If

```

GetNext

Parameters

C As Camera

Description

Returns the next Camera in a list of document's stored cameras.

Example

See GetFirst.

GetCamera

Parameters

Name As String
C As Camera

Description

Returns the camera with a name specified by the argument string or Nothing if there is no such a camera.

Example

```

Dim A As Application
Dim Doc As Document
Dim Cs As Cameras
Dim C As Camera

set A = New Application
A.GetActiveDocument Doc
Doc.GetCameras Cs
Cs.GetCamera "Main", C

```

AddCamera

Parameters

C As Camera

Description

Adds a new camera to a set of stored cameras of a document or replace a camera with the same name.

Note: the drawing should be opened in the 3D window for this method to work correctly.

Example

```

Dim A As Application
Dim Doc As Document
Dim Cs As Cameras
Dim C As Camera

set A = New Application
A.GetActiveDocument Doc
Doc.GetCameras Cs

```

```
Set C = New Camera ' default camera with perspective projection  
and top view is created  
C.Name = "Main"  
Cs.AddCamera C
```

DeleteCamera

Parameters

Name As String

Description

Removes a camera from the set of saved cameras of a document.

Example

```
Dim A As Application  
Dim Doc As Document  
Dim Cs As Cameras  
Dim C As Camera  
  
set A = New Application  
A.GetActiveDocument Doc  
Doc.GetCameras Cs  
Cs.DeleteCamera "Main"
```

Chapter 101 - XMLDocument

Description

An object of this class describes an XML document in the sense of the XML standard. Objects of this class cannot be created by `Set New` statement; this statement merely stores `Nothing`. Objects of this class can be created by methods `Application.CreateXMLDocument` and `Application.ReadXMLDocument`.

Example

```
Dim A As Application, Doc As XMLDocument, Name As String
Dim Code As XMLCompletionCode, Position() As Location
Dim Line As Long, Column As Long, I As Integer, S As String

Set Doc = _
    A.ReadXMLDocument ("C:\Parcels.xml", Completion:=Code, _
        Line:=Line, Column:=Column)
If Code = xmleSuccess Then
    ' Analyze data in Doc
    Set Doc = Nothing ' Clear read data
Else
    S = "Error " & Code & " in line " & Line & " column " & Column
    If A.GetXMLErrorPosition(Position) Then
        For I = LBound(Position) To UBound(Position)
            If Not A.UTF8ToASCII (Position(I).Entity, Name) Then
                Name = "(cannot be shown in current locale)"
            End If
            S = S & " -> " & Name &
            S = S & " line " & Position(I).Line
            S = S & " column " & Position(I).Column
        Next
    End If
    MsgBox S
End If
```

Properties

Version
Standalone
Comments1
Comments2
Main
Comments3

Methods

Clear
WhiteSpaceNormalize
Save
GetAsString

Properties

Version

Returns

XMLDocumentVersion

Description

Sets or returns a version of XML standard the document complies with. Changing version for a non empty document is not recommended because it can cause errors if some existing data do not comply with a version set. The main difference between versions is different sets of permitted characters in text values. But an error can occur only when using characters that normally are not used.

Example

```
Dim Doc As XMLDocument

Set Doc = New XMLDocument
```

```
Doc.Version = xmlV1_0
```

Standalone

Returns

Boolean

Description

Sets or returns a standalone property of the document. The current version of Pythagoras processes only standalone documents. Therefore, setting this property influences only what value of standalone argument the document will be written with.

Example

```
Dim Doc As XMLDocument

Set Doc = New XMLDocument
Doc.Standalone = False
' We wants that the output file will have attribute
' "standalone='no'" in the prolog
```

Comments1

Returns

XMLItems

Description

Returns the queue of comments and PIs standing between an XML declaration (document header) and a DTD. If DTD is absent a queue returned will contains all comments and PIs standing between an XML declaration and the main element.

Example

```
Dim Doc As XMLDocument
Dim Misc1 As XMLItems, Misc2 As XMLItems, Misc3 As XMLItems

Set Misc1 = Doc.Comments1
Set Misc2 = Doc.Comments2
Set Misc3 = Doc.Comments3
```

Comments2

Returns

XMLItems

Description

Returns the queue of comments and PIs standing between a DTD and the main element. If DTD is absent an empty queue is returned.

Example

See example Comments1

Comments3

Returns

XMLItems

Description

Returns the queue of comments and PIs standing after the main element.

Example

See example Comments1

Main

Returns

XMLElement

Description

Returns the main element of the document.

Example

```
Dim Doc As XmlDocument

MsgBox _
    "Name of the main element of the document is " & Doc.Main.Name
```

Methods

Clear

Parameters

None

Description

All document content is deleted. However, the name of the main element is retained.

Example

```
Dim A As Application, Doc As XmlDocument

Set Doc = A.ReadXMLDocument ("c:\Old.xml")
If Not Doc Is Nothing Then
    Doc.WhiteSpaceNormalize ' eliminate all not necessary whites
    ' Analyze data in Doc
Else
    MsgBox "XML document cannot be read"
End If
Doc.Clear
' Fill Doc with some information
If Doc.Save ("c:\New.xml") <> xmlcSuccess Then
    MsgBox "XML document cannot be written"
End If
Set Doc = Nothing
```

WhiteSpaceNormalize

Parameters

None

Description

White space normalization is done recursively through the whole document content. White space normalization process is described in details in the XML standard, see also description of `Application.XMWhiteSpaceNormalize` method in this document. This method recursively calls `XMWhiteSpaceNormalize` methods for the main element and for all comments and PIs standing out of the main element.

Example

See example `Clear`

Save

Parameters

```
FullFilename As String,
Optional WriteComments As Boolean = True,
Optional WritePI As Boolean = True,
Optional Encoding As IANACharset = csUTF8,
Optional LineSeparator As String = vbCrLf,
Optional MaxLine As Long = 80,
    ' Maximum line length
    ' (in characters except end of line characters)
Optional Indentation As Byte = 2,
    ' Number of spaces to indent on one level
Optional Quote As String = Chr(34)
    ' A character used to delimit attribute values
```

Returns

XMLCompletionCode

Description

The method writes content of the document to the file. The document remains untouched. Parameters `WriteComments` and `WritePI` define whether comments and PIs correspondingly will be written to the file.

Encoding must be equal to one of supported by Windows. It is strongly recommended that Encoding will be one of Unicode encodings. Otherwise characters could be present that cannot be converted and they will be replaced. Non Unicode encodings should be used only when such encoding is necessary for an application that will read an XML file. `LineSeparator` is a string used to separate lines of an output text file. Only the following values are allowed: `<LF>`, `<CR><LF>`, `<CR>`, `<CR><NEL>`, `<NEL>` and `<LS>`. Last three values can only be specified when version of the document is 1.1 and Encoding specifies a character set containing `<NEL>` or `<LS>` correspondingly. Note that any Unicode encoding contains these characters. Mentioned values can be coded in VBA in the following way correspondingly: `vbLf`, `vbCrLf`, `vbCr`, `vbCr & Chr(194) & Chr(113)`, `Chr(194) & Chr(133)` and `Chr(236) & Chr(128) & Chr(168)`. Note that the Unicode standard recommends `<LS>` as a line separator, however this recommendation is rarely followed. Both `MaxLine` and `Indentation` must be positive and are used to finely format an output file.

Quote must be either a single or a double quote.

Example

See example `Clear`

GetAsString

Parameters

Optional `IncludeComments` As Boolean = False,
Optional `IncludePI` As Boolean = False

Returns

String

Description

The method returns a textual XML document written in one string which content is identical to content the XML document contained in the class object.

Parameters `IncludeComments` and `IncludePI` are analogs of parameters `WriteComments` and `WritePI` of method `Save`, see its description.

Example

```
Dim Doc As XMLDocument, S As String

S = Doc.GetAsString()
```


Chapter 102 - XMLItem

Description

An object of this class describes one of the following XML objects: a PI, a comment, an element or a piece of text being the element content.

Objects of this class cannot be created by `Set New` statement. This statement merely stores `Nothing`. Objects of this class can be created by methods `XMLItems.AddText`, `XMLItems.AddPI`, `XMLItems.AddComment` and `XMLItems.AddElement`.

Example

```
Dim Doc As XMLDocument, Item As XMLItem

Set Item = Doc.Main.SubItems.FirstItem
' Get the first subitem of the main element of Doc
While Not Item Is Nothing
    ' ... process an item ...
    Set Item = Doc.Main.SubItems.NextItem
WEnd
```

Properties

ItemType
Text
PI
Comment
Element
Document
NextItem
WhiteSpace
Parent

Properties

ItemType

Returns

XMLItemType

Description

The property returns type of the item.

Example

```
Dim I As XMLItem
If I.ItemType = xmltText Then
    MsgBox "Content is " & I.Content
If
```

Text

Returns

XMLText

Description

The property returns the same item but represented as an object of class `XMLText`. If the item is not a text item, `Nothing` is returned.

Example

```
Dim A As Application, I As XMLItem, T As XMLText
Set T = I.Text
If Not T Is Nothing Then
    MsgBox "Content is " & A.UTF8ToASCIIV(T.Content)
End If
```

PI

Returns

XMLPI

Description

The property returns the same item but represented as an object of class XMLPI. If the item is not a PI, Nothing is returned.

Example

```
Dim A As Application, I As XMLItem, PI As XMLPI
Set PI = I.PI
If Not PI Is Nothing Then
    MsgBox "PI target is " & A.UTF8ToASCIIV(PI.Target)
End If
```

Comment

Returns

XMLComment

Description

The property returns the same item but represented as an object of class XMLComment. If the item is not comment, Nothing is returned.

Example

```
Dim A As Application, I As XMLItem, C As XMLComment
Set C = I.Comment
If Not C Is Nothing Then
    MsgBox "Comment content is " & A.UTF8ToASCIIV(C.Content)
End If
```

Element

Returns

XMLElement

Description

The property returns the same item but represented as an object of class XMLElement. If the item is not an element, Nothing is returned.

Example

```
Dim I As XMLItem, E As XMLElement
Set E = I.Element
If Not E Is Nothing Then
    MsgBox "Element name is " & E.Name
End If
```

Document

Returns

XMLDocument

Description

The property returns an XML document the item is a part of.

Example

```
Dim I As XMLItem
If I.Document.Version = xmlV1_1 Then
    ' ...
End If
```

NextItem

Returns

XMLItem

Description

The property returns an item standing after that item in a queue. It can be used to scan items in the queue.

Example

```
Dim Doc As XMLDocument, Item As XMLItem
```

```

Set Item = Doc.Main.SubItems.FirstItem
' Get the first subitem of the main element of Doc
While Not Item Is Nothing
    ' process an item
    Set Item = Item.NextItem
WEnd

```

WhiteSpace

Returns

XMLWhiteSpaceFacet

Description

Sets or returns a value defining how spaces will be processed during normalization. For textual items it is ignored. For elements it defines normalization process for textual subitems. By default this property is set to xmlCollapse.

Example

```

Dim Element As XMLLelement

Element.WhiteSpace = xmlReplace
' Define that in a subsequent call of WhiteSpaceNormalize method
' blanks will not be deleted in textual content of this element
XMLItems

```

Description

Each object of this class describes a queue of XML items. Objects of this class cannot be created by Set New statement. This statement merely stores Nothing. An object of this class is created together with its parent (either an XML document or an element).

Parent

Returns

XMLLelement

Description

The parent of the XML Item is returned. When the item is the main document Nothing is returned.

Example

```

Dim I As XMLItem, ParentName As String

If I.Parent Is Nothing Then
    ParentName = ""
Else
    ParentName = I.Parent.Name
End If

```


Chapter 103 - XMLItems

Description

Only one object of this class exists for an item queue. Therefore, when getting an object of this class twice (using property `Element.Items`, for example), the same object is returned. Embedded or parallel scanning is possible using `XMLItem.Next` property.

Example

See example class `XMLItem`

Properties

`Parent`
`NrOfItems`
`CurrentItem`
`FirstItem`
`NextItem`
`NrOfElements`
`FirstElement`
`NextElement`

Methods

`Clear`
`Copy`
`WhiteSpaceNormalize`
`AddText`
`AddPI`
`AddComment`
`AddElement`
`RemoveItem`

Properties

Parent

Returns

`Variant (XMLElement or XMLDocument)`

Description

Returns an element or an XML document this item directly belongs to. A document is returned only for PIs and comments being outside the main document.

Example

```
Dim Queue As XMLItems, P As Variant, E As XMLElement
Dim OfCoordinates As Boolean

' define if Queue is a queue of an element with name "Coordinates"
Set P = Queue.Parent
OfCoordinates = False
If TypeName(P) = "XMLElement" Then
    Set E = P
    OfCoordinates = E.Name = "Coordinates"
End If
```

NrOfItems

Returns

`Long`

Description

Returns the number of items in the queue. Note that number of items is not stored anywhere. It will be calculated each time you access this property.

Example

```
Dim Queue As XMLItems, N As Long

N = Queue.NrOfItems
```

CurrentItem

Returns

XMLItem

Description

Returns the current item in the queue. Initially it is the first item of a queue. Subsequent methods can change the current item. When the current item is undefined Nothing is returned.

FirstItem

Returns

XMLItem

Description

Returns the first item of the queue. The returned item becomes current. If the queue is empty Nothing is returned.

Example

See example class XMLItem

NextItem

Returns

XMLItem

Description

Returns the next item after the current one. The returned item becomes current. If the current item was the last, Nothing is returned.

Example

See example class XMLItem

NrOfElements

Returns

Long

Description

Returns the number of elements in the queue. Note that the number of elements is not stored anywhere. It will be calculated each time you access this property.

Example

```
Dim Queue As XMLItems, N As Long
```

```
N = Queue.NrOfElements
```

FirstElement

Returns

XMLElement

Description

Returns the first item being an element. The returned item becomes current. If the queue contains no elements then Nothing is returned.

Example

```
Dim A As Application, Doc As XMLDocument, Element As XMLElement  
Dim S As String
```

```
S = ""  
Set Element = Doc.Main.SubItems.FirstElement  
' Get the first subelement of the main element of Doc  
While Not Element Is Nothing  
    S = S & " " & A.UTF8ToASCIIV(Element.Name)  
    Set Element = Doc.Main.SubItems.NextElement  
WEnd
```

```
MsgBox "The document contains subelements with names:" + S
```

NextElement

Returns

XMLElement

Description

Returns the first item after the current one being an element. The returned item becomes current. If the queue contains no more elements then Nothing is returned.

Example

See example FirstElement

Methods

Clear

Parameters

None

Description

All queue items are destroyed, the queue becomes empty.

Example

```
Dim Doc As XMLDocument

Doc.Main.SubItems.Clear ' Delete the whole main element content
```

Copy

Parameters

Dest As XMLItems

Description

Dest is cleared, then the current queue is copied to Dest. For element items both attributes and subitems of all levels are copied.

Example

```
Dim Doc1 As XMLDocument, Doc2 As XMLDocument

Doc1.Main.SubItems.Copy Doc2.Main.SubItems
' Copy the whole main element content
```

WhiteSpaceNormalize

Parameters

None

Description

For text in all text items white space normalization is provided accordingly to the element WhiteSpace property value. Also WhiteSpaceNormalize method is called for all other items. See also method Application.WhiteSpaceNormalize description.

Example

```
Dim Doc As XMLDocument

Doc.Main.SubItems.WhiteSpaceNormalize
```

AddText

Parameters

Content As String,
Optional Position As PTPosition = ptEnd,
Optional UniteText As Boolean = True,
Optional SpaceBefore As Boolean = False

Returns

XMLText

Description

A new text item is inserted. Inserted text is inserted as it is without white space normalization.

A new text item is inserted correspondingly to `Position` argument value: either before the first element of the queue, or before the current item, or after the current item, or after the last item. In the third case the inserted item becomes current.

If `UniteText = True` and before or after the inserted item another text item stands then the inserted item is united with that item. `UniteText = False` could be useful when the programmer wants to insert another item after the just inserted one.

`SpaceBefore` flag signals that space should be present before inserted text; see also description of `XMLText.SpaceBefore` property.

The inserted (and possibly united) text item is returned.

Note. The XML standard declares that any line separator should be converted to `<LF>` during input. Therefore any text parameters should not contain other line separators than `<LF>`s.

Example

```
Dim Element As XMLElement, Text As XMLText

Element.SubItems.Clear
Set Text = Element.SubItems.AddText "3.14"
' Set content of Element to the textual value
```

AddPI

Parameters

Target As String,
Optional Content As String = "",
Optional Position As PTPosition = ptEnd,
Optional WhiteSpace As XMLWhiteSpaceFacet = xmlCollapse

Returns

XMLPI

Description

A new processing instruction is inserted. Target and Content must correspond to the following requirements of the XML standard: Target has to be a well-formed name not equal to "XML" (in any letter case), Content has not to contain character sequence `"?>"`. Content is inserted as it is without white space normalization.

The new PI is inserted correspondingly to `Position` argument value: either before the first element of the queue, or after the current item, or after the last item. In the second case the PI becomes current item.

The inserted PI is returned.

See also note `AddText`.

Example

```
Dim Element As XMLElement, PI As XMLPI

Set PI = Element.SubItems.AddPI "php", "echo $a;"
```

AddComment

Parameters

Content As String,
Optional Position As PTPosition = ptEnd,
Optional WhiteSpace As XMLWhiteSpaceFacet = xmlCollapse

Returns

XMLComment

Description

A new comment is inserted. Content must correspond to the requirement of the XML standard that it has not to contain character sequence "--". Content is inserted as it is without white space normalization.

The new comment is inserted correspondingly to Position argument value: either before the first element of the queue, or after the current item, or after the last item. In the second case the comment becomes the current item.

The inserted comment is returned.

See also note AddText.

Example

```
Dim Element As XMLElement, Comment As XMLComment

Set Comment = _
    Element.SubItems.AddComment _
        "This element describes an enclosing rectangle"
```

AddElement

Parameters

Name As String,
Optional Content As String = "",
Optional Position As PTPosition = ptEnd,
Optional WhiteSpace As XMLWhiteSpaceFacet = xmlCollapse

Returns

XMLElement

Description

A new element is inserted. Name must be a well-formed name in sense of the XML standard. The inserted element contains no attribute. If parameter Content is not empty then the element contains only one text subitem with the specified text, otherwise the element contains no subitem.

The new element is inserted correspondingly to Position argument value: either before the first element of the queue, or after the current item, or after the last item. In the second case the comment becomes the current item.

The inserted element is returned.

See also note AddText.

Example

```
Dim Element As XMLElement, SubElement As XMLElement

Set SubElement = Element.SubItems.AddElement "triangle"
```

RemoveItem

Parameters

Optional UniteText As Boolean = True

Description

The current item is deleted. If it is an element it is deleted together with all subitems. The next item becomes current.

If the current item is not defined then nothing is done.

Parameter UniteTexts: see description of UniteTexts parameter of AddText method.

Example

```
Dim Element As XMLElement

Element.SubItems.AddElement "triangle"
```


Chapter 104 - XMLText

Description

An object of this class describes a piece of text being inside an element content and limited on the left side either by an element start tag or by a subelement or by a comment or by a PI. On the right side it is limited either by a subelement or by a comment or by a PI or by an element end tag.

Objects of this class cannot be created by `Set New` statement. This statement merely stores `Nothing`. Objects of this class can be created by method `XMLItems.AddText`.

Example

See example class `XMLItem.Text`

Properties

`Item`
`NextItem`
`Document`
`Content`
`SpaceBefore`
`Parent`

Properties

Item

Returns

`XMLItem`

Description

Returns the same textual item but represented as an object of class `XMLItem`. It is useful where other type is syntactically required.

Example

```
Dim Item As XMLItem, Text As XMLText, Same As Boolean

Same = Item = Text.Item ' Check that they are the same item
```

NextItem

Returns

`XMLItem`

Description

The property returns an item standing after that item in a queue. It can be used to scan items in the queue.

Example

See `XMLItem.NextItem`

Document

Returns

`XMLDocument`

Description

The property returns the XML document the text is a part of.

Example

See `XMLItem.Document`

Content

Returns

`String`

Description

Sets or returns the textual content of the item.
See also note `XMLItem.AddText`.

Example

```
Dim Text As XMLText

Text.Content = Text.Content & " *"
' Add a blank and asterisk to the text
```

SpaceBefore

Returns

Boolean

Description

Sets or returns a special flag value. The flag has a meaning only when the parent document has `WhiteSpace = xmlCollapse` and has mixed content. This flag signals that content of the item should be space separated from content of the previous text item. Let us illustrate this by the following example. An XML element is like `<list>1<stop/> 5</list>`, it contains two textual items separated by a subelement. After white space collapse normalization content of these items will be "1" and "5" respectively. Content of the element will be "15" what is not correct. It should be "1 5" instead. To avoid such mistakes, the second text item should be marked by a `SpaceBefore` flag equal to `True` during white space collapse normalization. This property should be explicitly used in exceptional cases only.

Parent

Returns

XMLElement

Description

The parent `XMLElement` of the text is returned.

Example

See example `XMLItem.Parent` property.

Chapter 105 - XMLPI

Description

An object of this class describes a so called processing instruction.

Objects of this class cannot be created by `Set New` statement. This statement merely stores `Nothing`. Objects of this class can be created by method `XMLItems.AddPI`.

Example

See example class `XMLItem.PI`

Properties

`Item`
`NextItem`
`Document`
`Target`
`Content`
`WhiteSpace`
`Parent`

Properties

Item

Returns

`XMLItem`

Description

Returns the same textual item but represented as an object of class `XMLItem`. It is useful where other type is syntactically required.

Example

```
Dim Item As XMLItem, PI As XMLPI, Same As Boolean
```

```
Same = Item = PI.Item ' Check that they are the same item
```

NextItem

Returns

`XMLItem`

Description

The property returns an item standing after that item in a queue. It can be used to scan items in the queue

Example

See `XMLItem.NextItem`

Document

Returns

`XMLDocument`

Description

The property returns an XML document the PI is a part of.

Example

See `XMLItem.Document`

Target

Returns

`String`

Description

Sets or returns a target of a PI. New target must be a well-formed XML name not being "XML" (in any letter case).

Example

```
Dim PI As XMLPI

PI.Target = "php"
```

Content

Returns

String

Description

Sets or returns the content of a PI, called also a command. The new content must not contain ">" inside.

See also note `XMLItem.AddText`.

Example

```
Dim PI As XMLPI

PI.Content = "6 34"
```

WhiteSpace

Returns

XMLWhiteSpaceFacet

Description

Sets or returns a value defining how spaces will be processed during normalization.

Initially this property is set to `xmlCollapse`.

Example

See `XMLItem.Whitespace`

Parent

Returns

XMLElement

Description

If the processing instruction is a part of some XML element this element is returned.

Otherwise (when the processing instruction is before or after the main document element)

Nothing is returned.

Example

See example `XMLItem.Parent` property.

Chapter 106 - XMLComment

Description

An object of this class describes a comment.

Objects of this class cannot be created by `Set New` statement. This statement merely stores `Nothing`. Objects of this class can be created by method

`XMLItems.AddComment`.

Example

See example class `XMLItem.PI`

Properties

`Item`

`NextItem`

`Document`

`Content`

`WhiteSpace`

`Parent`

Properties

Item

Returns

`XMLItem`

Description

Returns the same textual item but represented as an object of class `XMLItem`. It is useful where other type is syntactically required.

Example

```
Dim Item As XMLItem, Comment As XMLComment, Same As Boolean
```

```
Same = Item = Comment.Item ' Check that they are the same item
```

NextItem

Returns

`XMLItem`

Description

The property returns an item standing after that item in a queue. It can be used to scan items in the queue.

Example

```
See XMLItem.NextItem
```

Document

Returns

`XMLDocument`

Description

The property returns the XML document the comment is a part of.

Example

```
See XMLItem.Document
```

Content

Returns

`String`

Description

Sets or returns the comment text. A new comment must not contain "--" inside.

See also the note in `XMLItem.AddText`.

Example

Dim Comment As XMLComment

Comment.Content = "It is an enclosing circle"

Whitespace

Returns

XMLWhiteSpaceFacet

Description

Sets or returns a value defining how spaces will be processed during normalization. It defines normalization process for the textual subitems. Initially this property is set to `xmlCollapse`.

Example

See `XMLItem.Whitespace`

Parent

Returns

XMLElement

Description

The parent `XMLElement` of the comment is returned. When the comment is in the main document element `Nothing` is returned.

Example

See example `XMLItem.Parent` property.

Chapter 107 - XMLElement

Description

An object of this class describes an XML element.

Objects of this class cannot be created by `Set New` statement. This statement merely stores `Nothing`. Main elements of XML documents are created together with their documents. Other objects of this class can be created by method `XMLItems.AddElement`.

Example

See example `XMLItems.FirstElement`

Properties

`Item`
`NextItem`
`Document`
`NextElement`
`NrOfAttribites`
`FirstAttribute`
`WhiteSpace`
`Name`
`Value`
`SubItems`
`Parent`

Methods

`GetAttribute`
`AddAttribute`
`RemoveAttribute`
`RemoveAttributes`
`CopyAttributes`
`Clear`
`Copy`
`WhiteSpaceNormalize`

Properties

Item

Returns

`XMLItem`

Description

Returns the same element but represented as an object of class `XMLItem`. It is useful where other type is syntactically required.

Example

```
Dim Item As XMLItem, Element As XMLElement, Same As Boolean

Same = Item = Element.Item ' Check that they are the same item
```

NextItem

Returns

`XMLItem`

Description

The property returns an item standing after that item in a queue. It can be used to scan items in the queue

Example

See `XMLItem.NextItem`

Document

Returns

XMLDocument

Description

The property returns the XML document the item is a part of.

Example

See XMLItem.Document

Whitespace

Returns

XMLWhiteSpaceFacet

Description

Sets or returns a value defining how spaces will be processed during normalization. It defines normalization process for the textual subitems. Initially this property is set to xmlCollapse.

Example

See XMLItem.Whitespace

NextElement

Returns

XMLElement

Description

Returns the next element in the queue. If there are no more elements in the queue Nothing is returned.

Example

```
Dim A As Application, Doc As XMLDocument, Element As XMLElement
Dim S As String

S = ""
Set Element = Doc.Main.SubItems.FirstElement
' Get the first subelement of the main element of Doc
While Not Element Is Nothing
    S = S & " " & A.UTF8ToASCIIV(Element.Name)
    Set Element = Element.NextElement
WEnd
MsgBox "The document contains subelements with names:" & S
```

NrOfAttributes

Returns

Long

Description

Returns the number of attributes.

Example

See example FirstAttribute

FirstAttribute

Returns

XMLAttribute

Description

The first attribute of the element is returned. The attributes are ordered in the lexicographic order of their names coded in Unicode. If the element contains no attribute, Nothing is returned. This property can be used together with XMLAttribute.Next property to look through all element attributes.

Example

```
Dim Element As XElement, Attr As XMLAttribute
Dim S As String

S = _
    "Element " & Element.Name & " contains " & _
    Element.NrOfAttributes & " attributes:" & vbCrLf
Set Attr = Element.FirstAttribute
While Not Attr Is Nothing
    S = S & Attr.Name & " = " & Attr.Value
    Select Case Attr.WhiteSpace
        Case xmlReplace
            S = S & " (replace)"
        Case xmlCollapse
            S = S & " (collapse)"
    End Select
    S = S & vbCrLf
    Set Attr = Attr.NextAttribute
WEnd
MsgBox S
```

Name

Returns

String

Description

Sets or returns the name of the element. The new name must be a well-formed XML name.

Example

```
Doc As XMLDocument
MsgBox "The document main element has name " & Doc.Main.Name
```

Value

Returns

String

Description

Returns the united text from all text subitems normalized accordingly to the WhiteSpace property value of the element.

For an example how to store the new value see XMLItems.AddText.

Example

```
Dim A As Application, Element As XElement, Height As Long

Height = A.XMLStringToInteger(Element.Value)
```

SubItems

Returns

XMLItems

Description

Returns a queue of all subitems.

Example

See example NextElement

Parent

Returns

XMLElement

Description

The parent of the XElement is returned. If the XElement is the Main element then Nothing is returned.

Example

See example `XMLItem.Parent` property.

Methods

GetAttribute

Parameters

Name As String

Returns

Variant(String)

Description

Returns a value of the attribute with the specified name. If an attribute with such name is absent, Empty is returned.

Example

```
Dim Element As XMLElement, V As Variant, InInches As Boolean
V = Element.GetAttribute ("Unit")
InInches = (VarType(V) = vbString) And (V = "Inch")
```

AddAttribute

Parameters

Name As String, Value As String,
Optional WhiteSpace As XMLWhiteSpaceFacet = xmlCollapse

Description

Adds a new attribute with the specified name, value and white space facet. Name must be a well-formed XML name. If an attribute with such name is already present, its value and white space facet are changed to the specified ones.

See also note `XMLItem.AddText`.

Example

```
Dim Element As XMLElement
Element.AddAttribute ("Unit", "Inch")
```

RemoveAttribute

Parameters

Name As String

Description

If an attribute with such name is found it is deleted.

Example

```
Dim Element As XMLElement
Element.RemoveAttribute ("Unit")
```

RemoveAttributes

Parameters

None

Description

All attributes are deleted.

Example

```
Dim Element As XMLElement
Element.RemoveAttributes
```

CopyAttributes

Parameters

Dest As XMLElement

Description

All attributes of `Dest` are deleted. Then all attributes are copied from this element to `Dest`.

Example

```
Dim E1 As XElement, E2 As Element
E1.CopyAttributes E2
```

Clear

Parameters

None

Description

All content of the element is deleted. Particularly all attributes and all subitems are deleted. However a name and a white space facet of the element are retained.

Example

```
Dim Element As XElement
Element.Clear
```

Copy

Parameters

`Dest As XElement`

Description

All content of `Dest` is deleted, then all content of this element is copied to `Dest`. Particularly a name, all attributes, a white space facet and all subitems are copied.

Example

```
Dim E1 As XElement, E2 As Element
E1.Copy E2
```

WhiteSpaceNormalize

Parameters

None

Description

`WhiteSpaceNormalize` method is called for the subitem queue (see its description) and all attribute values are normalized as well.

Example

```
Dim Doc As XmlDocument

Doc.Main.WhiteSpaceNormalize
```


Chapter 108 - XMLAttribute

Description

An object of this class describes an XML attribute.

Objects of this class cannot be created by `Set New` statement. This statement merely stores `Nothing`. Objects of this class can be created by method

`XMLElement.AddAttribute`.

Example

See example `XMLElement.FirstAttribute`

Properties

`Element`

`Name`

`Value`

`WhiteSpace`

`NextAttribute`

Properties

Element

Returns

`XMLElement`

Description

Returns the element the attribute belongs to.

Example

```
Dim A As XMLAttribute

If A.Element.Document.Version = xmlV1_1 Then
    ' ...
End If
```

Name

Returns

`String`

Description

Returns the name of the attribute.

Example

See example `XMLAttributes`

Value

Returns

`String`

Description

Sets or returns a value of the attribute.

Example

See example `XMLAttributes`

WhiteSpace

Returns

`XMLWhiteSpaceFacet`

Description

Sets or returns a value of white space facet of the attribute.

Example

See example `XMLAttributes`

NextAttribute

Returns

`XMLAttribute`

Description

Returns the next attribute. Attributes are ordered in lexicographical order of their names written in Unicode. If there are no more attributes `Nothing` is returned.

Example

See example `XMLElement.FirstAttribute`

Chapter 109 - KML Introduction

Purpose

The KML Objects are a collection of objects, properties and methods to create geometrical objects and save them in Google Earth KML format.

The objects `KMLFolder`, `KMLFeature`, `KMLFeatures`, `KMLPlacemark`, `KMLGeometry`, `KMLPoint`, `KMLLineString`, `KMLPolygon`, `KMLInnerBoundary`, `KMLCoordinates` and `KMLCoordinate` are used to group geometrical objects into `KMLFolders` and to specify KML coordinates and polygon outer and inner boundaries.

The Pythagoras Object Model and the KML Object Model allow you to retrieve data from a Pythagoras drawing, to convert it to KML and to view the result in Google Earth or in other applications that support KML.

You should be familear with the principles and terminology of KML before using the KML object model.

Structure of KML Object

The general structure of KML object accepted by Pythagoras can be presented in BNF as follows:

```
Folder = [Name] Features
Features = {Feature}
Feature = Placemark | Folder
Placemark = Name Style Geometry
Style = LineStyle [PolyStyle]
IconStyle = Color Scale Name
LineStyle = Color Width
PolyStyle = Color Fill
Geometry = Point | LineString | Polygon
LineString = Extrude AltitudeMode Coordinates
Polygon = Extrude AltitudeMode OuterBoundary {InnerBoundary}
OuterBoundary = Coordinates
InnerBoundary = Coordinates {Coordinates}
Coordinates = Coordinate {Coordinate}
Coordinate = Longitude Latitude Altitude
Name = String
Color = AA BB GG RR (* AA - opacity, 00-FF; BB, GG, RR -
blue, green red, 00-FF *)
Width = CARDINAL (* in pixels *)
Fill = BOOLEAN
Extrude = BOOLEAN
```



```
AltitudeMode = ClampToGround | RelativeToGround | Absolute  
Longitude = RealNumber (* -180 <= Longitude <= 180 *)  
Latitude = RealNumber (* -90 <= Latitude <= 90 *)
```

Example

The following macro generates and saves a KML file and displays the result in Google Earth.

You need to have a drawing in a CRS (e.g. UTM, GK, ...) containing points. Only the points will be exported to KML.

