AVAX Vector .NET v3

Developer Manual

Windows Forms control for .NET Framework 4.8
English edition - November 2026

Complete control reference, current v3 declarations, parameter descriptions, interactive commands, enumerations and VB.NET examples.

https://gardos-software.com/
© 1993-2026 Gardos Software LTD

About this manual

This printable reference is based on the English AVAX Vector .NET v3 compiled help and its current source topics. It includes every HTML topic from the supplied control CHM, using the more recent source where it updates a declaration, example or description.

Use Getting started for references, deployment and initialization. Moving from v2 to v3 explains numeric types and legacy file formats. The Methods, Properties, Events and Enumerations chapters follow the help tree. Additional topics and reference guides collects the supporting feature guides and API pages.

The contents and PDF bookmarks link directly to each topic. Links within topics jump to the corresponding topic in this manual when available. The editable RTF edition contains the same topic bodies, tables and examples; its contents can be refreshed in a word processor.

Read the complete declaration for the overload you call. Use the exact ByRef and array element types, preserve documented array indexing, initialize the control before drawing, and use it on its owning Windows Forms UI thread. File-format limitations and compatibility notes remain part of their individual topics.

Reference and examples

Examples are written in VB.NET. A sample may require named form controls or companion files; follow its Setup section and Using the VB.NET examples. Do not treat an older example file format as a new v3 binary layout merely by changing numeric declarations.

Topic coverage: 550 source HTML pages. Edition: 1 November 2026.

Contents

Introduction and setup 22

Introduction . . . 23

Getting started with AVAX Vector .NET v3 . . . 24

Moving from v2 to v3 . . . 25

Using the VB.NET examples . . . 28

Methods 30

Methods . . . 31

AbortPrinting . . . 40

ActionInProgress . . . 41

Add_AllDimensions . . . 43

Add_AngularDim . . . 46

Add_Arc . . . 49

Add_AVI . . . 52

Add_AVX . . . 54

Add_Bezier . . . 56

Add_Block . . . 59

Add_BlockLines . . . 61

Add_CBSpline . . . 64

Add_Circle . . . 67

Add_CSpline1 . . . 69

Add_CSpline2 . . . 72

Add_CurveArc . . . 75

Add_Data . . . 78

Add_Dim . . . 80

Add_Dim_Aligned . . . 83

Add_Dim_Angular . . . 85

Add_Dim_Diametric . . . 86

Add_Dim_Linear . . . 87
Add_Dim_Radial . . . 88
Add_Document . . . 89
Add_DWG . . . 91
Add_DXF . . . 94
Add_Ellipse . . . 97
Add_ExternalLines . . . 99
Add_Hatch . . . 102
Add_Line . . . 103
Add_LinearHatch . . . 106
Add_OBSpline . . . 109
Add_PCX . . . 112
Add_Picture . . . 114
Add_PictureFile . . . 116
Add_PLT . . . 118
Add_Point . . . 121
Add_PolyLine . . . 123
Add_Rectangle . . . 126
Add_RTF . . . 128
Add_Shape . . . 130
Add_ShapeArc . . . 134
Add_ShapeCircle . . . 138
Add_ShapeDocument . . . 142
Add_ShapeEllipse . . . 144
Add_ShapeLine . . . 148
Add_ShapePoint . . . 150
Add_ShapePolyLine . . . 152
Add_ShapeRectangle . . . 155
Add_ShapeRegularPolygon . . . 159
Add_ShapeText . . . 163

Add_SHP . . . 167
Add_SVG . . . 170
Add_Text . . . 172
Add_TextC . . . 175
Add_TextFile . . . 177
Add_WMF . . . 180
Add_XML . . . 182
AddCompoundPath . . . 185
AddEditableShape . . . 187
AddParametricShape . . . 188
AlignItem . . . 189
AvaxControlGetAllControls . . . 191
AvaxControlGetData . . . 192
AvaxControlGetShape . . . 193
AvaxControlRefresh . . . 194
AvaxControlRefreshAll . . . 195
AvaxControlSetData . . . 196
Editable text styles . . . 197
DoVertex hatch job . . . 199
BeginCompoundPath . . . 201
BeginCurveNodeEditing . . . 202
BeginUpdate . . . 203
CalcBoundsDrawing . . . 204
CenterWorkspacePage . . . 206
ChangeColor . . . 207
ChangeCompoundPath . . . 209
ChangeDocumentFontSize . . . 210
ChangeDocumentWidth . . . 211
ChangeElevation3D . . . 212
ChangeFontNumber . . . 214

ChangeHeight3D 216
ChangeItemOrder 218
ChangeLayer 219
ChangeLineStyle 221
ChangePathSegmentKind 223
ChangePathSegmentParameter 224
ChangePointStyle 225
ChangeProperties 227
ChangeShapeClosed 229
ChangeShapeControlPoint 230
ChangeShapeDocumentFontSize 231
ChangeShapeDocumentWidth 232
ChangeShapeFill 233
ChangeShapeParameter 237
ChangeShapeText 238
ChangeShapezOrder 241
ChangeStrokeWidth 243
ChangeText 244
ChangeWidth 246
ClearUndoRedoMemory 248
CombineCompoundPaths 249
ComparePolygons 250
ConvertShapeToCurves 252
ConvertToShape 253
Copy_Avax 255
CopyInside 257
CopyItems 259
CopyOutside 261
CopyToClipboard 263
CreateACADHatchPattern 265

CreateACADLineStyleString 268
CreateBlock 270
CreateExVectorHatchPattern 272
CreatePatternFlags 275
CreateSVGHatchPattern 277
CreateVectorHatchPattern 279
CurrentRedo 282
CurrentUndo 284
CurrentXY 286
CurrentZoomPercentage 288
DeleteAllItems 289
DeleteAvaxFile 290
DeleteItems 292
DestroyDrawing 294
DirsGet 296
DirsSet 298
DoAvaxAction 300
Draw_Line 302
EndAvax 304
EndUpdate 305
ExportNativeDrawing 306
FindHandlesArrByKey 307
FinishCompoundPath 309
FitDrawingToWorkspacePage 310
FitToAvax 311
Get_Dimension 312
Get_DimensionDefaults 313
Get_DimensionMeasurement 314
Get_DimensionStyle 315
Get_Hatch 316

Get_HatchArea . . . 317
Get_HatchDefaults . . . 318
GetAllHandlesArr . . . 319
GetAllNodes . . . 321
GetArea . . . 323
GetAttributes . . . 325
GetAvaxEvents . . . 327
GetAvaxItemType . . . 328
GetAvaxProperty . . . 329
GetAvaxStatus . . . 331
GetBorder . . . 335
GetBoundsAvax . . . 337
GetBoundsDrawing . . . 339
GetBoundsSelectBox . . . 341
GetBoundsSelectedItems . . . 343
GetBoundsZoom . . . 345
GetCompoundPath . . . 347
GetCurveLines . . . 348
GetDocumentClearText . . . 351
GetDocumentFontSize . . . 352
GetDocumentWidth . . . 353
GetEllipseGeometry . . . 354
GetFontName . . . 356
GetFonts . . . 358
GetHandleKey . . . 360
GetHandlesArrByKeyArr . . . 361
GetHandlesByKey . . . 363
GetHandlesUnderArea . . . 365
GetHandlesUnderSelectCursor . . . 367
GetImageEffectDefinitions . . . 369

GetImageEffects . . . 370
GetImageSource . . . 371
GetItemByLayer . . . 372
GetItemZOrder . . . 373
GetKeyArrByHandlesArr . . . 374
GetLayerColor . . . 376
GetLayerLineStyle . . . 377
GetLayers . . . 378
GetLineStartPoint . . . 380
GetLineStyle . . . 382
GetLineStyleNames . . . 384
GetLineStyles . . . 386
GetOSnapStatus . . . 387
GetPalColor . . . 388
GetPalette . . . 390
GetPaperSize . . . 392
GetParametricShapeKind . . . 395
GetParametricShapes . . . 396
GetPathSegmentParameters . . . 397
GetPerimeter . . . 398
GetPictureSize . . . 400
GetProperties . . . 402
GetSelectedHandlesArr . . . 404
GetSelectedPoints . . . 406
GetSelectRegionBorders . . . 408
GetShapeClosed . . . 410
GetShapeControlPoints . . . 411
GetShapeDocumentText . . . 412
GetShapeFill . . . 414
GetShapeParameters . . . 417

GetShapePicturePixels . . . 418
GetShapeText . . . 420
GetShapezOrder . . . 422
GetStrokeWidth . . . 424
GetText . . . 425
GetTextAttributes . . . 426
GetTextFontNumber . . . 428
GetTextGeometry . . . 430
GetTextStyle . . . 432
GetTextVectors . . . 433
GetWorkspacePageBounds . . . 436
InitProgramID . . . 437
IsImageShape . . . 438
isSelected . . . 439
IsSelectedItem . . . 440
IsTextShape . . . 441
ItemType . . . 442
LineStyleIndexByName . . . 443
Load_HatchPattern . . . 445
LoadACADLineStyles . . . 446
LoadAvaxPropertiesFromXML . . . 448
LoadFonts . . . 450
LoadLineStyles . . . 451
LoadPalette . . . 453
LongToSingle . . . 454
MaxItems . . . 456
MaxSelectedItems . . . 457
MirrorItems . . . 458
modbutton.DrawAvaxButton and NonThemeStyle . . . 460
MoveItems . . . 463

MoveNode . . . 465
NearestPaletteIndexColor . . . 468
OleDrag . . . 470
ParametricShapeDefinitions . . . 471
Paste_Avax . . . 472
PasteFromClipboard . . . 474
PrintScreenToBMP . . . 476
PrintToAPL . . . 477
PrintToBMP . . . 479
PrintToDWG . . . 481
PrintToDXF . . . 483
PrintToGIF . . . 485
PrintTohDC . . . 487
PrintToJPG . . . 490
PrintToPIC . . . 492
PrintToPLT . . . 494
PrintToPNG . . . 496
PrintToPrinter . . . 499
PrintToRTF . . . 502
PrintToSVG . . . 505
PrintToSWF . . . 509
PrintToTIF . . . 511
PrintToWMF . . . 513
PrintToXML . . . 516
PrintWorkspaceToImage . . . 518
PrintWorkspaceToPDF . . . 519
PrintWorkspaceToPrinter . . . 520
PrintWorkspaceToRTF . . . 521
PrintWorkspaceToSVG . . . 522
PrintWorkspaceToWMF . . . 523

RedrawItems . . . 524
Refresh . . . 526
Regen . . . 527
RegenControlsZoomBitmap . . . 528
RenderImagePreview . . . 529
RenderImagePreviewAsync . . . 530
RenderParametricShapeIcon . . . 532
RenderWorkspacePage . . . 533
Reset_DimensionTextPosition . . . 534
ResetImageEffects . . . 535
ReverseCompoundPath . . . 536
RotateItems . . . 537
SaveAs . . . 539
SaveAvaxPropertiesToXML . . . 541
SaveFonts . . . 543
SaveLineStyles . . . 545
SavePalette . . . 547
ScaleItems . . . 549
SelectItems . . . 551
SelectPoints . . . 553
Set_Dimension . . . 556
Set_DimensionStyle . . . 557
Set_DimensionTextPosition . . . 558
Set_Hatch . . . 559
Set_HatchBoundaries . . . 560
SetAvaxProperty . . . 561
SetBoundsAvax . . . 563
SetBoundsDrawing . . . 565
SetBoundsStretch . . . 567
SetBoundsZoom . . . 569

SetCoordinate 571
SetEditablePointSelectionPreview 573
SetFontName 574
SetFonts 576
SetHandleKey 578
SetImageEffects 579
SetItemByLayer 581
SetItemsByLayer 582
SetLayerColor 583
SetLayerLineStyle 584
SetLayers 585
SetLineStyle 587
SetLineStyles 589
SetPalColor 591
SetPalette 593
SetPathNodeMode 595
SetPointSelectionAspectRatio 596
SetPointSelectionPreview 597
SetProperties 598
SetSelectFilter 600
SetSelectRegionBoundingBox 602
SetSVGPalette 604
SetTextStyle 605
SetTextStyles 606
SingleToLong 607
SnapToStepSet 609
SplitPathSegment 611
StartAvax 612
StretchItems 613
TransformItems 616

UnitX . . . 618
UnitY . . . 620
Unlock . . . 622
ZoomIn . . . 623
ZoomOut . . . 625
ZoomWorkspacePage . . . 627

Properties **628**

Properties . . . 629
AutoRedraw . . . 632
Native dimension methods and options . . . 634
BackColor . . . 643
BorderStyle . . . 644
Command . . . 645
CommandFilePicker . . . 648
CompoundPathClosed . . . 650
CompoundPathSegmentKind . . . 651
ContinueUndo . . . 652
DefaultExtension . . . 654
DrawingUnitMM . . . 655
Enabled . . . 656
filename . . . 657
FontBold . . . 658
FontCharset . . . 659
FontItalic . . . 661
FontName . . . 662
FontSize . . . 664
FontStrikethrough . . . 665
FontUnderline . . . 666
ForeColor . . . 667
HasWorkspaceSettings . . . 668

hdc . . . 669
hwnd . . . 670
IsChanged . . . 671
IsCreatingCompoundPath . . . 673
Language . . . 674
LastHandle . . . 675
LastImportWarnings . . . 676
LastOperationError . . . 677
LockEdit . . . 678
OriginVisible . . . 679
PageBackgroundColor . . . 680
PageBackgroundPrintExport . . . 681
PageCustomHeightMM . . . 682
PageCustomWidthMM . . . 683
PageHeightMM . . . 684
PageMarginMM . . . 685
PageMarginsVisible . . . 686
PageNumber . . . 687
PageOrientation . . . 688
PageOriginX . . . 689
PageOriginY . . . 690
PagePaperSize . . . 691
PagePrintUnitMM . . . 692
PageScale . . . 693
PageShadowVisible . . . 694
PageTitle . . . 695
PageWidthMM . . . 696
ParameterDialogInput . . . 697
PauseAvax . . . 699
PauseAvaxEvents . . . 701

Pen3DElevetion . . . 702
Pen3DHeight . . . 703
PenColor . . . 704
PenColorByLayer . . . 706
PenFontNumber . . . 707
PenFontScale . . . 709
PenLayer . . . 711
PenLineStyle . . . 713
PenLineStyleByLayer . . . 715
PenLineStyleScale . . . 716
PenPointRadius . . . 718
PenPointStyle . . . 720
PenWidth . . . 721
PreserveSelectionAppearance . . . 723
ResolvedObjectSnap . . . 724
RulerX . . . 725
RulerY . . . 726
ScrollBars . . . 727
SnapToStepStatus . . . 729
SnapToStepX . . . 731
SnapToStepY . . . 733
UndoRecorderOn . . . 735
UserName . . . 736
UserSerialNumber . . . 737
WorkspaceMode . . . 738
XMLDoc . . . 739

Events **740**

Events . . . 741
OLE and paint completion events . . . 743
AutoOSnapType . . . 746

Embedded control events 748
BeforeClick 768
BeforeCopy 770
BeforeMirror 772
BeforeMove 774
BeforeRotate 776
BeforeScale 778
BeforeTransform 780
Change 782
DblClick 783
DrawingRepainted 784
EnterFocus 786
ExitFocus 787
ImportWarningsAvailable 788
MouseIn 789
MouseOut 790
MouseStop 791
NewUserItems 792
OnClick 794
OnKeyDown 795
OnKeyPress 797
OnKeyUp 799
OnMouseDown 801
OnMouseMove 803
OnMouseUp 805
OnPaint 807
OnResize 808
OperationFailed 810
PrintingPage 812
Progress 814

Redo . . . 816
SelectChange . . . 818
SelectedPoints . . . 820
SelectRegionStretch . . . 822
SetExternalLines . . . 825
Undo . . . 828
UserOsnap . . . 830
Canvas / Page workspace . . . 833
ZoomChange . . . 835

Enumerations **837**

Enumerations . . . 838
AvaxACADVersion . . . 840
AvaxAlignConstants . . . 841
AvaxClickType . . . 842
AvaxCommand . . . 843
Triangle and angle commands . . . 854
Center object snap . . . 856
AvaxFillType . . . 857
Native hatch methods and options . . . 858
AvaxImageEffectKind Member Reference . . . 862
AvaxItemType . . . 864
AvaxLanguage . . . 865
AvaxOSnapType . . . 867
AvaxPaperSize . . . 868
AvaxPathNodeMode . . . 870
AvaxPathSegmentKind . . . 871
AvaxPolygonsComp . . . 872
AvaxPolylineType . . . 873
AvaxProgress . . . 874
AvaxProp . . . 875

AvaxScrollBarsStatus . . . 892
AvaxShapeType . . . 893
AvaxUndo . . . 894

Additional topics and reference guides 895

Add_Document . . . 896
Add_ShapeDocument . . . 897
Control API reference . . . 898
Arc commands and segmentation . . . 899
Line, Point and Rectangle commands . . . 901
ByLayer color and line style . . . 903
GetPaperSize . . . 904
Add_TextC . . . 905
Circle and Ellipse commands . . . 906
CommandFilePicker . . . 908
cAvax Member Reference . . . 909
cAvControls Member Reference . . . 910
ConvertToShape . . . 919
Drawing coordinate limits . . . 920
CurrentZoomPercentage . . . 922
AvaxProp Member Reference . . . 923
DefaultExtension . . . 925
AvaxProp Member Reference . . . 926
Display properties . . . 928
GetDocumentClearText . . . 929
Document font height and wrapping width . . . 930
Dimension sizes and wall thickness . . . 931
cAvax Member Reference . . . 933
Drawing modes and settings . . . 934
Editable shape methods . . . 937
Using event handlers . . . 938

Grid and coordinate snap commands . . . 939
Hatch command . . . 941
cAvax Member Reference . . . 944
RenderImagePreviewAsync . . . 947
Item drawing order . . . 948
ItemType and GetAvaxItemType . . . 949
LastImportWarnings . . . 950
DoAvaxAction and public value structures . . . 951
Immediate actions and selection borders . . . 955
cAvax palette reference . . . 958
Resource table capacity . . . 960
Navigation commands and view history . . . 961
cAvax, AvaxCommand, AvaxProp Member Reference . . . 963
OperationFailed . . . 967
cAvax Member Reference . . . 968
Add_PictureFile . . . 970
SetPointSelectionAspectRatio . . . 971
SetPointSelectionPreview . . . 972
Polyline, Polygon and Close commands . . . 973
Property geometry layouts . . . 975
ResolvedObjectSnap . . . 976
Roof command . . . 977
Runtime and OLE methods . . . 979
cAvax, AvaxCommand Member Reference . . . 980
Selected dimension command . . . 982
Selected vertex curves, centers and labels . . . 984
PreserveSelectionAppearance . . . 991
Selection commands and type filters . . . 992
AvaxProp Member Reference . . . 994
Idle selection and spatial queries . . . 996

Precise stroke width 998

CreateSVGHatchPattern 999

Three-point dimension command 1000

AvaxProp Member Reference 1001

Triangle area annotations 1003

Drawing updates and refresh 1007

avVertex.cVertex Member Reference 1008

Vertex dimension commands 1010

DoVertex dimension job 1015

avVertex.cVertex Member Reference 1018

Vertex area and perimeter commands 1020

avVertex.cVertex Member Reference 1023

avVertex.cVertex Member Reference 1025

avVertex.cVertex Member Reference 1027

Vertex triangle command 1029

avVertex.cVertex Member Reference 1031

cAvax Member Reference 1033

Embedded-control refresh and enumeration 1034

FitDrawingToWorkspacePage 1035

Introduction and setup

Use the linked topic list in Contents or the PDF bookmarks to locate a member or guide.

Introduction

AVAX Vector .NET dll v.3.0

https://gardos-software.com/
© 1993-2026 Gardos Software LTD

Overview:

AVAX Vector for .NET is a powerful component that provides over 500 events, methods, properties, ready to use commands and parameters to view, create, edit, print and manage 2D vector drawings. It offers developers a full control of the end user's interactive drafting process.

Features:

1. Standard geometric drawing items (Line, Point, Polyline, Arc, Circle, Ellipse, Polygon, Bezier, etc).
2. Text with Windows based fonts.
3. Polygon shapes filled with any kind of patterns (AutoCAD .PAT files, API solid or pattern, Bitmaps, animations etc). Shapes have z order.
4. Blocks creation and insertion.
5. Color Palette, Layers, Width, Font table, Line and Point types.
6. Zoom Window, Extends, Dynamic, Pan, All and Previous.
7. Select (Window, In Window, All, Select Add-Sub).
8. Grid and Snap.
9. Every drawing object has Handle, KeyName and its own Properties (string array).
10. Copy, Array, Move, Scale, Rotate, Mirror, Transform etc.
11. Interactive graphics by using a Selection Region.
12. Snap to end line, nearest, middle, intersection, perpendicular etc.
13. Trim, Fillet, Extend, Offset and Hatch with lines.
14. Dimensions and Angles.
15. Insertion bitmaps and files (BMP, JPG, ICO, PCX, WMF, SHP, TXT, PLT, RTF).
16. Insertion drawings (AutoCAD® DXF).
17. Export to DXF, SVG, PLT, PIC, BMP, WMF, JPG, PDF, XML and RTF
18. Clipboard Copy, Cut and Paste.
19. Measure distance and area.
20. Print fit to page on multiple pages.
21. Print to any hDC, Print Preview
22. Undo/Redo command in every action (3000 Undo /1 Redo).
23. 230 Events, Methods and Properties with their source code examples.
24. 165 ready to use drawing commands and 110 drawing parameters.
25. Events like BeforeClick, SelectChange, BeforeCopy, BeforeMove, ZoomChange etc.
26. Horizontal and/or vertical scroll bars and rulers.
27. External files to customize Messages, Line Types, Font Table, Color Palette etc.

Getting started with AVAX Vector .NET v3

Developer manual home

This edition targets .NET Framework 4.8 and Windows Forms, with x86 and x64 controls on Windows x64. Reference Gardos-Software.Avax.x86.v3.dll for x86 applications or Gardos-Software.Avax.x64.v3.dll for x64 applications. Deploy the supporting managed and matching native runtime files supplied with that DLL build alongside the application.

Include FFmpeg-Avax.TrueSpeech.dll with the runtime files for TrueSpeech AVI soundtracks. This separate, replaceable library is distributed under LGPL 2.1 or later. Its license, notice, build instructions and complete matching source archive are supplied in Licenses\FFmpeg.TrueSpeech. Keep those files with redistributed copies. The AVAX control's own license is separate.

Initialize and draw

Place an Avax.cAvax control named cad on your form. Start the control before calling drawing methods. End it when the owning form closes.

```
Imports Avax

Private Sub Form1_Load(sender As Object, e As EventArgs) Handles MyBase.Load
    If Not cad.StartAvax() Then
        Throw New InvalidOperationException("The drawing could not be initialized.")
    End If
    Dim lineHandle As Integer = cad.Add_Line(0, 0, 0, 100, 50, 0)
    cad.Command = AvaxCommand.AutoLimits_c
End Sub

Private Sub Form1_FormClosed(sender As Object, e As FormClosedEventArgs) Handles MyBase.FormClosed
    cad.EndAvax()
End Sub
```

Porting R2 source

Geometry formerly declared Single is generally Double in the v3 API. Former Short parameters are generally Integer. Update ByRef variables and coordinate arrays to match the specific current declaration.

Use the API reference to check overloads and optional defaults. Retain one-based logical arrays where the member requires them; slot zero is often reserved. Work with the control on its owning UI thread.

Moving from v2 to v3 explains the numeric changes in detail. Using the VB.NET examples explains the form controls and supporting files required by the examples.

Find a member

Browse methods, properties, events or enumerations. Each member topic describes its purpose and provides the current declaration, setup instructions and its VB.NET example on the same page.

Moving from v2 to v3

API reference

AVAX Vector .NET v3 is a Windows Forms control for .NET Framework 4.8 applications targeting x86 or x64 on Windows x64. Reference the matching `Gardos-Software.Avax.x86.v3.dll` or `Gardos-Software.Avax.x64.v3.dll` and use the `Avax` namespace. Deploy its matching dependencies and native runtime folder. Recompile your application against v3; the assembly identity and several numeric signatures differ from v2.

New and improved in v3

- ByLayer color and line style, with independent object settings, editable layer appearance, Undo/Redo and retained inheritance in native/legacy drawings and DXF/DWG exchange.
- Center snap for circles, ellipses and circular arcs, including block contents and Auto OSnap. Center lookup is cached.
- Canvas / Page workspace, drawing units, paper size/orientation, print scale, page position and margins. Native AVAX files retain these settings. Fit drawing to paper adjusts paper placement and scale.
- Native hatches with editable boundaries, associative updates, PAT/user/solid/gradient fills, islands and transparency.
- Native dimensions: linear, aligned, angular, radial and diametric objects, styles, measurement formatting and editable text positions.
- Editable shape parameters, closure, stroke width and drawing order.
- Expanded resource tables for layers, line styles, colors and fonts, with Integer indices in the native format.
- Faster large-drawing display and picking, cached geometry and lighter transform previews. Curve/text display detail adapts to zoom without replacing stored geometry.
- Additional DWG import/export routes without the old ODX.dll dependency, plus improvements to DXF, SVG, RTF, text and polyline interoperability. Supported entities and format limits still apply.
- Native .avax files retain v3 data. Legacy AVX output converts new dimensions and hatches to old object graphics; it does not retain their new associations.
- Updated x86/x64 SDK packages and matching runtime files. Activate each development computer and retain the Unlock call in your customer application.

New control properties and event

These are direct cAvax properties. They are separate from the existing AvaxProp enumeration. Their individual topics are in the Properties section of the help tree.

Area	Properties
Workspace and units	WorkspaceMode, DrawingUnitMM, HasWorkspaceSettings
Paper, placement and scale	PageNumber, PageTitle, PageScale, PagePaperSize, PageOrientation, PageCustomWidthMM, PageCustomHeightMM, PageOriginX, PageOriginY, PageMarginMM, PageMarginsVisible, PageShadowVisible, PageWidthMM, PageHeightMM, PagePrintUnitMM
Display and files	RulerX, RulerY, OriginVisible, PreserveSelectionAppearance, DefaultExtension
Import diagnostics	LastImportWarnings

LastImportWarnings and ImportWarningsAvailable let your application report DXF/DWG objects that were omitted or simplified.

WorkspaceChanged is raised for workspace-setting changes. Read the event topic for its declaration and an example.

Default drawing extension and legacy files

The default extension for new drawings is now `.avax` (`DefaultExtension = "avax"`). Version 3 continues to read older `.avx` drawings; you do not need to convert them before opening them.

Existing drawings retain their current file format. Use Save As with `.avax` to save an older drawing in the native v3 format and retain v3 features such as editable dimensions and hatches, drawing units and page settings.

Numeric types

v2	v3	What to change
Single drawing coordinates, distances and angles	Double	Change variable declarations, method arguments and result receivers to the type in the v3 declaration.
Single() and Single(,) coordinate arrays	Double() and Double(,)	Change the element type. An existing Single array cannot be cast to a Double array.
Short indices and many Short() arrays	Integer and Integer()	Use the exact declared v3 type, especially for ByRef arguments and event fields.
Integer entity handles	Integer	These are drawing object identifiers, not native Windows handles.
Byte flags, Boolean values and enums	The declared type	Do not replace every numeric type. For example, a fill-type argument declared Byte still requires Byte.
SingleToLong / LongToSingle names	Names retained	The drawing-space arguments are now Double; the pixel coordinates remain Integer.

Updating existing VB.NET declarations

Keep the purpose and calling sequence of the old example. Change the variable types used by its v3 declaration, including the element type of arrays and variables receiving event data. Each affected control member has a **Numeric changes from v2** table followed by its complete updated example.

Original v2 code	Corresponding v3 code
`Dim x As Single = 100.125`	`Dim x As Double = 100.125`
`Dim coordinates(6) As Single`	`Dim coordinates(6) As Double`
`Dim layerNumber As Short = 2`	`Dim layerNumber As Integer = 2`
`Dim radius As Single = cAvax1.PenPointRadius`	`Dim radius As Double = cAvax1.PenPointRadius`
`Dim mouseButton As Short = e.Button` in a BeforeClick handler	`Dim mouseButton As Integer = e.Button` The BeforeClick handler still receives BeforeClickEventArgs.

Use `CDbl` when an explicit conversion to a drawing-coordinate value is needed, and `CInt` for a promoted Integer argument. Retaining `CSng` or a Single variable rounds the value before the control receives it; converting that rounded value to Double does not restore the lost precision.

A line using Double coordinates

Use separate variables for input arguments that the API declares ByRef. Start the control before this code.

```
Dim x1 As Double = 1000000000.125
Dim y1 As Double = 20.25
Dim z1 As Double = 0
Dim x2 As Double = x1 + 10.5
Dim y2 As Double = 30.75
Dim z2 As Double = 0
Dim lineHandle As Integer = cAvax1.Add_Line(x1, y1, z1, x2, y2, z2)
If lineHandle = 0 Then
    Throw New InvalidOperationException("The line could not be added.")
End If
```

ByRef arguments and Object results

A ByRef parameter can return a changed value. Passing a variable of a different type may require a conversion and copy-back, which can lose precision. Declare the exact parameter type instead. Where the signature says Object, receive the value as Object and then cast to the documented result type.

```
Dim ids() As Integer = Nothing
Dim count As Integer = cAvax1.GetAllHandlesArr(ids)
```

```
For i As Integer = 1 To count
    Dim entityHandle As Integer = ids(i)
    ' Use entityHandle with the drawing API.
Next
```

Many AVAX arrays reserve index 0 and use entries 1 through the returned count. Preserve the member's documented indexing. In VB.NET, `Dim points(3) As Double` creates four elements, not three.

Windows handles and helper cleanup

Keep the original Integer arguments for drawing DCs and window-handle queries. Windows USER/GDI handles have 32 significant bits even in an x64 process. When converting a handle returned by Graphics.GetHdc, preserve those bits, including the sign bit; a checked conversion of a zero-extended value can overflow. This rule applies to Windows handles, not memory addresses or COM pointers. See Microsoft's Windows handle interoperability guidance.

The original hwndPtr property remains available. Internal pointer overloads are not part of the developer API. A borrowed device context still belongs to its caller.

Events and drawing lifetime

Keep the R2 `StartAvax`/`EndAvax` lifecycle. Update event argument declarations to the v3 types shown in each topic. Handle control events on the owning UI thread. See Using event handlers and Using the VB.NET examples.

Using the VB.NET examples

API reference

The examples follow the original AVAX developer manual and use the v3 numeric declarations. Each method, property or event contains its example and setup instructions in the Example section on the same page. Copy those procedures into a Windows Forms form class, rather than into a standalone module.

Project setup

1. Create a Visual Basic Windows Forms project targeting .NET Framework 4.8 and x64.
2. Reference `Gardos-Software.Avax.x64.v3.dll`. Deploy its supplied managed and native dependencies with the application.
3. Add an `Avax.cAvax` control named `cAvax1` to the form.
4. Add the other Windows Forms controls referenced by the example. Designer-generated fields used in Handles clauses must be WithEvents.
5. Add `Imports Avax`. The original form examples use the Visual Basic defaults, including Option Strict Off. With Option Strict On, add explicit conversions for Object results and numeric input from controls.

Name in an example	Windows Forms control or component
Command1, Command2, Command3, Button1, Button2	Button
Combo1, Combo2	ComboBox
Check1	CheckBox
Text1, Text2	TextBox
List1	ListBox
Label1, Label2	Label
Picture1	PictureBox
ToolTip1	ToolTip

Each example lists its additional controls in the Setup section. XML examples also require a reference to the .NET Framework `System.Xml` assembly. Reference-type output variables are initialized to Nothing; the called method fills or replaces them. String accumulators start with String.Empty.

Start and stop the control

```
Private Sub Form1_Load(sender As Object, e As EventArgs) Handles MyBase.Load
    If Not cAvax1.StartAvax() Then
        Throw New InvalidOperationException("The drawing could not be initialized.")
    End If
End Sub

Private Sub Form1_FormClosed(sender As Object, e As FormClosedEventArgs) Handles Me.FormClosed
    cAvax1.EndAvax()
End Sub
```

Most examples already include these two handlers. When combining examples into one form, retain a single initialization and shutdown pair and give the other procedures unique names. Drawing methods belong after successful initialization.

Files and interactive commands

Import examples need an input file in the format named by the example. Replace the sample path with an existing file in your application. Export and save examples need a writable destination; keep source files and output files separate. Several original samples use a relative path one level above the executable folder. Adjust it for your own deployment.

The external-lines example uses the supplied v3 `map-double.dat` with Double coordinates beside the executable. The PLT import example uses `house.plt`. Font, line style, pattern and drawing-library examples need the corresponding resource files. An example that sets an interactive Command waits for the user to supply the requested points or selection. Printing and clipboard examples use

those system services when run.

Object arrays with Option Strict On

For example, GetProperties returns geometry through an Object array. Use an explicit cast before editing a line's coordinates:

```
Dim xyz() As Double = DirectCast(vData(itemIndex), Double())
xyz(1) += 5.0
vData(itemIndex) = xyz
```

See Property geometry layouts for the indices used by each object type, and Moving from v2 to v3 for ByRef and numeric type changes.

Find an example in the corresponding method, property or event.

Methods

Use the linked topic list in Contents or the PDF bookmarks to locate a member or guide.

Methods

Select a topic for its purpose, declaration, argument descriptions and example.

- AbortPrinting
- ActionInProgress
- Add_AllDimensions
- Add_AngularDim
- Add_Arc
- Add_AVI
- Add_AVX
- Add_Bezier
- Add_Block
- Add_BlockLines
- Add_CBSpline
- Add_Circle
- Add_CSpline1
- Add_CSpline2
- Add_CurveArc
- Add_Data
- Add_Dim
- Add_DWG
- Add_DXF
- Add_Ellipse
- Add_ExternalLines
- Add_Line
- Add_LinearHatch
- Add_OBSpline
- Add_PCX
- Add_Picture
- Add_PLT
- Add_Point
- Add_PolyLine
- Add_Rectangle
- Add_RTF
- Add_Shape
- Add_ShapeArc
- Add_ShapeCircle
- Add_ShapeEllipse
- Add_ShapeLine
- Add_ShapePoint

- Add_ShapePolyLine
- Add_ShapeRectangle
- Add_ShapeRegularPolygon
- Add_ShapeText
- Add_SHP
- Add_SVG
- Add_Text
- Add_TextFile
- Add_WMF
- Add_XML
- AlignItem
- CalcBoundsDrawing
- ChangeColor
- ChangeElevation3D
- ChangeFontNumber
- ChangeHeight3D
- ChangeLayer
- ChangeLineStyle
- ChangePointStyle
- ChangeProperties
- ChangeShapeFill
- ChangeShapeText
- ChangeShapezOrder
- ChangeText
- ChangeWidth
- ClearUndoRedoMemory
- ComparePolygons
- Copy_Avax
- CopyInside
- CopyItems
- CopyOutside
- CopyToClipboard
- CreateACADHatchPattern
- CreateACADLineStyleString
- CreateBlock
- CreateExVectorHatchPattern
- CreatePatternFlags
- CreateVectorHatchPattern
- CurrentRedo
- CurrentUndo

- CurrentXY
- DeleteAllItems
- DeleteAvaxFile
- DeleteItems
- DestroyDrawing
- DirsGet
- DirsSet
- DoAvaxAction
- Draw_Line
- EndAvax
- FindHandlesArrByKey
- FitToAvax
- GetAllHandlesArr
- GetAllNodes
- GetArea
- GetAttributes
- GetAvaxEvents
- GetAvaxProperty
- GetAvaxStatus
- GetBorder
- GetBoundsAvax
- GetBoundsDrawing
- GetBoundsSelectBox
- GetBoundsSelectedItems
- GetBoundsZoom
- GetCurveLines
- GetEllipseGeometry
- GetFontName
- GetFonts
- GetHandleKey
- GetHandlesArrByKeyArr
- GetHandlesByKey
- GetHandlesUnderArea
- GetHandlesUnderSelectCursor
- GetKeyArrByHandlesArr
- GetLayers
- GetLineStartPoint
- GetLineStyle
- GetLineStyleNames
- GetLineStyles

- GetOSnapStatus
- GetPalColor
- GetPalette
- GetPerimeter
- GetPictureSize
- GetProperties
- GetSelectedHandlesArr
- GetSelectedPoints
- GetSelectRegionBorders
- GetShapeFill
- GetShapePicturePixels
- GetShapeText
- GetShapezOrder
- GetText
- GetTextAttributes
- GetTextFontNumber
- GetTextGeometry
- GetTextVectors
- isSelected
- IsSelectedItem
- LineStyleIndexByName
- LoadACADLineStyles
- LoadAvaxPropertiesFromXML
- LoadFonts
- LoadLineStyles
- LoadPalette
- LongToSingle
- MaxItems
- MaxSelectedItems
- MirrorItems
- MoveItems
- MoveNode
- NearestPaletteIndexColor
- Paste_Avax
- PasteFromClipboard
- PrintScreenToBMP
- PrintToBMP
- PrintToDWG
- PrintToDXF
- PrintToGIF

- PrintTohDC
- PrintToJPG
- PrintToPIC
- PrintToPLT
- PrintToPNG
- PrintToPrinter
- PrintToRTF
- PrintToSVG
- PrintToSWF
- PrintToTIF
- PrintToWMF
- PrintToXML
- RedrawItems
- RotateItems
- SaveAs
- SaveAvaxPropertiesToXML
- SaveFonts
- SaveLineStyles
- SavePalette
- ScaleItems
- SelectItems
- SelectPoints
- SetAvaxProperty
- SetBoundsAvax
- SetBoundsDrawing
- SetBoundsStretch
- SetBoundsZoom
- SetCoordinate
- SetFontName
- SetFonts
- SetHandleKey
- SetLayers
- SetLineStyle
- SetLineStyles
- SetPalColor
- SetPalette
- SetProperties
- SetSelectFilter
- SetSelectRegionBoundingBox
- SetSVGPalette

- SingleToLong
- SnapToStepSet
- StartAvax
- StretchItems
- TransformItems
- UnitX
- UnitY
- ZoomIn
- ZoomOut

Additional members

- AvaxTextStyle.Clone
- avVertex.cVertex.DoVertex
- Add_Dim_Aligned
- Add_Dim_Angular
- Add_Dim_Diametric
- Add_Dim_Linear
- Add_Dim_Radial
- Add_Document
- Add_Hatch
- Add_PictureFile
- Add_ShapeDocument
- Add_TextC
- AddCompoundPath
- AddEditableShape
- AddParametricShape
- AvaxControlGetAllControls
- AvaxControlGetData
- AvaxControlGetShape
- AvaxControlRefresh
- AvaxControlRefreshAll
- AvaxControlSetData
- BeginCompoundPath
- BeginCurveNodeEditing
- BeginUpdate
- CenterWorkspacePage
- ChangeCompoundPath
- ChangeDocumentFontSize
- ChangeDocumentWidth
- ChangeItemOrder
- ChangePathSegmentKind

- ChangePathSegmentParameter
- ChangeShapeClosed
- ChangeShapeControlPoint
- ChangeShapeDocumentFontSize
- ChangeShapeDocumentWidth
- ChangeShapeParameter
- ChangeStrokeWidth
- CombineCompoundPaths
- ConvertShapeToCurves
- ConvertToShape
- CreateSVGHatchPattern
- CurrentZoomPercentage
- EndUpdate
- ExportNativeDrawing
- FinishCompoundPath
- FitDrawingToWorkspacePage
- Get_Dimension
- Get_DimensionDefaults
- Get_DimensionMeasurement
- Get_DimensionStyle
- Get_Hatch
- Get_HatchArea
- Get_HatchDefaults
- GetAvaxItemType
- GetCompoundPath
- GetDocumentClearText
- GetDocumentFontSize
- GetDocumentWidth
- GetImageEffectDefinitions
- GetImageEffects
- GetImageSource
- GetItemByLayer
- GetItemZOrder
- GetLayerColor
- GetLayerLineStyle
- GetPaperSize
- GetParametricShapeKind
- GetParametricShapes
- GetPathSegmentParameters
- GetShapeClosed

- GetShapeControlPoints
- GetShapeDocumentText
- GetShapeParameters
- GetStrokeWidth
- GetTextStyle
- GetWorkspacePageBounds
- InitProgramID
- IsImageShape
- IsTextShape
- ItemType
- Load_HatchPattern
- OleDrag
- ParametricShapeDefinitions
- PrintToAPL
- PrintWorkspaceToImage
- PrintWorkspaceToPDF
- PrintWorkspaceToPrinter
- PrintWorkspaceToRTF
- PrintWorkspaceToSVG
- PrintWorkspaceToWMF
- Refresh
- Regen
- RegenControlsZoomBitmap
- RenderImagePreview
- RenderImagePreviewAsync
- RenderParametricShapeIcon
- RenderWorkspacePage
- Reset_DimensionTextPosition
- ResetImageEffects
- ReverseCompoundPath
- Set_Dimension
- Set_DimensionStyle
- Set_DimensionTextPosition
- Set_Hatch
- Set_HatchBoundaries
- SetEditablePointSelectionPreview
- SetImageEffects
- SetItemByLayer
- SetItemsByLayer
- SetLayerColor

- SetLayerLineStyle
- SetPathNodeMode
- SetPointSelectionAspectRatio
- SetPointSelectionPreview
- SetTextStyle
- SetTextStyles
- SplitPathSegment
- Unlock
- ZoomWorkspacePage
- modbutton.DrawAvaxButton

AbortPrinting

cAvax method

Purpose

Requests cancellation of the current print operation.

VB.NET declaration

```
Function AbortPrinting() As Boolean
```

Return value

Boolean - True on success, false on error.

Example

Setup

Add an Avax.cAvax Windows Forms control named `cAvax1`. Paste the procedures into the form class. Example setup and supporting files explains references, initialization and file paths.

Add these Windows Forms controls or components to the form. Fields referenced by Handles clauses must be declared WithEvents, as the Visual Studio designer normally does.

Name	Type
Command1	Button

VB.NET example

```
Private Sub cAvax1_PrintingPage1(ByVal Sender As Object, ByVal e As Avax.cAvax.PrintingPageEventArgs)
    Handles cAvax1.PrintingPage
    Dim sMsg As String = String.Empty
    Dim Answer As Boolean
    sMsg = "Avax is printing page [" & Format(e.lNowPage) & "] from "
    sMsg = sMsg & "[" & Format(e.lMaxPage) & "] pages." & vbCrLf
    sMsg = sMsg & "Do you want stop printing?"
    If MsgBox(sMsg, MsgBoxStyle.YesNo Or MsgBoxStyle.Question, "") = MsgBoxResult.Yes Then
        Answer = cAvax1.AbortPrinting()
    End If
End Sub
Private Sub Form1_Load(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs) Handles
    MyBase.Load
    cAvax1.StartAvax()
    Call cAvax1.SetAvaxProperty(Avax.AvaxProp.PrintUnit, MM, 0, 20)
    Call cAvax1.Add_Line(0, 0, 0, 12, 0, 0)
End Sub
Private Sub Form1_FormClosed(ByVal eventSender As System.Object, ByVal eventArgs As
    System.Windows.Forms.FormClosedEventArgs) Handles Me.FormClosed
    cAvax1.EndAvax()
End Sub
Private Sub Command1_Click(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs)
    Handles Command1.Click
    cAvax1.Command = Avax.AvaxCommand.Print_c
End Sub
```

ActionInProgress

cAvax method

Purpose

Reports whether the control is currently carrying out an interactive drawing or editing action.

VB.NET declaration

```
Function ActionInProgress() As Boolean
```

Return value

Boolean - True on success, false on error.

Remarks

Reports whether the active control is carrying out an interactive operation. Returns False for an unstarted, disabled, ended or disposed control and for idle point-selection modes. Point requests, accepted drawing/editing tools, selection windows, active grip edits and managed Pan/Dynamic report activity. A raw zoomon value of zero does not imply that Pan/Dynamic is idle. New Text, import/export/block file selection and Wall/Polygon/Array/Offset parameter entry precede their active geometry phase: the command has not yet accepted construction or a placement/base point, and submitted construction points are rejected. File-only operations do not create a drawing action. Grid/Snap configuration retains the current drafting operation. Callback-started or nested accepted commands immediately own the query state; returning or throwing from an older setup request does not hide the newer operation. Use GetAvaxStatus to inspect individual mode fields or snapshot availability. This read-only query is separate from whether input or event delivery is paused.

Example

Setup

Add an Avax.cAvax Windows Forms control named `cAvax1`. Paste the procedures into the form class. Example setup and supporting files explains references, initialization and file paths.

Add these Windows Forms controls or components to the form. Fields referenced by Handles clauses must be declared WithEvents, as the Visual Studio designer normally does.

Name	Type
Command1	Button
Label1	Label

VB.NET example

```
Private Sub cAvax1_OnMouseMove(ByVal Sender As Object, ByVal e As Avax.cAvax.MouseMoveEventArgs) Handles
    cAvax1.OnMouseMove
    If cAvax1.ActionInProgress() = True Then
        Label1.Text = "Action in progress..."
    Else
        Label1.Text = "No action in progress..."
    End If
End Sub
Private Sub Form1_Load(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs) Handles
    MyBase.Load
    cAvax1.StartAvax()
End Sub
Private Sub Form1_FormClosed(ByVal eventSender As System.Object, ByVal eventArgs As
    System.Windows.Forms.FormClosedEventArgs) Handles Me.FormClosed
    cAvax1.EndAvax()
End Sub
Private Sub Command1_Click(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs)
    Handles Command1.Click
    cAvax1.Command = Avax.AvaxCommand.Zoom_c
```

```
End Sub
```

Add_AllDimensions

cAvax method

Purpose

Adds X and Y dimension chains from sorted drawing-node coordinates, omitting intervals at or below sSmallestDim.

VB.NET declarations

```
Function Add_AllDimensions(ByRef OffsetD As Double, ByRef sSmallestDim As Double) As Boolean

Function Add_AllDimensions(ByRef OffsetD As Double, ByRef sSmallestDim As Double, ByRef vItemProperties As
    Object, ByRef vRetHandles As Object, ByRef iClr As Integer, ByRef iWidth As Integer, ByRef iStyle As
    Integer, ByRef iLayer As Integer) As Boolean
```

Parameters

Name	Type and passing	Meaning
OffsetD	ByRef OffsetD As Double	Double offset from the drawing border. Nonpositive values use 1; normal success copies the effective value back to this ByRef argument.
sSmallestDim	ByRef sSmallestDim As Double	Double minimum coordinate interval. Adjacent sorted node coordinates whose exact difference is at or below this value are suppressed; negative thresholds act as zero. No extra world-unit tolerance is applied.
vItemProperties	ByRef vItemProperties As Object	A zero-based array with logical property strings starting at index 1. String() and string-valued Object() arrays are copied; Nothing is allowed. Non-string entries are rejected before editing.
vRetHandles	ByRef vRetHandles As Object	On normal success, an Integer() stored in Object with index 0 reserved and all handles from this operation. Active preparation/empty-result failure returns {0}. Early stopped/disabled/disposed rejection leaves it unchanged. A thrown host callback exception does not provide normal ByRef copy-out.
iClr	ByRef iClr As Integer	Integer color index; -1 uses PenColor. The effective value is copied back after normal success.
iWidth	ByRef iWidth As Integer	Integer line width; -1 uses PenWidth, zero normalizes to 1. The effective value is copied back after normal success.
iStyle	ByRef iStyle As Integer	Integer line-style index; -1 uses PenLineStyle. The effective value is copied back after normal success.
iLayer	ByRef iLayer As Integer	Integer layer index; -1 uses PenLayer. Values are not restricted to the former Integer range. The effective value is copied back after normal success.

Use the declaration for the overload you call. ByRef arguments can receive outputs; declare variables and arrays with the exact indicated type. Optional defaults are shown in the declaration.

Return value

Boolean - True on success, false on error.

Remarks

The method uses the same logical public nodes as GetAllNodes, from visible drawing objects. Hidden layers are excluded; visible locked and nonprinting layers remain included. Text, blocks and external references supply their logical envelope corners. Ellipses supply their envelope, center and any open-arc endpoints. Live external references are refreshed once before node collection. Provider exceptions retain their original identity; a replaced, restarted, stopped or disabled control cannot receive stale output.

X and Y coordinates are sorted independently, with coincident coordinates combined. Adjacent intervals at or below sSmallestDim are omitted; a negative minimum acts as zero. A chain with more than one interval also receives an overall dimension. Horizontal chains are below the visible drawing border; vertical chains are to its right. OffsetD is an unscaled drawing distance. A nonpositive

OffsetD becomes 1 and is returned through ByRef on successful completion. The complete node set and result must be finite and representable. No geometric nodes or no usable interval returns False without a partial dimension batch.

The result consists of ordinary drawing objects for the dimension lines, marks and label. The full overload returns all their handles as a one-based Integer() stored in Object. Dimension properties selected through AvaxProp control the label format, size, marks and extension lines. Coordinates and distances are Double; pen indices are Integer. Use -1 to inherit a current pen attribute. Resolved ByRef pen values are returned on successful completion. The control must be started and enabled. PauseAvax and LockEdit do not prevent these direct additions. The complete result is prepared before insertion and committed as one Undo operation, with Change and no NewUserItems notification. Invalid or degenerate geometry returns False without inserting a partial dimension. An active full call initializes vRetHandles to {0}; an unavailable control leaves it unchanged. Application event-handler exceptions propagate after committed creation.

Numeric changes from v2

Update the following variables, arrays or event-data receivers when porting the corresponding v2 call. The declarations above show the current overloads.

Parameter or value	v2 type	v3 type
OffsetD	Single	Double
sSmallestDim	Single	Double
iClr	Short	Integer
iWidth	Short	Integer
iStyle	Short	Integer
iLayer	Short	Integer

For ByRef parameters, use a variable with the exact v3 type. Preserve array dimensions and the documented indexing. Moving from v2 to v3 explains conversion and legacy file-storage rules.

Example

Setup

Add an Avax.cAvax Windows Forms control named `cAvax1`. Paste the procedures into the form class. Example setup and supporting files explains references, initialization and file paths.

Add these Windows Forms controls or components to the form. Fields referenced by Handles clauses must be declared WithEvents, as the Visual Studio designer normally does.

Name	Type
Command1	Button

VB.NET example

```
Private Sub Form1_Load(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs) Handles
    MyBase.Load
    cAvax1.StartAvax()
    Call cAvax1.SetAvaxProperty(Avax.AvaxProp.DimTextScale, 4.0!)
    Call cAvax1.Add_Line(0, 0, 0, 10, 10, 0)
    Call cAvax1.Add_Line(0, 10, 0, 15, 0, 0)
End Sub
Private Sub Form1_FormClosed(ByVal eventSender As System.Object, ByVal eventArgs As
    System.Windows.Forms.FormClosedEventArgs) Handles Me.FormClosed
    cAvax1.EndAvax()
End Sub
Private Sub Command1_Click(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs)
    Handles Command1.Click
    Static once As Integer
    Dim OffsetD As Double
    Dim sSmallestDim As Double
    Dim vItemProperties As Object = Nothing
    Dim vRetHandles As Object = Nothing
```

```
    Dim iClr As Integer
    Dim iWidth As Integer
    Dim iStyle As Integer
    Dim iLayer As Integer
    Dim lRetHandles() As Integer = Nothing
    If once = 0 Then
        once = 1
        OffsetD = 1
        If cAvax1.Add_AllDimensions(OffsetD, sSmallestDim, vItemProperties, vRetHandles, iClr, iWidth,
    iStyle, iLayer) = True Then
            lRetHandles = vRetHandles
        End If
    End If
End Sub
```

Add_AngularDim

cAvax method

Purpose

Adds an angular dimension between two intersecting lines. The placement point selects its sector and arc radius.

VB.NET declarations

```
Function Add_AngularDim(ByRef ax1 As Double, ByRef ay1 As Double, ByRef ax2 As Double, ByRef ay2 As
    Double, ByRef bx1 As Double, ByRef by1 As Double, ByRef bx2 As Double, ByRef by2 As Double, ByRef xDim As
    Double, ByRef yDim As Double) As Boolean

Function Add_AngularDim(ByRef ax1 As Double, ByRef ay1 As Double, ByRef ax2 As Double, ByRef ay2 As
    Double, ByRef bx1 As Double, ByRef by1 As Double, ByRef bx2 As Double, ByRef by2 As Double, ByRef xDim As
    Double, ByRef yDim As Double, ByRef vItemProperties As Object, ByRef vRetHandles As Object, ByRef iClr As
    Integer, ByRef iWidth As Integer, ByRef iStyle As Integer, ByRef iLayer As Integer) As Boolean
```

Parameters

Name	Type and passing	Meaning
ax1	ByRef ax1 As Double	Is the x coordinate of the first point of the first line of the angle
ay1	ByRef ay1 As Double	Is the y coordinate of the first point of the first line of the angle
ax2	ByRef ax2 As Double	Is the x coordinate of the second point of the first line of the angle
ay2	ByRef ay2 As Double	Is the y coordinate of the second point of the first line of the angle
bx1	ByRef bx1 As Double	Is the x coordinate of the first point of the second line of the angle
by1	ByRef by1 As Double	Is the y coordinate of the first point of the second line of the angle
bx2	ByRef bx2 As Double	Is the x coordinate of the second point of the second line of the angle
by2	ByRef by2 As Double	Is the y coordinate of the second point of the second line of the angle
xDim	ByRef xDim As Double	Is the x coordinate of a point that exists in the angle
yDim	ByRef yDim As Double	Is the y coordinate of a point that exists in the angle
vItemProperties	ByRef vItemProperties As Object	A zero-based array with logical property strings starting at index 1. String() and string-valued Object() arrays are copied; Nothing is allowed. Non-string entries are rejected before editing.
vRetHandles	ByRef vRetHandles As Object	On normal success, an Integer() stored in Object with index 0 reserved and all handles from this operation. Active preparation/empty-result failure returns {0}. Early stopped/disabled/disposed rejection leaves it unchanged. A thrown host callback exception does not provide normal ByRef copy-out.
iClr	ByRef iClr As Integer	Integer color index; -1 uses PenColor. The effective value is copied back after normal success.
iWidth	ByRef iWidth As Integer	Integer line width; -1 uses PenWidth, zero normalizes to 1. The effective value is copied back after normal success.
iStyle	ByRef iStyle As Integer	Integer line-style index; -1 uses PenLineStyle. The effective value is copied back after normal success.
iLayer	ByRef iLayer As Integer	Integer layer index; -1 uses PenLayer. Values are not restricted to the former Integer range. The effective value is copied back after normal success.

Use the declaration for the overload you call. ByRef arguments can receive outputs; declare variables and arrays with the exact indicated type. Optional defaults are shown in the declaration.

Return value

Boolean - True on success, false on error.

Remarks

The two supplied lines define an intersection and angle sectors. The point (xDim, yDim) selects the sector; its distance from the intersection sets the arc radius. Parallel or degenerate lines produce no dimension. The arc is stored as an analytic Curve_i.

The result consists of ordinary drawing objects for the dimension lines, marks and label. The full overload returns all their handles as a one-based Integer() stored in Object. Dimension properties selected through AvaxProp control the label format, size, marks and extension lines. Coordinates and distances are Double; pen indices are Integer. Use -1 to inherit a current pen attribute. Resolved ByRef pen values are returned on successful completion. The control must be started and enabled. PauseAvax and LockEdit do not prevent these direct additions. The complete result is prepared before insertion and committed as one Undo operation, with Change and no NewUserItems notification. Invalid or degenerate geometry returns False without inserting a partial dimension. An active full call initializes vRetHandles to {0}; an unavailable control leaves it unchanged. Application event-handler exceptions propagate after committed creation.

Numeric changes from v2

Update the following variables, arrays or event-data receivers when porting the corresponding v2 call. The declarations above show the current overloads.

Parameter or value	v2 type	v3 type
ax1	Single	Double
ay1	Single	Double
ax2	Single	Double
ay2	Single	Double
bx1	Single	Double
by1	Single	Double
bx2	Single	Double
by2	Single	Double
xDim	Single	Double
yDim	Single	Double
iClr	Short	Integer
iWidth	Short	Integer
iStyle	Short	Integer
iLayer	Short	Integer

For ByRef parameters, use a variable with the exact v3 type. Preserve array dimensions and the documented indexing. Moving from v2 to v3 explains conversion and legacy file-storage rules.

Example

Setup

Add an Avax.cAvax Windows Forms control named `cAvax1`. Paste the procedures into the form class. Example setup and supporting files explains references, initialization and file paths.

Add these Windows Forms controls or components to the form. Fields referenced by Handles clauses must be declared WithEvents, as the Visual Studio designer normally does.

Name	Type
Button1	Button

VB.NET example

```
Private Sub Form1_Load(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs) Handles
    MyBase.Load
    Call cAvax1.StartAvax()
    cAvax1.Add_Line(0, 0, 0, 10, 0, 0)
    cAvax1.Add_Line(0, 0, 0, 10, 10, 0)
End Sub
Private Sub Form1_FormClosed(ByVal eventSender As System.Object, ByVal eventArgs As
    System.Windows.Forms.FormClosedEventArgs) Handles Me.FormClosed
    Call cAvax1.EndAvax()
End Sub
Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
    Button1.Click
    Dim ax1 As Double
    Dim ay1 As Double
    Dim ax2 As Double
    Dim ay2 As Double
    Dim bx1 As Double
    Dim by1 As Double
    Dim bx2 As Double
    Dim by2 As Double
    Dim xDim As Double
    Dim yDim As Double
    Dim vItemProperties As Object = Nothing
    Dim vRetHandles As Object = Nothing
    Dim iClr As Integer
    Dim iWidth As Integer
    Dim iStyle As Integer
    Dim iLayer As Integer
    Dim Answer As Boolean
    Call cAvax1.SetAvaxProperty(Avax.AvaxProp.DimUnitFormat_n, Avax.AvaxUnitFormat.Architectural_u)
    Call cAvax1.SetAvaxProperty(Avax.AvaxProp.DimFractionalTick_n, 32)
    Call cAvax1.SetAvaxProperty(Avax.AvaxProp.DimAngularUnitFormat_n,
    Avax.AvaxAngularUnitFormat.Decimal_Degrees)
    Call cAvax1.SetAvaxProperty(Avax.AvaxProp.DimScale_n, 5)
    ax1 = 0
    ay1 = 0
    ax2 = 10
    ay2 = 0
    bx1 = 0
    by1 = 0
    bx2 = 10
    by2 = 10
    xDim = 5
    yDim = 2
    Answer = cAvax1.Add_AngularDim(ax1, ay1, ax2, ay2, bx1, by1, bx2, by2, xDim, yDim, vItemProperties,
    vRetHandles, iClr, iWidth, iStyle, iLayer)
End Sub
```

Add_Arc

cAvax method

Purpose

Adds a counterclockwise circular arc as a polyline of straight segments. Use Add_CurveArc to retain an analytic arc.

See Arc commands and segmentation for the three arc pick orders, standalone completion, polyline continuation and arc resolution settings.

VB.NET declarations

```
Function Add_Arc(ByRef x1 As Double, ByRef y1 As Double, ByRef z1 As Double, ByRef x2 As Double, ByRef y2
    As Double, ByRef z2 As Double, ByRef xc As Double, ByRef yc As Double, ByRef zc As Double) As Integer

Function Add_Arc(ByRef x1 As Double, ByRef y1 As Double, ByRef z1 As Double, ByRef x2 As Double, ByRef y2
    As Double, ByRef z2 As Double, ByRef xc As Double, ByRef yc As Double, ByRef zc As Double, ByRef
    vItemProperties As Object, ByRef iClr As Integer, ByRef iWidth As Integer, ByRef iStyle As Integer, ByRef
    iLayer As Integer) As Integer
```

Parameters

Name	Type and passing	Meaning
x1	ByRef x1 As Double	Is x coordinate of the arc's start point
y1	ByRef y1 As Double	Is y coordinate of the arc's start point
z1	ByRef z1 As Double	Starting endpoint Z as Double. The segmented arc retains this value and interpolates toward z2.
x2	ByRef x2 As Double	Final endpoint X as Double. Together with y2, determines the counterclockwise end direction; unequal endpoint radii are projected onto the first-radius circle.
y2	ByRef y2 As Double	Final endpoint Y as Double; see x2. Coincident positive endpoint directions create a complete circle.
z2	ByRef z2 As Double	Final endpoint Z as Double. Retained exactly on the last vertex; intermediate Z is interpolated without subtraction overflow.
xc	ByRef xc As Double	Is x coordinate of the arc's circle center
yc	ByRef yc As Double	Is y coordinate of the arc's circle center
zc	ByRef zc As Double	Center Z as Double. Used for the complete-circle special case. Non-coincident segmented arcs use their supplied endpoint elevations.
vItemProperties	ByRef vItemProperties As Object	One-dimensional String() or Object() containing strings/Nothing, with logical properties starting at index 1. The control stores its own copy and retains the original caller reference. Nothing means no properties, except Add_Data which requires an array.
iClr	ByRef iClr As Integer	-1 uses the corresponding current pen setting. Other negative values become 0. Positive values use Integer storage. When declared ByRef, the effective value is written back only after successful validation.
iWidth	ByRef iWidth As Integer	-1 uses the current pen. Nonpositive resolved widths become 1, as in R2. Positive values use Integer storage. When declared ByRef, the effective value is written back only after successful validation.
iStyle	ByRef iStyle As Integer	-1 uses the corresponding current pen setting. Other negative values become 0. Positive values use Integer storage. When declared ByRef, the effective value is written back only after successful validation.
iLayer	ByRef iLayer As Integer	-1 uses the corresponding current pen setting. Other negative values become 0. Positive values use Integer storage. When declared ByRef, the effective value is written back only after successful validation.

Use the declaration for the overload you call. ByRef arguments can receive outputs; declare variables and arrays with the exact indicated type. Optional defaults are shown in the declaration.

Return value

Integer - An Avax handle on success, zero on error.

Remarks

The start point determines the circle radius. The end point supplies a direction from the center; a different end-point radius is projected onto the circle. Start and end elevations are retained, with intermediate Z coordinates interpolated along the sampled arc. Coincident positive endpoint rays produce a complete Ellipse_i circle at the center elevation.

CircleSectors sets full-circle resolution; 0 selects the default 72. Positive ArcSegmentsLength derives a resolution from the circumference and clamps it between MinCircleSectors and MaxCircleSectors. The supported resolution is 3 through 1000000. An arc no wider than one sector uses a quarter-resolution subdivision. CurveArcOn does not change this explicit method. GetPerimeter and GetArea measure the stored chords.

The short overload uses the current pen. The full overload accepts copied String metadata and Integer pen attributes; -1 inherits the current pen, and resolved ByRef attributes are returned on success. A started, enabled control is required. Invalid input returns 0 before changing the drawing, history or caller arguments. Success is one Undo operation with Change and no NewUserItems. Application event-handler exceptions propagate after insertion.

Numeric changes from v2

Update the following variables, arrays or event-data receivers when porting the corresponding v2 call. The declarations above show the current overloads.

Parameter or value	v2 type	v3 type
x1	Single	Double
y1	Single	Double
z1	Single	Double
x2	Single	Double
y2	Single	Double
z2	Single	Double
xc	Single	Double
yc	Single	Double
zc	Single	Double
iClr	Short	Integer
iWidth	Short	Integer
iStyle	Short	Integer
iLayer	Short	Integer

For ByRef parameters, use a variable with the exact v3 type. Preserve array dimensions and the documented indexing. Moving from v2 to v3 explains conversion and legacy file-storage rules.

Example

Setup

Add an Avax.cAvax Windows Forms control named `cAvax1`. Paste the procedures into the form class. Example setup and supporting files explains references, initialization and file paths.

Add these Windows Forms controls or components to the form. Fields referenced by Handles clauses must be declared WithEvents, as the Visual Studio designer normally does.

Name	Type
Command1	Button

VB.NET example

```
Private Sub Form1_Load(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs) Handles
    MyBase.Load
    cAvax1.StartAvax()
End Sub
Private Sub Form1_FormClosed(ByVal eventSender As System.Object, ByVal eventArgs As
    System.Windows.Forms.FormClosedEventArgs) Handles Me.FormClosed
    cAvax1.EndAvax()
End Sub
Private Sub Command1_Click(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs)
    Handles Command1.Click
    Dim x1 As Double
    Dim y1 As Double
    Dim z1 As Double
    Dim x2 As Double
    Dim y2 As Double
    Dim z2 As Double
    Dim xc As Double
    Dim yc As Double
    Dim zc As Double
    Dim vItemProperties As Object = Nothing
    Dim iClr As Integer
    Dim iWidth As Integer
    Dim iStyle As Integer
    Dim iLayer As Integer
    Dim AvaxHandle As Integer
    x1 = 0
    y1 = 0
    z1 = 0
    x2 = 10
    y2 = 10
    z2 = 0
    xc = 5
    yc = 5
    zc = 0
    iWidth = 1
    iClr = 1
    AvaxHandle = cAvax1.Add_Arc(x1, y1, z1, x2, y2, z2, xc, yc, zc, vItemProperties, iClr, iWidth, iStyle,
     iLayer)
End Sub
```

Add_AVI

cAvax method

Purpose

This function inserts an .AVI file into the drawing

VB.NET declarations

```
Function Add_AVI(ByRef sAVIFile As String, ByVal xMin As Double, ByVal yMin As Double, ByVal xMax As
    Double, ByVal yMax As Double, ByRef zOrder As Integer, ByRef vItemProperties As Object, ByRef iLayer As
    Integer) As Integer

Function Add_AVI(ByRef sAVIFile As String, ByVal xMin As Double, ByVal yMin As Double, ByVal xMax As
    Double, ByVal yMax As Double) As Integer
```

Parameters

Name	Type and passing	Meaning
sAVIFile	ByRef sAVIFile As String	Filename of the AVI video to embed. The original file path is not needed after successful embedding.
xMin	ByVal xMin As Double	First X corner of the insertion rectangle, in Double drawing coordinates.
yMin	ByVal yMin As Double	First Y corner of the insertion rectangle, in Double drawing coordinates.
xMax	ByVal xMax As Double	Opposite X corner of the insertion rectangle, in Double drawing coordinates.
yMax	ByVal yMax As Double	Opposite Y corner of the insertion rectangle, in Double drawing coordinates.
zOrder	ByRef zOrder As Integer	Drawing order of the embedded picture or video.
vItemProperties	ByRef vItemProperties As Object	Application-defined item properties, using the legacy one-based String array convention when supplied.
iLayer	ByRef iLayer As Integer	Destination layer. A current-layer sentinel is resolved to the active pen layer and written back by the extended overload.

Use the declaration for the overload you call. ByRef arguments can receive outputs; declare variables and arrays with the exact indicated type. Optional defaults are shown in the declaration.

Return value

Integer - An Avax handle on success, zero on error.

Remarks

Embeds an AVI video in the specified drawing rectangle and returns its handle, or 0 if the control is not ready or the video cannot be decoded. Rectangle coordinates are Double drawing coordinates. The extended overload also accepts z-order, item properties and layer. The drawing retains the video and its soundtrack when saved in supported native AVAX, AVX and XML containers; the source file is no longer required after insertion. Video playback starts once for a new or reopened placement. The first audio stream plays through the default Windows audio output using bundled TrueSpeech support or available Windows decoders. Video and audio use the AVI presentation times, including an audio start offset. A hidden layer or a movie outside the view is silent. PauseAvax and disabling the control also silence playback; when it becomes active again, sound resumes at the current movie time. In a multi-picture sequence, only the currently displayed AVI slot is audible. AnimationOn and AnimationInterVal select the sequence slot; they do not repeatedly restart a movie. Removing the movie, replacing the drawing, EndAvax or disposing the control stops its soundtrack and releases its playback resources. Static image output, printing and previews use the poster frame and contain no sound. Unsupported or damaged video does not insert an empty object. If an audio decoder or output device is unavailable, the video remains in the drawing and its soundtrack cannot play. DSP Group TrueSpeech audio (WAVE format 0x0022, mono at 8000 Hz), including the original clock.avi example, is decoded by the supplied FFmpeg-Avax.TrueSpeech.dll without installing a system codec. Keep this separate runtime library and its accompanying LGPL license, notice and matching source archive with redistributed builds.

Numeric changes from v2

Update the following variables, arrays or event-data receivers when porting the corresponding v2 call. The declarations above show the current overloads.

Parameter or value	v2 type	v3 type
xMin	Single	Double
yMin	Single	Double
xMax	Single	Double
yMax	Single	Double
iLayer	Short	Integer

For ByRef parameters, use a variable with the exact v3 type. Preserve array dimensions and the documented indexing. Moving from v2 to v3 explains conversion and legacy file-storage rules.

Example

Setup

Add an Avax.cAvax Windows Forms control named `cAvax1`. Paste the procedures into the form class. Example setup and supporting files explains references, initialization and file paths.

Add these Windows Forms controls or components to the form. Fields referenced by Handles clauses must be declared WithEvents, as the Visual Studio designer normally does.

Name	Type
Button1	Button

VB.NET example

```
Private Sub Form1_Load(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs) Handles
    MyBase.Load
    Call cAvax1.StartAvax()
End Sub
Private Sub Form1_FormClosed(ByVal eventSender As System.Object, ByVal eventArgs As
    System.Windows.Forms.FormClosedEventArgs) Handles Me.FormClosed
    Call cAvax1.EndAvax()
End Sub
Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
    Button1.Click
    Dim sAVIFile As String = String.Empty
    Dim xMin As Double
    Dim yMin As Double
    Dim xMax As Double
    Dim yMax As Double
    Dim zOrder As Integer
    Dim vItemProperties As Object = Nothing
    Dim iLayer As Integer
    Dim Answer As Integer
    xMax = 10
    yMax = 10
    sAVIFile = My.Application.Info.DirectoryPath & "\..\clock.avi"
    Answer = cAvax1.Add_AVI(sAVIFile, xMin, yMin, xMax, yMax, zOrder, vItemProperties, iLayer)
End Sub
```

Add_AVX

cAvax method

Purpose

This function inserts an .AVX file into the drawing

VB.NET declarations

```
Function Add_AVX(ByRef sAvaxFile As String, ByRef x1 As Double, ByRef y1 As Double, ByRef z1 As Double,
    ByRef vItemProperties As Object, ByRef sScale As Double, ByRef sAngle As Double) As Integer

Function Add_AVX(ByRef sAvaxFile As String, ByRef x1 As Double, ByRef y1 As Double, ByRef z1 As Double) As
     Integer
```

Parameters

Name	Type and passing	Meaning
sAvaxFile	ByRef sAvaxFile As String	Is the name of the .AVX file.
x1	ByRef x1 As Double	Is the x coordinate of the insertion point
y1	ByRef y1 As Double	Is the y coordinate of the insertion point
z1	ByRef z1 As Double	Is the z coordinate of the insertion point
vItemProperties	ByRef vItemProperties As Object	Is a string array with properties for the new objects
sScale	ByRef sScale As Double	Is the importing drawing's scale factor
sAngle	ByRef sAngle As Double	Is the importing drawing's rotation angle relative to the insertion point

Use the declaration for the overload you call. ByRef arguments can receive outputs; declare variables and arrays with the exact indicated type. Optional defaults are shown in the declaration.

Return value

Integer - An Avax handle on success, zero on error.

Remarks

Inserts an existing AVX or native AVAX drawing as one block, preserving its child objects and merging its color, layer, font and line-style resources. Call StartAvax first and keep the control enabled. An existing supplied path takes precedence. Otherwise a relative name is resolved under the data directory configured by DirsSet; an extensionless name defaults to .avx. On successful fallback, ByRef sAvaxFile receives the resolved filename. Resolving a file does not change the configured data directory or modify the source file. The short overload uses scale 1 and angle 0. The full overload uses Double insertion coordinates, scale and angle; a nonpositive finite scale becomes 1 and is returned through sScale. Positive angles rotate counterclockwise in the XY plane, in degrees; z1 offsets the child elevations. The return value is the Integer handle of the new block, or 0 if it could not be inserted. Invalid input or an unreadable file leaves the drawing, resources, Undo/Redo and caller arguments unchanged. A successful import is one undoable operation. Native .avax storage retains the transformed Double coordinates; the original AVX file stores Single coordinates. The example uses the original Sample.avx beside the application folder, rotated by 45 degrees. This programmatic operation publishes Change without NewUserItems, as in R2. Use its documented return value/output parameters for the result. Interactive drawing commands and explicit announcing shape/picture/document helpers retain their own notification policy.

Numeric changes from v2

Update the following variables, arrays or event-data receivers when porting the corresponding v2 call. The declarations above show the current overloads.

Parameter or value	v2 type	v3 type
x1	Single	Double
y1	Single	Double

Parameter or value	v2 type	v3 type
z1	Single	Double
sScale	Single	Double
sAngle	Single	Double

For ByRef parameters, use a variable with the exact v3 type. Preserve array dimensions and the documented indexing. Moving from v2 to v3 explains conversion and legacy file-storage rules.

Example

Setup

Add an Avax.cAvax Windows Forms control named `cAvax1`. Paste the procedures into the form class. Example setup and supporting files explains references, initialization and file paths.

Add these Windows Forms controls or components to the form. Fields referenced by Handles clauses must be declared WithEvents, as the Visual Studio designer normally does.

Name	Type
Command1	Button

VB.NET example

```
Private Sub Form1_Load(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs) Handles
    MyBase.Load
    cAvax1.StartAvax()
End Sub
Private Sub Form1_FormClosed(ByVal eventSender As System.Object, ByVal eventArgs As
    System.Windows.Forms.FormClosedEventArgs) Handles Me.FormClosed
    cAvax1.EndAvax()
End Sub
Private Sub Command1_Click(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs)
    Handles Command1.Click
    Dim sAvaxFile As String = String.Empty
    Dim x1 As Double
    Dim y1 As Double
    Dim z1 As Double
    Dim vItemProperties As Object = Nothing
    Dim sScale As Double
    Dim sAngle As Double
    Dim AvaxHandle As Integer
    sAvaxFile = My.Application.Info.DirectoryPath & "\..\Sample.avx"
    sAngle = 45
    AvaxHandle = cAvax1.Add_AVX(sAvaxFile, x1, y1, z1, vItemProperties, sScale, sAngle)
    CAvax1.Command = Avax.AvaxCommand.AutoLimits_c
End Sub
```

Add_Bezier

cAvax method

Purpose

Adds a Bezier curve from XYZ control points and stores its sampled outline as a polyline.

VB.NET declarations

```
Function Add_Bezier(ByRef x1 As Double(), ByRef y1 As Double(), ByRef z1 As Double()) As Integer

Function Add_Bezier(ByRef x1 As Double(), ByRef y1 As Double(), ByRef z1 As Double(), ByRef
    vItemProperties As Object, ByRef iClr As Integer, ByRef iWidth As Integer, ByRef iStyle As Integer, ByRef
    iLayer As Integer) As Integer
```

Parameters

Name	Type and passing	Meaning
x1	ByRef x1 As Double()	One-based Double X control coordinates; index zero is reserved. Two to 3200 controls (at least three for a closed B-spline). Caller arrays are not retained or modified.
y1	ByRef y1 As Double()	One-based Double Y control coordinates, with the same array length as x1. Values must be finite.
z1	ByRef z1 As Double()	Optional one-based Double Z control coordinates. Nothing means zero; otherwise the length must match x1. Unlike R2, generated samples retain/interpolate the supplied Z.
vItemProperties	ByRef vItemProperties As Object	Optional string-array metadata, copied into the one resulting polyline. The caller object reference is preserved.
iClr	ByRef iClr As Integer	Integer color. -1 uses the current pen; successful full calls return the effective value ByRef.
iWidth	ByRef iWidth As Integer	Integer width. -1 uses the current pen; nonpositive resolved width becomes 1.
iStyle	ByRef iStyle As Integer	Integer line-style index; -1 uses the current pen. It is not truncated to the old byte/Integer range.
iLayer	ByRef iLayer As Integer	Integer layer index; -1 uses the current pen. Invalid input leaves the model and caller outputs unchanged.

Use the declaration for the overload you call. ByRef arguments can receive outputs; declare variables and arrays with the exact indicated type. Optional defaults are shown in the declaration.

Return value

Integer - An Avax handle on success, zero on error.

Remarks

The complete control polygon defines one Bezier curve of degree n-1. Interior controls guide the curve and need not lie on it. The result uses 10*n parameter samples, including the exact first and last points, and is stored as an editable Polyline_i with XYZ elevations. At least two controls are required.

Supply equal-length Double coordinate arrays with logical points at indices 1 through the upper bound; index 0 is ignored. Nothing for the Z array supplies zero elevations. The control copies the coordinates. The maximum is 3200 input controls. The short overload uses the current pen. The full overload accepts copied String metadata and Integer pen attributes; -1 selects the current pen. Only attributes declared ByRef are returned to the caller. A started, enabled control is required. Invalid coordinates or geometry return 0 before changing the drawing or caller outputs. A successful insertion is one Undo operation and raises Change without NewUserItems. Application event-handler exceptions propagate after insertion.

Numeric changes from v2

Update the following variables, arrays or event-data receivers when porting the corresponding v2 call. The declarations above show the current overloads.

Parameter or value	v2 type	v3 type
x1	Single()	Double()
y1	Single()	Double()
z1	Single()	Double()
iClr	Short	Integer
iWidth	Short	Integer
iStyle	Short	Integer
iLayer	Short	Integer

For ByRef parameters, use a variable with the exact v3 type. Preserve array dimensions and the documented indexing. Moving from v2 to v3 explains conversion and legacy file-storage rules.

Example

Setup

Add an Avax.cAvax Windows Forms control named `cAvax1`. Paste the procedures into the form class. Example setup and supporting files explains references, initialization and file paths.

Add these Windows Forms controls or components to the form. Fields referenced by Handles clauses must be declared WithEvents, as the Visual Studio designer normally does.

Name	Type
Command1	Button

VB.NET example

```
Private Sub Form1_Load(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs) Handles
    MyBase.Load
    cAvax1.StartAvax()
End Sub
Private Sub Form1_FormClosed(ByVal eventSender As System.Object, ByVal eventArgs As
    System.Windows.Forms.FormClosedEventArgs) Handles Me.FormClosed
    cAvax1.EndAvax()
End Sub
Private Sub Command1_Click(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs)
    Handles Command1.Click
    Dim x1() As Double = Nothing
    Dim y1() As Double = Nothing
    Dim z1() As Double = Nothing
    Dim vItemProperties As Object = Nothing
    Dim iClr As Integer
    Dim iWidth As Integer
    Dim iStyle As Integer
    Dim iLayer As Integer
    Dim AvaxHandle As Integer
    ReDim x1(4)
    ReDim y1(4)
    ReDim z1(4)
    ' Define the points
    x1(1) = 4 : y1(1) = 5 : z1(1) = 0
    x1(2) = 8 : y1(2) = 1 : z1(2) = 0
    x1(3) = 12 : y1(3) = 8 : z1(3) = 0
    x1(4) = 12 : y1(4) = 12 : z1(4) = 0
    ' Define the Color
    iClr = 1
```

```
    AvaxHandle = cAvax1.Add_Bezier(x1, y1, z1, vItemProperties, iClr, iWidth, iStyle, iLayer)
End Sub
```

Add_Block

cAvax method

Purpose

This function adds a block into the drawing

VB.NET declarations

```
Function Add_Block(ByRef sBlockName As String, ByRef x1 As Double, ByRef y1 As Double, ByRef z1 As Double)
     As Integer

Function Add_Block(ByRef sBlockName As String, ByRef x1 As Double, ByRef y1 As Double, ByRef z1 As Double,
     ByRef vItemProperties As Object, ByRef sScale As Double, ByRef sAngle As Double) As Integer
```

Parameters

Name	Type and passing	Meaning
sBlockName	ByRef sBlockName As String	Is the block's name
x1	ByRef x1 As Double	Is the x coordinate of the insertion point
y1	ByRef y1 As Double	Is the y coordinate of the insertion point
z1	ByRef z1 As Double	Is the z coordinate of the insertion point
vItemProperties	ByRef vItemProperties As Object	Is a string array with properties for the new object
sScale	ByRef sScale As Double	Is the block's scale factor
sAngle	ByRef sAngle As Double	Is the block's rotation angle relative to the insertion point

Use the declaration for the overload you call. ByRef arguments can receive outputs; declare variables and arrays with the exact indicated type. Optional defaults are shown in the declaration.

Return value

Integer - An Avax handle on success, zero on error.

Remarks

Requires a started, enabled control. Coordinates, scale and angle use Double; the returned handle is Integer. The short overload uses scale 1 and angle 0. A finite scale <= 0 is normalized to 1 and returned through ByRef sScale after success. Rotation is counterclockwise in degrees in XY; z1 offsets child Z, including nested blocks. All geometry, external document images and resource remapping are prepared before any destination mutation. Invalid input, unavailable/malformed files, overflowing geometry or exhausted handles return 0 and an English diagnostic; failed calls preserve the drawing, resource tables, selection and undo/redo. A successful insertion is one block handle and one undo record; Undo/Redo includes palette, layers, layer statuses, fonts, font charsets and line styles. The parent uses current pen/caller properties; children retain source metadata and mapped resources. Host Change/NewUserItems exceptions propagate after a completed edit; callback-started commands and replacement documents survive. Resolves a cached named definition or a file. Extensionless names search .blk, .avax and .avx in DirsSet sDirLib and then the current directory. Full paths and historical .blk files remain supported. Cached definitions own their geometry/resources independently of the active document. Font binding audit: Legacy embedded/sibling blocks with their own font tables remap referenced ordinary and markup fonts into the containing document without replacing its existing slots. A block without a font table inherits the parent table. Reserved slot zero is never imported as a real font entry. Nested font edits, resource-owning Undo/Redo and second native/AVX save are covered. Nested resource audit: Embedded AVX and sibling BLK drawings retain private palette, layer names/status, line-style and font references. Missing or empty header resource tags inherit the parent table. Only used references are merged; existing parent slots stay unchanged, including records decoded after the block. Point symbol styles are not line-style indices. Nested document color tags and editable-preview metadata are remapped with their owning resources. The prepared import and its tables participate in Undo/Redo. This programmatic operation publishes Change without NewUserItems, as in R2. Use its documented return value/output parameters for the result. Interactive drawing commands and explicit announcing shape/picture/document helpers retain their own notification policy.

Numeric changes from v2

Update the following variables, arrays or event-data receivers when porting the corresponding v2 call. The declarations above show the current overloads.

Parameter or value	v2 type	v3 type
x1	Single	Double
y1	Single	Double
z1	Single	Double
sScale	Single	Double
sAngle	Single	Double

For ByRef parameters, use a variable with the exact v3 type. Preserve array dimensions and the documented indexing. Moving from v2 to v3 explains conversion and legacy file-storage rules.

Example

Setup

Add an Avax.cAvax Windows Forms control named `cAvax1`. Paste the procedures into the form class. Example setup and supporting files explains references, initialization and file paths.

Add these Windows Forms controls or components to the form. Fields referenced by Handles clauses must be declared WithEvents, as the Visual Studio designer normally does.

Name	Type
Command1	Button

VB.NET example

```
Private Sub Form1_Load(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs) Handles
    MyBase.Load
    cAvax1.StartAvax()
End Sub
Private Sub Form1_FormClosed(ByVal eventSender As System.Object, ByVal eventArgs As
    System.Windows.Forms.FormClosedEventArgs) Handles Me.FormClosed
    cAvax1.EndAvax()
End Sub
Private Sub Command1_Click(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs)
    Handles Command1.Click
    Dim sBlockName As String = String.Empty
    Dim x1 As Double
    Dim y1 As Double
    Dim z1 As Double
    Dim vItemProperties As Object = Nothing
    Dim sScale As Double
    Dim sAngle As Double
    Dim AvaxHandle As Integer
    x1 = 0
    y1 = 0
    z1 = 0
    sBlockName = "test"
    sAngle = 45
    sScale = 0.25
    AvaxHandle = cAvax1.Add_Block(sBlockName, x1, y1, z1, vItemProperties, sScale, sAngle)
End Sub
```

Add_BlockLines

cAvax method

Purpose

Adds one object containing straight XYZ line segments from six one-based coordinate arrays; the segments need not join.

VB.NET declarations

```
Function Add_BlockLines(ByRef x1 As Double(), ByRef y1 As Double(), ByRef z1 As Double(), ByRef x2 As
    Double(), ByRef y2 As Double(), ByRef z2 As Double()) As Integer

Function Add_BlockLines(ByRef x1 As Double(), ByRef y1 As Double(), ByRef z1 As Double(), ByRef x2 As
    Double(), ByRef y2 As Double(), ByRef z2 As Double(), ByRef vItemProperties As Object, ByRef iClr As
    Integer, ByRef iWidth As Integer, ByRef iStyle As Integer, ByRef iLayer As Integer) As Integer
```

Parameters

Name	Type and passing	Meaning
x1	ByRef x1 As Double()	An array for the x coordinates of the lines start point
y1	ByRef y1 As Double()	An array for the y coordinates of the lines start point
z1	ByRef z1 As Double()	An array for the z coordinates of the lines start point
x2	ByRef x2 As Double()	An array for the x coordinates of the lines end point
y2	ByRef y2 As Double()	An array for the y coordinates of the lines end point
z2	ByRef z2 As Double()	An array for the z coordinates of the lines end point
vItemProperties	ByRef vItemProperties As Object	One-dimensional String() or Object() containing strings/Nothing, with logical properties starting at index 1. The control stores its own copy and retains the original caller reference. Nothing means no properties, except Add_Data which requires an array.
iClr	ByRef iClr As Integer	-1 uses the corresponding current pen setting. Other negative values become 0. Positive values use Integer storage. When declared ByRef, the effective value is written back only after successful validation.
iWidth	ByRef iWidth As Integer	-1 uses the current pen. Nonpositive resolved widths become 1, as in R2. Positive values use Integer storage. When declared ByRef, the effective value is written back only after successful validation.
iStyle	ByRef iStyle As Integer	-1 uses the corresponding current pen setting. Other negative values become 0. Positive values use Integer storage. When declared ByRef, the effective value is written back only after successful validation.
iLayer	ByRef iLayer As Integer	-1 uses the corresponding current pen setting. Other negative values become 0. Positive values use Integer storage. When declared ByRef, the effective value is written back only after successful validation.

Use the declaration for the overload you call. ByRef arguments can receive outputs; declare variables and arrays with the exact indicated type. Optional defaults are shown in the declaration.

Return value

Integer - An Avax handle on success, zero on error.

Remarks

Supply all six equal-length Double coordinate arrays with segments at indices 1 through the upper bound; index 0 is ignored. At least one segment is required. A single segment produces a Line_i; several segments produce one Polyline_i, including disconnected segments and every XYZ endpoint.

The short overload uses the current pen. The full overload accepts copied String metadata and Integer pen fields. -1 selects the current pen; other negative indices become 0 and nonpositive widths become 1. Resolved fields are returned only when declared ByRef. A started, enabled control is required. Invalid arrays or unrepresentable geometry return 0 without changing the drawing, history or caller arguments. Success is one Undo operation with Change and no NewUserItems. Application event-handler exceptions propagate after insertion.

Numeric changes from v2

Update the following variables, arrays or event-data receivers when porting the corresponding v2 call. The declarations above show the current overloads.

Parameter or value	v2 type	v3 type
x1	Single()	Double()
y1	Single()	Double()
z1	Single()	Double()
x2	Single()	Double()
y2	Single()	Double()
z2	Single()	Double()
iClr	Short	Integer
iWidth	Short	Integer
iStyle	Short	Integer
iLayer	Short	Integer

For ByRef parameters, use a variable with the exact v3 type. Preserve array dimensions and the documented indexing. Moving from v2 to v3 explains conversion and legacy file-storage rules.

Example

Setup

Add an Avax.cAvax Windows Forms control named `cAvax1`. Paste the procedures into the form class. Example setup and supporting files explains references, initialization and file paths.

Add these Windows Forms controls or components to the form. Fields referenced by Handles clauses must be declared WithEvents, as the Visual Studio designer normally does.

Name	Type
Command1	Button

VB.NET example

```
Private Sub Form1_Load(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs) Handles
    MyBase.Load
    cAvax1.StartAvax()
End Sub
Private Sub Form1_FormClosed(ByVal eventSender As System.Object, ByVal eventArgs As
    System.Windows.Forms.FormClosedEventArgs) Handles Me.FormClosed
    cAvax1.EndAvax()
End Sub
Private Sub Command1_Click(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs)
    Handles Command1.Click
    Static xStart, yStart As Double
    Dim x1() As Double = Nothing
    Dim y1() As Double = Nothing
    Dim z1() As Double = Nothing
    Dim x2() As Double = Nothing
    Dim y2() As Double = Nothing
    Dim z2() As Double = Nothing
    Dim vItemProperties As Object = Nothing
```

```
    Dim iClr As Integer
    Dim iWidth As Integer
    Dim iStyle As Integer
    Dim iLayer As Integer
    Dim Answer As Integer
    ReDim x1(4)
    ReDim y1(4)
    ReDim z1(4)
    ReDim x2(4)
    ReDim y2(4)
    ReDim z2(4)
    x1(1) = xStart : y1(1) = yStart : x2(1) = xStart : y2(1) = yStart + 10
    x1(2) = xStart : y1(2) = yStart + 10 : x2(2) = xStart + 10 : y2(2) = yStart + 10
    x1(3) = xStart + 10 : y1(3) = yStart + 10 : x2(3) = xStart + 10 : y2(3) = yStart
    x1(4) = xStart + 10 : y1(4) = yStart : x2(4) = xStart : y2(4) = yStart
    iClr = 1
    Answer = cAvax1.Add_BlockLines(x1, y1, z1, x2, y2, z2, vItemProperties, iClr, iWidth, iStyle, iLayer)
    xStart = xStart + 0.5
    yStart = yStart + 0.5
End Sub
```

Add_CBSpline

cAvax method

Purpose

Adds a closed periodic cubic B-spline from XYZ controls and stores its sampled outline as a closed polyline.

VB.NET declarations

```
Function Add_CBSpline(ByRef x1 As Double(), ByRef y1 As Double(), ByRef z1 As Double(), ByRef
    vItemProperties As Object, ByRef iClr As Integer, ByRef iWidth As Integer, ByRef iStyle As Integer, ByRef
    iLayer As Integer) As Integer

Function Add_CBSpline(ByRef x1 As Double(), ByRef y1 As Double(), ByRef z1 As Double()) As Integer
```

Parameters

Name	Type and passing	Meaning
x1	ByRef x1 As Double()	One-based Double X control coordinates; index zero is reserved. Two to 3200 controls (at least three for a closed B-spline). Caller arrays are not retained or modified.
y1	ByRef y1 As Double()	One-based Double Y control coordinates, with the same array length as x1. Values must be finite.
z1	ByRef z1 As Double()	Optional one-based Double Z control coordinates. Nothing means zero; otherwise the length must match x1. Unlike R2, generated samples retain/interpolate the supplied Z.
vItemProperties	ByRef vItemProperties As Object	Optional string-array metadata, copied into the one resulting polyline. The caller object reference is preserved.
iClr	ByRef iClr As Integer	Integer color. -1 uses the current pen; successful full calls return the effective value ByRef.
iWidth	ByRef iWidth As Integer	Integer width. -1 uses the current pen; nonpositive resolved width becomes 1.
iStyle	ByRef iStyle As Integer	Integer line-style index; -1 uses the current pen. It is not truncated to the old byte/Integer range.
iLayer	ByRef iLayer As Integer	Integer layer index; -1 uses the current pen. Invalid input leaves the model and caller outputs unchanged.

Use the declaration for the overload you call. ByRef arguments can receive outputs; declare variables and arrays with the exact indicated type. Optional defaults are shown in the declaration.

Return value

Integer - An Avax handle on success, zero on error.

Remarks

The periodic cubic B-spline is guided by its control polygon and need not pass through the controls. A final control exactly equal to the first is a closure marker; at least three controls must remain. For up to 12 controls, midpoint refinement is applied. Each resulting span has 10 straight segments. The sampled Polyline_i closes by repeating its first XYZ point exactly.

Supply equal-length Double coordinate arrays with logical points at indices 1 through the upper bound; index 0 is ignored. Nothing for the Z array supplies zero elevations. The control copies the coordinates. The maximum is 3200 input controls. The short overload uses the current pen. The full overload accepts copied String metadata and Integer pen attributes; -1 selects the current pen. Only attributes declared ByRef are returned to the caller. A started, enabled control is required. Invalid coordinates or geometry return 0 before changing the drawing or caller outputs. A successful insertion is one Undo operation and raises Change without NewUserItems. Application event-handler exceptions propagate after insertion.

Numeric changes from v2

Update the following variables, arrays or event-data receivers when porting the corresponding v2 call. The declarations above show the current overloads.

Parameter or value	v2 type	v3 type
x1	Single()	Double()
y1	Single()	Double()
z1	Single()	Double()
iClr	Short	Integer
iWidth	Short	Integer
iStyle	Short	Integer
iLayer	Short	Integer

For ByRef parameters, use a variable with the exact v3 type. Preserve array dimensions and the documented indexing. Moving from v2 to v3 explains conversion and legacy file-storage rules.

Example

Setup

Add an Avax.cAvax Windows Forms control named `cAvax1`. Paste the procedures into the form class. Example setup and supporting files explains references, initialization and file paths.

Add these Windows Forms controls or components to the form. Fields referenced by Handles clauses must be declared WithEvents, as the Visual Studio designer normally does.

Name	Type
Command1	Button

VB.NET example

```
Private Sub Form1_Load(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs) Handles
    MyBase.Load
    cAvax1.StartAvax()
End Sub
Private Sub Form1_FormClosed(ByVal eventSender As System.Object, ByVal eventArgs As
    System.Windows.Forms.FormClosedEventArgs) Handles Me.FormClosed
    cAvax1.EndAvax()
End Sub
Private Sub Command1_Click(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs)
    Handles Command1.Click
    Dim x1() As Double = Nothing
    Dim y1() As Double = Nothing
    Dim z1() As Double = Nothing
    Dim vItemProperties As Object = Nothing
    Dim iClr As Integer
    Dim iWidth As Integer
    Dim iStyle As Integer
    Dim iLayer As Integer
    Dim Answer As Integer
    ReDim x1(4)
    ReDim y1(4)
    ReDim z1(4)
    'Points for the spline nodes definition
    x1(1) = 1 : y1(1) = 4 : z1(1) = 0
    x1(2) = 3 : y1(2) = 18 : z1(2) = 0
    x1(3) = 6.5 : y1(3) = 12 : z1(3) = 0
    x1(4) = 3 : y1(4) = 2.5 : z1(4) = 0
    'Color and width definition
    iClr = 1
```

```
    iWidth = 1
    Answer = cAvax1.Add_CBSpline(x1, y1, z1, vItemProperties, iClr, iWidth, iStyle, iLayer)
End Sub
```

Add_Circle

cAvax method

Purpose

This function adds a circle into the drawing

See Circle and Ellipse commands for center/radius, diameter and three-point circles, ellipse radii, Double precision and one-shot command completion.

VB.NET declarations

```
Function Add_Circle(ByRef xc1 As Double, ByRef yc1 As Double, ByRef zc1 As Double, ByRef R As Double) As
    Integer

Function Add_Circle(ByRef xc1 As Double, ByRef yc1 As Double, ByRef zc1 As Double, ByRef R As Double,
    ByRef vItemProperties As Object, ByRef iClr As Integer, ByRef iWidth As Integer, ByRef iStyle As Integer,
    ByRef iLayer As Integer) As Integer
```

Parameters

Name	Type and passing	Meaning
xc1	ByRef xc1 As Double	Is the x coordinate of the circle's center
yc1	ByRef yc1 As Double	Is the y coordinate of the circle's center
zc1	ByRef zc1 As Double	Is the z coordinate of the circle's center
R	ByRef R As Double	Is the circle's radius
vItemProperties	ByRef vItemProperties As Object	One-dimensional String() or Object() containing strings/Nothing, with logical properties starting at index 1. The control stores its own copy and retains the original caller reference. Nothing means no properties, except Add_Data which requires an array.
iClr	ByRef iClr As Integer	-1 uses the corresponding current pen setting. Other negative values become 0. Positive values use Integer storage. When declared ByRef, the effective value is written back only after successful validation.
iWidth	ByRef iWidth As Integer	-1 uses the current pen. Nonpositive resolved widths become 1, as in R2. Positive values use Integer storage. When declared ByRef, the effective value is written back only after successful validation.
iStyle	ByRef iStyle As Integer	-1 uses the corresponding current pen setting. Other negative values become 0. Positive values use Integer storage. When declared ByRef, the effective value is written back only after successful validation.
iLayer	ByRef iLayer As Integer	-1 uses the corresponding current pen setting. Other negative values become 0. Positive values use Integer storage. When declared ByRef, the effective value is written back only after successful validation.

Use the declaration for the overload you call. ByRef arguments can receive outputs; declare variables and arrays with the exact indicated type. Optional defaults are shown in the declaration.

Return value

Integer - An Avax handle on success, zero on error.

Numeric changes from v2

Update the following variables, arrays or event-data receivers when porting the corresponding v2 call. The declarations above show the current overloads.

Parameter or value	v2 type	v3 type
xc1	Single	Double

Parameter or value	v2 type	v3 type
yc1	Single	Double
zc1	Single	Double
R	Single	Double
iClr	Short	Integer
iWidth	Short	Integer
iStyle	Short	Integer
iLayer	Short	Integer

For ByRef parameters, use a variable with the exact v3 type. Preserve array dimensions and the documented indexing. Moving from v2 to v3 explains conversion and legacy file-storage rules.

Example

Setup

Add an Avax.cAvax Windows Forms control named `cAvax1`. Paste the procedures into the form class. Example setup and supporting files explains references, initialization and file paths.

Add these Windows Forms controls or components to the form. Fields referenced by Handles clauses must be declared WithEvents, as the Visual Studio designer normally does.

Name	Type
Command1	Button

VB.NET example

```
Private Sub Form1_Load(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs) Handles
    MyBase.Load
    cAvax1.StartAvax()
End Sub
Private Sub Form1_FormClosed(ByVal eventSender As System.Object, ByVal eventArgs As
    System.Windows.Forms.FormClosedEventArgs) Handles Me.FormClosed
    cAvax1.EndAvax()
End Sub
Private Sub Command1_Click(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs)
    Handles Command1.Click
    Dim xc1 As Double
    Dim yc1 As Double
    Dim zc1 As Double
    Dim r As Double
    Dim vItemProperties As Object = Nothing
    Dim iClr As Integer
    Dim iWidth As Integer
    Dim iStyle As Integer
    Dim iLayer As Integer
    Dim Answer As Integer
    'Define the Center of the Circle
    xc1 = 8 : yc1 = 6 : zc1 = 0
    'Define the radius
    r = 10
    'Finally, define the drawing color and the drawind width
    iClr = 5
    iWidth = 1
    iStyle = 1
    Answer = cAvax1.Add_Circle(xc1, yc1, zc1, r, vItemProperties, iClr, iWidth, iStyle, iLayer)
End Sub
```

Add_CSpline1

cAvax method

Purpose

Adds a cubic spline through the XYZ guide points with zero first derivatives at both ends, stored as a sampled polyline.

VB.NET declarations

```
Function Add_CSpline1(ByRef x1 As Double(), ByRef y1 As Double(), ByRef z1 As Double(), ByRef
    vItemProperties As Object, ByRef iClr As Integer, ByRef iWidth As Integer, ByRef iStyle As Integer, ByRef
    iLayer As Integer) As Integer

Function Add_CSpline1(ByRef x1 As Double(), ByRef y1 As Double(), ByRef z1 As Double()) As Integer
```

Parameters

Name	Type and passing	Meaning
x1	ByRef x1 As Double()	One-based Double X control coordinates; index zero is reserved. Two to 3200 controls (at least three for a closed B-spline). Caller arrays are not retained or modified.
y1	ByRef y1 As Double()	One-based Double Y control coordinates, with the same array length as x1. Values must be finite.
z1	ByRef z1 As Double()	Optional one-based Double Z control coordinates. Nothing means zero; otherwise the length must match x1. Unlike R2, generated samples retain/interpolate the supplied Z.
vItemProperties	ByRef vItemProperties As Object	Optional string-array metadata, copied into the one resulting polyline. The caller object reference is preserved.
iClr	ByRef iClr As Integer	Integer color. -1 uses the current pen; successful full calls return the effective value ByRef.
iWidth	ByRef iWidth As Integer	Integer width. -1 uses the current pen; nonpositive resolved width becomes 1.
iStyle	ByRef iStyle As Integer	Integer line-style index; -1 uses the current pen. It is not truncated to the old byte/Integer range.
iLayer	ByRef iLayer As Integer	Integer layer index; -1 uses the current pen. Invalid input leaves the model and caller outputs unchanged.

Use the declaration for the overload you call. ByRef arguments can receive outputs; declare variables and arrays with the exact indicated type. Optional defaults are shown in the declaration.

Return value

Integer - An Avax handle on success, zero on error.

Remarks

The spline passes through the guide points and has zero first derivatives at both endpoints. Consecutive XY guide distances determine its parameter intervals; a Z-only interval uses its Z distance. Exact consecutive duplicate XYZ guides are combined. Each remaining interval has 15 straight segments, stored as one editable Polyline_i. At least two input guides are required.

Supply equal-length Double coordinate arrays with logical points at indices 1 through the upper bound; index 0 is ignored. Nothing for the Z array supplies zero elevations. The control copies the coordinates. The maximum is 3200 input controls. The short overload uses the current pen. The full overload accepts copied String metadata and Integer pen attributes; -1 selects the current pen. Only attributes declared ByRef are returned to the caller. A started, enabled control is required. Invalid coordinates or geometry return 0 before changing the drawing or caller outputs. A successful insertion is one Undo operation and raises Change without NewUserItems. Application event-handler exceptions propagate after insertion.

Numeric changes from v2

Update the following variables, arrays or event-data receivers when porting the corresponding v2 call. The declarations above show the current overloads.

Parameter or value	v2 type	v3 type
x1	Single()	Double()
y1	Single()	Double()
z1	Single()	Double()
iClr	Short	Integer
iWidth	Short	Integer
iStyle	Short	Integer
iLayer	Short	Integer

For ByRef parameters, use a variable with the exact v3 type. Preserve array dimensions and the documented indexing. Moving from v2 to v3 explains conversion and legacy file-storage rules.

Example

Setup

Add an Avax.cAvax Windows Forms control named `cAvax1`. Paste the procedures into the form class. Example setup and supporting files explains references, initialization and file paths.

Add these Windows Forms controls or components to the form. Fields referenced by Handles clauses must be declared WithEvents, as the Visual Studio designer normally does.

Name	Type
Command1	Button

VB.NET example

```
Private Sub Form1_Load(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs) Handles
    MyBase.Load
    cAvax1.StartAvax()
End Sub
Private Sub Form1_FormClosed(ByVal eventSender As System.Object, ByVal eventArgs As
    System.Windows.Forms.FormClosedEventArgs) Handles Me.FormClosed
    cAvax1.EndAvax()
End Sub
Private Sub Command1_Click(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs)
    Handles Command1.Click
    Dim x1() As Double = Nothing
    Dim y1() As Double = Nothing
    Dim z1() As Double = Nothing
    Dim vItemProperties As Object = Nothing
    Dim iClr As Integer
    Dim iWidth As Integer
    Dim iStyle As Integer
    Dim iLayer As Integer
    Dim AvaxHandle As Integer
    ReDim x1(4)
    ReDim y1(4)
    ReDim z1(4)
    ' Define the points
    x1(1) = 0 : y1(1) = 0 : z1(1) = 0
    x1(2) = 4 : y1(2) = 5 : z1(2) = 0
    x1(3) = 8 : y1(3) = 1 : z1(3) = 0
    x1(4) = 12 : y1(4) = 8 : z1(4) = 0
    'Define the Color of the spline
    iClr = 1
```

```
    AvaxHandle = cAvax1.Add_CSpline1(x1, y1, z1, vItemProperties, iClr, iWidth, iStyle, iLayer)
End Sub
```

Add_CSpline2

cAvax method

Purpose

Adds a natural cubic spline through the XYZ guide points with zero second derivatives at both ends, stored as a sampled polyline.

VB.NET declarations

```
Function Add_CSpline2(ByRef x1 As Double(), ByRef y1 As Double(), ByRef z1 As Double(), ByRef
    vItemProperties As Object, ByRef iClr As Integer, ByRef iWidth As Integer, ByRef iStyle As Integer, ByRef
    iLayer As Integer) As Integer

Function Add_CSpline2(ByRef x1 As Double(), ByRef y1 As Double(), ByRef z1 As Double()) As Integer
```

Parameters

Name	Type and passing	Meaning
x1	ByRef x1 As Double()	One-based Double X control coordinates; index zero is reserved. Two to 3200 controls (at least three for a closed B-spline). Caller arrays are not retained or modified.
y1	ByRef y1 As Double()	One-based Double Y control coordinates, with the same array length as x1. Values must be finite.
z1	ByRef z1 As Double()	Optional one-based Double Z control coordinates. Nothing means zero; otherwise the length must match x1. Unlike R2, generated samples retain/interpolate the supplied Z.
vItemProperties	ByRef vItemProperties As Object	Optional string-array metadata, copied into the one resulting polyline. The caller object reference is preserved.
iClr	ByRef iClr As Integer	Integer color. -1 uses the current pen; successful full calls return the effective value ByRef.
iWidth	ByRef iWidth As Integer	Integer width. -1 uses the current pen; nonpositive resolved width becomes 1.
iStyle	ByRef iStyle As Integer	Integer line-style index; -1 uses the current pen. It is not truncated to the old byte/Integer range.
iLayer	ByRef iLayer As Integer	Integer layer index; -1 uses the current pen. Invalid input leaves the model and caller outputs unchanged.

Use the declaration for the overload you call. ByRef arguments can receive outputs; declare variables and arrays with the exact indicated type. Optional defaults are shown in the declaration.

Return value

Integer - An Avax handle on success, zero on error.

Remarks

The natural spline passes through the guide points and has zero second derivatives at both endpoints. Consecutive XY guide distances determine its parameter intervals; a Z-only interval uses its Z distance. Exact consecutive duplicate XYZ guides are combined. Each remaining interval has 15 straight segments, stored as one editable Polyline_i. At least two input guides are required.

Supply equal-length Double coordinate arrays with logical points at indices 1 through the upper bound; index 0 is ignored. Nothing for the Z array supplies zero elevations. The control copies the coordinates. The maximum is 3200 input controls. The short overload uses the current pen. The full overload accepts copied String metadata and Integer pen attributes; -1 selects the current pen. Only attributes declared ByRef are returned to the caller. A started, enabled control is required. Invalid coordinates or geometry return 0 before changing the drawing or caller outputs. A successful insertion is one Undo operation and raises Change without NewUserItems. Application event-handler exceptions propagate after insertion.

Numeric changes from v2

Update the following variables, arrays or event-data receivers when porting the corresponding v2 call. The declarations above show the current overloads.

Parameter or value	v2 type	v3 type
x1	Single()	Double()
y1	Single()	Double()
z1	Single()	Double()
iClr	Short	Integer
iWidth	Short	Integer
iStyle	Short	Integer
iLayer	Short	Integer

For ByRef parameters, use a variable with the exact v3 type. Preserve array dimensions and the documented indexing. Moving from v2 to v3 explains conversion and legacy file-storage rules.

Example

Setup

Add an Avax.cAvax Windows Forms control named `cAvax1`. Paste the procedures into the form class. Example setup and supporting files explains references, initialization and file paths.

Add these Windows Forms controls or components to the form. Fields referenced by Handles clauses must be declared WithEvents, as the Visual Studio designer normally does.

Name	Type
Command1	Button

VB.NET example

```
Private Sub Form1_Load(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs) Handles
    MyBase.Load
    cAvax1.StartAvax()
End Sub
Private Sub Form1_FormClosed(ByVal eventSender As System.Object, ByVal eventArgs As
    System.Windows.Forms.FormClosedEventArgs) Handles Me.FormClosed
    cAvax1.EndAvax()
End Sub
Private Sub Command1_Click(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs)
    Handles Command1.Click
    Dim x1() As Double = Nothing
    Dim y1() As Double = Nothing
    Dim z1() As Double = Nothing
    Dim vItemProperties As Object = Nothing
    Dim iClr As Integer
    Dim iWidth As Integer
    Dim iStyle As Integer
    Dim iLayer As Integer
    Dim AvaxHandle As Integer
    ReDim x1(4)
    ReDim y1(4)
    ReDim z1(4)
    'Points for the spline nodes definition
    x1(1) = 1 : y1(1) = 4 : z1(1) = 0
    x1(2) = 3 : y1(2) = 1 : z1(2) = 0
    x1(3) = 6 : y1(3) = 5.25 : z1(3) = 0
    x1(4) = 12 : y1(4) = 0 : z1(4) = 0
    'Color and width definition
    iClr = 5
```

```
    iWidth = 1
    AvaxHandle = cAvax1.Add_CSpline2(x1, y1, z1, vItemProperties, iClr, iWidth, iStyle, iLayer)
End Sub
```

Add_CurveArc

cAvax method

Purpose

Adds an analytic counterclockwise circular arc from a start point, end direction and center.

See Arc commands and segmentation for the three arc pick orders, standalone completion, polyline continuation and arc resolution settings.

VB.NET declarations

```
Function Add_CurveArc(ByRef xStart As Double, ByRef yStart As Double, ByRef zStart As Double, ByRef xEnd
    As Double, ByRef yEnd As Double, ByRef zEnd As Double, ByRef xc As Double, ByRef yc As Double, ByRef zc As
     Double) As Integer

Function Add_CurveArc(ByRef xStart As Double, ByRef yStart As Double, ByRef zStart As Double, ByRef xEnd
    As Double, ByRef yEnd As Double, ByRef zEnd As Double, ByRef xc As Double, ByRef yc As Double, ByRef zc As
     Double, ByRef vItemProperties As Object, ByRef iClr As Integer, ByRef iWidth As Integer, ByRef iStyle As
    Integer, ByRef iLayer As Integer) As Integer
```

Parameters

Name	Type and passing	Meaning
xStart	ByRef xStart As Double	Is the x coordinate of the first point of the arc
yStart	ByRef yStart As Double	Is the y coordinate of the first point of the arc
zStart	ByRef zStart As Double	Is the z coordinate of the first point of the arc
xEnd	ByRef xEnd As Double	Is the x coordinate of the second point of the arc
yEnd	ByRef yEnd As Double	Is the y coordinate of the second point of the arc
zEnd	ByRef zEnd As Double	Is the z coordinate of the second point of the arc
xc	ByRef xc As Double	Is the x coordinate of the point to the center of the arc
yc	ByRef yc As Double	Is the y coordinate of the point to the center of the arc
zc	ByRef zc As Double	Is the z coordinate of the point to the center of the arc
vItemProperties	ByRef vItemProperties As Object	One-dimensional String() or Object() containing strings/Nothing, with logical properties starting at index 1. The control stores its own copy and retains the original caller reference. Nothing means no properties, except Add_Data which requires an array.
iClr	ByRef iClr As Integer	-1 uses the corresponding current pen setting. Other negative values become 0. Positive values use Integer storage. When declared ByRef, the effective value is written back only after successful validation.
iWidth	ByRef iWidth As Integer	-1 uses the current pen. Nonpositive resolved widths become 1, as in R2. Positive values use Integer storage. When declared ByRef, the effective value is written back only after successful validation.
iStyle	ByRef iStyle As Integer	-1 uses the corresponding current pen setting. Other negative values become 0. Positive values use Integer storage. When declared ByRef, the effective value is written back only after successful validation.
iLayer	ByRef iLayer As Integer	-1 uses the corresponding current pen setting. Other negative values become 0. Positive values use Integer storage. When declared ByRef, the effective value is written back only after successful validation.

Use the declaration for the overload you call. ByRef arguments can receive outputs; declare variables and arrays with the exact indicated type. Optional defaults are shown in the declaration.

Return value

Integer - An Avax handle on success, zero on error.

Remarks

The start point determines the radius; the end point supplies the terminal direction. Unequal endpoint radii do not distort the circle. The stored XYZ defining coordinates are retained. Coincident positive endpoint rays produce a complete Ellipse_i circle at the center elevation; an endpoint at the center is invalid. CurveArcOn and segment-resolution properties do not change this explicit analytic method.

The short overload uses the current pen. The full overload accepts copied String metadata and Integer color, width, line-style and layer attributes; -1 inherits the current pen. Resolved ByRef pen attributes are returned on success. A started, enabled control is required. Returns the new Integer handle, or 0 when input cannot be accepted. Preparation failures preserve drawing, history and caller arguments. Success is one Undo operation with Change and no NewUserItems. Application event-handler exceptions propagate after insertion.

Numeric changes from v2

Update the following variables, arrays or event-data receivers when porting the corresponding v2 call. The declarations above show the current overloads.

Parameter or value	v2 type	v3 type
xStart	Single	Double
yStart	Single	Double
zStart	Single	Double
xEnd	Single	Double
yEnd	Single	Double
zEnd	Single	Double
xc	Single	Double
yc	Single	Double
zc	Single	Double
iClr	Short	Integer
iWidth	Short	Integer
iStyle	Short	Integer
iLayer	Short	Integer

For ByRef parameters, use a variable with the exact v3 type. Preserve array dimensions and the documented indexing. Moving from v2 to v3 explains conversion and legacy file-storage rules.

Example

Setup

Add an Avax.cAvax Windows Forms control named `cAvax1`. Paste the procedures into the form class. Example setup and supporting files explains references, initialization and file paths.

Add these Windows Forms controls or components to the form. Fields referenced by Handles clauses must be declared WithEvents, as the Visual Studio designer normally does.

Name	Type
Button1	Button

VB.NET example

```
Private Sub Form1_Load(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs) Handles
    MyBase.Load
```

```
    Call cAvax1.StartAvax()
End Sub
Private Sub Form1_FormClosed(ByVal eventSender As System.Object, ByVal eventArgs As
    System.Windows.Forms.FormClosedEventArgs) Handles Me.FormClosed
    Call cAvax1.EndAvax()
End Sub
Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
    Button1.Click
    Dim xStart As Double
    Dim yStart As Double
    Dim zStart As Double
    Dim xEnd As Double
    Dim yEnd As Double
    Dim zEnd As Double
    Dim xc As Double
    Dim yc As Double
    Dim zc As Double
    Dim vItemProperties As Object = Nothing
    Dim iClr As Integer
    Dim iWidth As Integer
    Dim iStyle As Integer
    Dim iLayer As Integer
    Dim Answer As Integer
    xStart = 10
    yStart = 10
    zStart = 0
    xEnd = 40
    yEnd = 40
    zEnd = 0
    xc = 20
    yc = 30
    zc = 0
    Answer = CAvax1.Add_CurveArc(xStart, yStart, zStart, xEnd, yEnd, zEnd, xc, yc, zc, vItemProperties,
    iClr, iWidth, iStyle, iLayer)
    CAvax1.Command = Avax.AvaxCommand.AutoLimits_c
End Sub
```

Add_Data

cAvax method

Purpose

Adds a non-visible data object that holds application properties. It is stored with the drawing like the visible objects.

VB.NET declaration

```
Function Add_Data(ByRef vItemProperties As Object) As Integer
```

Parameters

Name	Type and passing	Meaning
vItemProperties	ByRef vItemProperties As Object	One-dimensional String() or Object() containing strings/Nothing, with logical properties starting at index 1. The control stores its own copy and retains the original caller reference. Nothing means no properties, except Add_Data which requires an array.

Use the declaration for the overload you call. ByRef arguments can receive outputs; declare variables and arrays with the exact indicated type. Optional defaults are shown in the declaration.

Return value

Integer - An Avax handle on success, zero on error.

Example

Setup

Add an Avax.cAvax Windows Forms control named cAvax1. Paste the procedures into the form class. Example setup and supporting files explains references, initialization and file paths.

Add these Windows Forms controls or components to the form. Fields referenced by Handles clauses must be declared WithEvents, as the Visual Studio designer normally does.

Name	Type
Command1	Button

VB.NET example

```
Private Sub Form1_Load(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs) Handles
    MyBase.Load
    cAvax1.StartAvax()
    Call cAvax1.Add_Line(0, 0, 0, 12, 0, 0)
End Sub
Private Sub Form1_FormClosed(ByVal eventSender As System.Object, ByVal eventArgs As
    System.Windows.Forms.FormClosedEventArgs) Handles Me.FormClosed
    cAvax1.EndAvax()
End Sub
Private Sub Command1_Click(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs)
    Handles Command1.Click
    Dim sItemProperties(2) As String
    Dim AvaxHandle As Integer
    Dim sMsg As String = String.Empty
    sItemProperties(1) = "Data 1..."
    sItemProperties(2) = "Data 2..."
    AvaxHandle = cAvax1.Add_Data(sItemProperties)
    sMsg = "Now drawing has " & cAvax1.MaxItems & " objects." & vbCrLf
    sMsg = sMsg & "One [Line] and one invisible [Data] object."
    MsgBox(sMsg, MsgBoxStyle.Information, "")
    Me.Close()
```

```
End Sub
```

Add_Dim

cAvax method

Purpose

Adds a linear dimension between two points, using the direction point and offset to position the dimension line and label.

VB.NET declarations

```
Function Add_Dim(ByRef x1 As Double, ByRef y1 As Double, ByRef x2 As Double, ByRef y2 As Double, ByRef
    OffsetD As Double, ByRef xDir As Double, ByRef yDir As Double) As Boolean

Function Add_Dim(ByRef x1 As Double, ByRef y1 As Double, ByRef x2 As Double, ByRef y2 As Double, ByRef
    OffsetD As Double, ByRef xDir As Double, ByRef yDir As Double, ByRef vItemProperties As Object, ByRef
    vRetHandles As Object, ByRef iClr As Integer, ByRef iWidth As Integer, ByRef iStyle As Integer, ByRef
    iLayer As Integer) As Boolean
```

Parameters

Name	Type and passing	Meaning
x1	ByRef x1 As Double	Is the x coordinate of the line's start point
y1	ByRef y1 As Double	Is the y coordinate of the line's start point
x2	ByRef x2 As Double	Is the x coordinate of the line's end point
y2	ByRef y2 As Double	Is the y coordinate of the line's end point
OffsetD	ByRef OffsetD As Double	Positive Double offset distance from the source line. Zero/negative values return False without inserting geometry, matching R2. Positive offsets below coordinate storage precision fail during preparation.
xDir	ByRef xDir As Double	Is the x coordinate of a point to determine the direction of the dimension line
yDir	ByRef yDir As Double	The Y coordinate of a point choosing the side of the dimension line (the original CHM incorrectly says X).
vItemProperties	ByRef vItemProperties As Object	A zero-based array with logical property strings starting at index 1. String() and string-valued Object() arrays are copied; Nothing is allowed. Non-string entries are rejected before editing.
vRetHandles	ByRef vRetHandles As Object	On normal success, an Integer() stored in Object with index 0 reserved and all handles from this operation. Active preparation/empty-result failure returns {0}. Early stopped/disabled/disposed rejection leaves it unchanged. A thrown host callback exception does not provide normal ByRef copy-out.
iClr	ByRef iClr As Integer	Integer color index; -1 uses PenColor. The effective value is copied back after normal success.
iWidth	ByRef iWidth As Integer	Integer line width; -1 uses PenWidth, zero normalizes to 1. The effective value is copied back after normal success.
iStyle	ByRef iStyle As Integer	Integer line-style index; -1 uses PenLineStyle. The effective value is copied back after normal success.
iLayer	ByRef iLayer As Integer	Integer layer index; -1 uses PenLayer. Values are not restricted to the former Integer range. The effective value is copied back after normal success.

Use the declaration for the overload you call. ByRef arguments can receive outputs; declare variables and arrays with the exact indicated type. Optional defaults are shown in the declaration.

Return value

Boolean - True on success, false on error.

Remarks

The endpoints define the measured segment. (xDir, yDir) chooses the side of that segment, and positive OffsetD sets the dimension-line distance. The label uses the current dimension factor and formatting properties.

The result consists of ordinary drawing objects for the dimension lines, marks and label. The full overload returns all their handles as a one-based Integer() stored in Object. Dimension properties selected through AvaxProp control the label format, size, marks and extension lines. Coordinates and distances are Double; pen indices are Integer. Use -1 to inherit a current pen attribute. Resolved ByRef pen values are returned on successful completion. The control must be started and enabled. PauseAvax and LockEdit do not prevent these direct additions. The complete result is prepared before insertion and committed as one Undo operation, with Change and no NewUserItems notification. Invalid or degenerate geometry returns False without inserting a partial dimension. An active full call initializes vRetHandles to {0}; an unavailable control leaves it unchanged. Application event-handler exceptions propagate after committed creation.

Numeric changes from v2

Update the following variables, arrays or event-data receivers when porting the corresponding v2 call. The declarations above show the current overloads.

Parameter or value	v2 type	v3 type
x1	Single	Double
y1	Single	Double
x2	Single	Double
y2	Single	Double
OffsetD	Single	Double
xDir	Single	Double
yDir	Single	Double
iClr	Short	Integer
iWidth	Short	Integer
iStyle	Short	Integer
iLayer	Short	Integer

For ByRef parameters, use a variable with the exact v3 type. Preserve array dimensions and the documented indexing. Moving from v2 to v3 explains conversion and legacy file-storage rules.

Example

Setup

Add an Avax.cAvax Windows Forms control named `cAvax1`. Paste the procedures into the form class. Example setup and supporting files explains references, initialization and file paths.

Add these Windows Forms controls or components to the form. Fields referenced by Handles clauses must be declared WithEvents, as the Visual Studio designer normally does.

Name	Type
Command1	Button

VB.NET example

```
Private Sub Form1_Load(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs) Handles
    MyBase.Load
    cAvax1.StartAvax()
    Call cAvax1.SetAvaxProperty(Avax.AvaxProp.DimTextScale, 5.0!)
    Call cAvax1.SetAvaxProperty(Avax.AvaxProp.DimTextAbove, True)
End Sub
Private Sub Form1_FormClosed(ByVal eventSender As System.Object, ByVal eventArgs As
    System.Windows.Forms.FormClosedEventArgs) Handles Me.FormClosed
```

```
    cAvax1.EndAvax()
End Sub
Private Sub Command1_Click(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs)
    Handles Command1.Click
    Dim x1 As Double
    Dim y1 As Double
    Dim x2 As Double
    Dim y2 As Double
    Dim OffsetD As Double
    Dim xDir As Double
    Dim yDir As Double
    Dim vItemProperties As Object = Nothing
    Dim vRetHandles As Object = Nothing
    Dim iClr As Integer
    Dim iWidth As Integer
    Dim iStyle As Integer
    Dim iLayer As Integer
    Dim AvaxHandle As Boolean
    x1 = 0 : y1 = 5
    x2 = 10 : y2 = 5
    OffsetD = 1
    iClr = 1
    AvaxHandle = cAvax1.Add_Dim(x1, y1, x2, y2, OffsetD, xDir, yDir, vItemProperties, vRetHandles, iClr,
    iWidth, iStyle, iLayer)
End Sub
```

Add_Dim_Aligned

Purpose

Adds one editable aligned dimension measuring the distance between two points, and returns its handle.

VB.NET declaration

```
Function Add_Dim_Aligned(ByVal x1 As Double, ByVal y1 As Double, ByVal x2 As Double, ByVal y2 As Double,
    ByVal dx As Double, ByVal dy As Double, ByVal textH As Double, Optional ByVal tx As Nullable(Of Double) =
    Nothing, Optional ByVal ty As Nullable(Of Double) = Nothing, Optional ByVal options As
    AvaxDimensionOptions = Nothing) As Integer
```

Parameters

Parameter	Meaning
x1	X of the first measured point (or first angular ray point).
y1	Y of the first measured point (or first angular ray point).
x2	X of the second measured point (or second angular ray point).
y2	Y of the second measured point (or second angular ray point).
dx	X of a point on the dimension line; controls its offset.
dy	Y of a point on the dimension line; controls its offset.
textH	0 inherits effective TextHeight; a positive value overrides it before OverallScale. Negative or nonfinite values are invalid.
tx	Manual text-position X. For Linear/Aligned supply both nullable tx and ty, or omit both for automatic placement.
ty	Manual text-position Y; paired with tx.
options	Detached options, copied during creation. Nothing uses native style defaults. Pass Get_DimensionDefaults(kind) to copy existing Dim properties.

Return value and behavior

A positive handle for one native dimension; 0 if creation fails.

For manual circular text at the center, supply an explicit angle. Existing geometry is not automatically associated with the dimension.

Example

Call this procedure on the owning UI thread with a control already initialized by `StartAvax()`. Handles must belong to that control. File paths are supplied by the caller.

```
Option Strict On
Imports System
Imports Avax

Public Module Help_Add_Dim_Aligned
    Public Sub Run(cad As cAvax)
        Dim defaults = cad.Get_DimensionDefaults(AvaxDimensionKind.Aligned)
        Dim options = defaults.Clone()
        options.TextOverride = "Length <> mm"
        options.TextAttachment = AvaxDimTextAttachment.TopCenter
        options.LineSpacingStyle = AvaxDimLineSpacing.Exact
        options.LineSpacingFactor = 1.25
        Dim first As New AvaxDimensionPoint(0, 0)
        Dim second As New AvaxDimensionPoint(80, 60)
        Dim handle As Integer = cad.Add_Dim_Aligned(first.X, first.Y, second.X, second.Y, 10, 80, 0,
    options:=options)
        If handle = 0 Then Throw New InvalidOperationException("Dimension creation failed.")
    End Sub
End Module
```

Related options and reference

Add_Dim_Angular

Purpose

Adds one editable angular dimension between two rays about a vertex, and returns its handle.

VB.NET declaration

```
Function Add_Dim_Angular(ByVal x0 As Double, ByVal y0 As Double, ByVal x1 As Double, ByVal y1 As Double,
    ByVal x2 As Double, ByVal y2 As Double, ByVal ax As Double, ByVal ay As Double, ByVal tx As Double, ByVal
    ty As Double, ByVal textH As Double, Optional ByVal options As AvaxDimensionOptions = Nothing) As Integer

Function Add_Dim_Angular(ByVal x0 As Double, ByVal y0 As Double, ByVal x1 As Double, ByVal y1 As Double,
    ByVal x2 As Double, ByVal y2 As Double, ByVal ax As Double, ByVal ay As Double, ByVal textH As Double,
    Optional ByVal options As AvaxDimensionOptions = Nothing) As Integer
```

Parameters

Parameter	Meaning
x0	X of angular vertex.
y0	Y of angular vertex.
x1	X of the first measured point (or first angular ray point).
y1	Y of the first measured point (or first angular ray point).
x2	X of the second measured point (or second angular ray point).
y2	Y of the second measured point (or second angular ray point).
ax	X of the arc-position point; determines angular radius and sector.
ay	Y of the arc-position point.
tx	Manual text-position X. For Linear/Aligned supply both nullable tx and ty, or omit both for automatic placement.
ty	Manual text-position Y; paired with tx.
textH	0 inherits effective TextHeight; a positive value overrides it before OverallScale. Negative or nonfinite values are invalid.
options	Detached options, copied during creation. Nothing uses native style defaults. Pass Get_DimensionDefaults(kind) to copy existing Dim properties.

Return value and behavior

A positive handle for one native dimension; 0 if creation fails.

For manual circular text at the center, supply an explicit angle. Existing geometry is not automatically associated with the dimension.

Example

Call this procedure on the owning UI thread with a control already initialized by `StartAvax()`. Handles must belong to that control. File paths are supplied by the caller.

```
Option Strict On
Imports System
Imports Avax

Public Module Help_Add_Dim_Angular
    Public Sub Run(cad As cAvax)
        Dim options = cad.Get_DimensionDefaults(AvaxDimensionKind.Angular)
        Dim handle As Integer = cad.Add_Dim_Angular(0, 0, 30, 0, 0, 30, 20, 20, textH:=0,
    options:=options)
        If handle = 0 Then Throw New InvalidOperationException("Dimension creation failed.")
    End Sub
End Module
```

Related options and reference

Add_Dim_Diametric

Purpose

Adds one editable diameter dimension for a circle, and returns its handle. Supply the radius, not the diameter.

VB.NET declaration

```
Function Add_Dim_Diametric(ByVal cx As Double, ByVal cy As Double, ByVal radius As Double, ByVal textH As
    Double, Optional ByVal angleDeg As Double = 0, Optional ByVal leaderLength As Nullable(Of Double) =
    Nothing, Optional ByVal options As AvaxDimensionOptions = Nothing) As Integer

Function Add_Dim_Diametric(ByVal cx As Double, ByVal cy As Double, ByVal radius As Double, ByVal tx As
    Double, ByVal ty As Double, ByVal textH As Double, Optional ByVal angleDeg As Nullable(Of Double) =
    Nothing, Optional ByVal leaderLength As Nullable(Of Double) = Nothing, Optional ByVal options As
    AvaxDimensionOptions = Nothing) As Integer
```

Parameters

Parameter	Meaning
cx	Circle center X.
cy	Circle center Y.
radius	Positive circle radius, also for Diametric. Diametric measures twice this value.
textH	0 inherits effective TextHeight; a positive value overrides it before OverallScale. Negative or nonfinite values are invalid.
angleDeg	Dimension-axis direction in degrees. Linear: 0 is horizontal, 90 vertical. Circular dimensions: optional manual overload derives it from center to text when Nothing; the automatic overload defaults to 0.
leaderLength	Optional finite, nonnegative leader length in drawing units. A positive value requests outside text unless ForceTextInside is True; 0 allows automatic inside fitting. Nothing derives the distance when outside text is needed.
options	Detached options, copied during creation. Nothing uses native style defaults. Pass Get_DimensionDefaults(kind) to copy existing Dim properties.
tx	Manual text-position X. For Linear/Aligned supply both nullable tx and ty, or omit both for automatic placement.
ty	Manual text-position Y; paired with tx.

Return value and behavior

A positive handle for one native dimension; 0 if creation fails.

For manual circular text at the center, supply an explicit angle. Existing geometry is not automatically associated with the dimension.

Example

Call this procedure on the owning UI thread with a control already initialized by `StartAvax()`. Handles must belong to that control. File paths are supplied by the caller.

```
Option Strict On
Imports System
Imports Avax

Public Module Help_Add_Dim_Diametric
    Public Sub Run(cad As cAvax)
        Dim options = cad.Get_DimensionDefaults(AvaxDimensionKind.Diametric)
        Dim handle As Integer = cad.Add_Dim_Diametric(cx:=40, cy:=40, radius:=20, tx:=75, ty:=65,
    textH:=0, options:=options)
        If handle = 0 Then Throw New InvalidOperationException("Dimension creation failed.")
    End Sub
End Module
```

Related options and reference

Add_Dim_Linear

Purpose

Adds one editable dimension measuring the projection of two points onto a chosen axis, and returns its handle.

VB.NET declaration

```
Function Add_Dim_Linear(ByVal x1 As Double, ByVal y1 As Double, ByVal x2 As Double, ByVal y2 As Double,
    ByVal dx As Double, ByVal dy As Double, ByVal textH As Double, Optional ByVal angleDeg As Double = 0,
    Optional ByVal tx As Nullable(Of Double) = Nothing, Optional ByVal ty As Nullable(Of Double) = Nothing,
    Optional ByVal options As AvaxDimensionOptions = Nothing) As Integer
```

Parameters

Parameter	Meaning
x1	X of the first measured point (or first angular ray point).
y1	Y of the first measured point (or first angular ray point).
x2	X of the second measured point (or second angular ray point).
y2	Y of the second measured point (or second angular ray point).
dx	X of a point on the dimension line; controls its offset.
dy	Y of a point on the dimension line; controls its offset.
textH	0 inherits effective TextHeight; a positive value overrides it before OverallScale. Negative or nonfinite values are invalid.
angleDeg	Dimension-axis direction in degrees. Linear: 0 is horizontal, 90 vertical. Circular dimensions: optional manual overload derives it from center to text when Nothing; the automatic overload defaults to 0.
tx	Manual text-position X. For Linear/Aligned supply both nullable tx and ty, or omit both for automatic placement.
ty	Manual text-position Y; paired with tx.
options	Detached options, copied during creation. Nothing uses native style defaults. Pass Get_DimensionDefaults(kind) to copy existing Dim properties.

Return value and behavior

A positive handle for one native dimension; 0 if creation fails.

For manual circular text at the center, supply an explicit angle. Existing geometry is not automatically associated with the dimension.

Example

Call this procedure on the owning UI thread with a control already initialized by `StartAvax()`. Handles must belong to that control. File paths are supplied by the caller.

```
Option Strict On
Imports System
Imports Avax

Public Module Help_Add_Dim_Linear
    Public Sub Run(cad As cAvax)
        Dim options = cad.Get_DimensionDefaults(AvaxDimensionKind.Linear)
        Dim handle As Integer = cad.Add_Dim_Linear(0, 0, 100, 30, 0, 45, 0, angleDeg:=0, options:=options)
        If handle = 0 Then Throw New InvalidOperationException("Dimension creation failed.")
    End Sub
End Module
```

Related options and reference

Add_Dim_Radial

Purpose

Adds one editable radius dimension for a circle, and returns its handle.

VB.NET declaration

```
Function Add_Dim_Radial(ByVal cx As Double, ByVal cy As Double, ByVal radius As Double, ByVal textH As
    Double, Optional ByVal angleDeg As Double = 0, Optional ByVal leaderLength As Nullable(Of Double) =
    Nothing, Optional ByVal options As AvaxDimensionOptions = Nothing) As Integer

Function Add_Dim_Radial(ByVal cx As Double, ByVal cy As Double, ByVal radius As Double, ByVal tx As
    Double, ByVal ty As Double, ByVal textH As Double, Optional ByVal angleDeg As Nullable(Of Double) =
    Nothing, Optional ByVal leaderLength As Nullable(Of Double) = Nothing, Optional ByVal options As
    AvaxDimensionOptions = Nothing) As Integer
```

Parameters

Parameter	Meaning
cx	Circle center X.
cy	Circle center Y.
radius	Positive circle radius, also for Diametric. Diametric measures twice this value.
textH	0 inherits effective TextHeight; a positive value overrides it before OverallScale. Negative or nonfinite values are invalid.
angleDeg	Dimension-axis direction in degrees. Linear: 0 is horizontal, 90 vertical. Circular dimensions: optional manual overload derives it from center to text when Nothing; the automatic overload defaults to 0.
leaderLength	Optional finite, nonnegative leader length in drawing units. A positive value requests outside text unless ForceTextInside is True; 0 allows automatic inside fitting. Nothing derives the distance when outside text is needed.
options	Detached options, copied during creation. Nothing uses native style defaults. Pass Get_DimensionDefaults(kind) to copy existing Dim properties.
tx	Manual text-position X. For Linear/Aligned supply both nullable tx and ty, or omit both for automatic placement.
ty	Manual text-position Y; paired with tx.

Return value and behavior

A positive handle for one native dimension; 0 if creation fails.

For manual circular text at the center, supply an explicit angle. Existing geometry is not automatically associated with the dimension.

Example

Call this procedure on the owning UI thread with a control already initialized by `StartAvax()`. Handles must belong to that control. File paths are supplied by the caller.

```
Option Strict On
Imports System
Imports Avax

Public Module Help_Add_Dim_Radial
    Public Sub Run(cad As cAvax)
        Dim options = cad.Get_DimensionDefaults(AvaxDimensionKind.Radial)
        Dim handle As Integer = cad.Add_Dim_Radial(cx:=40, cy:=40, radius:=20, textH:=0, angleDeg:=30,
    options:=options)
        If handle = 0 Then Throw New InvalidOperationException("Dimension creation failed.")
    End Sub
End Module
```

Related options and reference

Add_Document

API reference

Creates a document drawing object with wrapped multiline text and supported AVAX document markup. Use the returned handle to edit or query the document.

Declarations

```
Public Function Add_Document(ByVal sDocumentText As String, ByRef x As Double, ByRef y As Double, ByRef z
    As Double, ByRef DocumentWidth As Double, ByRef DocumentTextHeight As Double, ByRef vItemProperties As
    Object, ByRef iFontNumber As Integer, ByRef Angle As Double, ByRef iClr As Integer, ByRef iWidth As
    Integer, ByRef iStyle As Integer, ByRef iLayer As Integer) As Integer

Public Function Add_Document(ByRef sDocumentText As String, ByRef x As Double, ByRef y As Double, ByRef z
    As Double, ByRef DocumentWidth As Double, ByRef DocumentTextHeight As Double) As Integer
```

Parameters

Name	Purpose
sDocumentText	Document source text, including line breaks and supported AVAX document markup.
x	World X position (Double).
y	World Y position (Double).
z	World Z position (Double).
DocumentWidth	Positive document layout width in drawing units (Double).
DocumentTextHeight	Positive base text height in drawing units (Double).
vItemProperties	Optional application properties stored with the drawing object; pass Nothing for no properties.
iFontNumber	Font table index (Integer). The short Add_Document overload uses font1.
Angle	Rotation angle in degrees (Double); the short Add_Document overload uses zero.
iClr	Palette index (Integer); -1 uses the current pen color.
iWidth	Line width (Integer); -1 uses the current pen width.
iStyle	Line style index (Integer); -1 uses the current pen style.
iLayer	Layer index (Integer); -1 uses the current pen layer.

Return value

The new Integer handle, or zero if the control is not started/enabled or the request is rejected. Call StartAvax before inserting documents.

Behavior

The short overload uses font1 and angle0. The method returns the created handle, correcting the discarded return value in the old short wrapper. A successful edit publishes Change, then the explicit NewUserItems notification, as in R2.

The drawing and complete reversible record exist before the notifications. A callback that supersedes the drawing or control prevents stale trailing notifications. Handler exceptions propagate after the committed edit. PauseAvaxEvents suppresses notifications without replay. BeginUpdate and EndUpdate suspend display while model notifications retain their normal timing. Invalid geometry is rejected before insertion.

VB.NET example

Pass an initialized control. Coordinates, layout width and text height are Double in v3; handles and font/layer indices are Integer.

```
Option Strict On
Imports Avax

Public Module AddDocumentExample
    Public Function Add(cad As cAvax) As Integer
```

```
        Dim text As String = "AVAX example" & vbCrLf & "Second line"
        Dim x As Double = 10.5, y As Double = 20.25, z As Double = 0
        Dim width As Double = 50, textHeight As Double = 2
        Return cad.Add_Document(text, x, y, z, width, textHeight)
    End Function
End Module
```

See Add_ShapeDocument and NewUserItems.

Add_DWG

cAvax method

Purpose

This function inserts a .DWG file into the drawing

VB.NET declarations

```
Function Add_DWG(ByRef sDWGFile As String, ByRef sDWGLayers As String, ByRef x As Double, ByRef y As
    Double, ByRef z As Double) As Boolean

Function Add_DWG(ByRef sDWGFile As String, ByRef sDWGLayers As String, ByRef x As Double, ByRef y As
    Double, ByRef z As Double, ByRef vItemProperties As Object, ByRef fNoText As Boolean, ByRef fNoBlock As
    Boolean, ByRef fNoDimension As Boolean, ByRef vRetHandles As Object) As Boolean
```

Parameters

Name	Type and passing	Meaning
sDWGFile	ByRef sDWGFile As String	Is the name of the DWG file
sDWGLayers	ByRef sDWGLayers As String	Is the names of the DWG's layers which you want to import (ex. sDWGLayers="[walls][windows]")
x	ByRef x As Double	Is the x coordinate of the insertion point
y	ByRef y As Double	Is the y coordinate of the insertion point
z	ByRef z As Double	Is the z coordinate of the insertion point
vItemProperties	ByRef vItemProperties As Object	Is a string array with properties for the new objects
fNoText	ByRef fNoText As Boolean	A Boolean expression that defines the non insertion of text
fNoBlock	ByRef fNoBlock As Boolean	A Boolean expression that defines the non insertion of blocks
fNoDimension	ByRef fNoDimension As Boolean	A Boolean expression that defines the non insertion of dimensions
vRetHandles	ByRef vRetHandles As Object	Returns an integer array with the handles of the new objects Double digit integer array with the new object handles

Use the declaration for the overload you call. ByRef arguments can receive outputs; declare variables and arrays with the exact indicated type. Optional defaults are shown in the declaration.

Return value

Boolean - True on success, false on error.

Remarks

Imports a DWG drawing into the active control. The full overload accepts a bracketed layer list such as [walls][windows], Double XYZ insertion offsets, caller String() metadata and fNoText/fNoBlock/fNoDimension filters. Layer matching is case insensitive; an empty list imports all layers. vRetHandles is an Integer() stored in Object, with index 0 reserved. A valid filter with no matches returns True and {0}. The complete import forms one undo step. Straight polylines are editable Polyline_i objects with ordered XYZ endpoints. Horizontal zero-width bulged arcs retain an analytic composite representation; tilted arcs are sampled in their source plane and transformed into world XYZ, with exact shared endpoints. MakeSv and boundary commands accept sampled polylines and imported analytic polylines containing straight and curved segments. They preserve source order, reversed bulges and closure while leaving the original curves unchanged. Actual INSERT blocks remain separate objects and are not expanded by these commands. fNoBlock excludes INSERTs, not polylines. Assigned block attributes retain their values and world coordinates. Unsupported drawing entities or invalid geometry are rejected before insertion. Nonzero polyline widths are geometric distances: constant and tapered widths follow the source plane and block transformations, while public geometry remains the editable centreline. Wide bulges are sampled, including when DXFCurveArcOn is enabled. Picking and crossing selection include the visible width. Width profiles follow node edits, stretch, trim, split, joining and closing, with Undo/Redo. Wide polylines follow numeric dash/gap/dot patterns and complex text/SHX markers. Pattern distances continue across connected segments and restart at disconnected segments; LineStyleScale changes their spacing without changing geometric width. ComplexLineTypesSupport disables markers while retaining numeric dashes. For LWPOLYLINE, an explicitly supplied vertex width of zero remains zero; it is distinct from an omitted vertex width. If a file supplies both constant and vertex widths, the importer uses the explicit vertex-width representation, including zero. New DXF/DWG writes use a single unambiguous width representation. Imported geometric-width polylines also retain their original CAD path for editable DXF/DWG re-export while it still describes the current geometry. Native .avax format 8 preserves that path together with the displayed centreline and width vectors. Geometric-width line patterns measure distance along the XYZ centreline, so tilting a path does not stretch the pattern in its own plane. Viewport clipping retains that same pattern phase. This programmatic operation publishes Change without NewUserItems, as in R2. Use its documented return value/output parameters for the result. Interactive drawing commands and explicit announcing shape/picture/document helpers retain their own notification policy. Full CIRCLE entities remain native circles with DXFCurveArcOn_p either False or True, including their thickness. Partial arcs follow the representation preference. Complete analytic arcs become circles, and sampled complete arcs have exact closing endpoints. Imported circles remain actual CAD CIRCLE entities when exported back to DXF or DWG.

Numeric changes from v2

Update the following variables, arrays or event-data receivers when porting the corresponding v2 call. The declarations above show the current overloads.

Parameter or value	v2 type	v3 type
x	Single	Double
y	Single	Double
z	Single	Double

For ByRef parameters, use a variable with the exact v3 type. Preserve array dimensions and the documented indexing. Moving from v2 to v3 explains conversion and legacy file-storage rules.

Example

Setup

Add an Avax.cAvax Windows Forms control named `cAvax1`. Paste the procedures into the form class. Example setup and supporting files explains references, initialization and file paths.

Add these Windows Forms controls or components to the form. Fields referenced by Handles clauses must be declared WithEvents, as the Visual Studio designer normally does.

Name	Type
Command1	Button

VB.NET example

```
Private Sub Form1_Load(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs) Handles
    MyBase.Load
    cAvax1.StartAvax()
End Sub
```

```
Private Sub Form1_FormClosed(ByVal eventSender As System.Object, ByVal eventArgs As
    System.Windows.Forms.FormClosedEventArgs) Handles Me.FormClosed
    cAvax1.EndAvax()
End Sub
Private Sub Command1_Click(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs)
    Handles Command1.Click
    Dim sDWGFile As String = String.Empty
    Dim sDWGLayers As String = String.Empty
    Dim x As Double
    Dim y As Double
    Dim z As Double
    Dim vItemProperties As Object = Nothing
    Dim fNoText As Boolean
    Dim fNoBlock As Boolean
    Dim fNoDimension As Boolean
    Dim vRetHandles As Object = Nothing
    Dim AvaxHandle As Boolean
    'Definition of the DXF File Name
    sDWGFile = My.Application.Info.DirectoryPath & "\..\Sample.DWG"
    ' Import all the Layers
    sDWGLayers = ""
    AvaxHandle = cAvax1.Add_DWG(sDWGFile, sDWGLayers, x, y, z, vItemProperties, fNoText, fNoBlock,
    fNoDimension, vRetHandles)
End Sub
```

Add_DXF

cAvax method

Purpose

This function inserts a .DXF file into the drawing

VB.NET declarations

```
Function Add_DXF(ByRef sDxfFile As String, ByRef sDxfLayers As String, ByRef x As Double, ByRef y As
    Double, ByRef z As Double) As Boolean

Function Add_DXF(ByRef sDxfFile As String, ByRef sDxfLayers As String, ByRef x As Double, ByRef y As
    Double, ByRef z As Double, ByRef vItemProperties As Object, ByRef fNoText As Boolean, ByRef fNoBlock As
    Boolean, ByRef fNoDimension As Boolean, ByRef vRetHandles As Object) As Boolean
```

Parameters

Name	Type and passing	Meaning
sDxfFile	ByRef sDxfFile As String	Is the name of the DXF file
sDxfLayers	ByRef sDxfLayers As String	Is the mames of the DXF's layers which you want to import (ex. sDWGLayers="[walls][windows]")
x	ByRef x As Double	Is the x coordinate of the insertion point
y	ByRef y As Double	Is the y coordinate of the insertion point
z	ByRef z As Double	Is the z coordinate of the insertion point
vItemProperties	ByRef vItemProperties As Object	Is a string array with properties for the new objects
fNoText	ByRef fNoText As Boolean	A Boolean expression that defines the non insertion of text
fNoBlock	ByRef fNoBlock As Boolean	A Boolean expression that defines the non insertion of block
fNoDimension	ByRef fNoDimension As Boolean	A Boolean expression that defines the non insertion of dimensions
vRetHandles	ByRef vRetHandles As Object	Returns an integer array with the handles of the new objects Double digit long array with the new object handles

Use the declaration for the overload you call. ByRef arguments can receive outputs; declare variables and arrays with the exact indicated type. Optional defaults are shown in the declaration.

Return value

Boolean - True on success, false on error.

Remarks

Imports an ASCII or binary DXF drawing into the active control. The full overload accepts a bracketed layer list such as [walls][windows], Double XYZ insertion offsets, caller String() metadata and fNoText/fNoBlock/fNoDimension filters. Layer matching is case insensitive; an empty list imports all layers. vRetHandles is an Integer() stored in Object, with index 0 reserved. A valid filter with no matches returns True and {0}. The complete import forms one undo step. Straight polylines are editable Polyline_i objects. With DXFCurveArcOn_p=False, sampled bulged arcs join the same ordered polyline with exact shared XYZ endpoints; CircleSectors_p and the arc segmentation parameters control the approximation. With DXFCurveArcOn_p=True, horizontal zero-width bulged arcs retain an analytic composite representation. Tilted arcs are sampled in their source plane and transformed into world XYZ. MakeSv and boundary commands accept sampled polylines and imported analytic polylines containing straight and curved segments. They preserve source order, reversed bulges and closure while leaving the original curves unchanged. Actual INSERT blocks remain separate objects and are not expanded by these commands. fNoBlock excludes INSERTs, not polylines. Unsupported drawing entities or invalid geometry are rejected before insertion. Nonzero polyline widths are geometric distances: constant and tapered widths follow the source plane and block transformations, while public geometry remains the editable centreline. Wide bulges are sampled, including when DXFCurveArcOn is enabled. Picking and crossing selection include the visible width. Width profiles follow node edits, stretch, trim, split, joining and closing, with Undo/Redo. Wide polylines follow numeric dash/gap/dot patterns and complex text/SHX markers. Pattern distances continue across connected segments and restart at disconnected segments; LineStyleScale changes their spacing without changing geometric width. ComplexLineTypesSupport disables markers while retaining numeric dashes. For LWPOLYLINE, an explicitly supplied vertex width of zero remains zero; it is distinct from an omitted vertex width. If a file supplies both constant and vertex widths, the importer uses the explicit vertex-width representation, including zero. New DXF/DWG writes use a single unambiguous width representation. Imported geometric-width polylines also retain their original CAD path for editable DXF/DWG re-export while it still describes the current geometry. Native .avax format 8 preserves that path together with the displayed centreline and width vectors. Geometric-width line patterns measure distance along the XYZ centreline, so tilting a path does not stretch the pattern in its own plane. Viewport clipping retains that same pattern phase. This programmatic operation publishes Change without NewUserItems, as in R2. Use its documented return value/output parameters for the result. Interactive drawing commands and explicit announcing shape/picture/document helpers retain their own notification policy. Full CIRCLE entities remain native circles with DXFCurveArcOn_p either False or True, including their thickness. Partial arcs follow the representation preference. Complete analytic arcs become circles, and sampled complete arcs have exact closing endpoints. Imported circles remain actual CAD CIRCLE entities when exported back to DXF or DWG.

Numeric changes from v2

Update the following variables, arrays or event-data receivers when porting the corresponding v2 call. The declarations above show the current overloads.

Parameter or value	v2 type	v3 type
x	Single	Double
y	Single	Double
z	Single	Double

For ByRef parameters, use a variable with the exact v3 type. Preserve array dimensions and the documented indexing. Moving from v2 to v3 explains conversion and legacy file-storage rules.

Example

Setup

Add an Avax.cAvax Windows Forms control named `cAvax1`. Paste the procedures into the form class. Example setup and supporting files explains references, initialization and file paths.

Add these Windows Forms controls or components to the form. Fields referenced by Handles clauses must be declared WithEvents, as the Visual Studio designer normally does.

Name	Type
Command1	Button

VB.NET example

```
Private Sub Form1_Load(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs) Handles
    MyBase.Load
    cAvax1.StartAvax()
```

```
End Sub
Private Sub Form1_FormClosed(ByVal eventSender As System.Object, ByVal eventArgs As
    System.Windows.Forms.FormClosedEventArgs) Handles Me.FormClosed
    cAvax1.EndAvax()
End Sub
Private Sub Command1_Click(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs)
    Handles Command1.Click
    Dim sDxfFile As String = String.Empty
    Dim sDxfLayers As String = String.Empty
    Dim x As Double
    Dim y As Double
    Dim z As Double
    Dim vItemProperties As Object = Nothing
    Dim fNoText As Boolean
    Dim fNoBlock As Boolean
    Dim fNoDimension As Boolean
    Dim vRetHandles As Object = Nothing
    Dim AvaxHandle As Boolean
    'Definition of the DXF File Name
    sDxfFile = My.Application.Info.DirectoryPath & "\..\Sample.DXF"
    ' Import all the Layers
    sDxfLayers = ""
    AvaxHandle = cAvax1.Add_DXF(sDxfFile, sDxfLayers, x, y, z, vItemProperties, fNoText, fNoBlock,
    fNoDimension, vRetHandles)
End Sub
```

Add_Ellipse

cAvax method

Purpose

This function adds an ellipse into the drawing

See Circle and Ellipse commands for center/radius, diameter and three-point circles, ellipse radii, Double precision and one-shot command completion.