```
Dim A As Application
Dim Doc As Document
Dim DestCRS As CoordinateReferenceSystem
Dim CRSs As CoordinateReferenceSystems

Sub PM_ExportPointsToGoogleEarth
    Dim mf As KMLFolder
    Dim dir As String
    Dim docName As String
    Dim fname As String
    Dim gei As Object
    Dim bres As Long, backSlash As Long

    Set A = New Application
    A.GetCoordinateReferenceSystems CRSs
    CRSs.GetCoordinateReferenceSystem "WGS84", DestCRS

    A.GetActiveDocument Doc
    If Doc Is Nothing Then
        Exit Sub
    End If
    ProcessPoints mf
    If (Not (mf Is Nothing)) And (Not (Doc Is Nothing)) Then
        Set gei = CreateObject("GoogleEarth.ApplicationGE")
        If gei Is Nothing Then
            MsgBox "Google Earth is not installed"
            Exit Sub
        End If
        fname = "C:\TestExample.kml"
        mf.Save fname
        A.Wait 500
        Do
            bres = gei.IsInitialized
            If bres <> 0 Then Exit Do
        Loop
        gei.OpenKmlFile(fname, True)
        Set gei = Nothing
    End If
End Sub

Sub AddFeature(ByRef mfeats As KMLFeatures, afeat As KMLFeature, aname As String)
    Dim feat As KMLFeature
    Dim feats As KMLFeatures
    Dim name As String
    Dim folder As KMLFolder
    Dim mfi As KMLFeaturesIterator
    Dim found As Boolean

    Set mfi = mfeats.Iterator
    mfi.GetFirst feat
    Set folder = Nothing
    found = False
    While Not ((feat Is Nothing) Or found)
        If feat.FeatureType = ptKMLFolder Then
            feat.GetFolder folder
            If StrComp(folder.name, aname) = 0 Then
                found = True
            End If
        End If
        If Not found Then
            mfi.GetNext feat
        End If
    Wend
```

```

        If Not found Then
            Set folder = New KMLFolder
            folder.Name = aname
            folder.GetFeature feat
            mfeats.Add feat
            Set feats = New KMLFeatures
            Set folder.Features = feats
        End If
        Set feats = folder.Features
        feats.Add afeat
    End Sub

Sub DoPnt(pnt As Point, ByRef feat As KMLFeature)
    Dim p3 As XYZ
    Dim p2 As XY
    Dim LatLong As XY
    Dim kgp As KMLGeoPoint
    Dim pm As KMLPlacemark
    Dim geom As KMLGeometry
    Dim pt As KMLPoint
    Dim color As KMLColor

    Set feat = Nothing
    p3 = pnt.Coordinates

    color.Red = pnt.TrueColor.Red
    color.Green = pnt.TrueColor.Green
    color.Blue = pnt.TrueColor.Blue
    color.Alpha = 255

    If Doc.TransformPossible(DestCRS) Then
        p2.X = p3.X
        p2.Y = p3.Y
        LatLong = Doc.TransformXYTo(DestCRS, p2)
        kgp.Longitude = LatLong.Y
        kgp.Latitude = LatLong.X
        kgp.Altitude = p3.Z
        Set pt = New KMLPoint
        pt.GeoPoint = kgp
    Else
        Set feat = Nothing
        Exit Sub
    End If

    Set pm = New KMLPlacemark
    pm.Name = pnt.Layer

    pm.SetIconStyle color, 1.0, "C:\MyIcon"

    pt.GetGeometry geom
    Set pm.Geometry = geom
    pm.GetFeature feat
End Sub

Sub ProcessPoints(ByRef mf As KMLFolder)
    Dim obs As Selection
    Dim o As CadObject
    Dim N As Long
    Dim i As Long
    Dim pnt As Point
    Dim feat As KMLFeature
    Dim mfeats As KMLFeatures

    Doc.GetSelection obs
    Set mfeats = New KMLFeatures
    Set mf = New KMLFolder

    N = obs.Count(ptPoint)
    obs.GetFirst ptPoint, o
    For i = 1 To N
        If o Is Nothing Then
            ' No more objects
        Else
            o.GetPoint pnt
            DoPnt pnt, feat
        End If
    Next i
End Sub

```

```
        If Not (feat Is Nothing) Then
            AddFeature mfeats, feat, o.Layer
        End If
    End If
    obs.GetNext o
Next i

Set mf.Features = mfeats
End Sub
```

Chapter 110 - KMLFolder

Description

A KMLFolder object has a name, a list of KML Features, and a collection of methods to add and remove KML Features.

Example

```
Dim KF As KMLFolder

Set KF = New KMLFolder
KF.Name = "NewFolder"
```

Properties

Name
Features

Methods

GetFeature
Save

Properties

Name

Returns

String

Description

This property sets or returns the name of the KML Folder.

Example

```
Dim KF As KMLFolder

Set KF = New KMLFolder
MsgBox KF.Name
```

Features

Returns

KMLFeatures

Description

This property sets or returns a KMLFeatures object in the KML folder.

Example

```
Dim Features As KMLFeatures
Dim KF As KMLFolder

Set Features = New KMLFeatures
Set KF = New KMLFolder
Set KF.Features = Features
```

Methods

GetFeature

Parameters

Feature As KMLFeature

Description

This method sets the reference to the KMLFolder in a variable of type KMLFeature. Note that the reference is copied, not the object.

Example

```
Dim Feature As KMLFeature
Dim KF As KMLFolder
```

```
Set KF = New KMLFolder
KF.GetFeature Feature
```

Save

Parameters

FileName As String

Description

The contents of the KMLFolder object is saved as a file in KML format.

Example

```
Dim KF As KMLFolder

Set KF = New KMLFolder
KF.Save "C:\MyRegion.kml"
```

Chapter 111 - KMLFeatures

Description

A KMLFeatures object contains a list of KMLFeature objects and contains properties and methods to manage this list.

Example

```
Dim FS As KMLFeatures

Set FS = New KMLFeatures
```

Properties

NrOfFeatures
IsEmpty
Iterator

Methods

Add
Clear

Properties

NrOfFeatures

Returns

Long

Description

This property returns the number of KMLFeature objects in the list.

Example

```
Dim FS As KMLFeatures
Dim n As Long

n = FS.NrOfFeatures
```

IsEmpty

Returns

Boolean

Description

This property returns True if the list of KMLFeature objects is empty.

Example

```
Dim FS As KMLFeatures

If Not FS.IsEmpty Then
    '...
End If
```

Iterator

Returns

KMLFeaturesIterator

Description

This property creates and returns a KMLFeaturesIterator object.

Example

```
Dim FS As KMLFeatures
Dim FSI As KMLFeaturesIterator

Set FSI = FS.Iterator
```

Methods

Add

Parameters

Feature As KMLFeature

Description

This method adds a KMLFeature object to the list.

Example

```
Dim FS As KMLFeatures
Dim F As KMLFeature
Dim i As Integer

For i = 1 To 5
    Set F = New KMLFeature
    FS.Add F
Next i
```

Clear

Parameters

None

Description

This method clears the KMLFeatures object. The list will become empty.

Example

```
Dim FS As KMLFeatures

FS.Clear
```


Chapter 112 - KMLFeaturesIterator

Description

A `KMLFeaturesIterator` object is an auxiliary object used to traverse a `KMLFeatures` object. Multiple `KMLFeaturesIterators` can be used simultaneously. An iterator becomes invalid when another iterator uses `Insert` or `Remove`. Invalid iterators can not be used anymore.

Example

```
Dim FS As KMLFeatures
Dim FSI As KMLFeaturesIterator

Set FS = New KMLFeatures
Set FSI = FS.Iterator
```

Properties

`Valid`
`EndOfList`

Methods

`GetFirst`
`GetNext`
`Insert`
`Remove`

Properties

Valid

Returns

`Boolean`

Description

This property returns `False` if the iterator is invalid and may not be used anymore. (See description at the start of this chapter.)

Example

```
Dim FSI As KMLFeaturesIterator

If FSI.Valid Then
    ' ....
End If
```

EndOfList

Returns

`Boolean`

Description

This property returns `True` if the iterator is at the end of the list.

Example

```
Dim FSI As KMLFeaturesIterator

While Not FSI.EndOfList
    ' ....
Loop
```

Methods

GetFirst

Parameters

`Feature As KMLFeature`

Description

Returns the first `KMLFeature` in the list.

Example

```
Dim FSI As KMLFeaturesIterator
Dim F As KMLFeature

FSI.GetFirst F
Do While Not FSI.EndOfList
    FSI.GetNext F
Loop
```

GetNext

Parameters

Feature As KMLFeature

Description

Returns the next Feature in the list.

Example

See GetFirst.

Insert

Parameters

Feature As KMLFeature, Position As Long

Description

A KMLFeature is inserted in the list at the position specified by parameter Position.

The valid values of Position are: PtBegin,PtEnd,ptBefore or ptAfter.

ptBefore and ptAfter refer to the positions just before or just after the Feature obtained by the methods GetFirst and GetNext. The current position in the list will be the position of the element that is last inserted.

Example

```
Dim FSI As KMLFeaturesIterator
Dim F As KMLFeature
Dim NewF As KMLFeature

FSI.GetFirst F
Do While Not FSI.EndOfList
    If SomeCondition(F) Then
        FSI.Insert NewF, ptBefore
        Exit Do
    End If
    FSI.GetNext F
Loop
```

Remove

Parameters

F As KMLFeature

Description

This method removes the specified feature from the list.

Example

```
Dim FSI As KMLFeaturesIterator
Dim F As KMLFeature

FSI.GetFirst F
If SomeCondition(F) Then
    FSI.Remove F
End If
```

Chapter 113 - KMLFeature

Description

A KMLFeature object is a generic object encompassing KMLPlaceMark and KMLFolder objects. When the type of a KMLFeature object is known, the KMLFeature can be converted to its specific object type. The methods GetPlacemark and GetFolder do not create a new object, but assign instead the reference to the object of a specific type.

Example

```
Dim F As KMLFeature
Dim Folder As KMLFolder
Dim Placemark As KMLPlacemark

If F.FeatureType = ptKMLFolder Then
    F.GetFolder Folder
Else 'ptKMLPlacemark
    F.GetPlacemark Placemark
End If
```

Properties

FeatureType

Methods

GetPlacemark
GetFolder

Properties

FeatureType

Returns

Long

Description

This property returns the type of the KMLFeature object: ptKMLPlacemark or ptKMLFolder.

Example

```
Dim F As KMLFeature
Dim Folder As KMLFolder

If F.FeatureType = ptKMLFolder Then
    F.GetFolder Folder
End If
```

Methods

GetPlacemark

Parameters

P As KMLPlacemark

Description

This method sets the reference to the object in a variable of type ptKMLPlacemark.

Example

See introduction

GetFolder

Parameters

F As KMLFolder

Description

This method sets the reference to the object in a variable of type ptKMLFolder.

Example

See introduction

Chapter 114 - KMLPlacemark

Description

A KMLPlacemark object is a geometry (see KMLGeometry) that has a Name and a Style.

Example

```
Dim F As KMLFeature
Dim Placemark As KMLPlacemark
Dim Geo As KMLGeometry

If F.FeatureType = ptKMLPlacemark Then
    F.GetPlacemark Placemark
    Set Geo = Placemark.Geometry
    MsgBox Placemark.Name
End If
```

Properties

Name
Geometry

Methods

GetFeature
SetLineStyle
GetLineStyle
SetPolyStyle
GetPolyStyle
SetPolyStyle

Properties

Name

Returns

String

Description

This property sets or returns the name of the KML Placemark.

Example

See introduction.

Geometry

Returns

KMLGeometry

Description

This property sets or returns the Geometry of the KML Placemark.

Example

```
Dim Placemark As KMLPlacemark
Dim Geo As KMLGeometry

Set Placemark.Geometry = Geo
```

Methods

GetFeature

Parameters

F As Feature

Description

This method sets the reference to the object in a variable of type KMLFeature.

Example

```
Dim F As KMLFeature
Dim Placemark As KMLPlacemark

Placemark.GetFeature F
```

SetLineStyle

Parameters

Color As KMLColor, Width As Double

Description

This method sets the color and the line width of the Line Strings or Polygon Boundaries in the Geometry of the Placemark.

Example

```
Dim PM As KMLPlacemark
Dim Color As KMLColor

Set PM = New KMLPlacemark
PM.SetLineStyle Color, 1.5
```

GetLineStyle

Parameters

Color As KMLColor, Width As Double

Description

This method returns the color and the line width of the Line Strings or Polygon Boundaries in the Geometry of the Placemark.

Example

```
Dim PM As KMLPlacemark
Dim Color As KMLColor
Dim Width As Double

PM.GetLineStyle Color, Width
```

SetPolyStyle

Parameters

Color As KMLColor, Fill As Boolean

Description

This method sets the color and the fill flag of the Polygon in the Geometry of the Placemark.

Example

```
Dim PM As KMLPlacemark
Dim Color As KMLColor

PM.SetPolyStyle Color, True
```

GetPolyStyle

Parameters

Color As KMLColor, Fill As Boolean

Description

This method returns the color and the fill flag of the Polygon in the Geometry of the Placemark.

Example

```
Dim PM As KMLPlacemark
Dim Color As KMLColor
Dim Fill As Boolean

PM.GetPolyStyle Color, Fill
```

SetIconStyle

Parameters

Color As KMLColor, Scale As Double, IconName As String

Description

This method sets the color, the scale and the name of the icon in the Geometry of the Placemark.

Example

```
Dim PM As KMLPlacemark
Dim Color As KMLColor
```

```
Set PM = New KMLPlacemark  
PM.SetIconStyle Color, 2.5, "MyIcon"
```

GetIconStyle

Parameters

Color As KMLColor, Scale As Double, IconName As String

Description

This method gets the color, the scale and the name of the icon in the Geometry of the Placemark.

Example

```
Dim PM As KMLPlacemark  
Dim Color As KMLColor  
Dim Scale As Double  
Dim Name As String  
  
Set PM = New KMLPlacemark  
PM.GetIconStyle Color, Scale, Name
```

Chapter 115 - KMLGeometry

Description

A KMLGeometry object is a generic object encompassing KMLPoint, KMLLineString, and KMLPolygon. When the type of a KMLGeometry object is known, the KMLGeometry can be converted to its specific object type. Methods GetPoint, GetLineString and GetPolygon do not create a new object but assign instead the reference to the object of the specific type.

Example

```
Dim G As KMLGeometry
Dim Poly As KMLPolygon
Dim LS As KMLLineString
Dim P As KMLPoint

If G.GeometryType = ptKMLPolygon Then
    G.GetPolygon Poly
ElseIf G.GeometryType = ptKMLPoint Then
    G.GetPoint P
ElseIf G.GeometryType = ptKMLLineString Then
    G.GetLineString LS
End If
```

Properties

GeometryType

Methods

GetPoint
GetLineString
GetPolygon

Properties

GeometryType

Returns

Long

Description

This property returns the type of the KMLGeometry object. The value is either: ptKMLPoint, ptKMLLineString or ptKMLPolygon.

Example

```
Dim G As KMLGeometry
Dim Type as Long

Type = G.GeometryType
```

Methods

GetPoint

Parameters

P As KMLPoint

Description

This method sets the reference to the KMLGeometry in a variable of type KMLPoint.

Example

```
Dim G As KMLGeometry
Dim P As KMLPoint

G.GetPoint P
```

GetLineString

Parameters

L As KMLLineString

Description

This method sets the reference to the KMLGeometry in a variable of type KMLLineString.

Example

```
Dim G As KMLGeometry
Dim L As KMLLineString

G.GetLineString L
```

GetPolygon

Parameters

P As KMLPolygon

Description

This method sets the reference to the KMLGeometry in a variable of type KMLPolygon.

Example

```
Dim G As KMLGeometry
Dim P As KMLPolygon

G.GetPolygon P
```


Chapter 116 - KMLPoint

Description

A KMLPoint object is a KMLGeometry.

Example

```
Dim G As KMLGeometry
Dim P As KMLPoint
Dim Pos As KMLGeoPoint

Pos.Latitude = 51.3147
Pos.Longitude = 4.8774
Pos.Altitude = 22.5
P.GeoPoint = Pos
P.GetGeometry G
```

Properties

GeoPoint

Methods

GetGeometry

Properties

GeoPoint

Returns

KMLGeoPoint

Description

This property sets or returns the coordinates of the point.

Example

```
Dim P As KMLPoint
Dim Pos As KMLGeoPoint

Pos = P.GeoPoint
```

Methods

GetGeometry

Parameters

G As KMLGeometry

Description

This method sets the reference to the KMLPoint in a variable of type KMLGeometry.

Example

```
Dim G As KMLGeometry
Dim P As KMLPoint

P.GetGeometry G
```

Chapter 117 - KMLLineString

Description

A KMLLineString object is a KMLGeometry. It contains a sequence of lines.

Example

```
Dim G As KMLGeometry
Dim L As KMLLineString
Dim Coords As KMLCoordinates

Set L = New KMLLineString
Set L.Coordinates = Coords
L.GetGeometry G
```

Properties

Coordinates
LinearRing
AltitudeMode
Extruded

Methods

GetGeometry

Properties

Coordinates

Returns

KMLCoordinates

Description

This property sets or returns the coordinates of the vertices of the KMLLineString.

Example

```
Dim L As KMLLineString
Dim Coords As KMLCoordinates

Set Coords = L.Coordinates
```

LinearRing

Returns

Boolean

Description

This property sets or returns the indication that the KMLLineString is a closed line string. As linear ring is used, in particular, as an outer bound of a polygon.

Example

```
Dim L As KMLLineString

L.LinearRing = True
```

AltitudeMode

Returns

Long

Description

This property sets or returns the altitude mode. The value can be ptClampToGround, ptRelativeToGround or ptAbsoluteAltitude. The meaning of those modes is defined in the KML Reference manual of Google.

Example

```
Dim L As KMLLineString

L.AltitudeMode = ptClampToGround
```

Extruded

Returns

Boolean

Description

This property sets or returns the indication Extruded of the `KMLLineString`. Specifies whether to connect the `LineString` to the ground. For more information see the [KML Reference manual of Google](#).

Example

```
Dim L As KMLLineString

L.Extruded = True
```

Methods

GetGeometry

Parameters

G As `KMLGeometry`

Description

This method sets the reference to the `KMLLineString` in a variable of type `KMLGeometry`.

Example

```
Dim G As KMLGeometry
Dim L As KMLLineString

L.GetGeometry G
```

Chapter 118 - KMLPolygon

Description

A KMLPolygon object is a KMLGeometry.

Example

```
Dim Outer As KMLCoordinates
Dim Inner As KMLCoordinates
Dim Poly As KMLPolygon

Set Poly = New KMLPolygon
' Initialise Outer & Inner
Set Poly.SetOuterBoundary = Outer
Set Poly.SetInnerBoundary = Inner
Poly.AltitudeMode = ptRelativeToGround
Poly.Extruded = True
```

Properties

AltitudeMode
Extruded
OuterBoundary
InnerBoundary

Methods

GetGeometry

Properties

AltitudeMode

Returns

Long

Description

This property sets or returns the altitude mode. The value can be ptClampToGround, ptRelativeToGround or ptAbsoluteAltitude. The meaning of those modes is defined in the KML Reference manual of Google.

Example

```
Dim L As KMLLineString
L.AltitudeMode = ptRelativeToGround
```

Extruded

Returns

Boolean

Description

This property sets or returns the indication Extruded of the KMLPolygon. Specifies whether to connect the polygon to the ground. For more information see the KML Reference manual of Google.

Example

```
Dim L As KMLLineString

L.Extruded = True
```

OuterBoundary

Returns

KMLCoordinates

Description

This property sets or returns the outer boundary of the KMLPolygon. The first and the last coordinate must be the same.

Example

```
Dim L As KMLPolygon
Dim Outer As KMLCoordinates

Set Poly.OuterBoundary = Outer
```

InnerBoundary

Returns

KMLCoordinates

Description

This property sets or returns the inner boundary of the KMLPolygon.

Example

```
Dim L As KMLPolygon
Dim Inner As KMLCoordinates

Set Poly.InnerBoundary = Inner
```

Methods

GetGeometry

Parameters

G As KMLGeometry

Description

This method sets the reference to the KMLPolygon in a variable of type KMLGeometry.

Example

```
Dim G As KMLGeometry
Dim Poly As KMLPolygon

Poly.GetGeometry G
```

Chapter 119 - KMLInnerBoundary

Description

A `KMLInnerBoundary` gives the boundary of one hole or of multiple holes of a `KMLPolygon`. If the number of cutouts (see `NrOfCutouts`) = 1, the `KMLInnerBoundary` consists of only one `KMLCoordinates` object. If there are multiple holes, `KMLInnerBoundary` consists of multiple `KMLCoordinates` objects.

Example

```
Dim Inner = KMLInnerBoundary

Set Inner = New KMLInnerBoundary
```

Properties

`NrOfCutouts`
`IsEmpty`
`Iterator`

Methods

`Add`
`Clear`

Properties

NrOfCutouts

Returns

`Long`

Description

This property returns the number of cutouts (= holes) in the inner boundary.

Example

```
Dim Inner As KMLInnerBoundary
Dim NrHoles As Long

NrHoles = Inner.NrOfCutouts
```

IsEmpty

Returns

`Boolean`

Description

This property returns true if the `InnerBoundary` is empty.

Example

```
Dim Inner As KMLInnerBoundary

If = Inner.IsEmpty Then
' ....
End If
```

Iterator

Returns

`KMLInnerBoundaryIterator`

Description

This property returns an iterator that allows to traverse a list of `KMLCoordinates`.

Example

```
Dim Inner As KMLInnerBoundary
Dim InnerIterator As KMLInnerBoundaryIterator

InnerIterator = Inner.Iterator
```

Methods

Add

Parameters

Coord As KMLCoordinates

Description

This method adds a KMLCoordinates object (a cut out) to the inner boundary. The object is added at the end of the list.

Example

```
Dim Inner As KMLInnerBoundary
Dim C As KMLCoordinates
Dim i As Integer
'...
For i = 1 To 5
    C = New KMLCoordinates
    '...
    Inner.Add C
Next i
```

Clear

Parameters

None

Description

This method makes the inner boundary empty. Result:

NrOfCutouts = 0

IsEmpty = True

Example

```
Dim Inner As KMLInnerBoundary

Inner.Clear
```

Chapter 120 - KMLInnerBoundaryIterator

Description

A KMLInnerBoundaryIterator allows to traverse the list of KMLCoordinates objects of a KMLInnerBoundary. Multiple iterators can be created and may be used simultaneously. When the methods Insert or Remove are used behaviour of other iterators may become incorrect. In that case the other iterators may become invalid. (See property Valid).

An iterator is destroyed when the parent KMLInnerBoundary is destroyed.

Example

```
Dim Inner = KMLInnerBoundary
Dim Iter As KMLInnerBoundaryIterator
Dim CutOut As KMLCoordinates
Dim First As Boolean

Set Inner = New KMLInnerBoundary
'....
Set Iter = Inner.Iterator

'....
If Iter.Valid = False Then
    Exit Sub
End If
First = True
While Not Iter.EndOfList
    If First Then
        Iter.GetFirst CutOut
        First = False
    Else
        Iter.GetNext CutOut
    End If
Wend
```

Properties

Valid
EndOfList

Methods

GetFirst
GetNext
Insert
Remove

Properties

Valid

Returns

Boolean

Description

This property returns true if the iterator is valid. An iterator may become invalid if the methods Insert or Remove are used by another iterator of the same KMLInnerBoundary.

Example

```
Dim Iterator As KMLInnerBoundaryIterator

If Iterator.Valid Then
'...
End If
```

EndOfList

Returns

Boolean

Description

This property returns true if the iterator is at the end of the list.

Example

```
Dim Iterator As KMLInnerBoundary

While Not Iterator.EndOfList
    '....
Wend
```

Methods

GetFirst

Parameters

CutOut As KMLCoordinates

Description

This method returns the first KMLCoordinates object (a cut out) from the inner boundary.

Example

```
Dim Iter As KMLInnerBoundaryIterator
Dim CutOut As KMLCoordinates
Dim First As Boolean

First = True
While Not Iter.EndOfList
    If First Then
        Iter.GetFirst CutOut
        First = False
    Else
        Iter.GetNext CutOut
    End If
Wend
```

GetNext

Parameters

CutOut As KMLCoordinates

Description

This method returns the next KMLCoordinates object (a cut out) from the inner boundary.

Example

See GetFirst

Insert

Parameters

CutOut As KMLCoordinates, Position As Long

Description

This method inserts a KMLCoordinates object (a cut out) at the given position.

Position has one of the following values: ptBegin, ptEnd, ptBefore, or ptAfter.

ptBegin and ptEnd insert respectively at the start or the end of the list.

ptBefore and ptAfter insert before or after the current position in the list. The current position is changed by the methods GetFirst and GetNext.

After Insert, the current position will point to the inserted element.

Example

```
Dim Iter As KMLInnerBoundaryIterator
Dim CutOut As KMLCoordinates

Iter.Insert CutOut, ptBegin
```

Remove

Parameters

CutOut As KMLCoordinates

Description

This method removes a cutout defined by its `KMLCoordinates` from the inner boundary.

Example

```
Dim Iter As KMLInnerBoundaryIterator
Dim CutOut As KMLCoordinates

Iter.Remove CutOut
```

Chapter 121 - KMLCoordinates

Description

KMLCoordinates is a list of coordinates of type KMLCoordinate.

Example

```
Dim Coords = KMLCoordinates

Set Coords = New KMLCoordinates
```

Properties

```
NrOfCoordinates
IsEmpty
Iterator
```

Methods

```
Add
Clear
```

Properties

NrOfCoordinates

Returns

Number As Long

Description

This property returns the number of coordinates in the list.

Example

```
Dim Coords As KMLCoordinates
Dim Nr As Long

Nr = Coords.NrOfCoordinates
```

IsEmpty

Returns

Boolean

Description

This property returns true if the list is empty.

Example

```
Dim Coords As KMLCoordinates

If = Coords.IsEmpty Then
' ....
End If
```

Iterator

Returns

KMLCoordinatesIterator

Description

This property returns an iterator that allows to traverse a list of KMLCoordinates.

Example

```
Dim Coords As KMLCoordinates
Dim Iter As KMLCoordinatesIterator

Iter = Coords.Iterator
```

Methods

Add

Parameters

Coord As KMLCoordinates

Description

This method adds a `KMLCoordinate` object to the list. The object is added at the end of the list.
The object is added by reference.

Example

```
Dim Coords As KMLCoordinates
Dim C As KMLCoordinate
Dim i As Integer
'...
For i = 1 To 5
    C = New KMLCoordinate
    '...
    Coords.Add C
Next i
```

Clear

Parameters

None

Description

This method clears the list. Result:
 `NrOfCoordinates = 0`
 `IsEmpty = True`

Example

```
Dim Coords As KMLCoordinates

Coords.Clear
```

Chapter 122 - KMLCoordinatesIterator

Description

A KMLCoordinatesIterator allows to traverse the list of KMLCoordinates. Multiple iterators can be created and may be used simultaneously. When the methods Insert or Remove are used behaviour of other iterators may become incorrect. In that case the other iterators may become invalid. (See property Valid).

An iterator is destroyed when the parent KMLCoordinates is destroyed.

Example

```
Dim Coords = KMLCoordinates
Dim Iter As KMLCoordinatesIterator
Dim C As KMLCoordinate
Dim First As Boolean

Set Coords = New KMLCoordinates
'....
Set Iter = Coords.Iterator

'....
If Iter.Valid = False Then
    Exit Sub
End If
First = True
While Not Iter.EndOfList
    If First Then
        Iter.GetFirst C
        First = False
    Else
        Iter.GetNext C
    End If
Wend
```

Properties

Valid
EndOfList

Methods

GetFirst
GetNext
Insert
Remove

Properties

Valid

Returns

Boolean

Description

This property returns true if the iterator is valid. An iterator may become invalid if the methods Insert or Remove are used by another iterator obtained from the same KMLCoordinates object.

Example

```
Dim Iterator As KMLCoordinatesIterator

If Iterator.Valid Then
'...
End If
```

EndOfList

Returns

Boolean

Description

This property returns true if the iterator is at the end of the list.

Example

```
Dim Iterator As KMLCoordinates

While Not Iterator.EndOfList
    '....
Wend
```

Methods

GetFirst

Parameters

C As KMLCoordinate

Description

This method returns the first coordinate.

Example

```
Dim Iter As KMLCoordinatesIterator
Dim C As KMLCoordinate
Dim First As Boolean

First = True
While Not Iter.EndOfList
    If First Then
        Iter.GetFirst C
        First = False
    Else
        Iter.GetNext C
    End If
Wend
```

GetNext

Parameters

C As KMLCoordinate

Description

This method returns the next coordinate.

Example

See GetFirst

Insert

Parameters

C As KMLCoordinate, Position As Long

Description

This method inserts a KMLCoordinate object at the given position.

Position has one of the following values: ptBegin, ptEnd, ptBefore, or ptAfter.

ptBegin and ptEnd insert respectively at the start or the end of the list.

ptBefore and ptAfter insert before or after the current position in the list. The current position is changed by the methods GetFirst and GetNext.

After Insert, the current position will point to the inserted element.

Example

```
Dim Iter As KMLCoordinatesIterator
Dim C As KMLCoordinate

Iter.Insert C, ptBegin
```

Remove

Parameters

C As KMLCoordinate

Description

This method removes the coordinate specified by the argument from the list.

Example

```
Dim C As KMLCoordinate
Dim Iter As KMLCoordinatesIterator
...
Iter.GetFirst C
If SomeCondition(C) Then
    Iter.Remove C
End If
```

Chapter 123 - KMLCoordinate

Description

A `KMLCoordinate` represents a `KMLPoint`.

Properties

`GeoPoint`

Properties

GeoPoint

Returns

`KMLGeoPoint`

Description

This property returns or sets the coordinates of a KML point

Example

```
Dim P As KMLGeoPoint
Dim C As KMLCoordinate
...
P.Longitude = 30.5
P.Latitude = 50.5
P.Altitude = 0.0
C.GeoPoint = P
'...
P = C.GeoPoint
```


Chapter 124 - JSON

Description

JSON represents a JSON data structure as described in the JSON standard (see json.org).

Example

```
Dim JS As JSON
Dim O As JSONObject

Set JS = New JSON      ' Creates an empty JSON data structure.
Set O = JS.JSONObject
    ' Fill O with data. See JSONObject example
    '

JS.AssignObject O
MsgBox JS.JSON          ' Converts to structured text and display.
```

Properties

Type
JSONObject
JSONArray
JSON

Methods

AssignObject
AssignArray
Clear
AssignJSON

Properties

Type

Returns

Integer

Description

This property returns either `ptJSONEmpty`, `ptJSONObject` or `ptJSONArray`.

Example

```
Dim JS As JSON
Dim O As JSONObject
Dim A As JSONArray

If JS.Type = ptJSONObject Then
    Set O = JS.JSONObject
Else
    Set A = JS.JSONArray
End If
```

JSONObject

Returns

JSONObject

Description

This property returns a JSONObject.

Example

See example Type.

JSONArray

Returns

JSONArray

Description

This property returns a JSONArray.

Example

See example Type.

JSON

Returns

S As String

Description

The JSON data is returned in JSON text format.

Example

```
Dim S As String
Dim JS As JSON

S = JS.JSON
```

Methods

AssignObject

Parameters

O As JSONObject

Description

The JSON structure becomes an object whose structure is defined by O.
If the JSON structure was not empty, an error is raised.

Example

```
Dim O As JSONObject
Dim JS As JSON

JS.AssignObject O
```

AssignArray

Parameters

A As JSONArray

Description

The JSON structure becomes an array whose structure is defined by A.
If the JSON structure was not empty, an error is raised.

Example

```
Dim A As JSONArray
Dim JS As JSON

JS.AssignArray A
```

Clear

Parameters

None

Description

Clears the JSON structure.

Example

```
Dim JS As JSON

JS.Clear S
```

AssignJSON

Parameters

S As String

Description

S should be a valid JSON structure. S is parsed and converted to an object or array.
The string may contain Tab's, space, line feeds, ... as defined in the JSON standard.
If S is not valid JSON, the error 3002, "JSON Syntax Error, ..." is raised.

Example

```
Dim S As String
Dim JS As JSON
```

```
Set JS = New JSON
S = "{""username"": ""admin"", ""password"": ""123""}"
JS.AssignJSON S
```

Search

Parameters

Name As String, Value As Variant

Description

Looks for the (first) child object or child array with the given name and returns its value.
If no object or array element with the given name is found, Value = Empty is returned.

Example

```
Dim JS As JSON
Dim V As Variant
Dim Address As JSONObject

JS.Search "address", V
If TypeName(V) = "JSONOBJECT" Then
    Set Address = V
    Address.Search "community", V
    ' etc
Else
    MsgBox " JSON Object expected."
End If
```

Chapter 125 - JSONObject

Description

A `JSONObject` represents a JSON object structure as described in the JSON standard (see json.org).

Example

```
Dim O As JSONObject
Dim A As JSONArray
Dim JS As JSON

Set JS = New JSON
Set O = New JSONObject
O.AddNameValue "Name1", -123456789
O.AddNameValue "Name2", True
O.AddNameValue "Name3", Null
O.AddNameValue "Name4", -12345.6789

Set A = New JSONArray
A.AddValue "A1_Value1"
A.AddValue "A1_Value2"
A.AddValue "A1_Value3"

O.AddNameValue "NameArray", A
JS.AssignObject O
MsgBox JS.JSON
```

Properties

`JSON`
`NrOfElements`

Methods

`AddNameValue`
`GetFirst`
`GetNext`

Properties

JSON

Returns

`S As String`

Description

The JSON data is returned in JSON text format.

Example

```
Dim S As String
Dim O As JSONObject

S = O.JSON
```

NrOfElements

Returns

`Nr As Long`

Description

Returns the number of name value pairs in the `JSONObject`. A nested `JSONObject` or `JSONArray` is considered one element, irrespective of the number of elements in the nested array or object.

Example

```
Sub JSONGetKeyValuePairs(ByVal O As JSONObject, _
                          ByRef NrOfKeys As Long, _
```

```

ByRef KeyValue() As tKeyValue)

Dim Name As String
Dim Value As Variant
Dim i As Long

NrOfKeys = O.NrOfElements
Redim KeyValue(1 To NrOfKeys)
i = 0
O.GetFirst Name, Value
Do While Not IsEmpty(Value)
    i = i + 1
    KeyValue(i).Key = Name
    KeyValue(i).Value = Value
    O.GetNext Name, Value
Loop
End Sub

```

Methods

AddNameValue

Parameters

Name As String, Value As Variant

Description

A name/value pair is added at the end of the object.

This method is used to add values of type *string*, *number*, *boolean*, *null*, *JSONObject* or *JSONArray*.

Example

```

Dim O As JSONObject
Dim ObjMaintenance As JSONObject

Set ObjMaintenance = New JSONObject
ObjMaintenance.AddNameValue "RepairedBy", "Fred"
Set O = New JSONObject
O.AddNameValue "username", "admin"
O.AddNameValue "Maintenance", ObjMaintenance
MsgBox O.JSON

```

GetFirst

Parameters

Name As String, Value As Variant

Description

Returns the first name/value pair of the object.

If there is no object, Name = "" and Value = Empty.

The returned value can be a *JSONObject* or a *JSONArray*. See example how to handle the nested objects/arrays.

Example

```

Dim O As JSONObject
Dim V As Variant
Dim Name As String
Dim S As String
Dim NestedO As JSONObject
Dim NestedA As JSONArray

S = ""
Set O = New JSONObject
O.AddNameValue "Name1", "Value1"
O.GetFirst Name, V
While Not (IsEmpty(V))
    S = S & Name & ", "
    If TypeName(V) = "JSONOBJECT" Then
        Set NestedO = V
    End If

```

```

        S = S & GetObject(NestedO)
    ElseIf TypeName(V) = "JSONARRAY" Then
        Set NestedA = V
        S = S & GetArray(NestedA)
    Else
        S = S & V
    End If
    S = S & vbCrLf
    O.GetNext Name, V
Loop
MsgBox S

```

GetNext

Parameters

Name As String, Value As Variant

Description

Returns the next name/value pair of the object.

If there are no more objects, Name = "" and Value = Empty.

Example

See GetFirst

Search

Parameters

Name As String, Value As Variant

Description

Looks for the (first) child object or child array with the given name and returns its value.

If no object or array element with the given name is found, Value = Empty is returned.

Example

```

Dim O As JSONObject
Dim V As Variant
Dim Address As JSONObject

O.Search "address", V
If TypeName(V) = "JSONOBJECT" Then
    Set Address = V
    Address.Search "community", V
    ' etc
Else
    MsgBox " JSON Object expected."
End If

```

Chapter 126 - JSONArray

Description

A JSONArray represents a JSON array structure as described in the JSON standard (see json.org).

Example

```
Dim JS As JSONArray

Set JS = New JSONArray 'Creates an empty object data structure.
```

Properties

JSON
NrOfElements

Methods

AddObject
AddArray

Properties

JSON

Returns

S As String

Description

The JSON data is returned in JSON text format.

Example

```
Dim S As String
Dim A As JSONArray

S = A.JSON
```

NrOfElements

Returns

Nr As Long

Description

Returns the number of name value pairs in the JSONArray. A nested JSONObject or JSONArray is considered one element, irrespective of the number of elements in the nested array or object.

Example

```
Dim A As JSONArray
Dim Nr As Long

Nr = A.NrOfElements
```

Methods

AddValue

Parameters

Value As Variant

Description

A value is added at the end of the object.

This method is used to add values of type *string*, *number*, *boolean*, *null*, JSONObject or JSONArray.

Example

```
Dim Obj As JSONObject
```

```

Dim Value As Variant

Obj.AddValue "V1"
Set Value = Obj.Maintenance
Obj.AddValue Value

```

GetFirst

Parameters

Value As Variant

Description

Returns the first value of the array.

If there is no object Value = Empty.

Example

```

Function GetArray(A As JSONArray) As String
    Dim V As Variant
    Dim S As String
    Dim NestedO As JSONObject
    Dim NestedA As JSONArray

    S = ""
    A.GetFirst V
    While Not (IsEmpty(V))
        If TypeName(V) = "JSONObject" Then
            Set NestedO = V
            S = S & GetObject(NestedO)
        ElseIf TypeName(V) = "JSONArray" Then
            Set NestedA = V
            S = S & GetArray(NestedA)
        Else
            S = S & V
        End If
        S = S & vbCrLf
        A.GetNext V
    Loop
    GetArray = S
End Function

```

GetNext

Parameters

Value As Variant

Description

Returns the next value (JSONArray) pair of the object.

If there are no more objects, Value = Empty.

Example

See GetFirst

Search

Parameters

Index As Long, Name As String, Value As Variant

Description

Looks in the array element defined by the parameter Index for the (first) child object or child array with the given name and returns its value. The element at the given index must be either a JSONObject or a JSONArray.

If no object or array element with the given name is found, Value = Empty is returned.

Example