VB.NET declarations

```
Function Add_Ellipse(ByRef xc1 As Double, ByRef yc1 As Double, ByRef zc1 As Double, ByRef aR As Double,
    ByRef bR As Double, ByRef vItemProperties As Object, ByRef iClr As Integer, ByRef iWidth As Integer, ByRef
     iStyle As Integer, ByRef iLayer As Integer) As Integer

Function Add_Ellipse(ByRef xc1 As Double, ByRef yc1 As Double, ByRef zc1 As Double, ByRef aR As Double,
    ByRef bR As Double) As Integer
```

Parameters

Name	Type and passing	Meaning
xc1	ByRef xc1 As Double	Is the x coordinate of the center of the ellipse
yc1	ByRef yc1 As Double	Is the y coordinate of the center of the ellipse
zc1	ByRef zc1 As Double	Is the z coordinate of the center of the ellipse
aR	ByRef aR As Double	Is the ellipse's horizontal radius
bR	ByRef bR As Double	Is the ellipse's vertical radius
vItemProperties	ByRef vItemProperties As Object	One-dimensional String() or Object() containing strings/Nothing, with logical properties starting at index 1. The control stores its own copy and retains the original caller reference. Nothing means no properties, except Add_Data which requires an array.
iClr	ByRef iClr As Integer	-1 uses the corresponding current pen setting. Other negative values become 0. Positive values use Integer storage. When declared ByRef, the effective value is written back only after successful validation.
iWidth	ByRef iWidth As Integer	-1 uses the current pen. Nonpositive resolved widths become 1, as in R2. Positive values use Integer storage. When declared ByRef, the effective value is written back only after successful validation.
iStyle	ByRef iStyle As Integer	-1 uses the corresponding current pen setting. Other negative values become 0. Positive values use Integer storage. When declared ByRef, the effective value is written back only after successful validation.
iLayer	ByRef iLayer As Integer	-1 uses the corresponding current pen setting. Other negative values become 0. Positive values use Integer storage. When declared ByRef, the effective value is written back only after successful validation.

Use the declaration for the overload you call. ByRef arguments can receive outputs; declare variables and arrays with the exact indicated type. Optional defaults are shown in the declaration.

Return value

Integer - An Avax handle on success, zero on error.

Numeric changes from v2

Update the following variables, arrays or event-data receivers when porting the corresponding v2 call. The declarations above show the current overloads.

Parameter or value	v2 type	v3 type
xc1	Single	Double
yc1	Single	Double
zc1	Single	Double
aR	Single	Double
bR	Single	Double
iClr	Short	Integer
iWidth	Short	Integer
iStyle	Short	Integer
iLayer	Short	Integer

For ByRef parameters, use a variable with the exact v3 type. Preserve array dimensions and the documented indexing. Moving from v2 to v3 explains conversion and legacy file-storage rules.

Example

Setup

Add an Avax.cAvax Windows Forms control named `cAvax1`. Paste the procedures into the form class. Example setup and supporting files explains references, initialization and file paths.

Add these Windows Forms controls or components to the form. Fields referenced by Handles clauses must be declared WithEvents, as the Visual Studio designer normally does.

Name	Type
Command1	Button

VB.NET example

```
Private Sub Form1_Load(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs) Handles
    MyBase.Load
    cAvax1.StartAvax()
End Sub
Private Sub Form1_FormClosed(ByVal eventSender As System.Object, ByVal eventArgs As
    System.Windows.Forms.FormClosedEventArgs) Handles Me.FormClosed
    cAvax1.EndAvax()
End Sub
Private Sub Command1_Click(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs)
    Handles Command1.Click
    Dim xc1 As Double
    Dim yc1 As Double
    Dim zc1 As Double
    Dim aR As Double
    Dim bR As Double
    Dim vItemProperties As Object = Nothing
    Dim iClr As Integer
    Dim iWidth As Integer
    Dim iStyle As Integer
    Dim iLayer As Integer
    Dim AvaxHandle As Integer
    'Define the Center of the Ellipse
    xc1 = 8 : yc1 = 2 : zc1 = 0
    ' Define the lengths of two axis
    aR = 6 : bR = 3.5
    ' Define the Color
    iClr = 6
    AvaxHandle = cAvax1.Add_Ellipse(xc1, yc1, zc1, aR, bR, vItemProperties, iClr, iWidth, iStyle, iLayer)
End Sub
```

Add_ExternalLines

cAvax method

Purpose

Adds a drawing reference to named host-supplied geometry. SetExternalLines provides the source line coordinates when requested.

VB.NET declarations

```
Function Add_ExternalLines(ByRef sName As String, ByRef vItemProperties As Object, ByRef iClr As Integer,
    ByRef iWidth As Integer, ByRef iStyle As Integer, ByRef iLayer As Integer) As Integer

Function Add_ExternalLines(ByRef sName As String) As Integer
```

Parameters

Name	Type and passing	Meaning
sName	ByRef sName As String	Host source name, distinct from the independently assignable handle Key. Passed ByRef; callback renaming writes back only after successful creation.
vItemProperties	ByRef vItemProperties As Object	Optional one-dimensional String() or Object() containing strings/Nothing. Stored as an owned snapshot; the original reference is retained.
iClr	ByRef iClr As Integer	-1 uses the corresponding current pen setting. Other negative values become 0. Positive values use Integer storage. When declared ByRef, the effective value is written back only after successful validation.
iWidth	ByRef iWidth As Integer	-1 uses the current pen. Nonpositive resolved widths become 1, as in R2. Positive values use Integer storage. When declared ByRef, the effective value is written back only after successful validation.
iStyle	ByRef iStyle As Integer	-1 uses the corresponding current pen setting. Other negative values become 0. Positive values use Integer storage. When declared ByRef, the effective value is written back only after successful validation.
iLayer	ByRef iLayer As Integer	-1 uses the corresponding current pen setting. Other negative values become 0. Positive values use Integer storage. When declared ByRef, the effective value is written back only after successful validation.

Use the declaration for the overload you call. ByRef arguments can receive outputs; declare variables and arrays with the exact indicated type. Optional defaults are shown in the declaration.

Return value

Integer - An Avax handle on success, zero on error.

Remarks

sName identifies a host source independently of the object key set by SetHandleKey. The new ExLines_i object requests its line geometry through SetExternalLines. The handler supplies equal-length Double arrays with logical segments starting at index 1. Finite coincident endpoints are allowed. A missing or malformed first result returns 0 without adding a reference. Success returns one reference handle and is one Undo operation with Change and no NewUserItems.

The source name and reference transform are stored with the drawing. The host supplies source-space coordinates on later requests; do not apply an already stored reference transform again. The control must be started and enabled. Creation returns 0 while events are paused or an external resolver is already running. Application resolver exceptions propagate.

Numeric changes from v2

Update the following variables, arrays or event-data receivers when porting the corresponding v2 call. The declarations above show the current overloads.

Parameter or value	v2 type	v3 type
iClr	Short	Integer

Parameter or value	v2 type	v3 type
iWidth	Short	Integer
iStyle	Short	Integer
iLayer	Short	Integer

For ByRef parameters, use a variable with the exact v3 type. Preserve array dimensions and the documented indexing. Moving from v2 to v3 explains conversion and legacy file-storage rules.

Example

Setup

Add an Avax.cAvax Windows Forms control named `cAvax1`. Paste the procedures into the form class. Example setup and supporting files explains references, initialization and file paths.

Add these Windows Forms controls or components to the form. Fields referenced by Handles clauses must be declared WithEvents, as the Visual Studio designer normally does.

Name	Type
Command1	Button

VB.NET example

```
Private Sub Form1_Load(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs) Handles
    MyBase.Load
    cAvax1.StartAvax()
End Sub
Private Sub Form1_FormClosed(ByVal eventSender As System.Object, ByVal eventArgs As
    System.Windows.Forms.FormClosedEventArgs) Handles Me.FormClosed
    cAvax1.EndAvax()
End Sub
Private Sub CAvax1_SetExternalLines(ByVal Sender As Object, ByVal e As
    Avax.cAvax.SetExternalLinesEventArgs) Handles cAvax1.SetExternalLines
    Select Case e.sName
        Case "My rectangle"
        Dim x1(4) As Double
        Dim y1(4) As Double
        Dim z1(4) As Double
        Dim x2(4) As Double
        Dim y2(4) As Double
        Dim z2(4) As Double
        x1(1) = 0 : y1(1) = 0 : x2(1) = 0 : y2(1) = 10
        x1(2) = 0 : y1(2) = 10 : x2(2) = 10 : y2(2) = 10
        x1(3) = 10 : y1(3) = 10 : x2(3) = 10 : y2(3) = 0
        x1(4) = 10 : y1(4) = 0 : x2(4) = 0 : y2(4) = 0
        e.x1 = x1
        e.y1 = y1
        e.z1 = z1
        e.x2 = x2
        e.y2 = y2
        e.z2 = z2
    End Select
End Sub
Private Sub Command1_Click(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs)
    Handles Command1.Click
    Dim sName As String = String.Empty
    Dim vItemProperties As Object = Nothing
    Dim iClr As Integer
    Dim iWidth As Integer
    Dim iStyle As Integer
    Dim iLayer As Integer
    Dim Answer As Integer
    sName = "My rectangle"
```

```
    iClr = 1
    iWidth = 2
    Answer = cAvax1.Add_ExternalLines(sName, vItemProperties, iClr, iWidth, iStyle, iLayer)
End Sub
```

Add_Hatch

Purpose

Create one native hatch from source handles, or from a complete explicit definition.

VB.NET declaration

```
Function Add_Hatch(ByVal boundaryHandles As Integer(), Optional ByVal options As AvaxHatchOptions =
    Nothing) As Integer

Function Add_Hatch(ByVal data As AvaxHatchData) As Integer
```

Parameters

Parameter	Meaning
boundaryHandles	Handles of closed boundary sources; include holes. Normal zero-based arrays are accepted.
options	Nothing copies Get_HatchDefaults(); explicit options are cloned. Associative defaults to True.
data	Complete AvaxHatchData with Options and Loops. Missing SourceHandles cause creation to be independent.

Return value and behavior

Positive handle on success; 0 on invalid boundaries/options or failure. Successful creation selects the new hatch. Source objects remain separate objects.

Example

Call this procedure on the owning UI thread with a control already initialized by `StartAvax()`. Handles must belong to that control. File paths are supplied by the caller.

```
Option Strict On
Imports System
Imports Avax

Public Module Help_Add_Hatch
    Public Sub Run(cad As cAvax)
        Dim outer = cad.Add_Rectangle(0, 0, 100, 60)
        Dim hole = cad.Add_Ellipse(35, 30, 0, 12, 12)
        Dim options = cad.Get_HatchDefaults()
        options.Spacing = 3
        Dim hatchHandle = cad.Add_Hatch(New Integer() {outer, hole}, options)
        If hatchHandle = 0 Then Throw New InvalidOperationException("Hatch creation failed.")
    End Sub
End Module
```

Related options and reference

Add_Line

cAvax method

Purpose

This function adds a straight line segment

VB.NET declarations

```
Function Add_Line(ByRef x1 As Double, ByRef y1 As Double, ByRef z1 As Double, ByRef x2 As Double, ByRef y2
     As Double, ByRef z2 As Double, ByRef vItemProperties As Object, ByRef iClr As Integer, ByRef iWidth As
    Integer, ByRef iStyle As Integer, ByRef iLayer As Integer) As Integer

Function Add_Line(ByRef x1 As Double, ByRef y1 As Double, ByRef z1 As Double, ByRef x2 As Double, ByRef y2
     As Double, ByRef z2 As Double) As Integer
```

Parameters

Name	Type and passing	Meaning
x1	ByRef x1 As Double	Is the x coordinate of the line segment's first point
y1	ByRef y1 As Double	Is the y coordinate of the line segment's first point
z1	ByRef z1 As Double	Is the z coordinate of the line segment's first point
x2	ByRef x2 As Double	Is the x coordinate of the line segment's end point
y2	ByRef y2 As Double	Is the y coordinate of the line segment's end point
z2	ByRef z2 As Double	Is the z coordinate of the line segment's end point
vItemProperties	ByRef vItemProperties As Object	One-dimensional String() or Object() containing strings/Nothing, with logical properties starting at index 1. The control stores its own copy and retains the original caller reference. Nothing means no properties, except Add_Data which requires an array.
iClr	ByRef iClr As Integer	-1 uses the corresponding current pen setting. Other negative values become 0. Positive values use Integer storage. When declared ByRef, the effective value is written back only after successful validation.
iWidth	ByRef iWidth As Integer	-1 uses the current pen. Nonpositive resolved widths become 1, as in R2. Positive values use Integer storage. When declared ByRef, the effective value is written back only after successful validation.
iStyle	ByRef iStyle As Integer	-1 uses the corresponding current pen setting. Other negative values become 0. Positive values use Integer storage. When declared ByRef, the effective value is written back only after successful validation.
iLayer	ByRef iLayer As Integer	-1 uses the corresponding current pen setting. Other negative values become 0. Positive values use Integer storage. When declared ByRef, the effective value is written back only after successful validation.

Use the declaration for the overload you call. ByRef arguments can receive outputs; declare variables and arrays with the exact indicated type. Optional defaults are shown in the declaration.

Return value

Integer - An Avax handle on success, zero on error.

Numeric changes from v2

Update the following variables, arrays or event-data receivers when porting the corresponding v2 call. The declarations above show the current overloads.

Parameter or value	v2 type	v3 type
x1	Single	Double
y1	Single	Double
z1	Single	Double
x2	Single	Double
y2	Single	Double
z2	Single	Double
iClr	Short	Integer
iWidth	Short	Integer
iStyle	Short	Integer
iLayer	Short	Integer

For ByRef parameters, use a variable with the exact v3 type. Preserve array dimensions and the documented indexing. Moving from v2 to v3 explains conversion and legacy file-storage rules.

Example

Setup

Add an Avax.cAvax Windows Forms control named `cAvax1`. Paste the procedures into the form class. Example setup and supporting files explains references, initialization and file paths.

Add these Windows Forms controls or components to the form. Fields referenced by Handles clauses must be declared WithEvents, as the Visual Studio designer normally does.

Name	Type
Command1	Button

VB.NET example

```
Private Sub Form1_Load(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs) Handles
    MyBase.Load
    cAvax1.StartAvax()
End Sub
Private Sub Form1_FormClosed(ByVal eventSender As System.Object, ByVal eventArgs As
    System.Windows.Forms.FormClosedEventArgs) Handles Me.FormClosed
    cAvax1.EndAvax()
End Sub
Private Sub Command1_Click(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs)
    Handles Command1.Click
    Dim x1 As Double
    Dim y1 As Double
    Dim z1 As Double
    Dim x2 As Double
    Dim y2 As Double
    Dim z2 As Double
    Dim vItemProperties As Object = Nothing
    Dim iClr As Integer
    Dim iWidth As Integer
    Dim iStyle As Integer
    Dim iLayer As Integer
    Dim AvaxHandle As Integer
    ' Define the endpoints of the line
    x1 = 2 : y1 = 2 : z1 = 0
    x2 = 15 : y2 = 10 : z2 = 0
    ' Define the drawing color
    iClr = 1
    ' Define the drawing the line type
    iStyle = 3
```

```
    AvaxHandle = cAvax1.Add_Line(x1, y1, z1, x2, y2, z2, vItemProperties, iClr, iWidth, iStyle, iLayer)
End Sub
```

Add_LinearHatch

cAvax method

Purpose

Use this method if you want to add a Linear Hatch into the drawing

VB.NET declarations

```
Function Add_LinearHatch(ByRef xV As Double(), ByRef yV As Double(), ByRef zV As Double(), ByRef Distance
    As Double, ByRef Angle As Double, ByRef fOutside As Boolean, ByRef fBorder As Boolean, ByRef Mask As
    String) As Integer

Function Add_LinearHatch(ByRef xV As Double(), ByRef yV As Double(), ByRef zV As Double(), ByRef Distance
    As Double, ByRef Angle As Double, ByRef fOutside As Boolean, ByRef fBorder As Boolean, ByRef Mask As
    String, ByRef vItemProperties As Object, ByRef iClr As Integer, ByRef iWidth As Integer, ByRef iStyle As
    Integer, ByRef iLayer As Integer) As Integer
```

Parameters

Name	Type and passing	Meaning
xV	ByRef xV As Double()	One-based Double array of polygon X coordinates; index 0 is reserved. At least three vertices are required.
yV	ByRef yV As Double()	One-based Double array of polygon Y coordinates, matching xV.
zV	ByRef zV As Double()	One-based Double array matching xV/yV. Hatch segments use zV(1), as in R2; Nothing uses Pen3DElevetion in v3.
Distance	ByRef Distance As Double	Positive finite Double spacing between parallel hatch lines, in drawing units.
Angle	ByRef Angle As Double	Finite Double hatch-line angle in degrees.
fOutside	ByRef fOutside As Boolean	True draws the parts outside the polygon but inside its rectangular bounds.
fBorder	ByRef fBorder As Boolean	True also draws the axis-aligned rectangular frame around the polygon.
Mask	ByRef Mask As String	Empty draws every row. A string of 0 and 1 repeats across rows; 0 skips a row and 1 draws it.
vItemProperties	ByRef vItemProperties As Object	Optional one-dimensional string array with logical properties starting at index 1; values are copied.
iClr	ByRef iClr As Integer	Is the color index number
iWidth	ByRef iWidth As Integer	Integer pen width. -1 uses PenWidth and zero normalizes to one. The normalized value is returned on success.
iStyle	ByRef iStyle As Integer	Is the line type
iLayer	ByRef iLayer As Integer	Is the layer number

Use the declaration for the overload you call. ByRef arguments can receive outputs; declare variables and arrays with the exact indicated type. Optional defaults are shown in the declaration.

Return value

Integer - An Avax handle on success, zero on error.

Remarks

Creates parallel hatch segments as one compound/polyline object and returns its handle; returns zero for invalid or empty output. Both overloads use one-based Double coordinate arrays. X/Y/Z arrays must have matching lengths and finite active coordinates; omitted Z uses Pen3DElevetion. All output segments use the first vertex Z. Distance is positive finite perpendicular spacing; Angle is a finite number of degrees, equivalent modulo 180. Rows are phased from the polygon bounding rectangle and include an upper boundary row when the spacing reaches it exactly. Vertical segments run bottom to top. fOutside fills the complement within the bounding rectangle. fBorder prepends the rectangle edges in left-up, top-right, right-down, bottom-left order. Mask is empty for all rows or a repeating string of 0 and 1; a zero-only mask suppresses both fill and border. A one-row hatch remains a real segment with distinct endpoints. The polygon is implicitly closed; an explicit closing vertex is allowed. Self-crossing interiors use even-odd filling; unrepresentable or excessive output is rejected before insertion. Caller arrays and string properties are copied. Current or explicit pen settings apply to the compound object. Normalized ByRef pen values are returned only on success. Insertion is one Undo/Redo operation. Invalid requests preserve drawing, history and accepted input. Stopped, disabled or disposed controls return zero. PauseAvax and LockEdit do not block programmatic insertion. Exceptions from host event handlers propagate after committed output. This programmatic call publishes Change without NewUserItems, as in R2.

Hatch rows, borders and masks

Numeric changes from v2

Update the following variables, arrays or event-data receivers when porting the corresponding v2 call. The declarations above show the current overloads.

Parameter or value	v2 type	v3 type
xV	Single()	Double()
yV	Single()	Double()
zV	Single()	Double()
Distance	Single	Double
Angle	Single	Double
iClr	Short	Integer
iWidth	Short	Integer
iStyle	Short	Integer
iLayer	Short	Integer

For ByRef parameters, use a variable with the exact v3 type. Preserve array dimensions and the documented indexing. Moving from v2 to v3 explains conversion and legacy file-storage rules.

Example

Setup

Add an Avax.cAvax Windows Forms control named `cAvax1`. Paste the procedures into the form class. Example setup and supporting files explains references, initialization and file paths.

Add these Windows Forms controls or components to the form. Fields referenced by Handles clauses must be declared WithEvents, as the Visual Studio designer normally does.

Name	Type
Button1	Button

VB.NET example

```
Private Sub Form1_Load(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs) Handles
    MyBase.Load
    Call cAvax1.StartAvax()
End Sub
Private Sub Form1_FormClosed(ByVal eventSender As System.Object, ByVal eventArgs As
    System.Windows.Forms.FormClosedEventArgs) Handles Me.FormClosed
    Call cAvax1.EndAvax()
```

```
End Sub
Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
    Button1.Click
    Dim xV() As Double = Nothing
    Dim yV() As Double = Nothing
    Dim zV() As Double = Nothing
    Dim Distance As Double
    Dim Angle As Double
    Dim fOutside As Boolean
    Dim fBorder As Boolean
    Dim Mask As String = String.Empty
    Dim vItemProperties As Object = Nothing
    Dim iClr As Integer
    Dim iWidth As Integer
    Dim iStyle As Integer
    Dim iLayer As Integer
    Dim Answer As Integer
    ReDim xV(5)
    ReDim yV(5)
    ReDim zV(5)
    xV(1) = 2
    yV(1) = 2
    xV(2) = 2
    yV(2) = 12
    xV(3) = 12
    yV(3) = 12
    xV(4) = 12
    yV(4) = 2
    xV(5) = 10
    yV(5) = -1
    Distance = 0.3
    Angle = 45
    fOutside = False
    fBorder = False
    Mask = "100"
    Answer = cAvax1.Add_LinearHatch(xV, yV, zV, Distance, Angle, fOutside, fBorder, Mask, vItemProperties,
     iClr, iWidth, iStyle, iLayer)
End Sub
```

Add_OBSpline

cAvax method

Purpose

Adds an open, clamped quadratic B-spline from XYZ control points and stores its sampled outline as a polyline. The curve passes through its first and last controls.

VB.NET declarations

```
Function Add_OBSpline(ByRef x1 As Double(), ByRef y1 As Double(), ByRef z1 As Double()) As Integer

Function Add_OBSpline(ByRef x1 As Double(), ByRef y1 As Double(), ByRef z1 As Double(), ByRef
    vItemProperties As Object, ByVal iClr As Integer, ByRef iWidth As Integer, ByRef iStyle As Integer, ByRef
    iLayer As Integer) As Integer
```

Parameters

Name	Type and passing	Meaning
x1	ByRef x1 As Double()	One-based Double X control coordinates; index zero is reserved. Two to 3200 controls (at least three for a closed B-spline). Caller arrays are not retained or modified.
y1	ByRef y1 As Double()	One-based Double Y control coordinates, with the same array length as x1. Values must be finite.
z1	ByRef z1 As Double()	Optional one-based Double Z control coordinates. Nothing means zero; otherwise the length must match x1. Unlike R2, generated samples retain/interpolate the supplied Z.
vItemProperties	ByRef vItemProperties As Object	Optional string-array metadata, copied into the one resulting polyline. The caller object reference is preserved.
iClr	ByVal iClr As Integer	Integer color. -1 uses the current pen; successful full calls return the effective value ByRef.
iWidth	ByRef iWidth As Integer	Integer width. -1 uses the current pen; nonpositive resolved width becomes 1.
iStyle	ByRef iStyle As Integer	Integer line-style index; -1 uses the current pen. It is not truncated to the old byte/Integer range.
iLayer	ByRef iLayer As Integer	Integer layer index; -1 uses the current pen. Invalid input leaves the model and caller outputs unchanged.

Use the declaration for the overload you call. ByRef arguments can receive outputs; declare variables and arrays with the exact indicated type. Optional defaults are shown in the declaration.

Return value

Integer - An Avax handle on success, zero on error.

Remarks

The open, clamped quadratic B-spline passes through its first and last controls. Interior controls guide its shape. The full knot range is sampled at 20 parameter values and stored as an editable Polyline_i. Two controls produce a straight path. In the full overload iClr is ByVal, so resolving -1 does not change the caller color variable.

Supply equal-length Double coordinate arrays with logical points at indices 1 through the upper bound; index 0 is ignored. Nothing for the Z array supplies zero elevations. The control copies the coordinates. The maximum is 3200 input controls. The short overload uses the current pen. The full overload accepts copied String metadata and Integer pen attributes; -1 selects the current pen. Only attributes declared ByRef are returned to the caller. A started, enabled control is required. Invalid coordinates or geometry return 0 before changing the drawing or caller outputs. A successful insertion is one Undo operation and raises Change without NewUserItems. Application event-handler exceptions propagate after insertion.

Numeric changes from v2

Update the following variables, arrays or event-data receivers when porting the corresponding v2 call. The declarations above show the current overloads.

Parameter or value	v2 type	v3 type
x1	Single()	Double()
y1	Single()	Double()
z1	Single()	Double()
iClr	Short	Integer
iWidth	Short	Integer
iStyle	Short	Integer
iLayer	Short	Integer

For ByRef parameters, use a variable with the exact v3 type. Preserve array dimensions and the documented indexing. Moving from v2 to v3 explains conversion and legacy file-storage rules.

Example

Setup

Add an Avax.cAvax Windows Forms control named `cAvax1`. Paste the procedures into the form class. Example setup and supporting files explains references, initialization and file paths.

Add these Windows Forms controls or components to the form. Fields referenced by Handles clauses must be declared WithEvents, as the Visual Studio designer normally does.

Name	Type
Command1	Button

VB.NET example

```
Private Sub Form1_Load(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs) Handles
    MyBase.Load
    cAvax1.StartAvax()
End Sub
Private Sub Form1_FormClosed(ByVal eventSender As System.Object, ByVal eventArgs As
    System.Windows.Forms.FormClosedEventArgs) Handles Me.FormClosed
    cAvax1.EndAvax()
End Sub
Private Sub Command1_Click(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs)
    Handles Command1.Click
    Dim x1() As Double = Nothing
    Dim y1() As Double = Nothing
    Dim z1() As Double = Nothing
    Dim vItemProperties As Object = Nothing
    Dim iClr As Integer
    Dim iWidth As Integer
    Dim iStyle As Integer
    Dim iLayer As Integer
    Dim AvaxHandle As Integer
    ReDim x1(4)
    ReDim y1(4)
    ReDim z1(4)
    'Points for the spline nodes definition
    x1(1) = -1 : y1(1) = 8 : z1(1) = 0
    x1(2) = 3 : y1(2) = 1 : z1(2) = 0
    x1(3) = 6 : y1(3) = 5.25 : z1(3) = 0
    x1(4) = 12 : y1(4) = 0 : z1(4) = 0
    'Color and width definition
    iClr = 6
```

```
    iWidth = 1
    AvaxHandle = cAvax1.Add_OBSpline(x1, y1, z1, vItemProperties, iClr, iWidth, iStyle, iLayer)
End Sub
```

Add_PCX

cAvax method

Purpose

This function inserts a .PCX file into the drawing

VB.NET declarations

```
Function Add_PCX(ByRef sPCXFile As String, ByRef x As Double, ByRef y As Double) As Integer

Function Add_PCX(ByRef sPCXFile As String, ByRef x As Double, ByRef y As Double, ByRef zOrder As Integer
    ByRef vItemProperties As Object, ByRef iLayer As Integer, ByRef nMapMode As Integer) As Integer
```

Parameters

Name	Type and passing	Meaning
sPCXFile	ByRef sPCXFile As String	Is the name of the PCX file
x	ByRef x As Double	Is the x coordinate of the insertion point
y	ByRef y As Double	Is the y coordinate of the insertion point
zOrder	ByRef zOrder As Integer	Is the objects elevation in the whole drawing is defined by the parameter zOrder
vItemProperties	ByRef vItemProperties As Object	Is a string array with properties for the new objects
iLayer	ByRef iLayer As Integer	Is the layer number
nMapMode	ByRef nMapMode As Integer	Is the API's mapmode

Use the declaration for the overload you call. ByRef arguments can receive outputs; declare variables and arrays with the exact indicated type. Optional defaults are shown in the declaration.

Return value

Integer - An Avax handle on success, zero on error.

Remarks

Embeds the decoded PCX image with its lower-left drawing corner at (x, y). The natural drawing width is pixelWidth * 25.4 / displayDpiX / PrintUnit_MM; the height uses pixelHeight and displayDpiY. Embedded image DPI is not used. The short overload uses zOrder 0, the current layer and mapping mode 8 (anisotropic). The full overload retains the specified drawing order, string-array properties, layer and mapping mode 1 through 8. Coordinates are Double and the layer is Integer in v3. StartAvax and an enabled control are required. A missing or malformed PCX file, invalid mapping mode or unrepresentable insertion rectangle returns zero without inserting an object. A successful call embeds the pixels and creates one undo step. The source PCX file is not required when reopening the saved drawing. Exceptions raised by application event handlers remain visible to the application.

Numeric changes from v2

Update the following variables, arrays or event-data receivers when porting the corresponding v2 call. The declarations above show the current overloads.

Parameter or value	v2 type	v3 type
x	Single	Double
y	Single	Double
iLayer	Short	Integer

For ByRef parameters, use a variable with the exact v3 type. Preserve array dimensions and the documented indexing. Moving from v2 to v3 explains conversion and legacy file-storage rules.

Example

Setup

Add an Avax.cAvax Windows Forms control named `cAvax1`. Paste the procedures into the form class. Example setup and supporting files explains references, initialization and file paths.

Add these Windows Forms controls or components to the form. Fields referenced by Handles clauses must be declared WithEvents, as the Visual Studio designer normally does.

Name	Type
Command1	Button

VB.NET example

```
Private Sub Form1_Load(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs) Handles
    MyBase.Load
    cAvax1.StartAvax()
End Sub
Private Sub Form1_FormClosed(ByVal eventSender As System.Object, ByVal eventArgs As
    System.Windows.Forms.FormClosedEventArgs) Handles Me.FormClosed
    cAvax1.EndAvax()
End Sub
Private Sub Command1_Click(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs)
    Handles Command1.Click
    Dim sPCXFile As String = String.Empty
    Dim x As Double
    Dim y As Double
    Dim Answer As Integer
    sPCXFile = My.Application.Info.DirectoryPath & "\..\car.pcx"
    Answer = cAvax1.Add_PCX(sPCXFile, x, y)
End Sub
```

Add_Picture

cAvax method

Purpose

This function inserts picture file(s) (.BMP, .ICO, .GIF and .JPG) into the drawing

VB.NET declarations

```
Function Add_Picture(ByRef sPictureFiles As String(), ByVal xMin As Double, ByVal yMin As Double, ByVal
    xMax As Double, ByVal yMax As Double) As Integer

Function Add_Picture(ByRef sPictureFiles As String(), ByVal xMin As Double, ByVal yMin As Double, ByVal
    xMax As Double, ByVal yMax As Double, ByRef zOrder As Integer, ByRef vItemProperties As Object, ByRef
    iLayer As Integer, ByRef nMapMode As Integer) As Integer
```

Parameters

Name	Type and passing	Meaning
sPictureFiles	ByRef sPictureFiles As String()	One-based array of source filenames. Element 0 is unused; every element 1 through N must name a readable picture, AVI or AAF file.
xMin	ByVal xMin As Double	First X corner of the insertion rectangle, in Double drawing coordinates.
yMin	ByVal yMin As Double	First Y corner of the insertion rectangle, in Double drawing coordinates.
xMax	ByVal xMax As Double	Opposite X corner of the insertion rectangle, in Double drawing coordinates.
yMax	ByVal yMax As Double	Opposite Y corner of the insertion rectangle, in Double drawing coordinates.
zOrder	ByRef zOrder As Integer	Drawing order of the embedded picture or video.
vItemProperties	ByRef vItemProperties As Object	Application-defined item properties, using the legacy one-based String array convention when supplied.
iLayer	ByRef iLayer As Integer	Destination layer. A current-layer sentinel is resolved to the active pen layer and written back by the extended overload.
nMapMode	ByRef nMapMode As Integer	Windows map mode, from 1 through 8. The Integer overload uses 8 (MM_ANISOTROPIC).

Use the declaration for the overload you call. ByRef arguments can receive outputs; declare variables and arrays with the exact indicated type. Optional defaults are shown in the declaration.

Return value

Integer - An Avax handle on success, zero on error.

Remarks

Embeds one or more picture files in a drawing rectangle and returns the new handle. sPictureFiles uses logical indices starting at 1; index 0 is unused. Multiple files, animated image frames and AAF files form a sequence. AVI files may be included in the same sequence as still images. The extended overload accepts z-order, item properties, layer and map mode. Coordinates remain Double values in drawing units. Invalid input returns 0 without inserting part of a sequence. See Add_AVI for AVI sound playback, sequence visibility and playback lifetime.

Numeric changes from v2

Update the following variables, arrays or event-data receivers when porting the corresponding v2 call. The declarations above show the current overloads.

Parameter or value	v2 type	v3 type
xMin	Single	Double
yMin	Single	Double
xMax	Single	Double
yMax	Single	Double
iLayer	Short	Integer

For ByRef parameters, use a variable with the exact v3 type. Preserve array dimensions and the documented indexing. Moving from v2 to v3 explains conversion and legacy file-storage rules.

Example

Setup

Add an Avax.cAvax Windows Forms control named `cAvax1`. Paste the procedures into the form class. Example setup and supporting files explains references, initialization and file paths.

Add these Windows Forms controls or components to the form. Fields referenced by Handles clauses must be declared WithEvents, as the Visual Studio designer normally does.

Name	Type
Command1	Button

VB.NET example

```
Private Sub Form1_Load(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs) Handles
    MyBase.Load
    cAvax1.StartAvax()
End Sub
Private Sub Form1_FormClosed(ByVal eventSender As System.Object, ByVal eventArgs As
    System.Windows.Forms.FormClosedEventArgs) Handles Me.FormClosed
    cAvax1.EndAvax()
End Sub
Private Sub Command1_Click(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs)
    Handles Command1.Click
    Dim sPictureFiles() As String = Nothing
    Dim xMin As Double
    Dim yMin As Double
    Dim xMax As Double
    Dim yMax As Double
    Dim AvaxHandle As Integer
    ReDim sPictureFiles(1)
    sPictureFiles(1) = My.Application.Info.DirectoryPath & "\..\Clouds1.bmp"
    xMin = 0
    yMin = 0
    xMax = 5
    yMax = 5
    AvaxHandle = cAvax1.Add_Picture(sPictureFiles, xMin, yMin, xMax, yMax)
End Sub
```

Add_PictureFile

API reference

Embeds one image file in a rectangular area of the drawing and returns its object handle. Use this method when the source is one filename; Add_Picture accepts a one-based array of filenames. Image pixels or metafile records are stored in the drawing, so the source file is not needed when reopening it.

AVI video files retain their soundtrack inside the drawing. See Add_AVI for audio playback, visibility and playback lifetime. Animated images and AAF files use the same sequence rules as Add_Picture.

VB.NET declarations

```
Public Function Add_PictureFile(ByRef sPictureFile As String,
    ByVal xMin As Double, ByVal yMin As Double,
    ByVal xMax As Double, ByVal yMax As Double) As Integer

Public Function Add_PictureFile(ByRef sPictureFile As String,
    ByVal xMin As Double, ByVal yMin As Double,
    ByVal xMax As Double, ByVal yMax As Double,
    ByRef zOrder As Integer, ByRef vItemProperties As Object,
    ByRef iLayer As Integer, ByRef nMapMode As Integer) As Integer
```

Parameters

Parameter	Meaning
sPictureFile	Path to a readable image file. The filename is not modified.
xMin, yMin, xMax, yMax	Opposite corners in drawing coordinates. Reversed corners are normalized. Both dimensions must be positive and finite. The supplied rectangle determines the displayed size; source DPI does not resize it.
zOrder	Drawing order of the image. The short overload uses zero.
vItemProperties	String array stored as user properties, or Nothing. Logical property values start at index 1. The control owns a copy.
iLayer	Destination layer. Use -1 for the current layer; successful insertion writes the resolved layer back to this variable. The short overload uses the current layer.
nMapMode	Windows mapping mode, from 1 through 8. The short overload uses 8 (anisotropic). The mapping mode is retained for metafile rendering.

Result and remarks

Returns a positive Integer handle on success, or zero when the input cannot be inserted. Call StartAvax first and keep the control enabled. Input failures preserve the drawing and Undo history. Success adds one object with one Undo step. Exceptions raised by the application's event handlers propagate after the insertion has committed.

Bitmap inputs include BMP, PNG, JPEG, GIF, TIFF, ICO and PCX. GIF and AVAX AAF image sequences remain animated. WMF and EMF input retains its original records. AVI video is also accepted, directly or inside an AAF sequence; its original video data remains embedded. The control renders video frames only and does not play audio tracks. Add_AVI provides an AVI-specific entry point.

In v3, the rectangle coordinates are Double and iLayer is Integer. Handles, zOrder and nMapMode remain Integer. Supply variables of the exact declared type for ByRef arguments.

VB.NET example

Pass an already started control and a valid image filename. The example places an image in a 20 by 10 drawing-unit rectangle.

```
Option Strict On
Imports System

Public Module PictureFileExample
    Public Function AddLogo(cad As Avax.cAvax, filename As String) As Integer
        If cad Is Nothing Then Throw New ArgumentNullException(NameOf(cad))
```

```
        Dim handle As Integer = cad.Add PictureFile(filename, 12.5, 8.0, 32.5, 18.0)
        If handle = 0 Then Throw New InvalidOperationException("The image could not be inserted. Check the
     filename and control state.")
        Return handle
    End Function
End Module
```

Add_PLT

cAvax method

Purpose

This function inserts a .PLT file into the drawing

VB.NET declarations

```
Function Add_PLT(ByRef sPLTFile As String, ByRef x As Double, ByRef y As Double, ByRef z As Double) As
    Boolean

Function Add_PLT(ByRef sPLTFile As String, ByRef x As Double, ByRef y As Double, ByRef z As Double, ByRef
    sScale As Double, ByRef sAngle As Double, ByRef vItemProperties As Object, ByRef iLayer As Integer, ByRef
    vRetHandles As Object) As Boolean
```

Parameters

Name	Type and passing	Meaning
sPLTFile	ByRef sPLTFile As String	Existing HP-GL file path, including its extension. Relative paths use the working directory. The string and source file are preserved.
x	ByRef x As Double	Double X coordinate of the insertion point in drawing units.
y	ByRef y As Double	Double Y coordinate of the insertion point in drawing units.
z	ByRef z As Double	Double elevation assigned to the imported planar drawing.
sScale	ByRef sScale As Double	Scale applied after PltUnit_p conversion. Zero is replaced by 1 on success. A negative value reverses both XY axes.
sAngle	ByRef sAngle As Double	Counterclockwise rotation in degrees about the insertion point, in the XY plane.
vItemProperties	ByRef vItemProperties As Object	Nothing or a String() of application properties to copy to each inserted object.
iLayer	ByRef iLayer As Integer	Integer destination layer. -1 selects the current layer and receives that layer number on success.
vRetHandles	ByRef vRetHandles As Object	Output Object containing an Integer() of the inserted handles. Index 0 is reserved; handles occupy indices 1 through the upper bound. Preserved on failed import.

Use the declaration for the overload you call. ByRef arguments can receive outputs; declare variables and arrays with the exact indicated type. Optional defaults are shown in the declaration.

Return value

Boolean - True on success, false on error.

Remarks

Inserts an HP-GL plot as individual drawing objects. Call StartAvax first and keep the control enabled. Supply an existing file path, including its extension; relative paths use the application working directory. The input file is read without modification. Plot coordinates are multiplied by PltUnit_p, whose default is 0.025, then by sScale, rotated counterclockwise by sAngle degrees in the XY plane, and translated by (x, y, z). The short overload uses scale 1, angle 0 and the current layer. In the full overload, a zero scale becomes 1 and is written back; a negative scale reverses both XY axes. Layer -1 resolves to the current layer and is written back on success. Placement, scale and angle use Double; layers and handles use Integer. The full overload copies the supplied String() user properties to every inserted object and returns an Integer() of new handles through vRetHandles. Index zero is reserved; actual handles start at index one. Pen selections become palette indices. Lines retain width 1 and continuous style. Supported commands include IN, DF, SP, PU, PD, PA, PR, SI, DI, LB, CI and PG; IP is accepted with the legacy PltUnit conversion, and VS does not affect geometry. Labels use the current font; labels in legacy files are read as Windows-1253. Unsupported commands, including SC scaling, cause the import to fail. Full HP-GL/2 is not supported. True means that objects were inserted. Missing, empty or malformed input, invalid placement values or insufficient object capacity return False without partially importing the plot or replacing caller outputs. A successful import is one Undo step; Redo restores all inserted objects. Native .avax storage retains the resulting Double coordinates. Exceptions raised by application completion handlers propagate after the import commits. The example imports the original house.plt from the parent of the application directory and fits the resulting drawing to the view. This programmatic operation publishes Change without NewUserItems, as in R2. Use its documented return value/output parameters for the result. Interactive drawing commands and explicit announcing shape/picture/document helpers retain their own notification policy.

Numeric changes from v2

Update the following variables, arrays or event-data receivers when porting the corresponding v2 call. The declarations above show the current overloads.

Parameter or value	v2 type	v3 type
x	Single	Double
y	Single	Double
z	Single	Double
sScale	Single	Double
sAngle	Single	Double
iLayer	Short	Integer

For ByRef parameters, use a variable with the exact v3 type. Preserve array dimensions and the documented indexing. Moving from v2 to v3 explains conversion and legacy file-storage rules.

Example

Setup

Add an Avax.cAvax Windows Forms control named `cAvax1`. Paste the procedures into the form class. Example setup and supporting files explains references, initialization and file paths.

Add these Windows Forms controls or components to the form. Fields referenced by Handles clauses must be declared WithEvents, as the Visual Studio designer normally does.

Name	Type
Command1	Button

VB.NET example

```
Private Sub Form1_Load(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs) Handles
    MyBase.Load
    cAvax1.StartAvax()
End Sub
Private Sub Form1_FormClosed(ByVal eventSender As System.Object, ByVal eventArgs As
    System.Windows.Forms.FormClosedEventArgs) Handles Me.FormClosed
    cAvax1.EndAvax()
End Sub
Private Sub Command1_Click(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs)
```

```
    Handles Command1.Click
    Dim sPLTFile As String = String.Empty
    Dim x As Double
    Dim y As Double
    Dim z As Double
    Dim Answer As Boolean
    sPLTFile = My.Application.Info.DirectoryPath & "\..\house.plt"
    Answer = cAvax1.Add_PLT(sPLTFile, x, y, z)
    CAvax1.Command = Avax.AvaxCommand.AutoLimits_c
End Sub
```

Add_Point

cAvax method

Purpose

This function adds a point into the drawing

See Line, Point and Rectangle commands for connected XYZ lines, the planar Point tool versus Add_Point, rectangular closed polylines, completion and host callbacks.

VB.NET declarations

```
Function Add_Point(ByRef x1 As Double, ByRef y1 As Double, ByRef z1 As Double) As Integer

Function Add_Point(ByRef x1 As Double, ByRef y1 As Double, ByRef z1 As Double, ByRef vItemProperties As
    Object, ByRef iClr As Integer, ByRef iWidth As Integer, ByRef iPointStyle As Integer, ByRef iLayer As
    Integer) As Integer
```

Parameters

Name	Type and passing	Meaning
x1	ByRef x1 As Double	Is the x coordinate of the point
y1	ByRef y1 As Double	Is the y coordinate of the point
z1	ByRef z1 As Double	Is the z coordinate of the point
vItemProperties	ByRef vItemProperties As Object	One-dimensional String() or Object() containing strings/Nothing, with logical properties starting at index 1. The control stores its own copy and retains the original caller reference. Nothing means no properties, except Add_Data which requires an array.
iClr	ByRef iClr As Integer	-1 uses the corresponding current pen setting. Other negative values become 0. Positive values use Integer storage. When declared ByRef, the effective value is written back only after successful validation.
iWidth	ByRef iWidth As Integer	-1 uses the current pen. Nonpositive resolved widths become 1, as in R2. Positive values use Integer storage. When declared ByRef, the effective value is written back only after successful validation.
iPointStyle	ByRef iPointStyle As Integer	-1 uses the corresponding current pen setting. Other negative values become 0. Positive values use Integer storage. When declared ByRef, the effective value is written back only after successful validation.
iLayer	ByRef iLayer As Integer	-1 uses the corresponding current pen setting. Other negative values become 0. Positive values use Integer storage. When declared ByRef, the effective value is written back only after successful validation.

Use the declaration for the overload you call. ByRef arguments can receive outputs; declare variables and arrays with the exact indicated type. Optional defaults are shown in the declaration.

Return value

Integer - An Avax handle on success, zero on error.

Numeric changes from v2

Update the following variables, arrays or event-data receivers when porting the corresponding v2 call. The declarations above show the current overloads.

Parameter or value	v2 type	v3 type
x1	Single	Double
y1	Single	Double

Parameter or value	v2 type	v3 type
z1	Single	Double
iClr	Short	Integer
iWidth	Short	Integer
iPointStyle	Short	Integer
iLayer	Short	Integer

For ByRef parameters, use a variable with the exact v3 type. Preserve array dimensions and the documented indexing. Moving from v2 to v3 explains conversion and legacy file-storage rules.

Example

Setup

Add an Avax.cAvax Windows Forms control named `cAvax1`. Paste the procedures into the form class. Example setup and supporting files explains references, initialization and file paths.

Add these Windows Forms controls or components to the form. Fields referenced by Handles clauses must be declared WithEvents, as the Visual Studio designer normally does.

Name	Type
Command1	Button

VB.NET example

```
Private Sub Form1_Load(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs) Handles
    MyBase.Load
    cAvax1.StartAvax()
    Call CAvax1.SetAvaxProperty(Avax.AvaxProp.PointRadius, 0.5!)
End Sub
Private Sub Form1_FormClosed(ByVal eventSender As System.Object, ByVal eventArgs As
    System.Windows.Forms.FormClosedEventArgs) Handles Me.FormClosed
    cAvax1.EndAvax()
End Sub
Private Sub Command1_Click(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs)
    Handles Command1.Click
    Dim x1 As Double
    Dim y1 As Double
    Dim z1 As Double
    Dim vItemProperties As Object = Nothing
    Dim iClr As Integer
    Dim iWidth As Integer
    Dim iPointStyle As Integer
    Dim iLayer As Integer
    Dim AvaxHandle As Integer
    cAvax1.AutoRedraw = False
    x1 = -3 : y1 = 5 : z1 = 0
    iClr = 5
    For iPointStyle = 0 To 16
        AvaxHandle = cAvax1.Add_Point(x1, y1, z1, vItemProperties, iClr, iWidth, iPointStyle, iLayer)
        x1 = x1 + 1.5
    Next iPointStyle
    cAvax1.AutoRedraw = True
End Sub
```

Add_PolyLine

cAvax method

Purpose

This function adds a polyline into the drawing

See Polyline, Polygon and Close commands for pending XYZ chains, exact closing edges, regular polygons, vertex counts, completion and complete VB.NET examples.

VB.NET declarations

```
Function Add_PolyLine(ByRef x1 As Double(), ByRef y1 As Double(), ByRef z1 As Double()) As Integer

Function Add_PolyLine(ByRef x1 As Double(), ByRef y1 As Double(), ByRef z1 As Double(), ByRef
    vItemProperties As Object, ByRef iClr As Integer, ByRef iWidth As Integer, ByRef iStyle As Integer, ByRef
    iLayer As Integer) As Integer
```

Parameters

Name	Type and passing	Meaning
x1	ByRef x1 As Double()	An array for the x coordinates of the poline's apexes
y1	ByRef y1 As Double()	An array for the y coordinates of the poline's apexes
z1	ByRef z1 As Double()	An array for the z coordinates of the poline's apexes
vItemProperties	ByRef vItemProperties As Object	One-dimensional String() or Object() containing strings/Nothing, with logical properties starting at index 1. The control stores its own copy and retains the original caller reference. Nothing means no properties, except Add_Data which requires an array.
iClr	ByRef iClr As Integer	-1 uses the corresponding current pen setting. Other negative values become 0. Positive values use Integer storage. When declared ByRef, the effective value is written back only after successful validation.
iWidth	ByRef iWidth As Integer	-1 uses the current pen. Nonpositive resolved widths become 1, as in R2. Positive values use Integer storage. When declared ByRef, the effective value is written back only after successful validation.
iStyle	ByRef iStyle As Integer	-1 uses the corresponding current pen setting. Other negative values become 0. Positive values use Integer storage. When declared ByRef, the effective value is written back only after successful validation.
iLayer	ByRef iLayer As Integer	-1 uses the corresponding current pen setting. Other negative values become 0. Positive values use Integer storage. When declared ByRef, the effective value is written back only after successful validation.

Use the declaration for the overload you call. ByRef arguments can receive outputs; declare variables and arrays with the exact indicated type. Optional defaults are shown in the declaration.

Return value

Integer - An Avax handle on success, zero on error.

Remarks

Supply matching Double() coordinate arrays with at least two logical vertices at indices 1 through Length-1. Element 0 is ignored. X and Y are required. Nothing for Z supplies zero elevations; a supplied Z array must have the same length. The method retains the caller array references.

The vertices define straight segments in their supplied order. To close the polyline, repeat the first XYZ as the final vertex. Consecutive duplicate records and an explicit closing record are retained. A successful call creates one Polyline_i root and returns its Integer handle. The control owns its coordinate and metadata copies, so later caller-array edits do not change the stored result.

Logical coordinates must be finite and the prepared XY extents must fit Double storage. Missing X/Y, mismatched lengths, too few vertices, invalid metadata or unrepresentable extents return 0 before insertion. Validation failure preserves caller references, ByRef pen values, the drawing, accepted construction input and previous Redo. Stopped or disabled controls return 0 without changing outputs.

The short overload uses the current pen; the full overload resolves its documented sentinels and publishes effective ByRef values after successful preparation. A successful direct addition publishes Change and is one Undo/Redo operation. It preserves unrelated active construction input and follows the R2 ordinary programmatic-add notification contract without NewUserItems. Host callback exceptions propagate after the completed insertion. Coordinates use Double and pen/handle values use Integer in v3.

Numeric changes from v2

Update the following variables, arrays or event-data receivers when porting the corresponding v2 call. The declarations above show the current overloads.

Parameter or value	v2 type	v3 type
x1	Single()	Double()
y1	Single()	Double()
z1	Single()	Double()
iClr	Short	Integer
iWidth	Short	Integer
iStyle	Short	Integer
iLayer	Short	Integer

For ByRef parameters, use a variable with the exact v3 type. Preserve array dimensions and the documented indexing. Moving from v2 to v3 explains conversion and legacy file-storage rules.

Example

Setup

Add an Avax.cAvax Windows Forms control named `cAvax1`. Paste the procedures into the form class. Example setup and supporting files explains references, initialization and file paths.

Add these Windows Forms controls or components to the form. Fields referenced by Handles clauses must be declared WithEvents, as the Visual Studio designer normally does.

Name	Type
Command1	Button

VB.NET example

```
Private Sub Form1_Load(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs) Handles
    MyBase.Load
    cAvax1.StartAvax()
End Sub
Private Sub Form1_FormClosed(ByVal eventSender As System.Object, ByVal eventArgs As
    System.Windows.Forms.FormClosedEventArgs) Handles Me.FormClosed
    cAvax1.EndAvax()
End Sub
Private Sub Command1_Click(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs)
    Handles Command1.Click
    Dim x1() As Double = Nothing
    Dim y1() As Double = Nothing
    Dim z1() As Double = Nothing
    Dim vItemProperties As Object = Nothing
    Dim iClr As Integer
    Dim iWidth As Integer
    Dim iStyle As Integer
    Dim iLayer As Integer
    Dim AvaxHandle As Integer
```

```
    ReDim x1(5)
    ReDim y1(5)
    ReDim z1(5)
    x1(1) = 1 : y1(1) = 1 : z1(1) = 0
    x1(2) = 4 : y1(2) = 1 : z1(2) = 0
    x1(3) = 4 : y1(3) = 4 : z1(3) = 0
    x1(4) = 1 : y1(4) = 4 : z1(4) = 0
    x1(5) = 0 : y1(5) = 0 : z1(5) = 0
    iClr = 5
    iWidth = 1
    AvaxHandle = cAvax1.Add_PolyLine(x1, y1, z1, vItemProperties, iClr, iWidth, iStyle, iLayer)
End Sub
```

Add_Rectangle

cAvax method

Purpose

This function adds a rectangle into the drawing

VB.NET declarations

```
Function Add_Rectangle(ByRef x0 As Double, ByRef y0 As Double, ByRef nWidth As Double, ByRef nHeight As
    Double) As Integer

Function Add_Rectangle(ByRef x0 As Double, ByRef y0 As Double, ByRef nWidth As Double, ByRef nHeight As
    Double, ByRef vItemProperties As Object, ByRef iClr As Integer, ByRef iWidth As Integer, ByRef iStyle As
    Integer, ByRef iLayer As Integer) As Integer
```

Parameters

Name	Type and passing	Meaning
x0	ByRef x0 As Double	Is the x coordinate of the bottom left point of the rectanngle
y0	ByRef y0 As Double	Is the y coordinate of the bottom left point of the rectangle
nWidth	ByRef nWidth As Double	Is the width of the rectangle
nHeight	ByRef nHeight As Double	Is the height of the rectangle
vItemProperties	ByRef vItemProperties As Object	One-dimensional String() or Object() containing strings/Nothing, with logical properties starting at index 1. The control stores its own copy and retains the original caller reference. Nothing means no properties, except Add_Data which requires an array.
iClr	ByRef iClr As Integer	-1 uses the corresponding current pen setting. Other negative values become 0. Positive values use Integer storage. When declared ByRef, the effective value is written back only after successful validation.
iWidth	ByRef iWidth As Integer	-1 uses the current pen. Nonpositive resolved widths become 1, as in R2. Positive values use Integer storage. When declared ByRef, the effective value is written back only after successful validation.
iStyle	ByRef iStyle As Integer	-1 uses the corresponding current pen setting. Other negative values become 0. Positive values use Integer storage. When declared ByRef, the effective value is written back only after successful validation.
iLayer	ByRef iLayer As Integer	-1 uses the corresponding current pen setting. Other negative values become 0. Positive values use Integer storage. When declared ByRef, the effective value is written back only after successful validation.

Use the declaration for the overload you call. ByRef arguments can receive outputs; declare variables and arrays with the exact indicated type. Optional defaults are shown in the declaration.

Return value

Integer - An Avax handle on success, zero on error.

Numeric changes from v2

Update the following variables, arrays or event-data receivers when porting the corresponding v2 call. The declarations above show the current overloads.

Parameter or value	v2 type	v3 type
x0	Single	Double
y0	Single	Double

Parameter or value	v2 type	v3 type
nWidth	Single	Double
nHeight	Single	Double
iClr	Short	Integer
iWidth	Short	Integer
iStyle	Short	Integer
iLayer	Short	Integer

For ByRef parameters, use a variable with the exact v3 type. Preserve array dimensions and the documented indexing. Moving from v2 to v3 explains conversion and legacy file-storage rules.

Example

Setup

Add an Avax.cAvax Windows Forms control named `cAvax1`. Paste the procedures into the form class. Example setup and supporting files explains references, initialization and file paths.

Add these Windows Forms controls or components to the form. Fields referenced by Handles clauses must be declared WithEvents, as the Visual Studio designer normally does.

Name	Type
Command1	Button

VB.NET example

```
Private Sub Form1_Load(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs) Handles
    MyBase.Load
    cAvax1.StartAvax()
End Sub
Private Sub Form1_FormClosed(ByVal eventSender As System.Object, ByVal eventArgs As
    System.Windows.Forms.FormClosedEventArgs) Handles Me.FormClosed
    cAvax1.EndAvax()
End Sub
Private Sub Command1_Click(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs)
    Handles Command1.Click
    Dim x0 As Double
    Dim y0 As Double
    Dim nWidth As Double
    Dim nHeight As Double
    Dim vItemProperties As Object = Nothing
    Dim iClr As Integer
    Dim iWidth As Integer
    Dim iStyle As Integer
    Dim iLayer As Integer
    Dim AvaxHandle As Integer
    'Define the starting point of the Rectangle
    x0 = 3 : y0 = 3
    'Define the Drawing Color and the Drawing Width
    iClr = 6 : iWidth = 2
    ' Finally, define thse lengths of the sides
    nWidth = 8 : nHeight = 4
    AvaxHandle = cAvax1.Add_Rectangle(x0, y0, nWidth, nHeight, vItemProperties, iClr, iWidth, iStyle,
    iLayer)
End Sub
```

Add_RTF

cAvax method

Purpose

Imports an RTF file as individually editable text, line and picture objects. The full overload returns their one-based Integer handles.

VB.NET declarations

```
Function Add_RTF(ByRef sRTFFile As String, ByRef x As Double, ByRef y As Double, ByRef z As Double, ByRef
    vItemProperties As Object, ByRef iLayer As Integer, ByRef vRetHandles As Object) As Boolean

Function Add_RTF(ByRef sRTFFile As String, ByRef x As Double, ByRef y As Double, ByRef z As Double) As
    Boolean
```

Parameters

Name	Type and passing	Meaning
sRTFFile	ByRef sRTFFile As String	Is the name of the RTF file
x	ByRef x As Double	Is the x coordinate of the insertion point
y	ByRef y As Double	Is the y coordinate of the insertion point
z	ByRef z As Double	Is the z coordinate of the insertion point
vItemProperties	ByRef vItemProperties As Object	Is a string array with properties for the new object
iLayer	ByRef iLayer As Integer	Is the layer number
vRetHandles	ByRef vRetHandles As Object	Returns an integer array with the handles of the new objects

Use the declaration for the overload you call. ByRef arguments can receive outputs; declare variables and arrays with the exact indicated type. Optional defaults are shown in the declaration.

Return value

Boolean - True on success, false on error.

Remarks

The full overload accepts copied application metadata and an Integer layer, and returns a one-based Integer() in vRetHandles. The imported text, lines and pictures can be edited separately. Add_Document retains content in one document-layout object.

The file is laid out before any imported objects are inserted. Supported content includes Unicode text, font/style/color changes, table borders and embedded pictures. Images retain their embedded pixels and physical sizes. Unsupported drawing records or glyphs that cannot be mapped back to source text cause import to fail. Complex external floating frames and shape records are outside this import contract.

Call StartAvax first and keep the control enabled. The importer captures insertion coordinates, print units, current pen/elevation/height and metadata before the first Progress callback. Progress reports preparation from 0 through values below 100; one 100 follows a complete import and any automatic fitting that still belongs to the current request. PauseAvax and LockEdit do not block the direct import. PauseAvaxEvents suppresses notifications.

The complete batch is committed as one Undo operation and raises Change without NewUserItems. Expected file/layout failures return False without inserting part of the batch. A nested import, replacement drawing, unavailable control or changed construction input can supersede preparation. Ordinary edits made by a Progress handler in the same drawing are retained, and returned handles identify only this import. Application event-handler exceptions propagate. An exception after commit leaves the complete import undoable; an exception before commit leaves no imported batch.

Numeric changes from v2

Update the following variables, arrays or event-data receivers when porting the corresponding v2 call. The declarations above show the current overloads.

Parameter or value	v2 type	v3 type
x	Single	Double
y	Single	Double
z	Single	Double
iLayer	Short	Integer

For ByRef parameters, use a variable with the exact v3 type. Preserve array dimensions and the documented indexing. Moving from v2 to v3 explains conversion and legacy file-storage rules.

Example

Setup

Add an Avax.cAvax Windows Forms control named `cAvax1`. Paste the procedures into the form class. Example setup and supporting files explains references, initialization and file paths.

Add these Windows Forms controls or components to the form. Fields referenced by Handles clauses must be declared WithEvents, as the Visual Studio designer normally does.

Name	Type
Command1	Button
Command2	Button
Command3	Button

VB.NET example

```
Private Sub Command2_Click(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs)
    Handles Command2.Click
    CAvax1.Command = Avax.AvaxCommand.Zoom_c
End Sub
Private Sub Command3_Click(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs)
    Handles Command3.Click
    CAvax1.Command = Avax.AvaxCommand.Previous_c
End Sub
Private Sub Form1_Load(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs) Handles
    MyBase.Load
    cAvax1.StartAvax()
End Sub
Private Sub Form1_FormClosed(ByVal eventSender As System.Object, ByVal eventArgs As
    System.Windows.Forms.FormClosedEventArgs) Handles Me.FormClosed
    cAvax1.EndAvax()
End Sub
Private Sub Command1_Click(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs)
    Handles Command1.Click
    Dim sRTFFile As String = String.Empty
    Dim x As Double
    Dim y As Double
    Dim z As Double
    Dim Answer As Boolean
    sRTFFile = My.Application.Info.DirectoryPath & "\..\Introduction.rtf"
    Answer = cAvax1.Add_RTF(sRTFFile, x, y, z)
End Sub
```

Add_Shape

cAvax method

Purpose

This function adds a closed polygon shape with a specific fill type and z order

Fill resources

The full overload accepts an AvaxFillType family and its pattern, colors, flags and z-order. BMP and Windows metafile fills accept a local file path or an already embedded image resource name. The control validates and stores the image bytes before returning the new handle. You can then remove the source file; saving the drawing retains the embedded image.

An invalid fill or missing image returns zero without inserting an object, changing the caller's ByRef arguments or consuming Undo/Redo history. The short overload retains its R2 default fill.

VB.NET declarations

```
Function Add_Shape(ByRef xV As Double(), ByRef yV As Double(), ByRef zV As Double(), Optional ByVal
    fireNewItemsEvent As Boolean = True) As Integer

Function Add_Shape(ByRef xV As Double(), ByRef yV As Double(), ByRef zV As Double(), ByRef vItemProperties
     As Object, ByRef Fill_iType As Byte, ByRef Fill_lHatchIndex As Integer, ByRef Fill_iInForeColor As
    Integer, ByRef Fill_iInBackColor As Integer, ByRef Fill_sPattern As String, ByRef Fill_zOrder As Integer,
    ByRef Fill_lFlags As Integer, ByRef iClr As Integer, ByRef iWidth As Integer, ByRef iStyle As Integer,
    ByRef iLayer As Integer, ByRef PolylineType As AvaxPolylineType, Optional ByVal fireNewItemsEvent As
    Boolean = True) As Integer
```

Parameters

Name	Type and passing	Meaning
xV	ByRef xV As Double()	An array for the x coordinates of the polygon's apexes
yV	ByRef yV As Double()	An array for the y coordinates of the polygon's apexes
zV	ByRef zV As Double()	An array for the z coordinates of the polygon's apexes
fireNewItemsEvent	Optional ByVal fireNewItemsEvent As Boolean = True	True includes the created objects in the NewUserItems notification. False suppresses that notification for this call.
vItemProperties	ByRef vItemProperties As Object	One-dimensional String() or Object() containing strings/Nothing, with logical properties starting at index 1. The control stores its own copy and retains the original caller reference. Nothing means no properties, except Add_Data which requires an array.
Fill_iType	ByRef Fill_iType As Byte	A Byte expression that defines the AvaxFillType
Fill_lHatchIndex	ByRef Fill_lHatchIndex As Integer	Is the fill hatch index
Fill_iInForeColor	ByRef Fill_iInForeColor As Integer	Is the stroke color
Fill_iInBackColor	ByRef Fill_iInBackColor As Integer	Is the fill color
Fill_sPattern	ByRef Fill_sPattern As String	A String expression that defines the fill pattern's data (see also function CreateACADHatchPattern or CreateExVectorHatchPattern)
Fill_zOrder	ByRef Fill_zOrder As Integer	Is the object's elevation in the whole drawing
Fill_lFlags	ByRef Fill_lFlags As Integer	Is a numeric expresion which defines some more parameters (see also function CreatePatternFlags)
iClr	ByRef iClr As Integer	-1 uses the corresponding current pen setting. Other negative values become 0. Positive values use Integer storage. When declared ByRef, the effective value is written back only after successful validation.
iWidth	ByRef iWidth As Integer	-1 uses the current pen. Nonpositive resolved widths become 1, as in R2. Positive values use Integer storage. When declared ByRef, the effective value is written back only after successful validation.
iStyle	ByRef iStyle As Integer	-1 uses the corresponding current pen setting. Other negative values become 0. Positive values use Integer storage. When declared ByRef, the effective value is written back only after successful validation.
iLayer	ByRef iLayer As Integer	-1 uses the corresponding current pen setting. Other negative values become 0. Positive values use Integer storage. When declared ByRef, the effective value is written back only after successful validation.
PolylineType	ByRef PolylineType As AvaxPolylineType	An expression that defines the AvaxPolylineType of the polygon shape's border line

Use the declaration for the overload you call. ByRef arguments can receive outputs; declare variables and arrays with the exact indicated type. Optional defaults are shown in the declaration.

Return value

Integer - An Avax handle on success, zero on error.

Remarks

For more details for the fill parameters see the example ShapeFill.vbp

Numeric changes from v2

Update the following variables, arrays or event-data receivers when porting the corresponding v2 call. The declarations above show the current overloads.

Parameter or value	v2 type	v3 type
xV	Single()	Double()

Parameter or value	v2 type	v3 type
yV	Single()	Double()
zV	Single()	Double()
Fill_iInForeColor	Short	Integer
Fill_iInBackColor	Short	Integer
iClr	Short	Integer
iWidth	Short	Integer
iStyle	Short	Integer
iLayer	Short	Integer

For ByRef parameters, use a variable with the exact v3 type. Preserve array dimensions and the documented indexing. Moving from v2 to v3 explains conversion and legacy file-storage rules.

Example

Setup

Add an Avax.cAvax Windows Forms control named `cAvax1`. Paste the procedures into the form class. Example setup and supporting files explains references, initialization and file paths.

Add these Windows Forms controls or components to the form. Fields referenced by Handles clauses must be declared WithEvents, as the Visual Studio designer normally does.

Name	Type
Command1	Button

VB.NET example

```
Private Sub Form1_Load(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs) Handles
    MyBase.Load
    cAvax1.StartAvax()
End Sub
Private Sub Form1_FormClosed(ByVal eventSender As System.Object, ByVal eventArgs As
    System.Windows.Forms.FormClosedEventArgs) Handles Me.FormClosed
    cAvax1.EndAvax()
End Sub
Private Sub Command1_Click(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs)
    Handles Command1.Click
    Dim xV() As Double = Nothing
    Dim yV() As Double = Nothing
    Dim zV() As Double = Nothing
    Dim vItemProperties As Object = Nothing
    Dim Fill_iType As Byte
    Dim Fill_lHatchIndex As Integer
    Dim Fill_iInForeColor As Integer
    Dim Fill_iInBackColor As Integer
    Dim Fill_sPattern As String = String.Empty
    Dim Fill_zOrder As Integer
    Dim Fill_lFlags As Integer
    Dim iClr As Integer
    Dim iWidth As Integer
    Dim iStyle As Integer
    Dim iLayer As Integer
    Dim PolylineType As Avax.AvaxPolylineType
    Dim AvaxHandle As Integer
    ReDim xV(5)
    ReDim yV(5)
    ReDim zV(5)
    xV(1) = 2
    yV(1) = 2
```

```
    xV(2) = 2
    yV(2) = 12
    xV(3) = 12
    yV(3) = 12
    xV(4) = 12
    yV(4) = 2
    xV(5) = 10
    yV(5) = -1
    Fill.iType = Avax.AvaxFillType.Pattern f
    Fill.lHatchIndex = 2
    iClr = 1
    iWidth = 2
    Fill.iBgBackColor = 4
    AvaxHandle = cAvax1.Add_Shape(xV, yV, zV, vItemProperties, Fill.iType, Fill.lHatchIndex,
    Fill.iBgForeColor, Fill.iBgBackColor, Fill.sPattern, Fill.zOrder, Fill.lFlags, iClr, iWidth, iStyle,
    iLayer, PolylineType)
End Sub
```

Add_ShapeArc

cAvax method

Purpose

This function adds a closed arc shape with a specific fill type and z order

Fill resources

The full overload accepts an AvaxFillType family and its pattern, colors, flags and z-order. BMP and Windows metafile fills accept a local file path or an already embedded image resource name. The control validates and stores the image bytes before returning the new handle. You can then remove the source file; saving the drawing retains the embedded image.

An invalid fill or missing image returns zero without inserting an object, changing the caller's ByRef arguments or consuming Undo/Redo history. The short overload retains its R2 default fill.

VB.NET declarations

```
Function Add_ShapeArc(ByRef x1 As Double, ByRef y1 As Double, ByRef x2 As Double, ByRef y2 As Double,
    ByRef xc As Double, ByRef yc As Double, ByRef vItemProperties As Object, ByRef Fill_iType As Byte, ByRef
    Fill_lHatchIndex As Integer, ByRef Fill_iInForeColor As Integer, ByRef Fill_iInBackColor As Integer, ByRef
     Fill_sPattern As String, ByRef Fill_zOrder As Integer, ByRef Fill_lFlags As Integer, ByRef nVert As
    Integer, ByRef iClr As Integer, ByRef iWidth As Integer, ByRef iStyle As Integer, ByRef iLayer As Integer)
     As Integer

Function Add_ShapeArc(ByRef x1 As Double, ByRef y1 As Double, ByRef x2 As Double, ByRef y2 As Double,
    ByRef xc As Double, ByRef yc As Double) As Integer
```

Parameters

Name	Type and passing	Meaning
x1	ByRef x1 As Double	Is x coordinate of the arc's start point
y1	ByRef y1 As Double	Is y coordinate of the arc's start point
x2	ByRef x2 As Double	X coordinate defining the end direction from the center. If its radius differs from the start radius, it is projected to the start-radius circle.
y2	ByRef y2 As Double	Y coordinate defining the end direction; sweep proceeds counterclockwise. The original example has unequal endpoint radii.
xc	ByRef xc As Double	Is x coordinate of the arc's circle center
yc	ByRef yc As Double	Is y coordinate of the arc's circle center
vItemProperties	ByRef vItemProperties As Object	One-dimensional String() or Object() containing strings/Nothing, with logical properties starting at index 1. The control stores its own copy and retains the original caller reference. Nothing means no properties, except Add_Data which requires an array.
Fill_iType	ByRef Fill_iType As Byte	A Byte expression that defines the AvaxFillType
Fill_lHatchIndex	ByRef Fill_lHatchIndex As Integer	Is the fill hatch index
Fill_iInForeColor	ByRef Fill_iInForeColor As Integer	Is the stroke color
Fill_iInBackColor	ByRef Fill_iInBackColor As Integer	Is the fill color
Fill_sPattern	ByRef Fill_sPattern As String	A String expression that defines the fill pattern's data (see also function CreateACADHatchPattern or CreateExVectorHatchPattern)
Fill_zOrder	ByRef Fill_zOrder As Integer	Is the object's elevation in the whole drawing
Fill_lFlags	ByRef Fill_lFlags As Integer	Is a numeric expresion which defines some more parameters (see also function CreatePatternFlags)
nVert	ByRef nVert As Integer	Effective segment count (Integer). Zero or -1 chooses automatic resolution and writes the effective count back ByRef. An explicit count from 3 through 1000000 is preserved. Closed circles/ellipses store one extra exact closing vertex; nVert itself is not incremented.
iClr	ByRef iClr As Integer	-1 uses the corresponding current pen setting. Other negative values become 0. Positive values use Integer storage. When declared ByRef, the effective value is written back only after successful validation.
iWidth	ByRef iWidth As Integer	-1 uses the current pen. Nonpositive resolved widths become 1, as in R2. Positive values use Integer storage. When declared ByRef, the effective value is written back only after successful validation.
iStyle	ByRef iStyle As Integer	-1 uses the corresponding current pen setting. Other negative values become 0. Positive values use Integer storage. When declared ByRef, the effective value is written back only after successful validation.
iLayer	ByRef iLayer As Integer	-1 uses the corresponding current pen setting. Other negative values become 0. Positive values use Integer storage. When declared ByRef, the effective value is written back only after successful validation.

Use the declaration for the overload you call. ByRef arguments can receive outputs; declare variables and arrays with the exact indicated type. Optional defaults are shown in the declaration.

Return value

Integer - An Avax handle on success, zero on error.

Remarks

For more details for the fill parameters see the example ShapeFill.vbp

Numeric changes from v2

Update the following variables, arrays or event-data receivers when porting the corresponding v2 call. The declarations above show the current overloads.

Parameter or value	v2 type	v3 type
x1	Single	Double
y1	Single	Double
x2	Single	Double
y2	Single	Double
xc	Single	Double
yc	Single	Double
Fill_iInForeColor	Short	Integer
Fill_iInBackColor	Short	Integer
iClr	Short	Integer
iWidth	Short	Integer
iStyle	Short	Integer
iLayer	Short	Integer

For ByRef parameters, use a variable with the exact v3 type. Preserve array dimensions and the documented indexing. Moving from v2 to v3 explains conversion and legacy file-storage rules.

Example

Setup

Add an Avax.cAvax Windows Forms control named `cAvax1`. Paste the procedures into the form class. Example setup and supporting files explains references, initialization and file paths.

Add these Windows Forms controls or components to the form. Fields referenced by Handles clauses must be declared WithEvents, as the Visual Studio designer normally does.

Name	Type
Command1	Button

VB.NET example

```
Private Sub Form1_Load(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs) Handles
    MyBase.Load
    cAvax1.StartAvax()
End Sub
Private Sub Form1_FormClosed(ByVal eventSender As System.Object, ByVal eventArgs As
    System.Windows.Forms.FormClosedEventArgs) Handles Me.FormClosed
    cAvax1.EndAvax()
End Sub
Private Sub Command1_Click(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs)
    Handles Command1.Click
    Dim x1 As Double
    Dim y1 As Double
    Dim x2 As Double
    Dim y2 As Double
    Dim xc As Double
    Dim yc As Double
    Dim vItemProperties As Object = Nothing
    Dim Fill_iType As Byte
    Dim Fill_lHatchIndex As Integer
    Dim Fill_iInForeColor As Integer
    Dim Fill_iInBackColor As Integer
```

```
    Dim Fill_sPattern As String = String.Empty
    Dim Fill_zOrder As Integer
    Dim Fill_lFlags As Integer
    Dim nVert As Integer
    Dim iClr As Integer
    Dim iWidth As Integer
    Dim iStyle As Integer
    Dim iLayer As Integer
    Dim AvaxHandle As Integer
    x1 = 11 : y1 = 6
    x2 = 7 : y2 = 8
    xc = 5 : yc = 5
    iClr = 1
    nVert = 72
    AvaxHandle = cAvax1.Add_ShapeArc(x1, y1, x2, y2, xc, yc, vItemProperties, Fill_iType,
    Fill_lHatchIndex, Fill_iInForeColor, Fill_iInBackColor, Fill_sPattern, Fill_zOrder, Fill_lFlags, nVert,
    iClr, iWidth, iStyle, iLayer)
End Sub
```

Add_ShapeCircle

cAvax method

Purpose

This function adds a circle with a specific fill type and z order

Fill resources

The full overload accepts an AvaxFillType family and its pattern, colors, flags and z-order. BMP and Windows metafile fills accept a local file path or an already embedded image resource name. The control validates and stores the image bytes before returning the new handle. You can then remove the source file; saving the drawing retains the embedded image.

An invalid fill or missing image returns zero without inserting an object, changing the caller's ByRef arguments or consuming Undo/Redo history. The short overload retains its R2 default fill.

VB.NET declarations

```
Function Add_ShapeCircle(ByRef xc As Double, ByRef yc As Double, ByRef R As Double, ByRef vItemProperties
    As Object, ByRef Fill iType As Byte, ByRef Fill lHatchIndex As Integer, ByRef Fill iInForeColor As
    Integer, ByRef Fill iInBackColor As Integer, ByRef Fill sPattern As String, ByRef Fill zOrder As Integer,
    ByRef Fill lFlags As Integer, ByRef nVert As Integer, ByRef iClr As Integer, ByRef iWidth As Integer,
    ByRef iStyle As Integer, ByRef iLayer As Integer) As Integer

Function Add_ShapeCircle(ByRef xc As Double, ByRef yc As Double, ByRef R As Double) As Integer
```

Parameters

Name	Type and passing	Meaning
xc	ByRef xc As Double	Is the x coordinate of the circle's center
yc	ByRef yc As Double	Is the y coordinate of the circle's center
R	ByRef R As Double	Is the circle's radius
vItemProperties	ByRef vItemProperties As Object	One-dimensional String() or Object() containing strings/Nothing, with logical properties starting at index 1. The control stores its own copy and retains the original caller reference. Nothing means no properties, except Add_Data which requires an array.
Fill_iType	ByRef Fill_iType As Byte	A Byte expression that defines the AvaxFillType
Fill_lHatchIndex	ByRef Fill_lHatchIndex As Integer	Is the fill hatch index
Fill_iInForeColor	ByRef Fill_iInForeColor As Integer	Is the stroke color
Fill_iInBackColor	ByRef Fill_iInBackColor As Integer	Is the fill color
Fill_sPattern	ByRef Fill_sPattern As String	A String expression that defines the fill pattern's data (see also function CreateACADHatchPattern or CreateExVectorHatchPattern)
Fill_zOrder	ByRef Fill_zOrder As Integer	Is the object's elevation in the whole drawing
Fill_lFlags	ByRef Fill_lFlags As Integer	Is a numeric expresion which defines some more parameters (see also function CreatePatternFlags)
nVert	ByRef nVert As Integer	Effective segment count (Integer). Zero or -1 chooses automatic resolution and writes the effective count back ByRef. An explicit count from 3 through 1000000 is preserved. Closed circles/ellipses store one extra exact closing vertex; nVert itself is not incremented. Circle values 1 and 2 also request automatic resolution, as in R2.
iClr	ByRef iClr As Integer	-1 uses the corresponding current pen setting. Other negative values become 0. Positive values use Integer storage. When declared ByRef, the effective value is written back only after successful validation.
iWidth	ByRef iWidth As Integer	-1 uses the current pen. Nonpositive resolved widths become 1, as in R2. Positive values use Integer storage. When declared ByRef, the effective value is written back only after successful validation.
iStyle	ByRef iStyle As Integer	-1 uses the corresponding current pen setting. Other negative values become 0. Positive values use Integer storage. When declared ByRef, the effective value is written back only after successful validation.
iLayer	ByRef iLayer As Integer	-1 uses the corresponding current pen setting. Other negative values become 0. Positive values use Integer storage. When declared ByRef, the effective value is written back only after successful validation.

Use the declaration for the overload you call. ByRef arguments can receive outputs; declare variables and arrays with the exact indicated type. Optional defaults are shown in the declaration.

Return value

Integer - An Avax handle on success, zero on error.

Remarks

For more details for the fill parameters see the example ShapeFill.vbp

Numeric changes from v2

Update the following variables, arrays or event-data receivers when porting the corresponding v2 call. The declarations above show the current overloads.

Parameter or value	v2 type	v3 type
xc	Single	Double

Parameter or value	v2 type	v3 type
yc	Single	Double
R	Single	Double
Fill_iInForeColor	Short	Integer
Fill_iInBackColor	Short	Integer
iClr	Short	Integer
iWidth	Short	Integer
iStyle	Short	Integer
iLayer	Short	Integer

For ByRef parameters, use a variable with the exact v3 type. Preserve array dimensions and the documented indexing. Moving from v2 to v3 explains conversion and legacy file-storage rules.

Example

Setup

Add an Avax.cAvax Windows Forms control named `cAvax1`. Paste the procedures into the form class. Example setup and supporting files explains references, initialization and file paths.

Add these Windows Forms controls or components to the form. Fields referenced by Handles clauses must be declared WithEvents, as the Visual Studio designer normally does.

Name	Type
Command1	Button

VB.NET example

```
Private Sub Form1_Load(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs) Handles
    MyBase.Load
    cAvax1.StartAvax()
End Sub
Private Sub Form1_FormClosed(ByVal eventSender As System.Object, ByVal eventArgs As
    System.Windows.Forms.FormClosedEventArgs) Handles Me.FormClosed
    cAvax1.EndAvax()
End Sub
Private Sub Command1_Click(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs)
    Handles Command1.Click
    Dim xc As Double
    Dim yc As Double
    Dim r As Double
    Dim vItemProperties As Object = Nothing
    Dim Fill_iType As Byte
    Dim Fill_lHatchIndex As Integer
    Dim Fill_iInForeColor As Integer
    Dim Fill_iInBackColor As Integer
    Dim Fill_sPattern As String = String.Empty
    Dim Fill_zOrder As Integer
    Dim Fill_lFlags As Integer
    Dim nVert As Integer
    Dim iClr As Integer
    Dim iWidth As Integer
    Dim iStyle As Integer
    Dim iLayer As Integer
    Dim AvaxHandle As Integer
    ' Define the coordinates of the circle's center
    xc = 8 : yc = 8
    ' Define the radious of the circle
    r = 5
    iClr = 5 'Color RED is selected
```

```
    'Fill with solid magenta color
    Fill.iInForeColor = 3
    AvaxHandle = cAvax1.Add_ShapeCircle(xc, yc, r, vItemProperties, Fill.iType, Fill.lHatchIndex,
    Fill.iInForeColor, Fill.iInBackColor, Fill.sPattern, Fill.zOrder, Fill.lFlags, nVert, iClr, iWidth,
    iStyle, iLayer)
End Sub
```

Add_ShapeDocument

API reference

Creates an AVAX document inside a drawing shape. Use the returned handle to manage the complete document as one object.

Declarations

```
Public Function Add_ShapeDocument(ByVal sDocumentText As String, ByVal x As Double, ByVal y As Double,
    ByVal z As Double, ByVal DocumentWidth As Double, ByVal DocumentTextHeight As Double, ByVal Angle As
    Double, ByVal vItemProperties As Object, ByVal iFontNumber As Integer, ByVal iClr As Integer, ByVal iWidth
     As Integer, ByVal iStyle As Integer, ByVal iLayer As Integer, Optional ByVal zOrder As Integer = 0,
    Optional ByVal fireNewItemsEvent As Boolean = True) As Integer
```

Parameters

Name	Purpose
sDocumentText	Document source text, including line breaks and supported AVAX document markup.
x	World X position (Double).
y	World Y position (Double).
z	World Z position (Double).
DocumentWidth	Positive document layout width in drawing units (Double).
DocumentTextHeight	Positive base text height in drawing units (Double).
Angle	Rotation angle in degrees (Double); the short Add_Document overload uses zero.
vItemProperties	Optional application properties stored with the drawing object; pass Nothing for no properties.
iFontNumber	Font table index (Integer). The short Add_Document overload uses font1.
iClr	Palette index (Integer); -1 uses the current pen color.
iWidth	Line width (Integer); -1 uses the current pen width.
iStyle	Line style index (Integer); -1 uses the current pen style.
iLayer	Layer index (Integer); -1 uses the current pen layer.
zOrder	Drawing order for the document shape (Integer), zero by default.
fireNewItemsEvent	True (default) announces the completed shape through NewUserItems. False suppresses that creation notification while retaining Change and Undo.

Return value

The new Integer handle, or zero if the control is not started/enabled or the request is rejected. Call StartAvax before inserting documents.

Behavior

The method stores one document shape and one reversible edit. NewUserItems precedes Change when fireNewItemsEvent=True. Set it to False to populate a drawing without announcing this creation.

The drawing and complete reversible record exist before the notifications. A callback that supersedes the drawing or control prevents stale trailing notifications. Handler exceptions propagate after the committed edit. PauseAvaxEvents suppresses notifications without replay. BeginUpdate and EndUpdate suspend display while model notifications retain their normal timing. Invalid geometry is rejected before insertion.

VB.NET example

Pass an initialized control. Coordinates, layout width and text height are Double in v3; handles and font/layer indices are Integer.

```
Option Strict On
Imports Avax

Public Module AddShapeDocumentExample
```

```
    Public Function Add(cad As cAvax, announce As Boolean) As Integer
        Return cad.Add_ShapeDocument("AVAX example", 10.5, 20.25, 0,
            50, 2, 0, Nothing, 1, -1, -1, -1, -1, 0, announce)
    End Function
End Module
```

See Add_Document and NewUserItems.

Add_ShapeEllipse

cAvax method

Purpose

This function adds an ellipse with a specific fill type and z order

Fill resources

The full overload accepts an AvaxFillType family and its pattern, colors, flags and z-order. BMP and Windows metafile fills accept a local file path or an already embedded image resource name. The control validates and stores the image bytes before returning the new handle. You can then remove the source file; saving the drawing retains the embedded image.

An invalid fill or missing image returns zero without inserting an object, changing the caller's ByRef arguments or consuming Undo/Redo history. The short overload retains its R2 default fill.

VB.NET declarations

```
Function Add_ShapeEllipse(ByRef xc As Double, ByRef yc As Double, ByRef aR As Double, ByRef bR As Double,
    ByRef vItemProperties As Object, ByRef Fill iType As Byte, ByRef Fill lHatchIndex As Integer, ByRef
    Fill iInForeColor As Integer, ByRef Fill iInBackColor As Integer, ByRef Fill sPattern As String, ByRef
    Fill zOrder As Integer, ByRef Fill lFlags As Integer, ByRef nVert As Integer, ByRef iClr As Integer, ByRef
     iWidth As Integer, ByRef iStyle As Integer, ByRef iLayer As Integer, Optional ByVal fireNewItemsEvent As
    Boolean = True) As Integer

Function Add_ShapeEllipse(ByRef xc As Double, ByRef yc As Double, ByRef aR As Double, ByRef bR As Double,
    Optional ByVal fireNewItemsEvent As Boolean = True) As Integer
```

Parameters

Name	Type and passing	Meaning
xc	ByRef xc As Double	Is the x coordinate of the ellipse's center
yc	ByRef yc As Double	Is the y coordinate of the ellipse's center
aR	ByRef aR As Double	Is the ellipse's horizontal radius
bR	ByRef bR As Double	Is the ellipse's vertical radius
vItemProperties	ByRef vItemProperties As Object	One-dimensional String() or Object() containing strings/Nothing, with logical properties starting at index 1. The control stores its own copy and retains the original caller reference. Nothing means no properties, except Add_Data which requires an array.
Fill_iType	ByRef Fill_iType As Byte	A Byte expression that defines the AvaxFillType
Fill_lHatchIndex	ByRef Fill_lHatchIndex As Integer	Is the fill hatch index
Fill_iInForeColor	ByRef Fill_iInForeColor As Integer	Is the stroke color
Fill_iInBackColor	ByRef Fill_iInBackColor As Integer	Is the fill color
Fill_sPattern	ByRef Fill_sPattern As String	A String expression that defines the fill pattern's data (see also function CreateACADHatchPattern or CreateExVectorHatchPattern)
Fill_zOrder	ByRef Fill_zOrder As Integer	Is the object's elevation in the whole drawing
Fill_lFlags	ByRef Fill_lFlags As Integer	Is a numeric expresion which defines some more parameters (see also function CreatePatternFlags)
nVert	ByRef nVert As Integer	Effective segment count (Integer). Zero or -1 chooses automatic resolution and writes the effective count back ByRef. An explicit count from 3 through 1000000 is preserved. Closed circles/ellipses store one extra exact closing vertex; nVert itself is not incremented.
iClr	ByRef iClr As Integer	-1 uses the corresponding current pen setting. Other negative values become 0. Positive values use Integer storage. When declared ByRef, the effective value is written back only after successful validation.
iWidth	ByRef iWidth As Integer	-1 uses the current pen. Nonpositive resolved widths become 1, as in R2. Positive values use Integer storage. When declared ByRef, the effective value is written back only after successful validation.
iStyle	ByRef iStyle As Integer	-1 uses the corresponding current pen setting. Other negative values become 0. Positive values use Integer storage. When declared ByRef, the effective value is written back only after successful validation.
iLayer	ByRef iLayer As Integer	-1 uses the corresponding current pen setting. Other negative values become 0. Positive values use Integer storage. When declared ByRef, the effective value is written back only after successful validation.
fireNewItemsEvent	Optional ByVal fireNewItemsEvent As Boolean = True	True includes the created objects in the NewUserItems notification. False suppresses that notification for this call.

Use the declaration for the overload you call. ByRef arguments can receive outputs; declare variables and arrays with the exact indicated type. Optional defaults are shown in the declaration.

Return value

Integer - An Avax handle on success, zero on error.

Remarks

For more details for the fill parameters see the example ShapeFill.vbp

Numeric changes from v2

Update the following variables, arrays or event-data receivers when porting the corresponding v2 call. The declarations above show the current overloads.

Parameter or value	v2 type	v3 type
xc	Single	Double
yc	Single	Double
aR	Single	Double
bR	Single	Double
Fill_iInForeColor	Short	Integer
Fill_iInBackColor	Short	Integer
iClr	Short	Integer
iWidth	Short	Integer
iStyle	Short	Integer
iLayer	Short	Integer

For ByRef parameters, use a variable with the exact v3 type. Preserve array dimensions and the documented indexing. Moving from v2 to v3 explains conversion and legacy file-storage rules.

Example

Setup

Add an Avax.cAvax Windows Forms control named `cAvax1`. Paste the procedures into the form class. Example setup and supporting files explains references, initialization and file paths.

Add these Windows Forms controls or components to the form. Fields referenced by Handles clauses must be declared WithEvents, as the Visual Studio designer normally does.

Name	Type
Command1	Button

VB.NET example

```
Private Sub Form1_Load(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs) Handles
    MyBase.Load
    cAvax1.StartAvax()
End Sub
Private Sub Form1_FormClosed(ByVal eventSender As System.Object, ByVal eventArgs As
    System.Windows.Forms.FormClosedEventArgs) Handles Me.FormClosed
    cAvax1.EndAvax()
End Sub
Private Sub Command1_Click(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs)
    Handles Command1.Click
    Dim xc As Double
    Dim yc As Double
    Dim aR As Double
    Dim bR As Double
    Dim vItemProperties As Object = Nothing
    Dim Fill_iType As Byte
    Dim Fill_lHatchIndex As Integer
    Dim Fill_iInForeColor As Integer
    Dim Fill_iInBackColor As Integer
    Dim Fill_sPattern As String = String.Empty
    Dim Fill_zOrder As Integer
    Dim Fill_lFlags As Integer
    Dim nVert As Integer
    Dim iClr As Integer
    Dim iWidth As Integer
    Dim iStyle As Integer
    Dim iLayer As Integer
    Dim AvaxHandle As Integer
    ' Define the coordinates of the center
```

```
    xc = 8 : yc = 5
    ' Define the radius of the ellipse
    aR = 8 : bR = 6
    iClr = 4 'Color CYAN is selected
    iWidth = 2
    nVert = 36
    Fill iInForeColor = 1 'Color RED is selected
    AvaxHandle = cAvax1.Add_ShapeEllipse(xc, yc, aR, bR, vItemProperties, Fill iType, Fill lHatchIndex,
    Fill iInForeColor, Fill iInBackColor, Fill sPattern, Fill zOrder, Fill lFlags, nVert, iClr, iWidth,
    iStyle, iLayer)
End Sub
```

Add_ShapeLine

cAvax method

Purpose

This function adds a line segment. The difference from add_Line is that this line segment has z order

VB.NET declarations

```
Function Add_ShapeLine(ByRef x1 As Double, ByRef y1 As Double, ByRef x2 As Double, ByRef y2 As Double,
    ByRef vItemProperties As Object, ByRef iClr As Integer, ByRef iWidth As Integer, ByRef iStyle As Integer,
    ByRef iLayer As Integer, ByRef Fill_zOrder As Integer) As Integer

Function Add_ShapeLine(ByRef x1 As Double, ByRef y1 As Double, ByRef x2 As Double, ByRef y2 As Double) As
    Integer
```

Parameters

Name	Type and passing	Meaning
x1	ByRef x1 As Double	Is the x coordinate of the line segment's first point
y1	ByRef y1 As Double	Is the y coordinate of the line segment's first point
x2	ByRef x2 As Double	Is the x coordinate of the line segment's end point
y2	ByRef y2 As Double	Is the y coordinate of the line segment's end point
vItemProperties	ByRef vItemProperties As Object	One-dimensional String() or Object() containing strings/Nothing, with logical properties starting at index 1. The control stores its own copy and retains the original caller reference. Nothing means no properties, except Add_Data which requires an array.
iClr	ByRef iClr As Integer	-1 uses the corresponding current pen setting. Other negative values become 0. Positive values use Integer storage. When declared ByRef, the effective value is written back only after successful validation.
iWidth	ByRef iWidth As Integer	-1 uses the current pen. Nonpositive resolved widths become 1, as in R2. Positive values use Integer storage. When declared ByRef, the effective value is written back only after successful validation.
iStyle	ByRef iStyle As Integer	-1 uses the corresponding current pen setting. Other negative values become 0. Positive values use Integer storage. When declared ByRef, the effective value is written back only after successful validation.
iLayer	ByRef iLayer As Integer	-1 uses the corresponding current pen setting. Other negative values become 0. Positive values use Integer storage. When declared ByRef, the effective value is written back only after successful validation.
Fill_zOrder	ByRef Fill_zOrder As Integer	Is the object's elevation in the whole drawing

Use the declaration for the overload you call. ByRef arguments can receive outputs; declare variables and arrays with the exact indicated type. Optional defaults are shown in the declaration.

Return value

Integer - An Avax handle on success, zero on error.

Remarks

For more details for the fill parameters see the example ShapeFill.vbp

Numeric changes from v2

Update the following variables, arrays or event-data receivers when porting the corresponding v2 call. The declarations above show the current overloads.

Parameter or value	v2 type	v3 type
x1	Single	Double
y1	Single	Double
x2	Single	Double
y2	Single	Double
iClr	Short	Integer
iWidth	Short	Integer
iStyle	Short	Integer
iLayer	Short	Integer

For ByRef parameters, use a variable with the exact v3 type. Preserve array dimensions and the documented indexing. Moving from v2 to v3 explains conversion and legacy file-storage rules.

Example

Setup

Add an Avax.cAvax Windows Forms control named `cAvax1`. Paste the procedures into the form class. Example setup and supporting files explains references, initialization and file paths.

Add these Windows Forms controls or components to the form. Fields referenced by Handles clauses must be declared WithEvents, as the Visual Studio designer normally does.

Name	Type
Command1	Button

VB.NET example

```
Private Sub Form1_Load(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs) Handles
    MyBase.Load
    cAvax1.StartAvax()
End Sub
Private Sub Form1_FormClosed(ByVal eventSender As System.Object, ByVal eventArgs As
    System.Windows.Forms.FormClosedEventArgs) Handles Me.FormClosed
    cAvax1.EndAvax()
End Sub
Private Sub Command1_Click(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs)
    Handles Command1.Click
    Dim x1 As Double
    Dim y1 As Double
    Dim x2 As Double
    Dim y2 As Double
    Dim vItemProperties As Object = Nothing
    Dim iClr As Integer
    Dim iWidth As Integer
    Dim iStyle As Integer
    Dim iLayer As Integer
    Dim Fill_zOrder As Integer
    Dim AvaxHandle As Integer
    x1 = 0
    y1 = 0
    x2 = 12
    y2 = 0
    iClr = 1
    iStyle = 1
    AvaxHandle = cAvax1.Add_ShapeLine(x1, y1, x2, y2, vItemProperties, iClr, iWidth, iStyle, iLayer,
    Fill_zOrder)
End Sub
```

Add_ShapePoint

cAvax method

Purpose

This function adds a point. The difference from add_Point is that this point has z order

VB.NET declarations

```
Function Add_ShapePoint(ByRef x1 As Double, ByRef y1 As Double, ByRef vItemProperties As Object, ByRef
    iClr As Integer, ByRef iWidth As Integer, ByRef iPointStyle As Integer, ByRef iLayer As Integer, ByRef
    Fill_zOrder As Integer) As Integer

Function Add_ShapePoint(ByRef x1 As Double, ByRef y1 As Double) As Integer
```

Parameters

Name	Type and passing	Meaning
x1	ByRef x1 As Double	Is the x coordinate of the point
y1	ByRef y1 As Double	Is the y coordinate of the point
vItemProperties	ByRef vItemProperties As Object	One-dimensional String() or Object() containing strings/Nothing, with logical properties starting at index 1. The control stores its own copy and retains the original caller reference. Nothing means no properties, except Add_Data which requires an array.
iClr	ByRef iClr As Integer	-1 uses the corresponding current pen setting. Other negative values become 0. Positive values use Integer storage. When declared ByRef, the effective value is written back only after successful validation.
iWidth	ByRef iWidth As Integer	-1 uses the current pen. Nonpositive resolved widths become 1, as in R2. Positive values use Integer storage. When declared ByRef, the effective value is written back only after successful validation.
iPointStyle	ByRef iPointStyle As Integer	-1 uses the corresponding current pen setting. Other negative values become 0. Positive values use Integer storage. When declared ByRef, the effective value is written back only after successful validation.
iLayer	ByRef iLayer As Integer	-1 uses the corresponding current pen setting. Other negative values become 0. Positive values use Integer storage. When declared ByRef, the effective value is written back only after successful validation.
Fill_zOrder	ByRef Fill_zOrder As Integer	Is the object's elevation in the whole drawing

Use the declaration for the overload you call. ByRef arguments can receive outputs; declare variables and arrays with the exact indicated type. Optional defaults are shown in the declaration.

Return value

Integer - An Avax handle on success, zero on error.

Remarks

For more details for the fill parameters see the example ShapeFill.vbp

Numeric changes from v2

Update the following variables, arrays or event-data receivers when porting the corresponding v2 call. The declarations above show the current overloads.

Parameter or value	v2 type	v3 type
x1	Single	Double

Parameter or value	v2 type	v3 type
y1	Single	Double
iClr	Short	Integer
iWidth	Short	Integer
iPointStyle	Short	Integer
iLayer	Short	Integer

For ByRef parameters, use a variable with the exact v3 type. Preserve array dimensions and the documented indexing. Moving from v2 to v3 explains conversion and legacy file-storage rules.

Example

Setup

Add an Avax.cAvax Windows Forms control named `cAvax1`. Paste the procedures into the form class. Example setup and supporting files explains references, initialization and file paths.

Add these Windows Forms controls or components to the form. Fields referenced by Handles clauses must be declared WithEvents, as the Visual Studio designer normally does.

Name	Type
Command1	Button

VB.NET example

```
Private Sub Form1_Load(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs) Handles
    MyBase.Load
    cAvax1.StartAvax()
End Sub
Private Sub Form1_FormClosed(ByVal eventSender As System.Object, ByVal eventArgs As
    System.Windows.Forms.FormClosedEventArgs) Handles Me.FormClosed
    cAvax1.EndAvax()
End Sub
Private Sub Command1_Click(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs)
    Handles Command1.Click
    Dim x1 As Double
    Dim y1 As Double
    Dim vItemProperties As Object = Nothing
    Dim iClr As Integer
    Dim iWidth As Integer
    Dim iPointStyle As Integer
    Dim iLayer As Integer
    Dim Fill_zOrder As Integer
    Dim AvaxHandle As Integer
    x1 = 1
    y1 = 1
    iPointStyle = 2
    AvaxHandle = cAvax1.Add_ShapePoint(x1, y1, vItemProperties, iClr, iWidth, iPointStyle, iLayer,
    Fill_zOrder)
End Sub
```

Add_ShapePolyLine

cAvax method

Purpose

This function adds a polyline. The difference from add_Polyline is that this polyline has z order

VB.NET declarations

```
Function Add_ShapePolyLine(ByRef xV As Double(), ByRef yV As Double(), ByRef zV As Double(), ByRef
    vItemProperties As Object, ByRef iClr As Integer, ByRef iWidth As Integer, ByRef iStyle As Integer, ByRef
    iLayer As Integer, ByRef Fill_zOrder As Integer, ByRef PolylineType As AvaxPolylineType) As Integer

Function Add_ShapePolyLine(ByRef xV As Double(), ByRef yV As Double(), ByRef zV As Double()) As Integer
```

Parameters

Name	Type and passing	Meaning
xV	ByRef xV As Double()	An array for the x coordinates of the poline's apexes
yV	ByRef yV As Double()	An array for the y coordinates of the poline's apexes
zV	ByRef zV As Double()	An array for the z coordinates of the poline's apexes
vItemProperties	ByRef vItemProperties As Object	One-dimensional String() or Object() containing strings/Nothing, with logical properties starting at index 1. The control stores its own copy and retains the original caller reference. Nothing means no properties, except Add_Data which requires an array.
iClr	ByRef iClr As Integer	-1 uses the corresponding current pen setting. Other negative values become 0. Positive values use Integer storage. When declared ByRef, the effective value is written back only after successful validation.
iWidth	ByRef iWidth As Integer	-1 uses the current pen. Nonpositive resolved widths become 1, as in R2. Positive values use Integer storage. When declared ByRef, the effective value is written back only after successful validation.
iStyle	ByRef iStyle As Integer	-1 uses the corresponding current pen setting. Other negative values become 0. Positive values use Integer storage. When declared ByRef, the effective value is written back only after successful validation.
iLayer	ByRef iLayer As Integer	-1 uses the corresponding current pen setting. Other negative values become 0. Positive values use Integer storage. When declared ByRef, the effective value is written back only after successful validation.
Fill_zOrder	ByRef Fill_zOrder As Integer	Is the object's elevation in the whole drawing
PolylineType	ByRef PolylineType As AvaxPolylineType	An expression that defines the AvaxPolylineType

Use the declaration for the overload you call. ByRef arguments can receive outputs; declare variables and arrays with the exact indicated type. Optional defaults are shown in the declaration.

Return value

Integer - An Avax handle on success, zero on error.

Remarks

For more details for the fill parameters see the example ShapeFill.vbp

Numeric changes from v2

Update the following variables, arrays or event-data receivers when porting the corresponding v2 call. The declarations above show the current overloads.

Parameter or value	v2 type	v3 type
xV	Single()	Double()
yV	Single()	Double()
zV	Single()	Double()
iClr	Short	Integer
iWidth	Short	Integer
iStyle	Short	Integer
iLayer	Short	Integer

For ByRef parameters, use a variable with the exact v3 type. Preserve array dimensions and the documented indexing. Moving from v2 to v3 explains conversion and legacy file-storage rules.

Example

Setup

Add an Avax.cAvax Windows Forms control named `cAvax1`. Paste the procedures into the form class. Example setup and supporting files explains references, initialization and file paths.

Add these Windows Forms controls or components to the form. Fields referenced by Handles clauses must be declared WithEvents, as the Visual Studio designer normally does.

Name	Type
Command1	Button

VB.NET example

```
Private Sub Form1_Load(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs) Handles
    MyBase.Load
    cAvax1.StartAvax()
End Sub
Private Sub Form1_FormClosed(ByVal eventSender As System.Object, ByVal eventArgs As
    System.Windows.Forms.FormClosedEventArgs) Handles Me.FormClosed
    cAvax1.EndAvax()
End Sub
Private Sub Command1_Click(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs)
    Handles Command1.Click
    Dim xV() As Double = Nothing
    Dim yV() As Double = Nothing
    Dim zV() As Double = Nothing
    Dim vItemProperties As Object = Nothing
    Dim iClr As Integer
    Dim iWidth As Integer
    Dim iStyle As Integer
    Dim iLayer As Integer
    Dim Fill_zOrder As Integer
    Dim PolylineType As Avax.AvaxPolylineType
    Dim Answer As Integer
    ReDim xV(5)
    ReDim yV(5)
    ReDim zV(5)
    xV(1) = 1 : yV(1) = 1 : zV(1) = 0
    xV(2) = 8 : yV(2) = 5 : zV(2) = 0
    xV(3) = 16 : yV(3) = 1 : zV(3) = 0
    xV(4) = 24 : yV(4) = 8 : zV(4) = 0
    xV(5) = 1 : yV(5) = 1 : zV(5) = 0
    iClr = 6
    iWidth = 3
    Answer = cAvax1.Add_ShapePolyLine(xV, yV, zV, vItemProperties, iClr, iWidth, iStyle, iLayer,
    Fill_zOrder, PolylineType)
End Sub
```

Add_ShapeRectangle

cAvax method

Purpose

This function adds a rectangle with a specific fill type and z order

Fill resources

The full overload accepts an AvaxFillType family and its pattern, colors, flags and z-order. BMP and Windows metafile fills accept a local file path or an already embedded image resource name. The control validates and stores the image bytes before returning the new handle. You can then remove the source file; saving the drawing retains the embedded image.

An invalid fill or missing image returns zero without inserting an object, changing the caller's ByRef arguments or consuming Undo/Redo history. The short overload retains its R2 default fill.

VB.NET declarations

```
Function Add_ShapeRectangle(ByRef x0 As Double, ByRef y0 As Double, ByRef nWidth As Double, ByRef nHeight
    As Double, ByRef vItemProperties As Object, ByRef Fill_iType As Byte, ByRef Fill_lHatchIndex As Integer,
    ByRef Fill_iInForeColor As Integer, ByRef Fill_iInBackColor As Integer, ByRef Fill_sPattern As String,
    ByRef Fill_zOrder As Integer, ByRef Fill_lFlags As Integer, ByRef iClr As Integer, ByRef iWidth As
    Integer, ByRef iStyle As Integer, ByRef iLayer As Integer) As Integer

Function Add_ShapeRectangle(ByRef x0 As Double, ByRef y0 As Double, ByRef nWidth As Double, ByRef nHeight
    As Double) As Integer
```

Parameters

Name	Type and passing	Meaning
x0	ByRef x0 As Double	Is the x coordinate of the rectangle's bottom left point
y0	ByRef y0 As Double	Is the y coordinate of the rectangle's bottom left point
nWidth	ByRef nWidth As Double	Is the width of the rectangle
nHeight	ByRef nHeight As Double	Is the height of the rectangle
vItemProperties	ByRef vItemProperties As Object	One-dimensional String() or Object() containing strings/Nothing, with logical properties starting at index 1. The control stores its own copy and retains the original caller reference. Nothing means no properties, except Add_Data which requires an array.
Fill_iType	ByRef Fill_iType As Byte	A Byte expression that defines the AvaxFillType
Fill_lHatchIndex	ByRef Fill_lHatchIndex As Integer	Is the fill hatch index
Fill_iInForeColor	ByRef Fill_iInForeColor As Integer	Is the stroke color
Fill_iInBackColor	ByRef Fill_iInBackColor As Integer	Is the fill color
Fill_sPattern	ByRef Fill_sPattern As String	A String expression that defines the fill pattern's data (see also function CreateACADHatchPattern or CreateExVectorHatchPattern)
Fill_zOrder	ByRef Fill_zOrder As Integer	Is the object's elevation in the whole drawing
Fill_lFlags	ByRef Fill_lFlags As Integer	Is a numeric expresion which defines some more parameters (see also function CreatePatternFlags)
iClr	ByRef iClr As Integer	-1 uses the corresponding current pen setting. Other negative values become 0. Positive values use Integer storage. When declared ByRef, the effective value is written back only after successful validation.
iWidth	ByRef iWidth As Integer	-1 uses the current pen. Nonpositive resolved widths become 1, as in R2. Positive values use Integer storage. When declared ByRef, the effective value is written back only after successful validation.
iStyle	ByRef iStyle As Integer	-1 uses the corresponding current pen setting. Other negative values become 0. Positive values use Integer storage. When declared ByRef, the effective value is written back only after successful validation.
iLayer	ByRef iLayer As Integer	-1 uses the corresponding current pen setting. Other negative values become 0. Positive values use Integer storage. When declared ByRef, the effective value is written back only after successful validation.

Use the declaration for the overload you call. ByRef arguments can receive outputs; declare variables and arrays with the exact indicated type. Optional defaults are shown in the declaration.

Return value

Integer - An Avax handle on success, zero on error.

Remarks

For more details for the fill parameters see the example ShapeFill.vbp

Numeric changes from v2

Update the following variables, arrays or event-data receivers when porting the corresponding v2 call. The declarations above show the current overloads.

Parameter or value	v2 type	v3 type
x0	Single	Double
y0	Single	Double
nWidth	Single	Double

Parameter or value	v2 type	v3 type
nHeight	Single	Double
Fill_iInForeColor	Short	Integer
Fill_iInBackColor	Short	Integer
iClr	Short	Integer
iWidth	Short	Integer
iStyle	Short	Integer
iLayer	Short	Integer

For ByRef parameters, use a variable with the exact v3 type. Preserve array dimensions and the documented indexing. Moving from v2 to v3 explains conversion and legacy file-storage rules.

Example

Setup

Add an Avax.cAvax Windows Forms control named `cAvax1`. Paste the procedures into the form class. Example setup and supporting files explains references, initialization and file paths.

Add these Windows Forms controls or components to the form. Fields referenced by Handles clauses must be declared WithEvents, as the Visual Studio designer normally does.

Name	Type
Command1	Button

VB.NET example

```
Private Sub Form1_Load(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs) Handles
    MyBase.Load
    cAvax1.StartAvax()
End Sub
Private Sub Form1_FormClosed(ByVal eventSender As System.Object, ByVal eventArgs As
    System.Windows.Forms.FormClosedEventArgs) Handles Me.FormClosed
    cAvax1.EndAvax()
End Sub
Private Sub Command1_Click(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs)
    Handles Command1.Click
    Dim x0 As Double
    Dim y0 As Double
    Dim nWidth As Double
    Dim nHeight As Double
    Dim vItemProperties As Object = Nothing
    Dim Fill_iType As Byte
    Dim Fill_lHatchIndex As Integer
    Dim Fill_iInForeColor As Integer
    Dim Fill_iInBackColor As Integer
    Dim Fill_sPattern As String = String.Empty
    Dim Fill_zOrder As Integer
    Dim Fill_lFlags As Integer
    Dim iClr As Integer
    Dim iWidth As Integer
    Dim iStyle As Integer
    Dim iLayer As Integer
    Dim AvaxHandle As Integer
    ' Define the starting point of the rectangle
    x0 = 3 : y0 = 3
    nWidth = 10 : nHeight = 5
    ' Define the drawing and the fill color
    iClr = 6
    Fill_iInForeColor = 2
    AvaxHandle = cAvax1.Add_ShapeRectangle(x0, y0, nWidth, nHeight, vItemProperties, Fill_iType,
```

```
    Fill lHatchIndex, Fill iInForeColor, Fill iInBackColor, Fill sPattern, Fill zOrder, Fill lFlags, iClr,
    iWidth, iStyle, iLayer)
End Sub
```

Add_ShapeRegularPolygon

cAvax method

Purpose

This function adds a regular polygon shape with a specific fill type and z order

Fill resources

The full overload accepts an AvaxFillType family and its pattern, colors, flags and z-order. BMP and Windows metafile fills accept a local file path or an already embedded image resource name. The control validates and stores the image bytes before returning the new handle. You can then remove the source file; saving the drawing retains the embedded image.

An invalid fill or missing image returns zero without inserting an object, changing the caller's ByRef arguments or consuming Undo/Redo history. The short overload retains its R2 default fill.

See Polyline, Polygon and Close commands for pending XYZ chains, exact closing edges, regular polygons, vertex counts, completion and complete VB.NET examples.

VB.NET declarations

```
Function Add_ShapeRegularPolygon(ByRef xc As Double, ByRef yc As Double, ByRef R As Double, ByRef nVert As
     Integer) As Integer

Function Add_ShapeRegularPolygon(ByRef xc As Double, ByRef yc As Double, ByRef R As Double, ByRef nVert As
     Integer, ByRef vItemProperties As Object, ByRef Fill_iType As Byte, ByRef Fill_lHatchIndex As Integer,
    ByRef Fill_iInForeColor As Integer, ByRef Fill_iInBackColor As Integer, ByRef Fill_sPattern As String,
    ByRef Fill_zOrder As Integer, ByRef Fill_lFlags As Integer, ByRef iClr As Integer, ByRef iWidth As
    Integer, ByRef iStyle As Integer, ByRef iLayer As Integer) As Integer
```

Parameters

Name	Type and passing	Meaning
xc	ByRef xc As Double	Is the x coordinate of the polygon's center
yc	ByRef yc As Double	Is the y coordinate of the polygon's center
R	ByRef R As Double	Is the radius of the circle contour of the polygon
nVert	ByRef nVert As Integer	Number of distinct polygon vertices, from 3 through 1000000. Remains unchanged ByRef. One extra exact closing vertex is stored. Circle resolution properties do not override this count.
vItemProperties	ByRef vItemProperties As Object	One-dimensional String() or Object() containing strings/Nothing, with logical properties starting at index 1. The control stores its own copy and retains the original caller reference. Nothing means no properties, except Add_Data which requires an array.
Fill_iType	ByRef Fill_iType As Byte	A Byte expression that defines the AvaxFillType
Fill_lHatchIndex	ByRef Fill_lHatchIndex As Integer	Is the fill hatch index
Fill_iInForeColor	ByRef Fill_iInForeColor As Integer	Is the stroke color
Fill_iInBackColor	ByRef Fill_iInBackColor As Integer	Is the fill color
Fill_sPattern	ByRef Fill_sPattern As String	A String expression that defines the fill pattern's data (see also function CreateACADHatchPattern or CreateExVectorHatchPattern)
Fill_zOrder	ByRef Fill_zOrder As Integer	Is the object's elevation in the whole drawing
Fill_lFlags	ByRef Fill_lFlags As Integer	Is a numeric expresion which defines some more parameters (see also function CreatePatternFlags)
iClr	ByRef iClr As Integer	-1 uses the corresponding current pen setting. Other negative values become 0. Positive values use Integer storage. When declared ByRef, the effective value is written back only after successful validation.
iWidth	ByRef iWidth As Integer	-1 uses the current pen. Nonpositive resolved widths become 1, as in R2. Positive values use Integer storage. When declared ByRef, the effective value is written back only after successful validation.
iStyle	ByRef iStyle As Integer	-1 uses the corresponding current pen setting. Other negative values become 0. Positive values use Integer storage. When declared ByRef, the effective value is written back only after successful validation.
iLayer	ByRef iLayer As Integer	-1 uses the corresponding current pen setting. Other negative values become 0. Positive values use Integer storage. When declared ByRef, the effective value is written back only after successful validation.

Use the declaration for the overload you call. ByRef arguments can receive outputs; declare variables and arrays with the exact indicated type. Optional defaults are shown in the declaration.

Return value

Integer - An Avax handle on success, zero on error.

Remarks

For more details for the fill parameters see the example ShapeFill.vbp

Numeric changes from v2

Update the following variables, arrays or event-data receivers when porting the corresponding v2 call. The declarations above show the current overloads.

Parameter or value	v2 type	v3 type
xc	Single	Double
yc	Single	Double

Parameter or value	v2 type	v3 type
R	Single	Double
Fill_iInForeColor	Short	Integer
Fill_iInBackColor	Short	Integer
iClr	Short	Integer
iWidth	Short	Integer
iStyle	Short	Integer
iLayer	Short	Integer

For ByRef parameters, use a variable with the exact v3 type. Preserve array dimensions and the documented indexing. Moving from v2 to v3 explains conversion and legacy file-storage rules.

Example

Setup

Add an Avax.cAvax Windows Forms control named `cAvax1`. Paste the procedures into the form class. Example setup and supporting files explains references, initialization and file paths.

Add these Windows Forms controls or components to the form. Fields referenced by Handles clauses must be declared WithEvents, as the Visual Studio designer normally does.

Name	Type
Command1	Button

VB.NET example

```
Private Sub Form1_Load(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs) Handles
    MyBase.Load
    cAvax1.StartAvax()
End Sub
Private Sub Form1_FormClosed(ByVal eventSender As System.Object, ByVal eventArgs As
    System.Windows.Forms.FormClosedEventArgs) Handles Me.FormClosed
    cAvax1.EndAvax()
End Sub
Private Sub Command1_Click(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs)
    Handles Command1.Click
    Dim xc As Double
    Dim yc As Double
    Dim r As Double
    Dim nVert As Integer
    Dim vItemProperties As Object = Nothing
    Dim Fill_iType As Byte
    Dim Fill_lHatchIndex As Integer
    Dim Fill_iInForeColor As Integer
    Dim Fill_iInBackColor As Integer
    Dim Fill_sPattern As String = String.Empty
    Dim Fill_zOrder As Integer
    Dim Fill_lFlags As Integer
    Dim iClr As Integer
    Dim iWidth As Integer
    Dim iStyle As Integer
    Dim iLayer As Integer
    Dim AvaxHandle As Integer
    xc = 10 : yc = 10
    r = 5 : nVert = 6
    iClr = 7
    Fill_iInForeColor = 2
    AvaxHandle = cAvax1.Add_ShapeRegularPolygon(xc, yc, r, nVert, vItemProperties, Fill_iType,
    Fill_lHatchIndex, Fill_iInForeColor, Fill_iInBackColor, Fill_sPattern, Fill_zOrder, Fill_lFlags, iClr,
    iWidth, iStyle, iLayer)
```

```
    xc = 10 : yc = 10
    r = 4 : nVert = 6
    iClr = 7
    Fill iInForeColor = 3
    AvaxHandle = cAvax1.Add_ShapeRegularPolygon(xc, yc, r, nVert, vItemProperties, Fill.iType,
    Fill.lHatchIndex, Fill.iInForeColor, Fill.iInBackColor, Fill.sPattern, Fill.zOrder, Fill.lFlags, iClr,
    iWidth, iStyle, iLayer)
    xc = 10 : yc = 10
    r = 3 : nVert = 6
    iClr = 7
    Fill.iInForeColor = 1
    AvaxHandle = cAvax1.Add_ShapeRegularPolygon(xc, yc, r, nVert, vItemProperties, Fill.iType,
    Fill.lHatchIndex, Fill.iInForeColor, Fill.iInBackColor, Fill.sPattern, Fill.zOrder, Fill.lFlags, iClr,
    iWidth, iStyle, iLayer)
End Sub
```

Add_ShapeText

cAvax method

Purpose

This function adds a text. The difference from add_Text is that this text has z order

Text and document frames

Text, RTF and document fills follow the shape's stored frame, including rotation, shear and reflection. Changing a document fill reflows the content while retaining the existing frame axes and corner elevations. Changing only colors, flags or drawing order preserves the frame. Content placed outside a document's nominal margins remains included in its drawing and selection bounds.

Native AVAX files retain Double frame coordinates. AVX and XML exports preserve the R2 corner convention; reopening the drawing retains the editable frame and formatted content.

VB.NET declarations

```
Function Add_ShapeText(ByRef sText As String, ByRef x As Double, ByRef y As Double, ByRef z As Double,
    ByRef vItemProperties As Object, ByRef sHeight As Double, ByRef iFontNumber As Integer, ByRef sAngle As
    Double, ByRef iClr As Integer, ByRef iWidth As Integer, ByRef iStyle As Integer, ByRef iLayer As Integer,
    ByRef fBold As Boolean, ByRef fItalic As Boolean, ByRef fUnderline As Boolean, ByRef fStrikeOut As
    Boolean, ByRef lCharSet As Integer, Optional ByVal zOrder As Integer = 0, Optional ByVal fBorder As
    Boolean = False) As Integer

Function Add_ShapeText(ByRef sText As String, ByRef x As Double, ByRef y As Double, ByRef Height As
    Double, ByRef vItemProperties As Object, ByRef sAngle As Double, ByRef iFontNumber As Integer, ByRef iClr
    As Integer, ByRef iWidth As Integer, ByRef iStyle As Integer, ByRef iLayer As Integer, ByRef fBold As
    Boolean, ByRef fItalic As Boolean, ByRef fUnderline As Boolean, ByRef fStrikeOut As Boolean, ByRef
    lCharSet As Integer, ByVal zOrder As Integer, ByVal fBorder As Boolean) As Integer

Function Add_ShapeText(ByRef sText As String, ByRef x As Double, ByRef y As Double, ByRef Height As
    Double) As Integer
```

Parameters

Name	Type and passing	Meaning
sText	ByRef sText As String	Text source. The current four-argument shortcut uses the current font and pen attributes.
x	ByRef x As Double	Double insertion X coordinate.
y	ByRef y As Double	Double insertion Y coordinate.
z	ByRef z As Double	Double elevation in the later overload; the four/eighteen-argument CHM forms insert at Z=0.
vItemProperties	ByRef vItemProperties As Object	Is a string array with properties for the new object
sHeight	ByRef sHeight As Double	Positive finite Double font height in the later overload that accepts Z.
iFontNumber	ByRef iFontNumber As Integer	Integer table index1..256. Nonpositive values or values above256 resolve to1. Positive missing entries inherit the current font family/charset. A ByRef argument receives the normalized index after successful creation.
sAngle	ByRef sAngle As Double	Double angle in degrees. Position differs between the CHM and later overload; use the displayed signature.
iClr	ByRef iClr As Integer	Is the color index number
iWidth	ByRef iWidth As Integer	Is the pen width
iStyle	ByRef iStyle As Integer	Is the line type
iLayer	ByRef iLayer As Integer	Is the layer number
fBold	ByRef fBold As Boolean	A Boolean expression that if it is True then the text is bold
fItalic	ByRef fItalic As Boolean	A Boolean expression that if it is True then the text is italic
fUnderline	ByRef fUnderline As Boolean	A Boolean expression that if it is True then the text is underlined
fStrikeOut	ByRef fStrikeOut As Boolean	A Boolean expression that if it is True then the text is strikethrough
lCharSet	ByRef lCharSet As Integer	Resolved charset of the selected font-table entry, returned ByRef on successful full-overload creation. The named shape retains this value independently of later table edits.
zOrder	Optional ByVal zOrder As Integer = 0	Is the object's elevation in the whole drawing
fBorder	Optional ByVal fBorder As Boolean = False	True draws all four stored frame edges using iClr, iWidth and iStyle. False (the default) suppresses the frame. The corresponding fill flag is BorderOff (4096); clearing it shows the frame.
Height	ByRef Height As Double	Positive finite Double font height in the original CHM overloads, which have no Z parameter.
zOrder	ByVal zOrder As Integer	Is the object's elevation in the whole drawing
fBorder	ByVal fBorder As Boolean	True draws all four stored frame edges using iClr, iWidth and iStyle. False (the default) suppresses the frame. The corresponding fill flag is BorderOff (4096); clearing it shows the frame.

Use the declaration for the overload you call. ByRef arguments can receive outputs; declare variables and arrays with the exact indicated type. Optional defaults are shown in the declaration.

Return value

Integer - An Avax handle on success, zero on error.

Remarks

Three overloads are supported: the four-argument CHM shortcut (text, X, Y, Height), the eighteen-argument CHM form without Z, and the later nineteen-argument form with Z. In the original long form, argument 4 is Height and argument 6 is sAngle; in the later form, argument 4 is Z, argument 6 is sHeight and argument 8 is sAngle. Use the displayed signature rather than mechanically inserting an argument. Coordinates/heights/angles use Double; font/pen/charset indices use Integer. The shortcut uses current pen/font attributes and Z=0. Adds validate finite geometry and positive height before changing the drawing, copy metadata, resolve pen sentinels, and observe the Started/Enabled lifecycle. Full overloads normalize font indices to1..256 and resolve the charset from the font table, returning it ByRef as R2 did. Named text shapes then retain that family/charset independently of later table changes. The four-argument CHM/helper form retains its documented v3 current-pen behavior; it is not the fixed-default ordinary Add_Text shortcut. With fBorder=True, the four stored frame edges are drawn using the shape color, width and line style. The frame follows rotated, sheared, reflected or individually moved corners. fBorder=False (the default) hides the edges. ChangeShapeFill can subsequently toggle the BorderOff flag (4096) without changing the text or frame. Bitmap/GDI output retains the complete border stroke outside the text raster bounds. Ordinary text is fitted to the stored frame. Rotation, shear and reflection of that frame also transform the glyphs. Multiline text divides the frame height into equal line slots; bold, italic, underline and strikeout are applied before fitting. FastText shows the frame in place of the glyphs; NoText hides the text. Imported R2 text frames retain their original corner order. When editing geometry returned by GetProperties, transform the returned corners in place rather than rebuilding or reordering the box. The full overloads retain signed Integer zOrder values and use them when composing overlapping shapes. Changing text does not change this drawing order.

Numeric changes from v2

Update the following variables, arrays or event-data receivers when porting the corresponding v2 call. The declarations above show the current overloads.

Parameter or value	v2 type	v3 type
x	Single	Double
y	Single	Double
z	Single	Double
sHeight	Single	Double
iFontNumber	Short	Integer
sAngle	Single	Double
iClr	Short	Integer
iWidth	Short	Integer
iStyle	Short	Integer
iLayer	Short	Integer
Height	Single	Double

For ByRef parameters, use a variable with the exact v3 type. Preserve array dimensions and the documented indexing. Moving from v2 to v3 explains conversion and legacy file-storage rules.

Example

Setup

Add an Avax.cAvax Windows Forms control named `cAvax1`. Paste the procedures into the form class. Example setup and supporting files explains references, initialization and file paths.

Add these Windows Forms controls or components to the form. Fields referenced by Handles clauses must be declared WithEvents, as the Visual Studio designer normally does.

Name	Type
Command1	Button

VB.NET example

```
Private Sub Form1_Load(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs) Handles
```

```
    MyBase.Load
    cAvax1.StartAvax()
End Sub
Private Sub Form1_FormClosed(ByVal eventSender As System.Object, ByVal eventArgs As
    System.Windows.Forms.FormClosedEventArgs) Handles Me.FormClosed
    cAvax1.EndAvax()
End Sub
Private Sub Command1_Click(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs)
    Handles Command1.Click
    Dim sText As String = String.Empty
    Dim x As Double
    Dim y As Double
    'UPGRADE_NOTE: Height was upgraded to Height_Renamed. Click for more:
    'ms-help://MS.VSCC.v80/dv_commoner/local/redirect.htm?keyword="A9F4979A-37FA-4718-9994-97DD76ED70A7"'
    Dim Height_Renamed As Double
    Dim vItemProperties As Object = Nothing
    Dim sAngle As Double
    Dim iFontNumber As Integer
    Dim iClr As Integer
    Dim iWidth As Integer
    Dim iStyle As Integer
    Dim iLayer As Integer
    Dim fBold As Boolean
    Dim fItalic As Boolean
    Dim fUnderline As Boolean
    Dim fStrikeOut As Boolean
    Dim lCharSet As Integer
    Dim zOrder As Integer
    Dim fBorder As Boolean
    Dim Answer As Integer
    sText = "Sample Text"
    x = 0
    y = 0
    Height_Renamed = 6
    fItalic = True
    fBold = True
    iClr = 1
    Answer = cAvax1.Add_ShapeText(sText, x, y, Height_Renamed, vItemProperties, sAngle, iFontNumber, iClr,
     iWidth, iStyle, iLayer, fBold, fItalic, fUnderline, fStrikeOut, lCharSet, zOrder, fBorder)
End Sub
```

Add_SHP

cAvax method

Purpose

This function inserts a .SHP file into the drawing

VB.NET declarations

```
Function Add_SHP(ByRef sSHPFile As String, ByRef x As Double, ByRef y As Double, ByRef z As Double, ByRef
    vItemProperties As Object, ByRef Fill_iType As Byte, ByRef Fill_lHatchIndex As Integer, ByRef
    Fill_iInForeColor As Integer, ByRef Fill_iInBackColor As Integer, ByRef Fill_sPattern As String, ByRef
    Fill_zOrder As Integer, ByRef Fill_lFlags As Integer, ByRef iClr As Integer, ByRef iWidth As Integer,
    ByRef iStyle As Integer, ByRef iLayer As Integer, ByRef vRetHandles As Object) As Boolean

Function Add_SHP(ByRef sSHPFile As String, ByRef x As Double, ByRef y As Double, ByRef z As Double) As
    Boolean
```

Parameters

Name	Type and passing	Meaning
sSHPFile	ByRef sSHPFile As String	Existing ESRI SHP file path including its extension. Relative paths use the working directory. The filename is preserved.
x	ByRef x As Double	Double destination X coordinate for the lower-left bound recorded in the SHP header.
y	ByRef y As Double	Double destination Y coordinate for the lower-left bound recorded in the SHP header.
z	ByRef z As Double	Double elevation for plain/M geometry, or an offset added to source Z coordinates.
vItemProperties	ByRef vItemProperties As Object	Nothing or a String() of application properties copied to every inserted object. DBF fields are not imported automatically.
Fill_iType	ByRef Fill_iType As Byte	AvaxFillType value passed as Byte for polygon parts. The Integer overload uses Solid_f (0).
Fill_lHatchIndex	ByRef Fill_lHatchIndex As Integer	Integer pattern index stored with polygon fills.
Fill_iInForeColor	ByRef Fill_iInForeColor As Integer	Integer foreground color of the polygon fill.
Fill_iInBackColor	ByRef Fill_iInBackColor As Integer	Integer background color of the polygon fill.
Fill_sPattern	ByRef Fill_sPattern As String	Pattern definition or image filename for the selected polygon fill type.
Fill_zOrder	ByRef Fill_zOrder As Integer	Integer drawing order for polygon shapes.
Fill_lFlags	ByRef Fill_lFlags As Integer	Integer flags controlling the polygon fill and border.
iClr	ByRef iClr As Integer	Integer pen color. -1 uses PenColor and receives the resolved value on success.
iWidth	ByRef iWidth As Integer	Integer pen width. -1 uses PenWidth; zero becomes 1. Receives the resolved value on success.
iStyle	ByRef iStyle As Integer	Integer style applied to created objects. -1 uses PenLineStyle and receives the resolved value on success.
iLayer	ByRef iLayer As Integer	Integer destination layer. -1 uses PenLayer and receives the resolved value on success.
vRetHandles	ByRef vRetHandles As Object	Output Object containing Integer() handles for the new objects. Index zero is reserved; actual handles start at index one. Preserved when no import commits.

Use the declaration for the overload you call. ByRef arguments can receive outputs; declare variables and arrays with the exact indicated type. Optional defaults are shown in the declaration.

Return value

Boolean - True on success, false on error.

Remarks

Inserts ESRI shapefile geometry as editable drawing objects. Call StartAvax first and keep the control enabled. Supply an existing SHP path, including its extension; relative paths use the working directory. The source file and filename are preserved. SHX and DBF companion files are not required. DBF fields are not automatically converted to object properties. The file header's lower-left XY bound is placed at (x, y), as in R2. Each source coordinate is converted using (coordinate - header minimum) + insertion coordinate. Subtracting first retains small offsets in survey drawings with large coordinates. Plain and M-type shapes use the supplied z elevation. Z-type shapes retain each source Z value plus z; M measures are ignored. Coordinates use Double throughout. Point and multipoint records create point objects. Each polyline or polygon part creates a separate object, retaining vertex order and polygon closing vertices. Null records create no objects. Point, PolyLine, Polygon and MultiPoint are supported, including their Z and M variants. MultiPatch is not supported. Multipart polygon rings currently remain separate shapes; filled holes are not combined into a single compound region. The short overload uses the current pen color, width, line style and layer, with a solid fill whose remaining fields are zero or empty. It does not inherit the active fill pattern or flags. The full overload accepts the supplied fill fields and copies vItemProperties to every created object. Pen arguments equal to -1 resolve to the corresponding current pen value and are returned through their ByRef parameters on success. Width zero becomes 1. Integer attributes are retained without Short conversion. Bitmap fills are embedded so saved drawings can reopen without the source image. Pattern and image resources are prepared before drawing changes. True indicates that at least one object was inserted. The full overload returns an Integer() through vRetHandles; index zero is reserved and the new handles occupy indices one through the upper bound. Empty or null-only files return False. Missing, unsupported or malformed files, invalid geometry/properties and insufficient handle capacity return False without partial insertion or changes to caller outputs, selection or prior Undo/Redo. A successful import is one Undo step. Exceptions from application completion handlers propagate after the import commits. The original example imports abarb_bl.shp from the parent of the application directory and fits the drawing to the view. This programmatic operation publishes Change without NewUserItems, as in R2. Use its documented return value/output parameters for the result. Interactive drawing commands and explicit announcing shape/picture/document helpers retain their own notification policy. Header-origin placement and insertion offsets use the complete Double result. Small coordinates are retained when large origins and offsets cancel; a finite final coordinate is accepted even if a separate intermediate subtraction would overflow. A final coordinate outside the finite Double range is rejected before any part of the import is committed, preserving the existing drawing and caller outputs.

Numeric changes from v2

Update the following variables, arrays or event-data receivers when porting the corresponding v2 call. The declarations above show the current overloads.

Parameter or value	v2 type	v3 type
x	Single	Double
y	Single	Double
z	Single	Double
Fill_iInForeColor	Short	Integer
Fill_iInBackColor	Short	Integer
iClr	Short	Integer
iWidth	Short	Integer
iStyle	Short	Integer
iLayer	Short	Integer

For ByRef parameters, use a variable with the exact v3 type. Preserve array dimensions and the documented indexing. Moving from v2 to v3 explains conversion and legacy file-storage rules.

Example

Setup

Add an Avax.cAvax Windows Forms control named `cAvax1`. Paste the procedures into the form class. Example setup and supporting files explains references, initialization and file paths.

Add these Windows Forms controls or components to the form. Fields referenced by Handles clauses must be declared WithEvents, as the Visual Studio designer normally does.

Name	Type
Command1	Button

VB.NET example

```
Private Sub Form1_Load(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs) Handles
    MyBase.Load
    cAvax1.StartAvax()
End Sub
Private Sub Form1_FormClosed(ByVal eventSender As System.Object, ByVal eventArgs As
    System.Windows.Forms.FormClosedEventArgs) Handles Me.FormClosed
    cAvax1.EndAvax()
End Sub
Private Sub Command1_Click(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs)
    Handles Command1.Click
    Dim sSHPFile As String = String.Empty
    Dim x As Double
    Dim y As Double
    Dim z As Double
    Dim Answer As Boolean
    sSHPFile = My.Application.Info.DirectoryPath & "\..\abarb_bl.shp"
    sSHPFile = IO.Path.GetFullPath(sSHPFile)
    Answer = cAvax1.Add_SHP(sSHPFile, x, y, z)
    CAvax1.Command = Avax.AvaxCommand.AutoLimits_c
End Sub
```

Add_SVG

cAvax method

Purpose

Imports an SVG file at a supplied XYZ insertion point. The full overload returns the imported object handles.

VB.NET declarations

```
Function Add_SVG(ByRef sSVGFile As String, ByRef x As Double, ByRef y As Double, ByRef z As Double, ByRef
    vItemProperties As Object, ByRef vRetHandles As Object) As Boolean

Function Add_SVG(ByRef sSVGFile As String, ByRef x As Double, ByRef y As Double, ByRef z As Double) As
    Boolean
```

Parameters

Name	Type and passing	Meaning
sSVGFile	ByRef sSVGFile As String	Is the name of the SVG file
x	ByRef x As Double	Is the x coordinate of the insertion point
y	ByRef y As Double	Is the y coordinate of the insertion point
z	ByRef z As Double	Is the z coordinate of the insertion point
vItemProperties	ByRef vItemProperties As Object	Is a string array with properties for the new object
vRetHandles	ByRef vRetHandles As Object	Returns an integer array with the handles of the new objects

Use the declaration for the overload you call. ByRef arguments can receive outputs; declare variables and arrays with the exact indicated type. Optional defaults are shown in the declaration.

Return value

Boolean - True on success, false on error.

Remarks

The full overload accepts copied application metadata and returns a one-based Integer() in vRetHandles. The insertion coordinates and imported geometry use Double drawing units. Path commands M/L/H/V/C/S/Q/T/A/Z accept upper- and lower-case forms, repeated arguments, compact numeric tokens and multiple subpaths. Relative path coordinates follow each segment start; closing a subpath returns to its starting point.

Fill-only text remains editable Text_i. Text with a visible stroke is stored as compound Shape_i glyph contours, with one object for each paint: fill first, then stroke. Glyph counters and holes are retained. Stroke width, dash pattern/offset, caps, joins and miter limit are applied in the SVG source plane before the complete affine transform, including rotation, skew, reflection and nonuniform scaling. A zero stroke width produces no stroke. Font family, bold/italic face, text-chunk anchoring and tspan advances are retained.

Call StartAvax first and keep the control enabled. The complete entity and resource batch is prepared before insertion. Missing, malformed or unrepresentable input returns False without a partial import or Undo entry. Success is one Undo operation and raises Change without NewUserItems. Application event-handler exceptions propagate after committed insertion. Unsupported or unrepresentable text strokes fail during preparation. SVG paint servers, opacity effects, clipping, masks, filters and per-tspan transforms/rotation are not supported. Skewed fill-only text is outside the editable-text path and is rejected.

Numeric changes from v2

Update the following variables, arrays or event-data receivers when porting the corresponding v2 call. The declarations above show the current overloads.

Parameter or value	v2 type	v3 type
x	Single	Double

Parameter or value	v2 type	v3 type
y	Single	Double
z	Single	Double

For ByRef parameters, use a variable with the exact v3 type. Preserve array dimensions and the documented indexing. Moving from v2 to v3 explains conversion and legacy file-storage rules.

Example

Setup

Add an Avax.cAvax Windows Forms control named `cAvax1`. Paste the procedures into the form class. Example setup and supporting files explains references, initialization and file paths.

Add these Windows Forms controls or components to the form. Fields referenced by Handles clauses must be declared WithEvents, as the Visual Studio designer normally does.

Name	Type
Command1	Button

Use the supplied `Rectangles.svg` from `AVAX Vector .NET v.3 Examples/HelpExamples`. Copy it to `Data/HelpExamples/Rectangles.svg` below your application executable. In Visual Studio, keep that relative folder and set the file to Content with Copy if newer. The supplied AddSVG project already copies the file to this location. To import your own SVG, replace `sSVGFile` with its full path.

VB.NET example

```
Private Sub Form1_Load(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs) Handles
    MyBase.Load
    cAvax1.StartAvax()
End Sub
Private Sub Form1_FormClosed(ByVal eventSender As System.Object, ByVal eventArgs As
    System.Windows.Forms.FormClosedEventArgs) Handles Me.FormClosed
    cAvax1.EndAvax()
End Sub
Private Sub Command1_Click(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs)
    Handles Command1.Click
    Dim sSVGFile As String = String.Empty
    Dim x As Double
    Dim y As Double
    Dim z As Double
    Dim vItemProperties As Object = Nothing
    Dim vRetHandles As Object = Nothing
    Dim AvaxHandle As Boolean
    sSVGFile = System.IO.Path.Combine(System.AppDomain.CurrentDomain.BaseDirectory, "Data",
    "HelpExamples", "Rectangles.svg")
    AvaxHandle = cAvax1.Add_SVG(sSVGFile, x, y, z, vItemProperties, vRetHandles)
    CAvax1.Command = Avax.AvaxCommand.AutoLimits_c
End Sub
```

Add_Text

cAvax method

Purpose

This function adds a text into the drawing

VB.NET declarations

```
Function Add_Text(ByRef sText As String, ByRef x As Double, ByRef y As Double, ByRef z As Double, ByRef
    vItemProperties As Object, ByRef sScale As Double, ByRef iFontNumber As Integer, ByRef Angle As Double,
    ByRef iClr As Integer, ByRef iWidth As Integer, ByRef iStyle As Integer, ByRef iLayer As Integer, ByRef
    fBold As Boolean, ByRef fItalic As Boolean, ByRef fUnderline As Boolean, ByRef fStrikeOut As Boolean,
    ByRef lCharSet As Integer) As Integer

Function Add_Text(ByRef sText As String, ByRef x As Double, ByRef y As Double, ByRef z As Double) As
    Integer
```

Parameters

Name	Type and passing	Meaning
sText	ByRef sText As String	Is the name of the text (change text line with CrLf)
x	ByRef x As Double	Is the x coordinate of the text's first character bottom left point
y	ByRef y As Double	Is the y coordinate of the text's first character bottom left point
z	ByRef z As Double	Is the z coordinate of the text's first character bottom left point
vItemProperties	ByRef vItemProperties As Object	Is a string array with properties for the new object
sScale	ByRef sScale As Double	Finite Double multiplier of TextInitHeight_p. Nonpositive values become1 and are written back ByRef on successful creation. The Integer overload uses1 rather than PenFontScale.
iFontNumber	ByRef iFontNumber As Integer	Integer table index1..256. Nonpositive values or values above256 resolve to1. Positive missing entries inherit the current font family/charset. A ByRef argument receives the normalized index after successful creation.
Angle	ByRef Angle As Double	Is the angle of the text
iClr	ByRef iClr As Integer	Is the color index number
iWidth	ByRef iWidth As Integer	Is the pen width
iStyle	ByRef iStyle As Integer	Is the line type
iLayer	ByRef iLayer As Integer	Is the layer number
fBold	ByRef fBold As Boolean	A Boolean expression that if it is True then the text is bold
fItalic	ByRef fItalic As Boolean	A Boolean expression that if it is True then the text is italic
fUnderline	ByRef fUnderline As Boolean	A Boolean expression that if it is True then the text is underlined
fStrikeOut	ByRef fStrikeOut As Boolean	A Boolean expression that if it is True then the text is strikethrough
lCharSet	ByRef lCharSet As Integer	The caller argument is retained ByRef. Ordinary indexed text resolves its effective charset from the font table/current FontCharset, as observed through the R2 GetTextAttributes contract.

Use the declaration for the overload you call. ByRef arguments can receive outputs; declare variables and arrays with the exact indicated type. Optional defaults are shown in the declaration.

Return value

Integer - An Avax handle on success, zero on error.

Remarks

Creates ordinary text at the supplied insertion point. The four-argument overload uses scale 1, font 1, angle 0 and all four style flags False, independently of PenFontScale, PenFontNumber and the current style flags. Its color, line width, line style and layer use the current pen settings. In the full overload, sScale is a Double multiplier of TextInitHeight_p; a nonpositive finite scale becomes 1. Explicit style flags are retained. With Windows fonts disabled, the initial vector advance also uses TextInitWidth_p. The effective charset is resolved through the font table; the caller lCharSet argument is unchanged. Font indices 1 through 256 are supported; zero, negative or larger values normalize to 1. A missing or blank table entry inherits the current FontName and FontCharset. Successful ByRef font, scale and pen arguments receive their normalized values. Invalid or nonfinite geometry is rejected before insertion or ByRef changes and reports an error; an unstarted or disabled control returns 0. A successful insertion creates one undo step and raises NewUserItems and Change. Exceptions from application event handlers propagate after the object has been added. Changing PenFontScale later does not resize this text. TextAlignWidth_p applies only to the full overload with Center_a or Right_a as iStyle, a positive field width and PltTextQuality_p other than -1. It moves a single line only when the line is narrower than the field, along the supplied Angle. Multiline text remains at its insertion point; the consumed alignment flag is not stored as a line style. Inheriting a current pen line style with the same numeric value as Center_a or Right_a does not request alignment and does not move the insertion point.

Numeric changes from v2

Update the following variables, arrays or event-data receivers when porting the corresponding v2 call. The declarations above show the current overloads.

Parameter or value	v2 type	v3 type
x	Single	Double
y	Single	Double
z	Single	Double
sScale	Single	Double
iFontNumber	Short	Integer
Angle	Single	Double
iClr	Short	Integer
iWidth	Short	Integer
iStyle	Short	Integer
iLayer	Short	Integer

For ByRef parameters, use a variable with the exact v3 type. Preserve array dimensions and the documented indexing. Moving from v2 to v3 explains conversion and legacy file-storage rules.

Example

Setup

Add an Avax.cAvax Windows Forms control named `cAvax1`. Paste the procedures into the form class. Example setup and supporting files explains references, initialization and file paths.

Add these Windows Forms controls or components to the form. Fields referenced by Handles clauses must be declared WithEvents, as the Visual Studio designer normally does.

Name	Type
Command1	Button

VB.NET example

```
Private Sub Form1_Load(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs) Handles
    MyBase.Load
    cAvax1.StartAvax()
End Sub
Private Sub Form1_FormClosed(ByVal eventSender As System.Object, ByVal eventArgs As
    System.Windows.Forms.FormClosedEventArgs) Handles Me.FormClosed
```

```
    cAvax1.EndAvax()
End Sub
Private Sub Command1_Click(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs)
    Handles Command1.Click
    Dim sText As String = String.Empty
    Dim x As Double
    Dim y As Double
    Dim z As Double
    Dim vItemProperties As Object = Nothing
    Dim sScale As Double
    Dim iFontNumber As Integer
    Dim Angle As Double
    Dim iClr As Integer
    Dim iWidth As Integer
    Dim iStyle As Integer
    Dim iLayer As Integer
    Dim fBold As Boolean
    Dim fItalic As Boolean
    Dim fUnderline As Boolean
    Dim fStrikeOut As Boolean
    Dim lCharSet As Integer
    Dim Answer As Integer
    'Definition of the text
    sText = "Hello !!!"
    'The scale
    sScale = 10
    'Start point of the text
    x = 1 : y = 4 : z = 0
    'Angle of the text (degrees)
    Angle = 30
    iClr = 6
    fBold = True
    fUnderline = True
    Answer = CAvax1.Add_Text(sText, x, y, z, vItemProperties, sScale, iFontNumber, Angle, iClr, iWidth,
    iStyle, iLayer, fBold, fItalic, fUnderline, fStrikeOut, lCharSet)
End Sub
```

Add_TextC

API reference

Creates text with its visible content centered on a specified point and returns the new object handle. Use it for labels whose position should remain centered regardless of their length. The full overload also selects font, scale, rotation, style, layer and custom properties. For text placed at an insertion point instead, use Add_Text.

VB.NET declarations

```
Public Function Add_TextC(ByRef sText As String,
    ByRef xTextCenter As Double, ByRef yTextCenter As Double,
    ByRef zTextCenter As Double) As Integer

Public Function Add_TextC(ByRef sText As String,
    ByRef xTextCenter As Double, ByRef yTextCenter As Double,
    ByRef zTextCenter As Double, ByRef vItemProperties As Object,
    ByRef TextScale As Double, ByRef iFontNumber As Integer,
    ByRef Angle As Double, ByRef iClr As Integer,
    ByRef iWidth As Integer, ByRef iStyle As Integer,
    ByRef iLayer As Integer, ByRef fBold As Boolean,
    ByRef fItalic As Boolean, ByRef fUnderline As Boolean,
    ByRef fStrikeOut As Boolean, ByRef lCharSet As Integer) As Integer
```

Parameters

Parameter	Meaning
sText	Text to insert. Unicode text and multiple lines are supported; formatted RTF is also accepted. The input string is retained.
xTextCenter, yTextCenter, zTextCenter	Requested center in drawing coordinates. These variables are not changed. X and Y locate the center of the visible text; Z supplies its elevation.
vItemProperties	Custom string-array properties, or Nothing. Logical property values start at index 1. The drawing stores its own copy.
TextScale	Multiplier of TextInitHeight_p. A finite value of zero or less is normalized to 1 after successful creation. The short overload uses 1.
iFontNumber	Font-table entry, from 1 through 256. Other values normalize to 1 after success. A blank entry uses FontName and FontCharset. The short overload uses entry 1.
Angle	Rotation in degrees about the supplied center. The stored angle is reduced modulo 360; the caller's variable is unchanged. The short overload uses zero.
iClr, iWidth, iStyle, iLayer	Color, line width, line style and layer indices. Use -1 to select the corresponding current pen value. Successful calls return the resolved values through these variables. The short overload uses the current pen values.
fBold, fItalic, fUnderline, fStrikeOut	Requested font styles. The short overload uses False for all four, independently of the current font-style properties.
lCharSet	Historical charset argument. The effective charset comes from the selected font-table entry; this caller variable is not changed.

Return value

Returns a positive Integer handle on success, or zero if creation fails. Call StartAvax before inserting text and keep the control enabled. Invalid geometry or resources are rejected before changing the drawing or ByRef outputs. Successful insertion creates one Undo step and raises NewUserItems and Change; exceptions raised by application event handlers propagate after the insertion has committed.

Remarks

The control centers the unrotated visible content in both X and Y, then rotates it about the requested point. Glyph overhangs, descenders, leading spaces, multiple lines and text styles are included. TextAlignWidth_p and pen alignment do not move this explicitly supplied center. Empty text, or spaces with no visible decoration, retain the requested insertion anchor.

The current OriginTopLeft_p setting is respected when creating the item. Later font-table, view or display-property changes do not recenter its stored geometry. Fixed/minimum display sizes, text-excess backgrounds and selection handles do not define the text center. OPT vector fonts are measured from their strokes. RTF uses occupied RichEdit page content, including highlights and images; its bounds are sampled at four samples per layout pixel, capped at 4096 pixels per page axis.

In v3, coordinates, scale and angle use Double in place of Single; font and pen indices use Integer in place of Short. Handles remain Integer. Supply variables of the declared types for ByRef arguments.

VB.NET example

Pass a started control. This creates a bold label centered at (100.25, 50.5), rotated by 30 degrees, using font-table entry 1 and the current pen attributes.

```
Option Strict On
Imports System

Public Module CenteredTextExample
    Public Function AddLabel(cad As Avax.cAvax) As Integer
        If cad Is Nothing Then Throw New ArgumentNullException(NameOf(cad))
        Dim label As String = "Room 101"
        Dim x As Double = 100.25, y As Double = 50.5, z As Double = 0.0
        Dim properties As Object = Nothing
        Dim scale As Double = 2.0, angle As Double = 30.0
        Dim fontNumber As Integer = 1
        Dim ink As Integer = -1, width As Integer = -1
        Dim style As Integer = -1, layer As Integer = -1
        Dim bold As Boolean = True, italic As Boolean = False
        Dim underline As Boolean = False, strikeOut As Boolean = False
        Dim charset As Integer = 0
        Dim handle As Integer = cad.Add_TextC(label, x, y, z, properties,
            scale, fontNumber, angle, ink, width, style, layer,
            bold, italic, underline, strikeOut, charset)
        If handle = 0 Then Throw New InvalidOperationException("Text could not be inserted.")
        Return handle
    End Function
End Module
```

Add_TextFile

cAvax method

Purpose

This function adds a text File into the drawing

VB.NET declaration

```
Function Add_TextFile(ByRef sTextFile As String, ByRef x As Double, ByRef y As Double, ByRef z As Double,
    ByRef sScale As Double, ByRef TStyle As Integer, ByRef sAngle As Double, ByRef vItemProperties As Object
    ByRef iClr As Integer, ByRef iWidth As Integer, ByRef iStyle As Integer, ByRef iLayer As Integer) As
    Integer
```

Parameters

Name	Type and passing	Meaning
sTextFile	ByRef sTextFile As String	Existing plain-text file path including the extension. Relative paths use the working directory; the filename is preserved.
x	ByRef x As Double	Double X coordinate of the insertion point.
y	ByRef y As Double	Double Y coordinate of the insertion point.
z	ByRef z As Double	Double elevation of the text.
sScale	ByRef sScale As Double	Multiplier of TextInitHeight_p. A nonpositive finite value becomes 1 and is returned on success.
TStyle	ByRef TStyle As Integer	Integer font-table index. Invalid indices resolve to entry 1 and are returned on success.
sAngle	ByRef sAngle As Double	Counterclockwise text rotation in degrees about the insertion point.
vItemProperties	ByRef vItemProperties As Object	Nothing or a String() of application properties copied to the text object.
iClr	ByRef iClr As Integer	Integer palette index or direct color; -1 uses the current PenColor and receives its value on success.
iWidth	ByRef iWidth As Integer	Integer line width; -1 uses PenWidth, while zero becomes 1. Receives the resolved width on success.
iStyle	ByRef iStyle As Integer	Integer line-style index; -1 uses PenLineStyle and receives its value on success.
iLayer	ByRef iLayer As Integer	Integer layer index; -1 uses PenLayer and receives its value on success.

Use the declaration for the overload you call. ByRef arguments can receive outputs; declare variables and arrays with the exact indicated type. Optional defaults are shown in the declaration.

Return value

Integer - An Avax handle on success, zero on error.