```

Dim A As JSONArray
Dim V As Variant
Dim Address As JSONObject

```



```
A.Search 3, "address", V
If TypeName(V) = "JSONOBJECT" Then
    Set Address = V
    Address.Search "community", V
    ' etc
Else
    MsgBox " JSON Object expected."
End If
```

Chapter 127 - Units

Description

The `Units` object provides a series of functions that deal with conversion of Units (e.g. Feet to Meter, Gons to Radians) and the conversion from Double to String and from String to Double (e.g. N45°15'13"E to an azimuth in Radians). The values of the `Units` properties (e.g. `UnitForLength`, `UnitForAngle`, etc.) are parameters for the conversion functions.

The internal units used by VBA (eg. Sin, Cos) and by Pythagoras VBA properties and methods (eg.

`CreatePoint`) :

- unit of length : meter
- units for angles : radians
- angles are counter clockwise

The default values when executing : `Set U = New Units :`

```
UnitOfLength = ptM
AngleUnit = ptRADIAN
HorizontalAngleClockwise = False
VertAngleType = ptVALLevel
XYZRepresentation = ptRectangular
ScaleFormat = ptScaleMetric
RectangleCoordRepr = ptMathematicalXY
ItemSeparator = Asc(",")
DecimalSeparator = Asc(".")
DecimalPlacesForAngles = 4
DecimalPlacesForLengths = 3
DecimalPlacesForXYCoords = 3
DecimalPlacesForAreas = 2
DecimalPlacesForVolumes = 2
LongDateFormat = "yyyy-MMM-dd"
ShortDateFormat = "yyyy-MM-dd"
TimeFormat = "HH:mm:ss"
```

Example

```
Dim U As Units
Dim Angle As Double

Set U = New Units
U.UnitOfLength = ptUSfeet
U.AngleUnit = ptSURVEY
U.DecimalPlacesForAngle = 2
U.DecimalPlacesForLengths = 2
U.DecimalPlacesForXYCoord = 3
U.DecimalPlacesForAreas = 2
U.DecimalPlacesForVolumes = 2
U.StrConvertAngleToInternal "N45°15'13E", ptBearingAngle, Angle
```

Properties

```
UnitOfLength
AngleUnit
HorizontalAngleClockwise
VertAngleType
XYZRepresentation
RectangleCoordRepr
ScaleFormat
ChainageFormat
DecimalSeparator
ItemSeparator
DecimalPlacesForAngles
DecimalPlacesForLengths
DecimalPlacesForXYCoords
DecimalPlacesForAreas
DecimalPlacesForVolumes
LongDateFormat
ShortDateFormat
TimeFormat
ValidDateFormat
ValidTimeFormat
```

MaximumYear

Methods

SetTransformation
GetTransformation
ConvertCoordinatesToUser
ConvertCoordinatesToInternal
ConvertLengthToUser
ConvertLengthToInternal
ConvertAngleToUser
ConvertAngleToInternal
ConvertAreaToUser
ConvertAreaToInternal
ConvertVolumeToUser
ConvertVolumeToInternal
StrConvertCoordinatesToUser
StrConvertCoordinatesToInternal
StrConvertLengthToUser
StrConvertLengthToInternal
StrConvertChainageToUser
StrConvertAngleToUser
StrConvertAngleToInternal
StrConvertAreaToUser
StrConvertAreaToInternal
StrConvertVolumeToUser
StrConvertVolumeToInternal
FormatLongDate
FormatShortDate
FormatTime
StringToDateTime
StringToDate
StringToTime
StringToDateTime
GetPreferences
SetPreferences

Properties

UnitOfLength

Returns

Integer

Description

Returns or sets the unit of length used by the conversion methods. The unit of length must be one of the following values: ptMM, ptCM, ptDM, ptM, ptDAM, ptHM, ptKM, ptInch, ptMiles, ptFeet, ptUSfeet.

Example

```
Dim U As Units
```

```
U.UnitOfLength = ptUSfeet
```

AngleUnit

Returns

Integer

Description

Returns or sets the angle used by the conversion methods. The angle unit must be one of the following values: ptGRAD, ptDEGdec, ptDEGMinSec, ptMIL, ptSURVEY, ptRADIANT.

Example

```
Dim U As Units
```

```
U.AngleUnit = ptSURVEY
```

HorizontalAngleClockwise

Returns

Boolean

Description

Returns or sets the direction of the angle.

Example

```
Dim U As Units

U.HorizontalAngleClockwise = True
```

VertAngleType

Returns

Integer

Description

Returns or sets the type of vertical angle used by the conversion methods. The vertical angle type must have one of the following values : ptVALevel, ptVA90, ptVA270.

Example

```
Dim U As Units

U.VertAngleType = ptVA90
```

XYZRepresentation

Returns

Integer

Description

Returns or sets the meaning of X,Y,Z used by the conversion methods. The XYZ Representation must be one of the following values : ptRectangular, ptPolar, ptHVD, ptHVS.

Example

```
Dim U As Units

U.XYZRepresentation = ptHVS
```

RectangleCoordRepr

Returns

Integer

Description

Returns or sets the meaning of the rectangular coordinates. RectangleCoordRepr must have one of the following values : ptMathematicalXY, ptNorthEast, ptGermanYX, ptUserDefined.

Example

```
Dim U As Units

U.RectangleCoordRepr = ptNorthEast
```

ScaleFormat

Returns

Integer

Description

Returns or sets the scale format to either the metric or imperial standard. ScaleFormat must have one of the following values : ptScaleMetric, ptScaleImperial.

Example

```
Dim U As Units

Set U = New Units
U.GetPreferences
U.ScaleFormat = ptScaleImperial
U.SetPreferences
```

ChainageFormat

Returns

Integer

Description

Returns or sets the format used in converting a chainage to a string. ChainageFormat must have one of the following values: ptChainageFormatNormal, ptChainageFormatPlus.

Example

```
Dim U As Units

Set U = New Units
U.GetPreferences
U.ChainageFormat = ptChainageFormatPlus
U.SetPreferences
```

DecimalSeparator

Returns

Byte

Description

Returns or sets the decimal separator. The value is either "," (comma) or "." (point). When another value is attempted to be assigned an exception is raised. When this and ItemSeparator properties become both "," then value of ItemSeparator is changed to ";".

Example

```
Dim U As Units

U.DecimalSeparator = Asc(",")
```

ItemSeparator

Returns

Byte

Description

Returns or sets the item separator. An item separator is used when representing a list of items. The value is either "," (comma) or ";" (semicolon). When another value is attempted to be assigned an exception is raised. When this and DecimalSeparator properties become both ";" then value of DecimalSeparator is changed to ".".

Example

```
Dim U As Units

U.ItemSeparator = Asc(";")
```

DecimalPlacesForAngles

Returns

Integer

Description

Returns or sets the number of decimal places when an angle is converted to a string. The value must be in the range 0 to 5.

Example

```
Dim U As Units

U.DecimalPlacesForAngles = 5
```

DecimalPlacesForLengths

Returns

Integer

Description

Returns or sets the number of decimal places when a distance is converted to a string. The value must be in the range 0 to 5.

Example

```
Dim U As Units

U.DecimalPlacesForLengths = 2
```

DecimalPlacesForXYCoords

Returns

Integer

Description

Returns or sets the number of decimal places when coordinates (X,Y) are converted to strings. The value must be in the range 0 to 5.

Example

```
Dim U As Units

U.DecimalPlacesForXYCoords = 3
```

DecimalPlacesForAreas

Returns

Integer

Description

Returns or sets the number of decimal places when an area is converted to a string. The value must be in the range 0 to 5.

Example

```
Dim U As Units

U.DecimalPlacesForAreas = 1
```

DecimalPlacesForVolumes

Returns

Integer

Description

Returns or sets the number of decimal places when an volume is converted to a string. The value must be in the range 0 to 5.

Example

```
Dim U As Units

U.DecimalPlacesForVolumes = 0
```

LongDateFormat

Returns

String

Description

Returns or sets the string used for conversion of a date to a string.

The following ICU format designations may be used :

y	Year as a 1-digit number (0 - 9)
yy	Year as a 2-digit number(00 - 99)
yyyy	Year as a 4-digit number(100 - 9999)
M	Number of the month (1 - 12)
MM	Number of the month (01 - 12)
MMM	Abbreviated name of the month (Jan - Dec)
MMMM	Full name of the month (January - December)
d	Day of the month (1 - 31)
dd	Day of the month (01 - 31)
D	Day of the year (1 - 366)
DDD	Day of the year (001 - 366)
EEE	Abbreviated name of the day of the week (Sun - Sat)
EEEE	Full name of the day of the week (Sunday - Saturday)

Example

```
Dim U As Units

U.LongDateFormat = "'Today is : ' EEE d MMMM yyyy"
```

ShortDateFormat

Returns

String

Description

ShortDateFormat returns or sets the string used for conversion of a "short date" string to a date.

The following ICU format designations may be used (this is a subset of the format designations of

LongDateFormat :

yy	Year as a 2-digit number(00 - 99)
yyyy	Year as a 4-digit number(100 - 9999)
M	Number of the month (1 - 12)
MM	Number of the month (01 - 12)
d	Day of the month (1 - 31)
dd	Day of the month (01 - 31)

Notes:

- The 3 elements *day*, *month* and *year***must** be present, and only once.
- The designators must be separated by at least one character.

Example

```
Dim U As Units
```

```
U.ShortDateFormat = "yy-mm-dd"
```

TimeFormat

Returns

String

Description

Returns or sets the string used for conversion of the time to a string or vice versa. The following ICU format designations may be used :

h	Hour of the day AM/PM (1 - 12)
hh	Hour of the day AM/PM (01 - 12)
H	Hour of the day (0 - 23)
HH	Hour of the day (00 - 23)
m	Minute of the hour (0 - 59)
mm	Minute of the hour (00 - 59)
s	Second of the minute (0 - 59)
ss	Second of the minute (00 - 59)
S	Millisecond of second (0 - 999)
SSS	Millisecond of second (000 - 999)
a	AM/PM marker (AM/PM)

Notes:

- *Hours* and *minutes***must** be present, and only once.
- *Second* designator may be used only once.
- *Millisecond* designator may only be used together with second designator and be used only once.
- AM/PM designator may only be used with 12 hour designator (h).
- All designators except AM/PM must be separated at least by one character.

Example

```
Dim U As Units
```

```
U.TimeFormat = "hh-mm-ss"
```

AM

Returns

String

Description

Returns or sets the string used for AM (Ante Meridium) :

Example

```
Dim U As Units
```

```
U.AM = "am"
```

PM

Returns
String

Description
Returns or sets the string used for PM (Post Meridium) :

Example
Dim U As Units

U.PM = "pm"

ValidLongDateFormat

Returns
Boolean

Description
Returns true if the long date format string in the current class is a valid date string.

Example
Dim U As Units

If U.ValidLongDateFormat Then
End If

ValidShortDateFormat

Returns
Boolean

Description
Returns true if the short date format string in the current class is a valid date string.

Example
Dim U As Units

If U.ValidShortDateFormat Then
End If

ValidTimeFormat

Returns
Boolean

Description
Returns true if the time format string in the current class is a valid time string.

Example
Dim U As Units

If U.ValidTimeFormat Then

DateTimeFixedLength

Returns
Boolean

Description
If set True, the number of digits in the input string of the methods `StringToDateTime`, `StringToDate` and `StringToTime` must exactly match the number of digits specified in the format string. The format string is defined by the property `ShortDateFormat` or `TimeFormat`. This property is by default = False. It must only be set if the format string contains no separators (see example).

Example
Dim U As Units


```
U.ShortDateFormat = "yyyyMMdd"  
U.DateTimeFixedLength = True
```

MaximumYear

Returns

Integer

Description

Returns are sets the value used to convert a string to a date and only two digits of the year are specified. A year is chosen in the range : MaximumYear - 99 .. MaximumYear.

Example

```
Dim U As Units  
  
If U.ValidTimeFormat Then  
End If
```

Methods

SetTransformation

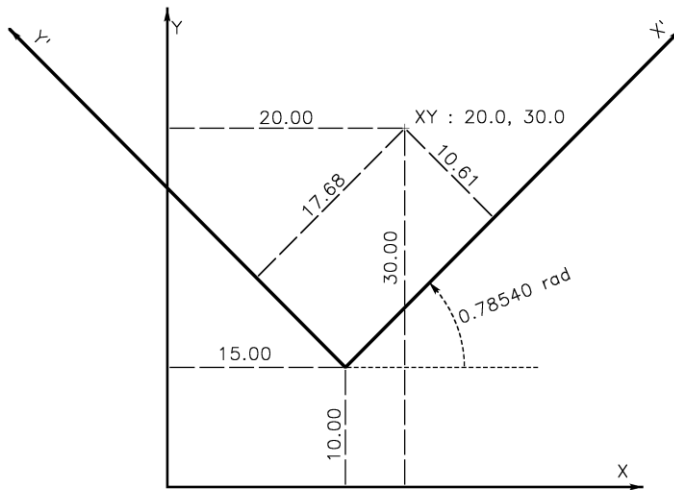
Parameters

TrPars As Transformation

Description

Sets the transformation parameters. The transformation parameters are used by the conversion routines to execute a transformation on the input coordinates. Note that Fi must be given in radians, and that D must be given in meter. The transformation parameters of a coordinate system can be obtained (see object `CoordinateSystem` property `TransformationParameters`). This allows to use the methods further described in this section to convert local coordinates from and to coordinates in the respective coordinate system.

The meaning of the transformation parameters is made clear by the following illustration:



Transformation XY -> X'Y'

```

Dim TrPars As Transformation
Dim U As Units
Dim P As XYZ
Dim X As Double, Y As Double, Z As Double

TrPars.Fi = 0.0
TrPars.D.X = 15.0
TrPars.D.Y = 10.0
TrPars.D.Z = 10.0
TrPars.Fi = 0.78540
TrPars.ScaleFactor = 0.5

Set U = New Units
U.SetTransformation TrPars

P.x = 20.0
P.y = 30.0
P.z = 0.0
U.ConvertCoordinatesToUser P, X, Y, Z

```

The above transformation returns :

```

X = 8.84
Y = 5.30
Z = -5.00

```

Example

```

Dim U As Units
Dim TrPars As Transformation

U.SetTransformation TrPars

```

GetTransformation

Parameters

```
TrPars As Transformation
```

Description

Returns the transformation parameters.

Example

```

Dim U As Units
Dim TrPars As Transformation

U.GetTransformation TrPars

```

ConvertCoordinatesToUser

Parameters

```
Pnt As XYZ, C1 As Double, C2 As Double, C3 As Double
```

Description

Converts coordinates (Pnt) from internal (meter) to new coordinates (C1, C2, C3) taking into account the UnitOfLength, XYZrepresentation (XYZ, HDZ, HVD or HVS), RectangleCoordRepr and the transformation.

Sequence of the operations to calculate C1, C2 and C3:

1. Transformation
 - 1.1. Translation
 - 1.2. Rotation
 - 1.3. Scaling
2. Conversion to XYZ representation
3. Conversion of units (unit of length, angle unit)

Example

```
' Conversion of rectangular to polar coordinates (HVS)
Dim U As Units
Dim P As XYZ
Dim H As Double, V As Double, S As Double

Set U = New Units
U.XYZRepresentation = ptHVS
U.AngleUnit = ptDegMinSec

P.x = 20.0
P.y = 30.0
P.z = 0.0
U.ConvertCoordinatesToUser P, H, V, S ' Result H=326.18, V=0.0, S=36.05
```

ConvertCoordinatesToInternal

Parameters

C1 As Double, C2 As Double, C3 As Double, Pnt As XYZ

Description

The inverse transformation of ConvertCoordinatesToUser. Converts coordinates (C1, C2, C3) whose meaning is defined by the settings in Units namely : UnitOfLength, XYZrepresentation (XYZ, HDZ, HVD or HVS), RectangleCoordRepr and the transformation, to a coordinate (Pnt).

Sequence of the operations to calculate Pnt starting from C1, C2 and C3:

1. Conversion of units (unit of length, angle unit)
2. Conversion to XYZ
3. Inverse Transformation
 - 3.1. Scaling
 - 3.2. Rotation
 - 3.3. Translation

It is obvious that the result of ConvertCoordinatesToInternal must be equal to the input of ConvertCoordinatesToUser, when the output of ConvertCoordinatesToUser is used as input for ConvertCoordinatesToInternal. See also next example.

Example

```
' Conversion of rectangular to polar coordinates (HVS)
Dim U As Units
Dim P As XYZ
Dim H As Double, V As Double, S As Double

Set U = New Units
U.XYZRepresentation = ptHVS
U.AngleUnit = ptDegMinSec

P.x = 20.0
P.y = 30.0
P.z = 0.0
U.ConvertCoordinatesToUser P, H, V, S ' Result H=326.18, V=0.0, S=36.05
U.ConvertCoordinatesToInternal H, V, S, P ' Result P = (20.0, 30.0, 0.0)
```

ConvertLengthToUser

Parameters

In As Double, Out As Double

Description

Converts a length, given by parameter In, from its internal value (meter) to a length in units, defined by UnitOfLenth.

Example

```
Dim U As Units
```

```

Dim DMeter As Double, DFeet As Double

Set U = New Units
U.UnitOfLength = ptFeet

U.ConvertLengthToUser 1.0, DFeet
U.ConvertLengthToInternal DFeet, DMeter

```

ConvertLengthToInternal

Parameters

In As Double, Out As Double

Description

Inverse conversion of: ConvertLengthToUser.

Example

See ConvertLenthToUser.

ConvertAngleToUser

Parameters

In As Double, AngleType As Integer, Out As Double

Description

Converts an angle, parameter In, from its internal value (radians) to an angle in units defined by AngleUnit. AngleType must have one of the following values ptHorAngle, ptVertAngle, ptArcAngle, ptBearingAngle.

If AngleType <> ptArcAngle, then an extra conversion will be done on the angle depending on the setting of HorizontalAngleClockwise or VertAngleType.

Example

```

Dim U As Units
Dim Bearing As Double
Dim A As Integer
Dim Out As Double

Set U = New Units
U.AngleUnit = ptSURVEY

U.ConvertAngleToUser 0.0, ptBearingAngle, Bearing
U.ConvertAngleToInternal Bearing, A, Out

```

ConvertAngleToInternal

Parameters

In As Double, AngleType As Integer, Out As Double

Description

Inverse conversion of: ConvertAngleToUser.

Example

See ConvertAngleToUser.

ConvertAreaToUser

Parameters

In As Double, Out As Double

Description

Converts an area, given by parameter In, from its internal value (square meter) to the area in the units defined by UnitOfLenth.

Example

```

Dim U As Units
Dim AreaInternal As Double, AreaSqFeet As Double

Set U = New Units
U.UnitOfLength = ptFeet

U.ConvertAreaToUser 2500.0, AreaSqFeet

```

```
U.ConvertAreaToInternal AreaSqFeet, AreaInternal
```

ConvertAreaToInternal

Parameters

In As Double, Out As Double

Description

Inverse conversion of: ConvertAreaToUser.

Example

See ConvertAreaToUser.

ConvertVolumeToUser

Parameters

In As Double, Out As Double

Description

Converts a volume, given by parameter In, from its internal value (cubic meter) to the volume in the units defined by UnitOfLength.

Example

```
Dim U As Units
Dim VolumeInternal As Double, VolumeCubicFeet As Double

Set U = New Units
U.UnitOfLength = ptFeet

U.ConvertVolumeToUser 25000.0, VolumeCubicFeet
U.ConvertVolumeToInternal VolumeCubicFeet, VolumeInternal
```

ConvertVolumeToInternal

Parameters

In As Double, Out As Double

Description

Inverse conversion of: ConvertVolumeToUser.

Example

See ConvertVolumeToUser.

StrConvertCoordinatesToUser

Parameters

Pnt As XYZ, C1 As String, C2 As String, C3 As String

Description

Converts coordinates (Pnt) from internal (meter) to new coordinates (C1, C2, C3) taking into account the UnitOfLength, XYZrepresentation (XYZ, HDZ, HVD or HVS), RectangleCoordRepr and the transformation. The result is identical to the one obtained from ConvertCoordinatesToUser with the exception that the results (C1, C2, C3) are strings. The number of decimal places in the result is defined by DecimalPlacesForLengths.

Sequence of the operations : see ConvertCoordinatesToUser

Example

```
' Conversion of rectangular to polar coordinates (HVS)
Dim U As Units
Dim P As XYZ
Dim HStr As String, VStr As String, SStr As String

Set U = New Units
U.XYZRepresentation = ptHVS
U.AngleUnit = ptDegMinSec

P.x = 20.0
P.y = 30.0
P.z = 0.0
U.StrConvertCoordinatesToUser P, HStr, VStr, SStr
```

StrConvertCoordinatesToInternal

Parameters

C1 As String, C2 As String, C3 As String, Pnt As XYZ

Description

The inverse transformation of ConvertCoordinatesToUser. Converts coordinates (C1, C2, C3) whose meaning is defined by the settings in Units namely : UnitOfLength, XYZrepresentation (XYZ, HDZ, HVD or HVS), RectangleCoordRepr and the transformation, to a coordinate (Pnt).

StrConvertCoordinatesToInternal will mainly be used to transform input data to a coordinate for insertion in the drawing.

Sequence of the operations : see ConvertCoordinatesToUser.

It is obvious that the result of StrConvertCoordinatesToInternal must be equal to the input of StrConvertCoordinatesToUser, when the output of StrConvertCoordinatesToUser is used as input for StrConvertCoordinatesToInternal. See also next example.

Example

```
' Conversion of polar (HVS) to rectangular coordinates
Dim U As Units
Dim P As XYZ
Dim HStr As String, VStr As String, SStr As String

Set U = New Units
U.XYZRepresentation = ptHVS
U.AngleUnit = ptDegMinSec

P.x = 20.0
P.y = 30.0
P.z = 0.0
U.StrConvertCoordinatesToUser P, HStr, VStr, SStr
U.StrConvertCoordinatesToInternal HStr, VStr, SStr, P ' Result = 20, 30, 0.0
```

StrConvertLengthToUser

Parameters

In As Double, Out As String

Description

Converts a length, given by parameter In, from its internal value (meter) to a length in units, defined by UnitOfLength. This method is identical to ConvertLengthToUser with this exception that the result is a string.

Example

```
Dim U As Units
Dim DMeter As Double, DFeet As String

Set U = New Units
U.UnitOfLength = ptFeet

U.StrConvertLengthToUser 1.0, DFeet
U.StrConvertLengthToInternal DFeet, DMeter
```

StrConvertLengthToInternal

Parameters

In As String, Out As Double

Description

Inverse conversion of : StrConvertLengthToUser.

Example

See StrConvertLengthToUser.

StrConvertChainageToUser

Parameters

In As Double, Out As String

Description

Converts a length, given by parameter *In*, from its internal value (meter) to a length in units, defined by *UnitOfLength* and using the *ChainageFormat*.

Example

```
Dim U As Units
Dim Chainage As Double
Dim S As String

Set U = New Units
U.UnitOfLength = ptMeter
U.ChainageFormat = ptChainageFormatPlus
U.StrConvertChainageToUser Chainage, S
```

StrConvertAngleToUser

Parameters

In As Double, *AngleType* As Integer, *Out* As String

Description

Converts an angle, parameter *In*, from its internal value (radians) to an angle in units defined by *AngleUnit*. *AngleType* must have one of the following values *ptHorAngle*, *ptVertAngle*, *ptArcAngle*, *ptBearingAngle*.

If *AngleType* < *ptArcAngle*, then an extra conversion will be done on the angle depending on the setting of *HorizontalAngleClockwise* or *VertAngleType*. This method is identical to *ConvertAngleToUser* with this exception that the result is a string.

Example

```
Dim U As Units
Dim A As Double, Bearing As String

Set U = New Units
U.AngleUnit = ptDegMinSec

U.StrConvertAngleToUser 0.0, ptBearingAngle, Bearing
U.StrConvertAngleToInternal Bearing, ptBearingAngle, A
```

StrConvertAngleToInternal

Parameters

In As String, *AngleType* As Integer, *Out* As Double

Description

Inverse conversion of: *ConvertAngleToUser*.

Example

See *StrConvertAngleToUser*.

StrConvertAreaToUser

Parameters

In As Double, *Out* As String.

Description

Converts an area, given by parameter *In*, from its internal value (square meter) to the area in the units defined by *UnitOfLength*. This method is identical to *ConvertAreaToUser* with this exception that the result is a string.

Example

```
Dim U As Units
Dim AreaInternal As Double, AreaSqFeet As String

Set U = New Units
U.UnitOfLength = ptFeet

U.StrConvertAreaToUser 2500.0, AreaSqFeet
U.StrConvertAreaToInternal AreaSqFeet, AreaInternal
```

StrConvertAreaToInternal

Parameters

In As String, Out As Double

Description

Inverse conversion of: StrConvertAreaToUser.

Example

See StrConvertAreaToUser.

StrConvertVolumeToUser

Parameters

In As Double, Out As String

Description

Converts a volume, given by parameter In, from its internal value (cubic meter) to the volume in the units defined by UnitOfLenth. This method is identical to ConvertVolumeToUser with this exception that the result is a string.

Example

```
Dim U As Units
Dim VolumeInternal As Double, VolumeCubicFeet As String

Set U = New Units
U.UnitOfLength = ptFeet

U.StrConvertVolumeToUser 25000.0, VolumeCubicFeet
U.StrConvertVolumeToInternal VolumeCubicFeet, VolumeInternal
```

StrConvertVolumeToInternal

Parameters

In As String, Out As Double

Description

Inverse conversion of: StrConvertVolumeToUser.

Example

See StrConvertVolumeToUser.

FormatLongDate

Parameters

D As Date

Returns

String

Description

Converts the date given by the parameter D to a string. The format corresponds with the setting of the property LongDateFormat. If the format is invalid the string "?????" will be returned.

Example

```
Dim U As Units

MsgBox U.FormatLongDate(Date)
```

FormatShortDate

Parameters

D As Date

Returns

String

Description

Converts the date given by the parameter D to a string. The format corresponds with the setting of the property ShortDateFormat. If the format is invalid the string "?????" will be returned.

Example

```
Dim U As Units
```



```
MsgBox U.FormatShortDate(Date)
```

FormatTime

Parameters

T As Date

Returns

String

Description

Converts the time given by the parameter T to a string. The format corresponds with the setting of the property TimeFormat. If the format is invalid the string "?????" will be returned.

Example

```
Dim U As Units

MsgBox U.FormatTime(Time)
```

StringToDateTime

Parameters

S As String

Returns

Date

Description

Converts the string containing date and time to a date. The setting of the property ShortDateFormat and ShortTimeFormat are used. If the format is an invalid date/time string, Error value is returned.

Example

```
Dim U As Units
Dim D As Date

D = U.StringToDateTime("25-09-1946 05:00")
If IsError(D) Then
    ' Invalid date/time string
End If
```

StringToDate

Parameters

S As String

Returns

Date

Description

Converts the string to a date. The setting of the property ShortDateFormat is used. If the format is an invalid date string, Error value is returned.

Example

```
Dim U As Units
Dim D As Date

D = U.StringToDate("25-09-1946")
```

StringToTime

Parameters

S As String

Returns

Date

Description

Converts the string to a date. The setting of the property TimeFormat is used. If the format is an invalid date string, Error is returned.

Example

```
Dim U As Units
```

```
Dim D As Date

D = U.StringToTime("10am")
```

GetPreferences

Parameters

Optional DecimalPlacesOrigin As Integer

Description

Sets the units to the preferences set in the *Defaults, Preferences* dialog. Decimals are taken over from defaults. Optional parameter `DecimalPlacesOrigin` must be `ptDPControlPanel`, `ptDPDrawing` (default) or `ptDPExport`.

Example

```
Dim U As Units

Set U = New Units
U.GetPreferences DecimalPlacesOrigin:=ptDPExport
```

SetPreferences

Parameters

None

Description

Sets the the preferences (see *Defaults, Preferences* dialog) to the values stored in the object Units. Regarding decimal places : only the number of decimal places for annotations can be changed with this method.

Example

```
U.SetPreferences ' Preferences of user are changed to values in object
```

Chapter 128 - Comm

Description

The Comm object allows communication with devices via the serial port. Multiple instances of Comm may exist simultaneously so that communication with 2 serial devices is possible at the same time.

Example

```
Dim C As COMM
Dim Nr As LONG
Dim Data() As BYTE
Dim Ok As Boolean

Set C = New COMM
C.Port = "COM1"
C.Baudrate = 9600
C.Parity = ptNoParity
C.NrOfBits = 8
C.StopBits = ptStopbit1
C.Handshaking = ptNoFlowControl
C.OpenLink ptCommReadaccess, Ok
If Ok THEN
    Nr = 0
    ReDim Data(1 to 200)
    C.Read 200, Data, Nr
    ' etc.
End If
```

Properties

Baudrate
Stopbits
Parity
Handshaking
NrOfBits
Port

Methods

OpenLink
CloseLink
Read
Write
ClearInputBuffer

Properties

Baudrate

Returns

Long

Description

Returns or sets the baudrate of the serial port. Baudrate must have one of the following values : 110, 220, 300, 1200, 4800, 9600, 19200, 38400, 57600, 115200, 128000, 256000 or 19200.

Example

```
Dim C As Comm

C.Baudrate = 9600
```

StopBits

Returns

Long

Description

Returns or sets the number of stop bits. StopBits must have one of the following values : ptStopBit1, ptStopbit1_5, ptStopbit2

Example

```
Dim C As Comm
```

```
C.StopBits = ptStopBit1
```

Parity

Returns

Long

Description

Returns or sets the parity. Parity must have one of the following values : ptNoParity, ptOddParity, ptEvenParity, ptMarkParity, ptSpaceParity.

Example

```
Dim C As Comm

C.Parity = ptNoParity
```

HandShaking

Returns

Long

Description

Returns or sets the handshaking protocol. HandShaking must have one of the following values : ptXonXoff, ptHardware, ptNoFlowControl.

Example

```
Dim C As Comm

C.HandShaking = ptHardware
```

NrOfBits

Returns

Long

Description

Returns or sets the number of bits per byte. The value must be in the range 4 to 8. Normally this value = 8.

Example

```
Dim C As Comm

C.NrOfBits = 8
```

Port

Returns

String

Description

Returns or sets the communication port that will be used for the serial communication. The name of the communication port starts with "COM" followed by the digit 1, 2, 3 or 4. Most computers have 1 or 2 communication ports named COM1 and COM2.

Example

```
Dim C As Comm

C.Port = "COM1"
```

Methods

OpenLink

Parameters

Access As Integer, Ok As Boolean

Description

OpenLink opens the communication port using the parameters Baudrate, Parity, NrOfBits, StopBits, HandShaking. The parameter Access must have one of the following values : ptCommReadAccess, ptCommWriteAccess or ptCommReadWriteAccess. If the communication link is opened, Ok returns True.

Example

```
Dim C As COMM
Dim Nr As LONG
Dim Data() As BYTE
Dim Ok As Boolean

Set C = New COMM
C.Port = "COM1"
C.Baudrate = 9600
C.Parity = ptNoParity
C.NrOfBits = 8
C.StopBits = ptStopbit1
C.Handshaking = ptNoFlowControl
C.OpenLink ptCommReadAccess, Ok
```

CloseLink

Parameters

None

Description

CloseLink closes the serial port communication.

Example

```
Dim C As COMM

Set C = New COMM
C.CloseLink
```

Read

Parameters

NrOfBytes As Long, Data() As Byte, NrOfBytesRead As Long

Description

The method Read, reads data which is in the input buffer of the serial port. The dynamic array Data must be large enough to read the number of bytes requested (NrOfBytes). The real number of bytes read is returned in the parameter NrOfBytesRead.

Example

```
Dim C As COMM
Dim Nr As LONG
Dim Data() As BYTE
Dim Ok As Boolean

Set C = New COMM
C.Port = "COM1"
C.Baudrate = 9600
C.Parity = ptNoParity
C.NrOfBits = 8
C.StopBits = ptStopbit1
C.Handshaking = ptNoFlowControl
C.OpenLink ptCommReadAccess, Ok
If Ok THEN
    Nr = 0
    ReDim Data(1 to 200)
    C.Read 200, Data, Nr
    ' etc.
End If
```

Write

Parameters

NrOfBytes As Long, Data() As Byte, NrOfBytesWritten As Long

Description

The method Write, writes data to the serial port. The dynamic array Data must be large enough to read the number of bytes requested (NrOfBytes). The real number of bytes read is returned in the parameter NrOfBytesRead.

Example

```
Dim C As Comm
Dim Data() As Byte
Dim j As Long
Dim Nr As Long
Dim T As String
Dim s As String

ReDim Data(1 to Len(s))
for j = 1 to Len(s)
    t = Mid(s, j, 1)
    Data(j) = Cbyte(Asc(t))
next j
C.Write Len(s), Data, Nr
```

ClearInputBuffer

Parameters

None

Description

ClearInputBuffer, empties the input buffer of the serial port.

Example

```
Dim C As Comm

C.ClearInputBuffer
```

Chapter 129 - Timer

Description

The `Timer` object allows invocation of macros when a certain time has elapsed.

Since the call back macro can be called while the user is drawing objects (eg. busy drawing a line), care must be taken which actions are done in the call back macro. The call back macro should not modify documents on the screen, unless the Private Tool is selected. (See `Application` method `PrivateToolSelected`). In the private tool mode, the macro has control over interactive functions on the screen.

Destroying a timer object (`Set T = Nothing`), will stop the timer. Therefore, **timer objects should be declared globally**. If declared in a procedure, the timer object is destroyed by exit of the procedure.

Example

```
Dim T As Timer                ' Must be declared globally

Set T = New Timer
T.Interval = 2000              ' Interval = 2.0 sec
T.CallBackMacro = "CallMe"    ' Name of the macro to be called every 2 seconds
T.Start
```

Properties

`ElapsedTime`
`Interval`
`CallBackMacro`

Methods

`Start`
`Stop`

Properties

ElapsedTime

Returns

`Long`

Description

Returns the time in milliseconds since the timer is started.

Example

```
Dim T As Timer                ' Must be declared globally

If T.ElapsedTime > 100000 Then
    MsgBox "wake up"
End If
```

Interval

Returns

`Long`

Description

Returns or sets the timer interval. When the Interval time is elapsed, the callback macro will be executed. The timer interval is expressed in milliseconds.

Example

```
Dim T As Timer                ' Must be declared globally

Set T = New Timer
T.Interval = 10000
T.CallBackMacro = "CallMe"    ' Macro "CallMe" to be called every 10 seconds
```

CallBackMacro

Returns

`String`

Description

Returns or sets the name of the macro which is executed when the timer interval is elapsed.

Example

```
Dim T As Timer                                ' Must be declared globally

Set T = New Timer
T.Interval = 10000
T.CallBackMacro = "CallMe"                    ' Macro "CallMe" to be called every 10 seconds
T.Start
```

Methods

Start

Parameters

None

Description

Starts the timer. Only when the timer has been started the call back macro will be executed and the elapsed time will start to run.

Example

```
Dim T As Timer                                ' Must be declared globally

Set T = New Timer
T.Interval = 10000
T.CallBackMacro = "CallMe"                    ' Macro "CallMe" to be called every 10 seconds
T.Start                                        ' This starts the timer
```

Stop

Parameters

None

Description

Stops the timer. The timer object still exists, but the timer stops to run and the call back macro will not be called again. Destroying the object (Set T = Nothing) would also stop the timer.

Example

```
Dim T As Timer                                ' Must be declared globally

T.Stop                                        ' Stop calling CallMe.
```


Chapter 130 - Progress

Description

The `Progress` object allows to display a progress dialog box while a macro is executed. The progress dialog contains a title, a message (that may change during the progress), and a progress bar. Optionally the dialog contains a Cancel button. The progress dialog is removed with the `Stop` method. If the `Stop` method is not called, the progress will remain on the screen until the `Progress` object is destroyed.

Example

```
Dim P As Progress           ' May be declared globally
Dim I As Long, NrofObjects As Long

Set P = New Progress
P.CancelButton = True       ' Progress dialog with Cancel button
P.Title = "Import progress" ' Name of the macro to be called every 2
                             ' seconds
P.Message = "Part 1/2 : reading data from disk"
P.Start                    ' Dialog box will appear
Do Until (I >= NrofObjects) OR P.Cancelled
    P.Percentage = CDBL(I * 100 / NrofObjects)
    ' ....
Loop
P.Stop
```

Properties

CancelButton
Message
Title
Percentage
Cancelled

Methods

Start
Stop

Properties

CancelButton

Returns

Boolean

Description

This property should be called before calling the method `Start`. When set = True, the dialog box gets a Cancel button. By default the progress dialog has no cancel button. When the property is called when the dialog box is displayed, this has no effect.

Example

```
Dim P As Progress

Set P = New Progress
P.CancelButton = True ' Progress dialog with Cancel button
```

Message

Returns

String

Description

Returns or sets the message displayed in the progress dialog box.

Example