Remarks

Reads a plain-text file and inserts its contents as one text object. Call StartAvax first and keep the control enabled. Supply an existing path with its extension; relative paths use the working directory. The path and source bytes remain unchanged. The return value is the new Integer handle, or zero when no object is inserted. The insertion coordinates, scale and rotation are Double. The text height is sScale multiplied by TextInitHeight_p. A nonpositive finite scale becomes 1 and is returned through sScale. TStyle is the Integer font-table index; invalid indices resolve to entry 1 and are returned through TStyle. The font and charset come from that table entry. Imported text uses plain styling: bold, italic, underline and strikeout are False, independently of the current font flags. The current pen and font settings are preserved. Color, width, line style and layer -1 values resolve to the current pen values and are returned through their ByRef arguments; zero width becomes 1. vItemProperties accepts Nothing or a String() copied to the inserted object. UTF-8, UTF-16 and UTF-32 files with a byte-order mark are decoded as Unicode. Without a byte-order mark, valid UTF-8 is accepted; otherwise the current Windows ANSI code page is used, as with legacy text files. Use a byte-order mark for unambiguous Unicode encoding. Each input line is stored with a carriage-return terminator, matching the R2 text-file convention. Empty or space-only lines become a single space. The complete file is imported, including text beyond 30000 characters. An empty file, unreadable file, invalid geometry or unsupported property value returns zero. Preparation occurs before drawing edits, so rejected input preserves selection, Undo/Redo and caller outputs. Success adds one Undo step. Exceptions from application completion handlers propagate after the text has been inserted. Native .avax storage retains the text and Double coordinates. The original example reads AvaxLineStyle.txt from the parent of the application directory and inserts it at (4,10,0), scale 5 and angle 45 degrees. This programmatic operation publishes Change without NewUserItems, as in R2. Use its documented return value/output parameters for the result. Interactive drawing commands and explicit announcing shape/picture/document helpers retain their own notification policy.

Numeric changes from v2

Update the following variables, arrays or event-data receivers when porting the corresponding v2 call. The declarations above show the current overloads.

Parameter or value	v2 type	v3 type
x	Single	Double
y	Single	Double
z	Single	Double
sScale	Single	Double
TStyle	Short	Integer
sAngle	Single	Double
iClr	Short	Integer
iWidth	Short	Integer
iStyle	Short	Integer
iLayer	Short	Integer

For ByRef parameters, use a variable with the exact v3 type. Preserve array dimensions and the documented indexing. Moving from v2 to v3 explains conversion and legacy file-storage rules.

Example

Setup

Add an Avax.cAvax Windows Forms control named `cAvax1`. Paste the procedures into the form class. Example setup and supporting files explains references, initialization and file paths.

Add these Windows Forms controls or components to the form. Fields referenced by Handles clauses must be declared WithEvents, as the Visual Studio designer normally does.

Name	Type
Command1	Button

VB.NET example

```
Private Sub Form1_Load(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs) Handles
    MyBase.Load
    cAvax1.StartAvax()
End Sub
Private Sub Form1_FormClosed(ByVal eventSender As System.Object, ByVal eventArgs As
    System.Windows.Forms.FormClosedEventArgs) Handles Me.FormClosed
    cAvax1.EndAvax()
End Sub
Private Sub Command1_Click(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs)
    Handles Command1.Click
    Dim sTextFile As String = String.Empty
    Dim x As Double
    Dim y As Double
    Dim z As Double
    Dim sScale As Double
    Dim TStyle As Integer
    Dim sAngle As Double
    Dim vItemProperties As Object = Nothing
    Dim iClr As Integer
    Dim iWidth As Integer
    Dim iStyle As Integer
    Dim iLayer As Integer
    Dim AvaxHandle As Integer
    ' Name of text file
    sTextFile = My.Application.Info.DirectoryPath & "\..\AvaxLineStyle.txt"
    ' Start points of the text
    x = 4 : y = 10 : z = 0
    'Scale
    sScale = 5
    'Text Rotation
    sAngle = 45
    AvaxHandle = cAvax1.Add_TextFile(sTextFile, x, y, z, sScale, TStyle, sAngle, vItemProperties, iClr,
    iWidth, iStyle, iLayer)
    CAvax1.Command = Avax.AvaxCommand.AutoLimits_c
End Sub
```

Add_WMF

cAvax method

Purpose

This function inserts windows metafile(s) .WMF into the drawing

VB.NET declarations

```
Function Add_WMF(ByRef sWinMetaFiles As String(), ByVal xMin As Double, ByVal yMin As Double, ByVal xMax
    As Double, ByVal yMax As Double, ByRef zOrder As Integer, ByRef vItemProperties As Object, ByRef iLayer As
     Integer, ByRef nMapMode As Integer) As Integer

Function Add_WMF(ByRef sWinMetaFiles As String(), ByVal xMin As Double, ByVal yMin As Double, ByVal xMax
    As Double, ByVal yMax As Double) As Integer
```

Parameters

Name	Type and passing	Meaning
sWinMetaFiles	ByRef sWinMetaFiles As String()	One-based array of WMF, EMF or metafile AAF filenames; index 0 is unused.
xMin	ByVal xMin As Double	First X corner of the insertion rectangle, in Double drawing coordinates.
yMin	ByVal yMin As Double	First Y corner of the insertion rectangle, in Double drawing coordinates.
xMax	ByVal xMax As Double	Opposite X corner of the insertion rectangle, in Double drawing coordinates.
yMax	ByVal yMax As Double	Opposite Y corner of the insertion rectangle, in Double drawing coordinates.
zOrder	ByRef zOrder As Integer	Drawing order of the embedded picture or video.
vItemProperties	ByRef vItemProperties As Object	Application-defined item properties, using the legacy one-based String array convention when supplied.
iLayer	ByRef iLayer As Integer	Destination layer. A current-layer sentinel is resolved to the active pen layer and written back by the extended overload.
nMapMode	ByRef nMapMode As Integer	Windows map mode, from 1 through 8. The Integer overload uses 8 (MM_ANISOTROPIC).

Use the declaration for the overload you call. ByRef arguments can receive outputs; declare variables and arrays with the exact indicated type. Optional defaults are shown in the declaration.

Return value

Integer - An Avax handle on success, zero on error.

Remarks

Embeds Windows metafile graphics from one or more files and returns the drawing handle. The filename array uses indices 1 through N, leaving index 0 unused. Every frame must contain valid WMF or EMF metafile data, including frames supplied through AAF. The rectangle uses Double drawing coordinates. The extended overload accepts z-order, item properties, layer and map mode. Invalid data returns 0 without inserting an incomplete object.

Numeric changes from v2

Update the following variables, arrays or event-data receivers when porting the corresponding v2 call. The declarations above show the current overloads.

Parameter or value	v2 type	v3 type
xMin	Single	Double

Parameter or value	v2 type	v3 type
yMin	Single	Double
xMax	Single	Double
yMax	Single	Double
iLayer	Short	Integer

For ByRef parameters, use a variable with the exact v3 type. Preserve array dimensions and the documented indexing. Moving from v2 to v3 explains conversion and legacy file-storage rules.

Example

Setup

Add an Avax.cAvax Windows Forms control named `cAvax1`. Paste the procedures into the form class. Example setup and supporting files explains references, initialization and file paths.

Add these Windows Forms controls or components to the form. Fields referenced by Handles clauses must be declared WithEvents, as the Visual Studio designer normally does.

Name	Type
Command1	Button

VB.NET example

```
Private Sub Form1_Load(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs) Handles
    MyBase.Load
    cAvax1.StartAvax()
End Sub
Private Sub Form1_FormClosed(ByVal eventSender As System.Object, ByVal eventArgs As
    System.Windows.Forms.FormClosedEventArgs) Handles Me.FormClosed
    cAvax1.EndAvax()
End Sub
Private Sub Command1_Click(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs)
    Handles Command1.Click
    Dim sWinMetaFiles() As String = Nothing
    Dim xMin As Double
    Dim yMin As Double
    Dim xMax As Double
    Dim yMax As Double
    Dim AvaxHandle As Integer
    ReDim sWinMetaFiles(1)
    sWinMetaFiles(1) = My.Application.Info.DirectoryPath & "\..\Door.wmf"
    xMin = 0
    yMin = 0
    xMax = 8
    yMax = 12
    AvaxHandle = cAvax1.Add_WMF(sWinMetaFiles, xMin, yMin, xMax, yMax)
End Sub
```

Add_XML

cAvax method

Purpose

This function inserts a .XML file that contains a drawing and it was exported by another AVAX by the PrintToXML method

VB.NET declarations

```
Function Add_XML(ByRef sXMLFile As String, ByRef x As Double, ByRef y As Double, ByRef z As Double, ByRef
    vRetHandles As Object) As Boolean

Function Add_XML(ByRef sXMLFile As String, ByRef x As Double, ByRef y As Double, ByRef z As Double) As
    Boolean
```

Parameters

Name	Type and passing	Meaning
sXMLFile	ByRef sXMLFile As String	Local XML drawing filename. Relative paths use the working directory. The file is read without modification; the ByRef filename is not changed.
x	ByRef x As Double	Destination X coordinate of the XML Drawings xMin origin, in drawing units (Double). Earlier AvaxDrawing version="3" files use an origin of zero.
y	ByRef y As Double	Destination Y coordinate of the XML Drawings yMin origin, in drawing units (Double).
z	ByRef z As Double	Destination Z coordinate of the XML Drawings zMin origin, in drawing units (Double). Applied to the imported object coordinates, including block children.
vRetHandles	ByRef vRetHandles As Object	On success, receives a one-based Integer array of new root-object handles; index zero is reserved. Remains unchanged when no objects are inserted or the import fails.

Use the declaration for the overload you call. ByRef arguments can receive outputs; declare variables and arrays with the exact indicated type. Optional defaults are shown in the declaration.

Return value

Boolean - True on success, false on error.

Remarks

Inserts all root objects from an AVAX XML drawing into the current drawing. Call StartAvax first and keep the control enabled. The filename is a local filesystem path; relative paths are resolved from the application working directory. The source file is not modified. Both overloads return True only when at least one object is inserted. R2 XML uses an AVAX root with General Version="1" and a MainDrawing definition. The Drawings xMin/yMin/zMin values define the source insertion origin: each source point is translated by subtracting that origin and then adding x/y/z. Coordinates and insertion offsets use Double. For text and nested blocks, the XML Transform is applied before insertion. Lines, points, polylines, analytic arcs/ellipses, text, shapes, documents and nested blocks retain their available attributes and properties. Referenced drawing definitions are resolved case-insensitively; missing or recursive definitions are rejected. Used palette/layer entries and font/line-style tables are merged into the destination, with imported object indices remapped to preserve their appearance and the existing drawing. The imported LineStyleScale becomes the current line-style scale after success. Embedded image data and document resources are prepared before the drawing is changed. XML numbers are parsed independently of the Windows decimal separator. Integer fields such as widths, layers and colors are not narrowed to Short. Files written by PrintToXML retain readable R2 XML together with an optional R3Native supplement for v3-specific data. The supplement is used only when its checksum matches the readable XML; editing the readable XML makes that content authoritative. Earlier v3 AvaxDrawing version="3" files containing a native drawing are also accepted; these retain their original origin-zero insertion behavior. Unknown XML schemas/versions and unsupported native file versions are rejected. The longer overload returns an Integer array in vRetHandles, with index zero reserved and new root-object handles starting at index one. Input filename and insertion coordinates are unchanged. Missing, invalid or unsupported files, nonfinite coordinates, invalid resources and insufficient handle capacity return False before changing the drawing or returned handles. An empty drawing returns False. An unstarted, disabled, stopped or disposed control also returns False. XML DTDs and external entity expansion are disabled. A successful insertion is one undoable batch and raises the drawing-change/new-items notifications. Undo and Redo restore the imported objects, resource tables and line-style scale together. Exceptions raised by application event handlers after the import has committed propagate to the application; they do not undo that committed import. The VB.NET example loads drawing.xml from the parent of the executable directory. Deploy that sample in the shown location or supply the full path to your drawing. Creation notifications follow the original XML helper types: each shape/document root is announced separately in source order; ordinary primitive and block roots are not. The complete import and one reversible record are committed before any callback. One aggregate Change is published before the first ordinary-document notice, or after the notices when no such document is present. Clearing/replacing the drawing, suppression or lifetime changes stop the superseded sequence. Removing a pending root skips its notice.

Numeric changes from v2

Update the following variables, arrays or event-data receivers when porting the corresponding v2 call. The declarations above show the current overloads.

Parameter or value	v2 type	v3 type
x	Single	Double
y	Single	Double
z	Single	Double

For ByRef parameters, use a variable with the exact v3 type. Preserve array dimensions and the documented indexing. Moving from v2 to v3 explains conversion and legacy file-storage rules.

Example

Setup

Add an Avax.cAvax Windows Forms control named `cAvax1`. Paste the procedures into the form class. Example setup and supporting files explains references, initialization and file paths.

Add these Windows Forms controls or components to the form. Fields referenced by Handles clauses must be declared WithEvents, as the Visual Studio designer normally does.

Name	Type
Button1	Button

VB.NET example

```
Private Sub Form1_Load(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs) Handles
    MyBase.Load
    Call cAvax1.StartAvax()
```

```
End Sub
Private Sub Form1_FormClosed(ByVal eventSender As System.Object, ByVal eventArgs As
    System.Windows.Forms.FormClosedEventArgs) Handles Me.FormClosed
    Call cAvax1.EndAvax()
End Sub
Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
    Button1.Click
    Dim sXMLFile As String = String.Empty
    Dim x As Double
    Dim y As Double
    Dim z As Double
    Dim vRetHandles As Object = Nothing
    Dim Answer As Boolean
    sXMLFile = My.Application.Info.DirectoryPath & "\..\drawing.xml"
    Answer = cAvax1.Add_XML(sXMLFile, x, y, z, vRetHandles)
End Sub
```

AddCompoundPath

Purpose

Adds one editable drawing object containing one or more subpaths. Each subpath can mix straight lines, elliptical arcs and cubic Bezier segments. Coordinates use Double values in drawing units.

Declaration

```
Function AddCompoundPath(ByVal path As AvaxPathData) As Integer
```

Parameters and result

Name	Meaning
path	The complete path definition. The control copies its figures and segments; later edits to the caller's objects do not change the drawing.

Returns the new Integer drawing handle, or 0 if the path cannot be added. The control must be started, enabled and editable. The object uses the current drawing attributes. An accepted edit participates in normal Undo/Redo history.

Path values

Create values with `New Avax.AvaxPathData`, `New Avax.AvaxPathFigure` and the segment factory methods below. Figures and Segments are ordinary zero-based .NET lists; these lists have no reserved element 0.

Value	Type	Meaning
AvaxPathData.Figures	List(Of AvaxPathFigure), read-only property	Subpaths in drawing order. Add or remove list entries to construct the input.
AvaxPathData.EvenOddFill	Boolean	True uses the even-odd filling rule, allowing nested contours to form holes. Default True.
AvaxPathFigure.X, Y	Double	The start node, in world coordinates. Default 0, 0.
AvaxPathFigure.Closed	Boolean	Closes the figure back to its start when True. Default False.
AvaxPathFigure.Segments	List(Of AvaxPathSegment), read-only property	Ordered segments. Each begins at the preceding segment's endpoint; the first begins at the figure's X/Y.
AvaxPathFigure.NodeMode	AvaxPathNodeMode	Tangent policy of the start node; default Corner. See SetPathNodeMode.
AvaxPathSegment.Kind	AvaxPathSegmentKind	Line = 0, Arc = 1 or Bezier = 2. A newly constructed segment defaults to Line.
AvaxPathSegment.X, Y	Double	The segment's endpoint.
Control1X, Control1Y, Control2X, Control2Y	Double	The two control points of a cubic Bezier segment, in world coordinates.
CenterX, CenterY	Double	Centre of an arc's ellipse, in world coordinates.
AxisUX, AxisUY, AxisVX, AxisVY	Double	Arc axis vectors U and V. A point at angle a is Centre + U cos(a) + V sin(a). These vectors can represent a rotated or transformed ellipse.
StartAngle, SweepAngle	Double	Arc angles in degrees. Sweep is signed, nonzero and at most 360 degrees in magnitude. Arc endpoints must meet their adjacent segments.

Value	Type	Meaning
AvaxPathSegment.NodeMode	AvaxPathNodeMode	Tangent policy of this segment's endpoint; default Corner.

`AvaxPathSegment.LineTo(x, y)` constructs a straight segment. `BezierTo(c1x, c1y, c2x, c2y, x, y)` constructs a cubic segment. `ArcTo(cx, cy, rx, ry, start, sweep)` constructs an axis-aligned elliptical arc and calculates its endpoint; its angles are in degrees. All geometric arguments are Double.

`AvaxPathData.Clone()` copies the path, its figures and their segments. `AvaxPathFigure.Clone()` copies the figure and segments. `AvaxPathSegment.Clone()` copies the segment's scalar values. These methods return independent mutable values and do not edit a drawing.

VB.NET example

```
Option Strict On
Public Module AddCompoundPathExample
    Public Function AddPath(cad As Avax.cAvax) As Integer
        Dim path As New Avax.AvaxPathData With {.EvenOddFill = True}
        Dim figure As New Avax.AvaxPathFigure With {
            .X = 0.0, .Y = 0.0, .Closed = True}
        figure.Segments.Add(Avax.AvaxPathSegment.LineTo(20.0, 0.0))
        figure.Segments.Add(Avax.AvaxPathSegment.ArcTo(
            20.0, 10.0, 10.0, 10.0, -90.0, 90.0))
        figure.Segments.Add(Avax.AvaxPathSegment.BezierTo(
            30.0, 20.0, 0.0, 20.0, 0.0, 0.0))
        path.Figures.Add(figure)
        Return cad.AddCompoundPath(path)
    End Function
End Module
```

Related options and reference

AddEditableShape

Purpose

Creates an editable native vector outline. Use it when the application needs to change dimensions or corner radii after insertion.

VB.NET declaration

```
Function AddEditableShape(ByVal kind As String, ByVal x As Double, ByVal y As Double, ByVal sizeX As
    Double, ByVal sizeY As Double, Optional ByVal startDegrees As Double = 0, Optional ByVal sweepDegrees As
    Double = 360, Optional ByVal vertices As Integer = 6) As Integer
```

Parameters

Parameter	Meaning
kind	Case-sensitive shape name: Rectangle, Circle, Ellipse, Polygon, Arc or Pie.
x, y	Double anchor: lower-left corner for Rectangle; centre for every other kind.
sizeX, sizeY	Positive finite Double width/height for Rectangle, or radii along the two initial axes for other kinds. Equal radii produce a circle; independent radii produce an elliptical outline.
startDegrees	Finite angular position of the first outline vertex or arc start, in degrees. Rectangle does not use this angle.
sweepDegrees	Finite sweep greater than zero and at most 360. Arc and Pie use this sweep; complete outlines use 360 degrees.
vertices	Polygon vertex count, from 3 to 10000. Ignored for other kinds.

Return value and behavior

Returns a new Integer handle, or zero if the control is not started/enabled or the input cannot be prepared. Validation failure leaves the drawing unchanged. Preparation errors are reported through the ErrorMessage event. The new object uses the current pen attributes and participates in normal creation Undo/Redo. Native .avax saving retains its parameter information.

VB.NET example

Pass a started control on its owning UI thread. This is a complete standalone module.

```
Option Strict On
Imports Avax

Public Module EditableShapeAddExample
    Public Function CreateRectangle(cad As cAvax) As Integer
        Return cad.AddEditableShape("Rectangle", 10.0, 20.0, 80.0, 40.0)
    End Function
End Module
```

Related options and reference

AddParametricShape

Purpose

Adds a preset shape with an editable parameter recipe and a compound-path representation. The recipe supports later parameter/control-point edits; the drawing object uses the current drawing attributes.

Declaration

```
Function AddParametricShape(ByVal kind As String, ByVal x As Double, ByVal y As Double, ByVal width As
    Double, ByVal height As Double, Optional ByVal parameters As IDictionary(Of String, Double) = Nothing) As
    Integer
```

Parameters and result

Name	Meaning
kind	Case-sensitive Kind identifier returned by either catalog method.
x, y	Finite world origin of the shape's local frame, in drawing units.
width, height	Signed spans along its initial local X and Y axes. Neither may be zero, and the absolute frame determinant must be at least 1E-20. Negative spans reflect the corresponding axis.
parameters	Optional dictionary of the preset's parameter names and finite Double values. Omitted entries use catalog defaults. Unknown names, fractional Points or values outside the preset's supported range are rejected. The dictionary is read during creation; later caller edits do not change the drawing.

Returns the new Integer handle, or 0 if the object cannot be added. A started, enabled control with LockEdit=False is required. The new object uses the current pen attributes and participates in creation Undo/Redo. Use GetShapeParameters and ChangeShapeParameter to read or change its recipe values. Keep the parametric association for preset editing, or call ConvertShapeToCurves to work with independent curve nodes.

VB.NET example

```
Option Strict On
Public Module AddParametricShapeExample
    Public Function AddStar(cad As Avax.cAvax) As Integer
        Dim parameters As New System.Collections.Generic.Dictionary(Of String, Double) From {
            {"Points", 7.0}, {"InnerRatio", 0.4}}
        Return cad.AddParametricShape("Star", 10.0, 20.0, 100.0, 80.0, parameters)
    End Function
End Module
```

Related options and reference

AlignItem

cAvax method

Purpose

Moves one drawing object so that its world-coordinate bounds align with a specified rectangle. Combine one horizontal and one vertical alignment with Or.

See AvaxAlignConstants for the alignment flags and their values.

VB.NET declaration

```
Function AlignItem(ByRef xMin As Double, ByRef yMin As Double, ByRef xMax As Double, ByRef yMax As Double,
     ByRef Handle As Integer, ByRef Align As AvaxAlignConstants) As Boolean
```

Parameters

Name	Type and passing	Meaning
xMin	ByRef xMin As Double	Finite Double X coordinate of one target corner. Exchanged with xMax when needed to return the lower X bound.
yMin	ByRef yMin As Double	Finite Double Y coordinate of one target corner. Exchanged with yMax when needed to return the lower Y bound.
xMax	ByRef xMax As Double	Finite Double X coordinate of the opposite corner. Returns the upper X bound.
yMax	ByRef yMax As Double	Finite Double Y coordinate of the opposite corner. Returns the upper Y bound.
Handle	ByRef Handle As Integer	Existing drawable object handle. The ByRef Integer variable is unchanged.
Align	ByRef Align As AvaxAlignConstants	AvaxAlignConstants flags: at most one of Left_a/Right_a/Center_a and one of Top_a/Bottom_a/Middle_a, combined with Or. None_a and conflicting or unknown flags are rejected. The ByRef value is unchanged.

Use the declaration for the overload you call. ByRef arguments can receive outputs; declare variables and arrays with the exact indicated type. Optional defaults are shown in the declaration.

Return value

Boolean - True on success, false on error

Remarks

Aligns the geometric world bounds of one object to an axis-aligned rectangle without scaling, rotating or changing its Z coordinates. Use Left_a, Right_a or Center_a horizontally, and Top_a, Bottom_a or Middle_a vertically. Combine at most one value from each axis with Or; an omitted axis stays unchanged. Top means maximum world Y, including when the screen origin is at the top left. None_a, unknown bits or conflicting flags return False. The control must be started and enabled, and Handle must identify drawable geometry. Finite xMin/xMax and yMin/yMax are sorted and written back through the ByRef arguments, as in R2; Handle and Align are unchanged. The target can be a line or point. Bounds come from the same geometry as GetBorder, including transformed text/documents, curves, blocks and refreshed external references. BeforeMove receives the proposed displacement and a one-based handle array. Its cancellation, replacement handles and displacement changes are honored. Duplicate target handles move once. Invalid targets, nonfinite or overflowing results are rejected before edits. If a handler changes the drawing or invalidates the control, the alignment is abandoned while retaining the handler's own changes. A successful move preserves handles, keys, custom properties, pen/fill settings, object dimensions and selection, and creates one Move undo operation subject to ContinueUndo and UndoRecorderOn. An already aligned object returns True without another undo step or move notification. This method does not create objects. Coordinates use Double in v3. Conflicting combinations and invalid inputs return False; the original manual example Right_a Or Middle_a is supported.

Numeric changes from v2

Update the following variables, arrays or event-data receivers when porting the corresponding v2 call. The declarations above show the current overloads.

Parameter or value	v2 type	v3 type
xMin	Single	Double
yMin	Single	Double
xMax	Single	Double
yMax	Single	Double

For ByRef parameters, use a variable with the exact v3 type. Preserve array dimensions and the documented indexing. Moving from v2 to v3 explains conversion and legacy file-storage rules.

Example

Setup

Add an Avax.cAvax Windows Forms control named `cAvax1`. Paste the procedures into the form class. Example setup and supporting files explains references, initialization and file paths.

Add these Windows Forms controls or components to the form. Fields referenced by Handles clauses must be declared WithEvents, as the Visual Studio designer normally does.

Name	Type
Command1	Button

VB.NET example

```
Dim entityHandle As Integer
Private Sub Form1_Load(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs) Handles
    MyBase.Load
    cAvax1.StartAvax()
    Call cAvax1.Add_Rectangle(0, 0, 10, 10)
    entityHandle = cAvax1.Add_Line(2, 2, 0, 5, 2, 0)
End Sub
Private Sub Form1_FormClosed(ByVal eventSender As System.Object, ByVal eventArgs As
    System.Windows.Forms.FormClosedEventArgs) Handles Me.FormClosed
    cAvax1.EndAvax()
End Sub
Private Sub Command1_Click(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs)
    Handles Command1.Click
    Dim xMin As Double
    Dim yMin As Double
    Dim xMax As Double
    Dim yMax As Double
    Dim Align As Avax.AvaxAlignConstants
    Dim Answer As Boolean
    xMin = 0
    yMin = 0
    xMax = 10
    yMax = 10
    Align = Avax.AvaxAlignConstants.Right_a Or Avax.AvaxAlignConstants.Middle_a
    Answer = cAvax1.AlignItem(xMin, yMin, xMax, yMax, entityHandle, Align)
End Sub
```

AvaxControlGetAllControls

```
Public Function AvaxControlGetAllControls(ParentHandle() As Integer,
                                          Index() As Integer,
                                          sName() As String,
                                          sTag() As String,
                                          sData() As String) As Integer
```

Returns the number of embedded controls and copies their identities and current data into caller-supplied arrays. Results start at index 1; index zero is unused. Entries are ordered by parent handle and then control index. Each position across the five arrays describes the same control.

Array	Contents
ParentHandle	Owning drawing-object handle.
Index	Control index within the owner.
sName	Control name.
sTag	Application-defined tag.
sData	Encoded control pattern containing its current data.

The original signature passes the arrays ByVal. Allocate them before requesting their contents; the method cannot replace an array variable in the caller. Pass Nothing for any array whose contents are not needed, or for all five arrays to query only the count. If an array is shorter than the result, only entries that fit are copied and the return value still reports the full count. Unused trailing positions retain their contents. Before StartAvax, after EndAvax, after disposal, or if a provider changes the drawing or target data while synchronizing, the method returns zero and leaves supplied arrays unchanged.

Document children start at zero; saved shape indices are retained. Stored and active temporary rich-text controls participate. Enumeration remains available when Enabled=False or display updates are suspended. Read an individual current pattern with AvaxControlGetData, query its Double world coordinates with AvaxControlGetShape, or replace its data with AvaxControlSetData.

In v3, handles and control indices are Integer. The example uses the original public signatures and does not require any helper classes.

Related options and reference

AvaxControlGetData

Purpose

Returns the current encoded data pattern of one embedded control, including resolved default values and values supplied by the application's AvaxControl events. Reading defaults does not change the stored drawing.

Declaration

```
Public Function AvaxControlGetData(ByRef ParentHandle As Integer,
                                   ByRef Index As Integer) As String
```

Parameter	Description
ParentHandle	Handle of the document, shape or block that owns the control. An active temporary rich-text control can also use its assigned parent identity.
Index	Child index within that owner. This is not a one-based registry slot; zero is a valid child index.

Returns Nothing if the drawing is not started, the identity is absent, or a provider changes the drawing or control while the request is being prepared. Both identity arguments remain unchanged. The method remains available when Enabled is False or display updates are suspended. PauseAvaxEvents suppresses provider callbacks.

The returned string is a snapshot. Editing it does not edit the control. Submit a complete replacement using AvaxControlSetData. Temporary control data belongs to the current rich-text display and is not saved with the drawing.

VB.NET example

```
Option Strict On
Public Module WidgetDataReadExample
    Public Function ReadControl(cad As Avax.cAvax,
                                parentHandle As Integer,
                                childIndex As Integer) As String
        Dim pattern As String = cad.AvaxControlGetData(parentHandle, childIndex)
        If pattern Is Nothing Then Return Nothing
        System.Diagnostics.Debug.WriteLine(pattern)
        Return pattern
    End Function
End Module
```

Related options and reference

AvaxControlGetShape

Purpose

Retrieves the world-coordinate polygon of an embedded control. Use it to locate or measure the control independently of the current zoom and native editor's screen bounds.

Declaration

```
Public Function AvaxControlGetShape(ByRef ParentHandle As Integer,
                                    ByRef Index As Integer,
                                    ByRef ShapeX() As Double,
                                    ByRef ShapeY() As Double) As Integer
```

Parameter	Description
ParentHandle, Index	Owning object and child index. Neither value is changed.
ShapeX, ShapeY	Newly allocated coordinate arrays. Valid vertices occupy positions 1 through the returned count; position zero is unused. Coordinates are Double in v3.

Returns the vertex count. Document fields retain all six legacy vertices, including the closing and text-edge vertices. A shape control returns its saved outline. Each successful call supplies independent arrays; changing them cannot change the drawing.

Returns zero before StartAvax, after EndAvax or disposal, for an absent identity, or if a provider replaces the requested target while synchronizing. On these paths, both caller arrays retain their identities and contents, including Nothing. The method remains available during disabled or suspended display states.

VB.NET example

```
Option Strict On
Public Module WidgetShapeQueryExample
    Public Function ReadControlBounds(cad As Avax.cAvax
                                      parentHandle As Integer,
                                      childIndex As Integer) As Double()
        Dim x() As Double = Nothing, y() As Double = Nothing
        Dim count As Integer = cad.AvaxControlGetShape(parentHandle, childIndex, x, y)
        If count = 0 Then Return Nothing
        Dim minX As Double = x(1), maxX As Double = x(1)
        Dim minY As Double = y(1), maxY As Double = y(1)
        For i As Integer = 2 To count
            minX = System.Math.Min(minX, x(i))
            maxX = System.Math.Max(maxX, x(i))
            minY = System.Math.Min(minY, y(i))
            maxY = System.Math.Max(maxY, y(i))
        Next
        Return New Double() {minX, minY, maxX, maxY}
    End Function
End Module
```

Related options and reference

AvaxControlRefresh

```
Public Function AvaxControlRefresh(ByRef ParentHandle As Integer,
                                   ByRef Index As Integer) As Boolean
```

Requests current display data for one embedded control and updates its presentation. Use this method after the application changes a caption, text, list, value, picture or other data supplied by the AvaxControl events. It does not replace the stored document text or create an Undo operation.

Parameter	Description
ParentHandle	Handle of the drawing object that owns the control. This can be a document or a block containing controls.
Index	The control's index within that owner. Use the index returned by AvaxControlGetAllControls; the index can be zero.

Returns True when the identified control's data refresh completes. Returns False if the drawing is not started, the control does not exist, a provider changes the drawing or target data before completion, or the request is made from an embedded-control callback. ParentHandle and Index are not changed by the operation. Exceptions raised by application event handlers propagate to the caller.

The method updates both the native editor and the drawing presentation. Display-data changes do not edit the stored shape or document source. If a referenced image file has changed at the same path, its new pixels are loaded. A missing or undecodable image clears the image display; it does not make the control unusable.

Related options and reference

AvaxControlRefreshAll

```
Public Function AvaxControlRefreshAll() As Boolean
```

Requests current display data for all registered embedded controls. It has no parameters. Returns True when the refresh completes in the active drawing; returns False when there is no active drawing, there are no controls, or the refresh cannot run. Calls during control synchronization or from a text-box or combo-box event are refused to avoid restarting the operation that raised the event.

Both refresh methods honor PauseAvaxEvents. With notifications paused, application data providers are not called. Refreshing display data does not itself mark the drawing as changed.

Both methods can resolve control data while Enabled=False, AutoRedraw=False or BeginUpdate is active. Presentation uses the normal refresh mechanism; nested BeginUpdate calls defer the drawing presentation until the matching EndUpdate calls. A successful data refresh requests an explicit presentation even when automatic redraw is disabled.

The data request completes before presentation callbacks run. AfterPaint can draw or register the next temporary rich-text form without turning a completed refresh into a failure. Changes made by those later callbacks remain owned by the application. Active temporary fields, including both rich-text recording modes, participate without adding stored drawing entities.

Related options and reference

AvaxControlSetData

Purpose

Replaces the complete data pattern of an existing embedded control. Use it to change stored captions, text, tags, lists, behavior or other pattern fields. For a document field, the replacement updates that child in the document source and reflows the layout.

Declaration

```
Public Function AvaxControlSetData(ByRef ParentHandle As Integer,
                                   ByRef Index As Integer,
                                   ByRef sData As String) As Boolean
```

Parameter	Description
ParentHandle, Index	Owning object and child index. The target must already exist.
sData	Complete replacement pattern. Include the control type and all fields that must be retained. Omitted fields use defaults. The supplied string is not modified.

Returns True when the update commits. Stored controls create one undoable drawing edit; temporary rich-text controls update only their live temporary source. Shape geometry is retained. A change of control type replaces its native editor.

Returns False if the drawing is not started, the identity is absent, or a callback changes the target before the update commits. Invalid numeric or document-layout data raises an exception before publication. Exceptions from application event handlers propagate. A provider exception does not retain an uncommitted replacement. If Change throws after a stored edit commits, that edit remains complete and undoable. A newer edit made by a callback is retained.

Enabled=False and suspended display updates do not prevent data editing. To request only fresh event-supplied display values without editing stored data, use AvaxControlRefresh.

VB.NET example

This example supplies a full button pattern with world dimensions 30 by 10, a name, application tag and caption. Call it after StartAvax for an existing button identity.

```
Option Strict On
Public Module WidgetDataWriteExample
    Public Function UpdateButton(cad As Avax.cAvax,
                                 parentHandle As Integer,
                                 childIndex As Integer) As Boolean
        Dim pattern As String = "<button><Width30><Height10><name|RunButton><tag|Run><caption|Run
    now></button>"
        Return cad.AvaxControlSetData(parentHandle, childIndex, pattern)
    End Function
End Module
```

Related options and reference

Editable text styles

API reference

These methods apply character fill, outline, spacing and alignment to editable text shapes created with Add_ShapeText. Call them on the control's owning UI thread after StartAvax. Plain text items and rich-text documents use their existing APIs.

IsTextShape

GetTextStyle

SetTextStyle

SetTextStyles

AvaxTextStyle reference

A detached style value. Drawing-unit measurements use Double. Validation occurs when a style is applied. AVAX storage retains the style with the editable object.

New

```
Sub New()
```

Create a style with the defaults listed below. Use GetTextStyle to retain an existing object's settings.

Clone

Purpose

Return an independent copy of this style, including its appearance, font and spacing values.

VB.NET declaration

```
Function Clone() As AvaxTextStyle
```

Example

```
Option Strict On
Public Module Help_CloneTextStyle
    Public Sub Run()
        Dim original As New Avax.AvaxTextStyle()
        Dim copy As Avax.AvaxTextStyle = original.Clone()
        copy.Bold = True
    End Sub
End Module
```

FontName

String; Windows font family name. Default: Segoe UI. Supply a nonempty valid font name.

FontHeight

Double; character height in drawing units. Default: 10. Must be greater than 0 and at most 1,000,000.

Bold

Boolean; use bold characters. Default: False.

Italic

Boolean; use italic characters. Default: False.

Underline

Boolean; underline the text. Default: False.

Strikethrough

Boolean; strike through the text. Default: False.

FillEnabled

Boolean; paint the character interiors with FillColor. Default: True. An outline can remain visible when fill is disabled.

FillColor

System.Drawing.Color; opaque character fill color. Default: Black. Use Opacity for transparency.

OutlineColor

System.Drawing.Color; opaque character outline color. Default: Black.

OutlineWidth

Double; character outline width in drawing units. Default: 0, which disables the outline. Range: 0 to 1,000,000.

Opacity

Double; overall character opacity as a percentage. Default: 100. Range: 0 to 100.

LetterSpacing

Double; additional spacing between characters, in drawing units. Default: 0. Its absolute value must not exceed 100 times FontHeight.

WordSpacing

Double; additional spacing between words, in drawing units. Default: 0. Its absolute value must not exceed 100 times FontHeight.

LineSpacing

Double; baseline separation as a multiple of FontHeight. Default: 1. Range: 0.1 to 20.

Alignment

AvaxTextAlignment; horizontal text alignment. Default: Left. Use one of the defined values below.

All Double values must be finite. FillColor and OutlineColor must be opaque, nonempty colors. A style with both FillEnabled=False and OutlineWidth=0 has no visible character paint.

Example

```
Option Strict On
Public Module Help_TextStyle_5
    Public Sub Run()
        Dim style As New Avax.AvaxTextStyle()
        Dim copy As Avax.AvaxTextStyle = style.Clone()
        copy.FontHeight = 12.0
        copy.Alignment = Avax.AvaxTextAlignment.Centre
    End Sub
End Module
```

AvaxTextAlignment reference

Left

Value 0. Align text to the left.

Centre

Value 1. Center text horizontally.

Right

Value 2. Align text to the right.

DoVertex hatch job

API reference

DoVertex

Purpose

Processes a legacy vertex job file. This topic describes command 511, which writes parallel hatch segments for a polygon. Applications using the control directly can normally use Add_LinearHatch or Hatch_c.

Declaration

```
Public Function DoVertex(ByRef sFile As String) As Boolean
```

sFile is the job file path and is retained. True means the job was processed; read the reply's first field to determine whether hatch geometry was produced. False indicates invalid settings, malformed input or a file access/publication failure. Failed processing preserves the existing job, input and output files.

Job and settings files

The job contains five lines: command number 511; record size; input byte count; packed input path; settings/output path. Paths must identify different files. A record is 8, 12 or 24 bytes of little-endian IEEE Single values. Only the first X/Y pair of each record supplies a polygon vertex; Z and any remaining values are ignored for hatching. The packed file format stays Single even though the calculation and control APIs use Double.

The settings/output file initially contains five lines: positive finite perpendicular spacing; finite angle in degrees; inside flag (1 for inside or 0 for the complement within the bounding rectangle); border flag; binary mask. Border flags are 0 for none, 1 for the bounding rectangle, 2 for the polygon boundary and 3 for both. The mask is empty for every row or a cyclic sequence of 0 and 1.

The polygon closes implicitly; an explicit closing vertex is also accepted. Self-crossing interiors use even-odd filling. Rows include the upper boundary when the spacing reaches it exactly. Vertical segments run bottom to top. Requested borders precede the rows; a zero-only mask suppresses borders too. The Hatch command reference describes the shared row and border order.

Reply and output

For nonempty output the job is replaced by four lines: 1, 3, the original input byte count, and the output path. The output file contains six invariant-culture numeric lines per segment: X1, Y1, Z1, X2, Y2, Z2. Both Z values are zero; there is no additional entity-kind prefix.

If a valid request produces no segments, the reply is 0, 0, the original byte count and output path. The settings file is preserved. Packed input bytes are always preserved. Invalid settings or a locked destination do not partially publish a reply or replace the existing settings file. Use a separate set of files for each job.

VB.NET example

The example creates a new subfolder inside an existing working directory, writes a rectangle job, checks its reply and returns the hatch output path. Remove that owned subfolder after importing the result.

```
Option Strict On
Imports System
Imports System.IO
Imports System.Text

Public Module Help_VertexHatch
    Public Function Create(workingDirectory As String) As String
        Dim folder = Path.Combine(workingDirectory, "hatch-" & Guid.NewGuid().ToString("N"))
        Directory.CreateDirectory(folder)
        Dim data = Path.Combine(folder, "vertices.bin")
        Dim output = Path.Combine(folder, "hatch.txt")
        Dim job = Path.Combine(folder, "hatch.job")
        Using writer As New BinaryWriter(File.Create(data))
            For Each value As Single In New Single() {0, 0, 4, 0, 4, 2, 0, 2}
                writer.Write(value)
            Next
        End Using
        File.WriteAllLines(output, {"1", "0", "1", "0", ""}, Encoding.Default)
        File.WriteAllLines(job, {"511", "8", "32", data, output}, Encoding.Default)
```

```
        Dim processor As New Avax.avVertex.cVertex()
        If Not processor.DoVertex(job) Then Throw New IOException("The batch job could not be processed.")
        Dim reply = File.ReadAllLines(job, Encoding.Default)
        If reply.Length <> 4 OrElse reply(0) <> "1" OrElse reply(1) <> "3" Then
            Throw New InvalidOperationException("The batch job produced no segments.")
        End If
        Return output
    End Function
End Module
```

BeginCompoundPath

Purpose

Starts an interactive path containing straight, circular-arc and cubic Bezier segments in one open or closed figure. Points can come from mouse clicks or SetCoordinate. The control must be started, enabled and editable.

Declaration

```
Function BeginCompoundPath(Optional ByVal closed As Boolean = False) As Boolean
```

Parameters and result

`closed` determines whether the finished figure closes back to its start. Returns True when the session starts, False when the control cannot begin editing. Beginning cancels the preceding drawing tool, clears selection and resets CompoundPathSegmentKind to Line. Compound paths use X/Y drawing coordinates; the Z argument of SetCoordinate is not retained.

Point order

Stage or kind	Points to supply
First point	The figure's starting X/Y, independent of the chosen segment kind.
Line	One endpoint.
Arc	One intermediate point on the circular arc, then its endpoint. The arc starts at the preceding completed endpoint. Three collinear points are invalid.
Bezier	First control point, second control point, then endpoint.

Set CompoundPathSegmentKind between complete segments. Call FinishCompoundPath to create the drawing object. Setting Command to Cancel_c abandons the draft without adding an object.

VB.NET example

```
Option Strict On
Public Module BeginCompoundPathExample
    Public Function DrawMixedPath(cad As Avax.cAvax) As Integer
        If Not cad.BeginCompoundPath(False) Then Return 0
        cad.SetCoordinate(0.0, 0.0, 0.0)
        cad.SetCoordinate(20.0, 0.0, 0.0)
        cad.CompoundPathSegmentKind = Avax.AvaxPathSegmentKind.Arc
        cad.SetCoordinate(25.0, 5.0, 0.0)
        cad.SetCoordinate(20.0, 10.0, 0.0)
        cad.CompoundPathSegmentKind = Avax.AvaxPathSegmentKind.Bezier
        cad.SetCoordinate(15.0, 20.0, 0.0)
        cad.SetCoordinate(5.0, 20.0, 0.0)
        cad.SetCoordinate(0.0, 10.0, 0.0)
        Return cad.FinishCompoundPath()
    End Function
End Module
```

Related options and reference

BeginCurveNodeEditing

Purpose

Enables interactive node editing for selected vector paths. The canvas displays available node handles, Bezier tangent controls and arc-bend controls for the selected object.

Declaration

```
Function BeginCurveNodeEditing() As Boolean
```

Parameters and result

This method has no parameters. Returns True when the editing mode is enabled, False if the control is unstarted, disabled, disposed or locked for editing. It retains selection and switches to ordinary selection mode. The method itself does not change geometry and can succeed with no selected object; select an eligible object to expose its handles. A completed handle drag commits the edit through normal history. Another drawing command ends this mode. For programmatic editing, use GetShapeControlPoints and ChangeShapeControlPoint.

Control-point identifiers

For an ordinary path, identifiers have the form `figure:node:role`. Figure and node numbers are zero-based: node 0 is the figure start and node n > 0 is the endpoint of Segments(n - 1). Role `node` identifies that node. Roles `c1` and `c2` identify the two Bezier controls of segment n - 1; role `bend` identifies the midpoint control of arc n - 1. Bezier control1 is geometrically attached to the segment's start, even though its identifier uses the segment's endpoint number. Obtain current identifiers from GetShapeControlPoints; edits such as splitting or reversing change their indices.

VB.NET example

```
Option Strict On
Public Module BeginCurveNodeEditingExample
    Public Function AddEditableCubic(cad As Avax.cAvax) As Boolean
        Dim path As New Avax.AvaxPathData
        Dim figure As New Avax.AvaxPathFigure
        figure.Segments.Add(Avax.AvaxPathSegment.BezierTo(
            10.0, 20.0, 20.0, 20.0, 30.0, 0.0))
        path.Figures.Add(figure)
        Dim handle As Integer = cad.AddCompoundPath(path)
        If handle = 0 Then Return False
        Dim selected As Integer() = {0, handle}
        cad.SelectItems(selected)
        Return cad.BeginCurveNodeEditing()
    End Function
End Module
```

Related options and reference

BeginUpdate

```
Public Sub BeginUpdate()
```

Suspends presentation of drawing changes until the matching EndUpdate. Use it when adding or changing several objects to avoid intermediate images. The drawing continues to accept edits, and model events such as Change remain active. This method does not merge the operations into one Undo record.

The last displayed drawing stays visible if Windows requests a repaint during suspension. Nested BeginUpdate calls are supported: each requires a matching EndUpdate. Use Try...Finally so an exception cannot leave the display suspended. To suppress notifications separately, use PauseAvaxEvents.

VB.NET example

Pass an already started control. The two lines keep their own Undo entries and model notifications, while only the completed result is presented.

```
Option Strict On
Imports System

Public Module UpdateRefreshExample
    Public Function AddTwoLines(cad As Avax.cAvax) As Integer()
        If cad Is Nothing Then Throw New ArgumentNullException(NameOf(cad))
        Dim itemHandles(2) As Integer
        cad.BeginUpdate()
        Try
            itemHandles(1) = cad.Add.Line(5.0, 8.0, 0.0, 25.0, 8.0, 0.0)
            itemHandles(2) = cad.Add.Line(5.0, 16.0, 0.0, 25.0, 16.0, 0.0)
            If itemHandles(1) = 0 OrElse itemHandles(2) = 0 Then
                Throw New InvalidOperationException("A line could not be created.")
            End If
            Return itemHandles
        Finally
            cad.EndUpdate()
        End Try
    End Function
End Module
```

Index zero of the returned handle array is unused, following the control's one-based handle-array convention. This example does not roll back successful edits if a later operation fails.

Related options and reference

CalcBoundsDrawing

cAvax method

Purpose

Recalculates the XYZ extents of the current drawable geometry and updates the cached drawing bounds returned by GetBoundsDrawing.

VB.NET declaration

```
Sub CalcBoundsDrawing()
```

Result

This procedure has no return value. Any outputs are supplied through the documented ByRef arguments.

Remarks

Call this procedure after inserting, editing or deleting geometry when you need fresh drawing extents. GetBoundsDrawing reads the cached result; it does not perform this recalculation itself. CalcBoundsDrawing expands the total work area when necessary, but does not change the current view, raise a zoom-change event or add an undo record. An empty drawing produces six zero extents. CalcBounds_c uses this recalculation; CalcExtends_c additionally fits the view. Coordinates returned by GetBoundsDrawing are Double drawing units. The control must be started and enabled.

Example

Setup

Add an Avax.cAvax Windows Forms control named `cAvax1`. Paste the procedures into the form class. Example setup and supporting files explains references, initialization and file paths.

Add these Windows Forms controls or components to the form. Fields referenced by Handles clauses must be declared WithEvents, as the Visual Studio designer normally does.

Name	Type
Command1	Button

VB.NET example

```
Private Sub Form1_Load(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs) Handles
    MyBase.Load
    cAvax1.StartAvax()
    Call cAvax1.Add_Line(0, 0, 0, 5, 2, 0)
End Sub
Private Sub Form1_FormClosed(ByVal eventSender As System.Object, ByVal eventArgs As
    System.Windows.Forms.FormClosedEventArgs) Handles Me.FormClosed
    cAvax1.EndAvax()
End Sub
Private Sub Command1_Click(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs)
    Handles Command1.Click
    Dim yMin1, xMin1, zMin1 As Double
    Dim yMax1, xMax1, zMax1 As Double
    Dim yMin2, xMin2, zMin2 As Double
    Dim yMax2, xMax2, zMax2 As Double
    Dim incDy, incDx, incDz As Double
    Dim sMsg As String = String.Empty
    Call cAvax1.GetBoundsDrawing(xMin1, yMin1, zMin1, xMax1, yMax1, zMax1)
    Call cAvax1.Add_Line(0, 0, 0, 200, 100, 0)
    Call cAvax1.CalcBoundsDrawing()
    Call cAvax1.GetBoundsDrawing(xMin2, yMin2, zMin2, xMax2, yMax2, zMax2)
    incDx = System.Math.Abs(xMax2 - xMin2) - System.Math.Abs(xMax1 - xMin1)
    incDy = System.Math.Abs(yMax2 - yMin2) - System.Math.Abs(yMax1 - yMin1)
    incDz = System.Math.Abs(zMax2 - zMin2) - System.Math.Abs(zMax1 - zMin1)
```

```
    sMsg = "incDx=" & Format(incDx) & vbCrLf
    sMsg = sMsg & "incDy=" & Format(incDy) & vbCrLf
    sMsg = sMsg & "incDz=" & Format(incDz) & vbCrLf
    CAvax1.Command = Avax.AvaxCommand.Extends_c
    MsgBox(sMsg, MsgBoxStyle.Information, "CalcBoundsDrawing")
End Sub
```

CenterWorkspacePage

Purpose

Center the paper on drawing extents, or on the current view when the drawing is empty.

VB.NET declaration

```
Sub CenterWorkspacePage()
```

Parameters

None.

Return value and behavior

No return value. Changes PageOriginX/Y as one undoable workspace edit; does not move objects or change zoom/scale. A drawing containing only data records uses the current visible view, just like an empty drawing. The midpoint is calculated without overflowing when finite drawing coordinates have a very large span. If the resulting sheet cannot have representable positive dimensions at that location, throws ArgumentException and retains the previous page setup. Repeated centering on the same location is a no-op.

Example

Call this procedure on the owning UI thread with a control already initialized by `StartAvax()`. Handles must belong to that control. File paths are supplied by the caller.

```
Option Strict On
Imports System
Imports Avax

Public Module Help_CenterWorkspacePage
    Public Sub Run(cad As cAvax)
        cad.CenterWorkspacePage()
    End Sub
End Module
```

Related options and reference

ChangeColor

cAvax method

Purpose

Changes the color of the objects described in a specific handles array

VB.NET declaration

```
Function ChangeColor(ByRef h As Integer(), ByRef iColor As Integer) As Boolean
```

Parameters

Name	Type and passing	Meaning
h	ByRef h As Integer()	One-based Integer() containing the target object handles. Element 0 is unused. Supply at least one valid handle; duplicate handles are accepted. The array is not rewritten.
iColor	ByRef iColor As Integer	Integer palette index. -1 uses PenColor. Legacy indices wrap within the current palette; invalid negative or out-of-range indices become 0. The input variable is unchanged.

Use the declaration for the overload you call. ByRef arguments can receive outputs; declare variables and arrays with the exact indicated type. Optional defaults are shown in the declaration.

Return value

Boolean - True on success, false on error.

Remarks

Changes the palette color index on each requested object record. Use -1 to inherit PenColor. Negative values other than -1, or values above 255 that do not identify an existing extended palette entry, use index 0. Indices in the legacy 0..255 range wrap within the current palette. The color is a palette index, not an RGB value. Shape fill colors and explicit colors inside rich text remain separate from this object color. Supply a one-based Integer() handle array: index 0 is unused. The control must be started and enabled. All handles are checked before any object is changed; a missing handle or invalid input returns False and leaves the batch unchanged. Duplicate handles are processed once. Successful edits retain handles and the current selection and create one Change undo operation, subject to ContinueUndo and UndoRecorderOn. The ByRef inputs are not rewritten. An unchanged or inapplicable batch returns True without creating an undo entry or Change event. Stored geometry, object metadata and document content remain unchanged.

Numeric changes from v2

Update the following variables, arrays or event-data receivers when porting the corresponding v2 call. The declarations above show the current overloads.

Parameter or value	v2 type	v3 type
iColor	Short	Integer

For ByRef parameters, use a variable with the exact v3 type. Preserve array dimensions and the documented indexing. Moving from v2 to v3 explains conversion and legacy file-storage rules.

Example

Setup

Add an Avax.cAvax Windows Forms control named `cAvax1`. Paste the procedures into the form class. Example setup and supporting files explains references, initialization and file paths.

Add these Windows Forms controls or components to the form. Fields referenced by Handles clauses must be declared WithEvents, as the Visual Studio designer normally does.

Name	Type
Command1	Button

VB.NET example

```
Dim h() As Integer = Nothing
Private Sub Form1_Load(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs) Handles
    MyBase.Load
    cAvax1.StartAvax()
    ReDim h(1)
    h(1) = CAvax1.Add_Line(1, 1, 0, 8, 8, 0, Nothing, 6, 3, -1, -1)
End Sub
Private Sub Form1_FormClosed(ByVal eventSender As System.Object, ByVal eventArgs As
    System.Windows.Forms.FormClosedEventArgs) Handles Me.FormClosed
    cAvax1.EndAvax()
End Sub
Private Sub Command1_Click(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs)
    Handles Command1.Click
    Static iColor As Integer
    Dim fOk As Boolean
    iColor = iColor + 1
    If iColor > 15 Then iColor = 0
    fOk = cAvax1.ChangeColor(h, iColor)
End Sub
```

ChangeCompoundPath

Purpose

Replaces the geometry of an existing compound path while retaining its handle. The input is copied and validated before committing the edit.

Declaration

```
Function ChangeCompoundPath(ByVal handle As Integer, ByVal path As AvaxPathData) As Boolean
```

Parameters and result

Name	Meaning
handle	Handle of an object readable by GetCompoundPath.
path	The complete replacement value, usually obtained with GetCompoundPath and edited by the application.

Returns True for an accepted edit, False otherwise. Editing requires a started, enabled and editable control. The object retains its handle and drawing attributes. This operation removes any parametric recipe from the object, leaving directly editable segments. Coordinate changes must preserve arc connections; moving a neighbouring node without updating the arc definition can make the replacement invalid. Use ChangeShapeControlPoint for control-point edits performed by the control.

VB.NET example

```
Option Strict On
Public Module ChangeCompoundPathExample
    Public Function WidenFirstCorner(cad As Avax.cAvax) As Boolean
        Dim initial As New Avax.AvaxPathData
        Dim figure As New Avax.AvaxPathFigure With {.Closed = True}
        figure.Segments.Add(Avax.AvaxPathSegment.LineTo(20.0, 0.0))
        figure.Segments.Add(Avax.AvaxPathSegment.LineTo(20.0, 10.0))
        figure.Segments.Add(Avax.AvaxPathSegment.LineTo(0.0, 10.0))
        initial.Figures.Add(figure)
        Dim handle As Integer = cad.AddCompoundPath(initial)
        If handle = 0 Then Return False
        Dim edited As Avax.AvaxPathData = cad.GetCompoundPath(handle)
        If edited Is Nothing Then Return False
        edited.Figures(0).Segments(0).X = 25.0
        Return cad.ChangeCompoundPath(handle, edited)
    End Function
End Module
```

Related options and reference

ChangeDocumentFontSize

Changes the intrinsic font height of a Document_i object and recomputes its layout. The insertion point, Z coordinate, transformation matrix, text, font binding and object metadata remain unchanged. It accepts Document_i only; use ChangeShapeDocumentFontSize for a document shape.

```
Public Function ChangeDocumentFontSize(ByRef avHandleDocument As Integer,
                                       ByRef DocumentFontSize As Double) As Boolean
```

Related options and reference

ChangeDocumentWidth

Changes the logical wrapping width of a Document_i object. As in R2, this method also accepts a shape with a Document_f fill. For either supported type, it returns True when the change succeeds. Changing the width can increase or decrease the number of lines and the total document height.

```
Public Function ChangeDocumentWidth(ByRef avHandleDocument As Integer,
                                    ByRef DocumentWidth As Double) As Boolean
```

Related options and reference

ChangeElevation3D

cAvax method

Purpose

Changes the 3D elevation of the objects described in a specific handles array

VB.NET declaration

```
Function ChangeElevation3D(ByRef h As Integer(), ByRef Elevation3D As Double) As Boolean
```

Parameters

Name	Type and passing	Meaning
h	ByRef h As Integer()	One-based Integer() containing the target object handles. Element 0 is unused. Supply at least one valid handle; duplicate handles are accepted. The array is not rewritten.
Elevation3D	ByRef Elevation3D As Double	Finite Double elevation attribute in drawing units. Signed values are permitted; vertex coordinates and the input variable remain unchanged.

Use the declaration for the overload you call. ByRef arguments can receive outputs; declare variables and arrays with the exact indicated type. Optional defaults are shown in the declaration.

Return value

Boolean - True on success, false on error.

Remarks

Sets the stored 3D elevation attribute on each requested object record. The value is a finite Double in drawing units and may be positive, zero or negative. It does not translate the object or overwrite its vertex Z coordinates. In particular, -1 is a literal elevation, not a current-pen sentinel. Supply a one-based Integer() handle array: index 0 is unused. The control must be started and enabled. All handles are checked before any object is changed; a missing handle or invalid input returns False and leaves the batch unchanged. Duplicate handles are processed once. Successful edits retain handles and the current selection and create one Change undo operation, subject to ContinueUndo and UndoRecorderOn. The ByRef inputs are not rewritten. An unchanged or inapplicable batch returns True without creating an undo entry or Change event. Stored geometry, object metadata and document content remain unchanged.

Numeric changes from v2

Update the following variables, arrays or event-data receivers when porting the corresponding v2 call. The declarations above show the current overloads.

Parameter or value	v2 type	v3 type
Elevation3D	Single	Double

For ByRef parameters, use a variable with the exact v3 type. Preserve array dimensions and the documented indexing. Moving from v2 to v3 explains conversion and legacy file-storage rules.

Example

Setup

Add an Avax.cAvax Windows Forms control named `cAvax1`. Paste the procedures into the form class. Example setup and supporting files explains references, initialization and file paths.

Add these Windows Forms controls or components to the form. Fields referenced by Handles clauses must be declared WithEvents, as the Visual Studio designer normally does.

Name	Type
Command1	Button
Label1	Label

VB.NET example

```
Dim h(1) As Integer
Dim Elevation3D As Double
Private Sub Form1_Load(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs) Handles
    MyBase.Load
    Command1.Text = "ChangeElevation3D"
    cAvax1.StartAvax()
    h(1) = cAvax1.Add_Line(0, 0, 0, 10, 10, 0)
    Label1.Text = "The 3D Elevation of the line is 0"
End Sub
Private Sub Form1_FormClosed(ByVal eventSender As System.Object, ByVal eventArgs As
    System.Windows.Forms.FormClosedEventArgs) Handles Me.FormClosed
    cAvax1.EndAvax()
End Sub
Private Sub Command1_Click(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs)
    Handles Command1.Click
    Elevation3D = Elevation3D + 3
    If cAvax1.ChangeElevation3D(h, Elevation3D) = True Then
        Label1.Text = "The 3D Elevation of the line is" & Str(Elevation3D)
    End If
End Sub
```

ChangeFontNumber

cAvax method

Purpose

Changes the font nunber of a text object

VB.NET declaration

```
Function ChangeFontNumber(ByRef hTextItem As Integer, ByRef iFontNumber As Integer) As Integer
```

Parameters

Name	Type and passing	Meaning
hTextItem	ByRef hTextItem As Integer	Existing text, document or text-shape handle. Returns the same handle on success, or zero on failure. The input variable is unchanged.
iFontNumber	ByRef iFontNumber As Integer	Integer font-table index. -1 selects PenFontNumber; zero selects the host font. Other negatives and values above 256 normalize to zero. A blank or unavailable supported slot resolves to the host font. The input variable is unchanged.

Use the declaration for the overload you call. ByRef arguments can receive outputs; declare variables and arrays with the exact indicated type. Optional defaults are shown in the declaration.

Return value

Integer - Is the Avax handle of the new text

Remarks

Changes the font binding of Text_i, Document_i or a text shape (Shape_i with Text_s and Text_f fill). Other types, including document shapes, return zero. The control must be started and enabled. Success returns the existing handle; failure returns zero. Neither ByRef argument is changed.

iFontNumber is an Integer in v3; it was Short in v2. -1 selects the current PenFontNumber. Zero selects the current host font. Other negative values and values above 256 normalize to zero. A positive supported index selects that font-table entry; a missing or blank entry uses the current host family and charset. Text and documents retain a live table binding. Zero clears an independently named imported font binding. A text shape stores the resolved font family and charset, as in R2. Content and bold/italic/underline/strikeout remain unchanged. Ordinary text is remeasured with the new font. Documents retain their dimensions and transformation matrix, with the font index synchronized for saving. Text shapes retain their existing layout width and height and rebuild their rectangular frame; changing their font does not resize the frame to the new natural glyph width. Position, Z, rotation, keys, custom properties, selection and embedded resources are preserved. The edit uses one Change undo operation, subject to ContinueUndo and UndoRecorderOn, without creating a new object. Reassigning the same effective binding succeeds without adding history. Invalid preparation leaves the original object and history unchanged.

Numeric changes from v2

Update the following variables, arrays or event-data receivers when porting the corresponding v2 call. The declarations above show the current overloads.

Parameter or value	v2 type	v3 type
iFontNumber	Short	Integer

For ByRef parameters, use a variable with the exact v3 type. Preserve array dimensions and the documented indexing. Moving from v2 to v3 explains conversion and legacy file-storage rules.

Example

Setup

Add an Avax.cAvax Windows Forms control named `cAvax1`. Paste the procedures into the form class. Example setup and supporting files explains references, initialization and file paths.

Add these Windows Forms controls or components to the form. Fields referenced by Handles clauses must be declared WithEvents, as the Visual Studio designer normally does.

Name	Type
Command1	Button

VB.NET example

```
Dim hTextItem As Integer
Dim iFontNumber As Integer
Private Sub Form1_Load(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs) Handles
    MyBase.Load
    Command1.Text = "ChangeFontNumber"
    cAvax1.StartAvax()
    Dim sFonts(5) As String
    Dim lfontsSet(5) As Integer
    sFonts(1) = "Arial"
    sFonts(2) = "Courier New"
    sFonts(3) = "Times New Roman"
    sFonts(4) = "Arial Black"
    sFonts(5) = "Arial Narrow"
    Call cAvax1.SetFonts(sFonts, lfontsSet)
    CAvax1.Font = New Font(sFonts(1), CAvax1.Font.Size)
    cAvax1.FontCharset = 0
    iFontNumber = 1
    hTextItem = CAvax1.Add_Text("This is my text", 0, 5, 0, Nothing, 8.0!, iFontNumber, 0, -1, -1, -1, -1,
     False, False, False, False, -1)
End Sub
Private Sub Form1_FormClosed(ByVal eventSender As System.Object, ByVal eventArgs As
    System.Windows.Forms.FormClosedEventArgs) Handles Me.FormClosed
    cAvax1.EndAvax()
End Sub
Private Sub Command1_Click(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs)
    Handles Command1.Click
    iFontNumber = iFontNumber + 1
    If iFontNumber > 5 Then iFontNumber = 0
    hTextItem = CAvax1.ChangeFontNumber(hTextItem, iFontNumber)
    CAvax1.Refresh()
End Sub
```

ChangeHeight3D

cAvax method

Purpose

Changes the 3D height of the objects described in a specific handles array

VB.NET declaration

```
Function ChangeHeight3D(ByRef h As Integer(), ByRef Height3D As Double) As Boolean
```

Parameters

Name	Type and passing	Meaning
h	ByRef h As Integer()	One-based Integer() containing the target object handles. Element 0 is unused. Supply at least one valid handle; duplicate handles are accepted. The array is not rewritten.
Height3D	ByRef Height3D As Double	Finite Double height attribute in drawing units. Signed values are permitted; vertex coordinates and the input variable remain unchanged.

Use the declaration for the overload you call. ByRef arguments can receive outputs; declare variables and arrays with the exact indicated type. Optional defaults are shown in the declaration.

Return value

Boolean - True on success, false on error.

Remarks

Sets the stored 3D height attribute on each requested object record. The value is a finite Double in drawing units and may be positive, zero or negative. It does not scale the object or overwrite its vertex Z coordinates. In particular, -1 is a literal height, not a current-pen sentinel. Supply a one-based Integer() handle array: index 0 is unused. The control must be started and enabled. All handles are checked before any object is changed; a missing handle or invalid input returns False and leaves the batch unchanged. Duplicate handles are processed once. Successful edits retain handles and the current selection and create one Change undo operation, subject to ContinueUndo and UndoRecorderOn. The ByRef inputs are not rewritten. An unchanged or inapplicable batch returns True without creating an undo entry or Change event. Stored geometry, object metadata and document content remain unchanged.

Numeric changes from v2

Update the following variables, arrays or event-data receivers when porting the corresponding v2 call. The declarations above show the current overloads.

Parameter or value	v2 type	v3 type
Height3D	Single	Double

For ByRef parameters, use a variable with the exact v3 type. Preserve array dimensions and the documented indexing. Moving from v2 to v3 explains conversion and legacy file-storage rules.

Example

Setup

Add an Avax.cAvax Windows Forms control named `cAvax1`. Paste the procedures into the form class. Example setup and supporting files explains references, initialization and file paths.

Add these Windows Forms controls or components to the form. Fields referenced by Handles clauses must be declared WithEvents, as the Visual Studio designer normally does.

Name	Type
Command1	Button

Name	Type
Label1	Label

VB.NET example

```
Dim h(1) As Integer
Dim Height3D As Double
Private Sub Form1_Load(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs) Handles
    MyBase.Load
    Command1.Text = "ChangeHeight3D"
    cAvax1.StartAvax()
    h(1) = cAvax1.Add_Line(0, 0, 0, 10, 10, 0)
    Label1.Text = "The 3D Height of the line is 0"
End Sub
Private Sub Form1_FormClosed(ByVal eventSender As System.Object, ByVal eventArgs As
    System.Windows.Forms.FormClosedEventArgs) Handles Me.FormClosed
    cAvax1.EndAvax()
End Sub
Private Sub Command1_Click(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs)
    Handles Command1.Click
    Height3D = Height3D + 3
    If cAvax1.ChangeHeight3D(h, Height3D) = True Then
        Label1.Text = "The 3D Height of the line is" & Str(Height3D)
    End If
End Sub
```

ChangeItemOrder

Purpose

Changes the front-to-back order of all current root items as one reversible operation. It includes groups, native shapes, lines, imported items and non-drawing data roots.

VB.NET declaration

```
Function ChangeItemOrder(ByVal frontToBack As Integer()) As Boolean
```

Parameters

Parameter	Meaning
frontToBack	One-based Integer array containing every current root handle exactly once, in front-to-back order. Slot zero is reserved and ignored. Include hidden and non-drawing roots; group children are represented by their root handle.

Return value and behavior

Returns True on success. An identical current sequence is a no-op and retains its original stored values. Otherwise the method assigns descending positive order values, from the root count down to one. Nothing, missing/duplicate/unknown/extra roots or a disabled/inactive control return False before an edit is committed. Preparation errors are reported through ErrorMessage. An empty drawing accepts an array containing only the reserved slot. The caller array, geometry, fills other than order, user properties, group children, selection and current drawing command are retained. One complete change is one Undo/Redo action; native .avax saving retains the result.

When ShapeZorderOn_p is enabled, native controls follow the same root order as the vector scene; local stacking within a root is retained. Disabling the setting returns both drawing and native windows to handle order. Normal redraw settings control when the frame is presented. The Change event sees the complete committed permutation. A newer order or drawing replacement made by its callback is retained; host callback exceptions can propagate after the order has committed.

VB.NET example

Pass a started control on its owning UI thread. For the reversal example, supply every current root once in the current front-to-back order, with slot zero reserved. Enable ShapeZorderOn_p when you want the stored order to determine the visible foreground.

```
Option Strict On
Imports Avax

Public Module ItemOrderChangeExample
    Public Function Reverse(cad As cAvax, currentFrontToBack As Integer()) As Boolean
        If currentFrontToBack Is Nothing OrElse currentFrontToBack.Length = 0 Then Return False
        Dim wanted(currentFrontToBack.Length - 1) As Integer
        For i = 1 To wanted.Length - 1
            wanted(i) = currentFrontToBack(currentFrontToBack.Length - i)
        Next
        Return cad.ChangeItemOrder(wanted)
    End Function
End Module
```

Related options and reference

ChangeLayer

cAvax method

Purpose

Changes the layer of the objects described in a specific handles array

VB.NET declaration

```
Function ChangeLayer(ByRef h As Integer(), ByRef iLayer As Integer) As Boolean
```

Parameters

Name	Type and passing	Meaning
h	ByRef h As Integer()	An array of Integers for the handles of the objects
iLayer	ByRef iLayer As Integer	Nonnegative Integer destination index. The current PenLayer property remains 0..255; explicit larger entity indices are a v3 extension with legacy-format limits.

Use the declaration for the overload you call. ByRef arguments can receive outputs; declare variables and arrays with the exact indicated type. Optional defaults are shown in the declaration.

Return value

Boolean - True on success, false on error.

Numeric changes from v2

Update the following variables, arrays or event-data receivers when porting the corresponding v2 call. The declarations above show the current overloads.

Parameter or value	v2 type	v3 type
iLayer	Short	Integer

For ByRef parameters, use a variable with the exact v3 type. Preserve array dimensions and the documented indexing. Moving from v2 to v3 explains conversion and legacy file-storage rules.

Example

Setup

Add an Avax.cAvax Windows Forms control named `cAvax1`. Paste the procedures into the form class. Example setup and supporting files explains references, initialization and file paths.

Add these Windows Forms controls or components to the form. Fields referenced by Handles clauses must be declared WithEvents, as the Visual Studio designer normally does.

Name	Type
Command1	Button
Label1	Label

VB.NET example

```
Dim iLayer As Integer
Dim h(1) As Integer
Private Sub Form1_Load(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs) Handles
    MyBase.Load
    Command1.Text = "ChangeLayer"
    cAvax1.StartAvax()
    iLayer = 0
```

```
    h(1) = CAvax1.Add_Line(0, 0, 0, 10, 10, 0, Nothing, -1, -1, -1, iLayer)
    Label1.Text = "Line is in the layer" & Str(iLayer)
End Sub
Private Sub Form1_FormClosed(ByVal eventSender As System.Object, ByVal eventArgs As
    System.Windows.Forms.FormClosedEventArgs) Handles Me.FormClosed
    cAvax1.EndAvax()
End Sub
Private Sub Command1_Click(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs)
    Handles Command1.Click
    iLayer = iLayer + 1
    If iLayer > 255 Then iLayer = 0
    If cAvax1.ChangeLayer(h, iLayer) = True Then
        Label1.Text = "Now the line is in the layer" & Str(iLayer)
    End If
End Sub
```

ChangeLineStyle

cAvax method

Purpose

Changes the line style of the linear objects described in a specific handles array

VB.NET declaration

```
Function ChangeLineStyle(ByRef h As Integer(), ByRef iLineStyle As Integer) As Boolean
```

Parameters

Name	Type and passing	Meaning
h	ByRef h As Integer()	One-based Integer() containing the target object handles. Element 0 is unused. Supply at least one valid handle; duplicate handles are accepted. The array is not rewritten.
iLineStyle	ByRef iLineStyle As Integer	Integer line-style table index, 0..255. -1 uses PenLineStyle; other invalid indices become 0. The input variable is unchanged.

Use the declaration for the overload you call. ByRef arguments can receive outputs; declare variables and arrays with the exact indicated type. Optional defaults are shown in the declaration.

Return value

Boolean - True on success, false on error.

Remarks

Changes the line-style index of lines, polylines, ellipses, shape borders, curves and external-line references in the array. Use -1 to inherit PenLineStyle. Other values outside 0..255 become 0 (continuous). Points, text, documents, blocks and data records are skipped. This method does not change text alignment or shape fill patterns. Supply a one-based Integer() handle array: index 0 is unused. The control must be started and enabled. All handles are checked before any object is changed; a missing handle or invalid input returns False and leaves the batch unchanged. Duplicate handles are processed once. Successful edits retain handles and the current selection and create one Change undo operation, subject to ContinueUndo and UndoRecorderOn. The ByRef inputs are not rewritten. An unchanged or inapplicable batch returns True without creating an undo entry or Change event. Stored geometry, object metadata and document content remain unchanged.

Numeric changes from v2

Update the following variables, arrays or event-data receivers when porting the corresponding v2 call. The declarations above show the current overloads.

Parameter or value	v2 type	v3 type
iLineStyle	Short	Integer

For ByRef parameters, use a variable with the exact v3 type. Preserve array dimensions and the documented indexing. Moving from v2 to v3 explains conversion and legacy file-storage rules.

Example

Setup

Add an Avax.cAvax Windows Forms control named `cAvax1`. Paste the procedures into the form class. Example setup and supporting files explains references, initialization and file paths.

Add these Windows Forms controls or components to the form. Fields referenced by Handles clauses must be declared WithEvents, as the Visual Studio designer normally does.

Name	Type
Command1	Button
Label1	Label

VB.NET example

```
Dim iLineStyle As Integer
Dim AvaxHandle As Integer
Private Sub Form1_Load(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs) Handles
    MyBase.Load
    Command1.Text = "ChangeLineStyle"
    cAvax1.StartAvax()
    iLineStyle = 0
    Label1.Text = cAvax1.GetLineStyle(iLineStyle)
    AvaxHandle = CAvax1.Add_Line(0, 0, 0, 25, 10, 0, Nothing, 5, -1, iLineStyle, -1)
End Sub
Private Sub Form1_FormClosed(ByVal eventSender As System.Object, ByVal eventArgs As
    System.Windows.Forms.FormClosedEventArgs) Handles Me.FormClosed
    cAvax1.EndAvax()
End Sub
Private Sub Command1_Click(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs)
    Handles Command1.Click
    Dim h(1) As Integer
    Dim Answer As Boolean
    iLineStyle = iLineStyle + 1
    If iLineStyle > 9 Then iLineStyle = 0
    h(1) = AvaxHandle
    Answer = cAvax1.ChangeLineStyle(h, iLineStyle)
    Label1.Text = cAvax1.GetLineStyle(iLineStyle)
End Sub
```

ChangePathSegmentKind

Purpose

Replaces one completed segment with a line, cubic Bezier or circular arc joining the same endpoints.

Declaration

```
Function ChangePathSegmentKind(ByVal handle As Integer, ByVal figure As Integer, ByVal segment As Integer,
     ByVal kind As AvaxPathSegmentKind) As Boolean
```

Parameters and result

`handle` identifies the path; `figure` and `segment` are zero-based indices. `kind` is Line = 0, Arc = 1 or Bezier = 2. Line uses the endpoint chord. Bezier places initial controls at one-third and two-thirds of that chord. Arc constructs a circular arc through the endpoints and an automatically chosen intermediate point. An arc conversion requires distinct usable endpoints.

Returns True for an accepted edit, False for invalid input or an object that cannot be edited. The segment's endpoint NodeMode is retained. This replaces its geometry even if kind equals its existing Kind, so existing Bezier controls or arc curvature can change. Use ChangePathSegmentParameter for a subsequent control/radius adjustment. The edit retains the object handle, removes a parametric recipe and participates in Undo/Redo.

VB.NET example

```
Option Strict On
Public Module ChangePathSegmentKindExample
    Public Function AddArcFromChord(cad As Avax.cAvax) As Integer
        Dim path As New Avax.AvaxPathData
        Dim figure As New Avax.AvaxPathFigure
        figure.Segments.Add(Avax.AvaxPathSegment.LineTo(30.0, 0.0))
        path.Figures.Add(figure)
        Dim handle As Integer = cad.AddCompoundPath(path)
        If handle = 0 Then Return 0
        If Not cad.ChangePathSegmentKind(
            handle, 0, 0, Avax.AvaxPathSegmentKind.Arc) Then Return 0
        Return handle
    End Function
End Module
```

Related options and reference

ChangePathSegmentParameter

Purpose

Changes one named coordinate, Bezier control, arc radius or arc angle while keeping the object handle.

Declaration

```
Function ChangePathSegmentParameter(ByVal handle As Integer, ByVal figure As Integer, ByVal segment As
    Integer, ByVal name As String, ByVal value As Double) As Boolean
```

Parameters and result

Name	Meaning
handle	Object readable by GetCompoundPath.
figure, segment	Zero-based figure and segment indices. Segment 0 ends at node 1.
name	An exact key returned by GetPathSegmentParameters for this segment kind.
value	A finite Double. Coordinates/radii use drawing units; StartAngle/SweepAngle use degrees. Radii must be positive; arc sweeps must remain nonzero and within ±360 degrees.

Returns True for an accepted edit, False for invalid data or an object that cannot be edited. End-coordinate edits move the corresponding path node; Bezier controls and connected arcs are adjusted by the node-editing rules. Arc centre, radius and angle changes recalculate its start/end connections. A neighbouring arc may be reconstructed as a circular arc through its existing midpoint and updated endpoints. Radius edits rescale the corresponding axis vector while retaining its direction. Like ChangeCompoundPath, an accepted edit removes a parametric recipe and participates in Undo/Redo.

VB.NET example

```
Option Strict On
Public Module ChangePathSegmentParameterExample
    Public Function AddHigherTangent(cad As Avax.cAvax) As Integer
        Dim path As New Avax.AvaxPathData
        Dim figure As New Avax.AvaxPathFigure
        figure.Segments.Add(Avax.AvaxPathSegment.BezierTo(
            10.0, 20.0, 20.0, 20.0, 30.0, 0.0))
        path.Figures.Add(figure)
        Dim handle As Integer = cad.AddCompoundPath(path)
        If handle = 0 Then Return 0
        If Not cad.ChangePathSegmentParameter(
            handle, 0, 0, "Control1Y", 30.0) Then Return 0
        Return handle
    End Function
End Module
```

Related options and reference

ChangePointStyle

cAvax method

Purpose

Changes the point style of the point objects described in a specific handles array

VB.NET declaration

```
Function ChangePointStyle(ByRef h As Integer(), ByRef iPointStyle As Integer) As Boolean
```

Parameters

Name	Type and passing	Meaning
h	ByRef h As Integer()	One-based Integer() containing the target object handles. Element 0 is unused. Supply at least one valid handle; duplicate handles are accepted. The array is not rewritten.
iPointStyle	ByRef iPointStyle As Integer	Integer point-symbol style. -1 uses PenPointStyle; other negative values become 0. Values above 16 use the fallback symbol. The input variable is unchanged.

Use the declaration for the overload you call. ByRef arguments can receive outputs; declare variables and arrays with the exact indicated type. Optional defaults are shown in the declaration.

Return value

Boolean - True on success, false on error.

Remarks

Changes the symbol style of Point_i objects in the array. Other object types are skipped, including text and point-shaped polygons. Use -1 to inherit PenPointStyle; other negative values become 0. Positive Integer values are retained without byte truncation. Styles above 16 use the diagonal-cross fallback, as with PenPointStyle. Supply a one-based Integer() handle array: index 0 is unused. The control must be started and enabled. All handles are checked before any object is changed; a missing handle or invalid input returns False and leaves the batch unchanged. Duplicate handles are processed once. Successful edits retain handles and the current selection and create one Change undo operation, subject to ContinueUndo and UndoRecorderOn. The ByRef inputs are not rewritten. An unchanged or inapplicable batch returns True without creating an undo entry or Change event. Stored geometry, object metadata and document content remain unchanged.

Numeric changes from v2

Update the following variables, arrays or event-data receivers when porting the corresponding v2 call. The declarations above show the current overloads.

Parameter or value	v2 type	v3 type
iPointStyle	Short	Integer

For ByRef parameters, use a variable with the exact v3 type. Preserve array dimensions and the documented indexing. Moving from v2 to v3 explains conversion and legacy file-storage rules.

Example

Setup

Add an Avax.cAvax Windows Forms control named `cAvax1`. Paste the procedures into the form class. Example setup and supporting files explains references, initialization and file paths.

Add these Windows Forms controls or components to the form. Fields referenced by Handles clauses must be declared WithEvents, as the Visual Studio designer normally does.

Name	Type
Command1	Button
Label1	Label

VB.NET example

```
Dim lPointHandle As Integer
Dim iPointStyle As Integer
Private Sub Form1_Load(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs) Handles
    MyBase.Load
    Command1.Text = "ChangePointStyle"
    cAvax1.StartAvax()
    Call CAvax1.SetAvaxProperty(Avax.AvaxProp.PointRadius,n, 5.0!)
    iPointStyle = 0
    Label1.Text = "Point Type =" & Str(iPointStyle)
    lPointHandle = CAvax1.Add_Point(10, 10, 0, Nothing, -1, -1, iPointStyle, -1)
End Sub
Private Sub Form1_FormClosed(ByVal eventSender As System.Object, ByVal eventArgs As
    System.Windows.Forms.FormClosedEventArgs) Handles Me.FormClosed
    cAvax1.EndAvax()
End Sub
Private Sub Command1_Click(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs)
    Handles Command1.Click
    Dim h(1) As Integer
    iPointStyle = iPointStyle + 1
    If iPointStyle > 17 Then iPointStyle = 0
    h(1) = lPointHandle
    If cAvax1.ChangePointStyle(h, iPointStyle) = True Then
        Label1.Text = "Point Type =" & Str(iPointStyle)
    End If
End Sub
```

ChangeProperties

cAvax method

Purpose

Updates one existing object from a String metadata array and its per-type Double geometry record, retaining handle, type and key.

VB.NET declaration

```
Function ChangeProperties(ByRef avHandle As Integer, ByRef sProp As String(), ByRef xyz As Double()) As
    Boolean
```

Parameters

Name	Type and passing	Meaning
avHandle	ByRef avHandle As Integer	Existing object handle, unchanged on success.
sProp	ByRef sProp As String()	One-based metadata String() copied by the control; Nothing clears metadata.
xyz	ByRef xyz As Double()	Per-item Double() geometry described in the Property geometry table. Data_i ignores this argument; all other types require complete finite geometry.

Use the declaration for the overload you call. ByRef arguments can receive outputs; declare variables and arrays with the exact indicated type. Optional defaults are shown in the declaration.

Return value

Boolean - True on success, false on error.

Remarks

sProp is a copied one-based String() of application metadata; Nothing supplies no metadata. xyz is the complete per-type Double() geometry record returned by GetProperties. Use Property geometry layouts and retain the object type. This method uses the same validation and commit rules as SetProperties for one handle.

A started, enabled control is required. Invalid input returns False before editing. Success retains the handle and key and creates one Undo operation with Change. Application event-handler exceptions propagate after committed editing.

Property geometry layouts

Numeric changes from v2

Update the following variables, arrays or event-data receivers when porting the corresponding v2 call. The declarations above show the current overloads.

Parameter or value	v2 type	v3 type
xyz	Single()	Double()

For ByRef parameters, use a variable with the exact v3 type. Preserve array dimensions and the documented indexing. Moving from v2 to v3 explains conversion and legacy file-storage rules.

The Object() containers keep their declared type. Drawing-coordinate arrays stored inside them use Double(); use the documented per-object layout when reading or replacing those arrays.

Example

Setup

Add an Avax.cAvax Windows Forms control named `cAvax1`. Paste the procedures into the form class. Example setup and supporting files explains references, initialization and file paths.

Add these Windows Forms controls or components to the form. Fields referenced by Handles clauses must be declared WithEvents, as the Visual Studio designer normally does.

Name	Type
Button1	Button

VB.NET example

```
Dim avHandle As Integer
Private Sub Form1_Load(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs) Handles
    MyBase.Load
    Call CAvax1.StartAvax()
    avHandle = CAvax1.Add_Line(0, 0, 0, 10, 10, 0)
End Sub
Private Sub Form1_FormClosed(ByVal eventSender As System.Object, ByVal eventArgs As
    System.Windows.Forms.FormClosedEventArgs) Handles Me.FormClosed
    Call cAvax1.EndAvax()
End Sub
Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
    Button1.Click
    Dim sProp() As String = Nothing
    Dim xyz() As Double = Nothing
    Dim Answer As Boolean
    Dim avHandles(1) As Integer
    Dim iType() As Avax.AvaxItemType = Nothing
    Dim vItemProperties() As Object = Nothing
    Dim vData() As Object = Nothing
    avHandles(1) = avHandle
    CAvax1.GetProperties(avHandles, iType, vItemProperties, vData)
    xyz = vData(1)
    sProp = vItemProperties(1)
    xyz(1) = 2
    xyz(2) = 5
    xyz(3) = 0
    xyz(4) = 13
    xyz(5) = 8
    xyz(6) = 0
    Answer = CAvax1.ChangeProperties(avHandle, sProp, xyz)
End Sub
```

ChangeShapeClosed

Purpose

Sets whether a shape path has an implicit closing edge, and establishes a state readable through GetShapeClosed.

VB.NET declaration

```
Function ChangeShapeClosed(ByVal handle As Integer, ByVal closed As Boolean) As Boolean
```

Parameters

Parameter	Meaning
handle	Integer handle of a native shape path. Embedded controls and non-shape objects are rejected.
closed	True enables implicit closure; False leaves the path open.

Return value and behavior

Returns True on success. The original vertices are retained: identical first and last vertices can still form a visible loop with closed=False. A real state change forms one Undo/Redo action. Repeating an already established state does not add an edit. Returns False on an unavailable/disabled control or invalid object, preserving embedded-control identity flags.

VB.NET example

Pass a started control on its owning UI thread. This is a complete standalone module.

```
Option Strict On
Imports Avax

Public Module EditableShapeCloseExample
    Public Function ClosePath(cad As cAvax, id As Integer) As Boolean
        Return cad.ChangeShapeClosed(id, True)
    End Function
End Module
```

Related options and reference

ChangeShapeControlPoint

Purpose

Moves a control point and rebuilds the affected shape or path. For a parametric shape the position changes its corresponding recipe value; for a compound path it changes the node, Bezier handle or arc bend.

Declaration

```
Function ChangeShapeControlPoint(ByVal handle As Integer, ByVal id As String, ByVal x As Double, ByVal y
    As Double) As Boolean
```

Parameters and result

Name	Meaning
handle	Integer handle of the object to edit.
id	Case-sensitive control point identifier returned by GetShapeControlPoints for this object.
x, y	Requested finite Double position in world coordinates.

Returns True for an accepted edit, False when the point or resulting geometry is invalid or the control cannot edit it. A started, enabled control with LockEdit=False is required, and the object's layer must be visible and unlocked. The handle is retained and the accepted edit participates in normal Undo/Redo.

For preset shapes, the requested world position is mapped into the shape's local frame and converted to the associated parameter. The parameter is clamped to its catalog range. A parameter handle follows its defined locus, so its final position need not equal the exact coordinates requested by the caller. By comparison, a direct ChangeShapeParameter call rejects values outside its range. For a path, this method edits the node, Bezier control or arc bend; it must produce valid geometry. Read the control points again after editing to display their actual updated positions.

VB.NET example

```
Option Strict On
Public Module ChangeShapeControlPointExample
    Public Function NarrowStarCentre(cad As Avax.cAvax) As Boolean
        Dim handle As Integer = cad.AddParametricShape("Star", 0.0, 0.0, 100.0, 80.0)
        If handle = 0 Then Return False
        For Each point As Avax.AvaxShapeControlPoint In cad.GetShapeControlPoints(handle)
            If point.Id = "InnerRatio" Then
                Dim x As Double = 50.0 + (point.X - 50.0) * 0.8
                Dim y As Double = 40.0 + (point.Y - 40.0) * 0.8
                Return cad.ChangeShapeControlPoint(handle, point.Id, x, y)
            End If
        Next
        Return False
    End Function
End Module
```

Related options and reference

ChangeShapeDocumentFontSize

Changes the font height of a document shape and reconstructs its frame from the resulting layout. It preserves the shape's insertion point, Z coordinate and rotation. It accepts Shape_i with a Document_f fill; ordinary text, text shapes and other shapes are rejected.

```
Public Function ChangeShapeDocumentFontSize(ByRef avHandleDocument As Integer,
                                            ByRef DocumentFontSize As Double) As Boolean
```

Related options and reference

ChangeShapeDocumentWidth

Changes a document shape's logical wrapping width and reconstructs its frame to fit the new width and content height. Its insertion point, Z coordinate and rotation remain unchanged. Use this method when the supplied handle is specifically a document shape.

```
Public Function ChangeShapeDocumentWidth(ByRef avHandleDocument As Integer,
                                         ByRef DocumentWidth As Double) As Boolean
```

Parameters and result for the four change methods

Parameter	Meaning
avHandleDocument	An existing handle of the type accepted by the method. The handle variable and object handle are retained.
DocumentFontSize	Finite Double font height greater than zero. The input variable is unchanged.
DocumentWidth	Finite Double wrapping width greater than zero. The input variable is unchanged.

The control must be started and enabled. The result is True on success, including when the requested value is already set. A missing or incorrect object type, invalid dimension or layout overflow returns False without modifying the object or undo history. A successful change creates one Change undo operation, subject to ContinueUndo and UndoRecorderOn. It retains the current selection, keys, custom properties, pen attributes, fill options, editable text and embedded document resources. Each shape size change rebuilds its rectangular layout frame, as in R2; it is not a stretch of independently edited frame corners.

Plain text, AVAX document markup and RTF content remain editable. Explicit run formatting and images are preserved. The ordinary Document_i transformation matrix, including rotation, reflection and shear, is not rewritten by either dimension setter.

Related options and reference

ChangeShapeFill

cAvax method

Purpose

Use this method if you want to change the fill properties of a shape

Text and document frames

Text, RTF and document fills follow the shape's stored frame, including rotation, shear and reflection. Changing a document fill reflows the content while retaining the existing frame axes and corner elevations. Changing only colors, flags or drawing order preserves the frame. Content placed outside a document's nominal margins remains included in its drawing and selection bounds.

Native AVAX files retain Double frame coordinates. AVX and XML exports preserve the R2 corner convention; reopening the drawing retains the editable frame and formatted content.

VB.NET declaration

```
Function ChangeShapeFill(ByRef hShape As Integer, ByRef Fill_iType As Byte, ByRef Fill_lHatchIndex As
    Integer, ByRef Fill_iInForeColor As Integer, ByRef Fill_iInBackColor As Integer, ByRef Fill_sPattern As
    String, ByRef Fill_zOrder As Integer, ByRef Fill_lFlags As Integer, Optional ByRef sScaleX As Double = 1,
    Optional ByRef sScaleY As Double = 1) As Boolean
```

Parameters

Name	Type and passing	Meaning
hShape	ByRef hShape As Integer	Input handle of a shape, polyline or ellipse. It remains unchanged. Polyline/ellipse objects become shapes on success.
Fill_iType	ByRef Fill_iType As Byte	Byte fill family code from AvaxFillType. The legacy mapped-image codes 254 and 255 are also accepted; undefined codes are rejected.
Fill_lHatchIndex	ByRef Fill_lHatchIndex As Integer	Integer hatch/style index belonging to the selected fill family.
Fill_iInForeColor	ByRef Fill_iInForeColor As Integer	Integer foreground color. For text shapes this also becomes the text/outline color. Promoted from Integer in v2.
Fill_iInBackColor	ByRef Fill_iInBackColor As Integer	Integer background color used by the selected fill family and flags. Promoted from Integer in v2.
Fill_sPattern	ByRef Fill_sPattern As String	Family-specific pattern: opaque encoded text/hatch data, SVG/control markup, or an image/metafile path/resource name. Use the matching creation method or GetShapeFill. Solid_f also accepts <Gradient:firstColor,lastColor[,angle]> with OLE colors and degrees (0 vertical, 90 horizontal). Solid_f/Pattern_f accept <OpacityPercentN>; the two decorations can be combined. Percentages are clamped to 0..100. The descriptor is retained unchanged.
Fill_zOrder	ByRef Fill_zOrder As Integer	Signed Integer stacking order; higher values draw in front when ShapeZorderOn_p is enabled.
Fill_lFlags	ByRef Fill_lFlags As Integer	Fill option bits, including border/background/stretch/polyline options from CreatePatternFlags. Their effect depends on the fill family.
sScaleX	Optional ByRef sScaleX As Double = 1	Optional finite Double X scale about the first vertex; defaults to 1. Negative values reflect X. This parameter was Double in v2.
sScaleY	Optional ByRef sScaleY As Double = 1	Optional finite Double Y scale about the first vertex; defaults to 1. Negative values reflect Y. This parameter was Double in v2.

Use the declaration for the overload you call. ByRef arguments can receive outputs; declare variables and arrays with the exact indicated type. Optional defaults are shown in the declaration.

Return value

Boolean - True on success, false on error.

Remarks

Replaces the fill descriptor of a Shape_i. Polyline_i and Ellipse_i are also accepted and converted to Shape_i, as in R2. Other item types return False. The control must be started and enabled. Success returns True and retains the existing handle. None of the ByRef input variables are changed. Fill_iType selects the AvaxFillType family. The descriptor includes its hatch index, foreground/background colors, pattern data, signed Integer zOrder and flags. Preserve opaque pattern bytes returned by GetShapeFill. Undefined family values are rejected. An identical descriptor with unit scale succeeds without adding undo history.

sScaleX and sScaleY are Double in v3 (Single in R2). They scale the shape's X and Y coordinates around its first vertex; they are not hatch-spacing settings. Each vertex retains its Z coordinate. Negative values reflect the corresponding axis. Zero can collapse ordinary polygon geometry, but a text/document fill cannot have a collapsed required dimension. Nonfinite factors or results outside the finite Double range fail before modifying the drawing. Text_f decodes the supplied text/font/style pattern. When a polygon becomes a text shape, the first-to-second corner edge defines its height and the first-to-fourth corner edge defines its width, as in R2. When an existing text pattern changes, its frame convention, shear and reflection are retained. Text is remeasured using the existing frame height divided over the new text lines. Document_f uses the document dimensions in its pattern and reflows the content along the existing frame axes, retaining shear and reflection. Both preserve the target insertion point, orientation and corner Z values. A color/order-only change using the current text pattern preserves the exact existing frame. Scaling is then applied to the prepared geometry. Unicode, final CR characters and custom imported font weights are retained. Switching to a nontext fill removes obsolete editable text and layout data. BMP_f and WMF_f accept a local image/metafile path or a matching already embedded resource name. Image bytes are validated and retained with the shape, so its source file need not remain present. Image fills are clipped to the polygon; ordinary mapped pictures retain their image mapping. Switching away from an image fill removes its old image payload. Invalid image, vector/PAT/AVP/SVG/control or text-pattern preparation returns False, reports an error without partially changing the object. Geometry, resources and the new descriptor are prepared before a single Change undo operation, subject to ContinueUndo and UndoRecorderOn. Keys, custom properties, selection, outline attributes and the handle are retained, except that text-shape foreground changes also update the text color as in R2. Undo/Redo includes type conversions and owned resources. No NewUserItems event is raised for this edit; subsequent object creation raises it normally. Solid_f accepts <Gradient:firstColor,lastColor[,angle]> in Fill_sPattern. Colors are OLE/COLORREF integers, not palette indexes. The angle is in degrees: 0 (the default) runs from the first color at the top to the last at the bottom; 90 runs left to right. Other finite angles are supported. Equal endpoints produce a solid color, including black. The gradient is clipped to the closed polygon. Open/polyline flags suppress its filled interior. Solid_f and Pattern_f accept a leading <OpacityPercentN>, with a finite percentage clamped to 0 through 100. Zero is fully transparent. The fill, pattern background and visible outline are composited together at that opacity. Pattern bytes following the prefix retain their normal meaning. Gradient and opacity prefixes may be combined in either order. For example, Fill_sPattern = "<OpacityPercent50><Gradient:255,16711680,90>" with Solid_f gives a 50-percent-opaque red-to-blue horizontal gradient. The stored pattern and all ByRef inputs remain unchanged by rendering. Malformed or duplicate prefixes fail before editing. Native AVAX, legacy AVX and R2 XML retain these decorations. Bitmap output and the offscreen polygon cache use the same fill semantics. DXF/DWG solid-fill export cannot represent these gradients or partial opacity; it returns False and preserves an existing destination instead of silently writing an opaque solid.

Numeric changes from v2

Update the following variables, arrays or event-data receivers when porting the corresponding v2 call. The declarations above show the current overloads.

Parameter or value	v2 type	v3 type
Fill_iInForeColor	Short	Integer
Fill_iInBackColor	Short	Integer
sScaleX	Single	Double
sScaleY	Single	Double

For ByRef parameters, use a variable with the exact v3 type. Preserve array dimensions and the documented indexing. Moving from v2 to v3 explains conversion and legacy file-storage rules.

Example

Setup

Add an Avax.cAvax Windows Forms control named `cAvax1`. Paste the procedures into the form class. Example setup and supporting files explains references, initialization and file paths.

Add these Windows Forms controls or components to the form. Fields referenced by Handles clauses must be declared WithEvents, as the Visual Studio designer normally does.

Name	Type
Button1	Button

VB.NET example

```
Dim AvaxHandle As Integer
Private Sub Form1_Load(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs) Handles
    MyBase.Load
    Call cAvax1.StartAvax()
    AddShape()
End Sub
Private Sub Form1_FormClosed(ByVal eventSender As System.Object, ByVal eventArgs As
    System.Windows.Forms.FormClosedEventArgs) Handles Me.FormClosed
    Call cAvax1.EndAvax()
End Sub
Private Sub AddShape()
    Dim xV() As Double = Nothing
    Dim yV() As Double = Nothing
    Dim zV() As Double = Nothing
    Dim vItemProperties As Object = Nothing
    Dim Fill iType As Byte
    Dim Fill lHatchIndex As Integer
    Dim Fill iInForeColor As Integer
    Dim Fill iInBackColor As Integer
    Dim Fill sPattern As String = String.Empty
    Dim Fill zOrder As Integer
    Dim Fill lFlags As Integer
    Dim iClr As Integer
    Dim iWidth As Integer
    Dim iStyle As Integer
    Dim iLayer As Integer
    Dim PolylineType As Avax.AvaxPolylineType
    ReDim xV(5)
    ReDim yV(5)
    ReDim zV(5)
    xV(1) = 2
    yV(1) = 2
    xV(2) = 2
    yV(2) = 12
    xV(3) = 12
    yV(3) = 12
    xV(4) = 12
    yV(4) = 2
    xV(5) = 10
    yV(5) = -1
    Fill iType = Avax.AvaxFillType.Pattern_f
    Fill lHatchIndex = 2
    iClr = 1
    iWidth = 2
    Fill iInBackColor = 4
    AvaxHandle = cAvax1.Add_Shape(xV, yV, zV, vItemProperties, Fill iType, Fill lHatchIndex,
    Fill iInForeColor, Fill iInBackColor, Fill sPattern, Fill zOrder, Fill lFlags, iClr, iWidth, iStyle,
    iLayer, PolylineType)
End Sub
Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
    Button1.Click
```

```
    Dim Fill iType As Byte
    Dim Fill lHatchIndex As Integer
    Dim Fill iInForeColor As Integer
    Dim Fill iInBackColor As Integer
    Dim Fill sPattern As String = String.Empty
    Dim Fill zOrder As Integer
    Dim Fill lFlags As Integer
    Dim Answer As Boolean
    Fill.iType = Avax.AvaxFillType.Pattern_f
    Fill.lHatchIndex = 1
    Fill.iInBackColor = 3
    Answer = CAvax1.ChangeShapeFill(AvaxHandle, Fill.iType, Fill.lHatchIndex, Fill.iInForeColor,
    Fill.iInBackColor, Fill.sPattern, Fill.zOrder, Fill.lFlags)
End Sub
```

ChangeShapeParameter

Purpose

Changes one supported dimension and regenerates the actual native outline, while retaining the shape handle and other object attributes.

VB.NET declaration

```
Function ChangeShapeParameter(ByVal handle As Integer, ByVal name As String, ByVal value As Double) As
    Boolean
```

Parameters

Parameter	Meaning
handle	Integer handle of a shape that currently exposes parameters.
name	Case-sensitive key returned by GetShapeParameters.
value	Finite Double replacement value. See the parameter rules below.

Return value and behavior

Returns True on success, including an unchanged value. Unchanged values do not add an edit. Other successful changes form one Undo/Redo action. Returns False for an unavailable/disabled control, invalid handle, unsupported key or invalid value. Validation failure preserves the original geometry and revision; Preparation errors are reported through the ErrorMessage event.

Parameter names

Shape	Keys
Rectangle	ShapeWidth, ShapeHeight, CornerRadiusX, CornerRadiusY, CornerRadius. CornerRadius is NaN when the two radii differ.
Circle / ellipse	Radius for equal orthogonal axes; otherwise RadiusX and RadiusY.
Polygon	Radius or RadiusX/RadiusY, plus Vertices.
Arc / pie	Radius or RadiusX/RadiusY, plus StartAngle and SweepAngle.

Widths, heights and radii must be positive. Corner radii range from zero to half their corresponding side; a common CornerRadius cannot exceed half the shorter side. Both corner radii must be positive to round the corners. Vertices must be an integer from 3 to 10000. StartAngle is normalized to [0, 360); SweepAngle must be greater than zero and at most 360. Rotated, reflected and unevenly scaled conics retain editable principal-axis radii when their metadata still matches the native outline.

VB.NET example

Pass a started control on its owning UI thread. This is a complete standalone module.

```
Option Strict On
Imports Avax

Public Module EditableShapeParameterChangeExample
    Public Function Widen(cad As cAvax, id As Integer) As Boolean
        Return cad.ChangeShapeParameter(id, "ShapeWidth", 100.0)
    End Function
End Module
```

Related options and reference

ChangeShapeText

cAvax method

Purpose

Changes content and optional formatting of a text-filled shape, retaining and returning its existing handle on success.

Text and document frames

Text, RTF and document fills follow the shape's stored frame, including rotation, shear and reflection. Changing a document fill reflows the content while retaining the existing frame axes and corner elevations. Changing only colors, flags or drawing order preserves the frame. Content placed outside a document's nominal margins remains included in its drawing and selection bounds.

Native AVAX files retain Double frame coordinates. AVX and XML exports preserve the R2 corner convention; reopening the drawing retains the editable frame and formatted content.

VB.NET declarations

```
Function ChangeShapeText(ByRef hTextShape As Integer, ByRef sNewText As String, ByRef sFontName As String,
     ByRef fBold As Boolean, ByRef fItalic As Boolean, ByRef fUnderline As Boolean, ByRef fStrikeOut As
    Boolean, ByRef lCharSet As Integer, ByRef fForeColor As Integer, ByRef fBackColor As Integer, ByRef
    TxtHeight As Double, ByRef TxtWidth As Double) As Integer

Function ChangeShapeText(ByRef hTextShape As Integer, ByRef sNewText As String) As Integer

Function ChangeShapeText(ByRef hTextShape As Integer, ByRef sNewText As String, ByRef sFontName As String,
     ByRef fBold As Boolean, ByRef fItalic As Boolean, ByRef fUnderline As Boolean, ByRef fStrikeOut As
    Boolean, ByRef lCharSet As Integer) As Integer
```

Parameters

Name	Type and passing	Meaning
hTextShape	ByRef hTextShape As Integer	Integer handle of an existing text-filled Shape_i object. The returned handle is unchanged on success.
sNewText	ByRef sNewText As String	Replacement text. Unicode and original line endings are retained.
sFontName	ByRef sFontName As String	Replacement named font. Empty or whitespace-only selects current FontName and is returned through ByRef on success. Null characters and line breaks are invalid. The two-argument overload also selects the current FontName.
fBold	ByRef fBold As Boolean	Replacement bold flag. False clears bold; the two-argument overload supplies False for all four style flags.
fItalic	ByRef fItalic As Boolean	Replacement italic flag.
fUnderline	ByRef fUnderline As Boolean	Replacement underline flag.
fStrikeOut	ByRef fStrikeOut As Boolean	Replacement strikeout flag.
lCharSet	ByRef lCharSet As Integer	Integer charset 0..255, or -1 to use current FontCharset. The input variable remains unchanged; the stored shape receives the resolved charset.
fForeColor	ByRef fForeColor As Integer	Integer foreground color; -1 retains existing ink. Explicit colors update visible text ink.
fBackColor	ByRef fBackColor As Integer	Integer background descriptor color; -1 retains it. As in the R2 Text_f path, this field does not fill the text frame. Text-excess background settings are separate.
TxtHeight	ByRef TxtHeight As Double	Double font height. Zero retains existing height; positive sets height. Negative/nonfinite values reject the whole edit.
TxtWidth	ByRef TxtWidth As Double	Double text width. Zero remeasures new text while retaining horizontal stretch; positive sets an explicit width.

Use the declaration for the overload you call. ByRef arguments can receive outputs; declare variables and arrays with the exact indicated type. Optional defaults are shown in the declaration.

Return value

Integer - The same existing shape handle on success, or 0 when the change cannot be made.

Remarks

The handle must identify a text-filled Shape_i. The same handle is returned on success; another object type or an unavailable control returns 0. The two-argument overload uses the current FontName and clears the four style flags. The full overload controls font, styles, charset, colors, text height and text width. A zero height retains the existing height. A zero width measures the replacement text while retaining its horizontal stretch factor; -1 colors retain the corresponding descriptors. An empty font name selects FontName and the resolved name is returned through ByRef.

The frame resizes to the new text while retaining its orientation, shear, reflection and corner elevations. Layer, key and application metadata remain attached to the same handle. A complete valid layout is prepared before replacement. Invalid font names or layout data return 0 without a partial edit or Undo operation. An unchanged request returns the handle without adding history; a successful change is one Undo operation with Change and no NewUserItems. Application event-handler exceptions propagate after committed editing.

Numeric changes from v2

Update the following variables, arrays or event-data receivers when porting the corresponding v2 call. The declarations above show the current overloads.

Parameter or value	v2 type	v3 type
fForeColor	Short	Integer
fBackColor	Short	Integer
TxtHeight	Single	Double
TxtWidth	Single	Double

For ByRef parameters, use a variable with the exact v3 type. Preserve array dimensions and the documented indexing. Moving from v2 to v3 explains conversion and legacy file-storage rules.

Example

Setup

Add an Avax.cAvax Windows Forms control named `cAvax1`. Paste the procedures into the form class. Example setup and supporting files explains references, initialization and file paths.

Add these Windows Forms controls or components to the form. Fields referenced by Handles clauses must be declared WithEvents, as the Visual Studio designer normally does.

Name	Type
Command1	Button

VB.NET example

```
Dim hTextShape As Integer
Private Sub Form1_Load(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs) Handles
    MyBase.Load
    cAvax1.StartAvax()
    Command1.Text = "ChangeShapeText"
    hTextShape = CAvax1.Add_ShapeText("This is the old text", 0, 5, 5.0!)
End Sub
Private Sub Form1_FormClosed(ByVal eventSender As System.Object, ByVal eventArgs As
    System.Windows.Forms.FormClosedEventArgs) Handles Me.FormClosed
    cAvax1.EndAvax()
End Sub
Private Sub Command1_Click(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs)
    Handles Command1.Click
```

```
    Dim sNewText As String = String.Empty
    Dim sFontName As String = String.Empty
    Dim fBold As Boolean
    Dim fItalic As Boolean
    Dim fUnderline As Boolean
    Dim fStrikeOut As Boolean
    Dim lCharSet As Integer
    sNewText = "This is the new text"
    fItalic = True
    fUnderline = True
    hTextShape = cAvax1.ChangeShapeText(hTextShape, sNewText, sFontName, fBold, fItalic, fUnderline,
    fStrikeOut, lCharSet)
End Sub
```

ChangeShapezOrder

cAvax method

Purpose

Changes the z order of the shape object with a specific handle

VB.NET declaration

```
Function ChangeShapezOrder(ByRef hShape As Integer, ByRef zOrder As Integer) As Integer
```

Parameters

Name	Type and passing	Meaning
hShape	ByRef hShape As Integer	Input handle of a shape, polyline or ellipse. Success returns the same handle; failure returns zero. The input variable is unchanged.
zOrder	ByRef zOrder As Integer	Requested signed Integer stacking order. Larger values appear in front when ShapeZorderOn_p is enabled. The input variable is unchanged.

Use the declaration for the overload you call. ByRef arguments can receive outputs; declare variables and arrays with the exact indicated type. Optional defaults are shown in the declaration.

Return value

Integer - An Avax handle on success, zero on error.

Remarks

Changes the stacking order of an existing shape. Returns its existing handle on success, or zero on failure. Both ByRef input variables remain unchanged. The control must be started and enabled. zOrder accepts the full signed Integer range. Higher values draw in front when ShapeZorderOn_p is True. Existing shapes retain their exact vertices, text/document frames, other fill fields, outline attributes, custom properties, keys, selection and embedded resources. The method does not rebuild a text frame or reopen its image files. Repeating the same order succeeds without creating another undo entry. For R2 compatibility, a Polyline_i or Ellipse_i is converted to an unfilled Shape_i with the requested order. Ellipses are tessellated from their actual transformed geometry, preserving rotation and Z. Other item types are rejected. Converted objects retain the same handle. A changed object creates one Change undo operation, subject to ContinueUndo and UndoRecorderOn. Undo restores the previous order and, for converted objects, the original entity representation. No NewUserItems event is raised for the edit; later object creation continues to raise it normally. Invalid preparation leaves the original drawing and history intact.

Example

Setup

Add an Avax.cAvax Windows Forms control named `cAvax1`. Paste the procedures into the form class. Example setup and supporting files explains references, initialization and file paths.

Add these Windows Forms controls or components to the form. Fields referenced by Handles clauses must be declared WithEvents, as the Visual Studio designer normally does.

Name	Type
Command1	Button

VB.NET example

```
Dim h1 As Integer
Dim h2 As Integer
Private Sub Form1_Load(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs) Handles
    MyBase.Load
    Dim zOrder As Integer
    Command1.Text = "ChangeShapezOrder"
```

```
    cAvax1.StartAvax()
    zOrder = 0
    h1 = CAvax1.Add_ShapeRectangle(0, 0, 12, 5, Nothing, 1, 1, 1, 1, "", zOrder, 0, -1, -1, -1, -1)
    zOrder = 1
    h2 = CAvax1.Add_ShapeRectangle(1, 1, 12, 5, Nothing, 1, 3, 1, 5, "", zOrder, 0, -1, -1, -1, -1)
End Sub
Private Sub Form1_FormClosed(ByVal eventSender As System.Object, ByVal eventArgs As
    System.Windows.Forms.FormClosedEventArgs) Handles Me.FormClosed
    cAvax1.EndAvax()
End Sub
Private Sub Command1_Click(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs)
    Handles Command1.Click
    Dim zOrder As Integer
    zOrder = 1
    h1 = cAvax1.ChangeShapezOrder(h1, zOrder)
    zOrder = 0
    h2 = cAvax1.ChangeShapezOrder(h2, zOrder)
    Swap(h1, h2)
End Sub
Sub Swap(ByRef a As Integer, ByRef b As Integer)
    Dim c As Integer
    c = a
    a = b
    b = c
End Sub
```

ChangeStrokeWidth

Purpose

Sets a precise pen width on one or more items as one reversible operation. For a group, the new value is applied to the group and its children.

VB.NET declaration

```
Function ChangeStrokeWidth(ByVal entityHandles As Integer(), ByVal width As Double) As Boolean
```

Parameters

Parameter	Meaning
entityHandles	One-based Integer array of item handles, with at least one actual handle. Slot zero is reserved and ignored. Repeated handles are processed once; the caller's array is retained.
width	Finite Double from zero through Integer.MaxValue. Zero disables the drawing pen. Positive values enable an ordinary shape's outline and set its pen width.

Return value and behavior

Returns True after a successful change. Nothing, an array without an actual handle, an unknown handle or an out-of-range/non-finite width returns False before any item is changed. Preparation errors are reported through ErrorMessage. An inactive or disabled control returns False. If all addressed items and children already have the requested width, the call succeeds without adding an Undo record.

A changed batch produces one Undo/Redo action and one Change notification. The notification sees the complete batch; a newer edit or drawing replacement made by the callback is retained. An exception thrown by host event code can propagate after the change has committed. Geometry, selection, the current PenWidth setting, user properties and unrelated fill flags are retained. Native .avax saving retains the precise width. Calling the legacy ChangeWidth method replaces the precise override on the addressed record with an Integer width; its existing group behavior is retained.

Zero suppresses a line or shape outline while retaining a filled shape's interior. Text, image content and embedded-control border styles keep their own rendering rules. SVG output represents fractional widths and disabled pens; WMF/GDI output retains pen visibility subject to the destination's resolution and width quantization.

VB.NET example

Pass a started, enabled control and two valid item handles on its owning UI thread.

```
Option Strict On
Imports Avax
Imports System.Globalization

Public Module StrokeWidthChangeExample
    Public Function SetFineStroke(cad As cAvax, first As Integer, second As Integer) As Boolean
        Dim selected As Integer() = {0, first, second}
        Return cad.ChangeStrokeWidth(selected, 2.5R)
    End Function
End Module
```

Related options and reference

ChangeText

cAvax method

Purpose

Changes the text of a text object

VB.NET declarations

```
Function ChangeText(ByRef hTextItem As Integer, ByRef sNewText As String) As Integer

Function ChangeText(ByRef hTextItem As Integer, ByRef sNewText As String, ByRef fBold As Boolean, ByRef
    fItalic As Boolean, ByRef fUnderline As Boolean, ByRef fStrikeOut As Boolean, ByRef lCharSet As Integer)
    As Integer
```

Parameters

Name	Type and passing	Meaning
hTextItem	ByRef hTextItem As Integer	Existing Text_i or Document_i handle. Success returns this handle; the ByRef variable is unchanged.
sNewText	ByRef sNewText As String	New Unicode content, including any document markup or RTF. Nothing sets empty content. The caller string is unchanged.
fBold	ByRef fBold As Boolean	Requested bold style. The Integer overload uses False.
fItalic	ByRef fItalic As Boolean	Requested italic style. The Integer overload uses False.
fUnderline	ByRef fUnderline As Boolean	Requested underline style. The Integer overload uses False.
fStrikeOut	ByRef fStrikeOut As Boolean	Requested strikeout style. The Integer overload uses False.
lCharSet	ByRef lCharSet As Integer	Integer charset from 0 to 255, or -1 to use the current FontCharset. On success -1 is replaced with that resolved value. The Integer overload uses the current charset.

Use the declaration for the overload you call. ByRef arguments can receive outputs; declare variables and arrays with the exact indicated type. Optional defaults are shown in the declaration.

Return value

Integer - Is the Avax handle of the new text

Remarks

Changes the content of an ordinary Text_i or Document_i object. Text shapes require ChangeShapeText; other types return zero. The control must be started and enabled. Returns the same nonzero handle on success, including an unchanged assignment; failure returns zero. The ByRef handle and string variables are unchanged. Nothing is treated as empty content, and Unicode and the supplied line separators are preserved. The two-argument overload clears Bold, Italic, Underline and Strikeout and uses the current FontCharset, as in R2. The longer overload uses its four Boolean inputs. Its lCharSet must be -1 or 0 through 255; -1 resolves to the current FontCharset and is written back through lCharSet on success. The host font properties are not changed. Existing explicit RTF and markup run formatting remains part of the supplied content. Ordinary text width is remeasured using its effective font family. Position, Z, drawing height and rotation remain unchanged. A document retains its logical page width, font height and complete transformation matrix; its content reflows within that layout. The font-table binding or independently named imported font is retained. The handle, key, custom properties, pen attributes and selection are preserved. Existing embedded images are reused. Newly referenced document images are prepared before committing; an invalid resource or layout leaves the original object and undo history intact. Each actual change creates one Change undo operation, subject to ContinueUndo and UndoRecorderOn, and emits no NewUserItems event. A completely unchanged content/style/charset assignment creates no extra history.

Example

Setup

Add an Avax.cAvax Windows Forms control named `cAvax1`. Paste the procedures into the form class. Example setup and supporting files explains references, initialization and file paths.

Add these Windows Forms controls or components to the form. Fields referenced by Handles clauses must be declared WithEvents, as the Visual Studio designer normally does.

Name	Type
Command1	Button

VB.NET example

```
Dim hTextItem As Integer
Private Sub Form1_Load(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs) Handles
    MyBase.Load
    Command1.Text = "ChangeText"
    cAvax1.StartAvax()
    hTextItem = CAvax1.Add_Text("This is my text", 0, 5, 0, Nothing, 8.0!, 1, 0, -1, -1, -1, -1, False,
    False, False, False, -1)
End Sub
Private Sub Form1_FormClosed(ByVal eventSender As System.Object, ByVal eventArgs As
    System.Windows.Forms.FormClosedEventArgs) Handles Me.FormClosed
    cAvax1.EndAvax()
End Sub
Private Sub Command1_Click(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs)
    Handles Command1.Click
    Dim sNewText As String = String.Empty
    Dim fBold As Boolean
    Dim fItalic As Boolean
    Dim fUnderline As Boolean
    Dim fStrikeOut As Boolean
    Dim lCharSet As Integer
    sNewText = "This is my" & Chr(13) & "NEW text"
    fBold = True
    fUnderline = True
    hTextItem = cAvax1.ChangeText(hTextItem, sNewText, fBold, fItalic, fUnderline, fStrikeOut, lCharSet)
End Sub
```

ChangeWidth

cAvax method

Purpose

Changes the width of the objects described in a specific handles array

VB.NET declaration

```
Function ChangeWidth(ByRef h As Integer(), ByRef iWidth As Integer) As Boolean
```

Parameters

Name	Type and passing	Meaning
h	ByRef h As Integer()	One-based Integer() containing the target object handles. Element 0 is unused. Supply at least one valid handle; duplicate handles are accepted. The array is not rewritten.
iWidth	ByRef iWidth As Integer	Integer pen width. -1 uses PenWidth; other nonpositive values become 1. Positive values are retained at full Integer precision. The input variable is unchanged.

Use the declaration for the overload you call. ByRef arguments can receive outputs; declare variables and arrays with the exact indicated type. Optional defaults are shown in the declaration.

Return value

Boolean - True on success, false on error.

Remarks

Changes the pen width on each requested object record. Use -1 to inherit PenWidth. Other nonpositive values become 1, as in R2. Positive Integer widths are retained without the old byte or Short truncation. Ordinary screen widths are pixels; physical width settings can change display and print interpretation. Supply a one-based Integer() handle array: index 0 is unused. The control must be started and enabled. All handles are checked before any object is changed; a missing handle or invalid input returns False and leaves the batch unchanged. Duplicate handles are processed once. Successful edits retain handles and the current selection and create one Change undo operation, subject to ContinueUndo and UndoRecorderOn. The ByRef inputs are not rewritten. An unchanged or inapplicable batch returns True without creating an undo entry or Change event. Stored geometry, object metadata and document content remain unchanged.

Numeric changes from v2

Update the following variables, arrays or event-data receivers when porting the corresponding v2 call. The declarations above show the current overloads.

Parameter or value	v2 type	v3 type
iWidth	Short	Integer

For ByRef parameters, use a variable with the exact v3 type. Preserve array dimensions and the documented indexing. Moving from v2 to v3 explains conversion and legacy file-storage rules.

Example

Setup

Add an Avax.cAvax Windows Forms control named `cAvax1`. Paste the procedures into the form class. Example setup and supporting files explains references, initialization and file paths.

Add these Windows Forms controls or components to the form. Fields referenced by Handles clauses must be declared WithEvents, as the Visual Studio designer normally does.

Name	Type
Command1	Button

VB.NET example

```
Dim h() As Integer = Nothing
Private Sub Form1_Load(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs) Handles
    MyBase.Load
    cAvax1.StartAvax()
    ReDim h(1)
    h(1) = CAvax1.Add_Line(5, 1, 0, 5, 12, 0, Nothing, 6, 1, -1, -1)
End Sub
Private Sub Form1_FormClosed(ByVal eventSender As System.Object, ByVal eventArgs As
    System.Windows.Forms.FormClosedEventArgs) Handles Me.FormClosed
    cAvax1.EndAvax()
End Sub
Private Sub Command1_Click(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs)
    Handles Command1.Click
    Static iWidth As Integer
    Dim fOk As Boolean
    iWidth = iWidth + 1
    If iWidth > 15 Then iWidth = 0
    fOk = cAvax1.ChangeWidth(h, iWidth)
End Sub
```

ClearUndoRedoMemory

cAvax method

Purpose

Clears the undo/redo memory

VB.NET declaration

```
Function ClearUndoRedoMemory() As Boolean
```

Return value

Boolean - True on success, false on error.

Example

Setup

Add an Avax.cAvax Windows Forms control named `cAvax1`. Paste the procedures into the form class. Example setup and supporting files explains references, initialization and file paths.

Add these Windows Forms controls or components to the form. Fields referenced by Handles clauses must be declared WithEvents, as the Visual Studio designer normally does.

Name	Type
Command1	Button

VB.NET example

```
Private Sub Form1_Load(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs) Handles
    MyBase.Load
    Command1.Text = "ClearUndoRedoMemory"
    cAvax1.StartAvax()
    Call cAvax1.Add_Line(0, 0, 0, 10, 10, 0)
    Call cAvax1.Add_Line(0, 10, 0, 10, 10, 0)
End Sub
Private Sub Form1_FormClosed(ByVal eventSender As System.Object, ByVal eventArgs As
    System.Windows.Forms.FormClosedEventArgs) Handles Me.FormClosed
    cAvax1.EndAvax()
End Sub
Private Sub Command1_Click(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs)
    Handles Command1.Click
    Dim UndoType As Avax.AvaxUndo
    Console.WriteLine("Number of available Undo(s) before Clear=" & CAvax1.CurrentUndo(UndoType))
    If CAvax1.ClearUndoRedoMemory() = True Then
        Console.WriteLine("Number of available Undo(s) after Clear=" & CAvax1.CurrentUndo(UndoType))
    End If
End Sub
```

CombineCompoundPaths

Purpose

Combines two or more editable vector objects into one compound-path object. It can retain independent contours or join connected open paths into a single contour.

Declaration

```
Function CombineCompoundPaths(ByVal ids As Integer(), Optional ByVal joinConnected As Boolean = False,
    Optional ByVal tolerance As Double = 0.001) As Integer
```

Parameters and result

Name	Meaning
ids	The usual AVAX handle array: element 0 is reserved and ignored; valid handles start at element 1. Supply at least two distinct editable handles. Input order matters because the first object supplies the surviving handle and drawing attributes.
joinConnected	False retains all source figures in one object. True joins open, single-figure inputs whose endpoints meet within tolerance; a source figure may reverse or be prepended to make the chain.
tolerance	A finite, nonnegative distance in drawing units. Default 0.001. Used for endpoint matching when joinConnected is True.

Returns the surviving first handle, or 0 on failure. Accepted input objects after the first are removed; the result is selected. The operation is one Undo/Redo edit and removes the surviving object's parametric recipe. The control must be started, enabled and editable, and each source must be readable as a vector path in the XY plane.

With joinConnected=True, disconnected inputs or closed/multiple-figure inputs are rejected. Matched endpoints are moved to coincide; this is not just a visual tolerance. If the final chain endpoints meet within tolerance, the result is closed. With joinConnected=False, no geometric union or intersection is performed: the result uses EvenOddFill=True, so nested closed contours can form holes. The first object's drawing attributes apply to the complete result; separate source colours are not retained as individual styles.

VB.NET example

```
Option Strict On
Public Module CombineCompoundPathsExample
    Public Function AddJoinedPath(cad As Avax.cAvax) As Integer
        Dim first As New Avax.AvaxPathData
        Dim a As New Avax.AvaxPathFigure
        a.Segments.Add(Avax.AvaxPathSegment.LineTo(20.0, 0.0))
        first.Figures.Add(a)
        Dim second As New Avax.AvaxPathData
        Dim b As New Avax.AvaxPathFigure With {.X = 20.0, .Y = 0.0}
        b.Segments.Add(Avax.AvaxPathSegment.LineTo(20.0, 10.0))
        second.Figures.Add(b)
        Dim h1 As Integer = cad.AddCompoundPath(first)
        Dim h2 As Integer = cad.AddCompoundPath(second)
        If h1 = 0 OrElse h2 = 0 Then Return 0
        Dim ids As Integer() = {0, h1, h2}
        Return cad.CombineCompoundPaths(ids, True, 0.001)
    End Function
End Module
```

Related options and reference

ComparePolygons

cAvax method

Purpose

Classifies the secondary polygon relative to the main polygon in XY, returning inside, outside, crossing or external contact.

VB.NET declaration

```
Function ComparePolygons(ByRef hMain As Integer, ByRef hSec As Integer, ByRef iAns As AvaxPolygonsComp) As
     Boolean
```

Parameters

Name	Type and passing	Meaning
hMain	ByRef hMain As Integer	Is the handle of the first compared polygon
hSec	ByRef hSec As Integer	Is the handle of the second compared polygon
iAns	ByRef iAns As AvaxPolygonsComp	An expression that defines the result of the comparison

Use the declaration for the overload you call. ByRef arguments can receive outputs; declare variables and arrays with the exact indicated type. Optional defaults are shown in the declaration.

Return value

Boolean - True on success, false on error.

Remarks

For distinct handles, sources must be simple Shape_i polygons or connected closed Polyline_i contours. Shape vertices close implicitly; polyline segments must form an exact connected closed XY chain. Z, stroke width and fill style do not affect classification. Circles, ellipses, text, curves and blocks are not converted into polygons by this method.

In_p means the secondary contour is inside or on the main boundary. Out_p means it is outside without contact. Cross_p means it passes between inside and outside. ExtContact_p means it remains outside but touches the main boundary. The result is asymmetric: a secondary polygon surrounding the main polygon is outside, while a contained polygon touching its boundary remains inside. The same existing handle returns In_p.

A started, enabled control and nonzero handles are required; failure of those early guards leaves iAns unchanged. Other invalid-handle or contour failures return False with iAns=0. Degenerate, zero-area, self-crossing, backtracking or repeated-nonadjacent rings are rejected. The method does not change geometry, selection, history or command input.

Example

Setup

Add an Avax.cAvax Windows Forms control named `cAvax1`. Paste the procedures into the form class. Example setup and supporting files explains references, initialization and file paths.

Add these Windows Forms controls or components to the form. Fields referenced by Handles clauses must be declared WithEvents, as the Visual Studio designer normally does.

Name	Type
Command1	Button

VB.NET example

```
Dim hMain As Integer
Dim hSec As Integer
Private Sub Form1_Load(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs) Handles
    MyBase.Load
    cAvax1.StartAvax()
```

```
    hMain = cAvax1.Add_ShapeRectangle(0, 0, 10, 10, Nothing, 1, 0, 1, 1, "", 0, 0, -1, -1, -1, -1)
    hSec = cAvax1.Add_ShapeRectangle(5, 5, 4, 4, Nothing, 1, 0, 1, 2, "", 0, 0, -1, -1, -1, -1)
End Sub
Private Sub Form1_FormClosed(ByVal eventSender As System.Object, ByVal eventArgs As
    System.Windows.Forms.FormClosedEventArgs) Handles Me.FormClosed
    cAvax1.EndAvax()
End Sub
Private Sub Command1_Click(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs)
    Handles Command1.Click
    Dim sMsg As String = String.Empty
    Dim iAns As Avax.AvaxPolygonsComp
    If cAvax1.ComparePolygons(hMain, hSec, iAns) = True Then
        Select Case iAns
            Case Avax.AvaxPolygonsComp.Cross_p
            sMsg = "The yellow rectangle crosses the red one"
            Case Avax.AvaxPolygonsComp.ExtContact_p
            sMsg = "The yellow rectangle is in external contact with the red one"
            Case Avax.AvaxPolygonsComp.In_p
            sMsg = "The yellow rectangle is inside the red one"
            Case Avax.AvaxPolygonsComp.Out_p
            sMsg = "The yellow rectangle is outside the red one"
        End Select
        MsgBox(sMsg, MsgBoxStyle.Information, "Compare polygons")
    End If
End Sub
```

ConvertShapeToCurves

Purpose

Turns an individual vector curve or shape in the XY drawing plane into a directly editable compound path and removes its parametric-shape association. An existing compound path retains its segments. Supported analytic lines, cubic Bezier curves, arcs and ellipses retain their segment definitions; other usable vector outlines supply straight segments.

Declaration

```
Function ConvertShapeToCurves(ByVal handle As Integer) As Boolean
```

Parameters and result

`handle` identifies the object to convert. Returns True for an accepted edit, False if the control cannot edit the object or no usable outline is available. Bitmap images, text, dimensions, hatches, ordinary groups and geometry with nonzero Z are rejected. Connected analytic polyline boundaries are supported. The handle is retained. After converting a parametric shape, GetParametricShapeKind returns an empty string and edits use the path/control-point methods. Conversion participates in normal Undo/Redo.

VB.NET example

```
Option Strict On
Public Module ConvertShapeToCurvesExample
    Public Function AddEditableStar(cad As Avax.cAvax) As Integer
        Dim handle As Integer = cad.AddParametricShape(
            "Star", 0.0, 0.0, 100.0, 80.0)
        If handle = 0 OrElse Not cad.ConvertShapeToCurves(handle) Then Return 0
        System.Diagnostics.Debug.WriteLine(cad.GetParametricShapeKind(handle))
        Return handle
    End Function
End Module
```

Related options and reference

ConvertToShape

API reference

Purpose

Creates a filled shape from an existing drawing object. Use it to apply a fill to the outline of a point, line, polyline or ellipse, or to create a new shape from an existing shape. A wide polyline uses its centreline vertices; its stroke-width profile is not part of the new shape.

VB.NET declaration

```
Function ConvertToShape(ByVal hAvax As Integer, ByRef Fill_iType As Byte, ByRef Fill_lHatchIndex As
    Integer, ByRef Fill_iInForeColor As Integer, ByRef Fill_iInBackColor As Integer, ByRef Fill_sPattern As
    String, ByRef Fill_zOrder As Integer, ByRef Fill_lFlags As Integer, ByVal bDelete As Boolean, Optional
    ByVal fireNewItemsEvent As Boolean = True) As Integer
```

Parameters

Parameter	Meaning
hAvax	Handle of a point, line, polyline, shape or ellipse in the current drawing.
Fill_iType	Fill family. Use a value from AvaxFillType converted to Byte.
Fill_lHatchIndex	Pattern or image mapping index for the selected fill family.
Fill_iInForeColor	Foreground palette index.
Fill_iInBackColor	Background palette index.
Fill_sPattern	Pattern data, image path or embedded image resource name. Use an empty string for an ordinary solid fill.
Fill_zOrder	Signed Integer drawing order for the new shape.
Fill_lFlags	Fill and border flags. Zero uses the ordinary fill/border behavior.
bDelete	True removes the source only after the new shape is ready. False keeps both objects.
fireNewItemsEvent	True raises NewUserItems for the created shape. False suppresses this notification.

The ByRef fill arguments are inputs; this method leaves their values unchanged. V3 uses Integer for the color indices where v2 used Short. Drawing coordinates use Double. See AvaxFillType for the fill families.

Return value and behavior

Returns the new shape handle, or zero when the control is not started/enabled, the source is unsupported, the fill is invalid or the operation cannot be prepared. A failed call leaves the source, selection and Undo/Redo history unchanged, including when bDelete is True.

The result retains the source key, user properties, pen attributes, elevation and height. Vertex Z values are retained. Ellipses are converted to a sampled outline in their existing transformed position. Point and collinear objects keep their geometry; a fill has no area until the resulting outline encloses an area.

Image and R2 AAF animation bytes are validated and stored with the new shape before returning. A matching image already embedded in the drawing can also be reused. The original image file is no longer needed afterward. The complete creation and optional source deletion form one Undo operation; Undo restores the source, and Redo restores the conversion.

VB.NET example

This example creates a rectangle and replaces it with a solid-filled shape. Pass a started control on its owning UI thread. The returned Integer is the new shape handle.

```
Option Strict On
Imports Avax

Public Module ShapeConversionExample
    Public Function Run(cAvax1 As cAvax) As Integer
        ' The control must already have been started.
        Dim source As Integer = cAvax1.Add_Rectangle(10.0, 20.0, 80.0, 40.0)
```

```
        If source = 0 Then Return 0

        Dim fillType As Byte = CByte(AvaxFillType.Solid f)
        Dim hatchIndex As Integer = 0
        Dim foreground As Integer = 1
        Dim background As Integer = 0
        Dim pattern As String = ""
        Dim zOrder As Integer = 0
        Dim flags As Integer = 0

        Return cAvax1.ConvertToShape(source, fillType, hatchIndex,
            foreground, background, pattern, zOrder, flags,
            bDelete:=True, fireNewItemsEvent:=True)
    End Function
End Module
```

Copy_Avax

cAvax method

Purpose

Creates a Avax clipboard copy for the specific handles array

VB.NET declaration

```
Function Copy_Avax(ByRef avMyHandles As Integer(), ByRef xBase As Double, ByRef yBase As Double) As Byte()
```

Parameters

Name	Type and passing	Meaning
avMyHandles	ByRef avMyHandles As Integer()	Input Integer() of the entity handles to process. Logical entries start at index 1; index 0 is reserved.
xBase	ByRef xBase As Double	Is the x coordinate of the base point
yBase	ByRef yBase As Double	Is the y coordinate of the base point

Use the declaration for the overload you call. ByRef arguments can receive outputs; declare variables and arrays with the exact indicated type. Optional defaults are shown in the declaration.

Return value

An array of Bytes with the Avax clipboard data

Remarks

Returns a Byte array containing copies of the explicitly requested objects, relative to xBase and yBase. Pass a one-based Integer handle array: slot zero is ignored. Duplicate handles are copied once in first-occurrence order. Nothing, an empty array or an array containing only slot zero returns Nothing. Missing or deleted handles reject the complete request. The source objects, resources, selection, current command and caller arguments remain unchanged. Invalid base coordinates or recognized serialization failures return Nothing and report an error. This method does not use the Windows clipboard; pass its returned bytes to Paste_Avax. When representable, the payload contains the R2 clipboard format plus a native v3 extension that retains Double coordinates and complete resources. The R2 representation has Single coordinate and bounded record/attribute limits. For valid drawings that exceed those limits, the result is a lossless native v3 payload readable by v3 Paste_Avax; that fallback cannot be read by the R2 DLL. Plain text-filled shapes follow the current OriginTopLeft_p setting in the legacy representation, including text inside nested blocks. Use the same origin mode in the receiving R2 control to retain the displayed orientation. Exporting does not reorder the source drawing geometry.

Numeric changes from v2

Update the following variables, arrays or event-data receivers when porting the corresponding v2 call. The declarations above show the current overloads.

Parameter or value	v2 type	v3 type
xBase	Single	Double
yBase	Single	Double

For ByRef parameters, use a variable with the exact v3 type. Preserve array dimensions and the documented indexing. Moving from v2 to v3 explains conversion and legacy file-storage rules.

Example

Setup

Add an Avax.cAvax Windows Forms control named `cAvax1`. Paste the procedures into the form class. Example setup and supporting files explains references, initialization and file paths.

Add these Windows Forms controls or components to the form. Fields referenced by Handles clauses must be declared WithEvents, as the Visual Studio designer normally does.

Name	Type
Command1	Button

VB.NET example

```
Dim h(2) As Integer
Private Sub Form1_Load(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs) Handles
    MyBase.Load
    cAvax1.StartAvax()
    h(1) = cAvax1.Add_Line(0, 0, 0, 10, 10, 0)
    h(2) = cAvax1.Add_Line(0, 10, 0, 10, 0, 0)
End Sub
Private Sub Form1_FormClosed(ByVal eventSender As System.Object, ByVal eventArgs As
    System.Windows.Forms.FormClosedEventArgs) Handles Me.FormClosed
    cAvax1.EndAvax()
End Sub
Private Sub Command1_Click(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs)
    Handles Command1.Click
    Dim x As Double
    Dim y As Double
    Dim AvaxClipboard() As Byte = Nothing
    Dim vRetHandles As Object = Nothing
    Dim Answer As Boolean
    x = 5
    y = 5
    AvaxClipboard = cAvax1.Copy_Avax(h, x, y)
    x = 10
    y = 5
    Answer = cAvax1.Paste_Avax(x, y, AvaxClipboard, vRetHandles)
End Sub
```

CopyInside

cAvax method

Purpose

Creates inward offset copies of object boundaries at a distance in drawing units, retaining originals. Negative distances reverse direction.

VB.NET declarations

```
Function CopyInside(ByRef avMyHandles As Integer(), ByRef Offset As Double, ByRef vRetHandles As Object)
    As Boolean

Function CopyInside(ByRef avMyHandles As Integer(), ByRef Offset As Double) As Boolean
```

Parameters

Name	Type and passing	Meaning
avMyHandles	ByRef avMyHandles As Integer()	Input Integer() of the entity handles to process. Logical entries start at index 1; index 0 is reserved.
Offset	ByRef Offset As Double	Is the distance form the border of the object that copy will occur
vRetHandles	ByRef vRetHandles As Object	Returns an integer array with the handles of the new objects

Use the declaration for the overload you call. ByRef arguments can receive outputs; declare variables and arrays with the exact indicated type. Optional defaults are shown in the declaration.

Return value

Boolean - True on success, false on error.

Remarks

Pass a one-based Integer() of source handles and a finite Double distance. Negative distances reverse the named direction. The full overload returns a one-based Integer() of new handles in vRetHandles. Originals, attributes and height profiles are retained.

Polygon offsets retain mitered corners. An inward region can split into several contours or disappear; no remaining result returns False. Circles and arcs retain analytic geometry. Ellipse offsets change the principal semiaxes; an inward distance reaching the minor semiaxis produces no copy. Wide polylines retain the surviving sides of their width profiles.

A started, enabled control is required. The complete result is prepared before one Undo operation, with Change and no NewUserItems. Invalid sources or unrepresentable output return False without a partial edit or caller-output change. Application event-handler exceptions propagate after committed insertion.

Numeric changes from v2

Update the following variables, arrays or event-data receivers when porting the corresponding v2 call. The declarations above show the current overloads.

Parameter or value	v2 type	v3 type
Offset	Single	Double

For ByRef parameters, use a variable with the exact v3 type. Preserve array dimensions and the documented indexing. Moving from v2 to v3 explains conversion and legacy file-storage rules.

Example

Setup

Add an Avax.cAvax Windows Forms control named `cAvax1`. Paste the procedures into the form class. Example setup and supporting files explains references, initialization and file paths.

Add these Windows Forms controls or components to the form. Fields referenced by Handles clauses must be declared WithEvents, as the Visual Studio designer normally does.

Name	Type
Button1	Button

VB.NET example

```
Dim AvaxHandle As Integer
Private Sub Form1_Load(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs) Handles
    MyBase.Load
    Call CAvax1.StartAvax()
    AddPolyline()
End Sub
Private Sub Form1_FormClosed(ByVal eventSender As System.Object, ByVal eventArgs As
    System.Windows.Forms.FormClosedEventArgs) Handles Me.FormClosed
    Call cAvax1.EndAvax()
End Sub
Sub AddPolyline()
    Dim x1() As Double = Nothing
    Dim y1() As Double = Nothing
    Dim z1() As Double = Nothing
    Dim vItemProperties As Object = Nothing
    Dim iClr As Integer
    Dim iWidth As Integer
    Dim iStyle As Integer
    Dim iLayer As Integer
    ReDim x1(6)
    ReDim y1(6)
    ReDim z1(6)
    x1(1) = 10 : y1(1) = 10 : z1(1) = 0
    x1(2) = 40 : y1(2) = 10 : z1(2) = 0
    x1(3) = 50 : y1(3) = 40 : z1(3) = 0
    x1(4) = 30 : y1(4) = 50 : z1(4) = 0
    x1(5) = 0 : y1(5) = 10 : z1(5) = 0
    x1(6) = 10 : y1(6) = 10 : z1(6) = 0
    iClr = 1
    iWidth = 1
    AvaxHandle = CAvax1.Add_PolyLine(x1, y1, z1, vItemProperties, iClr, iWidth, iStyle, iLayer)
    CAvax1.Command = Avax.AvaxCommand.AutoLimits_c
End Sub
Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
    Button1.Click
    Dim [Handles](1) As Integer
    Dim Offset As Double
    Dim vRetHandles As Object = Nothing
    Dim Answer As Boolean
    [Handles](1) = AvaxHandle
    Offset = 3
    Answer = CAvax1.CopyInside([Handles], Offset, vRetHandles)
End Sub
```

CopyItems

cAvax method

Purpose

Creates translated copies of supplied objects while retaining the originals; the full overload returns the new handles.

VB.NET declarations

```
Function CopyItems(ByRef avMyHandles As Integer(), ByRef xDist As Double, ByRef yDist As Double, ByRef
    vRetHandles As Object) As Boolean

Function CopyItems(ByRef avMyHandles As Integer(), ByRef xDist As Double, ByRef yDist As Double) As
    Boolean
```

Parameters

Name	Type and passing	Meaning
avMyHandles	ByRef avMyHandles As Integer()	Input Integer() of the entity handles to process. Logical entries start at index 1; index 0 is reserved.
xDist	ByRef xDist As Double	Is the relative x distance
yDist	ByRef yDist As Double	Is the relative y distance
vRetHandles	ByRef vRetHandles As Object	Returns an integer array with the handles of the new objects

Use the declaration for the overload you call. ByRef arguments can receive outputs; declare variables and arrays with the exact indicated type. Optional defaults are shown in the declaration.

Return value

Boolean - True on success, false on error.

Remarks

The originals and input handle array are retained. The full overload supplies an independent one-based Integer() of copied handles in vRetHandles. Copies retain object attributes and receive the supplied XY translation. The displacement uses Double in v3; Z coordinates, font sizes and pen attributes are retained.

The complete object subtree is copied, including text/RTF layout, picture resources and nested blocks. Persistent external-line transforms also apply when the provider later regenerates its cache. Native dimension text gaps, direction values and inherited scale remain unchanged by translation. Editable-shape position records and hatch pattern origins move with their geometry.

When an associative hatch and its boundary are copied together, the copied hatch references the copied boundary. Later boundary edits update that hatch without changing the originals. The complete changing batch is one Undo/Redo operation.

Pass a one-based Integer handle array; index 0 is ignored and duplicate handles are processed once. Geometry uses Double. The control must be started and enabled. PauseAvax and LockEdit govern interactive input and do not prohibit this direct call. Before events can change the proposed targets or parameters, or cancel the operation. Caller arrays are separate from event arrays. The complete batch is prepared before editing. Invalid handles, unrepresentable geometry or unavailable copy capacity return False without a partial edit. A successful batch is one Undo operation and raises Change without NewUserItems. PauseAvaxEvents suppresses notifications. Application event-handler exceptions propagate; they do not roll back an edit already committed. A callback that replaces the drawing or starts new input prevents the older operation from overwriting that state.

Numeric changes from v2

Update the following variables, arrays or event-data receivers when porting the corresponding v2 call. The declarations above show the current overloads.

Parameter or value	v2 type	v3 type
xDist	Single	Double

Parameter or value	v2 type	v3 type
yDist	Single	Double

For ByRef parameters, use a variable with the exact v3 type. Preserve array dimensions and the documented indexing. Moving from v2 to v3 explains conversion and legacy file-storage rules.

Example

Setup

Add an Avax.cAvax Windows Forms control named `cAvax1`. Paste the procedures into the form class. Example setup and supporting files explains references, initialization and file paths.

Add these Windows Forms controls or components to the form. Fields referenced by Handles clauses must be declared WithEvents, as the Visual Studio designer normally does.

Name	Type
Command1	Button

VB.NET example

```
Private Sub Form1_Load(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs) Handles
    MyBase.Load
    cAvax1.StartAvax()
    Call cAvax1.Add_Line(0, 0, 0, 10, 10, 0)
    Call cAvax1.Add_Line(0, 10, 0, 10, 0, 0)
End Sub
Private Sub Form1_FormClosed(ByVal eventSender As System.Object, ByVal eventArgs As
    System.Windows.Forms.FormClosedEventArgs) Handles Me.FormClosed
    cAvax1.EndAvax()
End Sub
Private Sub Command1_Click(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs)
    Handles Command1.Click
    Dim [Handles]() As Integer = Nothing
    Dim xDist As Double
    Dim yDist As Double
    Dim vRetHandles As Object = Nothing
    Dim Answer As Boolean
    Call CAvax1.GetAllHandlesArr([Handles])
    xDist = 10
    yDist = 2
    Answer = CAvax1.CopyItems([Handles], xDist, yDist, vRetHandles)
End Sub
```

CopyOutside

cAvax method

Purpose

Creates outward offset copies of object boundaries at a distance in drawing units, retaining originals. Negative distances reverse direction.

VB.NET declarations

```
Function CopyOutside(ByRef avMyHandles As Integer(), ByRef Offset As Double) As Boolean

Function CopyOutside(ByRef avMyHandles As Integer(), ByRef Offset As Double, ByRef vRetHandles As Object)
    As Boolean
```

Parameters

Name	Type and passing	Meaning
avMyHandles	ByRef avMyHandles As Integer()	Input Integer() of the entity handles to process. Logical entries start at index 1; index 0 is reserved.
Offset	ByRef Offset As Double	Is the distance form the border of the object that copy will occur
vRetHandles	ByRef vRetHandles As Object	Returns an integer array with the handles of the new objects

Use the declaration for the overload you call. ByRef arguments can receive outputs; declare variables and arrays with the exact indicated type. Optional defaults are shown in the declaration.

Return value

Boolean - True on success, false on error.

Remarks

Pass a one-based Integer() of source handles and a finite Double distance. Negative distances reverse the named direction. The full overload returns a one-based Integer() of new handles in vRetHandles. Originals, attributes and height profiles are retained.

Polygon offsets retain mitered corners. Concave boundaries can open or close bays and produce several contours. Circles and arcs retain analytic geometry; ellipse offsets change the principal semiaxes. Wide polylines retain the surviving sides of their width profiles.

A started, enabled control is required. The complete result is prepared before one Undo operation, with Change and no NewUserItems. Invalid sources, an empty result or unrepresentable output return False without a partial edit or caller-output change. Application event-handler exceptions propagate after committed insertion.

Numeric changes from v2

Update the following variables, arrays or event-data receivers when porting the corresponding v2 call. The declarations above show the current overloads.

Parameter or value	v2 type	v3 type
Offset	Single	Double

For ByRef parameters, use a variable with the exact v3 type. Preserve array dimensions and the documented indexing. Moving from v2 to v3 explains conversion and legacy file-storage rules.

Example

Setup

Add an Avax.cAvax Windows Forms control named `cAvax1`. Paste the procedures into the form class. Example setup and supporting files explains references, initialization and file paths.

Add these Windows Forms controls or components to the form. Fields referenced by Handles clauses must be declared WithEvents, as the Visual Studio designer normally does.

Name	Type
Button1	Button

VB.NET example

```
Dim AvaxHandle As Integer
Private Sub Form1_Load(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs) Handles
    MyBase.Load
    Call CAvax1.StartAvax()
    AddPolyline()
End Sub
Private Sub Form1_FormClosed(ByVal eventSender As System.Object, ByVal eventArgs As
    System.Windows.Forms.FormClosedEventArgs) Handles Me.FormClosed
    Call cAvax1.EndAvax()
End Sub
Sub AddPolyline()
    Dim x1() As Double = Nothing
    Dim y1() As Double = Nothing
    Dim z1() As Double = Nothing
    Dim vItemProperties As Object = Nothing
    Dim iClr As Integer
    Dim iWidth As Integer
    Dim iStyle As Integer
    Dim iLayer As Integer
    ReDim x1(6)
    ReDim y1(6)
    ReDim z1(6)
    x1(1) = 10 : y1(1) = 10 : z1(1) = 0
    x1(2) = 40 : y1(2) = 10 : z1(2) = 0
    x1(3) = 50 : y1(3) = 40 : z1(3) = 0
    x1(4) = 30 : y1(4) = 50 : z1(4) = 0
    x1(5) = 0 : y1(5) = 10 : z1(5) = 0
    x1(6) = 10 : y1(6) = 10 : z1(6) = 0
    iClr = 1
    iWidth = 1
    AvaxHandle = CAvax1.Add_PolyLine(x1, y1, z1, vItemProperties, iClr, iWidth, iStyle, iLayer)
    CAvax1.Command = Avax.AvaxCommand.AutoLimits_c
End Sub
Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
    Button1.Click
    Dim [Handles](1) As Integer
    Dim Offset As Double
    Dim vRetHandles As Object = Nothing
    Dim Answer As Boolean
    [Handles](1) = AvaxHandle
    Offset = 3
    Answer = CAvax1.ConvOutside([Handles], Offset, vRetHandles)
End Sub
```

CopyToClipboard

cAvax method

Purpose

Copies objects to the Clipboard

VB.NET declarations

```
Function CopyToClipboard(ByVal lWidth As Integer, ByVal lHeight As Integer) As Boolean

Function CopyToClipboard(ByVal lWidth As Integer, ByVal lHeight As Integer, ByRef fSelected As Boolean,
    ByRef xMin As Double, ByRef yMin As Double, ByRef xMax As Double, ByRef yMax As Double, ByRef sScale As
    Double) As Boolean
```

Parameters

Name	Type and passing	Meaning
lWidth	ByVal lWidth As Integer	Integer. Bitmap width in pixels. Zero normalizes to 1; negative or unrepresentable sizes fail. The R2-compatible metafile physical width is round(width * 2.54) hundredths of a millimetre.
lHeight	ByVal lHeight As Integer	Integer. Bitmap height in pixels. Zero normalizes to 1; negative or unrepresentable sizes fail. The R2-compatible metafile physical height is round(height * 2.54) hundredths of a millimetre.
fSelected	ByRef fSelected As Boolean	True copies the selected subset; False copies the whole drawing even when a selection exists. The two-argument overload uses False.
xMin	ByRef xMin As Double	Double in v3. Four zero values select full drawing bounds, including when fSelected=True. Four -1 values select current viewport bounds. Otherwise explicit normalized extents are used. Resolved extents return ByRef; a degenerate dimension is expanded for a usable image.
yMin	ByRef yMin As Double	Double in v3. Four zero values select full drawing bounds, including when fSelected=True. Four -1 values select current viewport bounds. Otherwise explicit normalized extents are used. Resolved extents return ByRef; a degenerate dimension is expanded for a usable image.
xMax	ByRef xMax As Double	Double in v3. Four zero values select full drawing bounds, including when fSelected=True. Four -1 values select current viewport bounds. Otherwise explicit normalized extents are used. Resolved extents return ByRef; a degenerate dimension is expanded for a usable image.
yMax	ByRef yMax As Double	Double in v3. Four zero values select full drawing bounds, including when fSelected=True. Four -1 values select current viewport bounds. Otherwise explicit normalized extents are used. Resolved extents return ByRef; a degenerate dimension is expanded for a usable image.
sScale	ByRef sScale As Double	Double in v3 (Double in R2). Multiplies projected image coordinates and dimensions about the output top-left. Nonpositive values return as 1; nonfinite/unrepresentable values fail before transport. Editable native payload geometry is unchanged.

Use the declaration for the overload you call. ByRef arguments can receive outputs; declare variables and arrays with the exact indicated type. Optional defaults are shown in the declaration.

Return value

Boolean - True on success, false on error.

Remarks

If xMin, yMin, xMax, yMax are zero, then the whole drawing will be copied to the Clipboard

Numeric changes from v2

Update the following variables, arrays or event-data receivers when porting the corresponding v2 call. The declarations above show the current overloads.

Parameter or value	v2 type	v3 type
xMin	Single	Double
yMin	Single	Double
xMax	Single	Double
yMax	Single	Double
sScale	Single	Double

For ByRef parameters, use a variable with the exact v3 type. Preserve array dimensions and the documented indexing. Moving from v2 to v3 explains conversion and legacy file-storage rules.