```
Dim P As Progress

Set P = New Progress
P.Message = "Part 2/2 : building Pythagoras drawing"
```

Title

Returns

String

Description

Returns or sets the title of the progress dialog box.

Example

```
Dim P As Progress

Set P = New Progress
P.Title = "Importing data"
```

Percentage

Returns

Double

Description

Returns or sets the progress. Percentage should be a value between 0.0 and 100.0. The value indicates the ratio of the progress bar.

Example

```
Dim P As Progress

Set P = New Progress
P.Start
P.Percentage = 15.0
```

Cancelled

Returns

Boolean

Description

Returns True if the Cancel button has been pressed.

A macro will NOT be interrupted when the Cancel button is pressed. A macro should call the Cancelled property frequently to allow a fast response when the user presses the Cancel button.

Example

```
Dim P As Progress

Set P = New Progress
P.Title = "Importing data"
P.Message = "Still importing..."
P.Start
If P.Cancelled Then
    P.Stop
End If
```

Methods

Start

Parameters

None

Description

Displays the progress dialog box on the screen.

Example

```
Dim P As Progress

Set P = New Progress
P.Title = "Importing data"
P.Message = "Still importing..."
P.Start
```

Stop

Parameters

None

Description

Removes the progress dialog box from the screen.

Example

```
Dim P As Progress
```

```
P.Stop
```

Chapter 131 - Fonts

Description

The `Fonts` object allows to get the fonts of the system.

Example

```
Dim Fnts As Fonts
Dim Fnt As Font ' See Object type

Dim i As Long

Set Fnts = New Fonts
For i = 1 To Fnts.NrOfFonts
    If i = 1 Then
        Fnts.GetFirst Fnt
    Else
        Fnts.GetNext Fnt
    End If
    ' do something with the Font
Next i
```

Properties

`NrOfFonts`

Methods

`GetFirst`
`GetNext`

Properties

NrOfFonts

Returns

`Long`

Description

This property returns the number of fonts in the system.

Example

```
Dim Fnts As Fonts

Set Fnts = New Fonts
MsgBox CStr(Fnts.NrOfFonts)
```

Methods

GetFirst

Parameters

`None`

Description

Returns the first font of the system.

Example

See introduction

GetNext

Parameters

`None`

Description

Returns the next font of the system.

Example

See introduction

Chapter 132 - Folder

Description

The **Folder** object allows to get information about a folder and to obtain the folders and files stored in the folder.

Note: the **Folder** object is only supported on Windows platform.

Example

```
Dim F As Folder, NF As Folder, FF As File
Dim i As Long

Set F = New Folder
F.Assign "C:\TestPrograms\VBA\"
For i = 1 To F.NrOfFolders
    If i = 1 Then
        Set NF = F.FirstFolder
    Else
        Set NF = F.NextFolder
    End If
    ' do something with the Folder
Next i
For i = 1 To F.NrOfFiles
    If i = 1 Then
        Set FF = F.FirstFile
    Else
        Set FF = F.NextFile
    End If
    ' do something with the File
Next i
```

Properties

NrOfFolders
NrOfFiles
Path

Methods

Assign
FirstFolder
NextFolder
FirstFile
NextFile

Properties

NrOfFolders

Returns

Long

Description

This property returns the number of folders in the folder.

Example

```
Dim F As Folder

Set F = New Folder
F.Assign "C:\TestPrograms\VBA\"
MsgBox CStr(F.NrOfFolders)
```

NrOfFiles

Returns

Long

Description

This property returns the number of files, not including folders, in the folder.

Example

```
Set F = New Folder
F.Assign "C:\TestPrograms\VBA\"
```

```
MsgBox CStr(F.NrOfFiles)
```

Path

Returns

String

Description

This property returns the path name of the folder.

Example

```
Set F = New Folder
F.Assign "C:\TestPrograms\VBA\"
MsgBox F.Path           'Returns "C:\TestPrograms\VBA\"
```

Methods

Assign

Parameters

String

Description

Assigns a path name to a folder.

Example

```
Dim F As Folder

Set F = New Folder
F.Assign "C:\TestPrograms\VBA\"
```

FirstFolder

Parameters

Folder

Description

Returns the first folder in the given folder. If the folder contains no other folders, the method returns Nothing.

Example

See introduction

NextFolder

Parameters

None

Description

Returns the next folder.

Example

See introduction

FirstFile

Parameters

File

Description

Returns the first file in the given folder. If the folder contains no other files, the method returns Nothing.

Example

See introduction

NextFile

Parameters

File

Description

Returns the next file in the given folder. If the folder contains no other files, the method returns Nothing.

Example

See introduction

Chapter 133 - File

Description

The File object allows to get information about a file.

Note: the File object is only supported on Windows platform.

Example

```
Dim F As Folder, FF As File
Dim i As Long

Set F = New Folder
F.Assign "C:\TestPrograms\VBA\"
For i = 1 To F.NrOfFiles
    If i = 1 Then
        Set FF = F.FirstFile
    Else
        Set FF = F.NextFile
    End If
    MsgBox File.Name
Next i
```

Properties

Drive
Path
Name
Extention
FullName
FileType
Application

Methods

Assign

Properties

Drive

Returns

String

Description

This property returns the drive name.

Example

```
Dim F As File

Set F = New File
F.Assign "C:\TestPrograms\VBA\Test.bas"
MsgBox F.Drive ' Returns "C:"
```

Path

Returns

String

Description

This property returns the full path name.

Example

```
Dim F As File

Set F = New File
F.Assign "C:\TestPrograms\VBA\Test.bas"
MsgBox F.Path ' Returns "C:\TestPrograms\VBA"
```

Name

Returns

String

Description

This property returns the name of the file.

Example

```
Dim F As File

Set F = New File
F.Assign "C:\TestPrograms\VBA\Test.bas"
MsgBox F.Name ' Displays "Test.bas"
```

Extention

Returns

String

Description

This property returns the extention of the file.

Example

```
Dim F As File

Set F = New File
F.Assign "C:\TestPrograms\VBA\Test.bas"
MsgBox F.Extention ' Displays "bas"
```

FullName

Returns

String

Description

This property returns the full name.

Example

```
Dim F As File

Set F = New File
F.Assign "C:\TestPrograms\VBA\Test.bas"
MsgBox F.Drive ' Returns "C:\TestPrograms\VBA\Test.bas"
```

FileType

Returns

String

Description

This property returns the type of file.

Example

```
Dim F As File

Set F = New File
F.Assign "C:\TestPrograms\VBA\Test.bas"
MsgBox F.Type ' Returns TBD
```

Application

Returns

String

Description

This property returns the name of the application (full path name) to open the file.

Example

```
Dim F As File

Set F = New File
F.Assign "C:\TestPrograms\VBA\Test.bas"
MsgBox F.Application ' Returns "C:\Program Files\Notepad"
```

Methods

Assign

Parameters
String

Description
Assigns a path name to a file.

Example

```
Dim F As File

Set F = New File
F.Assign "C:\TestPrograms\VBA\Test.txt"
```

EncodeToBase64

Parameters
MaxLength

Description
Encodes the content of a file to Base64. For more information on Base 64 see <https://en.wikipedia.org/wiki/Base64>
The parameter MaxLength gives the maximum encoded line length (see article). The value will mostly be equal to 76.
If no maximum encoded length, the value = 0.

Example

```
Dim F As File
Dim S As String

Set F = New File
F.Assign "C:\Desert.jpg"
S = F.EncodeToBase64(74)
```

Chapter 134 - ODBCDataSource

Description

The ODBCDataSource object allows to access ODBC data sources in a way that is much easier than by using directly the ODBC.DLL from VBA. The most frequently used functions are supported by Pythagoras VBA. These functions allow to retrieve data, edit data and add data to tables of an existing database. General knowledge of ODBC and a good knowledge of SQL is assumed when reading this chapter.

Example

```
Const SQL_SUCCESS = 0
Dim DB As ODBCDataSource
Dim Err As Integer
Dim DataBaseName As String
Dim StrError As String
Dim Res As Integer
Dim DataStr As String

Set DB = New ODBCDataSource

DataBaseName = "Parcels"
DB.Connect DataBaseName, "admin", "", Err, StrError
If Err <> SQL_SUCCESS Then
    MsgBox "error connect"
    SET DB = Nothing
    Exit Sub
End If
DB.Select "SELECT Name FROM Parcel WHERE ParcelNr = 'AB53DZ'", Res
If Res <> SQL_SUCCESS Then
    MsgBox "Error in Select Statement"
Else
    If DB.FetchNext() Then          ' Fetch the first row
        DB.GetData 1, DataStr, Res ' Get the first column
        If Res = SQL_Success Then
            ' Use data
        End If
    End If
End If
End If
```

Methods

Connect
Select
FetchNext
GetData
Cancel
FreeStmt

Methods

Connect

Parameters

DataSource As String, UserId As String, PassWord As String, Error As Integer, ErrorStr As String

Description

Connects with the database. The parameter DataSource is the name of the **ODBC User data source**. That name is given with the standard MS Windows tool "*ODBC Data Source Administrator*". Once this call is successfully executed, the VBA program can access the database. When a connection with a database is made, Pythagoras stores internally a number of handles that are used to access the database. When the ODBCDataSource object is released, either implicitly by the compiler when the Object variable is released or explicitly with the statement `Set DB = Nothing`, all ODBC statements to disconnect and to free memory are called by Pythagoras VBA environment. This method calls the function *SQLConnect* of the ODBC32.DLL. The parameters Error and ErrorStr are return parameters. These values are as defined by the ODBC interface standard.

Example

```
Const SQL_SUCCESS = 0
Dim DB As ODBCDataSource
Dim Err As Integer, StrError As String
Dim DataBaseName As String

Set DB = New ODBCDataSource

DataBaseName = "Parcels"
DB.Connect DataBaseName, "admin", "", Err, StrError
If Err <> SQL_SUCCESS Then
    MsgBox "error connect"
    SET DB = Nothing
End If
```

Select

Parameters

SQLStr As String, Error As Integer

Description

The Select method sends the *Select* statement with the parameter SQLStr to the database. The meaning of the return parameter Error is defined by the ODBC interface standard.

This method calls the function *SQLExecDirect* of the ODBC32.DLL.

The parameter Error is a return parameter.

Example

```
Const SQL_SUCCESS = 0
Dim DB As ODBCDataSource
Dim Err As Integer
Dim SQLStr As String
Dim res As Integer

Set DB = New ODBCDataSource
SQLStr = "SELECT Name FROM Parcel WHERE ParcelNr = 'AB53DZ'"
DB.Select SQLStr, res
If res <> SQL_SUCCESS Then
    MsgBox "Parcel not found"
Else
    ' Fetch & Get data
End If
```

FetchNext

Return parameter

Ok As Boolean

Description

This method must be called before calling GetData. The method will retrieve the next row and increment the cursor.

This method calls the function *SQLFetch* of the ODBC32.DLL.

Example

```
Dim ODBC As ODBCDataSource

If ODBC.FetchNext() Then
    ' Get data
End If
```

GetData

Parameters

ColumnNr As Integer, Data As String, SQLReturn As Integer

Description

Reads the column specified by ColumnNr from the current row and returns the content of the column in the parameter Data. The meaning of the return parameter SQLReturn is defined by the ODBC interface standard.

This method calls the function *SQLGetData* of the ODBC32.DLL.

Example

```
Const SQL_SUCCESS = 0
Dim DB As ODBCDataSource
Dim DataStr As String
Dim Res As Integer

DB.GetData 1, DataStr, Res
If Res = SQL_Success Then
    ' Use data
End If
```

Cancel

Return parameter

SQLReturn As Integer

Description

This method calls the function *SQLCancel* of the ODBC32.DLL.

The meaning of the return parameter SQLReturn is defined by the ODBC interface standard.

Example

```
Dim DB As ODBCDataSource
Dim Res As Integer

DB.Cancel Res
```

FreeStmt

Parameters

Option As Integer, SQLReturn As Integer

Description

This method calls the function *SQLFreeStmt* of the ODBC32.DLL.

The value of parameter Option is one of the following values : SQL_CLOSE, SQL_DROP, SQL_UNBIND or SQL_RESET_PARAMS.

Example

```
Const SQL_CLOSE = 0
Dim Res As Integer
Dim DB As ODBCDataSource

DB.FreeStmt SQL_CLOSE, Res
```

Chapter 135 - HTTP

Description

HTTP allows to communicate with web services using HTTP methods GET, POST, PUT, DELETE, ...

Example

```
Dim ObjHTTP As HTTP
Dim Response As String

Set ObjHTTP = New HTTP
ObjHTTP.DoOpen "GET", "https://api.mycumulus.com/v1/time", False
ObjHTTP.Send ""
If ObjHTTP.Status = 200 Then
    Response = ObjHTTP.ResponseText
    MsgBox Response
End If
```

Properties

Status
StatusText
ResponseText

Methods

DoOpen
SetRequestHeader
Send
ResponseBody
GetAllResponseHeaders
SetTimeouts

Properties

Status

Returns

status As Long

Description

The [status](#) returned by the HTTP server.

Example

```
Dim ObjHTTP As HTTP
Dim Response As String

Set ObjHTTP = New HTTP
ObjHTTP.DoOpen "GET", "https://api.mycumulus.com/v1/time", False
ObjHTTP.Send ""
If ObjHTTP.Status = 200 Then
    Response = ObjHTTP.ResponseText
    MsgBox Response
End If
```

StatusText

Returns

S As String

Description

Returns the status in a meaningful text (e.g. OK, Not found, ...)

Example

```
Dim ObjHTTP As HTTP
Dim Response As String

Set ObjHTTP = New HTTP
ObjHTTP.DoOpen "GET", "https://api.mycumulus.com/v1/time", False
ObjHTTP.Send ""
MsgBox ObjHTTP.StatusText
```

ResponseText

Returns

Response As String

Description

The response returned by the HTTP server.

Example

```
Dim ObjHTTP As HTTP
Dim Response As String

Set ObjHTTP = New HTTP
ObjHTTP.DoOpen "GET", "https://api.mycumulus.com/v1/time", False
ObjHTTP.Send ""
If ObjHTTP.Status = 200 Then
    MsgBox ObjHTTP.ResponseText
End If
```

Methods

DoOpen

Parameters

Method As String, URL As String, Asynchronous As Boolean

Description

DoOpen will set up the command with its parameters ready to be send to the server. Common methods supported by servers are GET, POST, PUT, DELETE. E.g. [REST API](#).

Example

```
Dim ObjHTTP As HTTP
Dim Response As String

Set ObjHTTP = New HTTP
ObjHTTP.DoOpen "POST", "https://api.mycumulus.com/v1/files/search", False
```

SetRequestHeader

Parameters

Name As String, Value As String

Description

Sets an HTTP request header. For each header the command must be given

Example

```
Dim ObjHTTP As HTTP
Dim Response As String

Set ObjHTTP = New HTTP
ObjHTTP.DoOpen "POST", "https://api.mycumulus.com/v1/crs/dz", False
ObjHTTP.SetRequestHeader "Content-Type", "application/json"
ObjHTTP.SetRequestHeader "X-Auth-Token", "23ab8f9xf8sx"
ObjHTTP.Send " {"crs" : "RDNAP",
               "coordinates" : {
                   "latitude" : 51.7010500000000,
                   "longitude" : 5.35511000000000
               }
            }"
```

Send

Parameters

RequestBody As String

Description

Sends the HTTP method, request headers and the request body to the server.

Example

See SetRequestHeader

ResponseBody

Parameters

Data() As Byte

Description

Returns the response in byte format. The dimension of the given array will be assigned by the method so that the response just fits in it. See example below.

Example

```
Dim ObjHTTP As HTTP
Dim Data() As Byte

Set ObjHTTP = New HTTP
ObjHTTP.DoOpen "GET", "https://api.mycumulus.com/v1/files/foto.jpg", False
...
ObjHTTP.Send "GET"
If ObjHTTP.Status = 200 Then
    ObjHTTP.ResponseBody Data. ' The response body will be moved to Data
    For i = LBound(Data) To UBound(Data)
        ...
    Next i
End If
```

GetAllResponseHeaders

Parameters

Headers As String

Description

Returns all response headers in string format

Example

```
Dim ObjHTTP As HTTP
Dim Response As String

Set ObjHTTP = New HTTP
ObjHTTP.DoOpen "GET", "https://api.mycumulus.com/v1/time", False
ObjHTTP.Send ""
MsgBox ObjHTTP.GetAllResponseHeaders
```

SetTimeOuts

Parameters

ResolveTimeout As Long,
ConnectTimeout As Long,
SendTimeout As Long,
ReceiveTimeout As Long

Description

[This link](#) gives detailed information for each parameter.

Example

```
Dim ObjHTTP As HTTP
Dim Response As String

Set ObjHTTP = New HTTP
objHTTP.SetTimeOuts 50000, 100000, 100000, 100000
```


Chapter 136 - Dialogs

Description

The Dialogs object is created by the Application method `GetDialogs`.

Example

```
Dim A As Application
Dim MyDialogs As Dialogs

Set A = New Application
A.GetDialogs MyDialogs
MyDialogs.RequestTextFileView "C:\Test.txt"
```

Properties

None

Methods

```
RequestFileOpen
RequestFilesOpen
RequestFolderOpen
RequestFileSave
RequestTextFileView
GetTrueColor
```

Methods

RequestFileOpen

Parameters

FileName As String, Title As String, FileType As String, Extension As String, ReadOnly As Boolean, Ok As Boolean

Description

Will invoke the standard Windows "Open" dialog.

When a file is selected, the complete pathname of the selected file will be returned in the parameter `FileName`.

`FileName` : if the string `<> ""`, the path will be used to show the files in the corresponding directory.

The `FileName` may contain wild characters ("`*`" and "`?`") so that only matching names are displayed in the open dialog box list.

`Title` : the description in the title bar of the dialog.

`FileType` : A description of the type of files. All files of this type can be shown.

Examples :

- 1) "Pythagoras File|*.PYT|Text File|*.TXT" : will show 2 types (*.PYT and *.TXT)
- 2) "All Files|*.*|Text File|*.TXT" : will show 2 types (*.PYT and *.TXT)
- 3) "Text File" : will show 1 type defined by Extension.

`Extension` : gives the default extension.

Examples matching the above:

- 1) "*.PYT" : Pythagoras documents are the default
- 2) "*.*" : all documents will be the default
- 3) ".TXT" : TXT files only.

`ReadOnly` : if = FALSE, only files which can be opened for read/write will be shown.

`Ok` : returns TRUE if a file is selected.

Example

```
Dim A As Application
Dim MyDialogs As Dialogs
Dim Ok As Boolean
Dim FileName As String

Set A = New Application
A.GetDialogs MyDialogs
FileName = "C:\Dir1\Test"
MyDialogs.RequestFileOpen FileName, "Select Pythagoras file", _
    "Pythagoras File|*.PYT|Pythagoras Prototype|*.PPD", "*.PYT", False, Ok
MsgBox "Requested File name : " & FileName
```

RequestFilesOpen

Parameters

Path As String, FileNames() As String, Title As String, FileType As String, Extension As String, ReadOnly As Boolean, Ok As Boolean

Description

Will invoke the standard Windows "Open" dialog.

When one or more files are selected, names of the selected files (without path) will be returned in the parameter FileNames.

Path: the path will be used to show the files in the corresponding directory.

Title : the description in the title bar of the dialog.

FileType : A description of the type of files. All files of this type can be shown.

Examples :

1) "Pythagoras File|*.PYT|Text File|*.TXT" : will show 2 types (*.PYT and *.TXT)

2) "All Files|*.*|Text File|*.TXT" : will show 2 types (*. * and *.TXT)

3) "Text File" : will show 1 type defined by Extension.

Extension : gives the default extension.

Examples matching the above:

1) "*.PYT" : Pythagoras documents are the default

2) "*.*" : all documents will be the default

3) "TXT" : TXT files only.

ReadOnly : if = FALSE, only files which can be opened for read/write will be shown.

Ok : returns TRUE if at least a one file is selected.

Example

```
Dim A As Application
Dim MyDialogs As Dialogs
Dim Ok As Boolean
Dim PathName As String
Dim FileNames() As String
Dim i As Long

Set A = New Application
A.GetDialogs MyDialogs
FileName = "."
MyDialogs.RequestFilesOpen PathName, FileNames, "Select Pythagoras file(s)", _
    "Pythagoras File|*.PYT|Pythagoras Prototype|*.PPD", "*.PYT", False, Ok
MsgBox "Path : " & FileName
For i = 0 To UBound(FileNames)
    MsgBox FileNames(i)
Next i
```

RequestFolderOpen

Parameters

Folder As String, Ok As Booleans, Optional ForSave As Boolean, Optional Title As String

Description

Will invoke the standard Windows "Open" dialog.

In this dialog box, only paths can be selected.

If Folder is an empty string the dialog box will display the current directory, otherwise the corresponding directory will be shown.

ForSave (Default = True) changes default values for button/title.

Title (Default = "") allows custom title. "" will use Windows defaults.

Example

```
Dim A As Application
Dim MyDialogs As Dialogs
Dim Ok As Boolean
Dim Path As String

Set A = New Application
Path = "C:\MyDir\"
RequestFolderOpen will display files of C:\MyDir\
A.GetDialogs MyDialogs
MyDialogs.RequestFolderOpen Path, Ok
If Ok Then
```

```

        MsgBox "Requested folder : " & Path
    End If

```

RequestFileSave

Parameters

FileName As String, Title As String, FileType As String, Extension As String, Ok As Boolean

Description

Will invoke the standard Windows "Save" dialog. On entry, FileName may contain a string. This name will be displayed initially in the dialog box.

The complete pathname of the name entered file will be returned in the parameter FileName.

FileName : if the string <> "", it must be the path of the directory initially shown.

Title : the description in the title bar of the dialog.

FileType : A description of the type of file. This description comes in the dialog box.

Extension : only files with this extension will be shown

Ok : returns TRUE if a file is selected.

Example

```

Dim A As Application
Dim MyDialogs As Dialogs
Dim Ok As Boolean
Dim FileName As String

Set A = New Application
A.GetDialogs MyDialogs
MyDialogs.RequestFileSave FileName, "My Title", "My description", "PYT", Ok
If Ok Then
    MsgBox "Requested File name : " & FileName
End If

```

RequestTextView

Parameters

FileName As String

Description

Opens the text file with the given name with the Pythagoras text editor.

The editor window is opened and remains inaccessible until the macro completes.

Typically the method should be called before exiting the macro.

Example

```

Dim A As Application
Dim MyDialogs As Dialogs

Set A = New Application
A.GetDialogs MyDialogs
MyDialogs.RequestTextView "C:\Test.txt"

```

GetTrueColor

Parameters

Color As RGB, Ok As Boolean

Description

Displays a dialog box allowing the user to select a color. If the Ok button is pressed, the selected color is returned.

Example

```

Dim A As Application
Dim Doc As Document
Dim AllObjs As AllObjects
Dim drawing As Overlay
Dim Color As RGB
Dim Dlgs As Dialogs
Dim Ok As BOOLEAN

Set A = New Application
Set Dlgs = New Dialogs

```

```
A.GetActiveDocument Doc
If NOT Doc is nothing then
    Doc.GetOverlay ptDrawing, drawing
    drawing.GetAllObjects AllObjs
    Color.Red = 128
    Color.Green = 200
    Color.Blue = 25
    Dlgs.GetTrueColor Color, Ok
    if Ok then
        AllObjs.TrueColor = Color
    end if
end if
```

Chapter 137 - Console

Description

A Console corresponds with the MS-Windows Console window.

Only one instance of Console is created. If multiple are made, they will all point to the same console Window.

When a Console object is set to Nothing, the Console window will be closed.

Example

```
Dim C As Console

Set C = New Console
C.WriteString "This is output to the console Window"
\ ..
\ ..

Set C = Nothing \ This will close the console window
```

Properties

Visible

Methods

WriteString
WriteLine
ReadLine

Properties

Visible

Returns

Boolean

Description

This property sets or returns the visible state of the Console Window.

Note: Visible does not mean that the console Windows is on top.

Example

```
Dim C As Console

C.Visible = False
```

Methods

ReadLine

Parameters

None

Returns

String

Description

This method returns the value entered in the console window when the Enter-key is pressed. This method is reserved for debugging a VBA application.

Example

```
Sub PM_Message
    Dim i As Integer
    Dim Count As Integer
    Dim S As String

    Set C = New Console

    C.WriteString "Give number of values : "
    Count = CInt(C.ReadLine)
    C.WriteString vbCrLf
    For i = 1000 TO 1000 + Count - 1
        C.WriteLine "i = " & CStr(i)
    Next i
End Sub
```

```

Next i

C.WriteString "Press Enter key "
S = C.ReadLine
C.WriteLine "Thank you for pressing Enter key"

C.WriteString "Press Enter key "
S = C.ReadLine
C.WriteLine "Thank you for pressing Enter key"
Set C = Nothing ' This will close the console
End Sub

```

WriteString

Parameters

String

Description

This method writes the string to the console window. This method is reserved for debugging a VBA application.

Example

See example `ReadLine`

WriteLine

Parameters

String

Description

This method writes the string and CRLF to the console window. This method is reserved for debugging a VBA application.

Example

See example `ReadLine`

Chapter 138 - Form Modules

Description

Form Modules contain both the Form Event Sub's and the Form Resource. A form is a special kind of Class Module. The Form module is instantiated in a Code Modules (Set F = New NameForm). The form control methods and properties can be accessed from within Form and/or Code modules. A form may be set as modal or non-modal at run-time. The default is a modal window. The specific event `Modality` may be used to set a form non-modal. A modal form has to be closed before the user can continue to operate with other windows of Pythagoras or continue to run the VBA code. A non-modal form can coexist with other form windows the user can operate on and VBA code can continue the execution after this window was created and opened. Several non-modal forms can be created for the same form. Display status of a modal form window is controlled by methods `Show` and `Hide`. A non-modal form has to be created with the method `OpenForm` and destroyed with the method `CloseForm`. Methods `Show` and `Hide` control the visibility of a non-modal form.

Example

```
Dim Form As MyForm           ' MyForm is the name of FormModule
Dim FirstName As String

Set Form = New MyForm        ' An instance of MyForm will be created and the
                              ' form event MyForm_Initialize will be executed.
Form.Show                    ' Will show the form on the screen.
                              ' This call does not return until the form is
                              ' hidden or unloaded.
                              '
                              ' Read the required data from the form controls
                              ' e.g.
FirstName = Form.FirstName.Value
                              ' Returns the value of the form Textbox
                              ' FirstName.

Set Form = Nothing
```

Example of a non-modal form

Suppose we wish to have a small window displaying the current time of day and being placed always on the top of any other window on the screen. We wish, also, to be able to do any work in Pythagoras, except of loading and compiling macros, while the time is displayed. We can do this by creating a non-modal, floating form. First, we create the system macro-library, say, `CurrentTime`. This library will contain two modules: the code module `TimeModule` and the form module `TimeForm`. By loading empty `TimeModule` in Editor and using the command **File/New Form** we get the empty form module. Rename it as `TimeForm` using the command **File/Modules Manager....** The window layout for our form can be created with the command **Run/Form**. In the from window we create the test label named as `Text` that will display the current time. We can adjust the size of the form window by moving any corner of the window. Close this window and go to the code of `TimeModule`. We place the following code in this module:

```
Dim Form As TimeForm
Dim T As Timer

Sub TimerCbK
    Form.Text.Caption = CStr(Time)
End Sub

Sub PM_CurrentTime
    Set Form = New TimeForm
    Form.OpenForm
    Form.Floating = True
    TimerCbK
    Form.Show
    Set T = New Timer
    T.Interval = 1000
    T.CallBackMacro = "TimerCbK"
    T.Start
End Sub

Sub OnTermination
    Set T = Nothing
End Sub
```

```

    If Not Form Is Nothing Then
        Form.CloseForm
        Set Form = Nothing
    End If
End Sub

```

In this code, we declare the form and the timer objects. The timer call-back subroutine `TimerCbk` just assigns the textual representation of the current time to the form label `Text`. Subroutine `CurrentTime` creates the form object, opens the form window, sets this window floating, i.e. top-most, and shows the window on the screen. The subroutine `TimerCbk` is called to show the initial value of the current time when the window is displayed the first time. Next, the new timer object is created, the interval of 1000 milliseconds is set to call the timer call-back subroutine, the name of the timer call-back subroutine is set, and, finally, the timer is started. It should be noted, that after the subroutine `PM_CurrentTime` finishes, form and timer objects declared globally exist until any macro-library will be compiled or loaded, or these objects will be destroyed explicitly, e.g. by closing the form and assigning the value of `Nothing` to corresponding variables. The subroutine `OnTermination` demonstrates the possibility to specify user actions being executed when this library is deactivated, that is, any macro-library is compiled or loaded. The form module `TimeForm` contains the following code:

```

Sub TimeForm_Modality(m As Boolean)
    m = FALSE
End Sub

Sub TimeForm_QueryClose(Cancel As Boolean, Mode As Integer)
    Cancel = False
    OnTermination
End Sub

```

This code processes two events `Modality` and `QueryClose`. The subroutine `TimeForm_Modality` is called when the form object is created to specify the modality of the form. We need a non-modal form, so the value of `FALSE` must be specified. The subroutine `TimeForm_QueryClose` is called when the form is closed (explicitly, with the call of the form method `CloseForm`, or implicitly, by clicking on the close button of the form window or when the active library is deactivated). In this subroutine we need to stop the timer and close the form if it is not closed yet, so we just call `OnTermination`.

Events

```

Initialize
Modality
Terminate
QueryClose
Click
DblClick
Change

```

Form Control Methods and Properties

See corresponding section

Form Events

Initialize

Parameters

None

Description

This event is called when a new instance of the Form has been created but not yet shown. The form itself is not displayed yet on the screen.

Captions and values of controls can be set in the `Initialize` procedure if the form is modal.

For non-modal forms all initialisations of the form controls must be done in the `Initialize` event.

See method *OpenForm*.

Example

```

Sub FormName_Initialize
    ' FormName is the name of the Form module.
    FirstName.Value = "Fred Flintstone"
    ' Initializes the form Textbox FirstName
    ' Works only if Form is modal.
End Sub

```


Modality

Parameters

Modal As Boolean

Description

This event is called immediately before a new instance of the Form is created. A form window is not created yet at this moment. For modal forms the corresponding procedure should not be included into the user code. For non-modal forms the procedure must be included and must set a parameter value to `False`.

A non-modal form should not call any functions of Pythagoras that change a document. The result could be unpredictable.

Example

```
Sub FormName_Modality(Modal As Boolean)
    ' FormName is the name of the Form module.
    Modal = False
End Sub
```

Terminate

Parameters

None

Description

This event is called when an instance of the Form is about to be deleted. This means that all references to the form have been set to Nothing.

Example

```
Sub FormName_Terminate
    ' FormName is the name of the Form module
    ' Execute termination code if any.
End Sub
```

QueryClose

Parameters

Cancel As Boolean, Mode As Integer

Description

This event is called when a request has been made to close the form. Setting `Cancel = True` will stop the form from being closed.

When `Mode = vbFormControlMenu`, the user has clicked the *Close box* of the form.

When `Mode = vbFormCode`, the close request has come from VBA using the `Unload` statement. The `Unload` statement is currently not supported.

Example

```
Sub FormName_QueryClose(Cancel As Boolean, Mode As Integer)
    'Do Something
End Sub
```

Click

Parameters

None

Description

This event is called when the mouse is clicked in one of the following controls: `CommandButton`, `ToggleButton`, `CheckBox` or `OptionButton`.

Example

```
Sub ControlName_Click
    ' ControlName is the name of the control
    'Do Something
End Sub
```

DropButtonClick

Parameters

None

Description

This event is called when the mouse is clicked in the drop button of a ComboBox or DropDownList control. The event is also called when the drop-down list disappears.

Example

```
Sub ControlName_DropButtonClick
    'Do Something
    'e.g. make a new list depending on information in other controls
End Sub
```

DbClick

Parameters

None

Description

This event is called when the mouse is double clicked in one of the following controls: OptionButton, ListBox.

Example

```
Sub ControlName_DblClick          'ControlName is the name of the control

End Sub
```

Change

Parameters

None

Description

This event is called when the value of the control has changed. This event is applicable to the following controls :

ToggleButton : when the push state of the button changes

CheckBox : when the checked state of the button changes

OptionButton : when the checked state of the button changes

TextBox : when the text in the control changes

ListBox : when the selected item changes

DropDownList: when the selected item changes

SpinButton : when the selected item changes

ComboBox : when the text in the control changes or selected item changes

MultiPage : when the selected tab changes

Example

```
Sub ControlName_Change 'ControlName is the name of the control
End Sub
```

Form Control Methods and Properties

Description

The following controls are supported in a Form :

TextBox, Label, CommandButton, ToggleButton, CheckBox, OptionButton, SpinButton, ListBox, DropDownList, ComboBox, Frame. MultiPage.

Each of these controls has its specific methods and properties. These methods and properties can be accessed from the Code Module or from the Form Module.

Warning: the name of a control may not contain an "_" (underscore) character.

Example

```
' Example 1 : code executed in a Code Module
' Form has 2 TextBox controls with the names FirstName and LastName.
```

```
Dim Name As String
Dim Form As MyForm          ' MyForm is the name of FormModule

Set Form = New MyForm       ' An instance of MyForm will be created.
Form.Show                  ' Will show the form on the screen.
```

```

` The following code is executed when the form is closed.
Name = Form.FirstName.Value
Name = Name & " " & Form.LastName.Value
Set Form = nothing

` Example 2 : code executed in a Form Module

Sub OkButton_Click          ` Executed when OkButton is clicked
    If (Len(Me.FirstName.Value) <> 0) And (Len(Me.LastName.Value) <> 0) Then
        OkClicked = True
        Hide
    Else
        MsgBox "Enter name"
    End If
End Sub

```

Form Object

Caption

Returns
String

Description
Sets or returns the text displayed in the form title.

Example

```

Sub FormName_Initialize          ` FormName is the name of the Form module.
    Me.Caption = "Name and Address"
                                ` Sets the title of the form.
End Sub

```

Floating

Value
Boolean

Description

For modal forms: this property should not be called.

For non-modal forms: when set True, the form will come on top of other windows. By default Floating = False.

Note that this property affects the form window, so it must be called after the method OpenForm is called and before the method CloseForm is called.

Example

```

Form.Floating = True            ` Form will be floating (= top window)

```

Show

Description

For modal forms: it shows the form on the screen and does not return until the form becomes hidden.

For non-modal forms: it shows the form and continues execution. The form should be opened beforehand.

Example
Form.Show

Hide

Description

For modal forms: It should be called inside a method or property procedure of the form. The form is hidden immediately. VBA program will continue with the statements following the Show method.

For non-modal forms: It should be called outside a method or property procedure of the form. The form is hidden immediately. VBA program will continue with the statements following the Hide method.

Example

Example
Form.Hide

OpenForm

Description

For modal forms: It acts as Show method.

For non-modal forms: It creates a form window and starts a window message loop in a separate thread, but the form is not shown on the screen. **The OpenForm method must be executed before changing any other Form Object. Initialisations of form controls must be done inside Initialize event procedure.**

Example

```
Form.OpenForm
```

CloseForm

Description

For modal forms: It acts as Hide method.

For non-modal forms: It closes and destroys the form window.

Example

```
Form.CloseForm
```

GetPosition

Parameters

X As Long, Y As Long

Description

Returns the position of a form on the screen in arguments X and Y.

Example

```
Form1.GetPosition X, Y
```

SetPosition

Parameters

X As Long, Y As Long

Description

Sets the position of a form on the screen as specified by arguments X and Y.

Example

```
Form1.SetPosition 0, 0           ' Upper-left corner of the screen
```

Width

Returns

Long

Description

Returns the form width in pixels.

Example

```
Dim W As Long

W = Form1.Width
```

Height

Returns

Long

Description

Returns the form height in pixels.

Example

```
Dim H As Long

H = Form1.Height
```

ScreenWidth

Returns

Long

Description

Returns the width of the screen in pixels the form is displayed on.

Example

```
Dim SW As Long

SW = Form1.ScreenWidth
```

ScreenHeight

Returns

Long

Description

Returns the height of the screen in pixels the form is displayed on.

Example

```
Dim SH As Long

SH = Form1.ScreenHeight
```

Common properties

SetFocus

Returns

nothing

Description

This method gives the input focus to this control.

Example

```
FirstName.SetFocus
```

Visible

Returns

Boolean

Description

Sets the control visible or invisible. When reading returns if the control is visible.

Example

```
OkButton.Visible = False
```

Enabled

Returns

Boolean

Description

Sets the control enabled or disabled. When reading returns if the control is enabled.

Example

```
OkButton.Enabled = False
```

ControlTipText

Returns

String

Description

Sets or returns the tool tip text.

Example

```
FirstName.ControlTipText = "Enter your first name"
```

TextEdit Control

Value (= Text)

Returns
String

Description
Sets or returns the text displayed in the TextBox control.

Example
`Form.FirstName.Value = "Andrey" ' Sets the text in the TextBox FirstName.
' or (when statement is in the Form Module)
FirstName.Value = "Andrey"
' =
FirstName.Text = "Andrey"`

MaxLength

Returns
Long

Description
Sets or returns the maximum number of characters that can be entered in the TextBox control.

Example
`FirstName.MaxLength = 16`

Locked

Returns
Boolean

Description
Sets the control enabled or disabled. When reading returns if the control is enabled.

Example
`OkButton.Locked = True`

SetFocus, Visible, Enabled, ControlTipText

Description
See common properties.

Label Control

Caption

Returns
String

Description
Sets or returns the text displayed in the Label control.

Example
`Form.LabelFirstName.Caption = "First Name"`

Visible, Enabled, ControlTipText

Description
See common properties.

CommandButton Control

Value

Returns
Boolean

Description

Reading this property always returns False. Setting this property to TRUE triggers the "Clicked" event for the control.

Example

```
Form.ButtonFind.Value = True ' This will trigger the Clicked event.
```

Caption

Returns

String

Description

Sets or returns the text displayed in the CommandButton control.

Example

```
Form.ButtonFind.Caption = "Search"
```

Visible, Enabled, SetFocus, ControlTipText

Description

See common properties.

ToggleButton Control

Value

Returns

Boolean

Description

Sets or returns the status of the ToggleButton control. True means the control is in the pressed down state.

Example

```
If Form.ToggleButton1.Value Then  
End If
```

Caption

Returns

String

Description

Sets or returns the text displayed in the ToggleButton control.

Example

```
Form.ToggleButton1.Caption = "Home address"  
Form.ToggleButton2.Caption = "Business address"
```

Visible, Enabled, SetFocus, ControlTipText

Description

See common properties.

CheckBox Control

Value

Returns

Boolean

Description

Sets or returns the checkedstate of the CheckBox control. TRUE means the control is checked.

Example

```
If Form.CheckBox1.Value Then  
End If
```

Caption

Returns
String

Description
Sets or returns the text displayed in the CheckBox control.

Example
`Form.CheckBox1.Caption = "I wish to receive the monthly newsletter"`

Visible, Enabled, SetFocus

Description
See common properties.

OptionButton Control

Value

Returns
Boolean

Description
Returns the status of the OptionButton control. True means the control is in the checked state. When setting the value to True, all other buttons in the group are cleared automatically. You can't set an option button unchecked because it is unknown which other button in the group must be checked.

Example
`If Form.OptionButton1.Value Then
End If`

Caption

Returns
String

Description
Sets or returns the text displayed in the OptionButton control.

Example
`Form.OptionButton1.Caption = "Home address"
Form.OptionButton2.Caption = "Business address"`

Visible, Enabled, SetFocus, ControlTipText

Description
See common properties.

SpinButton Control

Value

Returns
Long

Description
Sets or returns the numeric value of the control.

Example
`If Form.Spinbutton1.Value = 10 Then
End If`

Locked

Returns
Boolean

Description
Enables or disables user editing of the control . When reading returns if the control is user editable.

Example

```
Spinbutton1.Locked = False
```

Visible, Enabled, SetFocus, ControlTipText

Description

See common properties.

Frame Control

Caption

Returns

String

Description

Sets or returns the text displayed in the Frame control.

Example

```
Form.Frame.Caption = "Selection"
```

ControlTipText

Description

See common properties.

ListBox Control

ColumnTitle (synonym ListHeader)

Parameters

Column As Long

Returns

String

Description

Sets or returns the title of the column specified by the parameter Column. Column index starts from 0.

Example

```
Names.ColumnTitle(0) = "Names"
```

Text

Returns

String

Description

Sets or returns the text of the currently selected item in the list.

Example

```
SelectedName = Form.Names.Text
```

ListCount

Returns

Long

Description

Returns the number of items in the control.

Example

```
Dim NrOfItems As Long
```

```
NrOfItems = Names.ListCount
```

ListIndex

Returns

Long

Description

Reading returns the index of the currently selected item. Setting a value changes the selected item in the control. The first item in the control has an index value of zero.

Example

```
Names.SelectedIndex = 0      \ Selects the first item of the ListBox
```

List

Parameters

Row As Long, Column As Long

Returns

String

Description

Sets or returns the text of the item specified by the parameters Row and Column. Column is ignored if not applicable for the specific instance of the control.

Example

```
With Names
    Item1 = .AddItem("1")
    Item2 = .AddItem("2")
    Item3 = .AddItem("3")
    Item4 = .AddItem("4")
    .List(0, 1) = "Andrey"
    .List(1, 1) = "Alex"
    .List(2, 1) = "Michael"
    .List(3, 1) = "Corban"
End With
```

ColumnWidths

Returns

String

Description

Sets the widths of the columns in the control. A setting of -1 or blank, results in a calculated optimal width. A value greater than 0 explicitly specifies the width of the column in points. To separate column entries in the string use a semicolon character (;) as a list separator. (Eg. "4.5;7;10.0"). Passing an empty string sets all columns to their calculated optimal width. For the optimal width to be set you need to use this property after you have added or changed the text of the control items.

Example

```
Names.ColumnWidths := "30;70"
\ or
Names.ColumnWidths := "30;-1"
```

AddItem

Parameters

ItemText As String

Returns

Long

Description

This method adds an item to the control. The parameter is the string value of the item to add. For controls with multiple columns this sets the text of the first column (column 0). The returned value is the items position in the list of items in the control.

Example

```
Item1 = Names.AddItem("1")
```

RemoveItem

Parameters

Item As Long

Returns

Boolean

Description

This method removes an item from the control. The returned value indicates if the removal was successful.

Example

```
If Names.RemoveItem(3) Then  
End If
```

Clear

Returns

None

Description

This method removes all items from the control.

Example

```
Names.Clear
```

Selected

Parameters

Row As Long

Returns

Boolean

Description

Sets or returns the selected state of the item specified by the parameter Row. This property is useful in a multi-select list box. It can be used to determine the selected rows and to select or deselect rows in a list. Index of first Row starts counting from zero.

Example

```
If ListBox1.Selected(i) = True Then  
    ListBox1.Selected(i) = False  
End If
```

SetFocus, Visible, Enabled, ControlTipText

Description

See common properties.

DropDownList Control

Text

Returns

String

Description

Sets or returns the text of the currently selected item in the list.

Example

```
SelectedCountry = Form.Countries.Text
```

ListCount

Returns

Long

Description

Returns the number of items in the control.

Example

```
Dim NrOfItems As Long  
  
NrOfItems = Countries.ListCount
```

ListIndex

Returns

Long

Description

Reading returns the index of the currently selected item. Setting a value changes the selected item in the control. The first item in the control has an index value of zero.

Example

```
Countries.SelectedIndex = 0 ` Selects the first item of the ListBox
```

List

Parameters

Row As Long, Column As Long

Returns

String

Description

Sets or returns the text of the item specified by the parameters Row and Column. Column is ignored if not applicable for the specific instance of the control.

Example

AddItem

Parameters

ItemText As String

Returns

Long

Description

This method adds an item to the control. The parameter is the string value of the item to add. The returned value is the items position in the list of items in the control.

Example

```
Item = Countries.AddItem("Belgium")
```

RemoveItem

Parameters

Item As Long

Returns

Boolean

Description

This method removes an item from the control. The returned value indicates if the removal was successful.

Example

```
If Countrie.RemoveItem(3) Then  
End If
```

Clear

Returns

None

Description

This method removes all items from the control.

Example

```
Names.Clear
```

SetFocus, Visible, Enabled, ControlTipText

Description

See common properties.

Combobox Control

Value

Returns
String

Description
Sets or returns the text value of the currently selected row of the control.

Example
`SelectedCountry = Form.Countries.Value`

Text

Returns
String

Description
This property returns or sets the text of the control. In addition, when setting this property the text is matched against the items in the combo box list and if a match is found the selected item is changed to the matched item.

Example
`SelectedCountry = Form.Countries.Text`

MaxLength

Returns
Long

Description
Sets or returns the maximum number of characters that can be entered into the control. A value of less than or equal to zero defines an operating system default limit.

Example
`Names.MaxLength = 30`

Clear

Returns
None

Description
This method clears the text of the control and removes all items from the dropdown list of the control.

Example
`Names.Clear`

ListCount, ListIndex, List, AddItem, RemoveItem

Description
See properties and methods of DropDownList

SetFocus, Visible, Enabled, ControlTipText

Description
See common properties.

Multipage Control

Caption

Returns
String

Description
Sets or returns the text displayed in the selected tab of the Multipage control.

Example

```
Form.Pages.Value = 0  
Form.Pages.Caption = "First Tab"  
Form.Pages.Value = 1  
Form.Pages.Caption = "Second Tab"
```

Value

Returns

Integer

Description

Sets or returns the index of the selected tab. The pages are indexed starting at zero.

Example

```
ActTab = Settings.Value
```

Visible, Enabled, ControlTipText

Description

See common properties.

Chapter 139 - Pythagoras Data Types

XYZ

Fields

X As Double, Y As Double, Z As Double

Description

X,Y and Z coordinates. Mainly used for operations requiring 3D coordinates.

Example

```
Dim XYZValue As XYZ
Dim P2D As XY

XYZValue.X = 10
XYZValue.Y = 10
XYZValue.Z = 0
```

XY

Fields

X As Double, Y As Double

Description

X and Y coordinates. Mainly used for operations requiring 2D coordinates.

Example

```
Dim XYValue As XY

XYValue.X = 10
XYValue.Y = 10
```

XYV

Fields

X As Double, Y As Double, V() As Double
Or alternative
XY As XY, V() As Double

Description

X and Y coordinates, V : dynamic array of values

Example

```
Dim XYValue As XYV
Dim Pnt As XY

XYValue.X = 10
XYValue.Y = 10
XYValue.XY = Pnt
Redim XYValue.V(1 To 3)
```

KMLGeoPoint

Fields

Longitude As Double, Latitude As Double, Altitude As Double

Description

Longitude and Latitude coordinates in degrees decimal. E.g. 50.70356
-180 <= Longitude <= +180. -90 <= Latitude <= +90.
If used for export to KML: Lat. Long. in WGS84. Latitude is in meters.

Example

```
Dim Coord As KMLGeoPoint

Coord.Longitude = 1.227414
```

```
Coord.Latitude = 6.155460
Coord.Altitude = 17.0
```

RGB

Fields

Red As Byte, Green As Byte, Blue As Byte

Description

Use to read or set the true color.

Example

```
Dim Color As RGB

Color.Red = 100
Color.Green = 10
Color.Blue = 240
```

KMLColor

Fields

Red As Byte, Green As Byte, Blue As Byte, Alpha As Byte

Description

Use to read or set the color of KML objects.

Example

```
Dim Color As KMLColor

Color.Red = 100
Color.Green = 10
Color.Blue = 240
Color.Alpha = 255
```

Rectangle

Fields

P1 As XY, P2 As XY

alternative

x1 As Double, y1 As Double, x2 As Double, y2 As Double

Description

A rectangle is defined by 2 points, namely P1 and P2. Alternatively, the x and y values of the points can be accessed directly.

Warning : in some cases P1 must be top left point and P2 must be the bottom left point.

Example

```
Dim R As Rectangle

R.P1.X = 1.0
R.P1.Y = 2.0
R.P2.X = 10.0
R.P2.Y = 20.0

' following code gives the same result.
R.x1 = 1.0
R.y1 = 2.0
R.x2 = 10.0
R.y2 = 20.0
```

ObliqueRectangle

Fields

BottomLeft As XY, TopLeft As XY, TopRight As XY, BottomRight As XY

Description

The record type ObliqueRectangle consists of 4 points. These four points should define a rectangle which is not necessarily horizontal.

Example

```
Dim R As ObliqueRectangle

R.BottomLeft.x = 0.0
R.BottomLeft.y = 0.0
R.TopLeft.x = 0.0
R.TopLeft.y = 10.0
R.TopRight.x = 10.0
R.TopRight.y = 10.0
R.BottomRight.x = 10.0
R.BottomRight.y = 0.0
```

Transformation

Fields

```
Fi As Double, D As XYZ, ScaleFactor As Double
synonyms :
Fi : RotationAngle
D.X, D.Y, D.Z : dX, dY, dZ
```

Description

This record type defines the values to calculate a transformation from XY to X'Y'. Transformation is used in the Units object by all "Conversion" methods. For the exact meaning of the parameters, see the description of the Units method SetTransformation.

Example

```
Dim Tr As Transformation

Tr.Fi = 0.0           ' or Tr.RotationAngle = 0.0
Tr.D.X = 1000.0       ' or Tr.dX = 1000.0
Tr.D.Y = 1000.0       ' or Tr.dY = 1000.0
Tr.D.Z = 10.0         ' or Tr.dZ = 10.0
Tr.ScaleFactor = 1.0
```

TransformationParameters

Fields

```
RotationAngle As Double, dX As Double, dY As Double, dZ As Double, ScaleFactor
As Double.
synonyms : see Tranformation.
```

Description

This record is synonymous to the type Tranformation. It is used by the object HelmertTransformation.

Example

```
Dim Tr As TransformationParameters

Tr.RotationAngle = 0.0
Tr.dX = 1000.0
Tr.dY = 1000.0
Tr.dZ = 0.0
Tr.ScaleFactor = 1.0
```

Style

Fields

```
Symbol As Boolean, Group As String, Item As String, StyleId As Integer
```

Description

This record type is used to define point styles, line styles and hatch styles.

If the field Symbol = TRUE, the fields Group and Item need to be filled in. Group and Item define the style of the document or the system symbol.

If the field Symbol = FALSE, the field StyleId defines the standard styles provided by Pythagoras.

Example

```
Dim NewStyle As Style
Dim SelectedObjects As Selection

' Set the style of all selected points to Example, Doorstep.
With NewStyle
```

```

        .Symbol = True
        .Group = "Example"
        .Item = "Doorstep"
End With
SelectedObjects.PointStyle = NewStyle

' Set the style of all selected lines to Example, Wall.

With NewStyle
    .Symbol = True
    .Group = "Example"
    .Item = "Wall"
End With
SelectedObjects.LineStyle = NewStyle

' Set the style of all selected lines to DashDot.

With NewStyle
    .Symbol = False
    .StyleId = ptDashDot
End With
SelectedObjects.LineStyle = NewStyle

```

SelectionCriteria

Fields

```

SelectionType As Integer, Objects As Long,
Wallpaper As Boolean,
Pairing As PtPairing,
OverlappingPointSelection As PtOverlappingPointSelection,
PointDistance As Double, Distance As Double
OnElevation As Boolean, z_Selection As Integer, z As Double,
OnLength As Boolean, LengthSelection As Integer, Length As Double,
OnArea As Boolean, AreaSelection As Integer, Area As Double,
OnSubDocument As Boolean, SubDoc As SubDocument
OnLayer As Boolean, Layer As String,
OnColor As Boolean, Color As Integer,
OnTrueColor As Boolean, TrueColor As RGB,
OnPointStyle As Boolean, PStyle As Style,
OnLineStyle As Boolean, LStyle As Style,
OnLineWidth As Boolean, LWidth As Integer,
OnPattern As Boolean, Pattern As Style,
OnPointNumber As Boolean, FromPntNr As String, ToPntNr As String,
OnComment As Boolean, CommentSelection As Integer, CompareStr As String
OnAnSymbolStyle As Boolean, AnSymbolStyle As Style, AnSymbolType As Integer,

```

Description

This record type is currently only used as a parameter in the FlaggedObjects method MakeSet. The purpose is to provide criteria for building up FlaggedObjects.

SelectionType : must be either ptNewSelection, ptAddToSelectedObjects, ptOnSelectedObjects or ptRemoveFromSelection.

Field Wallpaper defines whether objects of wallpaper layers and wallpaper subdocuments are included. If this parameter is True then objects of all layers of all visible subdocuments are looked through.

Otherwise objects of wallpaper layers and of wallpaper subdocuments are included.

Value of OverlappingPointSelection and PointDistance are taken into account only when Pairing=ptOverlappingPoint. Value of Distance is taken into account only when Pairing=ptNearToSelected or ptFarFromSelected.

Each of z_Selection, LengthSelection and AreaSelection must be either ptEqual, ptAboutEqual, ptLessOrEqual or ptGreaterOrEqual. If it is ptAboutEqual, then the number of decimal places used for comparison is defined by the actual settings in the user preferences. Possible values of CommentSelection : see section Constants, *Compare*.

Conditions On... are applied only to objects of such types for which these conditions are meaningful. In contrary, when Pairing=ptOverlappingPoints or Pairing=ptOverlappingLines, then correspondingly only points or only lines will be found.

In case compound objects are to be included in the set no condition on Layer should be used since compound objects don't have a layer.

If `OnAnSymbolStyle` is true, then `AnSymbolType` should have the according symbol type of the symbol style. `AnSymbolType` can be either `ptPointerSymbol` or `ptRangeSymbol`.

`SelectionCriteria` can be initialised with the Application method `InitSelectionCriteria`.

Example

See the `FlaggedObjects` method `MakeSet`.

RowSetDefinition

Fields

`RowSetType` As `PtRowSetType`, `Name` As Variant (`String` or `DBQuery`),
`ObjectTypes` As Long, `Overlay` As `PtRowSetBaseOverlay`,
`SheetName` As String

Description

A reference to a rowset.

`Name` is a name of a DB table or a DB view or a temporary query; it is ignored for object sets.

`ObjectTypes` is set of CAD object types to be included to a rowset, it is ignored for tables and queries.

It must be defined in the same way as a value of class `DBTable` property `ObjectTypes`.

`Overlay` is meaningful only when `RowSetType=ptObjectSet`, it defines objects of what overlay are included to the rowset.

`SheetName` is meaningful only when `RowSetType=ptObjectSet` and

`Overlay<>ptDrawingOverlay`, it defines objects of what sheet are included to the rowset.

`SheetName=""` means that objects of the active sheet are included.

This data type is used to define base rowsets for queries.

Note that it is senseless to specify `Name` as a name of system table. Use instead

`RowSetType=ptObjectSet` with appropriate value of `ObjectTypes`.

Example

```
Dim RowSet As RowSetDefinition
```

```
RowSet.RowSetType = ptObjectSet
```

```
RowSet.ObjectTypes = ptLine + ptArc + ptClothoid + ptCircle
```

SelectCondition

Fields

`Column` As Variant
`Compare` As `PtCompare`
`Value` As Variant

Description

This record type is used as a parameter in the `FlaggedObjects` method `MakeSetDB`. The purpose is to provide criteria for building up `FlaggedObjects`. This record is also returned by the Application method `SelectCondition`.

`Column` : Variant (Subtype of `Column` is either `Integer` or `String`). When this value is NULL `SelectCondition` value is ignored. This feature may be useful to skip elements in an array of `SelectCondition` values.

`Compare`: must be either `ptCompareEqual`, `ptCompareLessOrEqual`,
`ptCompareGreaterOrEqual`, `ptCompareStartsWith`, `ptCompareContains`,
`ptCompareEndsWith`, `ptCompareIsNull` or `ptCompareIsNotNull`.

`Value`: Variant. Subtype depends on the type of data in the column.

Example

See the `FlaggedObjects` method `MakeSetDB` and the Application method `SelectCondition`.

MatchCondition

Fields

`Kind` As `PtMatchConditionKind`,
`Expression` As String, `Distance` As Double

Description

A match condition in an aggregation query.

When `Kind = ptConditionExpression` then `Expression` is a match condition.

When `Kind = ptConditionDistance` then a match condition is the following: distance between the main and the auxiliary objects is not more than `Distance` expressed in length units used for this query.

Note that so far as distance is always non-negative, definition `Distance=0` . actually defines a condition that distance between objects is 0 (i.e. they either touch or overlap).

When `Kind = ptConditionEncloses` then a match condition is the following: the auxiliary object lies inside the main object.

When `Kind = ptConditionInside` then a match condition is the following: the main object lies inside the auxiliary object.

Example

```
Dim Match As PtMatchCondition

' "Capital" column value in the main base equals to
' "Name" column value in the auxiliary base

Match.Kind = ptConditionExpression
Match.Expression = "Main.Capital=Aux.Name"
```

WhereCondition

Fields

```
Compare As PtCompare,
Feature As PtObjectFunction, Column As String, Value As Variant,
ValueType As Byte, Aggregation As PtAggregateFunction
```

Description

A 'where' condition in a query. An object property or a column value is compared with some value.

Permitted values for `Compare` are `ptCompareEqual`, `ptCompareGreater`, `ptCompareLess`, `ptCompareGreaterEqual`, `ptCompareLessEqual`, `ptCompareNotEqual`, `ptCompareStartsWith`, `ptCompareContains`, `ptCompareEndsWith`, `ptCompareIsNull`, `ptCompareIsNotNull`, `ptCompareIsTrue` and `ptCompareIsFalse`.

When `Feature = ptFunctionEmpty` then a value of column with name `Column` is compared with `Value`, otherwise the corresponding property of the CAD object is compared with `Value`.

When `Aggregation = ptFunctionNil` then a not aggregated value of the main dimension is compared. Other values may be specified only for aggregation queries and mean that an aggregated value of the auxiliary dimension is compared.

`Value` is ignored when `Compare` is `ptCompareIsNull`, `ptCompareIsNotNull`, `ptCompareIsTrue` or `ptCompareIsFalse`.

`ValueType` defines a type of `Value` and in most cases is equal to actual type of `Value`. Only when the where condition is applied to a column of type substandard for VBA (`ptDateOnly`, `ptTime` or `ptCadObjectType`) and `Aggregation<>ptFunctionCount`, `ValueType` is equal to one of these values. In the last case `Value` should have type `vbDate`, `vbDate` or one of integer types correspondingly.

Example

```
Dim Where As WhereCondition

' Sum of all Weight column values is less than 5

Where.Feature = ptFunctionEmpty
Where.Column = "Weight"
Where.Compare = ptCompareLess
Where.Value = 5
Where.ValueType = vbLong
Where.Aggregation = ptFunctionSum
```

ColumnDefinition

Fields

```
Name As String, Description As String,
IsComputed As Boolean, RowSet As String, Expression As String,
Aggregation As PtAggregateFunction,
Format As String, Fractional As Byte
```

Description

A definition of a query column.

Name and Description are correspondingly a name and textual description of the column.

Fields IsComputed, RowSet, Expression and Aggregation have the following meanings in the following cases:

- When IsComputed = True for extension or aggregation queries: this column is a computed one; Expression is a textual expression to compute; the other fields are ignored.
- When IsComputed = False for extension queries: this column is a reference to a base rowset column; RowSet contains an alias name of a base rowset, if it is empty then the main base rowset is referenced; Expression is a name of the referenced column; Aggregation is ignored.
- When IsComputed = False and Aggregation = ptFunctionNil for aggregation queries: this column is a reference to a main base rowset column; Expression is a name of the referenced column; RowSet is ignored.
- When IsComputed = False and Aggregation = ptFunctionCount for aggregation queries: this column is a count reference to the auxiliary base rowset; the other fields are ignored.
- When IsComputed = False and Aggregation is one of ptFunctionMin, ptFunctionMax, ptFunctionSum and ptFunctionAvg for aggregation queries: this column is an aggregated reference to an auxiliary base rowset column; Expression is a name of the column to be aggregated; Property and RowSet are ignored.

Format and Fractional define how column data are to be converted to textual form when displaying. Format is meaningful only for date/time type columns, it must contain a string complying with simplified ICU date/time format (see Pythagoras Reference Guide, the Defaults Menu, Preferences, Date and Time) Fractional is meaningful only real type column, it defines how many fractional decimal digits is to be displayed.

Example

```
Dim ColDef As ColumnDefinition
Dim Query As DBQuery
Dim ColNo As Integer

' Define a column for an extension view being a mirror
' of column "Name" of dimension "Person"

ColDef.Name = "OwnerName"
ColDef.Description = "Full name of an owner of the parcel"
ColDef.IsComputed = False
ColDef.RowSet = "Person"
ColDef.Expression = "Name"
ColNo = Query.AddColumn(ColDef)
```

DocInfo

Fields

Info1 As String, Info2 As String, Info3 As String, Info4 As String, Info5 As String, Info6 As String, Info7 As String, Info8 As String, Info9 As String,

Description

Fills in the Document Information Form

Example

```
Dim A As Application
Dim D As Document
Dim Txt As DocInfo
Dim Tst As DocInfo

Set A = New Application
A.GetActiveDocument D
If Not (D Is Nothing) then
    Txt.Info1 = " dit "
    Txt.Info2 = " is "
    Txt.Info3 = " een test natuurlijk "
    D.Info = Txt
    Tst = D.Info
    MsgBox Tst.Info1 & vbNewLine & Tst.Info2 & vbNewLine & Tst.Info3
End If
```

```
Set A = Nothing
Set D = Nothing
```

WindowCoord

Fields

X As Long, Y As Long

Description

Creates a 3D point with the given coordinates

Example

```
Dim A As Application
Dim D As Document
Dim W As Window
Dim P As WindowCoord

set A = New Application
A.GetActiveDocument D

If NOT D is nothing Then
    D.GetWindow W
    p = W.Position
    MsgBox "Position : " & str(P.x) & " , " & str(P.y)
    P.x = P.x + 24
    P.y = P.y - 30
    W.Position = P
    W.SetWindowSize W.Width\2, W.Height\2
end if
```

Line2D

Fields

P1 As XY, P2 As XY

Description

This record type defines a 2D line consisting of 2 points.

Example

```
Dim L As Line2D
Dim Pnt1 As XY, Pnt2 As XY

L.P1 = Pnt1
L.P2 = Pnt2
```

Line3D

Fields

P1 As XYZ, P2 As XYZ

Description

This record type defines a 3D line consisting of 2 points.

Example

```
Dim L As Line3D
Dim Pnt1 As XYZ, Pnt2 As XYZ

L.P1 = Pnt1
L.P2 = Pnt2
```

LinePair

Fields

L1 As CadObject, L2 As CadObject

Description

This record type contains may contain 2 Cadobjects. The CadObjects must be either of type Line or Arc. Any combination is allowed.

Example

```
Dim Lines As LinePair
```

```

Dim L As Line, A As Arc
Dim Obj As CadObject

L.GetCadObject Obj
Set Lines.L1 = Obj
A.GetCadObject Obj
Set Lines.L2 = Obj

```

ColumnValue

Fields

Column As Variant, Value As Variant

Description

This record type is used to read and write data from/to a row of a table.

Column is either the name or the number of the column. Numbering of columns starts from 0. Column may be = Null. When = Null, this means that this ColumnValue may not be taken into account when the data is assigned to a row.

Value : the value of the Column. When used to read/write rows of a table, the subtype of Value should correspond with the type of the column in the table or be Null.

See also object *DBRow*.

Example

```

Dim C As ColumnValue

C.Column = 5
C.Value = 123.55

```

ValueIndex

Fields

Value As Variant, Index As Long

Description

This record type is used to help to sort records of any type. For more information see the Application method SortValueIndex.

Example

```

Dim D As ValueIndex

C.Value = "My name"
C.Index = 55

```

UUID

Fields

No accessible fields.

Description

This record describes Universally Unique Identifier (see OpenGroup standard <http://www.opengroup.org/onlinepubs/9629399/apdx.htm>), also known as GUID (Globally Unique Identifier).

ProfileElement

Fields

IP As XY,	' Intersection Point
BC As XY,	' Begin of Curve
EC As XY,	' End of Curve
VCControl As Integer,	' ptVCAutomatic, ptVCFixedRadius or ptVCFixedLength
VCTYPE As Integer	' ptVCCircle or ptVCParabola
Center As XY	' Center of circular vertical curve.
	' Not given in case of Parabola
FixedRL As Double	' The predefined Radius (ptVCFixedRadius)
	' or Length (ptVCFixedLength)

```

Radius As Double          ' Actual Radius
Length As Double          ' Actual Length

```

Description

The record type `ProfileElement` contains all information related to a vertical curve in a profile. The first and the last point of the profile are also vertical curves with `Radius = 0` and `Length = 0`.

Example

See Road method `GetProfileData`

HorizontalTransition

Fields

```

IP As XY,                ' Intersection Point
Automatic As Boolean
Type As Integer          ' See constants: Road Horizontal Aligment Transition Type
NrOfElements As Long
Elements(1 To 3) As PathElement ' =3 if Spiral-Arc-Spiral

```

Description

The record type `HorizontalTransition` contains all information related to the horizontal transition.

Example

See Road method `GetHorIPData`

SuperElevationPoint

Fields

```

Type As Integer          ' ptNCIn, ptHCIn,ptFSEIn ptFSEOut, ptHCOut, ptNCOut
Pnt As XY                ' Coordinates of the point

```

Description

The record type `SuperElevationPoint` contains information (type and coordinates) related to a point in the superelevation transition.

Example

See data type `SuperElevation`

SuperElevationData

Fields

```

SuperElevated As Boolean          ' True if superelevated
AutomaticMaximum As Boolean       ' If True, FullSuperElevation
                                   ' is calculated by Pythagoras.
FullSuperElevation As Double     ' The amount of Superelevation
AutomaticPositions As Boolean     ' The positions are automatically defined.
NrOfPoints As Long
Points(1 To 6) As SuperElevationPoint

```

Description

The record type `SuperElevationData` contains all information related to the superelevation of one horizontal transition.

Example

See Road method `GetSuperElevationData`

SystemParameters

Fields

```

DWGExportAttribs As Boolean,      ' Default = True
DWGExportZAsAttrib As Boolean,    ' Default = False
DWGExportZAttribLabel As String,  ' Default = "Z"
DWGImportUCS As Boolean,          ' Default = True
DWGExport2D As BOOLEAN            ' Default = False

```

Description

This record type defines all system parameters that can be modified with Pythagoras VBA.
`DWGExportAttribs` : used when writing DWG/DXF files. If = True, BLOCKS (Pythagoras symbols) will have the DWG/DXF attributes "PointNumber" and "Comment". If = False, the attributes are not exported.

DWGExportZAsAttrib : used when writing DWG/DXF files. If = True, the elevation of the point will be exported as an attribute.

DWGExportZAttribLabel: used when writing DWG/DXF files. The label of the attribute for the elevation of the point. Only used if DWGExportZAsAttrib = True.

DWGImportUCS : used during import of DWG/DXF files. If = True, data in the DWG/DXF file will be converted to the actual User Coordinate system set in the DWG file.

DWGExport2D : used when writing DWG/DXF files. If = True, all elevations in the DWG/DXF file will become equal to 0.0.

The number of fields in this type will grow in future versions of Pythagoras.

Example

See application property `SystemParameters`.

Font

Fields

Source As Byte,
Name As String,
CodePage As Integer

Description

Source is either `ptSystemFont` or `ptPythagorasVectorFont`, CodePage can be either `ptWestern` or `ptSymbol`. The last value defines that the font is a dingbat one.

Example

See `Fonts` property `GetFirst`, `GetNext`.

Location

Fields

Entity As String, Line As Long, Column As Long

Description

A value of this type describes location inside a text entity. The type is useful to describe position of an error during syntactic analysis. Now it is used only in `Application.GetXMLErrorPosition` method.

Entity can contain a name of a textual entity, for example, a path to a file.

If the entity is a file then Line is a number of a line in the file and Column is a number of a character in the line. Otherwise Line = 0 and Column is a number of a character in the whole entity. Numeration of lines and columns starts form 1.

Example

See example `XMLDocument`

DTMReduce

Fields

Reduce As Boolean, Density As Double

Description

Parameters for reduction calculation. If Reduce is False then no reduction will be applied and other record fields are not considered. The Density parameter value indicates the maximum point density per square meter. In areas with higher density some points with high Z values are skipped. In areas with lower density all points are used.

VolumeCalculationDTM

Fields

Elevation As Double, UseDistance As Boolean, NrOfCrossSections As Long,
Distance As Double, NrOfBoundaryPoints As Long, BoundaryPoints() As XY

Description

A value of this type is used to group the input parameters for the volume calculation of a terrain model relative to a horizontal elevation level. You can choose between the distance and the number of cross sections to determine the number of cross sections used in the calculation.

VolumeCalculationDTMs

Fields

LMName1 As String, LMName2 As String, UseDistance As Boolean, NrofCrossSections As Long, Distance As Double, NrofBoundaryPoints As Long, BoundaryPoints() As XY

Description

A value of this type is used to group the input parameters for the volume calculation of a terrain model relative to another terrain model. You can choose between the distance and the number of cross sections to determine the number of cross sections used in the calculation.

VolumeCalculationResult

Fields

Extrapolated As Boolean, Accuracy As Double, VolumePlus As Double, VolumeMin As Double, ExtrapolatedVolumePlus As Double, ExtrapolatedVolumeMin As Double

Description

A value of this type is used for the result of the volume calculation. It holds the positive and negative volume in cubic meters. And the extrapolated positive and negative values in case the calculation required extrapolated. The accuracy of the extrapolation process is also added; this is a value between 0 and 100.

VectoriseData

Fields

type As VectorisationType,
isBox As Boolean,
BLN As XYZ, TLN As XYZ, BRN As XYZ, BLF As XYZ
p1 As XYZ, p2 As XYZ

Description

This record type is used as a parameter in the Document method Vectorise and the Application method InitVectoriseData. The purpose is to provide all the necessary data for the vectorisation of a point cloud.

type: must be either ptVectoriseSection, ptVectorise3DEdges or ptVectorise3DLines.

isBox: If this parameter is True then the points in BLN, TLN, BRN and BLF indicate the section box. The point cloud points in this box are considered for the vectorisation. If this parameter is False, the points in p1 and p2 indicate two parallel planes which form the section plane. The point cloud points inside these two planes are considered for the vectorisation.

When no section box or section plane is required these points should contain all zeros.

For ptVectoriseSection, there needs to be a section box or section plane, because one of the planes is used to vectorise on. When there is a section plane, the plane containing the first point, is the one to vectorise on. When there is a section box, the face of the box with the biggest area is chosen. From these two planes, the plane which contains three of the given points is vectorised on.

Example

See the Document method Vectorise.

Chapter 140 - Constants

Language Constants

Button Display constants

vbOkOnly
vbOkCancel
vbAbortRetryIgnore
vbYesNoCancel
vbYesNo
vbRetryCancel

Icon Display constants

vbCritical
vbQuestion
vbExclamation
vbInformation

Default button constants

vbDefaultButton1
vbDefaultButton2
vbDefaultButton3
vbDefaultButton4

Modality constants

vbApplicationModal
vbSystemModal

MsgBox return value constants

vbOK
vbCancel
vbAbort
vbRetry
vbIgnore
vbYes
vbNo

Types constants

vbEmpty
vbNull
vbInteger
vbLong
vbSingle
vbDouble
vbDate
vbString
vbObject
vbError
vbBoolean
vbByte
vbUserDefinedType
vbArray
vbLongLong

Compare constants

vbBinaryCompare
vbTextCompare
vbDatabaseCompare

String constants

vbBack
vbCr
vbCrLf

```
vbFormFeed
vbLf
vbNewLine
vbNullChar
vbNullString
vbTab
vbVerticalTab
```

String Conversion constants (vbStrConv)

```
vbFromUnicode
vbHiragana          -- ignored in current version
vbKatakana          -- ignored in current version
vbLowerCase
vbNarrow             -- ignored in current version
vbProperCase         -- ignored in current version
vbUnicode
vbUpperCase
vbWide              -- ignored in current version
```

Day of week constants

```
vbUseSystem
vbSunday
vbMonday
vbTuesday
vbWednesday
vbThursday
vbFriday
vbSaturday
```

File System constants

```
vbAlias
vbArchive
vbDirectory
vbHidden
vbNormal
vbReadOnly
vbSystem
vbVolume
```

Pythagoras Object Model Constants

The numeric values of the constants listed below are intentionally not documented. The values may change between versions of Pythagoras.

General