Example

Setup

Add an Avax.cAvax Windows Forms control named `cAvax1`. Paste the procedures into the form class. Example setup and supporting files explains references, initialization and file paths.

Add these Windows Forms controls or components to the form. Fields referenced by Handles clauses must be declared WithEvents, as the Visual Studio designer normally does.

Name	Type
Command1	Button

VB.NET example

```
Private Sub Form1_Load(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs) Handles
    MyBase.Load
    Command1.Text = "CopyToClipboard"
    cAvax1.StartAvax()
    Call cAvax1.Add_Rectangle(0, 0, 12, 10)
End Sub
Private Sub Form1_FormClosed(ByVal eventSender As System.Object, ByVal eventArgs As
    System.Windows.Forms.FormClosedEventArgs) Handles Me.FormClosed
    cAvax1.EndAvax()
End Sub
Private Sub Command1_Click(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs)
    Handles Command1.Click
    Dim lWidth As Integer
    Dim lHeight As Integer
    Dim Answer As Boolean
    lWidth = 1200
    lHeight = 1200
    Answer = cAvax1.CopyToClipboard(lWidth, lHeight)
End Sub
```

CreateACADHatchPattern

cAvax method

Purpose

Creates a string with the required facts to be provided to the parameter Fill_sPattern of the commands add_ShapeXXXX when Fill_iType=AvaxFillType.PatACAD_f

VB.NET declarations

```
Function CreateACADHatchPattern(ByRef sPatFile As String, ByRef sPatternName As String, ByRef RotAngle As
    Double, ByRef sScale As Double) As String

Function CreateACADHatchPattern(ByRef sPatFile As String, ByRef sPatternName As String) As String
```

Parameters

Name	Type and passing	Meaning
sPatFile	ByRef sPatFile As String	Path to an AutoCAD PAT text file. The file is read and left unchanged.
sPatternName	ByRef sPatternName As String	Name after the asterisk in the PAT header, excluding the description. Case and surrounding whitespace are ignored; a leading asterisk is accepted.
RotAngle	ByRef RotAngle As Double	Finite counterclockwise rotation in degrees; Double. Input only.
sScale	ByRef sScale As Double	Finite uniform scale; Double. Nonpositive input returns as 1, including when the file or pattern is unavailable.

Use the declaration for the overload you call. ByRef arguments can receive outputs; declare variables and arrays with the exact indicated type. Optional defaults are shown in the declaration.

Return value

String - The parameter Fill_sPattern for the Add_ShapeXXXX functions

Remarks

Reads a named hatch pattern from an AutoCAD PAT text file and returns an opaque pattern string for a PatACAD_f shape fill. Each pattern line defines its angle, origin, repeat offsets and optional dash sequence. Pattern names ignore case and surrounding whitespace; a leading asterisk is accepted. Rotation is counterclockwise in degrees and scale is uniform; both use Double. The short overload uses zero rotation and scale one. Nonpositive finite sScale returns as 1, including when the requested file or pattern is unavailable. Missing files, missing or empty names, malformed records and nonfinite transform values return an empty string. The source file and drawing are unchanged, and StartAvax is not required for parsing. Supply the returned string as the shape fill pattern; it is not a filename.

Numeric changes from v2

Update the following variables, arrays or event-data receivers when porting the corresponding v2 call. The declarations above show the current overloads.

Parameter or value	v2 type	v3 type
RotAngle	Single	Double
sScale	Single	Double

For ByRef parameters, use a variable with the exact v3 type. Preserve array dimensions and the documented indexing. Moving from v2 to v3 explains conversion and legacy file-storage rules.

Example

Setup

Add an Avax.cAvax Windows Forms control named `cAvax1`. Paste the procedures into the form class. Example setup and supporting files explains references, initialization and file paths.

Add these Windows Forms controls or components to the form. Fields referenced by Handles clauses must be declared WithEvents, as the Visual Studio designer normally does.

Name	Type
Command1	Button

VB.NET example

```
Private Sub Form1_Load(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs) Handles
    MyBase.Load
    Command1.Text = "AddShape (CreateACADHatchPattern)"
    cAvax1.StartAvax()
End Sub
Private Sub Form1_FormClosed(ByVal eventSender As System.Object, ByVal eventArgs As
    System.Windows.Forms.FormClosedEventArgs) Handles Me.FormClosed
    cAvax1.EndAvax()
End Sub
Private Sub Command1_Click(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs)
    Handles Command1.Click
    Dim Fill_iType As Byte
    Dim Fill_lHatchIndex As Integer
    Dim Fill_iInForeColor As Integer
    Dim Fill_iInBackColor As Integer
    Dim Fill_sPattern As String = String.Empty
    Dim Fill_zOrder, Fill_lFlags As Integer
    Dim iClr, iWidth As Integer
    Dim iStyle, iLayer As Integer
    Dim xV() As Double = Nothing
    Dim yV() As Double = Nothing
    Dim zV() As Double = Nothing
    Dim sPatFile, sPatternName As String
    Dim rotAngle, sScale As Double
    ReDim xV(5)
    ReDim yV(5)
    ReDim zV(5)
    xV(1) = 2
    yV(1) = 2
    xV(2) = 2
    yV(2) = 12
    xV(3) = 12
    yV(3) = 12
    xV(4) = 12
    yV(4) = 2
    xV(5) = 10
    yV(5) = -1
    sPatFile = My.Application.Info.DirectoryPath & "\..\AcadPatterns1.pat"
    sPatternName = "HEXAGON"
    rotAngle = 0
    sScale = 10
    Fill_sPattern = cAvax1.CreateACADHatchPattern(sPatFile, sPatternName, rotAngle, sScale)
    Fill_iType = Avax.AvaxFillType.PatACAD_f
    Fill_iInForeColor = 5
    Fill_lFlags = CAvax1.CreatePatternFlags(False, False, True, False, False)
    Fill_iInBackColor = 2
    iWidth = 2 'Border width
    iClr = 5 'Border color
    Call CAvax1.Add_Shape(xV, yV, zV, Nothing, Fill_iType, Fill_lHatchIndex, Fill_iInForeColor,
    Fill_iInBackColor, Fill_sPattern, Fill_zOrder, Fill_lFlags, iClr, iWidth, iStyle, iLayer,
    Avax.AvaxPolylineType.Polyline_pt)
```

```
End Sub
```

CreateACADLineStyleString

cAvax method

Purpose

Reads a named linetype from an AutoCAD LIN file and returns its encoded definition for the AVAX line style table.

VB.NET declaration

```
Function CreateACADLineStyleString(ByRef sStyleFile As String, ByRef sStyleName As String) As String
```

Parameters

Name	Type and passing	Meaning
sStyleFile	ByRef sStyleFile As String	ByRef source LIN path, unchanged. Only the requested definition needs its external shape files.
sStyleName	ByRef sStyleName As String	ByRef requested name, unchanged; trimmed, case-insensitive matching. Missing names return Nothing.

Use the declaration for the overload you call. ByRef arguments can receive outputs; declare variables and arrays with the exact indicated type. Optional defaults are shown in the declaration.

Return value

String - The line type name and data

Remarks

Reads one named definition from an AutoCAD LIN file and returns the AVAX line-style string. The filename and requested name are passed ByRef and remain unchanged. Name matching trims the requested name and ignores case. The first matching definition is returned; a missing name returns Nothing. The method does not install the result, change the table, modify objects or raise Change/NewUserItems. Use SetLineStyle to replace a particular table entry, or SetLineStyles to install a complete table. The returned text preserves signed dash/gap/dot lengths, finite Double numbers, quoted text and supported complex-symbol parameters. Referenced AutoCAD-86 SHX 1.0/1.1 symbols and subshapes are resolved beside the LIN file or in the DirsSet options directory and embedded into the string. Only the requested definition needs its external shape resources; unrelated missing SHX files do not prevent a valid named query. An invalid requested pattern, unreadable source or missing/malformed required SHX returns Nothing. The live table remains unchanged on success or failure. This method is also usable before startup and while stopped or disabled. To repair a damaged existing entry, obtain its valid source definition and pass the result to SetLineStyle using the object's original numeric style index; querying alone leaves the old entry intact.

Example

Setup

Add an Avax.cAvax Windows Forms control named `cAvax1`. Paste the procedures into the form class. Example setup and supporting files explains references, initialization and file paths.

Add these Windows Forms controls or components to the form. Fields referenced by Handles clauses must be declared WithEvents, as the Visual Studio designer normally does.

Name	Type
Command1	Button

VB.NET example

```
Dim iStyle As Integer
Private Sub Form1_Load(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs) Handles
    MyBase.Load
    Command1.Text = "CreateACADLineStyleString"
    cAvax1.StartAvax()
```

```
    Call cAvax1.SetAvaxProperty(Avax.AvaxProp.LineStyleScale, 0.5)
    iStyle = 1
    Call cAvax1.Add_Line(0, 0, 0, 15, 15, 0, Nothing, -1, -1, iStyle, -1)
End Sub
Private Sub Form1_FormClosed(ByVal eventSender As System.Object, ByVal eventArgs As
    System.Windows.Forms.FormClosedEventArgs) Handles Me.FormClosed
    cAvax1.EndAvax()
End Sub
Private Sub Command1_Click(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs)
    Handles Command1.Click
    Dim sStyleFile As String = String.Empty
    Dim sStyleName As String = String.Empty
    Dim sLineStyleString As String = String.Empty
    sStyleFile = My.Application.Info.DirectoryPath & "\..\acad1.lin"
    sStyleName = "PHANTOMX2"
    sLineStyleString = cAvax1.CreateACADLineStyleString(sStyleFile, sStyleName)
    Call cAvax1.SetLineStyle(iStyle, sLineStyleString)
End Sub
```

CreateBlock

cAvax method

Purpose

Creates a named reusable block from existing objects, writes its BLK library definition, and replaces the objects with one block.

VB.NET declaration

```
Function CreateBlock(ByRef lHandles As Integer(), ByRef sBlockName As String, ByRef xIns As Double, ByRef
    yIns As Double) As Integer
```

Parameters

Name	Type and passing	Meaning
lHandles	ByRef lHandles As Integer()	An array of Integers for the objects's handles
sBlockName	ByRef sBlockName As String	The block name
xIns	ByRef xIns As Double	Is the x coordinate of the base point
yIns	ByRef yIns As Double	Is the y coordinate of the base point

Use the declaration for the overload you call. ByRef arguments can receive outputs; declare variables and arrays with the exact indicated type. Optional defaults are shown in the declaration.

Return value

Integer - The new block handle after the library definition and drawing replacement succeed, or 0 on failure.

Remarks

Pass a one-based Integer() of existing handles; index 0 is ignored and duplicates are used once. The name must contain 1 through 8 filename-safe characters. The definition is written as name.blk in the library directory configured by DirsSet, or in the application directory when no library directory is configured. XY coordinates in the definition are relative to (xIns, yIns); Z is retained.

The requested objects are replaced by one block at their original world positions. The returned Integer is that block handle, or 0 on failure. File preparation and geometry validation finish before drawing replacement. Invalid handles/names, write errors or values outside legacy BLK Single storage limits preserve the existing drawing and destination. A successful replacement is one Undo operation. Undo restores the objects but leaves the reusable library file in place. Change and SelectChange observe the completed replacement; NewUserItems is not raised.

Plain text-filled shapes in the legacy definition follow OriginTopLeft_p, including text in nested blocks. Use the same origin mode when opening the block in another control.

Numeric changes from v2

Update the following variables, arrays or event-data receivers when porting the corresponding v2 call. The declarations above show the current overloads.

Parameter or value	v2 type	v3 type
xIns	Single	Double
yIns	Single	Double

For ByRef parameters, use a variable with the exact v3 type. Preserve array dimensions and the documented indexing. Moving from v2 to v3 explains conversion and legacy file-storage rules.

Example

Setup

Add an Avax.cAvax Windows Forms control named `cAvax1`. Paste the procedures into the form class. Example setup and supporting files explains references, initialization and file paths.

Add these Windows Forms controls or components to the form. Fields referenced by Handles clauses must be declared WithEvents, as the Visual Studio designer normally does.

Name	Type
Command1	Button

VB.NET example

```
Private Sub Form1_Load(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs) Handles
    MyBase.Load
    cAvax1.StartAvax()
    Call cAvax1.Add_Line(0, 0, 0, 10, 10, 0)
    Call cAvax1.Add_Line(0, 10, 0, 10, 0, 0)
End Sub
Private Sub Form1_FormClosed(ByVal eventSender As System.Object, ByVal eventArgs As
    System.Windows.Forms.FormClosedEventArgs) Handles Me.FormClosed
    cAvax1.EndAvax()
End Sub
Private Sub Command1_Click(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs)
    Handles Command1.Click
    Dim lHandles() As Integer = Nothing
    Dim sBlockName As String = String.Empty
    Dim xIns As Double
    Dim yIns As Double
    Dim Answer As Integer
    Call cAvax1.GetAllHandlesArr(lHandles)
    sBlockName = "test"
    xIns = 5
    yIns = 5
    Answer = cAvax1.CreateBlock(lHandles, sBlockName, xIns, yIns)
    Call cAvax1.Add_Block(sBlockName, 10, 10, 0)
End Sub
```

CreateExVectorHatchPattern

cAvax method

Purpose

Creates a string with the required facts to be provided to the parameter Fill_sPattern of the commands add_ShapeXXXX when Fill_iType=AvaxFillType.ExVector_f

VB.NET declarations

```
Function CreateExVectorHatchPattern(ByRef sPatFile As String, ByRef sPatternName As String, ByRef xStart
    As Double, ByRef yStart As Double, ByRef RotAngle As Double, ByRef sScale As Double) As String

Function CreateExVectorHatchPattern(ByRef sPatFile As String, ByRef sPatternName As String, ByRef xStart
    As Double, ByRef yStart As Double) As String
```

Parameters

Name	Type and passing	Meaning
sPatFile	ByRef sPatFile As String	Path to an AVP text file. The file is read; it is not modified.
sPatternName	ByRef sPatternName As String	Name after the asterisk in the AVP header, excluding the description. Case and surrounding whitespace are ignored; a leading asterisk is accepted.
xStart	ByRef xStart As Double	World X placement of the local motif origin; Double. Input only.
yStart	ByRef yStart As Double	World Y placement of the local motif origin; Double. Input only.
RotAngle	ByRef RotAngle As Double	Counterclockwise world rotation in degrees; Double. Input only.
sScale	ByRef sScale As Double	Finite uniform scale; Double. Nonpositive input returns as 1, including when the file or pattern is unavailable.

Use the declaration for the overload you call. ByRef arguments can receive outputs; declare variables and arrays with the exact indicated type. Optional defaults are shown in the declaration.

Return value

String - The parameter Fill_sPattern for the Add_ShapeXXXX functions

Remarks

Reads a named extended vector pattern from an AVP text file and returns an opaque pattern string for ExVector_f fills. The four-argument overload uses zero rotation and scale one. Origin coordinates, counterclockwise rotation in degrees and scale are Double. O coordinates in the file are added to motif vertices before rotation/scaling and placement at xStart/yStart. D supplies the local repeat distances. L, P and C records describe lines, polygon dots and circles/ellipses. Pattern-name matching ignores case and surrounding whitespace and accepts a leading asterisk. Nonpositive finite sScale returns as 1, including when the requested file or pattern is unavailable. Other successful inputs are unchanged. Missing files or patterns, malformed records and nonfinite transform arguments return an empty string. Pattern creation does not change the source file, drawing or Undo history and does not require StartAvax. Treat the returned string as opaque; it is not a filename.

Numeric changes from v2

Update the following variables, arrays or event-data receivers when porting the corresponding v2 call. The declarations above show the current overloads.

Parameter or value	v2 type	v3 type
xStart	Single	Double
yStart	Single	Double
RotAngle	Single	Double

Parameter or value	v2 type	v3 type
sScale	Single	Double

For ByRef parameters, use a variable with the exact v3 type. Preserve array dimensions and the documented indexing. Moving from v2 to v3 explains conversion and legacy file-storage rules.

Example

Setup

Add an Avax.cAvax Windows Forms control named `cAvax1`. Paste the procedures into the form class. Example setup and supporting files explains references, initialization and file paths.

Add these Windows Forms controls or components to the form. Fields referenced by Handles clauses must be declared WithEvents, as the Visual Studio designer normally does.

Name	Type
Command1	Button

VB.NET example

```
Private Sub Form1_Load(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs) Handles
    MyBase.Load
    Command1.Text = "AddShape (CreateExVectorHatchPattern)"
    cAvax1.StartAvax()
End Sub
Private Sub Form1_FormClosed(ByVal eventSender As System.Object, ByVal eventArgs As
    System.Windows.Forms.FormClosedEventArgs) Handles Me.FormClosed
    cAvax1.EndAvax()
End Sub
Private Sub Command1_Click(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs)
    Handles Command1.Click
    Dim Fill_iType As Byte
    Dim Fill_lHatchIndex As Integer
    Dim Fill_iInForeColor As Integer
    Dim Fill_iInBackColor As Integer
    Dim Fill_sPattern As String = String.Empty
    Dim Fill_zOrder, Fill_lFlags As Integer
    Dim iClr, iWidth As Integer
    Dim iStyle, iLayer As Integer
    Dim xV() As Double = Nothing
    Dim yV() As Double = Nothing
    Dim zV() As Double = Nothing
    Dim sPatFile, sPatternName As String
    Dim rotAngle, sScale As Double
    Dim xStart, yStart As Double
    ReDim xV(5)
    ReDim yV(5)
    ReDim zV(5)
    xV(1) = 2
    yV(1) = 2
    xV(2) = 2
    yV(2) = 12
    xV(3) = 12
    yV(3) = 12
    xV(4) = 12
    yV(4) = 2
    xV(5) = 10
    yV(5) = -1
    sPatFile = My.Application.Info.DirectoryPath & "\..\AvaxPatterns1.avp"
    sPatternName = "concrete"
    rotAngle = 0
    sScale = 5
    Fill_iType = Avax.AvaxFillType.FxVector_f
    Fill_sPattern = cAvax1.CreateExVectorHatchPattern(sPatFile, sPatternName, xStart, yStart, rotAngle,
```

```
    sScale)
    Fill iInForeColor = 5
    Fill lFlags = CAvax1.CreatePatternFlags(False, False, True, False, False)
    Fill iInBackColor = 2
    iWidth = 2 'Border width
    iClr = 5 'Border color
    Call CAvax1.Add_Shape(xV, yV, zV, Nothing, Fill.iType, Fill.lHatchIndex, Fill.iInForeColor,
    Fill.iInBackColor, Fill.sPattern, Fill.zOrder, Fill.lFlags, iClr, iWidth, iStyle, iLayer,
    Avax.AvaxPolylineType.Polyline_pt)
End Sub
```

CreatePatternFlags

cAvax method

Purpose

Creates an Integer to be provided to the parameter Fill_lFlags of the commands add_ShapeXXXX

VB.NET declarations

```
Function CreatePatternFlags() As Integer

Function CreatePatternFlags(ByRef fStretchXOn As Boolean, ByRef fStretchYOn As Boolean, ByRef
    fBackgroundOn As Boolean, ByRef fBorderOff As Boolean, ByRef fPolylineOn As Boolean) As Integer
```

Parameters

Name	Type and passing	Meaning
fStretchXOn	ByRef fStretchXOn As Boolean	ExVector_f: True renders only the origin column instead of repeating along local X.
fStretchYOn	ByRef fStretchYOn As Boolean	ExVector_f: True renders only the origin row instead of repeating along local Y.
fBackgroundOn	ByRef fBackgroundOn As Boolean	Historical name: True sets 8192 and suppresses the area background.
fBorderOff	ByRef fBorderOff As Boolean	True sets 4096 to hide the shape boundary.
fPolylineOn	ByRef fPolylineOn As Boolean	True sets 2048 to leave the shape boundary open.

Use the declaration for the overload you call. ByRef arguments can receive outputs; declare variables and arrays with the exact indicated type. Optional defaults are shown in the declaration.

Return value

Integer - The parameter Fill_lFlags for the Add_ShapeXXXX functions

Remarks

Returns the legacy fill option bit mask. StartAvax must have succeeded; unstarted or stopped controls return 0. A disabled but started control can still construct flags. The parameterless overload returns 0. The Boolean overload preserves its ByRef inputs: StretchX=32768, StretchY=16384, BackgroundOn=8192, BorderOff=4096 and PolylineOn=2048. For ExVector_f, StretchX/Y restrict repetition to the origin column/row in pattern-local coordinates. BackgroundOn=True suppresses the area background despite the historical name. BorderOff hides the boundary; PolylineOn leaves it open. The effect of these options depends on the fill family. This function does not create or modify drawing objects.

Example

Setup

Add an Avax.cAvax Windows Forms control named `cAvax1`. Paste the procedures into the form class. Example setup and supporting files explains references, initialization and file paths.

Add these Windows Forms controls or components to the form. Fields referenced by Handles clauses must be declared WithEvents, as the Visual Studio designer normally does.

Name	Type
Command1	Button

VB.NET example

```
Private Sub Form1_Load(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs) Handles
    MyBase.Load
    Command1.Text = "AddShape (CreatePatternFlags)"
```

```
    cAvax1.StartAvax()
End Sub
Private Sub Form1_FormClosed(ByVal eventSender As System.Object, ByVal eventArgs As
    System.Windows.Forms.FormClosedEventArgs) Handles Me.FormClosed
    cAvax1.EndAvax()
End Sub
Private Sub Command1_Click(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs)
    Handles Command1.Click
    Dim Fill_iType As Byte
    Dim Fill_lHatchIndex As Integer
    Dim Fill_iInForeColor As Integer
    Dim Fill_iInBackColor As Integer
    Dim Fill_sPattern As String = String.Empty
    Dim Fill_zOrder, Fill_lFlags As Integer
    Dim iClr, iWidth As Integer
    Dim iStyle, iLayer As Integer
    Dim xV() As Double = Nothing
    Dim yV() As Double = Nothing
    Dim zV() As Double = Nothing
    Dim sPatFile, sPatternName As String
    Dim rotAngle, sScale As Double
    Dim fBackgroundOn As Boolean
    Dim fBorderOff As Boolean
    ReDim xV(5)
    ReDim yV(5)
    ReDim zV(5)
    xV(1) = 2
    yV(1) = 2
    xV(2) = 2
    yV(2) = 12
    xV(3) = 12
    yV(3) = 12
    xV(4) = 12
    yV(4) = 2
    xV(5) = 10
    yV(5) = -1
    sPatFile = My.Application.Info.DirectoryPath & "\..\AcadPatterns1.pat"
    sPatternName = "HEXAGON"
    rotAngle = 0
    sScale = 10
    Fill_sPattern = cAvax1.CreateACADHatchPattern(sPatFile, sPatternName, rotAngle, sScale)
    Fill_iType = Avax.AvaxFillType.PatACAD_f
    Fill_iInForeColor = 5
    '---
    fBackgroundOn = False 'No Background
    fBorderOff = True 'No Border
    Fill_lFlags = CAvax1.CreatePatternFlags(False, False, fBackgroundOn, fBorderOff, False)
    '---
    Fill_iInBackColor = 2
    Call CAvax1.Add_Shape(xV, yV, zV, Nothing, Fill_iType, Fill_lHatchIndex, Fill_iInForeColor,
    Fill_iInBackColor, Fill_sPattern, Fill_zOrder, Fill_lFlags, iClr, iWidth, iStyle, iLayer,
    Avax.AvaxPolylineType.Polyline_pt)
End Sub
```

CreateSVGHatchPattern

API reference

Purpose

Creates the SVG pattern descriptor used by an SVG hatch fill. Assign the result to Fill_sPattern and use AvaxFillType.PatSVG_f as the fill type when adding a shape. The method returns a complete pattern element; it does not add an object to the drawing.

This R2-compatible string helper does not require StartAvax. It also works on a disabled control or after EndAvax. The pattern name is trimmed and XML-escaped. SVG content inherits the SVG namespace and the standard xlink namespace, so internal references such as xlink:href='#cell' need no extra declaration in the fragment. Empty content creates a transparent tile; an empty name remains supported for legacy anonymous patterns.

VB.NET declaration

```
Public Function CreateSVGHatchPattern(ByVal sPatternName As String,
    ByVal x As Double, ByVal y As Double,
    ByVal pWidth As Double, ByVal pHeight As Double,
    ByVal SvgPattern As String) As String
```

Parameters

Parameter	Meaning
sPatternName	Name stored in the pattern's id attribute. Leading and trailing whitespace is removed. The caller's string is not changed.
x, y	Finite Double coordinates of the repeated tile origin in drawing units.
pWidth, pHeight	Positive, finite Double tile dimensions in drawing units. The descriptor explicitly uses patternUnits='userSpaceOnUse'. These are tile dimensions, not the dimensions of the shape being filled.
SvgPattern	Well-formed SVG child markup, for example rect, path or defs/use content supported by the SVG fill renderer. Supply a fragment without an XML declaration or document type declaration. SVG's own downward Y direction applies inside the tile. Nothing or an empty string produces an empty tile.

Return value

The SVG pattern string, or an empty string for nonfinite coordinates, nonpositive dimensions or malformed XML. Geometry values are written using invariant numeric formatting in all Windows cultures. The v3 Double parameters replace the R2 Single parameters and retain their additional precision. The method does not change its ByVal arguments, the drawing, its selection or undo history.

Example

The following complete helper adds a patterned rectangle to an already started control. The tile is 10.5 by 12.75 drawing units; the filled rectangle is 100 by 80 units. The legacy fill flag 8192 keeps the spaces between motifs transparent.

```
Option Strict On
Imports System
Imports Avax

Public NotInheritable Class Help_CreateSVGHatchPattern
    Public Shared Function AddGreenPattern(cad As cAvax) As Integer
        If cad Is Nothing Then Throw New ArgumentNullException(NameOf(cad))
        Dim fragment As String =
            "<defs><rect id='cell' width='4' height='4' fill='#00ff00'/></defs>" &
            "<use xlink:href='#cell'/>"
        Dim pattern As String = cad.CreateSVGHatchPattern(
            "GreenSquares", 0.125, 0.25, 10.5, 12.75, fragment)
        If String.IsNullOrEmpty(pattern) Then Return 0

        Dim x As Double = 0, y As Double = 0
        Dim width As Double = 100, height As Double = 80
        Dim properties As Object = Nothing
        Dim fillType As Byte = CByte(AvaxFillType.PatSVG_f)
        Dim hatch As Integer = 0, foreground As Integer = 1, background As Integer = 0
```

```
        Dim order As Integer = 0, flags As Integer = 8192
        Dim color As Integer = 1, penWidth As Integer = 1, style As Integer = 0
        Dim layer As Integer = -1
        Return cad.Add_ShapeRectangle(x, y, width, height, properties,
            fillType, hatch, foreground, background, pattern, order, flags,
            color, penWidth, style, layer)
    End Function
End Class
```

R2 descriptors that omit namespace declarations and patternUnits remain readable. Complete standalone pattern XML with an XML declaration is also accepted by the fill reader, although this creator expects child markup only.

CreateVectorHatchPattern

cAvax method

Purpose

Creates a string with the required facts to be provided to the parameter Fill_sPattern of the commands add_ShapeXXXX when Fill_iType=AvaxFillType.Vector_f

VB.NET declarations

```
Function CreateVectorHatchPattern(ByRef x1 As Double(), ByRef y1 As Double(), ByRef x2 As Double(), ByRef
    y2 As Double(), ByRef dx As Double, ByRef dy As Double, ByRef RotAngle As Double, ByRef sScale As Double)
    As String

Function CreateVectorHatchPattern(ByRef x1 As Double(), ByRef y1 As Double(), ByRef x2 As Double(), ByRef
    y2 As Double()) As String
```

Parameters

Name	Type and passing	Meaning
x1	ByRef x1 As Double()	Double array of segment start X coordinates. Indices 1..N are used; index 0 is ignored. N must be 1..1024.
y1	ByRef y1 As Double()	Double array of segment start Y coordinates, with the same length as x1.
x2	ByRef x2 As Double()	Double array of segment end X coordinates, with the same length as x1.
y2	ByRef y2 As Double()	Double array of segment end Y coordinates, with the same length as x1.
dx	ByRef dx As Double	Local horizontal repeat distance before uniform scaling. Zero uses automatic spacing. Negative input returns as 0.
dy	ByRef dy As Double	Local vertical repeat distance before uniform scaling. Zero uses automatic spacing. Negative input returns as 0.
RotAngle	ByRef RotAngle As Double	Finite counterclockwise rotation in degrees; Double. Input only.
sScale	ByRef sScale As Double	Finite uniform scale; Double. Nonpositive input returns as 1 after the arrays are accepted.

Use the declaration for the overload you call. ByRef arguments can receive outputs; declare variables and arrays with the exact indicated type. Optional defaults are shown in the declaration.

Return value

String - The parameter Fill_sPattern for the Add_ShapeXXXX functions

Remarks

Creates an opaque vector pattern from 1 to 1024 line segments for a Vector_f shape fill. Supply four equally sized Double arrays using indices 1 through the segment count; index 0 is reserved and ignored. The short overload uses zero rotation, unit scale and automatic repeat distances. The full overload accepts counterclockwise rotation in degrees and uniform scale. Nonpositive scale returns as 1; negative dx or dy returns as 0. Positive repeat distances are scaled with the motif; zero selects automatic spacing. These normalized values are returned through the ByRef arguments. Coordinate arrays and rotation are unchanged. StartAvax must have succeeded; a disabled but started control can still create patterns. Invalid arrays, nonfinite input or an unrepresentable transform return an empty string; an unstarted or stopped control also returns an empty string. No drawing object is created and Undo history is unchanged. Pass the returned string to the shape fill pattern parameter; do not treat it as a filename.

Numeric changes from v2

Update the following variables, arrays or event-data receivers when porting the corresponding v2 call. The declarations above show the current overloads.

Parameter or value	v2 type	v3 type
x1	Single()	Double()
y1	Single()	Double()
x2	Single()	Double()
y2	Single()	Double()
dx	Single	Double
dy	Single	Double
RotAngle	Single	Double
sScale	Single	Double

For ByRef parameters, use a variable with the exact v3 type. Preserve array dimensions and the documented indexing. Moving from v2 to v3 explains conversion and legacy file-storage rules.

Example

Setup

Add an Avax.cAvax Windows Forms control named `cAvax1`. Paste the procedures into the form class. Example setup and supporting files explains references, initialization and file paths.

Add these Windows Forms controls or components to the form. Fields referenced by Handles clauses must be declared WithEvents, as the Visual Studio designer normally does.

Name	Type
Command1	Button

VB.NET example

```
Private Sub Form1_Load(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs) Handles
    MyBase.Load
    Command1.Text = "AddShape (CreateVectorHatchPattern)"
    cAvax1.StartAvax()
End Sub
Private Sub Form1_FormClosed(ByVal eventSender As System.Object, ByVal eventArgs As
    System.Windows.Forms.FormClosedEventArgs) Handles Me.FormClosed
    cAvax1.EndAvax()
End Sub
Private Sub Command1_Click(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs)
    Handles Command1.Click
    Dim Fill iType As Byte
    Dim Fill lHatchIndex As Integer
    Dim Fill iInForeColor As Integer
    Dim Fill iInBackColor As Integer
    Dim Fill sPattern As String = String.Empty
    Dim Fill zOrder, Fill lFlags As Integer
    Dim iClr, iWidth As Integer
    Dim iStyle, iLayer As Integer
    Dim xV() As Double = Nothing
    Dim yV() As Double = Nothing
    Dim zV() As Double = Nothing
    Dim x1() As Double = Nothing
    Dim y1() As Double = Nothing
    Dim x2() As Double = Nothing
    Dim y2() As Double = Nothing
    Dim dx, dy As Double
    Dim rotAngle, sScale As Double
    ReDim xV(5)
    ReDim yV(5)
    ReDim zV(5)
    xV(1) = 2
```

```
    vV(1) = 2
    xV(2) = 2
    vV(2) = 12
    xV(3) = 12
    vV(3) = 12
    xV(4) = 12
    vV(4) = 2
    xV(5) = 10
    vV(5) = -1
    ReDim x1(3)
    ReDim y1(3)
    ReDim x2(3)
    ReDim y2(3)
    x1(1) = 0
    y1(1) = 0
    x2(1) = 1
    y2(1) = 1
    x1(2) = 1
    y1(2) = 1
    x2(2) = 3
    y2(2) = 1
    x1(3) = 3
    y1(3) = 1
    x2(3) = 1
    y2(3) = 5
    rotAngle = 0
    sScale = 1
    Fill iType = Avax.AvaxFillType.Vector_f
    Fill sPattern = cAvax1.CreateVectorHatchPattern(x1, y1, x2, y2, dx, dy, rotAngle, sScale)
    Fill iInForeColor = 5
    Fill lFlags = CAvax1.CreatePatternFlags(False, False, True, False, False)
    Fill iInBackColor = 2
    iWidth = 2 'Width
    iClr = 1 'Border color
    Call CAvax1.Add_Shape(xV, yV, zV, Nothing, Fill iType, Fill lHatchIndex, Fill iInForeColor,
    Fill iInBackColor, Fill sPattern, Fill zOrder, Fill lFlags, iClr, iWidth, iStyle, iLayer,
    Avax.AvaxPolylineType.Polyline_pt)
End Sub
```

CurrentRedo

cAvax method

Purpose

Returns the number of redos available and the type of the latest redo

VB.NET declaration

```
Function CurrentRedo(ByRef RedoType As AvaxUndo) As Integer
```

Parameters

Name	Type and passing	Meaning
RedoType	ByRef RedoType As AvaxUndo	An expression that defines the type of the last redo

Use the declaration for the overload you call. ByRef arguments can receive outputs; declare variables and arrays with the exact indicated type. Optional defaults are shown in the declaration.

Return value

Integer - The number of redos

Example

Setup

Add an Avax.cAvax Windows Forms control named `cAvax1`. Paste the procedures into the form class. Example setup and supporting files explains references, initialization and file paths.

Add these Windows Forms controls or components to the form. Fields referenced by Handles clauses must be declared WithEvents, as the Visual Studio designer normally does.

Name	Type
Command1	Button
Command2	Button
Command3	Button

VB.NET example

```
Private Sub Form1_Load(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs) Handles
    MyBase.Load
    cAvax1.StartAvax()
    Call cAvax1.Add_Line(0, 0, 0, 10, 10, 0)
    Call cAvax1.Add_Line(0, 10, 0, 10, 10, 0)
End Sub
Private Sub Form1_FormClosed(ByVal eventSender As System.Object, ByVal eventArgs As
    System.Windows.Forms.FormClosedEventArgs) Handles Me.FormClosed
    cAvax1.EndAvax()
End Sub
Private Sub Command1_Click(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs)
    Handles Command1.Click
    Dim sUndoMsg As String = String.Empty
    Dim sRedoMsg As String = String.Empty
    Call GetUndoRedo(sUndoMsg, sRedoMsg)
    MsgBox("Undo=" & sUndoMsg & vbCrLf & "Redo=" & sRedoMsg, MsgBoxStyle.Information, "Undo/Redo")
End Sub
Private Sub Command2_Click(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs)
    Handles Command2.Click
    CAvax1.Command = Avax.AvaxCommand.Undo_c
End Sub
```

```
Private Sub Command3_Click(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs)
    Handles Command3.Click
    CAvax1.Command = Avax.AvaxCommand.Redo_c
End Sub
Public Sub GetUndoRedo(ByRef sUndoMsg As String, ByRef sRedoMsg As String)
    Dim UndoType As Avax.AvaxUndo
    Dim res1, res2 As Integer
    Dim s1, s2 As String
    UndoType = Avax.AvaxUndo.None_u
    res1 = cAvax1.CurrentUndo(UndoType)
    s1 = TakeMsg(UndoType)
    UndoType = Avax.AvaxUndo.None_u
    res2 = cAvax1.CurrentRedo(UndoType)
    s2 = s1
    s1 = TakeMsg(UndoType)
    swap(s1, s2)
    sUndoMsg = s1
    sRedoMsg = s2
    Exit Sub
End Sub
Private Function TakeMsg(ByVal UndoType As Avax.AvaxUndo) As String
    Select Case UndoType
        Case Avax.AvaxUndo.None_u
        Return "None"
        Case Avax.AvaxUndo.Erase_u
        Return "Erase"
        Case Avax.AvaxUndo.Unerase_u
        Return "Unerase"
        Case Avax.AvaxUndo.Create_u
        Return "Create"
        Case Avax.AvaxUndo.Rotate_u
        Return "Rotate"
        Case Avax.AvaxUndo.Mirror_u
        Return "Mirror"
        Case Avax.AvaxUndo.Scale_u
        Return "Scale"
        Case Avax.AvaxUndo.Move_u
        Return "Move"
        Case Avax.AvaxUndo.Transform_u
        Return "Transform"
        Case Avax.AvaxUndo.Zoom_u
        Return "Zoom"
        Case Avax.AvaxUndo.Stretch_u
        Return "Stretch"
        Case Avax.AvaxUndo.Freeze_u
        Return "Freeze"
        Case Avax.AvaxUndo.Thaw_u
        Return "Thaw"
        Case Avax.AvaxUndo.Change_u
        Return "Change"
        Case Else
        Return String.Empty
    End Select
End Function
Private Sub swap(ByRef a As Object, ByRef b As Object)
    Dim c As Object = Nothing
    c = b
    b = a
    a = c
End Sub
```

CurrentUndo

cAvax method

Purpose

Returns the number of undos available and the type of the latest undo

VB.NET declaration

```
Function CurrentUndo(ByRef UndoType As AvaxUndo) As Integer
```

Parameters

Name	Type and passing	Meaning
UndoType	ByRef UndoType As AvaxUndo	An expression that defines the type of the last undo

Use the declaration for the overload you call. ByRef arguments can receive outputs; declare variables and arrays with the exact indicated type. Optional defaults are shown in the declaration.

Return value

Integer - The number of undos

Example

Setup

Add an Avax.cAvax Windows Forms control named `cAvax1`. Paste the procedures into the form class. Example setup and supporting files explains references, initialization and file paths.

Add these Windows Forms controls or components to the form. Fields referenced by Handles clauses must be declared WithEvents, as the Visual Studio designer normally does.

Name	Type
Command1	Button
Command2	Button
Command3	Button

VB.NET example

```
Private Sub Form1_Load(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs) Handles
    MyBase.Load
    cAvax1.StartAvax()
    Call cAvax1.Add_Line(0, 0, 0, 10, 10, 0)
    Call cAvax1.Add_Line(0, 10, 0, 10, 10, 0)
End Sub
Private Sub Form1_FormClosed(ByVal eventSender As System.Object, ByVal eventArgs As
    System.Windows.Forms.FormClosedEventArgs) Handles Me.FormClosed
    cAvax1.EndAvax()
End Sub
Private Sub Command1_Click(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs)
    Handles Command1.Click
    Dim sUndoMsg As String = String.Empty
    Dim sRedoMsg As String = String.Empty
    Call GetUndoRedo(sUndoMsg, sRedoMsg)
    MsgBox("Undo=" & sUndoMsg & vbCrLf & "Redo=" & sRedoMsg, MsgBoxStyle.Information, "Undo/Redo")
End Sub
Private Sub Command2_Click(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs)
    Handles Command2.Click
    CAvax1.Command = Avax.AvaxCommand.Undo_c
End Sub
```

```
Private Sub Command3_Click(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs)
    Handles Command3.Click
    CAvax1.Command = Avax.AvaxCommand.Redo_c
End Sub
Public Sub GetUndoRedo(ByRef sUndoMsg As String, ByRef sRedoMsg As String)
    Dim UndoType As Avax.AvaxUndo
    Dim res1, res2 As Integer
    Dim s1, s2 As String
    UndoType = Avax.AvaxUndo.None_u
    res1 = cAvax1.CurrentUndo(UndoType)
    s1 = TakeMsg(UndoType)
    UndoType = Avax.AvaxUndo.None_u
    res2 = cAvax1.CurrentRedo(UndoType)
    s2 = s1
    s1 = TakeMsg(UndoType)
    swap(s1, s2)
    sUndoMsg = s1
    sRedoMsg = s2
End Sub
Private Function TakeMsg(ByVal UndoType As Avax.AvaxUndo) As String
    Select Case UndoType
        Case Avax.AvaxUndo.None_u
        Return "None"
        Case Avax.AvaxUndo.Erase_u
        Return "Erase"
        Case Avax.AvaxUndo.Unerase_u
        Return "Unerase"
        Case Avax.AvaxUndo.Create_u
        Return "Create"
        Case Avax.AvaxUndo.Rotate_u
        Return "Rotate"
        Case Avax.AvaxUndo.Mirror_u
        Return "Mirror"
        Case Avax.AvaxUndo.Scale_u
        Return "Scale"
        Case Avax.AvaxUndo.Move_u
        Return "Move"
        Case Avax.AvaxUndo.Transform_u
        Return "Transform"
        Case Avax.AvaxUndo.Zoom_u
        Return "Zoom"
        Case Avax.AvaxUndo.Stretch_u
        Return "Stretch"
        Case Avax.AvaxUndo.Freeze_u
        Return "Freeze"
        Case Avax.AvaxUndo.Thaw_u
        Return "Thaw"
        Case Avax.AvaxUndo.Change_u
        Return "Change"
        Case Else
        Return String.Empty
    End Select
End Function
Private Sub swap(ByRef a As Object, ByRef b As Object)
    Dim c As Object = Nothing
    c = b
    b = a
    a = c
End Sub
```

CurrentXY

cAvax method

Purpose

Returns the drawing coordinates of the mouse pointer's position.

VB.NET declaration

```
Function CurrentXY(ByRef x As Double, ByRef y As Double) As Boolean
```

Parameters

Name	Type and passing	Meaning
x	ByRef x As Double	Is the x coordinate of the mouse pointer's position
y	ByRef y As Double	Is the y coordinate of the mouse pointer's position

Use the declaration for the overload you call. ByRef arguments can receive outputs; declare variables and arrays with the exact indicated type. Optional defaults are shown in the declaration.

Return value

Boolean - True on success, false on error.

Numeric changes from v2

Update the following variables, arrays or event-data receivers when porting the corresponding v2 call. The declarations above show the current overloads.

Parameter or value	v2 type	v3 type
x	Single	Double
y	Single	Double

For ByRef parameters, use a variable with the exact v3 type. Preserve array dimensions and the documented indexing. Moving from v2 to v3 explains conversion and legacy file-storage rules.

Example

Setup

Add an Avax.cAvax Windows Forms control named `cAvax1`. Paste the procedures into the form class. Example setup and supporting files explains references, initialization and file paths.

Add these Windows Forms controls or components to the form. Fields referenced by Handles clauses must be declared WithEvents, as the Visual Studio designer normally does.

Name	Type
Label1	Label

VB.NET example

```
Private Sub Form1_Load(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs) Handles
    MyBase.Load
    cAvax1.StartAvax()
End Sub
Private Sub Form1_FormClosed(ByVal eventSender As System.Object, ByVal eventArgs As
    System.Windows.Forms.FormClosedEventArgs) Handles Me.FormClosed
    cAvax1.EndAvax()
End Sub
Private Sub CAvax1_OnMouseDown1(ByVal Sender As Object, ByVal e As Avax.cAvax.MouseDownEventArgs) Handles
```

```
    CAvax1.OnMouseDown
    Dim x1, y1 As Double
    If CAvax1.CurrentXY(x1, y1) = True Then
        Label1.Text = "X=" & Str(x1) & "   " & "Y=" & Str(y1)
    End If
End Sub
```

CurrentZoomPercentage

API reference

Returns the current zoom as a percentage of the drawing's work area. Use it to display the zoom level in a status bar. The reference area is set by SetBoundsAvax and can be read with GetBoundsAvax. Adding or deleting an object does not change the reference unless another operation changes the work area.

```
Public Function CurrentZoomPercentage() As Double
```

This method has no parameters. A view covering the work area returns 100; a view with half its width and height returns 200. The percentage is the smaller of the horizontal and vertical work-to-view ratios, multiplied by 100. The view is fitted to the drawing viewport, excluding rulers and scroll bars. Resizing the control can therefore change the percentage.

The return type is Double in v3 (Single in v2). Empty drawings have a valid percentage. The method returns zero when the control has not started, has ended or is disposed, or when a finite percentage cannot be represented. It remains readable while the control is disabled, paused or locked. Reading it does not change the drawing, selection, active command, Undo/Redo history or events.

VB.NET example

Pass a started AVAX control to this function and assign the returned text to a status-bar label. The invariant format below always uses a decimal point; use the application's culture instead if localized display is preferred.

```
Option Strict On
Imports System
Imports System.Globalization

Public Module ZoomPercentageExample
    Public Function ReadZoomLabel(cad As Avax.cAvax) As String
        If cad Is Nothing Then Throw New ArgumentNullException(NameOf(cad))
        Dim percentage As Double = cad.CurrentZoomPercentage()
        Return percentage.ToString("0.00", CultureInfo.InvariantCulture) & "%"
    End Function
End Module
```

For example, a 400-by-200-unit view inside an 800-by-400-unit work area displays `200.00%`. Read the value again after a ZoomChange event or when updating the application's status display.

DeleteAllItems

cAvax method

Purpose

Deletes all the objects in the drawing

VB.NET declaration

```
Function DeleteAllItems() As Boolean
```

Return value

Boolean - True on success, false on error.

Remarks

Deletes every drawing object in one undoable erase operation, including objects on hidden or locked layers. Returns False for an empty drawing or an inactive control. A successful call abandons the old interactive command and clears selection. Undo restores the deleted objects. The current filename, control settings and resource tables remain available. Use DestroyDrawing to discard the objects together with Undo/Redo history.

Example

Setup

Add an Avax.cAvax Windows Forms control named `cAvax1`. Paste the procedures into the form class. Example setup and supporting files explains references, initialization and file paths.

Add these Windows Forms controls or components to the form. Fields referenced by Handles clauses must be declared WithEvents, as the Visual Studio designer normally does.

Name	Type
Command1	Button

VB.NET example

```
Private Sub Form1_Load(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs) Handles
    MyBase.Load
    cAvax1.StartAvax()
    Call cAvax1.Add_Line(0, 0, 0, 10, 10, 0)
    Call cAvax1.Add_Line(0, 10, 0, 10, 10, 0)
End Sub
Private Sub Form1_FormClosed(ByVal eventSender As System.Object, ByVal eventArgs As
    System.Windows.Forms.FormClosedEventArgs) Handles Me.FormClosed
    cAvax1.EndAvax()
End Sub
Private Sub Command1_Click(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs)
    Handles Command1.Click
    Dim Answer As Boolean
    If cAvax1.MaxItems > 0 Then
        Answer = cAvax1.DeleteAllItems()
    Else
        MsgBox("Nothing to delete", MsgBoxStyle.Critical, "")
    End If
End Sub
```

DeleteAvaxFile

cAvax method

Purpose

Deletes the .AVX file

VB.NET declaration

```
Function DeleteAvaxFile(ByRef sFileName As String) As Boolean
```

Parameters

Name	Type and passing	Meaning
sFileName	ByRef sFileName As String	Is the filename of the .AVX file to be deleted from the disk

Use the declaration for the overload you call. ByRef arguments can receive outputs; declare variables and arrays with the exact indicated type. Optional defaults are shown in the declaration.

Return value

Boolean - True on success, false on error.

Remarks

Deletes one closed, valid AVAX drawing file and returns True only when deletion succeeds. Relative names use the control data directory; a name without an extension defaults to .avx. Supports .avx, .avax and legacy or native .blk files. The ByRef filename is unchanged. Empty or missing names, directories, linked paths, unreadable, read-only, invalid, corrupt or unsupported-version files return False. A drawing open in this or another control in the same process is protected; use StartAvax to switch to another drawing or EndAvax to close it before deleting. Does not change drawing contents, selection or Undo history. Available before StartAvax and while Enabled=False; a disposed control returns False. Only the specified drawing is removed; adjacent files and asset directories are retained. File and format errors are reported as error messages.

Example

Setup

Add an Avax.cAvax Windows Forms control named `cAvax1`. Paste the procedures into the form class. Example setup and supporting files explains references, initialization and file paths.

Add these Windows Forms controls or components to the form. Fields referenced by Handles clauses must be declared WithEvents, as the Visual Studio designer normally does.

Name	Type
Command1	Button

VB.NET example

```
Dim sFileName As String = String.Empty
Private Sub Form1_Load(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs) Handles
    MyBase.Load
    Command1.Text = "DeleteAvaxFile"
    sFileName = My.Application.Info.DirectoryPath & "\Temp.Avx"
    cAvax1.StartAvax()
    Call cAvax1.Add_Line(0, 0, 0, 10, 10, 0)
    Call cAvax1.SaveAs(sFileName)
    ' Switch away: DeleteAvaxFile protects drawings still open in a control.
    cAvax1.StartAvax()
End Sub
Private Sub Form1_FormClosed(ByVal eventSender As System.Object, ByVal eventArgs As
    System.Windows.Forms.FormClosedEventArgs) Handles Me.FormClosed
```

```
    cAvax1.EndAvax()
End Sub
Private Sub Command1_Click(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs)
    Handles Command1.Click
    MsgBox(cAvax1.DeleteAvaxFile(sFileName), MsgBoxStyle.Information, "DeleteAvaxFile")
End Sub
```

DeleteItems

cAvax method

Purpose

Deletes the objects described in a specific handles array

VB.NET declaration

```
Function DeleteItems(ByRef avMyHandles As Integer()) As Boolean
```

Parameters

Name	Type and passing	Meaning
avMyHandles	ByRef avMyHandles As Integer()	Input Integer() of the entity handles to process. Logical entries start at index 1; index 0 is reserved.

Use the declaration for the overload you call. ByRef arguments can receive outputs; declare variables and arrays with the exact indicated type. Optional defaults are shown in the declaration.

Return value

Boolean - True on success, false on error.

Remarks

Deletes the existing objects named in a one-based Integer handle array; element zero is ignored and the ByRef array is unchanged. Duplicate handles are deleted once; missing or nonpositive handles are skipped. Returns False without changing the drawing, command, selection or Undo history when no live handle remains. A successful call abandons the old interactive command, clears its selection, and records one undoable erase operation. Requires a started, enabled, undisposed control; LockEdit only restricts interactive editing and does not prevent this API call. Callback-created input, selection and replacement drawings are preserved.

Example

Setup

Add an Avax.cAvax Windows Forms control named `cAvax1`. Paste the procedures into the form class. Example setup and supporting files explains references, initialization and file paths.

Add these Windows Forms controls or components to the form. Fields referenced by Handles clauses must be declared WithEvents, as the Visual Studio designer normally does.

Name	Type
Command1	Button

VB.NET example

```
Private Sub Form1_Load(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs) Handles
    MyBase.Load
    cAvax1.StartAvax()
    Call cAvax1.Add_Line(0, 0, 0, 10, 10, 0)
    Call cAvax1.Add_Line(0, 10, 0, 10, 10, 0)
End Sub
Private Sub Form1_FormClosed(ByVal eventSender As System.Object, ByVal eventArgs As
    System.Windows.Forms.FormClosedEventArgs) Handles Me.FormClosed
    cAvax1.EndAvax()
End Sub
Private Sub Command1_Click(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs)
    Handles Command1.Click
    Dim [Handles]() As Integer = Nothing
    Dim Answer As Boolean
    If CAvax1.GetAllHandlesArr([Handles]) > 0 Then
```

```
        Answer = CAvax1.DeleteItems([Handles])
    Else
        MsgBox("Nothing to delete.", MsgBoxStyle.Critical, "DeleteItems")
    End If
End Sub
```

DestroyDrawing

cAvax method

Purpose

This method erases all the drawing objects and the undo/redo buffer

VB.NET declarations

```
Function DestroyDrawing() As Boolean

Function DestroyDrawing(ByRef fRedraw As Boolean) As Boolean
```

Parameters

Name	Type and passing	Meaning
fRedraw	ByRef fRedraw As Boolean	A Boolean expression that if is true then the drawing in the window is regenerated

Use the declaration for the overload you call. ByRef arguments can receive outputs; declare variables and arrays with the exact indicated type. Optional defaults are shown in the declaration.

Return value

Boolean - True on success, false on error.

Remarks

Clears all drawing objects and both Undo/Redo buffers. The erased drawing cannot be restored with Undo. Resets handle allocation, clears cached blocks and per-drawing stored metadata, and abandons the current command and selection. Retains the filename, view, pen settings and resource tables; it does not save over the disk file. The no-argument overload redraws; the Boolean overload redraws only when fRedraw is True and leaves that ByRef argument unchanged. AutoRedraw is not modified. A started, enabled control returns True even when already empty, so remaining Undo/Redo history can still be cleared. An inactive or disposed control returns False. Changes made by event handlers after the reset are preserved.

Example

Setup

Add an Avax.cAvax Windows Forms control named `cAvax1`. Paste the procedures into the form class. Example setup and supporting files explains references, initialization and file paths.

Add these Windows Forms controls or components to the form. Fields referenced by Handles clauses must be declared WithEvents, as the Visual Studio designer normally does.

Name	Type
Command1	Button

VB.NET example

```
Private Sub Form1_Load(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs) Handles
    MyBase.Load
    Command1.Text = "DestroyDrawing"
    cAvax1.StartAvax()
    Call cAvax1.Add_Line(0, 0, 0, 15, 15, 0)
    Call cAvax1.Add_Line(0, 15, 0, 15, 15, 0)
    Call cAvax1.Add_Line(15, 15, 0, 15, 0, 0)
    Call cAvax1.Add_Line(0, 15, 0, 15, 0, 0)
    Call cAvax1.Add_Line(0, 0, 0, 15, 0, 0)
    Call cAvax1.Add_Line(0, 0, 0, 0, 15, 0)
End Sub
Private Sub Form1_FormClosed(ByVal eventSender As System.Object, ByVal eventArgs As
```

```
    System.Windows.Forms.FormClosedEventArgs) Handles Me.FormClosed
    cAvax1.EndAvax()
End Sub
Private Sub Command1_Click(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs)
    Handles Command1.Click
    Dim fRedraw As Boolean
    Dim Answer As Boolean
    fRedraw = True
    Answer = cAvax1.DestroyDrawing(fRedraw)
End Sub
```

DirsGet

cAvax method

Purpose

Returns the directories which contains the AVAX support, drawing and block files

VB.NET declaration

```
Sub DirsGet(ByRef sDirOpt As String, ByRef sDirData As String, ByRef sDirLib As String, ByRef sDirCom As
    String)
```

Parameters

Name	Type and passing	Meaning
sDirOpt	ByRef sDirOpt As String	String ByRef output: configured support/options directory, with a trailing separator when nonempty.
sDirData	ByRef sDirData As String	String ByRef output: configured drawing/data directory, with a trailing separator when nonempty.
sDirLib	ByRef sDirLib As String	String ByRef output: configured block/library directory, with a trailing separator when nonempty.
sDirCom	ByRef sDirCom As String	String ByRef output: configured legacy command (unused) directory, with a trailing separator when nonempty.

Use the declaration for the overload you call. ByRef arguments can receive outputs; declare variables and arrays with the exact indicated type. Optional defaults are shown in the declaration.

Result

This procedure has no return value. Any outputs are supplied through the documented ByRef arguments.

Example

Setup

Add an Avax.cAvax Windows Forms control named `cAvax1`. Paste the procedures into the form class. Example setup and supporting files explains references, initialization and file paths.

Add these Windows Forms controls or components to the form. Fields referenced by Handles clauses must be declared WithEvents, as the Visual Studio designer normally does.

Name	Type
Command1	Button

VB.NET example

```
Dim sDirOpt As String = String.Empty
Dim sDirData As String = String.Empty
Dim sDirLib As String = String.Empty
Dim sDirCom As String = String.Empty
Private Sub Form1_Load(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs) Handles
    MyBase.Load
    Command1.Text = "DirsGet"
    cAvax1.StartAvax()
    sDirOpt = My.Application.Info.DirectoryPath & "\OptAvax"
    sDirData = My.Application.Info.DirectoryPath & "\DataAvax"
    sDirLib = My.Application.Info.DirectoryPath & "\LibAvax"
    sDirCom = "" 'unused
    Call cAvax1.DirsSet(sDirOpt, sDirData, sDirLib, sDirCom)
End Sub
```

```
Private Sub Form1_FormClosed(ByVal eventSender As System.Object, ByVal eventArgs As
    System.Windows.Forms.FormClosedEventArgs) Handles Me.FormClosed
    cAvax1.EndAvax()
End Sub
Private Sub Command1_Click(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs)
    Handles Command1.Click
    Dim sMsg As String = String.Empty
    Call cAvax1.DirsGet(sDirOpt, sDirData, sDirLib, sDirCom)
    sMsg = "DirOpt=" & sDirOpt & vbCrLf
    sMsg = sMsg & "DirData=" & sDirData & vbCrLf
    sMsg = sMsg & "DirLib=" & sDirLib & vbCrLf
    MsgBox(sMsg, MsgBoxStyle.Information, "DirsGet")
End Sub
```

DirsSet

cAvax method

Purpose

Sets the directories which contains the AVAX support, drawing and block files

VB.NET declaration

```
Sub DirsSet(ByRef sDirOpt As String, ByRef sDirData As String, ByRef sDirLib As String, ByRef sDirCom As
    String)
```

Parameters

Name	Type and passing	Meaning
sDirOpt	ByRef sDirOpt As String	String ByRef input, unchanged. Empty/Nothing preserves this slot; nonempty configures and creates the support/options directory.
sDirData	ByRef sDirData As String	String ByRef input, unchanged. Empty/Nothing preserves this slot; nonempty configures and creates the drawing/data directory.
sDirLib	ByRef sDirLib As String	String ByRef input, unchanged. Empty/Nothing preserves this slot; nonempty configures and creates the block/library directory.
sDirCom	ByRef sDirCom As String	String ByRef input, unchanged. Empty/Nothing preserves this slot; nonempty configures and creates the legacy command (unused) directory.

Use the declaration for the overload you call. ByRef arguments can receive outputs; declare variables and arrays with the exact indicated type. Optional defaults are shown in the declaration.

Result

This procedure has no return value. Any outputs are supplied through the documented ByRef arguments.

Example

Setup

Add an Avax.cAvax Windows Forms control named `cAvax1`. Paste the procedures into the form class. Example setup and supporting files explains references, initialization and file paths.

Add these Windows Forms controls or components to the form. Fields referenced by Handles clauses must be declared WithEvents, as the Visual Studio designer normally does.

Name	Type
Command1	Button

VB.NET example

```
Dim sDirOpt As String = String.Empty
Dim sDirData As String = String.Empty
Dim sDirLib As String = String.Empty
Dim sDirCom As String = String.Empty
Private Sub Form1_Load(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs) Handles
    MyBase.Load
    Command1.Text = "DirsGet"
    cAvax1.StartAvax()
    sDirOpt = My.Application.Info.DirectoryPath & "\OptAvax"
    sDirData = My.Application.Info.DirectoryPath & "\DataAvax"
    sDirLib = My.Application.Info.DirectoryPath & "\LibAvax"
    sDirCom = "" 'unused
    Call cAvax1.DirsSet(sDirOpt, sDirData, sDirLib, sDirCom)
End Sub
```

```
Private Sub Form1_FormClosed(ByVal eventSender As System.Object, ByVal eventArgs As
    System.Windows.Forms.FormClosedEventArgs) Handles Me.FormClosed
    cAvax1.EndAvax()
End Sub
Private Sub Command1_Click(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs)
    Handles Command1.Click
    Dim sMsg As String = String.Empty
    Call cAvax1.DirsGet(sDirOpt, sDirData, sDirLib, sDirCom)
    sMsg = "DirOpt=" & sDirOpt & vbCrLf
    sMsg = sMsg & "DirData=" & sDirData & vbCrLf
    sMsg = sMsg & "DirLib=" & sDirLib & vbCrLf
    MsgBox(sMsg, MsgBoxStyle.Information, "DirsGet")
End Sub
```

DoAvaxAction

cAvax method

Purpose

Executes a supported action directly on supplied handles. vIn supplies its parameters; OutHandles, vOut and errCode return its result.

See DoAvaxAction and public value structures for direct polyline closure, attribute/fill/text/stretch/status fields, widened numeric declarations and complete examples.

See Immediate actions and selection borders for direct mode changes, filtered selection, one overall Border, outputs, callbacks and accepted Polyline input.

VB.NET declaration

```
Function DoAvaxAction(ByRef InHandles As Integer(), ByRef vIn As Object, ByRef myAction As AvaxCommand,
    ByRef fNewHandles As Boolean, ByRef OutHandles As Integer(), ByRef vOut As Object, ByRef errCode As
    Integer) As Boolean
```

Parameters

Name	Type and passing	Meaning
InHandles	ByRef InHandles As Integer()	An array of Integers for the handles of the objects
vIn	ByRef vIn As Object	A Variant expression that defines additional data for this function
myAction	ByRef myAction As AvaxCommand	An expression that defines the command which you want to execute for the specific handles
fNewHandles	ByRef fNewHandles As Boolean	A Boolean expression that defines if there are new objects after the command's execution
OutHandles	ByRef OutHandles As Integer()	An array of Integers for the handles of the new objects if there are any
vOut	ByRef vOut As Object	A Variant expression that defines answers after the command's execution
errCode	ByRef errCode As Integer	If this parameter is not zero then an error was occured

Use the declaration for the overload you call. ByRef arguments can receive outputs; declare variables and arrays with the exact indicated type. Optional defaults are shown in the declaration.

Return value

Boolean - True when the selected action succeeds. False reports an expected failure through errCode; action outputs are described below.

Remarks

InHandles is a one-based Integer() of existing drawing handles. myAction selects a supported AvaxCommand. For move, rotate, mirror and scale actions, vIn contains the corresponding AvaxMove_t, AvaxRotate_t, AvaxMirror_t or AvaxScale_t value. Matrix transforms use a one-based Double(3,3) matrix. Use the action-specific value structures and related member topics for field meanings. A started, enabled control is required; edit guards depend on the selected action.

OutHandles contains this action result. fNewHandles indicates creation of new objects; vOut carries an additional result, such as the Double total from Measure_c or Area_c. Expected input or preparation failures return False and a nonzero errCode. Measurement actions do not edit the drawing.

Explode, Explode_Block, Pedit, Taxis and P_Break prepare replacement geometry before editing and return their created handles. P_Break splits line/polyline/shape intersections in XY and retains each source elevation and width profile. Block_c takes a one-based String() containing a name/path and optional invariant-culture XY base coordinates. Copy_In_c and Copy_Out_c use the magnitude of CopyInOutWidth_p, ignore vIn and retain the original objects.

Object-snap requests and other supported drafting aids use the Command route while retaining active construction input. A completed editing batch is undoable and publishes Change without NewUserItems. Before events may change targets or cancel.

Application callback exceptions propagate. Commands or drawings started by a callback remain owned by that callback and are not cleared by the returning older action.

Example

Setup

Add an Avax.cAvax Windows Forms control named `cAvax1`. Paste the procedures into the form class. Example setup and supporting files explains references, initialization and file paths.

Add these Windows Forms controls or components to the form. Fields referenced by Handles clauses must be declared WithEvents, as the Visual Studio designer normally does.

Name	Type
Command1	Button

VB.NET example

```
Dim InHandles(2) As Integer
Private Sub Form1_Load(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs) Handles
    MyBase.Load
    Command1.Text = "Do action [MOVE]"
    cAvax1.StartAvax()
    InHandles(1) = cAvax1.Add_Line(0, 0, 0, 10, 10, 0)
    InHandles(2) = cAvax1.Add_Line(10, 0, 0, 10, 10, 0)
End Sub
Private Sub Form1_FormClosed(ByVal eventSender As System.Object, ByVal eventArgs As
    System.Windows.Forms.FormClosedEventArgs) Handles Me.FormClosed
    cAvax1.EndAvax()
End Sub
Private Sub Command1_Click(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs)
    Handles Command1.Click
    Dim vIn As Object = Nothing
    Dim myAction As Avax.AvaxCommand
    Dim fNewHandles As Boolean
    Dim OutHandles() As Integer = Nothing
    Dim vOut As Object = Nothing
    Dim errCode As Integer
    Dim Answer As Boolean
    Dim myMove As Avax.AvaxMove_t
    myMove.xDist = 1
    myMove.yDist = 2
    vIn = myMove
    myAction = Avax.AvaxCommand.Move_c
    Answer = cAvax1.DoAvaxAction(InHandles, vIn, myAction, fNewHandles, OutHandles, vOut, errCode)
End Sub
```

Draw_Line

cAvax method

Purpose

Draws a straight line segment without including it in the current drawing

VB.NET declarations

```
Sub Draw_Line(ByRef x1 As Double, ByRef y1 As Double, ByRef x2 As Double, ByRef y2 As Double)
Sub Draw_Line(ByRef x1 As Double, ByRef y1 As Double, ByRef x2 As Double, ByRef y2 As Double, ByRef iClr
    As Integer, ByRef iWidth As Integer, ByRef iStyle As Integer)
```

Parameters

Name	Type and passing	Meaning
x1	ByRef x1 As Double	Double ByRef input (formerly Double), unchanged: start X coordinate.
y1	ByRef y1 As Double	Double ByRef input (formerly Double), unchanged: start Y coordinate.
x2	ByRef x2 As Double	Double ByRef input (formerly Double), unchanged: end X coordinate.
y2	ByRef y2 As Double	Double ByRef input (formerly Double), unchanged: end Y coordinate. The old CHM incorrectly labels this as X.
iClr	ByRef iClr As Integer	Integer ByRef input, unchanged. -1 uses PenColor; ordinary values select palette entries; 256 plus a system-color index selects a Windows system color.
iWidth	ByRef iWidth As Integer	Integer ByRef input/output. -1 resolves PenWidth; widths outside 1..255 become 1. Width is in pixels.
iStyle	ByRef iStyle As Integer	Integer ByRef input/output. -1 resolves PenLineStyle. Values above255 request reversible NOT-XOR painting and are reduced by256 before validation; invalid normalized styles become0. The returned value excludes the raster-operation flag.

Use the declaration for the overload you call. ByRef arguments can receive outputs; declare variables and arrays with the exact indicated type. Optional defaults are shown in the declaration.

Result

This procedure has no return value. Any outputs are supplied through the documented ByRef arguments.

Numeric changes from v2

Update the following variables, arrays or event-data receivers when porting the corresponding v2 call. The declarations above show the current overloads.

Parameter or value	v2 type	v3 type
x1	Single	Double
y1	Single	Double
x2	Single	Double
y2	Single	Double
iClr	Short	Integer
iWidth	Short	Integer
iStyle	Short	Integer

For ByRef parameters, use a variable with the exact v3 type. Preserve array dimensions and the documented indexing. Moving from v2 to v3 explains conversion and legacy file-storage rules.

Example

Setup

Add an Avax.cAvax Windows Forms control named `cAvax1`. Paste the procedures into the form class. Example setup and supporting files explains references, initialization and file paths.

Add these Windows Forms controls or components to the form. Fields referenced by Handles clauses must be declared WithEvents, as the Visual Studio designer normally does.

Name	Type
Command1	Button

VB.NET example

```
Private Sub Form1_Load(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs) Handles
    MyBase.Load
    cAvax1.StartAvax()
End Sub
Private Sub Form1_FormClosed(ByVal eventSender As System.Object, ByVal eventArgs As
    System.Windows.Forms.FormClosedEventArgs) Handles Me.FormClosed
    cAvax1.EndAvax()
End Sub
Private Sub Command1_Click(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs)
    Handles Command1.Click
    Dim x1 As Double
    Dim y1 As Double
    Dim x2 As Double
    Dim y2 As Double
    Dim iClr As Integer
    Dim iWidth As Integer
    Dim iStyle As Integer
    x1 = 0
    y1 = 0
    x2 = 12
    y2 = 12
    iClr = 1
    iWidth = 1
    iStyle = 1
    Call cAvax1.Draw_Line(x1, y1, x2, y2, iClr, iWidth, iStyle)
End Sub
```

EndAvax

cAvax method

Purpose

Closes the AVAX file and sets the control to its initial setting

VB.NET declaration

```
Sub EndAvax()
```

Result

This procedure has no return value. Any outputs are supplied through the documented ByRef arguments.

Remarks

Closes the current drawing and returns the reusable control to its unstarted state. The filename becomes empty; geometry, selection, accepted construction input, Undo/Redo, zoom history and update batching are reset. Repeated calls are harmless. SaveOnExit is honored before the reset, including when interactive input is disabled. If saving fails, the exception is propagated and the current drawing, modified state, active command and accepted points remain available for retry. A newer command or drawing started by a save callback retains ownership; the returning older close cannot discard it. Save callbacks do not recursively autosave. A native child disposal callback may restart the control; that newer session is retained. Successful closing cancels active printing/export, releases borrowed window DCs, stops the inactivity timer and releases the old document's native controls, retained graphics and input sessions. Ordinary drawing files are retained. Only exact temporary drawing files created by this control are removed; unrelated files in the temporary folder are preserved. EndAvax does not dispose the control. Call StartAvax or assign filename to use it again. Exit_c performs this same close operation. Dispose releases owned temporary files without implicitly saving an ordinary drawing; use EndAvax when configured SaveOnExit behavior is required.

Example

Setup

Add an Avax.cAvax Windows Forms control named `cAvax1`. Paste the procedures into the form class. Example setup and supporting files explains references, initialization and file paths.

Add these Windows Forms controls or components to the form. Fields referenced by Handles clauses must be declared WithEvents, as the Visual Studio designer normally does.

Name	Type
Command1	Button

VB.NET example

```
Private Sub Form1_Load(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs) Handles
    MyBase.Load
    cAvax1.StartAvax()
    Call cAvax1.Add_Line(0, 0, 0, 10, 10, 0)
End Sub
Private Sub Form1_FormClosed(ByVal eventSender As System.Object, ByVal eventArgs As
    System.Windows.Forms.FormClosedEventArgs) Handles Me.FormClosed
    cAvax1.EndAvax()
End Sub
Private Sub Command1_Click(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs)
    Handles Command1.Click
    System.Diagnostics.Debug.Print(cAvax1.IsChanged.ToString())
    cAvax1.EndAvax()
End Sub
```

EndUpdate

```
Public Sub EndUpdate()
```

Ends one level of display suspension. At the outermost level it refreshes the drawing, including changes made while AutoRedraw was False. It leaves the AutoRedraw property unchanged and does not repeat model events that were already delivered. Calling EndUpdate without a pending BeginUpdate is a refresh. Starting or ending a drawing resets old display suspensions.

Related options and reference

ExportNativeDrawing

Purpose

Writes a complete native drawing copy while retaining the active control's filename, dirty state, selection and Undo history.

Declaration

```
Function ExportNativeDrawing(ByVal fileName As String) As Boolean
```

Parameter and result

Name	Meaning
fileName	Destination path with an explicit .avax, .avx or .blk extension, matched without regard to case. A relative path is resolved against the application's current working directory. The destination directory must already exist.

Returns True after successful output, False when the operation cannot start or a handled preparation, file or cancellation condition prevents completion. Call this synchronous method on the control's owning thread after StartAvax or filename activation. The control must be enabled and undisposed, output must be available, and another print/export operation must not be active.

Drawing and resource ownership

The control refreshes live external references before taking the drawing snapshot. The copy includes the complete drawing rather than the current selection: object properties, labels and retained resources travel with it according to the selected native format. Embedded data remains part of the copy, and file-backed references retain their stored reference values. Applications preparing a drawing for distribution should account for the data retained by their own object properties and resources.

Writing the copy does not switch the control to the destination filename or mark the active drawing as saved. Existing selection and Undo history remain in the active control. Supplying the active drawing's own path overwrites that file on disk while retaining the control's existing dirty flag; use the ordinary save operation when the application needs its normal saved-state transition.

Format limits and failure

Native .avax retains Double coordinates. An .avx copy uses legacy Single-coordinate storage. A .blk copy normally uses legacy block storage; an extended resource table selects native block storage instead. The selected storage format's numeric and resource limits apply, and unrepresentable data is rejected. Point-radius and line-style-scale display settings must fit the legacy range when either .avx or .blk is requested. Use .avax to retain the full v3 drawing representation. Output is prepared in an owned temporary file in the destination directory and committed only after completion. A handled failure or cancellation preserves an existing destination. A drawing or control-lifetime change detected during preparation abandons the pending copy. AbortPrinting requests cancellation through the shared output mechanism. A False result can be accompanied by LastOperationError; that diagnostic can be empty or refer to an earlier failure.

VB.NET example

```
Option Strict On
Public Module ExportNativeDrawingExample
    ' Supply an existing started control and a destination chosen by the host.
    Public Function WriteNativeCopy(
        cad As Avax.cAvax, destination As String) As Boolean
        Dim originalName As String = cad.filename
        Dim originalDirty As Boolean = cad.IsChanged
        If cad.ExportNativeDrawing(destination) Then
            System.Diagnostics.Debug.Assert(cad.filename = originalName)
            System.Diagnostics.Debug.Assert(cad.IsChanged = originalDirty)
            Return True
        End If
        Dim message As String = cad.LastOperationError
        If Not String.IsNullOrEmpty(message) Then
            System.Diagnostics.Debug.WriteLine(message)
        End If
        Return False
    End Function
End Module
```

Related options and reference

FindHandlesArrByKey

cAvax method

Purpose

Finds the handles of the objects that contain a specific string in their key names

VB.NET declarations

```
Function FindHandlesArrByKey(ByRef sKey As String, ByRef lHandles As Integer(), ByRef fFindWholeWordOnly
    As Boolean, ByRef fMatchCase As Boolean) As Integer

Function FindHandlesArrByKey(ByRef sKey As String, ByRef lHandles As Integer()) As Integer
```

Parameters

Name	Type and passing	Meaning
sKey	ByRef sKey As String	Substring/phrase to find; Nothing/empty returns no matches. Query text is not trimmed.
lHandles	ByRef lHandles As Integer()	One-based Integer() output; index zero reserved. Returned count equals Length-1; a successful search with no matches returns {0}.
fFindWholeWordOnly	ByRef fFindWholeWordOnly As Boolean	True requires literal space or start/end boundaries around the matching phrase. Does not require the entire key to equal the query.
fMatchCase	ByRef fMatchCase As Boolean	True compares exact case. False follows R2 current-culture uppercase matching.

Use the declaration for the overload you call. ByRef arguments can receive outputs; declare variables and arrays with the exact indicated type. Optional defaults are shown in the declaration.

Return value

Integer - Is the total number of the objects found with the specific string key name

Example

Setup

Add an Avax.cAvax Windows Forms control named cAvax1. Paste the procedures into the form class. Example setup and supporting files explains references, initialization and file paths.

Add these Windows Forms controls or components to the form. Fields referenced by Handles clauses must be declared WithEvents, as the Visual Studio designer normally does.