```
ptMaxDouble
ptMinDouble
```

Ok - Not Ok

```
ptOk
ptNotOk
```

SortOrder

```
ptSortAlphabetical
ptSortAlphaNumerical
```

Group

```
ptNoGroup
```

Language

```
ptAppEnglish
ptAppGerman
ptAppFrench
```

ptAppDutch
ptAppGreek
ptAppPolish
ptAppItalian
ptAppRussian
ptAppSerbian
ptAppSerbianCyrillic
ptAppThai
ptAppSpanish
ptAppRomanian
ptAppBulgarian
ptAppFinnish
ptAppIndonesian
ptAppArabic
ptAppPersian
ptAppHungarian
ptAppCroatian
ptAppKorean
ptAppSlovak
ptAppLithuanian
ptAppVietnamese
ptAppAlbanian
ptAppBosnian
ptAppPortuguese
ptAppChineseTraditional
ptAppChineseSimplified
See Application method Language.

State of Subdocument

ptInvisibleSubDocument
ptWallPaperSubDocument
ptProtectedSubDocument
ptEditableSubDocument

SaveBeforeClose

ptDontSave
ptAskUser
ptSaveAs

Overlay type

ptPaper
ptDrawing

Tool type

ptToolSelect
ptToolZoomOut
ptToolZoomIn
ptToolFitWindow
ptToolDragWindow
ptToolPoint
ptToolFreeLine
ptDefineParallelLine
ptToolPerpendicularLine
ptToolArcTang1Line
ptToolArcTang2Lines
ptToolArc3P
ptToolArcCenterR
ptToolCircleCenterR
ptToolCircle3P
ptToolXAxis
ptToolPolygon
ptToolPath
ptToolOpenCurve
ptToolClosedCurve
ptToolEditCurve
ptToolText

ptToolRichText
 ptToolObliqueText
 ptToolObliqueRichText
 ptToolAnSegmentLengths
 ptToolAnPolygonArea
 ptToolAnPolygonLengthsAndArea
 ptToolAnCoordinates
 ptToolAnPointNumber
 ptToolAnElevation
 ptToolAnXYMinusOffset
 ptToolAnComment
 ptToolAnHorizontalLength
 ptToolAnSlopeLength
 ptToolAnTotalLength
 ptToolAnBearing
 ptToolAnDifferenceInHeight
 ptToolAnGradient
 ptToolAnRadius
 ptToolAnAngle
 ptToolAnDeltaX
 ptToolAnDeltaY
 ptToolAnVolume
 ptToolAlignLeft
 ptToolAlignHorizontalCentered
 ptToolAlignRight
 ptToolAlignTop
 ptToolAlignVerticalCentered
 ptToolAlignBottom
 ptToolAlignObject
 ptToolFreeClothoid
 ptToolClothoidLineCircle
 ptToolClArcClTang2Lines
 ptToolCutOffInsideAllObjects
 ptToolCutOffInsideWithoutImages
 ptToolCutOffInsideImages
 ptToolCutOffInsidePointClouds
 ptToolCutOffOutsideAllObjects
 ptToolCutOffOutsideWithoutImages
 ptToolCutOffOutsideImages
 ptToolCutOffOutsidePointClouds
 ptToolErasePointCloud
 ptToolEraseImageArea
 ptToolEraseImageAreaWhite
 ptToolEraseImageAreaBlack
 ptToolClipPointCloud
 ptToolClipImageArea
 ptToolClipImageAreaWhite
 ptToolClipImageAreaBlack
 ptToolCreateParallelObject
 ptToolCutLine
 ptToolDivideLine
 ptToolExtendLine
 ptToolJoinLines
 ptToolIntersectObjects

Colors

ptBlack
 ptRed
 ptGreen
 ptYellow
 ptBlue
 ptMagenta
 ptWhite
 ptGray

ptCyan

Length Units

ptMM
ptCM
ptDM
ptM
ptDAM
ptHM
ptKM
ptInch
ptMiles
ptFeet
ptUSfeet

Angle Units

ptGRAD : GRAD or GON : 400 GRAD = 360 degrees
ptDEGdec : Degrees decimal : 0.50 = 0°30'0"
ptDEGMinSec : Degrees Minutes Seconds : 0.453020 or 0°30'20"
ptMIL : 6400 MIL = 360°
ptSURVEY : DMS and bearing notation : N45°E
ptRADIANT : radians 2pi = 360°

Vertical Angle

ptVAlevel : vertical angle is Level based. Horizontal = 0°
ptVA90 : vertical angle is relative to zenith. Horizontal = 90°
ptVA270 : vertical angle is relative to zenith. Horizontal = 270°

XYZ Representation

ptRectangular : coordinates : NEZ, XYZ, ... depending on
Representation of rectangular coordinates
ptPolar : HDZ : Hor. Angle, Hor. Distance, Elevation
ptHVD : HDZ : Hor. Angle, Vert. Angle, Hor. Distance
ptHVS : HDZ : Hor. Angle, Vert. Angle, Slope Distance

Representation of rectangular coordinates

ptMathematicalXY
ptNorthEast
ptGermanYX
ptUserDefined

Angle type

ptHorAngle
ptVertAngle
ptArcAngle
ptBearingAngle

Scale Format

ptScaleImperial
ptScaleMetric

Chainage Format

ptChainageFormatNormal
ptChainageFormatPlus

Point Style

ptHiddenPoint
ptCross
ptXCross
ptHalfXCross
ptRoundDot
ptRoundDotHollow
ptRectDot
ptRectDotHollow
ptTriangle
ptSplitPoint

ptReferencePoint
ptHotspot

Line Style

ptSolid
ptDash
ptDot
ptDashDot
ptDashDotDot
ptHiddenLine

AnnotationStyle

ptRangeOnlyText
ptExternalRange
ptFullRange
ptFullRangeDotted
ptFullRangeSlashes
ptFullRangeVertSlashes
ptFullRangeExtensions
ptFullRangePullUp
ptIntermittedRange
ptIntermittedRangeDotted
ptIntermittedRangeSlashes
ptIntermittedRangeVertSlashes
ptIntermittedRangeExtensions
ptIntermittedRangePullUp

ptPointerTextOnly
ptPointerOnlyTextOnObject
ptSimplePointerLeftOfText
ptSimplePointerUnderText
ptPointerToCorner
ptPointerBroken
ptPointerBrokenToVerticalSide
ptPointerWithVerticalSlash
ptSimplePointerUnderlinedText
ptRadiusPointer
ptCirclePointer
ptPointerSidewaysText
ptPointerManHoleBroken
ptPointerManHoleBrokenToVerticalSide

Arrows on end point of lines

ptNoArrows
ptArrowP1
ptArrowP2
ptArrowP1P2

Document Version

ptV10
ptV11
ptV12
ptDXF13
ptDXF14
ptDXF2000
ptDXF2004
ptDXF2007
ptDXF2010
ptDXF2013
ptDXF2018
ptDWG13
ptDWG14
ptDWG2000
ptDWG2004
ptDWG2007
ptDWG2010
ptDWG2013
ptDWG2018
ptGRBFile2D

ptGRBFile2_5D
ptGRBFile3D
ptLastVersion

Selection

ptNewSelection
ptAddToSelectedObjects
ptOnSelectedObjects
ptRemoveFromSelection

Compare

ptCompareNil
ptCompareEqual
ptCompareGreater
ptCompareLess
ptCompareGreaterEqual
ptCompareLessEqual
ptCompareNotEqual
ptCompareStartsWith
ptCompareContains
ptCompareEndsWith
ptCompareIsNull
ptCompareIsNotNull
ptCompareIsTrue
ptCompareIsFalse
ptCompareAboutEqual

Type of intersection

ptPartiallyOverlapping
ptIdenticalLines
ptIntersectingLines
ptCommonEndPoint

ObjectTypes

ptNoObjects
ptPoint
ptLine
ptArc
ptClothoid
ptText
ptRichText
ptImage
ptCurve
ptPath
ptPolygon
ptCircle
ptCoordinateSystem
ptRoad
ptCompound
ptAnnotation
pt3DSolid
ptAll
ptAllDBLinkable

ElementType

ptObject
ptVertex
ptSegment

AnnotationKind

ptAn_XYOffset
ptAn_TotalLength
ptAn_Coordinate
ptAn_Elevation
ptAn_PointId
ptAn_Area
ptAn_LineLength

```
ptAn_Comment
ptAn_Bearing
ptAn_SlopeDistance
ptAn_Dz
ptAn_Gradient
ptAn_Angle
ptAn_Radius
ptAn_DeltaX
ptAn_DeltaY
ptAn_Volume
ptAn_GeneralAnnotation
```

AnnotationType

```
ptPointerAnnotation
ptRangeAnnotation
ptAngleAnnotation
ptRadiusAnnotation
ptDeltaXAnnotation
ptDeltaYAnnotation
```

Font source

```
ptSystemFont           (Font installed in OS)
ptPythagorasVectorFont
```

Codepage

```
ptWestern
ptEastEuropean
ptGreek
ptCyrillic
ptArabic
ptThai
ptHebrew
ptTurkish
ptVietnamese
ptJohab
ptBaltic
ptSymbol
```

All values except the first and the last are obsolete. Don't confuse Codepage with language of Pythagoras.

Text Style

```
ptUpright
ptItalic
```

Text Weight

```
ptLight
ptMedium
ptBold
ptExtraBold
```

Text: horizontal alignment

```
ptAlignLeft
ptAlignCenter
ptAlignRight
```

Text: vertical alignment

```
ptAlignBottom
ptAlignCenter
ptAlignTop
```

Text: underline

```
ptUnderlineNone
ptUnderlineSingle
ptUnderlineDouble
ptUnderlineDashed
```

Text: border

ptBorderNone
ptBorderSingle
ptBorderDouble

Text: line spacing

ptSpacingSmall
ptSpacingNormal
ptSpacingLarge

Text: orientation

ptTextHorizontal
ptTextOrientationReadable
ptTextOrientationAbsolute

Images: ColorDepth

ptMonochrome
ptPallet256
ptTrueColor
ptQuadColor

Polygon: styleId

ptNoFill
ptHorizontalFill
ptVerticalFill
ptBDiagonalFill
ptFDiagonalFill
ptCrossFill
ptBoundary1
ptBoundary2
ptBoundary1Open
ptBoundary2Open
ptSolidFill
ptGray75
ptGray50
ptGray25
ptGray12

Polygon: border

ptNoBorder
ptBorder1mm
ptBorder2mm
ptBorder3mm
ptBorder4mm
ptBorder5mm
ptBorder6mm
ptBorder7mm
ptBorder8mm
ptBorder9mm
ptBorder10mm
ptBorder1mmOpen
ptBorder2mmOpen
ptBorder3mmOpen
ptBorder4mmOpen
ptBorder5mmOpen
ptBorder6mmOpen
ptBorder7mmOpen
ptBorder8mmOpen
ptBorder9mmOpen
ptBorder10mmOpen

KML: Type of KML Objects

ptKMLPlacemark
ptKMLFolder
ptKMLPoint
ptKMLLineString
ptKMLPolygon

KML: Altitude Mode

ptClampToGround
ptRelativeToGround
ptAbsoluteAltitude

PTPosition

ptBeginning
ptEnd
ptBefore
ptAfter

Results from PointInPolygon method

ptInside
ptOutside
ptOnEdge
ptOnVertex

Type of coordinate system

ptLocalCS
ptUserCS
ptGlobalCS
ptPageCS
ptTempCS
ptProfileCS

Paste operations

ptPasteNormal
ptMirrorX
ptMirrorY
ptZToY
ptClearPointNumbers
ptAdjustSketch

Symbol Type

ptPointSymbol
ptLineSymbol
ptPattern
ptSectionSymbol
ptPointerSymbol
ptRangeSymbol

View: Visible attributes

ptPointNumbers
ptPointComments
ptComments
ptPointElevations
ptAreas
ptTexts
ptAnnotations
ptImages
ptSpecialLineStyle
ptPatterns

Symbol Class

ptLibrarySymbol
ptDocumentSymbol

Database: type of table constants

ptAttrTable
ptDictionary

Database: name category constants

ptStrictRegularName
ptRegularName
ptIrregularName

ptInvalidName

Database: table access constants

ptNoAccess
ptReadOnly
ptReadWrite
ptFullAccess

Database: reaction on pending references constants

ptPendingLeave
ptPendingNull
ptPendingDelete

Database: ptRowSetType

ptTable	A rowset is a DB table
ptQuery	A rowset is a DB query (including DB view and system table)
ptObjectSet	A rowset is a set of CAD objects of certain types

Database: RowSet Base Overlay (PtRowSetBaseOverlay)

ptDrawing	A rowset is based on the drawing overlay
ptPaper	A rowset is based on the paper overlay
ptCurrentOverlay	A rowset is based on the current overlay (either the drawing or the paper)

Database: PtQueryType

ptQueryExtension	A DB query is a simple or extension query
ptQueryAggregation	A DB query is an aggregation query

Database: PtMatchConditionKind

ptConditionExpression	Textual expression used to express condition
ptConditionDistance	Condition is Distance < constant
ptConditionEncloses	Auxiliary object is inside main object
ptConditionInside	Main object is inside auxiliary object

Database: PtObjectFunction

ptFunctionEmpty	No object function is defined
ptFunctionType	
ptFunctionSubdocument	
ptFunctionTable	
ptFunctionLayer	
ptFunctionDisplayLevel	
ptFunctionComment	
ptFunctionPointNumber	
ptFunctionZ	
ptFunctionLength	
ptFunctionDeltaZ	
ptFunctionSlopeLength	
ptFunctionMinZ	
ptFunctionMaxZ	
ptFunctionRadius	
ptFunctionAperture	
ptFunctionArea	
ptFunctionPerimeter	
ptFunctionN	
ptFunctionE	
ptFunctionSlopePerimeter	
ptFunctionKFactor	
ptFunctionStartingRadius	
ptFunctionEndRadius	
ptFunctionFileName	
ptFunctionVisible	

Database: PtAggregateFunction

ptFunctionNil	No aggregation is assumed

ptFunctionCount
 ptFunctionMin
 ptFunctionMax
 ptFunctionSum
 ptFunctionAvg

Database: PtSyntaxError

ptSyntaxSuccess	
ptSyntaxOverflow	Too big integer value
ptSyntaxUnlimString	String constant has no right delimiter
ptSyntaxUnlimColumn	Column name enclosed in apostrophes has no right delimiter
ptSyntaxBadLexeme	Unrecognized lexeme
ptSyntaxUnknownFunction	Unrecognized function name
ptSyntaxNotAttribute	CAD specific function is applied to a non-attribute table or query
ptSyntaxNoColumn	Column with such name not found
ptSyntaxNoFunction	Function with such name not found
ptSyntaxRightExpected	Right parenthesis is expected
ptSyntaxCommaExpected	Comma is expected
ptSyntaxCommaOrRight	Comma or right parenthesis is expected
ptSyntaxWrongArgType	Wrong argument type
ptSyntaxDifferentTypes	Types of arguments are different
ptSyntaxExprExpected	Expression expected
ptSyntaxNonArithmeticalType	Arithmetical operator is applied to an expression of improper type
ptSyntaxMultiplyTimes	Time cannot be multiplied by time
ptSyntaxDivideByTime	Numeric cannot be divided by time
ptSyntaxWrongSignType	Wrong type for unary + or -
ptSyntaxWrongConcatType	& is applied to not a string
ptSyntaxWrongAddTypes	Wrong type combination for addition
ptSyntaxWrongSubtractTypes	Wrong type combination for subtraction
ptSyntaxNotOrdered	Wrong type to compare on less or greater
ptSyntaxExtra	Something extra is present at the end of text
ptSyntaxResultType	Wrong type of the whole expression
ptSyntaxAmbiguousColumn	Column name is not prefixed and column with this name are present more than one dimension
ptSyntax2Prefixes	Two consecutive prefixes, for example, main.aux.
ptSyntaxWrongPrefix	Prefix is neither MAIN nor AUX for aggregation query
ptSyntaxExtraPrefix	Prefix is used in wrong location
ptSyntaxV11	A token is met that cannot be stored to Pythagoras v11 document
ptSyntaxAggrFunc	Aggregate function is not permitted here
ptSyntaxAuxAggrFunc	Aggregate function applied to an auxiliary base rowset is not permitted here
ptSyntaxMainAggrFunc	Aggregate function applied to the main base rowset is not permitted here
ptSyntaxWrongAlias	Prefix defines dimension prohibited here
ptSyntaxNoAlias	Prefix is not a dimension name for extension query

Database: PtQueryValidity

ptQueryValid	A query is a valid
ptQueryCircularQueries	Circular reference of queries
ptQueryBadAliasName	
ptQueryAliasNameRepeated	
ptQueryBadViewName	
ptQueryNoReferencedView	Reference to a view that does not exist
ptQueryNoReferencedTable	Reference to a table that does not exist
ptQueryBadLeftMatch	Error in the left match expression
ptQueryNoRightMatch	Match column not found
ptQueryNotUniqueMatch	Match column is not unique

ptQueryNoReferencedAlias	A where condition references not existing base
ptQueryNoReferencedColumn	A where condition references not existing column
ptQueryBadCondition	A where condition contains inconsistency
ptQueryBadWhereExpression	An error in where expression
ptQueryBadColumnExpression	An error in column expression
ptQueryNoMirroredColumn	A direct column reference references not existing column
ptQueryBadAggregationMatch	Incorrect match expression for an aggregation query
ptQueryBadSpatialAggregation	An attempt to build a spatial aggregation when one of base rowsets is not an attribute one
ptQueryBadAggregateType	An attempt to apply an aggregate function to not proper data type column
ptQueryBadColumnName	Some column has invalid name
ptQueryColumnNameRepeated	Column name is repeated
ptQueryNoReferencedSheet	Reference to a sheet that does not exist
ptQueryReferencesInvalidQuery	Reference to an invalid query
ptQueryNoReferencedQuery	Reference to a temporary query that has been deleted
ptQueryViewReferencesTempQuery	View base is a temporary query

Table Import: PtImportCompletion

A value of this type describes whether table import has been completed and a reason why it has not been completed.

ptImportSuccess	
ptImportSuccessNull	Imported, but some bad data were replaced by NULL values
ptImportNoFile	A file to import is not found or is locked
ptImportWrongDBFHeader	An error in DBF file header (probably it is not a DBF file)
ptImportWrongDBFSize	Size of DBF file contradicts data in the header
ptImportEmptyFieldName	Empty field name in DBF file header
ptImportWrongVersion	Wrong or unsupported DBF file type
ptImportNoFields	Empty field list in DBF file header
ptImportInvalidFieldDesc	Error in DBF field description
ptImportWrongFieldType	Wrong or unsupported column type
ptImportBadTableName	
ptImportBadColumnName	
ptImportTableExists	A table with such name already exists in the database and cannot be dropped

Vertical Curve : Fixed/Variable Radius or Length

ptVCAutomatic
ptVCFixedRadius
ptVCFixedLength

Vertical Curve

ptVCCircle
ptVCParabola

Path Sequence

ptP1P2
ptP2P1

Path Element Type

ptLineType
ptArcType
ptClothoidType

Road Cross Section Element Type

ptCenterLineAttr
ptSplitPointAttr
ptReferencePointAttr
ptSlopeInterceptAttr

Road Horizontal Alignment Transition Type

ptHorIP_Discontinuous
ptHorIP_Arc
ptHorIP_SpiralArcSpiral
ptHorIP_Spiral

Road Superelevation Point Type

ptNCIn
ptHCIn
ptFSEIn
ptFSEOut
ptHCOut
ptNCOut

Annotation format Areas

ptDefaultAreaFormat
ptAcres
ptHaAreCa

Serial port settings : stop bits

ptStopBit1
ptStopbit1_5
ptStopbit2

Serial port settings : parity

ptNoParity
ptOddParity
ptEvenParity
ptMarkParity
ptSpaceParity

Serial port settings : handshaking

ptXonXoff
ptHardware
ptNoFlowControl

Serial port settings : read/write

ptCommReadAccess
ptCommWriteAccess
ptCommReadWriteAccess

User Menu

ptNilUserMenuIndex

Type of Toolbar Buttons

ptMacroIcon_1 .. ptMacroIcon_9 and ptMacroIcon_A .. ptMacroIcon_Z and
ptMacroIcon_Next and ptMacroIconPrevious.

Bitmaps - Private Tool

ptGPSPositionFound
ptGPSSearchingPosition
ptGPSPositionFound_BatLow
ptGPSSearchingPosition_BatLow

Cursors

ptCursorArrow
ptCursorPoint
ptCursorLine
ptCursorParallel
ptCursorPerpendicular
ptCursorText
ptCursorPolygon
ptCursorCoordSystem
ptCursorMeasure
ptCursorWait
ptCursorMagnifyingGlass
ptCursorParallelSegment
ptCursorCenter
ptCursorMove1
ptCursorMove2
ptCursorRotate
ptCursorJoin
ptCursorToObject
ptCursorCamera
ptCursorTarget
ptCursorPath
ptCursorIP
ptCursorArrowDown
ptCursorMarker
ptCursorGrabber
ptCursorIntersect1
ptCursorIntersect2
ptCursorP1
ptCursorP2
ptCursorCut
ptCursorCutTo1
ptCursorCutTo2
ptCursorDivide
ptCursorDivideTo
ptCursorNone
ptCursorExtend1
ptCursorExtend2
ptCursorEraser
ptCursorLasso
ptCursorTextMargin
ptCursorAlign
ptCursorGrabHorizontal
ptCursorGrabVertical
ptCursorGrabCorner1
ptCursorGrabCorner2
ptCursorDrag
ptCursorAdd
ptCursorDelete
ptCursorCompound

End Of Cross Section

ptEndOfXSectionPart

Elevation/Value unknown

ptNilElevation
ptNilValue

Import DWG parameters

ptAsPoints
ptAsSymbols
ptUnpacked

ptNoImport

Export Shape parameters

ptShape2D
ptShape2DM
ptShape3D

Program options

ptPO_RW	ReadWrite
ptPO_Lite	Lite
ptPO_DTM	DTM
ptPO_DC	Datacollector
ptPO_3D	3D window
ptPO_RD	Road Design
ptPO_DB	GIS
ptPO_I	Images
ptPO_SB	Streckenberechnung

Macro options

ptMO_CM	Cadmap
ptMO_GRB	GRB

Pythagoras specific types

ptDateOnly
ptTime
ptXY
ptCadObjectType

Type of Coordinate Reference System

ptProjection
ptGeographical

Type of Interpolations

ptNoInterpolation
ptInTrianglePlane
ptNNDistance
ptNNArea
ptAllDistances

Synchronize Dictionary Data Options

Description

Values of this enumeration are used in methods `Application.OpenDocument` and `Subdocuments.AddSubDocument`. They define what to do when the main document and the subdocument to open/add contain a database dictionary table with the same name and structure, but with different datastamps.

See also Pythagoras Manual, Chapter “Database”, section “Tables”, subsection “Tables and subdocuments” for description of conception of datastamp.

ptDataOfOverview	: Ignore subdocument data, leave data as it is in the main overview document
ptDataOfSubdocument	: Delete data and replace them by data from subdocument
ptDataMostRecent	: Prefer newer data (when the last change was made later)
ptDataRaiseError	: Raise an error
ptDataDontOpenSubdocument:	Don't open subdocument in case of conflict

VectorisationType

ptVectoriseSection
ptVectorise3DEdges
ptVectorise3Dlines

PtPairing

ptSingle	: Search single objects
ptOverlappingPoints	: Search points being very close to other points
ptOverlappingLines	: Search lines (straight lines, arcs, circles or clothoids) having common parts with other lines
ptNearToSelected	: Search objects being not farther from some object belonging to the current set, than the specified distance
ptFarFromSelected	: Search objects being farther from any object belonging to the current set, than the specified distance

PtOverlappingPointSelection

ptOPAll	: Search all overlapping points
ptOPMinZ	: Search only points having elevation less than points overlapping with them
ptOPMaxZ	: Search only points having elevation greater than points overlapping with them

Decimal Places

ptDPControlPanel
ptDPDrawing
ptDPExport

Charsets

This enumeration is described completely according to an IANA standard (see <http://www.iana.org/assignments/ianacharset-mib>).

other
unknown
csASCII
csISOLatin1
csISOLatin2
csISOLatin3
csISOLatin4
csISOLatinCyrillic
csISOLatinArabic
csISOLatinGreek
csISOLatinHebrew
csISOLatin5
csISOLatin6
csISOTextComm
csHalfWidthKatakana
csJISEncoding
csShiftJIS
csEUCPkdFmtJapanese
csEUCFixWidJapanese
csISO4UnitedKingdom
csISO11SwedishForNames
csISO15Italian
csISO17Spanish
csISO21German
csISO60DanishNorwegian
csISO69French
csISO10646UTF1
csISO646basic1983
csINVARIANT
csISO2IntlRefVersion
csNATSSEFI
csNATSSEFIADD
csNATSDANO
csNATSDANOADD
csISO10Swedish
csKSC56011987
csISO2022KR
csEUCKR

csISO2022JP
csISO2022JP2
csISO13JISC6220jp
csISO14JISC6220ro
csISO16Portuguese
csISO18Greek7Old
csISO19LatinGreek
csISO25French
csISO27LatinGreek1
csISO5427Cyrillic
csISO42JISC62261978
csISO47BSViewdata
csISO49INIS
csISO50INIS8
csISO51INISCyrillic
csISO54271981
csISO5428Greek
csISO57GB1988
csISO58GB231280
csISO61Norwegian2
csISO70VideotexSuppl
csISO84Portuguese2
csISO85Spanish2
csISO86Hungarian
csISO87JISX0208
csISO88Greek7
csISO89ASMO449
csISO90
csISO91JISC62291984a
csISO92JISC62991984b
csISO93JIS62291984badd
csISO94JIS62291984hand
csISO95JIS62291984handadd
csISO96JISC62291984kana
csISO2033
csISO99NAPLPS
csISO102T617bit
csISO103T618bit
csISO111ECMACyrillic
csa71
csa72
csISO123CSAZ24341985gr
csISO88596E
csISO88596I
csISO128T101G2
csISO88598E
csISO88598I
csISO139CSN369103
csISO141JUSIB1002
csISO143IECP271
csISO146Serbian
csISO147Macedonian
csISO150
csISO151Cuba
csISO6937Add
csISO153GOST1976874
csISO8859Supp
csISO10367Box
csISO158Lap
csISO159JISX02121990
csISO646Danish
csUSDK
csDKUS
csKSC5636
csUnicode11UTF7
csISO2022CN
csISO2022CNEXT
csUTF8
csISO885913
csISO885914
csISO885915

csISO885916
csGBK
csGB18030
csOSDEBCDICDF0415
csOSDEBCDICDF03IRV
csOSDEBCDICDF041
csISO115481
csKZ1048
csUnicode
csUCS4
csUnicodeASCII
csUnicodeLatin1
csUnicodeIBM1261
csUnicodeIBM1268
csUnicodeIBM1276
csUnicodeIBM1264
csUnicodeIBM1265
csUnicode11
csSCSU
csUTF7
csUTF16BE
csUTF16LE
csUTF16
csCESU8
csUTF32
csUTF32BE
csUTF32LE
csBOCU1
csWindows30Latin1
csWindows31Latin1
csWindows31Latin2
csWindows31Latin5
csHPRoman8
csAdobeStandardEncoding
csVenturaUS
csVenturaInternational
csDECMCS
csPC850Multilingual
csPCp852
csPC8CodePage437
csPC8DanishNorwegian
csPC862LatinHebrew
csPC8Turkish
csIBMSymbols
csIBMThai
csHPLegal
csHPPiFont
csHPMath8
csHPPSMath
csHPDesktop
csVenturaMath
csMicrosoftPublishing
csWindows31J
csGB2312
csBig5
csMacintosh
csIBM037
csIBM038
csIBM273
csIBM274
csIBM275
csIBM277
csIBM278
csIBM280
csIBM281
csIBM284
csIBM285
csIBM290
csIBM297
csIBM420
csIBM423

csIBM424
csIBM500
csIBM851
csIBM855
csIBM857
csIBM860
csIBM861
csIBM863
csIBM864
csIBM865
csIBM868
csIBM869
csIBM870
csIBM871
csIBM880
csIBM891
csIBM903
csIBBM904
csIBM905
csIBM918
csIBM1026
csIBMEBCDICATDE
csEBCDICATDEA
csEBCDICCFAFR
csEBCDICDKNO
csEBCDICDKNOA
csEBCDICFISE
csEBCDICFISEA
csEBCDICFR
csEBCDICIT
csEBCDICPT
csEBCDICES
csEBCDICESA
csEBCDICESSE
csEBCDICUK
csEBCDICUS
csUnknown8BiT
csMnemonic
csMnem
csVISCII
csVIQR
csKOI8R
csHZGB2312
csIBM866
csPC775Baltic
csKOI8U
csIBM00858
csIBM00924
csIBM01140
csIBM01141
csIBM01142
csIBM01143
csIBM01144
csIBM01145
csIBM01146
csIBM01147
csIBM01148
csIBM01149
csBig5HKSCS
csIBM1047
csPTCP154
csAmiga1251
csKOI7switched
csBRF
csTSCII
csCP51932
cswindows874
cswindows1250
cswindows1251
cswindows1252
cswindows1253

cswindows1254
cswindows1255
cswindows1256
cswindows1257
cswindows1258
csTIS620

XMLDocumentVersion

xmlV1_0
xmlV1_1

XMLItemTypes

xmltText
xmltPI
xmltComment
xmltElement

XMLWhiteSpaceFacet

xmlPreserve
xmlReplace
xmlCollapse

XMLCompletionCode

xmlcSuccess	0
xmlcOpenError	1 Error opening text file
xmlcReadError	2 Error reading text file
xmlcNotXMLFile	3 Starting bytes of file are illegal for an XML file
xmlcNoFileInWeb	4 Error trying to download a file from the Web
xmlcEncodingNotSupported	5 Specified encoding is not supported on current OS
xmlcCreateError	6 Error creating text file
xmlcExternalEntity	7 Document contains external entity reference reading local file
xmlcExternalDTD	8 Document contains external DTD reference reading local file
xmlcParEntityNotDefined	9 Referenced parameter entity not defined
xmlcIllegalChar	10 Illegal character in text file
xmlcIllegalPubidChar	11 Public identifier contains prohibited character
xmlcIllegalCharData	12 "]]>" inside character data
xmlcIllegalAttrValue	13 ">" or "&" not beginning character or entity reference met
xmlcIllegalEntityValue	14 "%" not beginning parameter reference or "&" not beginning character reference met
xmlcUnlimitedString	15 Unpaired quotes found
xmlcUnlimitedCDATA	16 EOF during scanning CDATA block
xmlcInvalidComment	17 Comment contains
xmlcUnlimitedComment	18 EOF during scanning comment
xmlcIllegalCharRef	19
xmlcSemicolonExpected	20 Semicolon missing after character or entity reference
xmlcIllegalRefChar	21 Character reference defines illegal character
xmlcIllegalPI	22 <? is not followed by name
xmlcIllegalPIName	23 <? is followed by XML inside file
xmlcUnlimitedPI	24 EOF during scanning PI
xmlcEntityNotDefined	25 No entity has name defined in the entity reference
xmlcSpaceExpected	26 White spaces missing
xmlcEqualsExpected	27 Equals sign missing
xmlcStringExpected	28 Quoted string missing

xmlcAttrRefExtEntity	29 Attribute Value contains reference to external entity
xmlcIntEntityRefParameter	Entity Value in DTD contains reference to parameter entity
xmlcAttributeRepeated	30 Attribute name is repeated
xmlcEndTagExpected	31 End tag must begin here
xmlcTagNamesDiffer	32 Names in start and end tags are different
xmlcIllegalVersion	33 XML version is neither 1.0 nor 1.1
xmlcNoVersion	34 XML header in main file does not contain version designation
xmlcIllegalEncoding	35 Encoding name is not recognized
xmlcNoEncoding	36 XML header in not main file does not contain encoding designation
xmlcIllegalStandalone	37 Standalone parameter value is neither "yes" nor "no"
xmlcGreaterExpected	38 > missing
xmlcIllegalEntityContent	39 Referenced entity content is out of syntax
xmlcCircularReference	40 Entity circular reference
xmlcUnparsedEntity	41 Unparsed entity referenced as &name
xmlcUnlimitedStartTag	42 EOF during scanning start element tag
xmlcNameExpected	43
xmlcNmTokenExpected	44 Name token missing
xmlcLeftExpected	45 (missing
xmlcRightExpected	46) missing
xmlcExternalIDExpected	47 External ID missing
xmlcAttrTypeExpected	48 Attribute type definition should start here
xmlcConditionalInternal	49 Conditional section should not be in the internal DTD
xmlcRightBracketExpected	50] missing
xmlcNoMainElement	51 Main document element missing
xmlcExtraContent	52 Extra content at the end of file
xmlcInvalidDocument	53 Invalid Document
xmlcFragmentInURI	54 External ID contains fragment part
xmlcBadExternalDTD	55 Out of syntax in external DTD
xmlcLeftBracketExpected	56 [missing
xmlcBadInclude	57 Out of syntax in INCLUDE conditional section
xmlcUnlimitedIgnore	58 IGNORE section did not end before end of file
xmlcBadConditional	59 Neither INCLUDE nor IGNORE was found after <![
xmlcBadContent	60 Out of syntax in external element content
xmlcUnlimitedElementDecl	61 ">" pairing "<!ELEMENT" not found

Pythagoras virtual key codes Constants

Windows

VK_LBUTTON	= 001h
VK_RBUTTON	= 002h
VK_CANCEL	= 003h
VK_MBUTTON	= 004h
VK_BACK	= 008h
VK_TAB	= 009h
VK_CLEAR	= 00Ch
VK_RETURN	= 00Dh
VK_SHIFT	= 010h
VK_CONTROL	= 011h
VK_MENU	= 012h

VK_PAUSE	= 013h
VK_CAPITAL	= 014h
VK_KANA	= 015h
VK_HANGEUL	= 015h
VK_HANGUL	= 015h
VK_JUNJA	= 017h
VK_FINAL	= 018h
VK_HANJA	= 019h
VK_KANJI	= 019h
VK_ESCAPE	= 01Bh
VK_CONVERT	= 01Ch
VK_NONCONVERT	= 01Dh
VK_ACCEPT	= 01Eh
VK_MODECHANGE	= 01Fh
VK_SPACE	= 020h
VK_PRIOR	= 021h
VK_NEXT	= 022h
VK_END	= 023h
VK_HOME	= 024h
VK_LEFT	= 025h
VK_UP	= 026h
VK_RIGHT	= 027h
VK_DOWN	= 028h
VK_SELECT	= 029h
VK_PRINT	= 02Ah
VK_EXECUTE	= 02Bh
VK_SNAPSHOT	= 02Ch
VK_INSERT	= 02Dh
VK_DELETE	= 02Eh
VK_HELP	= 02Fh
VK_KEY_0	= 030h
VK_KEY_1	= 031h
VK_KEY_2	= 032h
VK_KEY_3	= 033h
VK_KEY_4	= 034h
VK_KEY_5	= 035h
VK_KEY_6	= 036h
VK_KEY_7	= 037h
VK_KEY_8	= 038h
VK_KEY_9	= 039h
VK_LWIN	= 05Bh
VK_RWIN	= 05Ch
VK_APPS	= 05Dh
VK_NUMPAD0	= 060h
VK_NUMPAD1	= 061h
VK_NUMPAD2	= 062h
VK_NUMPAD3	= 063h
VK_NUMPAD4	= 064h
VK_NUMPAD5	= 065h
VK_NUMPAD6	= 066h
VK_NUMPAD7	= 067h
VK_NUMPAD8	= 068h
VK_NUMPAD9	= 069h
VK_MULTIPLY	= 06Ah
VK_ADD	= 06Bh
VK_SEPARATOR	= 06Ch
VK_SUBTRACT	= 06Dh
VK_DECIMAL	= 06Eh
VK_DIVIDE	= 06Fh
VK_F1	= 070h
VK_F2	= 071h
VK_F3	= 072h
VK_F4	= 073h
VK_F5	= 074h
VK_F6	= 075h
VK_F7	= 076h
VK_F8	= 077h
VK_F9	= 078h
VK_F10	= 079h
VK_F11	= 07Ah
VK_F12	= 07Bh

VK_F13	= 07Ch
VK_F14	= 07Dh
VK_F15	= 07Eh
VK_F16	= 07Fh
VK_F17	= 080h
VK_F18	= 081h
VK_F19	= 082h
VK_F20	= 083h
VK_F21	= 084h
VK_F22	= 085h
VK_F23	= 086h
VK_F24	= 087h
VK_NUMLOCK	= 090h
VK_SCROLL	= 091h
VK_LSHIFT	= 0A0h
VK_RSHIFT	= 0A1h
VK_LCONTROL	= 0A2h
VK_RCONTROL	= 0A3h
VK_LMENU	= 0A4h
VK_RMENU	= 0A5h
VK_PROCESSKEY	= 0E5h
VK_ATTN	= 0F6h
VK_CRSEL	= 0F7h
VK_EXSEL	= 0F8h
VK_EREOF	= 0F9h
VK_PLAY	= 0FAh
VK_ZOOM	= 0FBh
VK_NONAME	= 0FCh
VK_PA1	= 0FDh
VK_OEM_CLEAR	= 0FEh

Chapter 141 - Error handling

Introduction

When a runtime error occurs, information about the error can be obtained from the standard VBA `Err` Object. Only the properties `Number` and `Description` are available.

Pythagoras VBA errors

The following runtime errors may be raised by standard VBA statements :

3	Return without GoSub
5	Invalid procedure call or argument
6	Overflow
7	Out of memory
9	Subscript out of range
10	This array is fixed or temporarily locked
11	Division by zero
13	Type mismatch
14	Out of string space
16	Expression too complex
17	Can't perform requested operation
18	User interrupt occurred
20	Resume without error
28	Out of stack space
35	Sub, Function, or Property not defined
47	Too many DLL application clients
48	Error in loading DLL
49	Bad DLL calling convention
51	Internal error
52	Bad file name or number
53	File not found
54	Bad file mode
55	File already open
57	Device I/O error
58	File already exists
59	Bad record length
61	Disk full
62	Input past end of file
63	Bad record number
67	Too many files
68	Device unavailable
70	Permission denied
71	Disk not ready
74	Can't rename with different drive
75	Path/File access error
76	Path not found
91	Object variable or With block variable not set
92	For loop not initialized
93	Invalid pattern string
94	Invalid use of Null
97	Can't call Friend procedure on an object that is not an instance of the defining class
298	System DLL could not be loaded
320	Can't use character device names in specified file names
321	Invalid file format
322	Can't create necessary temporary file
325	Invalid format in resource file
327	Data value named not found
328	Illegal parameter, can't write arrays
335	Could not access system registry
336	ActiveX component not correctly registered
337	ActiveX component not found

338	ActiveX component did not run correctly
360	Object already loaded
361	Can't load or unload this object
363	ActiveX control specified not found
364	Object was unloaded
365	Unable to unload within this context
368	The specified file is out of date. This program requires a later version
371	The specified object can't be used as an owner form for Show
380	Invalid property value
381	Invalid property-array index
382	Property Set can't be executed at run time
383	Property Set can't be used with a read-only property
385	Need property-array index
387	Property Set not permitted
393	Property Get can't be executed at run time
394	Property Get can't be executed on write-only property
400	Form already displayed, can't show modally
402	Code must close topmost modal form first
419	Permission to use object denied
422	Property not found
423	Property or method not found
424	Object required
425	Invalid object use
429	ActiveX component can't create object or return reference to this object
430	Class doesn't support Automation
432	File name or class name not found during Automation operation
438	Object doesn't support this property or method
440	Automation error
442	Connection to type library or object library for remote process has been lost
443	Automation object doesn't have a default value
445	Object doesn't support this action
446	Object doesn't support named arguments
447	Object doesn't support current locale setting
448	Named argument not found
449	Argument not optional or invalid property assignment
450	Wrong number of arguments or invalid property assignment
451	Object not a collection
452	Invalid ordinal
453	Specified DLL function not found
454	Code resource not found
455	Code resource lock error
457	This key is already associated with an element of this collection
458	Variable uses a type not supported in Basic
459	This component doesn't support events
460	Invalid Clipboard format
461	Specified format doesn't match format of data
480	Can't create AutoRedraw image
481	Invalid picture
482	Printer error
483	Printer driver does not support specified property
484	Problem getting printer information from the system. Make sure the printer is set up correctly
485	Invalid picture type
486	Can't print form image to this type of printer
520	Can't empty Clipboard
521	Can't open Clipboard
600	Modal form must be created and shown/hidden in the same form/module
601	Non-modal form cannot be closed inside its own property or method
735	Can't save file to TEMP directory
744	Search text not found
746	Replacements too long
31001	Out of memory
31004	No object

31018	Class is not set
31027	Unable to activate object
31032	Unable to create embedded object
31036	Error saving to file
31037	Error loading from file

Pythagoras errors

The following runtime errors may be raised by accessing Pythagoras properties and methods :

1000	Unknown Error
1001	No document
1002	Invalid or not existing document
1003	No object
1004	Invalid or not existing object
1005	Operation impossible on protected object
1006	This object does not have comment
1007	Too many layers
1008	Invalid value for a group id
1009	Object has invalid style
1010	Style can not be added to the document or is unknown
1011	Angle is not in the first positive or negative circle
1012	Not enough memory to do this operation
1013	Endpoints must be point objects
1014	Endpoints contain the same point
1015	Associated objects must be in the same overlay
1016	File has not been saved !!!
1017	Invalid or missing parameters
1018	File has not been saved !!!
1019	Not enough resources available to load drawing in memory.
1020	Protection dongle present ?
1021	Can't open new drawing.
1022	Not enough resources available to load drawing in memory
1023	Save As necessary
1024	This function is NOT supported in the read-only version
1025	File is not saved !
1026	Creating the file failed
1027	Invalid filename
1028	This document can not be saved in the selected file format
1029	Error during close of document
1030	File is already opened
1031	Document not closed correctly
1032	This file is not a Pythagoras file
1033	Invalid version. This file can not be edited with the current version of Pythagoras.
1034	Inconsistency in drawing.
1035	Impossible value for a vertical angle
1036	Selections in both overlays simultaneous not allowed
1037	Protected Layers can not be modified
1038	At least one object has a invalid display level (<-10 or > 10).
1039	Several objects could not be deleted, they are still used by a polygon or path. Delete the polygon or path first.
1040	Several objects could not be deleted, they are still used by other objects. Delete the other objects (Polygons, Paths, Coordinate Systems, ...) first.
1041	A point could not be deleted. It is used by a Coordinate System. Delete the Coordinate System if you want to delete the point(s).
1042	Not all points could be deleted. They are used by a polygon or path. Delete the polygon or path first if you want to delete the points.
1043	An arc could not be deleted. The arc is used by a polygon or path. Delete the polygon or path if you want to delete the arc(s).
1044	A circle could not be deleted. The circle is used by a polygon or path. Delete the polygon or path if you want to delete the circle(s).
1045	Can not add a new symbol.

1046	Not enough resources available to load drawing in memory.
1047	Paste from Local CS to Page CS is not possible
1048	Default layer is protected
1049	Invalid type of Coordinate system.
1050	Unknown Error
1051	Unknown Error
1052	Unknown Error
1053	Not enough free layers to paste or import the data.\n\nGroup layers.
1054	Unknown Error
1055	No arc can be constructed: C-P1 <> C-P2
1056	The parameters do not form a valid Clothoid
1057	Too many points for one curve
1058	Too many objects for one polygon or path
1059	These objects do not form a valid polygon or path
1060	Invalid style
1063	Coordinate system in wrong overlay
1064	** Unknown Coordinate system
1065	Nr of user coordinate systems is limited to 32. Can not allocate a new CS
1066	These points can not form a coordinate system
1067	Invalid scale factor for layer
1068	Can not allocate new layer anymore.
1069	Active group must contain the default layer
1070	Can not allocate a new group anymore.
1071	Can not delete the active group.
1072	Invalid string for symbol name
1073	Invalid string for symbol group
1074	Symbol is not in library
1075	Symbol can not be removed
1076	Layer name already exists
1077	Drawing is not on top.
1078	Method may only be used in Mouse Event Macro
1079	Method may only be used in Keyboard Event Macro
1080	Calculation of the Cross section failed
1081	Array overflow
1082	Too many points to calculate the cross section
1083	Invalid text file.
1084	Save As necessary
1085	This file is too large to be opened by the Pythagoras Editor.
1086	Serial communication : access denied
1087	CadObjects refer to different documents
1088	Coordinate System in wrong document
1089	Object is not editable
1090	This document can not be added as as subdocument. Too many layers.
1091	ODBC Error
1092	Invalid "Data collector" file
1093	Sheet with given name already exists
1094	The name of the main document may not be the same as the name of one of its subdocuments.
1095	View with given name already exist.
1096	Zoomfactor: invalid value. (0.0 <= f <= 1.0).
1097	Can't import or export in Page Coordinate System
1098	Associated objects must be in page CS
1099	Associated objects must be in the same sheet
1100	Special paste operations can not be combined with transformations between projection systems
1101	Databases from main document and a subdocument cannot be merged
1102	Intertable references are dropped
1103	Unknown Coordinate Reference System
1104	Tie Point Set not found
1105	Drawing can not be saved: a thematic map can not be converted
1106	Inconsistent image file
1107	The parameters do not form a valid circle
1108	Geographical coordinate systems can not be assigned to documents

1109	Changing the Coordinate Reference System is only possible if the drawing does not contain Georeferenced Web services
1110	Can not add the new User Default. User Default with same name already exists
1111	Warning : A name conflict between Web Services Definitions of the document and the library is detected. The Web Services of the document are automatically renamed.
1112	Deleting a road still used in Road Design window is impossible.
1113	This operation is impossible on a Road that is opened in Road Design window.
1114	Invalid Name. A terrain model with the same name already exists.
1115	Invalid operation on Layer of the Road that is opened in Road Design window.
1116	Invalid operation on subdocument of the Road that is opened in Road Design window.
1117	Image is too big to be embedded.
1118	Image is too big to be imported.
1119	Can not add the new User Default. Maximum of User Defaults is reached.
1121	Could not save document file to target folder. Check it is accessible and have enough free space.
1122	Could not open document file or read with losses.
1124	Resulting polyline is invalid.

1401 .. 1410 : File errors

1401	Illegal filename
1402	File with given name not found
1403	File already exists
1404	Hardware error
1405	No more room on device
1406	File is locked
1407	Operating upon unopened file
1408	Read attempted after EOF/EOL
1409	Position outside file
1410	unknown error

1500 .. : Other errors

1500	Error found during Validation of drawing. Automatic repair failed.
1501	Invalid Parameter
1502	Color Property not applicable for this object type
1503	Wrong object type or object is not allocated
1504	Conversion of string to REAL value failed
1505	String is too short
1506	Invalid array parameter passed
1507	String is too long
1508	String is missing
1509	String contains invalid characters
1510	Not enough memory for this operation
1511	Comment Property not applicable for this object type
1512	This combination of attributes is impossible
1513	Invalid index for submenu item
1514	Invalid index for menu item
1515	The maximum number of user menu items is exceeded
1516	The maximum number of user submenus is exceeded
1517	The maximum number of submenu items is exceeded
1518	Submenu items can only be added to popup menu items
1519	New timer can not be started
1520	Connection to ODBC data source failed
1521	Database option is not supported
1522	An actual variant argument has improper type
1523	An actual variant argument belongs to an improper class
1524	<i>Obsolete</i>
1525	A database table is dropped
1526	A database table column is dropped
1527	Too large value for DBColumn.PendingAction property specified
1528	A table row(s), not a query row(s) is required

1529	A row is deleted from the table
1530	A query is dropped
1531	A CAD object is linked to other table
1532	Wrong table type specified
1533	Objects belong to different overviews
1534	A row belongs to other table
1535	A column belongs to other table
1536	ObjectTypes value is wrong
1537	CAD Object belongs to other document than database row
1538	Invalid Document
1539	Type of constant to compare with differs from column type
1540	Comparison operator contradicts column type
1541	Document containing executing macro can not be closed
1542	A non-VBA expression has bad syntax or result type
1543	Macro not found
1544	An object cannot be included to a compound because it has improper type
1545	An object cannot be included to a compound because it is linked to the database
1546	Specified number is not a compound identifier
1547	Objects belong to different documents
1548	Objects belong to different overlays
1549	No permission to use VBA function
1550	DB table column index was dropped, position is lost
1551	Coordinate Reference Systems are not correlated
1552	Invalid Overlay
1553	The parameters of the method refer to different documents
1554	Insufficient number of points or all points are collinear
1555	There are 2 or more points with the same coordinates
1556	Triangulation can not be calculated
1557	Objects belong to different sheets
1558	Operation not possible on object that is not horizontal
1559	Can't import or export in Page Coordinate System
1560	Export shape failed
1561	Invalid KML Element type
1562	This operation can be applied only to a queue of items of an element
1563	Current item of queue is undefined
1564	Input matrix has wrong dimensions for this type of operation
1565	Pythagoras database is already closed
1566	An erratic definition of a base rowset
1567	A query is CAD object linked
1568	A query is not CAD object linked
1569	This operation can be applied only to a document database
1570	This operation can be applied only to a Pythagoras database
1571	A query row(s), not a table row(s) is required here
1572	A row is not more present in a query
1573	This operation can be applied only to a simple or an extension query
1574	This operation can be applied only to an aggregation query
1575	Specified object type set does not correspond to any system table
1576	Some data in the array have different type. Sorting is not possible
1577	The center of the arc can not be calculated

2001 .. : Database inconsistencies

2001	Type not supported
2002	Bad table, view or column name
2003	CAD object type specified is not allowed in databases
2004	No source object specified to do operation
2005	Table structure is protected
2006	Dictionary table cannot reference CAD objects
2007	A document is invisible or outside of an overview
2008	A document is read only
2009	A document is not main overview one
2010	An object belongs to protected layer

2011	Value type does not match column type
2012	Maximum < minimum in column definition
2013	Default < minimum in column definition
2014	Default > maximum in column definition
2015	Referencing and referenced columns have different types
2016	Referenced table is not a dictionary
2017	Table reference points outside the database
2018	Minimum or Maximum specified for a column whose type does not permit this
2019	A table or query does not contain a column with such number
2020	A table or query does not contain a column with such name
2021	A column is not changeable
2022	Pending reference action cannot be set to NULL because it contradicts other column properties
2023	NULL value is prohibited for a column
2024	Value to assign < Minimum
2025	Value to assign > Maximum
2026	Reference to a CAD object that does not exist
2027	Reference to a row that does not exist
2028	Column number repeated in the value list
2029	Insert aborted because of not all mandatory values are specified
2030	A set of allowed CAD object types must be empty for a non-attribute table
2031	A set of allowed CAD object types must be non-empty for an attribute table
2032	CAD object cannot be linked to a table because of its type is not allowed for the table
2033	CAD object already linked to the database
2034	A column cannot be added or renamed because there is already a column with such name in the table or in the query
2035	Maximum number of queries in the database exceeded
2036	Maximum number of column in the table or in the query exceeded
2037	A document does not contain a table with such number
2038	A document does not contain a table with such name
2039	A row value cannot be converted to new type
2040	Minimum cannot be changed because of column data
2041	Maximum cannot be changed because of column data
2042	A table or a view cannot be added or renamed because there is already a view with such name
2043	Default value does not refer to existing row
2044	A column cannot be added because the table is not empty and the column has no default
2045	A non unique value column cannot be referenced
2046	Column cannot be deleted because it is referenced
2047	Maximum number of tables in the database exceeded
2048	Table type cannot be changed
2049	A table cannot be added or renamed because there is already a table with such name in the database
2050	Set of allowed CAD object types cannot be changed because a table would be linked with CAD objects of not allowed types
2052	Table cannot be deleted because it is referenced
2053	A table cannot be copied to another document because of undefined table reference
2054	Referencing column not found
2055	A document does not contain a sheet with such name
2056	A database cannot be copied to itself
2057	An operation cannot be applied to an attribute table
2058	An operation cannot be applied to a dictionary table
2059	A query being a base of another query is deleted
2060	A view cannot be based on a temporary query
2061	Database is locked
2062	Operation cannot be executed because dongle database option is off
2063	Attempt to do index operation by column not indexed
2064	Data cannot be copied because tables have different structure
2065	Table couldn't be added because table with such name exists and has different structure
2066	Tables couldn't be added to Clipboard because they have wrong references
2067	A column of not CAD object type defined with some CAD object type allowed
2068	A column of CAD object type defined with no CAD object type allowed
2069	CAD object cannot be column minimum, maximum or default value
2070	A unique value column cannot contain repeating values

2071	Such value is already present in a unique value column
2072	A table cannot be created without name
2073	A column cannot be created without name
2074	Table data are read protected
2075	Table data are write protected
2076	Invalid table access rights code
2077	A column of an attribute table cannot be declared a unique value one
2078	Type of referenced column cannot be changed
2079	Pending action for CAD object referencing column cannot be set to ptPendingLeave
2080	Subdocuments have different lock names or passwords, unlocking is impossible
2081	Lock name is absent, but lock password is present
2082	Lock name is present, but lock password is absent
2083	Wrong name or password
2084	Not NULL default in a unique value column
2085	Object referencing column cannot be neither indexed nor unique
2086	Real type column cannot be declared unique
2087	A temporary query is probably deleted
2088	A database does not contain a view with such name
2089	A view cannot have empty name
2090	A row iterator is discontinued (table, query or column is deleted or index is destroyed)
2091	Attempt to make reciprocal references between columns
2092	A temporary query cannot have name
2093	MayBeNull for CAD object referencing column cannot be set to FALSE
2094	Condition contains incompatible data
2095	Type mismatch in expression
2096	Bad code of built-in function in condition definition
2097	Attempt to do operation with invalid query
2098	Attempt to do operation with dropped table or query
2099	An attempt is made to link an object of the page overlay to the database
2100	An attempt is made to link an element of compound to the database
2101	The operation cannot be applied to a document database
2102	The operation can be applied only to a document database
2103	NULL values cannot be prohibited for a column because the column already contains some NULL values
2104	A table or a view cannot be added or renamed because this name is a name of a system table
2105	A table or a view cannot be added or renamed because this name is an English name of a system table

2500 .. : Coordinate Reference Systems

2512	Transformation of coordinate in Coordinate Reference System failed: outside domain of CRS
2513	 : input not in valid range for geographical coordinates
2514	 : DLL is missing
2515	 : Calculation failed

2600 .. : Stack Errors

2600	Stack not initialized
2601	Stack underflow
2602	Stack overflow

2700 .. : Queue Errors

2700	Queue not initialized
2701	Queue underflow
2702	Queue overflow

2800 .. : Tree Errors

2800	An entry in the tree with the same Value Index pair already exists

2801 Data with the given key is not found in the tree

2900 .. : Console Errors

2800 An entry in the tree with the same Value Index pair already exists

2801 Data with the given key is not found in the tree

3000..: JSON Errors

3001 Object or Array is assigned to JSON_Data structure that is not empty

3002 JSON Syntax Error. (followed by description of type of error)

3003 Error in JSON Parser

3004 Object or Array is recursively assigned

Chapter 142 - Pythagoras VBA language elements

Option Statement

Option statements must be used at the beginning of a module. The Option is only valid for the module containing this statement.

Base

Syntax

```
Option Base (0 | 1)
```

Description

By default arrays start with index 0. `Option Base` allows to start indexing from 1.

Example

```
Option Base 1
```

Explicit

Description

In Pythagoras VBA `Explicit` is always true.

Compare

Syntax

```
Option Compare (Binary | Text)
```

Description

Sets the default for comparing strings. By default the comparing method = Binary.

Example

```
Option Compare Text
```

MSCompatible

Syntax

```
Option MSCompatible
```

Description

This option is introduced in Pythagoras 11. Previous versions of Pythagoras VBA were incompatible in a number of cases with Microsoft VBA.

For Pythagoras 11.0x, it was recommended to insert this statement in the first line of every module.

Since 11.10 we recommend to remove the option `MSCompatible`.

In that case when using a variable in parantheses in a procedure call and the parameter is `ByRef`, you will get the message:

"Ambiguous usage of variable in pararenthesis in procedure call".

When you want to use, this "dangereous" feature of VBA, you have to use `Option MSCompatible`.

Example

```
Option MSCompatible
```

PlusMSCompatible

Syntax

```
Option PlusMSCompatible
```

Description

This option makes the language closer to the official definition of Microsoft. With this option off language is closer to what it was in Pythagoras v13. The option influences on the following aspects of the language.

1. `+` operator. When this statement is absent in a module then `+` operator applied to a string and a non-string value in any order is treated as string concatenation operator. This means that the non-string value is converted to a string and then the two strings are concatenated. When this statement is present in a module then `+` operator applied to a string and a non-string value in any order is treated as addition operator. The string is converted to a numeric and then the two numeric values are added.
2. Conversion of `Null` value to a string. With the option off string “Null” is obtained; with the option on a conversion the error is raised. This concerns explicit conversions like `CStr` function and many cases of implicit conversions such as assignments.
3. `IsNumeric` function, see its description.

We recommend to include the option into VBA modules.

Examples

```
Option PlusMSCompatible
...
MsgBox "3.14" + 2    ' "5.14" will be displayed

' Option PlusMSCompatible is not present
MsgBox "3.14" + 2    ' "3.142" will be displayed
```

Special values

The values `Nothing`, `Empty` and `Null` are predefined. All 3 values mean absence of meaningful value.

Nothing

Description

This value may be assigned to a variable of any `Object` type or of a `Variant` type. If a variable = `Nothing`, this means that the variable does not reference to an object.

Rules

The operators `=` and `<>` may not be used to compare object with `Nothing`. See example on how to check if variable = `Nothing`.

Example

```
Dim P As Point

Set P = Nothing
...
If P Is Nothing Then
End If
```

Empty

Description

A variable of type `Variant` is always assigned to `Empty` after declaration. If a variable = `Empty`, this means that no value has been assigned to the variable yet. An attempt to do any operation with a variable with value = `Empty` raises an exception.

Rules

The value `Empty` may only be assigned to a `Variant` type. To determine if a variable of type `Variant` is initialised, the function `IsEmpty` must be used.

Example

```
Dim V As Variant

V = Empty
...
If IsEmpty(V) Then
End If
```

Null

Description

Null value indicates that the variable of type `Variant` does not contain any valid data.
In Pythagoras Null value is mainly used when handling database data.

Rules

Null may only be assigned to a `Variant` type.

To determine if a value of a variable is valid, the function `IsNull` must be used.

Example

```
Dim V As Variant

V = Null

...
If IsNull(V) Then
End If
```

Operators

Arithmetic Operators

List

- `+` : addition
- `-` : subtraction
- `/` : division: returns a double (floating point)
- `*` : multiplication
- `\` : integer division: returns integer result
- `Mod` : modulo
- `^` : exponent

String Operators

List

- `&` : concatenation

Comparison Operators

List

- `>` : greater than
- `<` : less than
- `<>` : not equal
- `>=` : greater than or equal
- `<=` : less than or equal
- `=` : equal

Logical Operators

List

- `And` : logical AND
- `Or` : logical OR
- `Xor` : logical XOR
- `Not` : logical negation
- `Eqv` : logical equivalence
- `Imp` : logical implication

Variable Declaration

Const

Description

Declares a constant.

Hexadecimal, octal or binary values must be preceded respectively by &H, &O and &B. When defining a 64-bit constants with 31 bit 1 and higher bits not defined, higher bits will be filled by 1s (sign extension). To overrule this an extra left zero digit should be included, see examples below.

Example

```
Const pi = 3.1415926535897932
Const BinaryTen = &B1010
Const Mask1 = &H80000000
Const Mask2 = &H080000000
Dim X As LongLong
...
X = Mask1 ' Hexadecimal FFFFFFFF800000000 is assigned
X = Mask2 ' Hexadecimal 00000000800000000 is assigned
```

Dim

Syntax

```
[Public | Private] Dim vardeclaration *(, vardeclaration)
vardeclaration: varname [(subscripts)] [As type]
type is one of the following data types:
Boolean, Byte, Date, Double, Integer, Long, LongLong, LongPtr,
Single, String, Error, Variant, User-Defined type.
```

Description

Declares a local or global variable. Only global variables may have the keyword `Public` or `Private`.

Multi-dimensional arrays have a fixed dimension for each subscript. Dynamic arrays must be defined with nothing in parentheses but may be redefined as multi-dimensional by `ReDim` statement.

Type `LongPtr` is the same as `Long` in 32-bit Pythagoras VBA and the same as `LongLong` in 64-bit one.

Example

```
Dim Str As String, B As Boolean.
Dim A1D(10 To 20) As Integer           ' first index starts from 10.
Dim A3D(2,5,3) As Double               ' 3-dimensional array
Dim Names() As String                  ' Dynamic array
Dim SymbolName As String * 24          ' String of 24 bytes
Dim Value As Long                      ' 32-bit Integer
Dim LongValue As LongLong              ' 64-bit Integer
Dim V As Variant                       ' variant type
Dim D As Date                          ' Date type
Dim X, Y(), a As Double, b(4,5) As Double 'X = variant
                                         'Y() = array of variants

A1D(1) = 5
A3D(0,0,0) = 12.5
i = UBound(Names)                      ' Returns upper-bound of Names
Value = &HFF                           ' Set value = hex FF = 255.
                                         ' &O37 Octal = &B11111 binary.
```

User-Defined type

Syntax

```
Type varname
DimStatement
[DimStatements]
End Type
```

Description

Declares a user-defined data type.

Example

```
Type tContactPerson
```

```

        Name As String
        Index As Long
    End Type

    Dim Contacts() As tContactPerson      ' Dynamic array

```

Enumeration type

Syntax

```

Enum varname
    DimStatement
    [DimStatements]
End Enum

```

Description

Declares an enumeration data type.

Example

```

Enum Ecolors
    ecRed = 1
    ecGreen
    ecBlue
End Enum

Dim cv As Ecolors      ' may accept any value of Long

```

ReDim

Syntax

```

ReDim [Preserve] variable[(subscripts)]

```

Description

Resizes and reallocates storage space for a dynamic array. Must be used within a procedure (Sub or Function).

Example

```

Sub DyArray

    Dim Names() As String      ' Dynamic array

    ReDim Names(1 To 5)
    Names(1) = "My name"

    ReDim Preserve Names(1 To 10)      ' Content is preserved

End Sub

```

Erase

Syntax

```

Erase variable *( , variable)

```

Description

The variable should be of a dynamic array type. Deletes all elements of the arrays.

Example:

```

Dim A() As Integer

Erase A

```

Procedure & Function Declaration

Sub

Syntax

```

[Public | Private] Sub name[(arglist)]
    [statements]
[Exit Sub]

```

```

    [statements]
End Sub

```

arglist : a comma-delimited list of parameters each having the following syntax :
 [ByVal | ByRef] [Optional] *varname*[()] As Type [= *defaultvalue*]

Description

Declares a procedure.

- Public is the default

- ByRef is the default method of passing parameters.

When the Optional keyword is used, all the arguments that follow must also be optional.

Example

```

Private Sub Test(P1 As Long, ByVal P2 As Double, _
    Optional ByVal TestEqual As Boolean = True, _
    Optional ByVal Value As Double = 0.1)

End Sub

' Valid ways of calling Sub with optional parameters

Sub CallTest
    Test 5, 1.2
    Test 5, 1.2, False, 0.2
    Test 5, 1.2, Value:=0.3      ' Named parameter
    Test 5, 1.2, , Value:=0.3    ' Omitted parameter + named
parameter
End Sub

```

Function

Syntax

```

[Public | Private] Function name[(arglist)] As Type
    [statements]
    [name = expression]
    [Exit Function]
    [statements]
    [name = expression]
End Function

```

arglist : a comma-delimited list of parameters each having the following syntax :
 [ByVal | ByRef] *varname*[()] As Type [= *defaultvalue*]

Description

Declares a function.

- Public is the default

- ByRef is the default method of passing parameters.

Example

```

Private Function Max(ByVal P1 As Double, ByVal P2 As Double) As
Double
    If P1 >= P2 Then
        Max = P1
    Else
        Max = P2
    End If
End Function

```

Calling Procedures and Functions

Syntax

```

procedurename [arglist]

```

arglist : a comma-delimited list of parameters.

Description

Calls a procedure (Sub). When the procedure contains optional arguments *Named parameters* may be used.

When a parameter is between brackets (see last example), it is considered an expression.

Examples

```
'Example 1

Function Max(P1 As Double, P2 As Double, _
    Optional P3 As Double = ptMinDouble, _
    Optional P4 As Double = ptMinDouble) As Double
    If P3 = ptMinDouble Then
        Max = Max(P1, P2)
    ElseIf P4 = ptMinDouble Then
        Max = Max(Max(P1, P2), P3)
    Else
        Max = Max(Max(Max(P1, P2), P3), P4)
    End If
End Function

Sub TestMax
    Dim V As Double, V1 As Double, V2 As Double, V3 As Double, V4
    As Double

    V = Max(V1, V2)
    V = Max(V1, V2, V3)
    V = Max(V1, V2, V3, V4)
End Sub

'Example 2

Sub Test(ByRef V1 As Long, _
    Optional ByVal Min As Long = -100, _
    Optional ByVal Max As Long = 100)

End Sub

Sub CallUsingNamedArguments
    Test 10, Max := 99, Min := -10
End Sub

Sub CallUsingExpression
    Dim V As Long

    V = 123
    Test (V), 3.0      ' The variable V is not changed on return
End Sub
```

Program Structure and flow

Do ... Loop

Syntax

```
Do [While | Until condition]
    [statements]
[Exit Do]
    [statements]
Loop

or

Do
    [statements]
[Exit Do]
    [statements]
Loop [While | Until condition]
```

Description

Repeatedly executes *statements* while or until *condition* is *True*.

Examples

```
Do While a < b
    a = a + 1
Loop

Do
    If a >= b Then
        Exit Do
    Else
        a = a + 1
    End If
Loop
```

Exit

Syntax

```
Exit Do
Exit For
Exit Function
Exit Sub
```

Description

Exits a loop or procedure.

Examples

```
Do
    If a >= b Then
        Exit Do
    Else
        a = a + 1
    End If
Loop
```

For ... Next

Syntax

```
For counter = from To To [Step value]
    [statements]
[Exit For]
    [statements]
Next [counter]
```

Description

Executes *statements* a given number of times

Examples

```
Dim i As Long
Dim Names() As String

Redim Names(1 To 10)
For i = LBound(Names) to UBound(Names)
    GetName(i, Names(i))
Next i
```

If ... Then ... Else

Syntax

```
If condition Then
    [statements]
{Elseif condition Then
    [statements]}
[Else
    [statements]]
End If

or
```

```
If condition Then [statements] [Else [statements]]
```

Description

Executes statements depending on condition(s)

Examples

```
If a > b Then
    aGTb
Elseif a = b Then
    aEQb
Else
    aLTb
End If
If a > b Then aGTB Else aLTb    ' Single line form of If statement.
```

Select Case ... End Select

Syntax

```
Select Case expression
{Case ExpressionList
    [statements]}
End Select
```

Description

Conditional executes statements depending on a value or a range of values.

Examples

```
Select Case i + j
Case 1, 2, 3
    ' Treat cases 1,2 and 3.
Case 10, 11, 20 to 30, 50 to 64
    ' Treat cases 10, 11, 20 to 30, 50 to 64
Case 50 to 70
    ' Treat cases 50 to 70
Case else
    ' Treat all other cases
End Select
```

With ... End With

Syntax

```
With object
    [statements]
End With
```

Description

Performs a set of assignments and executes other code against a particular object or user-defined type.

Examples

```
Type tContactPerson
    Name As String
    Index As Long
End Type

Dim Contacts(1 To 10) As tContactPerson

With Contacts(1)
    .Name = "Fred Flintstone"
    .Index = 1
End With
```

Data Type Conversion Functions

CBool

Syntax

`CBool(expression)`

Description

Converts *expression* to a Boolean data type. Expressions that evaluate to 0 are converted to False (0), and expressions that evaluate to nonzero values are converted to True (-1)

Examples

```
Dim B As Boolean

B = CBool("0")           ' returns False
```

CByte

Syntax

`CByte(expression)`

Description

Converts *expression* to a Byte data type.

Examples

```
Dim B As Byte

B = CByte("20")           ' returns the value 20 in B
B = CByte(5.6)            ' returns the value 6 in B
```

CDBl

Syntax

`CDBl(expression)`

Description

Converts *expression* to a Double data type. *Expression* may also be of type Date.

Examples

```
Dim D As Double
Dim i As Integer

D = CDBl("20.5")          ' returns the value 20.5 in B
D = CDBl(i)               ' Converts integer value i to Double
```

CInt

Syntax

`CInt(expression)`

Description

Converts *expression* to a Integer data type.

Examples

```
Dim i As Integer, D As Double

i = CInt("20.4")          ' returns the value 20 in i
i = CInt(-15.2)           ' returns -15.
```

CLng

Syntax

`CLng(expression)`

Description

Converts *expression* to a Long data type.

Examples

```
Dim L As Long
```

```
L = CLng("20.5")      ' returns the value 20.5 in L
L = CLng(-15.2)       ' returns -15.
```

CLngLng

Syntax

```
CLngLng(expression)
```

Description

Convert *expression* to a LongLong data type.

Example:

```
Dim L As LongLong

L = CLngLng(1.2E15)
```

CLngPtr

Syntax

```
CLngPtr(expression)
```

Description

Convert *expression* to a LongPtr data type. This means that it is equivalent to CLng in 32-bit version and to CLngLng in 64-bit version.

CSng

Syntax

```
CSng(expression)
```

Description

Converts *expression* to a Single data type.

Examples

```
Dim S As Single

S = CSng("20.5")      ' returns the value 20.5 in S
```

CStr

Syntax

```
CStr(expression)
```

Description

Converts *expression* to a String data type. When *expression* contains a date, the resulting string will contain the date always in the format "yyyy-MM-dd".

Examples

```
Dim S As String

S = CStr(-123.56)      ' returns the string "-123.56" in S
```

CDate

Syntax

```
CDate(expression)
```

Description

Converts *expression* to a Date data type. The *expression* must be of type string. The string must have the format "yyyy-MM-dd". Unlike MS VBA, the Pythagoras CDate format is fixed and does not depend on the locale settings of the computer.

Examples

```
Dim D As Date
Dim S As String

S = "2005-08-10"
D = CDate(S)
```


CVErr

Syntax

`CVErr (errornumber)`

Description

Returns a variant type containing a string. The string = "Error <errornumber>".

Examples

```
Dim V As Variant
Dim S As String

V = CVerErr(123)
S = V
```

Fix

Syntax

`Fix (expression)`

Description

Removes the fractional part of a number.

Examples

```
Dim D As Double

D = Fix(-123.56)      ' returns -123 in D
```

Int

Syntax

`Int (expression)`

Description

Removes the fractional part *expression* if the value is positive. For negative values, Int returns the first negative integer less than or equal to *expression*.

Examples

```
Dim D As Double

D = Int(-123.56)      ' returns -124 in D
```

Round

Syntax

`Round (Expression, NumberOfDecimalPlaces)`
`Round (Expression)`

Description

Returns the rounded value of *Expression*.

Examples

```
Dim D As Double
D = Round("12345.67890", 2)    ' Returns 12345.68
```

Str

Syntax

`Str (expression)`

Description

Converts *expression* to a String data type.

Examples

```
Dim S As String

S = Str(-123.56)      ' returns the string "-123.56" in S
```

Val

Syntax

```
Val (expression)
```

Description

Converts a string representation of a number into a numeric data type. (Integer, Long, Single, Double)

Examples

```
Dim D As Double
```

```
D = Val("-123.56")      ' returns the string "-123.56" in D
```

Err Object

Clear

Syntax

```
Err.Clear
```

Description

The Clear method clears the error object.

Examples

```
Err.Clear
```

Description

Syntax

```
Err.Description
```

Description

The Description property sets or returns a description of the runtime error.

Examples

```
MsgBox "Runtime error : " & Err.Description
```

Number

Syntax

```
Err.Number
```

Description

The Number property sets or returns the number of the runtime error.

The error numbers raised by Pythagoras VBA and Pythagoras Object Model are described in the section Error Handling.

Examples

```
MsgBox "Runtime error : " & Err.Number & vbCrLf & Err.Description
```

On Error

Syntax

```
On Error GoTo Label
```

or

```
On Error GoTo 0
```

or

```
On Error Resume Next
```

Description

On Error GoTo Label: When a run-time error occurs, program execution will continue at Label.

On Error GoTo 0 : Error handling is disabled until the next On Error statement is executed.

On Error Resume Next : When a run-time error occurs, program execution continues with the statement following the statement that caused the error.

Examples

```
On Error GoTo ErrorHandler
```

```
ErrorHandler:  
'Handle Error
```

Raise

Syntax

```
Err.Raise Number, Source, Description, HelpFile, HelpContext
```

Description

The Raise method raises a runtime error.

Number : Long
Source : String
Description : String
HelpFile : String
HelpContext : Long

This method should be called to reraise the currently processed error. For this the method should be called with the current property values of the object, see example on Database.LockDatabase method.

Examples

```
Err.Raise 125, "My program", "Invalid data", "", 0
```

Resume

Syntax

```
Resume  
Resume Next  
Resume Label
```

Description

The Resume statements are used to continue execution after error handling.

1. Resume :

a) The error-handling routine is in the same procedure as the statement that caused the error:

Program resumes (goes back to) the statement that caused the error.

b) The error occurred in a procedure called by the procedure containing the error handling:
Program resumes (goes back to) the statement in the procedure containing the error handler that called the external procedure.

2. Resume Next

a) The error-handling routine is in the same procedure as the statement that caused the error:

Program continues with the statement following the statement that caused the error.

b) The error occurred in a procedure called by the procedure containing the error handling:
Program continues with the statement in the procedure containing the error handler immediately following the statement that called the external procedure.

3. Resume Label

Program continues at the specified label. Label must be in the same procedure as the error handler.

Examples

```
On Error GoTo ErrorHandler
```

```
B = 0.0  
C = A/B  
If C = ptMaxDouble Then  
    ' Division by 0  
End If  
Exit Sub
```

```
ErrorHandle:
    C = ptMaxDouble
    Resume Next
```

Source

Syntax

```
Err.Source
```

Description

The **Source** property sets or returns the source (String) of the runtime error.
The error numbers raised by Pythagoras VBA and Pythagoras Object Model return as source "1"

Examples

```
MsgBox "Runtime error : Source = " & Err.Source
```

File Handling

FreeFile

Syntax

```
FreeFile
```

Description

Returns the number of the next available file.

Examples

```
Dim f As Integer
Dim FileName As String

f = FreeFile
Open FileName For Input As f
```

Open

Syntax

```
Open FileName For mode [Access access] [lock] [format] As
[#]FileNumber, [Len=RecLength]
```

Description

Opens an existing file or creates a new file.

mode: Append, Binary, Input, Output or Random

access: Read, Write or Read Write

lock: Shared, Lock Read, Lock Write or Lock Read Write

format: ASCII (default), UTF8, UTF16LE or UTF16BE.

A new file is created if *FileName* does not exists and **mode**: Append, Binary, Output or Random.

Examples

```
Type tRecord
    ClientNumber As Integer
    Name As String * 24
    Country As String * 24
End Type

Dim Record As tRecord

Open FileName1 For Random As 2 Len = Len(Record)
Open FileName2 For Input As 3
Open FileName3 For Random Access Read Shared As 4 Len = Len(Record)
```

Close

Syntax

```
Close [FileNumber]
```

Description

The `Close` statement closes the file opened with the `Open` statement.

Examples

```
Close #2
```

Reset

Syntax

```
Reset
```

Description

Closes all files opened with the `Open` statement.

Examples

```
Reset
```

Print

Syntax

```
Print #FileNumber, [outputlist]
```

Description

Writes formatted data to a file that is opened for append or output.

Examples

```
' Output the string "Hello World" i= 10"
i = 10
Print #2, "Hello"; Spc(2); "World"; Tab(20); "i="; i
```

Line Input

Syntax

```
Line Input #FileNumber, Str
```

Description

Reads one line from a file and assigns it to as String variable.

Examples

```
Dim Str As String

Line Input #1, Str
```

Write

Syntax

```
Write #FileNumber, [outputlist]
```

Description

Writes structured data to a file that is opened for append or output.

The counterpart of the `Write` statement is the `Input` statement.

Examples

```
Dim Str1 As String

Write #1, 1234, 12.34, True, Str1
Write #1, ' A blank line will be written
to the file
```

Input

Syntax

```
Input (Number, [#]FileNumber)
```

Description

Returns next *Number* (long) characters read from the file.

Examples

```
Open "t.txt" For Input As 1
MsgBox Input (100,1) " First 100 characters of a file are shown
```

```
Close #1
```

Put

Syntax

```
Put FileNumber, [record], Data
```

Description

Writes data from a program variable to a file opened in random or binary mode.
The counterpart of the Put statement is the Get statement.

Examples

```
Type tRecord
    ClientNumber As Integer
    Name As String * 24
    Country As String * 24
End Type

Dim Record As tRecord

Open FileName For Random As 2 Len = Len(Record)
Put #2, 15, Record           ' Writes record 15
Put #2, , Record             ' Writes next record (nr 16)
```

Get

Syntax

```
Get FileNumber, [record], Data
```

Description

Reads from a file opened in random or binary mode.

Examples

```
Type tRecord
    ClientNumber As Integer
    Name As String * 24
    Country As String * 24
End Type

Dim Record As tRecord

Open FileName For Random As 2 Len = Len(Record)
Get #2, 15, Record           ' Reads record 15
Get #2, , Record             ' Reads next record (nr 16)
```

Seek Function

Syntax

```
Seek(FileNumber)
```

Description

Returns the current position in the open file. If the file was opened in random mode, the number returned by the Seek function refers to the next record to be written or read. In all other modes (append, binary, input and output), the number returned by the Seek function is the byte position at which the next read or write operation starts.

Examples

```
Pos = Seek(2)
```

Seek Statement

Syntax

```
Seek [#]FileNumber, position
```

Description

Sets the current position in the open file. If the file was opened in random mode, *Position* refers to the next record to be written or read. In all other modes (append, binary, input and output), *Position* is the byte at which the next read or write operation starts.

Position must be > 0.

Examples

```
Seek #2, 15
```

Lock

Syntax

```
Lock [#]FileName[, RecordRange]
```

Description

Locks a file, a record or a range of records.

Examples

```
Lock #1
Unlock #1
Lock #1, 1 To 9
Unlock #1, 1 To 9
```

Unlock

Syntax

```
Unlock [#]FileName[, RecordRange]
```

Description

Unlocks a file, a record or a range of records.

Examples

```
See Lock
```

Kill

Syntax

```
Kill FileName
```

Description

Deletes one or more files/folders.

The file name may contain wild characters (?, *)

Examples

```
Kill FileName
Kill C"\temp\*.doc" ' Removes all files that end with .doc
```

Loc

Syntax

```
Loc (FileName)
```

Description

Returns the position in the file.

- If the file is open in random mode, Loc returns the record number in the file.

- If the file is open in input or output mode, Loc returns the current byte position in the file divided by 128.

- If the file is open in binary mode, Loc returns the current byte position.

Examples

```
Pos = Loc(2)
```

Eof

Syntax

```
Eof (FileName)
```

Description

Eof returns true if end of file is reached.

Examples

```
Open "test.txt" For Input as 1
Do While Not Eof(1)
    Input #1, i
```

```

Loop
Close #1

```

Lof

Syntax

```
Lof (FileName)
```

Description

Returns the length of the open file in bytes.

Examples

```
Length = Lof(2)
```

FileAttr

Syntax

```
FileAttr (FileName, 1)
```

Description

The file access mode is returned as a long integer:

```

Input      : 1
Output     : 2
Random     : 4
Append     : 8
Binary     : 32

```

Examples

```

Type tRecord
    ClientNumber As Integer
    Name As String * 24
    Country As String * 24
End Type

Dim Record As tRecord

Open "test.bin" For Random As 1 Len = Len(Record)
If FileAttr(1, 1) <> 4 then
    ' failure "FileAttr()"
End If

```

FileCopy

Syntax

```
FileCopy FileName, FileName
```

Description

Makes a copy of the file.

Examples

```
FileCopy "Original.txt", "Copy.txt"
```

FileLen

Syntax

```
FileLen (FileName)
```

Description

Returns the length of the file in bytes.

Examples

```
L = FileLen("Original.txt")
```

Dir

Syntax

```
Dir [(PathName[, Attributes])]
```


Description

Returns a string containing the name of a file or directory that matches a specified path name and file attributes.

PathName: is a string expression that may specify a drive, a directory, and a file name.

The wildcard characters '*' and '?' may be used to specify multiple files/directories.

Attributes: is a constant or numeric expression specifying file/directory attributes to be matched. This can be a sum of several attribute constants specified below. If the argument *Attributes* is omitted, vbNormal is supposed. The attribute constants are:

vbNormal	: 0	files without attributes.
vbReadOnly	: 1	read-only files/directories.
vbHidden	: 2	hidden files/directories.
vbSystem	: 4	system files/directories.
vbVolume	: 8	volume label. Other attributes will be ignored.
vbDirectory	: 16	directories.

Dir returns the first file name that matches pathname and attributes. To get additional matched file names the function should be called again without arguments. When no more file names match, an empty string ("") is returned. In this case, the next call of Dir must specify the argument *Pathname*, otherwise the run-time error 5 ("Invalid procedure call or argument") will result.

Examples

```
` Message all hidden system files with extension ".txt"  
` in the directory C:\Temp
```

```
Dim S As String
```

```
S = Dir("C:\Temp\*.txt", vbHidden+vbSystem)
```

```
Do While S <> ""
```

```
    MsgBox S
```

```
    S = Dir
```

```
Loop
```

MkDir

Syntax

```
MkDir PathName
```

Description

Creates a directory with the specified path name.

PathName: is a string expression that may specify a drive, a directory, and a file name.

Examples

```
MkDir "C:\Temp"
```

Interaction Functions

InputBox

Syntax

```
InputBox(prompt [, title] [, default] [, xpos] [, ypos])
```

Description

InputBox returns a string containing the contents of the edit field.

Prompt : the text of the message to be displayed in the message box.

Title : The title of the message box.

Default : The default value in the input field.

xpos, ypos : Position of the input box on the screen. (Windows only)

Examples

```
InputStr = InputBox("Enter 23", "My Title", "32")
```

MsgBox Function/Statement

Syntax

```
MsgBox(Prompt[, Buttons][, Title])
```

Description

Prompt : the text of the message to be displayed in the message box.

Buttons : the sum of the *Button*, *Icon*, *Default Button*, and *Modality* constant values.

For more information see chapter **Constants**.

Title : The title of the message box.

If used as a function, MsgBox returns one of the following values : vbOk, vbCancel, vbAbort, vbRetry, vbIgnore, vbYes, vbNo.

Examples

```
If MsgBox("Error. Continue?", vbYesNo+vbQuestion, "Title") = vbYes  
Then  
End If
```

```
MsgBox "This message is generated by the VBA Manual", _  
vbOkOnly+vbExclamation+vbApplicationModal, "VBA Manual"
```

Math Functions

Abs

Description

Returns the absolute value of a given number.

Atn

Description

Returns the arctangent of a given number.

Cos

Description

Returns the cosine of a number.

Exp

Description

Returns the base of the natural logarithm raised to a power.

Log

Description

Returns the natural logarithm of a number.

Sgn

Description

Returns -1 if expression is negative, and returns +1 if expression is positive.

Sin

Description

Returns the sine of a number.

Sqr

Description

Returns the square root of a number.

Tan

Description

Returns the tangent of a number.

String Manipulation

ASCII and Unicode strings

Starting from v14 all strings in Pythagoras VBA are represented in Unicode. But in some cases it may be necessary to work with ASCII string, mainly this concerns external procedure calls. Both ASCII and Unicode strings may be represented in Pythagoras VBA as expressions of two types: either `String` or `Byte()`.

Note that normal string procedures such as `Left()`, `Len()` work with ASCII strings contained in `String` variables wrongly and, therefore should not be applied, only `LenB()` function returns correct result. Such ASCII strings however can be transferred to external procedures or obtained from them, see also description of `Declare` statement.

When a string is contained in a value of type `Byte()` then every ASCII byte occupies one element of an array, every Unicode character occupies two elements. Particularly n -th letter of ASCII string B can be obtained as `Chr(B(n))`, length of the string can be obtained as `UBound(B)+1`. Such strings can be also transferred to external procedures or obtained from them.

Strings can be assigned between expressions and variables of these two types.

See examples for `Declare` statement.

Asc

Syntax

`Asc(String)`

Description

Returns the integer value being ASCII code of the first character of the string. -1 is returned if the character is not representable in ASCII.

Examples

```
Dim i As Integer
i = Asc("ABC")           ' i = 65
```

AscW

Syntax

`AscW(String)`

Description

Returns the integer value being Unicode code of the first character of the string.

Example

```
Dim i As Integer
i = AscW("∞")           ' i = &H221E
```

Chr

Syntax

`Chr(Integer)`

Description

Returns a string.

Examples

Returns a string containing only one character with ASCII code equal to an argument value. An argument value must be ≥ 0 and ≤ 255 .

ChrW

Syntax

`ChrW(Integer)`

Description

Returns a string containing only one character with Unicode code equal to an argument value.

Example

```
S = ChrW(&H221E) ' S = "∞"
```

Format

Syntax

`Format(expression[, format])`

Description

Format converts *expression* to a string. *Format*, is a valid name (eg "Fixed") or a user-defined format.

Expression : Any numeric expression. (E.g. V1 * V2 / 0.5).

Format :

Predefined formats :

General Number, Currency, Fixed, Standard, Percent, Scientific, Yes/No, True/False, On/Off

```
Examples : Format(300.123, "Scientific")      \returns
3.E+02
           Format(562486.2356, "Scientific")   \returns
5.62E+05
           Format(562486.2356, "Standard")     \returns
562,486.24
```

Characters that create user-defined number formats :

(0) : Digit placeholder. If *expression* contains a digit in the appropriate position, the digit is display. otherwise a 0 is displayed.

Example : `Format(1.316, "00.00")` returns 01.32

(#) : Digit placeholder

Example : `Format(1.316, "##.##")` returns 1.32

(.) : Decimal placeholder. Pythagoras VBA always displays a ".", in contrast to standard VBA where the actual character depends on the international settings of the Windows system.

(,) : Thousands separator. Pythagoras VBA always displays a ",", in contrast to standard VBA where the actual character depends on the international settings of the Windows system.

(%) : Percentage placeholder

Example : `Format(0.31, "##.##%")` returns 31.00%

(E- E+ e- e+) : Scientific format

Examples : `Format(300.123, "##.##E+")` returns 30.01E+01

`Format(300.123, "##.##E-")` returns 30.01E01

- + \$ () : Displays a literal character.

Example : `Format(300.123, "(000.00)")` returns (300.12)

\ : The character following the backslash is displayed as a literal character.

Example : `Format(300.123, "000.00\ \!")` returns 300.12 !

Predefined date and time formats :

General Date, Long Date, Medium Date, Short Date, Long Time, Medium Time, Short Time.

Characters that create user-defined date and time formats :

c : Displays date and/or time.

d : The day as a number without a leading zero (1..31)
 dd ; The day as a number with a leading zero (01..31)
 ddd : The day as an abbreviation (Sun..Sat)
 dddd : The day as a full name (Sunday..Saturday)
 ddddd : Short day format based on computer settings.
 ddddddd : Long day format based on computer settings.
 w : The day of the week. (1..7). 1 = Sunday
 ww : The week of the year. (1..54)
 m : The month without leading zero (1..12)
 mm : The month with leading zero (01..12)
 mmm : The month as an abbreviation (Jan..Dec)
 mmmm : The name of the month (January..December)
 q : The quarter of the year (1..4)
 y : The day of the year (1..366)
 yy : The year (00..99)
 yyyy : The year. Eg. 2004.
 h : The hour without leading zeros (0..23)
 hh : The hour with leading zeros (00..23)
 n : The minutes without leading zeros (0..59)
 nn : The minutes with leading zeros (00..59)
 s : The seconds without leading zeros (0..23)
 ss : The seconds with leading zeros (00..59)
 AM/PM : the 12-hour clock using uppercase AM and PM.
 am/pm : the 12-hour clock using lowercase am and pm.
 A/P : the 12-hour clock using uppercase A or P (for AM/PM)
 a/p : the 12-hour clock using lowercase a or p (for am/pm)
 tttt : the long time format based on computer settings.

Example :

```

Format(#2000-01-01 00:00:00#, "dd-mm-yyyy = dddd mmmm d yyyy hh-
nn-ss")
returns : "01-01-2000 = Saturday January 1 2000 00-00-00"
  
```

Examples

see above examples

InStr

Syntax

```
InStr([start, ] StringToSearch, StringToFind [, CompareMode])
```

Description

Returns the starting position of *StringToFind* in *StringToSearch*. If the string is not found the value 0 is returned.

Start argument must be given if *CompareMode* argument is present.

CompareMode: either *vbTextCompare* or *vbBinaryCompare* (see example below). If the argument is omitted *vbBinaryCompare* is used.

Examples

```
Dim Pos As Long
```

```

Pos = InStr(1, "1234ABC890", "abc", vbTextCompare)      ' Pos = 5
Pos = InStr(1, "1234ABC890", "abc", vbBinaryCompare)    ' Pos = 0
  
```

InStrRev

Syntax

```
InStrRev(StringToSearch, StringToFind [,start [, CompareMode]])
```

Description

Returns the ending position of *StringToFind* in *StringToSearch*. Searching is performed from right to left. If the string is not found the value 0 is returned. If *Start* position is not specified the value of -1 is assumed, which means that the search begins at the last character.

Start argument must be given if *CompareMode* argument is present.

Examples

```
Dim Pos As Long

Pos = InStrRev("C:\Work\Projects\BigRoad.PYT", "\")      ' Pos = 17
Pos = InStrRev("C:\Work\Projects\BigRoad.PYT", "\", 16)  ' Pos = 8
```

LCase

Syntax

LCase(*String*)

Description

Returns the *String* converted to lower case

Examples

```
S = LCase("How Are You")      ' Returns "how are you"
```

Left

Syntax

Left(*string*, *length*)

Description

Returns the leftmost *length* characters of *string*.

Examples

```
S = Left("1234567890", 5)      ' Returns "12345"
```

Len

Syntax

Len(*string* | *VarName*)

Description

Returns the number (long) of characters of *string*, or the size of *VarName* in bytes. It should not be applied to ASCII strings, use LenB() function instead. The size of a variable is calculated assuming that every character occupies a byte, such size is suitable as a record length in Open statement for files opened with ASCII option (default).

Examples

```
L = Len("12345")      ' Returns 5
L = Len(Record)        ' Returns the # of bytes in variable Record.
```

LenB

Syntax

LenB(*string* | *VarName*)

Description

Returns the number (long) of bytes in *string*, or the size of *VarName* in bytes. It is especially useful for ASCII strings where returned value represents length of a string. Being applied to variables it returns the same result as Len.

Examples

```
S = StrConv ("ABC", vbFromUnicode)
L = LenB(S)      ' Returns 3
```

LenW

Syntax

LenW(*string* | *VarName*)

Description

Returns the number (long) of characters of *string*, or the size of *VarName* in bytes. It should not be applied to ASCII strings, use LenB() function, please. The size of a variable is calculated assuming that every character occupies two bytes, such size is suitable as a record length in Open statement for files opened with UTF16LE option.

Examples

```
L = LenW("12345") ' Returns 5
L = LenW(Record)  ' Returns the # of bytes in variable Record.
```

StrConv

Syntax

```
StrConv (InputString, Conversion As VbStrConv) As String
```

Description

Converts *InputString* and returns the result.

Type of conversion depends on *Conversion* argument value:

vbFromUnicode : *InputString* is converted to ASCII, see also subsection “ASCII and Unicode strings”.

vbUnicode : *InputString* should be ASCII string and is converted to Unicode, see also subsection “ASCII and Unicode strings”.

vbLowerCase, *vbUpperCase* : *InputString* is converted to lower or upper case.

When type of conversion is specified as 0 then the same string is returned, but if an input string contains a zero character inside, the result will be cut.

Examples

```
Dim S As String
S = "Pythagoras VBA"
S = StrConv (S, vbUpperCase) ' S becomes "PYTHAGORAS VBA".
See also examples for Declare statement.
```

LTrim

Syntax

```
LTrim(StringExpression)
```

Description

Removes all leading spaces from *StringExpression* and returns the result in a string.

Examples

```
S = LTrim("   ABC" & "DEF") ' Returns "ABCDEF"
```

Mid Function

Syntax

```
Mid(string, start[, length])
```

Description

Returns a substring of *string* eventually with the given *length*.

Examples

```
S = Mid("1234567890", 5, 3) ' Returns "567"
S = Mid("1234567890", 5)    ' Returns "567890"
```

Mid Statement

Syntax

```
Mid(StringVar, start[, length]) = String
```

Description

Replaces the substring of *StringVar* with *String*

Examples

```
S = "1234567890"
Mid(S, 5, 3) = "ABC" ' S becomes "1234ABC890"
```

Right

Syntax

```
Right(string, length)
```

Description

Returns a string containing the rightmost *length* characters of *string*.

Examples

```
S = Right("1234567890", 5)      ' S becomes "67890"
```

RTrim

Syntax

```
RTrim(StringExpr)
```

Description

Removes all trailing spaces from *StringExpr*

Examples

```
S = RTrim("ABC   ")           ' S becomes "ABC"
```

Space

Syntax

```
Space(n)
```

Description

Returns *n* spaces.

Examples

```
MsgBox "Result =" & Space(5) & CStr(Value)
```

Spc

Syntax

```
Spc(n)
```

Description

Inserts *n* spaces between expressions in a Print statement.

Examples

```
Print #1, "Hello"; Spc(2); "World"; Tab(20); "i="; i
```

StrComp

Syntax

```
StrComp(string1, string2 [, compare])
```

Description

Compares *string1* and *string2* and returns the result.

Returns value:

```
string1 < string2 : -1
```

```
string1 = string2 : 0
```

```
string1 > string2 : 1
```

compare : vbBinaryCompare (default) or vbTextCompare.

Note the =, < and > operators may be used with Strings. E.g.

```
If S1 > S2 Then
```

```
End If
```

Examples

```
Dim i As Integer
i = StrComp("Hello", "HELLO", vbTextcompare)      ' i becomes = 0.
i = StrComp("Hello", "Hello2")                    ' i becomes = -
1.
i = StrComp("z", "A")                               ' i becomes = 1.
```

String

Syntax

```
String(number, character)
```

Description

Returns a string containing *character* repeated *number* of times.

Examples

```
S = String(5, "!")           ' Returns "!!!!!"
```


Tab

Syntax

`Tab (Column)`

Description

Moves the text insertion point to the given column.
May only be used in expressions in a `Print` statement.

Examples

`Print #1, Tab(10); "Value 1"; Tab(20); i`

Trim

Syntax

`Trim (StringExpr)`

Description

Removes all leading and trailing spaces from *StringExpr*

Examples

`S = Trim(" ET Phone Home ")` ' Returns "ET Phone Home"

UCase

Syntax

`UCase (String)`

Description

Returns the *String* converted to upper case

Examples

`S = UCase("How Are You")` ' Returns "HOW ARE YOU"

Date functions

Date literals

The following formats are supported :

#yyyy-MM-dd hh:mm:ss.ssssss#
#yyyy-MM-dd hh:mm:ss.ssssss#
#yyyy-MM-dd hh:mm:ss.sssss#
#yyyy-MM-dd hh:mm:ss.ssss#
#yyyy-MM-dd hh:mm:ss.sss#
#yyyy-MM-dd hh:mm:ss.ss#
#yyyy-MM-dd hh:mm:ss.s#
#yyyy-MM-dd hh:mm:ss#
#yyyy-MM-dd hh:mm#
#yyyy-MM-dd hh#
#yyyy-MM-dd#

Date

Syntax

`Date`

Description

Returns the current date.

Examples

`MsgBox Cstr(Date)`

DateAdd

Syntax

`DateAdd(interval, number, date)`

Description

Returns a date as variant.

interval : indicates the meaning of *number*. The interval must be a string.

Valid values: "yyyy" = Year
"q" = Quarter
"m" = Month
"y", "d", "w" = Days
"h" = Hours
"n" = Minutes
"s" = Seconds

number : the number of days, hours, minutes or seconds added to the date.

date : the input date.

Examples

```
Dim D As Variant
```

```
D = DateAdd("yyyy", 1, #2000-01-01 00:00:00#)
D = DateAdd("q", 1, D)
D = DateAdd("m", 1, D)
D = DateAdd("y", 1, D)
D = DateAdd("ww", 1, D)
D = DateAdd("h", 1, D)
D = DateAdd("n", 1, D)
D = DateAdd("s", 1, D)
MsgBox "Result = " & CStr(D)
```

DateDiff

Syntax

```
DateDiff(interval, date1, date2)
```

Description

Returns the difference of 2 dates. The meaning of the result depends on the parameter *interval*.

The result is of type variant. VarType of result = Long.

interval : see function DateAdd

date1 and *date2* : the input dates.

Examples

```
Dim S As String
Dim D As Variant

D = DateAdd("d", 7, Now)
MsgBox Cstr(DateDiff("s", Now, D))
```

DatePart

Syntax

```
DatePart(interval, date [, FirstDayOfWeek])
```

Description

Returns part (year, month, day,...) of the date. The meaning of the result depends on the parameter *interval*.

The result is of type variant. VarType of result = Integer.

interval : indicates the part (eg. month) of the date must be returned .

Valid values: see function DateAdd

date : the input date.

FirstDayOfWeek : by default = vbSunday. Indicates the starting day of the week (day 1).

Examples

```
Dim D As Date

D = #2005-08-13#
MsgBox "Date = " & CStr(D) & vbCrLf & _
"Day of week = " & CStr(DatePart("w", D, vbMonday))
```

DateSerial

Syntax

```
DateSerial(year, month, day)
```

Description

Returns the date based on the sum of *year*, *month*, *day*. Each of the input parameters is of type Integer. *month* may be larger than 12 and *day* may be larger than 31.

The result is of type variant. VarType of result = Date.

Examples

```
Dim D As Date
```

```
D = DateSerial(2000, 26, 30) ' returns 2002-03-02
```

Day

Syntax

```
Day(DateExpression)
```

Description

Returns the day (1..31) of the given date.

Examples

```
Dim D As Variant
```

```
D = Day(Now) ' Returns the current day.
```

Hour

Syntax

```
Hour(DateExpression)
```

Description

Returns the hour (0..23) of the given date.

Examples

```
Dim D As Variant
```

```
D = Hour(#2005-02-28 12#) ' Returns 12.
```

Minute

Syntax

```
Minute(DateExpression)
```

Description

Returns the minutes (0..59) of the given date.

DateExpression must be a valid date and time.

Examples

```
Dim D As Variant
```

```
D = Minute(#2005-02-28 12:35:20#) ' Returns 35.
```

Month

Syntax

```
Month(DateExpression)
```

Description

Returns the month (1..12) of the given date.

Examples

```
Dim D As Variant
```

```
D = Month(#2005-02-28 12#) ' Returns 2.
```

MonthName

Syntax

```
MonthName (MonthNumber [, Abbreviate])
```

Description

Returns the name of the month (January .. December). If `Abbreviate = True`, the abbreviated form of the name of the month is returned.

Examples

```
MsgBox MonthName(10)           ' Returns October
MsgBox MonthName(10, True)     ' Returns Oct
```

Now

Syntax

```
Now
```

Description

Returns the current date and time base on the system settings.

Examples

```
Dim D As Variant

D = Day(Now)                                ' Returns 14:23:40
```

Second

Syntax

```
Second(DateExpression)
```

Description

Returns the seconds (0..59) of the given date.

Examples

```
Dim D As Variant

D = Second(#2005-02-28 12:35:20#)           ' Returns 20.
```

Time

Syntax

```
Time
```

Description

Returns the current time.

Examples

```
MsgBox Cstr(Time)
```

TimeSerial

Syntax

```
TimeSerial(hours, minutes, seconds)
```

Description

Returns the time based on the sum of *hours*, *minutes*, *seconds*. Each of the input parameters is of type Integer.

The result is of type variant. VarType of result = Date.

Examples

```
Dim D As Date

D = TimeSerial(23, 30, 30)                  ' returns 23:30:30
```

TimeValue

Syntax

```
TimeValue(time)
```

Description

Returns the conversion of a String containing a date to a Variant type. The variant will be of type Date.

time = string containing a time.

Examples

```
Dim D As Date
```

```
D = TimeValue("#12:35:50#")           ' returns 12:35:50
```

Weekday

Syntax

```
Weekday(date [, FirstDayOfWeek])
```

Description

Returns the day of the week of *date* (1 .. 7). The result is a Variant of type Integer.

FirstDayOfWeek = specifies the first day of the week. Default = vbUseSystem.

Examples

```
Dim MyDay As Variant
```

```
MyDay = Weekday("#2005-10-12#", vbMonday)      ' Returns 5
```

WeekdayName

Syntax

```
WeekdayName(DayNumber [, Abbreviate [, FirstDayOfWeek]])
```

Description

Returns the name of the day (Monday .. Sunday) as a String. If *Abbreviate* = True, the abbreviated form of the name of the day is returned. If *FirstDayOfWeek* is omitted, Sunday is the first day of the week.

Examples

```
MsgBox WeekdayName(5, False, vbMonday)        ' Returns Friday  
MsgBox WeekdayName(5, True, vbMonday)         ' Returns Fri
```

Year

Syntax

```
Year(DateExpression)
```

Description

Returns the year of the given date.

Examples

```
Dim D As Variant
```

```
D = Year("#2005-02-28 12:35:20#")      ' Returns 2005.
```

Various

TypeName

Syntax

```
TypeName(variablename)
```

Description

Returns the name of the type of *variablename*. The variable must be of type Variant.

Examples

```
Dim V1 As Variant
```

```
V1 = 1.5  
MsgBox TypeName(V1)      ' Will display Double.
```

VarType

Syntax

`VarType(variablename)`

Description

Returns the actual type of *variablename*. The variable must be of type Variant.

Examples

```
Dim V1 As Variant

V1 = 1.5
If VarType(V1) = vbDouble Then
    MsgBox "Type = Double"
End If
```

IsDate, IsEmpty, IsNull, IsError, IsNumeric, IsObject

Syntax

`IsX(variablename)`

Description

Returns true if the variable is of the specified type.

When option PlusMSCompatible is off then IsNumeric function also returns True for string values convertible to either integer or real values.

Examples

```
Dim D As Variant

If IsDate(D) Then
End If

If IsNull(D) Then          ' Do not use If D = Null !!!
End If
```

IsEmptyArray

Syntax

`IsEmptyArray(varname)`

Description

It returns True if the dynamic array is empty. A variable of a dynamic array type initially is empty.

Examples

```
Dim A As Double(), Sum As Double, I As Long

' Calculate a sum
Sum = 0.0
If Not IsEmptyArray(A) Then
    For I = LBound(A) To UBound(A) Do
        Sum = Sum + A(I)
    Next I
End If
```

UBound

Syntax

`UBound(arrayname[, dimension])`

Description

The UBound function returns the largest index of the array, or of the dimension of the array.

Examples

```
Dim Names(1 to 10) As String
Dim i As Integer

For i = LBound(Names) To UBound(Names)
```

```
Print #1, Names(i)
Next i
```

LBound

Syntax

```
LBound(arrayname[, dimension])
```

Description

The LBound function returns the smallest index of the array, or of the dimension of the array.

Examples

See UBound.

Beep

Syntax

```
Beep
```

Description

Sends a Beep

Examples

```
On Error Resume Next
'Code

Resume Next
Beep
```

AppActivate

Syntax

```
AppActivate Title [,wait]
or
AppActivate ApplicationTaskId [,wait]
```

Description

Sets the focus to the application with the given *Title* or with the given *ApplicationTaskId*. The *ApplicationTaskId* is returned from the Shell function. The optional parameter *wait* is not used.

Examples

```
Dim AppId As Long

AppActivate AppId
```

Shell

Syntax

```
Shell (ProgramName, [windowStyle])
```

Description

Launches the program specified by *ProgramName*. If successful, the application's task Id is returned.

The applications task Id can be used as a parameter to the AppActivate statement. If the application can not be launched, the runtime error 53 is generated.

The parameter *windowStyle* can have the following values :

vbHide, vbNormalFocus, vbMinimizedFocus, vbMaximizedFocus, vbNormalNoFocus, vbMinimizedNoFocus. If this parameter is not given, the default is vbMinimizedFocus.

Examples

```
Dim AppId As Long

On Error GoTo ErrorHandler

AppId = _
```

```

    Shell("C:\Program Files\Internet Explorer\IExplore.exe
www.pythagoras.net", _
vbNormalFocus)
Exit Sub

ErrorHandler :
    MsgBox "Program could not be launched"

```

Declare

Syntax (for subroutines)

```

[Public | Private] Declare [PtrSafe] Sub name Lib "libname" [Alias
"name"] [(arglist)]

```

Syntax (for functions)

```

[Public | Private] Declare [PtrSafe] Function name Lib "libname"
[Alias "name"] [(arglist)] As type

```

Description

The `Declare` statement is used to define references to external procedures in a DLL. *Name* is used both for calling procedure and a name within DLL or code library. If a name of the procedure is inconvenient for the code `Alias` clause should be used. In this case *name* standing after `Sub` or `Function` keyword should be used to call the procedure, an alias name should be a name within DLL.

`PtrSafe` directive is ignored by 32-bit Pythagoras VBA compiler but is mandatory in 64-bit Pythagoras. Its goal is to prevent the code written for 32-bit Pythagoras to run in the 64-bit version without revision. VBA programmers should insert the directive into the old code only after checking that the procedure declaration does not need any type correction like changing `Long` to `LongPtr`.

Examples

```

'Declare:
Declare PtrSafe Function GetVersion Lib "kernel32"() As Long

Dim WindowsVersion As Long
WindowsVersion = GetVersion()

```

Rules how to describe types of arguments

Only arguments and return values of certain types are supported.

Numeric arguments should be described as arguments of the corresponding type and corresponding transfer mode (by value or by reference). For example, if an argument is described in Visual C++ notation as “`short`”, it should be described as “`ByVal Integer`”. If an argument is described in C++ notation as “`int *`” or “`int &`”, it should be described as “`ByRef Long`”.

Boolean values should be described as integers of the corresponding size.

Both ASCII strings transferred as “`char *`” and Unicode strings transferred as “`wchar_t *`” should be described as “`Byte()`” or “`String`”. See also subsection “ASCII and Unicode strings” of section “String manipulations” about how to handle ASCII strings in VBA code. Either “`ByRef`” or “`ByVal`” does not matter in this case.

If a called procedure returns a string value then it is necessary to transfer a variable of sufficient volume. For a variable of `String` type it could be made assigning a value of the necessary size, the most appropriate means is function `String()`. Necessary size for ASCII string is half of number of bytes. For a variable of `Byte()` type resizing could be made using `ReDim` statement. Note that returned string will have the same size as before the call, therefore function `Len()` and some other string procedures could give wrong results. Returned Unicode string can be cut to its actual size using `Left()` function. If a returned string contains a zero character then it can be cut using `StrConv()` function with 0 conversion argument. Note that ASCII strings cannot be cut easily, but they can be converted to Unicode and cut using same `StrConv()` function.

An argument can be an address of address of a string (like “`wchar_t **`”), this is not currently supported.

Arguments being addresses of something else could be described as of type `LongPtr`. Typical cases

are system types in Windows API calls such as HWND. Such values can be obtained, stored and transferred but cannot be processed in VBA code (except comparing on equality to 0 or between them).

Examples with string arguments

```
Declare PtrSafe Function GetSystemDirectoryA Lib "Kernel32" _
    (ByVal lpBuffer() As Byte, ByVal uSize As Long) As Long _
Dim S As String, B() As Byte, L As Long
...
ReDim B(64)
L = GetSystemDirectoryA (B, 64)
S = StrConv (B, vbUnicode)
S = StrConv (S, 0)      ' eliminate extra characters
```

The same result can be obtained simpler calling Unicode API function:

```
Declare PtrSafe Function GetSystemDirectoryW Lib "Kernel32" _
    (ByVal lpBuffer As String, ByVal uSize As Long) As Long _
Dim S As String, L As Long
...
S = String (64, " ")
L = GetSystemDirectoryW (S, 64)
S = StrConv (S, 0)      ' eliminate extra characters
```

Example with system arguments:

```
Declare PtrSafe Function GetForegroundWindow Lib "User32" () _
    As LongPtr
Declare PtrSafe Function MessageBoxW Lib "User32" ( _
    ByVal hWnd As LongPtr, _
    ByVal lpText As String, ByVal lpCaption As String, _
    ByVal uType As Integer _
) As Integer
Dim L As Integer

L = MessageBoxW (GetForegroundWindow(), "Ok", "Test", 0)
```

End

Syntax

End

Description

This is an obsolete feature and we do not recommend its usage.

End terminates the VBA program. If used in an event subroutine the statement terminates only the event subroutine. If used when a Modeless dialog box is active, this statement can cause incorrect termination of a macro.

Examples

End