Name	Type
Command1	Button

VB.NET example

```
Private Sub Form1_Load(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs) Handles
    MyBase.Load
    Dim h As Integer
    Command1.Text = "FindHandlesArrByKey"
    cAvax1.StartAvax()
    h = cAvax1.Add_Line(0, 0, 0, 15, 0, 0)
    Call cAvax1.SetHandleKey(h, "Line #1")
    h = cAvax1.Add_Line(0, 15, 0, 15, 15, 0)
    Call cAvax1.SetHandleKey(h, "Line #2")
    h = cAvax1.Add_Line(15, 0, 0, 15, 15, 0)
    Call cAvax1.SetHandleKey(h, "Line #3")
    h = cAvax1.Add_Line(0, 0, 0, 0, 15, 0)
    Call cAvax1.SetHandleKey(h, "Line #4")
```

```
    h = cAvax1.Add_Line(0, 0, 0, 15, 15, 0)
    Call cAvax1.SetHandleKey(h, "New Line #1")
End Sub
Private Sub Form1_FormClosed(ByVal eventSender As System.Object, ByVal eventArgs As
    System.Windows.Forms.FormClosedEventArgs) Handles Me.FormClosed
    cAvax1.EndAvax()
End Sub
Private Sub Command1_Click(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs)
    Handles Command1.Click
    Dim sKey As String = String.Empty
    Dim sStr As String = String.Empty
    Dim lHandles() As Integer = Nothing
    Dim fFindWholeWordOnly As Boolean
    Dim fMatchCase As Boolean
    Dim lMax, lCnt As Integer
    sKey = "Line"
    fFindWholeWordOnly = False
    fMatchCase = False
    lMax = cAvax1.FindHandlesArrByKey(sKey, lHandles, fFindWholeWordOnly, fMatchCase)
    If lMax <> 0 Then
        sStr = "Handles with the frase" & Chr(34) & sKey & Chr(34) & vbCrLf
        For lCnt = 1 To lMax
            sStr = sStr & Str(lHandles(lCnt)) & vbCrLf
        Next lCnt
        MsgBox(sStr, MsgBoxStyle.Information, "FindHandlesArrByKey")
    Else
        MsgBox("No handles found", MsgBoxStyle.Critical, "FindHandlesArrByKey")
    End If
End Sub
```

FinishCompoundPath

Purpose

Creates one drawing object from the completed segments of the current BeginCompoundPath session, using the current drawing attributes.

Declaration

```
Function FinishCompoundPath() As Integer
```

Parameters and result

This method has no parameters. Returns the new Integer handle, or 0 when no usable draft can be committed. At least one completed segment is required. A partially entered arc or Bezier segment prevents finishing; complete its points, or assign CompoundPathSegmentKind to discard those incomplete controls. When creation is attempted, the draft session ends. A successfully created object is selected and participates in Undo/Redo. The control must remain started, enabled and editable in the same drawing that began the session.

VB.NET example

```
Option Strict On
Public Module FinishCompoundPathExample
    Public Function DrawTriangle(cad As Avax.cAvax) As Integer
        If Not cad.BeginCompoundPath(True) Then Return 0
        cad.SetCoordinate(0.0, 0.0, 0.0)
        cad.SetCoordinate(30.0, 0.0, 0.0)
        cad.SetCoordinate(15.0, 20.0, 0.0)
        Dim handle As Integer = cad.FinishCompoundPath()
        If handle > 0 Then
            System.Diagnostics.Debug.WriteLine("Created: " & handle)
        End If
        Return handle
    End Function
End Module
```

Related options and reference

FitDrawingToWorkspacePage

API reference

Purpose

Fits the drawing extents inside the current paper margins by setting `PageScale`, `PageOriginX` and `PageOriginY`. The sheet is centered on the complete drawing, including objects outside the current selection. The calculation uses `DrawingUnitMM`, the selected paper size, orientation and margin.

VB.NET declaration

```
Sub FitDrawingToWorkspacePage()
```

Parameters

None.

Return value and behavior

No return value. Geometry, selection, current view and `WorkspaceMode` are preserved. A successful page change is one Undo/Redo action and raises the normal workspace/change notifications. Calling the method again with the same extents and settings produces no additional edit.

The scale includes 2% breathing room and is rounded upward to four significant digits. A single-point extent retains the current scale and centers the paper on that point. An empty drawing, or a drawing containing only non-drawing `Data_i` records, leaves the page setup unchanged. A stopped or disposed control performs no action. An unrepresentable scale throws an argument exception before changing the page setup.

Example

Call this complete procedure on the owning UI thread after `StartAvax()` and after adding or drawing the objects. The example uses meters, A4 portrait paper and 10 mm margins. It displays the fitted paper and then zooms the view to it.

```
Option Strict On
Imports Avax

Public Module Help_FitDrawingToWorkspacePage
    Public Sub Run(cad As cAvax)
        cad.DrawingUnitMM = 1000.0
        cad.PagePaperSize = AvaxPaperSize.epsA4
        cad.PageOrientation = AvaxPageOrientation.Portrait
        cad.PageMarginMM = 10.0
        cad.FitDrawingToWorkspacePage()
        cad.WorkspaceMode = AvaxWorkspaceMode.Page
        cad.ZoomWorkspacePage()
    End Sub
End Module
```

Related members

CenterWorkspacePage, ZoomWorkspacePage, GetWorkspacePageBounds.

FitToAvax

cAvax method

Purpose

Fits the complete drawing in the current view and recalculates cached drawing bounds and the total work area.

VB.NET declaration

```
Function FitToAvax() As Boolean
```

Return value

Boolean - True on success, false on error.

Remarks

The fit includes the complete drawing, refreshes external line sources and accounts for the current ScreenXFactor and fixed/minimum display sizes. DownLeftFactor and UpRightFactor are ignored and retained. Empty, point and line drawings receive a usable view. Drawing geometry, the active construction command and accepted points are retained.

The control must be started and enabled. An unrepresentable fit returns False and preserves the preceding view and bounds. If a source callback replaces the drawing or changes the current input, view or projection, the older fit stops rather than applying stale bounds. Application callback exceptions propagate.

Example

Setup

Add an Avax.cAvax Windows Forms control named `cAvax1`. Paste the procedures into the form class. Example setup and supporting files explains references, initialization and file paths.

Add these Windows Forms controls or components to the form. Fields referenced by Handles clauses must be declared WithEvents, as the Visual Studio designer normally does.

Name	Type
Command1	Button

VB.NET example

```
Private Sub Form1_Load(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs) Handles
    MyBase.Load
    Command1.Text = "FitToAvax"
    cAvax1.StartAvax()
    Call cAvax1.Add_Line(-10, -10, 0, 10, 10, 0)
End Sub
Private Sub Form1_FormClosed(ByVal eventSender As System.Object, ByVal eventArgs As
    System.Windows.Forms.FormClosedEventArgs) Handles Me.FormClosed
    cAvax1.EndAvax()
End Sub
Private Sub Command1_Click(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs)
    Handles Command1.Click
    Dim Answer As Boolean
    Answer = cAvax1.FitToAvax()
End Sub
```

Get_Dimension

Purpose

Read a detached, editable dimension definition.

VB.NET declaration

```
Function Get_Dimension(ByVal handle As Integer) As AvaxDimensionData
```

Parameters

Parameter	Meaning
handle	Handle of an existing native dimension in this control.

Return value and behavior

An independent AvaxDimensionData copy, or Nothing for an invalid/non-dimension handle. Apply edits with Set_Dimension.

Example

Call this procedure on the owning UI thread with a control already initialized by `StartAvax()`. Handles must belong to that control. File paths are supplied by the caller.

```
Option Strict On
Imports System
Imports Avax

Public Module Help_Get_Dimension
    Public Sub Run(cad As cAvax, handle As Integer)
        Dim data = cad.Get_Dimension(handle)
        If data Is Nothing Then Throw New ArgumentException("Native dimension required.")
        Dim value As Double = data.Measurement
    End Sub
End Module
```

Related options and reference

Get_DimensionDefaults

Purpose

Snapshot existing AVAX Dim defaults as per-object style overrides.

VB.NET declaration

```
Function Get_DimensionDefaults(Optional ByVal kind As AvaxDimensionKind = AvaxDimensionKind.Linear) As
    AvaxDimensionOptions
```

Parameters

Parameter	Meaning
kind	Dimension kind; defaults to Linear. Angular uses AngleDefText instead of DimDefText.

Return value and behavior

A new AvaxDimensionOptions. Editing it does not change global defaults or existing dimensions.

Example

Call this procedure on the owning UI thread with a control already initialized by `StartAvax()`. Handles must belong to that control. File paths are supplied by the caller.

```
Option Strict On
Imports System
Imports Avax

Public Module Help_Get_DimensionDefaults
    Public Sub Run(cad As cAvax)
        Dim options = cad.Get_DimensionDefaults(AvaxDimensionKind.Aligned)
        options.TextAttachment = AvaxDimTextAttachment.MiddleCenter
        Dim handle = cad.Add_Dim_Aligned(0, 0, 80, 60, 0, 70, 0, options:=options)
        If handle = 0 Then Throw New InvalidOperationException("Dimension creation failed.")
    End Sub
End Module
```

Related options and reference

Get_DimensionMeasurement

Purpose

Read the unformatted geometric measurement.

VB.NET declaration

```
Function Get_DimensionMeasurement(ByVal handle As Integer) As Double
```

Parameters

Parameter	Meaning
handle	Handle of an existing native dimension in this control.

Return value and behavior

Drawing units for length, radius or diameter; degrees for angular dimensions. Excludes measurement scaling, rounding and formatting. Double.NaN indicates failure.

Example

Call this procedure on the owning UI thread with a control already initialized by `StartAvax()`. Handles must belong to that control. File paths are supplied by the caller.

```
Option Strict On
Imports System
Imports Avax

Public Module Help_Get_DimensionMeasurement
    Public Sub Run(cad As cAvax, handle As Integer)
        Dim value = cad.Get_DimensionMeasurement(handle)
        If Double.IsNaN(value) Then Throw New ArgumentException("Native dimension required.")
    End Sub
End Module
```

Related options and reference

Get_DimensionStyle

Purpose

Read a detached named style.

VB.NET declaration

```
Function Get_DimensionStyle(Optional ByVal styleName As String = "AVAX_STANDARD") As AvaxDimensionStyle
```

Parameters

Parameter	Meaning
styleName	Style name; AVAX_STANDARD when omitted. Unknown names are invalid.

Return value and behavior

A copied AvaxDimensionStyle, or Nothing for an unknown style. AVAX_STANDARD has built-in defaults even before explicit registration.

Example

Call this procedure on the owning UI thread with a control already initialized by `StartAvax()`. Handles must belong to that control. File paths are supplied by the caller.

```
Option Strict On
Imports System
Imports Avax

Public Module Help_Get_DimensionStyle
    Public Sub Run(cad As cAvax)
        Dim style = cad.Get_DimensionStyle()
        If style Is Nothing Then Throw New InvalidOperationException("Style unavailable.")
        Dim height As Double = style.TextHeight
    End Sub
End Module
```

Related options and reference

Get_Hatch

Purpose

Read a detached native hatch definition.

VB.NET declaration

```
Function Get_Hatch(ByVal handle As Integer) As AvaxHatchData
```

Parameters

Parameter	Meaning
handle	Handle of an existing native hatch in this control.

Return value and behavior

AvaxHatchData copy, or Nothing for an invalid/non-hatch handle. Changes require Set_Hatch.

Example

Call this procedure on the owning UI thread with a control already initialized by `StartAvax()`. Handles must belong to that control. File paths are supplied by the caller.

```
Option Strict On
Imports System
Imports Avax

Public Module Help_Get_Hatch
    Public Sub Run(cad As cAvax, handle As Integer)
        Dim data = cad.Get_Hatch(handle)
        If data Is Nothing Then Throw New ArgumentException("Native hatch required.")
        Dim associated As Boolean = data.Options.Associative
    End Sub
End Module
```

Related options and reference

Get_HatchArea

Purpose

Calculate the net filled area using the hatch island rules.

VB.NET declaration

```
Function Get_HatchArea(ByVal handle As Integer) As Double
```

Parameters

Parameter	Meaning
handle	Handle of an existing native hatch in this control.

Return value and behavior

Nonnegative XY area in squared drawing units; Double.NaN on failure. The calculation uses the analytic line, conic and rational spline definitions, independently of display tessellation.

The calculation resolves intersections and coincident boundaries before integrating the selected filled region. Each loop uses even-odd filling. Normal retains all loops; Outer retains exterior regions and their first nested level; Ignore retains exterior regions. A loop is nested only when its complete filled region is a proper subset of another region: touching is allowed, but equal and merely overlapping regions are not nested. All original loops are preserved. Display, hit testing and CAD boundary flags use the same nesting policy. Conic and rational spline integration uses numerical estimates; results outside the finite Double range or arrangements that cannot be resolved report an error.

Example

Call this procedure on the owning UI thread with a control already initialized by `StartAvax()`. Handles must belong to that control. File paths are supplied by the caller.

```
Option Strict On
Imports System
Imports Avax

Public Module Help_Get_HatchArea
    Public Sub Run(cad As cAvax, handle As Integer)
        Dim area = cad.Get_HatchArea(handle)
        If Double.IsNaN(area) Then Throw New ArgumentException("Native hatch required.")
    End Sub
End Module
```

Related options and reference

Get_HatchDefaults

Purpose

Copy current legacy hatch angle, spacing and pen elevation into new native options.

VB.NET declaration

```
Function Get_HatchDefaults() As AvaxHatchOptions
```

Parameters

None.

Return value and behavior

A detached AvaxHatchOptions with UserDefined fill and Associative=True. Editing it does not alter the global hatch defaults.

Example

Call this procedure on the owning UI thread with a control already initialized by `StartAvax()`. Handles must belong to that control. File paths are supplied by the caller.

```
Option Strict On
Imports System
Imports Avax

Public Module Help_Get_HatchDefaults
    Public Sub Run(cad As cAvax)
        Dim options = cad.Get_HatchDefaults()
        options.AngleDeg = 30
        options.Spacing = 2
        options.DoublePattern = True
    End Sub
End Module
```

Related options and reference

GetAllHandlesArr

cAvax method

Purpose

Returns with the number and the handles of all the objects of the current drawing

VB.NET declarations

```
Function GetAllHandlesArr(ByRef avMyHandles As Integer(), ByRef ItemType As AvaxItemType, ByRef fzSort As
    Boolean) As Integer

Function GetAllHandlesArr(ByRef avMyHandles As Integer()) As Integer
```

Parameters

Name	Type and passing	Meaning
avMyHandles	ByRef avMyHandles As Integer()	Output Integer() containing the returned entity handles at indices 1 through the returned count. Index 0 is reserved.
ItemType	ByRef ItemType As AvaxItemType	An expression in which the user can input the type of object requested (ex: if all the text is requested then ItemType=text_i)
fzSort	ByRef fzSort As Boolean	A Boolean expression that if it is true then the array Handle(s) returns with the handles sorted by their z order

Use the declaration for the overload you call. ByRef arguments can receive outputs; declare variables and arrays with the exact indicated type. Optional defaults are shown in the declaration.

Return value

Integer - Is the total number of the handles

Example

Setup

Add an Avax.cAvax Windows Forms control named `cAvax1`. Paste the procedures into the form class. Example setup and supporting files explains references, initialization and file paths.

Add these Windows Forms controls or components to the form. Fields referenced by Handles clauses must be declared WithEvents, as the Visual Studio designer normally does.

Name	Type
Command1	Button
Text1	TextBox

VB.NET example

```
Private Sub Form1_Load(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs) Handles
    MyBase.Load
    Command1.Text = "GetAllHandlesArr"
    cAvax1.StartAvax()
    Call CAvax1.Add_Line(0, 5, 0, 10, 5, 0, Nothing, 1, 1, -1, -1)
    Call CAvax1.Add_Line(5, 0, 0, 5, 10, 0, Nothing, 1, 1, -1, -1)
    Call CAvax1.Add_Ellipse(5, 5, 0, 3, 2, Nothing, 5, 1, -1, -1)
End Sub
Private Sub Form1_FormClosed(ByVal eventSender As System.Object, ByVal eventArgs As
    System.Windows.Forms.FormClosedEventArgs) Handles Me.FormClosed
    cAvax1.EndAvax()
End Sub
Private Sub Command1_Click(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs)
    Handles Command1.Click
```

```
    Dim [Handles]() As Integer = Nothing
    Dim ItemType As Avax.AvaxItemType
    Dim fzSort As Boolean
    Dim AllHandles As Integer
    Dim EllipseCirclesHandles As Integer
    Dim i As Integer
    AllHandles = CAvax1.GetAllHandlesArr([Handles], ItemType, fzSort)
    Text1.Text = ""
    For i = 1 To AllHandles
        Text1.Text = Text1.Text & Str([Handles](i)) & vbCrLf
    Next i
    ItemType = Avax.AvaxItemType.Ellipse_1
    EllipseCirclesHandles = CAvax1.GetAllHandlesArr([Handles], ItemType, fzSort)
    MsgBox("This drawing has" & Str(AllHandles) & " objects and" & Str(EllipseCirclesHandles) & " of them
    is an ellipse ", MsgBoxStyle.Information, "GetAllHandlesArr")
End Sub
```

GetAllNodes

cAvax method

Purpose

Returns independent sorted lists of distinct X and Y node coordinates for visible objects, not paired XY vertices.

VB.NET declaration

```
Function GetAllNodes(ByRef x As Double(), ByRef y As Double()) As Boolean
```

Parameters

Name	Type and passing	Meaning
x	ByRef x As Double()	ByRef Double array of distinct ascending X coordinates, with reserved index zero. Independent of y; logical count is x.Length - 1, using Integer indexing. Failure on an unavailable control leaves the caller reference untouched.
y	ByRef y As Double()	ByRef Double array of distinct ascending Y coordinates, with reserved index zero. Its length can differ from x; do not index x and y as paired vertices. Empty active drawings assign {0} to each output and return False.

Use the declaration for the overload you call. ByRef arguments can receive outputs; declare variables and arrays with the exact indicated type. Optional defaults are shown in the declaration.

Return value

Boolean - True when visible drawing objects contribute coordinates. An empty active drawing returns False with {0} output arrays; an unavailable control returns False without replacing outputs.

Remarks

Each output is an independent, ascending, exactly deduplicated Double() with index 0 reserved. Their lengths can differ: never treat x(i), y(i) as a vertex pair. These coordinate lists support drawing dimension chains. Visible root objects contribute geometric nodes or their public property-envelope coordinates; hidden layers and non-drawing data are excluded. Blocks contribute envelope corners rather than all child vertices.

Returns True when any coordinates exist. An empty or entirely hidden active drawing returns False with fresh {0} arrays. A stopped, disabled or disposed control returns False without replacing outputs. External sources are refreshed before enumeration; a callback that replaces the drawing or changes its lifetime prevents stale results. The arrays are detached, and the query retains selection, construction input and Undo history.

Numeric changes from v2

Update the following variables, arrays or event-data receivers when porting the corresponding v2 call. The declarations above show the current overloads.

Parameter or value	v2 type	v3 type
x	Single()	Double()
y	Single()	Double()

For ByRef parameters, use a variable with the exact v3 type. Preserve array dimensions and the documented indexing. Moving from v2 to v3 explains conversion and legacy file-storage rules.

Example

Setup

Add an Avax.cAvax Windows Forms control named `cAvax1`. Paste the procedures into the form class. Example setup and supporting files explains references, initialization and file paths.

Add these Windows Forms controls or components to the form. Fields referenced by Handles clauses must be declared WithEvents, as the Visual Studio designer normally does.

Name	Type
Command1	Button
Text1	TextBox
Text2	TextBox

VB.NET example

```
Private Sub Form1_Load(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs) Handles
    MyBase.Load
    cAvax1.StartAvax()
    Call cAvax1.Add_Line(0, 0, 0, 10, 10, 0)
    Call cAvax1.Add_Line(7, 10, 0, 10, 10, 0)
    Call cAvax1.Add_Line(7, 0, 0, 14, 5, 0)
End Sub
Private Sub Form1_FormClosed(ByVal eventSender As System.Object, ByVal eventArgs As
    System.Windows.Forms.FormClosedEventArgs) Handles Me.FormClosed
    cAvax1.EndAvax()
End Sub
Private Sub Command1_Click(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs)
    Handles Command1.Click
    Dim x() As Double = Nothing
    Dim y() As Double = Nothing
    Dim i As Integer
    If cAvax1.GetAllNodes(x, y) = True Then
        For i = 1 To UBound(x)
            Text1.Text = Text1.Text & Str(x(i)) & vbCrLf
        Next i
        For i = 1 To UBound(y)
            Text2.Text = Text2.Text & Str(y(i)) & vbCrLf
        Next i
    End If
End Sub
```

GetArea

cAvax method

Purpose

Returns nonnegative object area projected onto XY, in squared drawing units; elevation and pen width do not contribute.

VB.NET declaration

```
Function GetArea(ByRef lAvaxHandle As Integer) As Double
```

Parameters

Name	Type and passing	Meaning
lAvaxHandle	ByRef lAvaxHandle As Integer	Existing entity handle (Integer, passed ByRef for R2 source compatibility; not modified). Missing, zero and negative handles return zero without changing the drawing.

Use the declaration for the overload you call. ByRef arguments can receive outputs; declare variables and arrays with the exact indicated type. Optional defaults are shown in the declaration.

Return value

Double - Nonnegative area in squared XY drawing units, or 0 for unsupported, invalid or unavailable input.

Remarks

Connected ordinary polylines and plain shape outlines are closed implicitly for area. Invalid ordinary boundaries, including self-intersections, backtracking or disconnected chains, return 0. A compound path includes only figures whose Closed property is True. Each figure closes independently; separate figures are never joined by an extra connecting side. Its EvenOddFill property selects even-odd or nonzero winding, so overlapping and self-intersecting figures contribute their actual filled region rather than the sum of separate absolute areas. Ellipses use their transformed axes. Circular and elliptical arcs contribute sector area. Native hatches use even-odd filling over the boundaries retained by IslandStyle: Normal keeps every loop, Outer keeps the exterior and first nested level, and Ignore keeps exterior regions. Nesting compares complete filled regions; touching is allowed, while equal or merely overlapping regions do not count as nested. All original hatch loops remain available for editing and saving. Sampled polylines are measured from their stored chords; analytic curves are measured from their curve definition.

The result is independent of display zoom, pen width and Z elevation. Points, lines, ordinary text/documents, blocks and unknown handles return 0. A started, enabled control is required. An invalid or out-of-range result reports an error and returns 0; a very small positive area can round to 0. The handle, drawing, selection and history are retained.

Numeric changes from v2

Update the following variables, arrays or event-data receivers when porting the corresponding v2 call. The declarations above show the current overloads.

Parameter or value	v2 type	v3 type
Return value	Single	Double

For ByRef parameters, use a variable with the exact v3 type. Preserve array dimensions and the documented indexing. Moving from v2 to v3 explains conversion and legacy file-storage rules.

Example

Setup

Add an Avax.cAvax Windows Forms control named `cAvax1`. Paste the procedures into the form class. Example setup and supporting files explains references, initialization and file paths.

Add these Windows Forms controls or components to the form. Fields referenced by Handles clauses must be declared WithEvents, as the Visual Studio designer normally does.

Name	Type
Command1	Button

VB.NET example

```
Dim h As Integer
Private Sub Form1_Load(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs) Handles
    MyBase.Load
    cAvax1.StartAvax()
    h = cAvax1.Add_Rectangle(0, 0, 10, 12)
    Call CAvax1.SetAvaxProperty(Avax.AvaxProp.DimScale_n, 5)
    Call cAvax1.Add_AllDimensions(2, 0)
End Sub
Private Sub Form1_FormClosed(ByVal eventSender As System.Object, ByVal eventArgs As
    System.Windows.Forms.FormClosedEventArgs) Handles Me.FormClosed
    cAvax1.EndAvax()
End Sub
Private Sub Command1_Click(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs)
    Handles Command1.Click
    Dim Answer As Double
    Answer = cAvax1.GetArea(h)
    MsgBox("Area =" & Str(Answer))
End Sub
```

GetAttributes

cAvax method

Purpose

Returns the attributes of the objects described in a specific handles array

See DoAvaxAction and public value structures for direct polyline closure, attribute/fill/text/stretch/status fields, widened numeric declarations and complete examples.

VB.NET declaration

```
Function GetAttributes(ByRef h As Integer(), ByRef iType As AvaxItemType(), ByRef iColor As Integer(),
    ByRef iWidth As Integer(), ByRef iStyle As Integer(), ByRef iLayer As Integer(), ByRef Elevation3D As
    Double(), ByRef Height3D As Double()) As Boolean
```

Parameters

Name	Type and passing	Meaning
h	ByRef h As Integer()	One-based input handle array. Index zero is ignored; order and duplicates are preserved.
iType	ByRef iType As AvaxItemType()	Independent one-based item-type output array. All outputs are published together only on success.
iColor	ByRef iColor As Integer()	Integer color indices, aligned with h.
iWidth	ByRef iWidth As Integer()	Integer line widths, aligned with h.
iStyle	ByRef iStyle As Integer()	Integer line/point styles, aligned with h.
iLayer	ByRef iLayer As Integer()	Integer layer indices, aligned with h.
Elevation3D	ByRef Elevation3D As Double()	Double stored elevation values, aligned with h.
Height3D	ByRef Height3D As Double()	Double stored 3D heights, aligned with h.

Use the declaration for the overload you call. ByRef arguments can receive outputs; declare variables and arrays with the exact indicated type. Optional defaults are shown in the declaration.

Return value

Boolean - True on success, false on error.

Numeric changes from v2

Update the following variables, arrays or event-data receivers when porting the corresponding v2 call. The declarations above show the current overloads.

Parameter or value	v2 type	v3 type
iColor	Short()	Integer()
iWidth	Short()	Integer()
iStyle	Short()	Integer()
iLayer	Short()	Integer()
Elevation3D	Single()	Double()
Height3D	Single()	Double()

For ByRef parameters, use a variable with the exact v3 type. Preserve array dimensions and the documented indexing. Moving from v2 to v3 explains conversion and legacy file-storage rules.

Example

Setup

Add an Avax.cAvax Windows Forms control named `cAvax1`. Paste the procedures into the form class. Example setup and supporting files explains references, initialization and file paths.

Add these Windows Forms controls or components to the form. Fields referenced by Handles clauses must be declared WithEvents, as the Visual Studio designer normally does.

Name	Type
Command1	Button

VB.NET example

```
Dim h(1) As Integer
Private Sub Form1_Load(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs) Handles
    MyBase.Load
    Command1.Text = "GetAttributes"
    cAvax1.StartAvax()
    h(1) = CAvax1.Add_Line(0, 0, 0, 10, 10, 0, Nothing, 5, 2, 1, 2)
End Sub
Private Sub Form1_FormClosed(ByVal eventSender As System.Object, ByVal eventArgs As
    System.Windows.Forms.FormClosedEventArgs) Handles Me.FormClosed
    cAvax1.EndAvax()
End Sub
Private Sub Command1_Click(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs)
    Handles Command1.Click
    Dim iType() As Avax.AvaxItemType = Nothing
    Dim iColor() As Integer = Nothing
    Dim iWidth() As Integer = Nothing
    Dim iStyle() As Integer = Nothing
    Dim iLayer() As Integer = Nothing
    Dim Elevation3D() As Double = Nothing
    Dim Height3D() As Double = Nothing
    Dim sMsg As String = String.Empty
    If CAvax1.GetAttributes(h, iType, iColor, iWidth, iStyle, iLayer, Elevation3D, Height3D) = True Then
        sMsg = "Type=" & iType(1) & vbCrLf
        sMsg = sMsg & "Color=" & iColor(1) & vbCrLf
        sMsg = sMsg & "Width=" & iWidth(1) & vbCrLf
        sMsg = sMsg & "Style=" & iStyle(1) & vbCrLf
        sMsg = sMsg & "Layer=" & iLayer(1) & vbCrLf
    End If
    MsgBox(sMsg, MsgBoxStyle.Information, "GetAttributes")
End Sub
```

GetAvaxEvents

cAvax method

Purpose

Synchronously processes pending control notifications and an already pending repaint on the owning UI thread.

VB.NET declaration

```
Function GetAvaxEvents() As Boolean
```

Return value

Boolean - True when a pending notification or repaint is delivered; False when nothing is delivered, including deferred or unavailable states. Exceptions from application handlers propagate.

Remarks

This method flushes pending selection, drawing-change and creation notifications and updates an already invalidated control surface. It does not manufacture keyboard, pointer, Before, progress or printing events, and it does not repeat ordinary API changes already notified synchronously. Call it on the control owning Windows Forms thread when pending notifications must be processed without waiting for the normal UI queue.

Active construction defers pending drawing notifications while retaining accepted points. BeginUpdate also retains the pending batch until EndUpdate. Returns False when idle, stopped, disabled, disposed, events-paused, inside BeginUpdate or recursively called from this dispatcher. Pending state is consumed before application callbacks, so recursive calls cannot replay it. Exceptions from handlers propagate. A replacement drawing, changed input or event-suppression change ends the older dispatch continuation.

Example

Setup

Add an Avax.cAvax Windows Forms control named `cAvax1`. Paste the procedures into the form class. Example setup and supporting files explains references, initialization and file paths.

Add these Windows Forms controls or components to the form. Fields referenced by Handles clauses must be declared WithEvents, as the Visual Studio designer normally does.

Name	Type
Command1	Button

VB.NET example

```
Private Sub Form1_Load(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs) Handles
    MyBase.Load
    Command1.Text = "GetAvaxEvents"
    cAvax1.StartAvax()
End Sub
Private Sub Form1_FormClosed(ByVal eventSender As System.Object, ByVal eventArgs As
    System.Windows.Forms.FormClosedEventArgs) Handles Me.FormClosed
    cAvax1.EndAvax()
End Sub
Private Sub Command1_Click(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs)
    Handles Command1.Click
    MsgBox(cAvax1.GetAvaxEvents(), MsgBoxStyle.Information, "GetAvaxEvents")
End Sub
```

GetAvaxItemType

Returns the same type information through the original ByRef calling signature. The handle is an input and is not modified.

```
Public Function GetAvaxItemType(ByRef hAvax As Integer) As Avax.AvaxItemType
```

Parameter	Meaning
hAvax	An Integer variable containing the object's handle. Its value is preserved.

Related options and reference

GetAvaxProperty

cAvax method

Purpose

Returns the value of a specific AVAX property (AvaxProp)

See Drawing coordinate limits for Double coordinate properties, individual versus complete work-area assignment, and transactional settings validation.

See Idle selection and spatial queries for automatic selection, filled-shape picking, one-based Integer query results, availability and Double-coordinate areas.

See Arc commands and segmentation for the three arc pick orders, standalone completion, polyline continuation and arc resolution settings.

VB.NET declaration

```
Function GetAvaxProperty(ByRef iProp As AvaxProp) As Object
```

Parameters

Name	Type and passing	Meaning
iProp	ByRef iProp As AvaxProp	The AvaxProp identifier to read. It is not changed by this call. The returned Object contains the documented type of the selected parameter.

Use the declaration for the overload you call. ByRef arguments can receive outputs; declare variables and arrays with the exact indicated type. Optional defaults are shown in the declaration.

Return value

Variant - Is the value of the specific AVAX property

Remarks

Returns the value identified by an AvaxProp member. The return type is Object: convert it to the documented type for that parameter. In v3, floating-point values such as PrintPaperX_MM_p are Double and former Short values are Integer. A parameter with a corresponding direct property, such as Language_p or ColorPenIndex_p, reads the same per-control setting. See AvaxProp for the available parameters.

Example

Setup

Add an Avax.cAvax Windows Forms control named `cAvax1`. Paste the procedures into the form class. Example setup and supporting files explains references, initialization and file paths.

Add these Windows Forms controls or components to the form. Fields referenced by Handles clauses must be declared WithEvents, as the Visual Studio designer normally does.

Name	Type
Command1	Button

VB.NET example

```
Private Sub Form1_Load(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs) Handles
    MyBase.Load
    Command1.Text = "GetAvaxProperty"
    cAvax1.StartAvax()
End Sub
Private Sub Form1_FormClosed(ByVal eventSender As System.Object, ByVal eventArgs As
    System.Windows.Forms.FormClosedEventArgs) Handles Me.FormClosed
```

```
    cAvax1.EndAvax()
End Sub
Private Sub Command1_Click(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs)
    Handles Command1.Click
    MsgBox(CAvax1.GetAvaxProperty(Avax.AvaxProp.PrintPaperX_MM,0) & " mm")
End Sub
```

GetAvaxStatus

cAvax method

Purpose

Returns with the status of AVAX

See Idle selection and spatial queries for automatic selection, filled-shape picking, one-based Integer query results, availability and Double-coordinate areas.

See DoAvaxAction and public value structures for direct polyline closure, attribute/fill/text/stretch/status fields, widened numeric declarations and complete examples.

VB.NET declarations

```
Function GetAvaxStatus(ByRef zoomon As Integer, ByRef selecton As Integer, ByRef rotateon As Integer,
    ByRef mirroron As Integer, ByRef scaleon As Integer, ByRef StretchOn As Integer, ByRef moveon As Integer,
    ByRef lineon As Integer, ByRef OsnapOn As Integer, ByRef intersecon As Integer, ByRef SelectVerticesOn As
    Integer, ByRef SnapOn As Integer, ByRef OrthoOn As Integer, ByRef GridOn As Integer, ByRef SelectAddOn As
    Integer, ByRef pointOn As Integer) As Boolean

Function GetAvaxStatus() As Boolean
```

Parameters

Name	Type and passing	Meaning
zoomon	ByRef zoomon As Integer	Integer ByRef output (formerly Integer). Engine zoom-window code: 1 for the Zoom tool, 2 for a selection window. Managed Pan/Dynamic Zoom can report zero here while ActionInProgress is True. Unchanged when the method returns False.
selecton	ByRef selecton As Integer	Integer ByRef output (formerly Integer). Selection progress; 2 for point selection and 3 for a selection window. Independent of scaleon. Unchanged when the method returns False.
rotateon	ByRef rotateon As Integer	Integer ByRef output (formerly Integer). Rotation progress, including an active rotation grip. Unchanged when the method returns False.
mirroron	ByRef mirroron As Integer	Integer ByRef output (formerly Integer). Mirror progress. Unchanged when the method returns False.
scaleon	ByRef scaleon As Integer	Integer ByRef output (formerly Integer). Scale progress. Unchanged when the method returns False.
StretchOn	ByRef StretchOn As Integer	Integer ByRef output (formerly Integer). Stretch progress, including a resize grip. Unchanged when the method returns False.
moveon	ByRef moveon As Integer	Integer ByRef output (formerly Integer). Move progress; also set during stretch. Copy reports lineon=2 with moveon=0. Unchanged when the method returns False.
lineon	ByRef lineon As Integer	Integer ByRef output (formerly Integer). Construction mode:1 for continuous geometric construction,2 for point/transform requests,including a triangle based on a selected line. Zero while Wall/Polygon parameters are awaiting acceptance. Unchanged when the method returns False.
OsnapOn	ByRef OsnapOn As Integer	Integer ByRef output (formerly Integer). Pending explicit AvaxOSnapType code:0 for no request,1..9 for original modes and10 for the v3 Center extension. It is not an automatic bitmask or last resolved candidate. Unchanged when the method returns False.
intersecon	ByRef intersecon As Integer	Integer ByRef output (formerly Integer). 1 while an explicit intersection snap is pending. Unchanged when the method returns False.
SelectVerticesOn	ByRef SelectVerticesOn As Integer	Integer ByRef output (formerly Integer). Vertex selection/point-request progress. Unchanged when the method returns False.
SnapOn	ByRef SnapOn As Integer	Integer ByRef output (formerly Integer). Snap-to-step enabled state. Unchanged when the method returns False.
OrthoOn	ByRef OrthoOn As Integer	Integer ByRef output (formerly Integer). Orthogonal input enabled state. Unchanged when the method returns False.
GridOn	ByRef GridOn As Integer	Integer ByRef output (formerly Integer). Grid display enabled state. Unchanged when the method returns False.
SelectAddOn	ByRef SelectAddOn As Integer	Integer ByRef output (formerly Integer). Persistent selection mode: 1 for add, 0 for subtract. SelectAdd_c/SelectSub_c change it; ordinary Select_c, Cancel and other drawing/view commands retain it. A new drawing starts in add mode; Trim/Extend force add after valid boundary selection. Unchanged when the method returns False.
pointOn	ByRef pointOn As Integer	Integer ByRef output (formerly Integer). Point entry or requested-point progress,including the Double remaining point for a triangle based on a selected line. Zero before Wall/Polygon parameter acceptance. Unchanged when the method returns False.

Use the declaration for the overload you call. ByRef arguments can receive outputs; declare variables and arrays with the exact indicated type. Optional defaults are shown in the declaration.

Return value

Boolean - True on success, false on error.

Remarks

Returns whether a current control-status snapshot is available. The parameterless overload checks the same availability as the sixteen-output overload; it does not report activity. The full overload returns Integer mode codes, not sixteen Boolean values. Outputs are unchanged when the method returns False, including before startup or while disabled, ended or disposed. Pausing input or events does not hide a readable status. selecton is independent of scaleon: point selection uses2 and window selection uses3, with zoomon=2 during a selection window. The ordinary Zoom tool uses zoomon=1; legacy plot-file window selection uses2. The legacy Print window is a two-point request with lineon=2, pointOn=1 and SelectVerticesOn=1. Managed Pan/Dynamic can report zoomon=0 while ActionInProgress is True. lineon=1 describes continuous geometric construction; lineon=2 describes point/transform requests. Prepared Text, Insert, CAD export, Block and Offset use lineon=2/pointOn=1. Array and Break use lineon=2/pointOn=0. Angle and Dim use2/1 when a selected guide is prepared, and1/0 for ordinary point construction. A triangle based on a selected line uses2/1. Copy keeps moveon=0; Move sets it, and Stretch sets both moveon and StretchOn. During new Text, import/export/block file selection and Wall/Polygon/Array/Offset parameter setup, construction flags remain zero and submitted construction points are rejected. Grid/Snap configuration retains the underlying drafting status. SnapOn, OrthoOn and GridOn describe their independent enabled settings. OsnapOn is the pending one-shot enum code, not an automatic bitmask or last resolved candidate. SelectAddOn is persistent:0 subtracts and1 adds, surviving ordinary Select/Cancel/drawing/view commands until explicitly changed or reset by a new drawing. Status queries do not consume coordinates, create entities or raise drawing events.

Numeric changes from v2

Update the following variables, arrays or event-data receivers when porting the corresponding v2 call. The declarations above show the current overloads.

Parameter or value	v2 type	v3 type
zoomon	Short	Integer
selecton	Short	Integer
rotateon	Short	Integer
mirroron	Short	Integer
scaleon	Short	Integer
StretchOn	Short	Integer
moveon	Short	Integer
lineon	Short	Integer
OsnapOn	Short	Integer
intersecon	Short	Integer
SelectVerticesOn	Short	Integer
SnapOn	Short	Integer
OrthoOn	Short	Integer
GridOn	Short	Integer
SelectAddOn	Short	Integer
pointOn	Short	Integer

For ByRef parameters, use a variable with the exact v3 type. Preserve array dimensions and the documented indexing. Moving from v2 to v3 explains conversion and legacy file-storage rules.

Example

Setup

Add an Avax.cAvax Windows Forms control named `cAvax1`. Paste the procedures into the form class. Example setup and supporting files explains references, initialization and file paths.

Add these Windows Forms controls or components to the form. Fields referenced by Handles clauses must be declared WithEvents, as the Visual Studio designer normally does.

Name	Type
Command1	Button
Label1	Label

VB.NET example

```
Private Sub CAvax1_OnMouseMove(ByVal Sender As Object, ByVal e As Avax.cAvax.MouseMoveEventArgs) Handles
    CAvax1.OnMouseMove
    Dim zoomon As Integer
    If CAvax1.GetAvaxStatus(zoomon, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0) = True Then
        Label1.Text = "ZoomOn=" & zoomon
    End If
End Sub
Private Sub Form1_Load(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs) Handles
    MyBase.Load
    Command1.Text = "Zoom"
    Label1.Text = "Zoom status"
    cAvax1.StartAvax()
End Sub
Private Sub Form1_FormClosed(ByVal eventSender As System.Object, ByVal eventArgs As
    System.Windows.Forms.FormClosedEventArgs) Handles Me.FormClosed
    cAvax1.EndAvax()
End Sub
Private Sub Command1_Click(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs)
    Handles Command1.Click
    CAvax1.Command = Avax.AvaxCommand.Zoom_c
End Sub
```

GetBorder

cAvax method

Purpose

Returns the combined XYZ bounding box of objects in a one-based handle array, using their current geometry.

VB.NET declaration

```
Function GetBorder(ByRef avMyHandles As Integer(), ByRef xMin As Double, ByRef yMin As Double, ByRef zMin
    As Double, ByRef xMax As Double, ByRef yMax As Double, ByRef zMax As Double) As Boolean
```

Parameters

Name	Type and passing	Meaning
avMyHandles	ByRef avMyHandles As Integer()	One-based Integer array of existing handles. Index zero is reserved; array and elements are not changed.
xMin	ByRef xMin As Double	Output xMin coordinate in Double world units. Unchanged on failure.
yMin	ByRef yMin As Double	Output yMin coordinate in Double world units. Unchanged on failure.
zMin	ByRef zMin As Double	Output zMin coordinate in Double world units. Unchanged on failure.
xMax	ByRef xMax As Double	Output xMax coordinate in Double world units. Unchanged on failure.
yMax	ByRef yMax As Double	Output yMax coordinate in Double world units. Unchanged on failure.
zMax	ByRef zMax As Double	Output zMax coordinate in Double world units. Unchanged on failure.

Use the declaration for the overload you call. ByRef arguments can receive outputs; declare variables and arrays with the exact indicated type. Optional defaults are shown in the declaration.

Return value

Boolean - True on success, false on error

Remarks

The result combines current object geometry, including nested block children and resolved external lines. Data records have no extents. Analytic curve extrema and document layouts contribute to the bounds; pen width, extrusion and selection-frame padding are excluded. Duplicate handles are harmless, but any missing handle, an empty array or a set with no drawable geometry returns False. External sources can invoke SetExternalLines; replacing the drawing or invalidating the control during that callback prevents stale output.

Coordinates are Double drawing units. A started, enabled control is required. Failure leaves all ByRef outputs unchanged. Reading bounds does not change selection or add an Undo operation.

Numeric changes from v2

Update the following variables, arrays or event-data receivers when porting the corresponding v2 call. The declarations above show the current overloads.

Parameter or value	v2 type	v3 type
xMin	Single	Double
yMin	Single	Double
zMin	Single	Double
xMax	Single	Double
yMax	Single	Double

Parameter or value	v2 type	v3 type
zMax	Single	Double

For ByRef parameters, use a variable with the exact v3 type. Preserve array dimensions and the documented indexing. Moving from v2 to v3 explains conversion and legacy file-storage rules.

Example

Setup

Add an Avax.cAvax Windows Forms control named `cAvax1`. Paste the procedures into the form class. Example setup and supporting files explains references, initialization and file paths.

Add these Windows Forms controls or components to the form. Fields referenced by Handles clauses must be declared WithEvents, as the Visual Studio designer normally does.

Name	Type
Command1	Button

VB.NET example

```
Dim h(2) As Integer
Private Sub Form1_Load(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs) Handles
    MyBase.Load
    Call cAvax1.StartAvax()
    h(1) = cAvax1.Add_Line(0, 0, 0, 10, 12, 0)
    h(2) = cAvax1.Add_Line(0, 10, 0, 10, 0, 0)
End Sub
Private Sub Form1_FormClosed(ByVal eventSender As System.Object, ByVal eventArgs As
    System.Windows.Forms.FormClosedEventArgs) Handles Me.FormClosed
    Call cAvax1.EndAvax()
End Sub
Private Sub Command1_Click(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs)
    Handles Command1.Click
    Dim xMin As Double
    Dim yMin As Double
    Dim zMin As Double
    Dim xMax As Double
    Dim yMax As Double
    Dim zMax As Double
    Dim Answer As Boolean
    Answer = cAvax1.GetBorder(h, xMin, yMin, zMin, xMax, yMax, zMax)
    Call cAvax1.Add_Rectangle(xMin, yMin, xMax - xMin, yMax - yMin, Nothing, 1, -1, -1, -1)
End Sub
```

GetBoundsAvax

cAvax method

Purpose

Returns XYZ limits of the total drawing work area, independently of object bounds and the current view.

VB.NET declaration

```
Function GetBoundsAvax(ByRef xMin As Double, ByRef yMin As Double, ByRef zMin As Double, ByRef xMax As
    Double, ByRef yMax As Double, ByRef zMax As Double) As Boolean
```

Parameters

Name	Type and passing	Meaning
xMin	ByRef xMin As Double	Output xMin coordinate in Double world units. Unchanged on failure.
yMin	ByRef yMin As Double	Output yMin coordinate in Double world units. Unchanged on failure.
zMin	ByRef zMin As Double	Output zMin coordinate in Double world units. Unchanged on failure.
xMax	ByRef xMax As Double	Output xMax coordinate in Double world units. Unchanged on failure.
yMax	ByRef yMax As Double	Output yMax coordinate in Double world units. Unchanged on failure.
zMax	ByRef zMax As Double	Output zMax coordinate in Double world units. Unchanged on failure.

Use the declaration for the overload you call. ByRef arguments can receive outputs; declare variables and arrays with the exact indicated type. Optional defaults are shown in the declaration.

Return value

Boolean - True on success, false on error.

Remarks

SetBoundsAvax, SetBoundsDrawing and CalcBoundsDrawing can change the total work area. It is independent of cached drawing extents and the current view. Native AVAX stores these three states separately. Legacy AVX stores the work area and uses it as the view when reopened; BLK uses that header area for its insertion base. OnLoadAutoLimitsOn=True recalculates work bounds during loading.

Coordinates are Double drawing units. A started, enabled control is required. Failure leaves all ByRef outputs unchanged. Reading bounds does not change selection or add an Undo operation.

Numeric changes from v2

Update the following variables, arrays or event-data receivers when porting the corresponding v2 call. The declarations above show the current overloads.

Parameter or value	v2 type	v3 type
xMin	Single	Double
yMin	Single	Double
zMin	Single	Double
xMax	Single	Double
yMax	Single	Double
zMax	Single	Double

For ByRef parameters, use a variable with the exact v3 type. Preserve array dimensions and the documented indexing. Moving from v2 to v3 explains conversion and legacy file-storage rules.

Example

Setup

Add an Avax.cAvax Windows Forms control named `cAvax1`. Paste the procedures into the form class. Example setup and supporting files explains references, initialization and file paths.

Add these Windows Forms controls or components to the form. Fields referenced by Handles clauses must be declared WithEvents, as the Visual Studio designer normally does.

Name	Type
Command1	Button
Text1	TextBox

VB.NET example

```
Private Sub Form1_Load(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs) Handles
    MyBase.Load
    cAvax1.StartAvax()
End Sub
Private Sub Form1_FormClosed(ByVal eventSender As System.Object, ByVal eventArgs As
    System.Windows.Forms.FormClosedEventArgs) Handles Me.FormClosed
    cAvax1.EndAvax()
End Sub
Private Sub Command1_Click(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs)
    Handles Command1.Click
    Dim xMin As Double
    Dim yMin As Double
    Dim zMin As Double
    Dim xMax As Double
    Dim yMax As Double
    Dim zMax As Double
    Dim Answer As Boolean
    Dim txt As String = String.Empty
    Answer = cAvax1.GetBoundsAvax(xMin, yMin, zMin, xMax, yMax, zMax)
    txt = "Xmin =" & Str(xMin) & vbCrLf
    txt = txt & "Xmax =" & Str(xMax) & vbCrLf
    txt = txt & "Ymin =" & Str(yMin) & vbCrLf
    txt = txt & "Ymax =" & Str(yMax) & vbCrLf
    Text1.Text = txt
End Sub
```

GetBoundsDrawing

cAvax method

Purpose

Returns cached XYZ drawing bounds. Call CalcBoundsDrawing first to include recent geometry changes.

VB.NET declaration

```
Function GetBoundsDrawing(ByRef xMin As Double, ByRef yMin As Double, ByRef zMin As Double, ByRef xMax As
    Double, ByRef yMax As Double, ByRef zMax As Double) As Boolean
```

Parameters

Name	Type and passing	Meaning
xMin	ByRef xMin As Double	Output xMin coordinate in Double world units. Unchanged on failure.
yMin	ByRef yMin As Double	Output yMin coordinate in Double world units. Unchanged on failure.
zMin	ByRef zMin As Double	Output zMin coordinate in Double world units. Unchanged on failure.
xMax	ByRef xMax As Double	Output xMax coordinate in Double world units. Unchanged on failure.
yMax	ByRef yMax As Double	Output yMax coordinate in Double world units. Unchanged on failure.
zMax	ByRef zMax As Double	Output zMax coordinate in Double world units. Unchanged on failure.

Use the declaration for the overload you call. ByRef arguments can receive outputs; declare variables and arrays with the exact indicated type. Optional defaults are shown in the declaration.

Return value

Boolean - True on success, false on error.

Remarks

Adding, editing or deleting objects does not automatically recalculate this snapshot. CalcBoundsDrawing calculates fresh geometry bounds; SetBoundsDrawing assigns a snapshot explicitly. A fresh control starts with (-5,-5,0) to (20,20,0). Recalculating an empty drawing produces six zeros.

Native AVAX and legacy AVX preserve these cached bounds, including Z. OnLoadAutoLimitsOn=True recalculates bounds during loading; set it to False to retain the saved bounds and view.

Coordinates are Double drawing units. A started, enabled control is required. Failure leaves all ByRef outputs unchanged. Reading bounds does not change selection or add an Undo operation.

Numeric changes from v2

Update the following variables, arrays or event-data receivers when porting the corresponding v2 call. The declarations above show the current overloads.

Parameter or value	v2 type	v3 type
xMin	Single	Double
yMin	Single	Double
zMin	Single	Double
xMax	Single	Double
yMax	Single	Double
zMax	Single	Double

For ByRef parameters, use a variable with the exact v3 type. Preserve array dimensions and the documented indexing. Moving from v2 to v3 explains conversion and legacy file-storage rules.

Example

Setup

Add an Avax.cAvax Windows Forms control named `cAvax1`. Paste the procedures into the form class. Example setup and supporting files explains references, initialization and file paths.

Add these Windows Forms controls or components to the form. Fields referenced by Handles clauses must be declared WithEvents, as the Visual Studio designer normally does.

Name	Type
Command1	Button
Text1	TextBox

VB.NET example

```
Private Sub Form1_Load(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs) Handles
    MyBase.Load
    cAvax1.StartAvax()
End Sub
Private Sub Form1_FormClosed(ByVal eventSender As System.Object, ByVal eventArgs As
    System.Windows.Forms.FormClosedEventArgs) Handles Me.FormClosed
    cAvax1.EndAvax()
End Sub
Private Sub Command1_Click(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs)
    Handles Command1.Click
    Dim xMin As Double
    Dim yMin As Double
    Dim zMin As Double
    Dim xMax As Double
    Dim yMax As Double
    Dim zMax As Double
    Dim txt As String = String.Empty
    If cAvax1.GetBoundsDrawing(xMin, yMin, zMin, xMax, yMax, zMax) = True Then
        txt = "Xmin =" & Str(xMin) & vbCr & vbLf
        txt = txt & "Xmax =" & Str(xMax) & vbCr & vbLf
        txt = txt & "Ymin =" & Str(yMin) & vbCr & vbLf
        txt = txt & "Ymax =" & Str(yMax) & vbCr & vbLf
        Text1.Text = txt
    End If
End Sub
```

GetBoundsSelectBox

cAvax method

Purpose

Returns the coordinates of the select box pointer

VB.NET declaration

```
Function GetBoundsSelectBox(ByRef xMinSel As Double, ByRef yMinSel As Double, ByRef xMaxSel As Double,
    ByRef yMaxSel As Double) As Boolean
```

Parameters

Name	Type and passing	Meaning
xMinSel	ByRef xMinSel As Double	Double world coordinate of the pixel pickbox clipped to the drawable canvas. Unchanged when the pointer is outside or the control cannot answer.
yMinSel	ByRef yMinSel As Double	Double world coordinate of the pixel pickbox clipped to the drawable canvas. Unchanged when the pointer is outside or the control cannot answer.
xMaxSel	ByRef xMaxSel As Double	Double world coordinate of the pixel pickbox clipped to the drawable canvas. Unchanged when the pointer is outside or the control cannot answer.
yMaxSel	ByRef yMaxSel As Double	Double world coordinate of the pixel pickbox clipped to the drawable canvas. Unchanged when the pointer is outside or the control cannot answer.

Use the declaration for the overload you call. ByRef arguments can receive outputs; declare variables and arrays with the exact indicated type. Optional defaults are shown in the declaration.

Return value

Boolean - True on success, false on error.

Numeric changes from v2

Update the following variables, arrays or event-data receivers when porting the corresponding v2 call. The declarations above show the current overloads.

Parameter or value	v2 type	v3 type
xMinSel	Single	Double
yMinSel	Single	Double
xMaxSel	Single	Double
yMaxSel	Single	Double

For ByRef parameters, use a variable with the exact v3 type. Preserve array dimensions and the documented indexing. Moving from v2 to v3 explains conversion and legacy file-storage rules.

Example

Setup

Add an Avax.cAvax Windows Forms control named `cAvax1`. Paste the procedures into the form class. Example setup and supporting files explains references, initialization and file paths.

Add these Windows Forms controls or components to the form. Fields referenced by Handles clauses must be declared WithEvents, as the Visual Studio designer normally does.

Name	Type
Text1	TextBox

VB.NET example

```
Private Sub Form1_Load(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs) Handles
    MyBase.Load
    cAvax1.StartAvax()
End Sub
Private Sub Form1_FormClosed(ByVal eventSender As System.Object, ByVal eventArgs As
    System.Windows.Forms.FormClosedEventArgs) Handles Me.FormClosed
    cAvax1.EndAvax()
End Sub
Private Sub CAvax1_OnMouseMove(ByVal Sender As Object, ByVal e As Avax.cAvax.MouseMoveEventArgs) Handles
    CAvax1.OnMouseMove
    Dim xMinSel As Double
    Dim yMinSel As Double
    Dim xMaxSel As Double
    Dim yMaxSel As Double
    Dim Answer As Boolean
    Dim txt As String = String.Empty
    Answer = CAvax1.GetBoundsSelectBox(xMinSel, yMinSel, xMaxSel, yMaxSel)
    txt = "Xmin =" & Str(xMinSel) & vbCr & vbLf
    txt = txt & "Xmax =" & Str(xMaxSel) & vbCr & vbLf
    txt = txt & "Ymin =" & Str(yMinSel) & vbCr & vbLf
    txt = txt & "Ymax =" & Str(yMaxSel) & vbCr & vbLf
    Text1.Text = txt
End Sub
```

GetBoundsSelectedItems

cAvax method

Purpose

Returns the combined XYZ geometry bounds of selected objects, excluding screen-frame padding and grips.

VB.NET declaration

```
Function GetBoundsSelectedItems(ByRef xMin As Double, ByRef yMin As Double, ByRef zMin As Double, ByRef
    xMax As Double, ByRef yMax As Double, ByRef zMax As Double) As Boolean
```

Parameters

Name	Type and passing	Meaning
xMin	ByRef xMin As Double	Output xMin coordinate in Double world units. Unchanged on failure.
yMin	ByRef yMin As Double	Output yMin coordinate in Double world units. Unchanged on failure.
zMin	ByRef zMin As Double	Output zMin coordinate in Double world units. Unchanged on failure.
xMax	ByRef xMax As Double	Output xMax coordinate in Double world units. Unchanged on failure.
yMax	ByRef yMax As Double	Output yMax coordinate in Double world units. Unchanged on failure.
zMax	ByRef zMax As Double	Output zMax coordinate in Double world units. Unchanged on failure.

Use the declaration for the overload you call. ByRef arguments can receive outputs; declare variables and arrays with the exact indicated type. Optional defaults are shown in the declaration.

Return value

Boolean - True on success, false on error.

Remarks

The result is the union of selected object geometry using the same extents rules as GetBorder. It excludes the visual selection-frame margin and grips. An empty selection returns False. External line sources can invoke their resolver; replacing the drawing or invalidating the control prevents stale output.

Coordinates are Double drawing units. A started, enabled control is required. Failure leaves all ByRef outputs unchanged. Reading bounds does not change selection or add an Undo operation.

Numeric changes from v2

Update the following variables, arrays or event-data receivers when porting the corresponding v2 call. The declarations above show the current overloads.

Parameter or value	v2 type	v3 type
xMin	Single	Double
yMin	Single	Double
zMin	Single	Double
xMax	Single	Double
yMax	Single	Double
zMax	Single	Double

For ByRef parameters, use a variable with the exact v3 type. Preserve array dimensions and the documented indexing. Moving from v2 to v3 explains conversion and legacy file-storage rules.

Example

Setup

Add an Avax.cAvax Windows Forms control named `cAvax1`. Paste the procedures into the form class. Example setup and supporting files explains references, initialization and file paths.

Add these Windows Forms controls or components to the form. Fields referenced by Handles clauses must be declared WithEvents, as the Visual Studio designer normally does.

Name	Type
Command1	Button
Command2	Button
Command3	Button
Text1	TextBox

VB.NET example

```
Private Sub Command2_Click(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs)
    Handles Command2.Click
    CAvax1.Command = Avax.AvaxCommand.Circle2p_c
End Sub
Private Sub Command3_Click(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs)
    Handles Command3.Click
    CAvax1.Command = Avax.AvaxCommand.Select_All_c
End Sub
Private Sub Form1_Load(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs) Handles
    MyBase.Load
    cAvax1.StartAvax()
End Sub
Private Sub Form1_FormClosed(ByVal eventSender As System.Object, ByVal eventArgs As
    System.Windows.Forms.FormClosedEventArgs) Handles Me.FormClosed
    cAvax1.EndAvax()
End Sub
Private Sub Command1_Click(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs)
    Handles Command1.Click
    Dim xMin As Double
    Dim yMin As Double
    Dim zMin As Double
    Dim xMax As Double
    Dim yMax As Double
    Dim zMax As Double
    Dim txt As String = String.Empty
    If cAvax1.GetBoundsSelectedItems(xMin, yMin, zMin, xMax, yMax, zMax) = True Then
        txt = "Xmin =" & Str(xMin) & vbCr & vbLf
        txt = txt & "Xmax =" & Str(xMax) & vbCr & vbLf
        txt = txt & "Ymin =" & Str(yMin) & vbCr & vbLf
        txt = txt & "Ymax =" & Str(yMax) & vbCr & vbLf
    End If
    Text1.Text = txt
End Sub
```

GetBoundsZoom

cAvax method

Purpose

Returns the XY drawing-coordinate rectangle represented by the current view.

VB.NET declaration

```
Function GetBoundsZoom(ByRef zxMin As Double, ByRef zyMin As Double, ByRef zxMax As Double, ByRef zyMax As
     Double) As Boolean
```

Parameters

Name	Type and passing	Meaning
zxMin	ByRef zxMin As Double	Current effective XY view coordinate (Double); unchanged on failure.
zyMin	ByRef zyMin As Double	Current effective XY view coordinate (Double); unchanged on failure.
zxMax	ByRef zxMax As Double	Current effective XY view coordinate (Double); unchanged on failure.
zyMax	ByRef zyMax As Double	Current effective XY view coordinate (Double); unchanged on failure.

Use the declaration for the overload you call. ByRef arguments can receive outputs; declare variables and arrays with the exact indicated type. Optional defaults are shown in the declaration.

Return value

Boolean - True on success, false on error.

Remarks

The returned rectangle is the effective world-coordinate view, including aspect-ratio and work-area adjustments made by SetBoundsZoom. Previous restores zoom history. Native AVAX can retain a separate view when OnLoadAutoLimitsOn=False; legacy AVX reopens using its saved work-area view.

Coordinates are Double drawing units. A started, enabled control is required. Failure leaves all ByRef outputs unchanged. Reading bounds does not change selection or add an Undo operation.

Numeric changes from v2

Update the following variables, arrays or event-data receivers when porting the corresponding v2 call. The declarations above show the current overloads.

Parameter or value	v2 type	v3 type
zxMin	Single	Double
zyMin	Single	Double
zxMax	Single	Double
zyMax	Single	Double

For ByRef parameters, use a variable with the exact v3 type. Preserve array dimensions and the documented indexing. Moving from v2 to v3 explains conversion and legacy file-storage rules.

Example

Setup

Add an Avax.cAvax Windows Forms control named `cAvax1`. Paste the procedures into the form class. Example setup and supporting files explains references, initialization and file paths.

Add these Windows Forms controls or components to the form. Fields referenced by Handles clauses must be declared WithEvents, as the Visual Studio designer normally does.

Name	Type
Command1	Button
Command2	Button
Command3	Button
Text1	TextBox

VB.NET example

```
Private Sub Form1_Load(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs) Handles
    MyBase.Load
    cAvax1.StartAvax()
    Call cAvax1.Add_Line(0, 0, 0, 10, 10, 0)
End Sub
Private Sub Form1_FormClosed(ByVal eventSender As System.Object, ByVal eventArgs As
    System.Windows.Forms.FormClosedEventArgs) Handles Me.FormClosed
    cAvax1.EndAvax()
End Sub
Private Sub Command1_Click(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs)
    Handles Command1.Click
    Dim zxMin As Double
    Dim zyMin As Double
    Dim zxMax As Double
    Dim zyMax As Double
    Dim Answer As Boolean
    Dim txt As String = String.Empty
    Answer = cAvax1.GetBoundsZoom(zxMin, zyMin, zxMax, zyMax)
    txt = "Xmin =" & Str(zxMin) & vbCr & vbLf
    txt = txt & "Xmax =" & Str(zxMax) & vbCr & vbLf
    txt = txt & "Ymin =" & Str(zyMin) & vbCr & vbLf
    txt = txt & "Ymax =" & Str(zyMax) & vbCr & vbLf
    Text1.Text = txt
End Sub
Private Sub Command2_Click(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs)
    Handles Command2.Click
    CAvax1.Command = Avax.AvaxCommand.Previous_c
End Sub
Private Sub Command3_Click(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs)
    Handles Command3.Click
    CAvax1.Command = Avax.AvaxCommand.Zoom_c
End Sub
```

GetCompoundPath

Purpose

Reads an independent path definition from a compound path, a parametric shape or a supported vector curve in the XY drawing plane. Supported source geometry includes ordinary lines and polylines, cubic Bezier curves, circular arcs, ellipses and connected analytic polyline boundaries. Reading does not convert or modify the source object.

Declaration

```
Function GetCompoundPath(ByVal handle As Integer) As AvaxPathData
```

Parameters and result

`handle` is the drawing object's Integer handle. Returns an independent AvaxPathData value, or Nothing when the handle is absent, its geometry cannot be read as a path, or the control is unstarted or disposed. Reading is also available while the control is disabled. Geometry outside the XY drawing plane is not projected into 2D. Modifying the returned value does not change the object; pass it to ChangeCompoundPath to commit an edit. The path value table describes its fields.

VB.NET example

```
Option Strict On
Public Module GetCompoundPathExample
    Public Sub Describe(cad As Avax.cAvax, handle As Integer)
        Dim path As Avax.AvaxPathData = cad.GetCompoundPath(handle)
        If path Is Nothing Then Return
        For Each figure As Avax.AvaxPathFigure In path.Figures
            System.Diagnostics.Debug.WriteLine(
                "Start: " & figure.X & ", " & figure.Y)
            For Each segment As Avax.AvaxPathSegment In figure.Segments
                System.Diagnostics.Debug.WriteLine(segment.Kind.ToString())
            Next
        Next
    End Sub
End Module
```

Related options and reference

GetCurveLines

cAvax method

Purpose

Use this method if you want to get a curve object as an array of lines

VB.NET declaration

```
Function GetCurveLines(ByRef hCurve As Integer, ByRef x1 As Double(), ByRef y1 As Double(), ByRef z1 As
    Double(), ByRef x2 As Double(), ByRef y2 As Double(), ByRef z2 As Double()) As Integer
```

Parameters

Name	Type and passing	Meaning
hCurve	ByRef hCurve As Integer	Is the handle of the curve
x1	ByRef x1 As Double()	Is an array of the x coordinates of the lines first point
y1	ByRef y1 As Double()	Is an array of the y coordinates of the lines first point
z1	ByRef z1 As Double()	Is an array of the z coordinates of the lines first point
x2	ByRef x2 As Double()	Is an array of the x coordinates of the lines second point
y2	ByRef y2 As Double()	Is an array of the y coordinates of the lines second point
z2	ByRef z2 As Double()	Is an array of the z coordinates of the lines first point

Use the declaration for the overload you call. ByRef arguments can receive outputs; declare variables and arrays with the exact indicated type. Optional defaults are shown in the declaration.

Return value

Integer - Returns the upper bound of the arrays

Remarks

Returns the number of line segments extracted from a Curve_i object. Each of the six output arrays is a Double() with indices 1 through the returned count; index zero is reserved and set to zero. x1/y1/z1 contain segment starts and x2/y2/z2 contain segment ends. Adjacent segments share the same XYZ endpoint. The arrays are independent copies; editing them does not change the source curve. For a circular arc, the subdivision count is half CircleSectors, rounded to the nearest Integer using VB rounding. CircleSectors values at or below 1 use 72 before halving. When ArcSegmentsLength is positive, the count is floor(2 * PI * radius / ArcSegmentsLength), clamped between MinCircleSectors and MaxCircleSectors. The resulting count must be between 3 and 1000000. Arc samples are evenly spaced along the stored arc direction, including arcs crossing zero degrees, and the affine transform is applied to each point. Endpoint Z values are retained and interpolated. V3 cubic curve records are subdivided in XYZ using their control points; vertex-backed curve records return their stored segments. These paths do not use the circular-arc sector settings. A circle or ellipse stored as Ellipse_i, an ordinary polyline, or any other non-Curve_i object is not accepted by this method. A missing or unsuitable handle, inactive control or invalid curve settings return zero and leave all six existing output arrays unchanged. Hidden or locked layers do not prevent this explicit read; LockEdit and PauseAvax also allow it. The method does not change selection, unfinished construction, geometry, Undo history or the drawing dirty flag.

Numeric changes from v2

Update the following variables, arrays or event-data receivers when porting the corresponding v2 call. The declarations above show the current overloads.

Parameter or value	v2 type	v3 type
x1	Single()	Double()
y1	Single()	Double()

Parameter or value	v2 type	v3 type
z1	Single()	Double()
x2	Single()	Double()
y2	Single()	Double()
z2	Single()	Double()

For ByRef parameters, use a variable with the exact v3 type. Preserve array dimensions and the documented indexing. Moving from v2 to v3 explains conversion and legacy file-storage rules.

Example

Setup

Add an Avax.cAvax Windows Forms control named `cAvax1`. Paste the procedures into the form class. Example setup and supporting files explains references, initialization and file paths.

Add these Windows Forms controls or components to the form. Fields referenced by Handles clauses must be declared WithEvents, as the Visual Studio designer normally does.

Name	Type
Button1	Button

VB.NET example

```
Dim AvaxHandle As Integer
Private Sub Form1_Load(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs) Handles
    MyBase.Load
    CAvax1.StartAvax()
    AddCurve()
End Sub
Private Sub Form1_FormClosed(ByVal eventSender As System.Object, ByVal eventArgs As
    System.Windows.Forms.FormClosedEventArgs) Handles Me.FormClosed
    CAvax1.EndAvax()
End Sub
Private Sub AddCurve()
    Dim xStart As Double
    Dim yStart As Double
    Dim zStart As Double
    Dim xEnd As Double
    Dim yEnd As Double
    Dim zEnd As Double
    Dim xc As Double
    Dim yc As Double
    Dim zc As Double
    Dim vItemProperties As Object = Nothing
    Dim iClr As Integer
    Dim iWidth As Integer
    Dim iStyle As Integer
    Dim iLayer As Integer
    xStart = 10
    yStart = 10
    zStart = 0
    xEnd = 40
    yEnd = 40
    zEnd = 0
    xc = 20
    yc = 30
    zc = 0
    AvaxHandle = CAvax1.Add_CurveArc(xStart, yStart, zStart, xEnd, yEnd, zEnd, xc, yc, zc,
    vItemProperties, iClr, iWidth, iStyle, iLayer)
    CAvax1.Command = Avax.AvaxCommand.AutoLimits_c
End Sub
Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
```

```
    Button1.Click
    Dim x1() As Double = Nothing
    Dim y1() As Double = Nothing
    Dim z1() As Double = Nothing
    Dim x2() As Double = Nothing
    Dim y2() As Double = Nothing
    Dim z2() As Double = Nothing
    Dim Answer as Integer
    Answer = CAvax1.GetCurveLines(AvaxHandle, x1, y1, z1, x2, y2, z2)
    For i As Integer = 1 To x1.Length - 1
        x1(i) += 1
        y1(i) += 1
    Next i
    For i As Integer = 1 To x2.Length - 1
        x2(i) -= 1
        y2(i) -= 1
    Next i
    CAvax1.Add_PolyLine(x1, y1, z1, Nothing, 1, 1, -1, -1)
    CAvax1.Add_PolyLine(x2, y2, z2, Nothing, 3, 1, -1, -1)
End Sub
```

GetDocumentClearText

API reference

Returns readable text from an ordinary text or document item. Use this method to obtain content for searching, displaying or copying without AVAX formatting tags or RTF control words. To retrieve the complete original source, use GetText.

VB.NET declaration

```
Public Function GetDocumentClearText(
    ByRef avHandleDocument As Integer) As String
```

Parameter

Parameter	Meaning
avHandleDocument	Handle of a Text_i or Document_i item. The method does not change this variable. A text-filled shape is a different item type and is not accepted here.

Return value and remarks

Returns the item's text with supported AVAX markup or RTF decoded by the document layout engine. Unicode and literal escaped delimiters are retained. Trailing spaces are removed; a result can still contain line breaks. An empty valid item can return an empty string.

Returns Nothing for missing or wrong-type handles, or when the control has ended or is disabled. Call StartAvax before querying. Reading does not change the original source, selection, active command, drawing or Undo history, and does not raise creation/change events.

The handle remains Integer in v3. Any coordinates or document dimensions used to create the source item should use the Double types shown in that creation method's declaration.

VB.NET example

Pass an already started control and the handle of an existing text or document item. The example handles unavailable content without changing the drawing.

```
Option Strict On
Imports System

Public Module DocumentClearTextExample
    Public Function ReadLabel(cad As Avax.cAvax, handle As Integer) As String
        If cad Is Nothing Then Throw New ArgumentNullException(NameOf(cad))
        Dim content As String = cad.GetDocumentClearText(handle)
        If content Is Nothing Then
            Return ""
        End If
        Return content
    End Function
End Module
```

GetDocumentFontSize

Returns the intrinsic font height of a Document_i object as Double. A disabled or unstarted control, missing handle or different object type returns zero. The handle argument is unchanged.

```
Public Function GetDocumentFontSize(ByRef avHandleDocument As Integer) As Double
```

Related options and reference

GetDocumentWidth

Returns the logical wrapping width of a Document_i object as Double. It does not return the occupied text width or the transformed world bounding box. Invalid handles, different object types and an unstarted or disabled control return zero.

```
Public Function GetDocumentWidth(ByRef avHandleDocument As Integer) As Double
```

Related options and reference

GetEllipseGeometry

cAvax method

Purpose

Use this method if you want to get the geometry of an ellipse object

VB.NET declaration

```
Function GetEllipseGeometry(ByRef avHandle As Integer, ByRef cx As Double, ByRef cy As Double, ByRef aR As
     Double, ByRef bR As Double, ByRef dAngle As Double) As Boolean
```

Parameters

Name	Type and passing	Meaning
avHandle	ByRef avHandle As Integer	Ellipse_i handle, including an elliptical arc.
cx	ByRef cx As Double	Output transformed center X in drawing units. Assigned only on success.
cy	ByRef cy As Double	Output transformed center Y in drawing units. Assigned only on success.
aR	ByRef aR As Double	Output semiaxis along dAngle. For a shear, this is the principal axis closest to transformed local +X; a 45-degree tie uses the major axis.
bR	ByRef bR As Double	Output perpendicular semiaxis. A zero value denotes a collapsed transform; an unrepresentable positive radius makes the method return False.
dAngle	ByRef dAngle As Double	Output counterclockwise orientation in degrees, in [0,360). Small positive angles are retained at Double precision.

Use the declaration for the overload you call. ByRef arguments can receive outputs; declare variables and arrays with the exact indicated type. Optional defaults are shown in the declaration.

Return value

Boolean - True on success, false on error.

Remarks

Returns the transformed center, two semiaxis lengths and orientation of an Ellipse_i object, including an elliptical arc. The result describes the full supporting ellipse, not its endpoints. Call StartAvax first and keep the control enabled. Declare cx, cy, aR, bR and dAngle as Double; avHandle remains Integer. Angles are counterclockwise degrees in [0,360). For perpendicular transformed axes, aR follows the transformed local X axis and bR follows the local Y axis. Under shear, the method returns the actual principal radii: aR uses the principal axis closest to transformed local +X, choosing the major axis on an exact 45-degree tie. A collapsed transform may produce a zero radius; a zero transform returns two zero radii and angle zero. Very different axis lengths and cancellation between large center terms are supported when the final values fit Double. A positive radius below the Double range, an overflowing center or radius, an unavailable or wrong-type handle, or an ended or disabled control returns False without changing any ByRef argument. Success returns True and updates all five geometry outputs together. The handle, drawing, selection, active command and Undo history remain unchanged. Add_Arc normally creates a segmented polyline and is therefore not accepted by this ellipse getter; its complete-circle special case is an ellipse.

Numeric changes from v2

Update the following variables, arrays or event-data receivers when porting the corresponding v2 call. The declarations above show the current overloads.

Parameter or value	v2 type	v3 type
cx	Single	Double
cy	Single	Double

Parameter or value	v2 type	v3 type
aR	Single	Double
bR	Single	Double
dAngle	Single	Double

For ByRef parameters, use a variable with the exact v3 type. Preserve array dimensions and the documented indexing. Moving from v2 to v3 explains conversion and legacy file-storage rules.

Example

Setup

Add an Avax.cAvax Windows Forms control named `cAvax1`. Paste the procedures into the form class. Example setup and supporting files explains references, initialization and file paths.

Add these Windows Forms controls or components to the form. Fields referenced by Handles clauses must be declared WithEvents, as the Visual Studio designer normally does.

Name	Type
Button1	Button

VB.NET example

```
Dim avHandle As Integer
Private Sub Form1_Load(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs) Handles
    MyBase.Load
    Dim avHandles(1) As Integer
    Call cAvax1.StartAvax()
    avHandle = cAvax1.Add_Ellipse(10, 10, 0, 4, 2)
    avHandles(1) = avHandle
    cAvax1.RotateItems(avHandles, 12, 11, 20)
End Sub
Private Sub Form1_FormClosed(ByVal eventSender As System.Object, ByVal eventArgs As
    System.Windows.Forms.FormClosedEventArgs) Handles Me.FormClosed
    Call cAvax1.EndAvax()
End Sub
Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
    Button1.Click
    Dim cx As Double
    Dim cy As Double
    Dim aR As Double
    Dim bR As Double
    Dim dAngle As Double
    Dim Answer as Boolean
    Answer = cAvax1.GetEllipseGeometry(avHandle, cx, cy, aR, bR, dAngle)
    MsgBox("cx=" & cx & Environment.NewLine & "cy=" & cy & Environment.NewLine & "aR=" & aR &
    Environment.NewLine & "bR=" & bR & Environment.NewLine & "Angle=" & dAngle)
End Sub
```

GetFontName

cAvax method

Purpose

Returns the font name and the character set for the specific font table index

VB.NET declaration

```
Sub GetFontName(ByVal Index As Integer, ByRef sFontName As String, ByRef lFontSet As Integer)
```

Parameters

Name	Type and passing	Meaning
Index	ByVal Index As Integer	Integer font-table index (promoted from Integer). Nonpositive or out-of-range indices resolve to the current control font.
sFontName	ByRef sFontName As String	ByRef String output for the effective name; blank table names resolve to FontName.
lFontSet	ByRef lFontSet As Integer	ByRef Integer output for the effective charset; inherited -1 resolves to FontCharset.

Use the declaration for the overload you call. ByRef arguments can receive outputs; declare variables and arrays with the exact indicated type. Optional defaults are shown in the declaration.

Result

This procedure has no return value. Any outputs are supplied through the documented ByRef arguments.

Numeric changes from v2

Update the following variables, arrays or event-data receivers when porting the corresponding v2 call. The declarations above show the current overloads.

Parameter or value	v2 type	v3 type
Index	Short	Integer

For ByRef parameters, use a variable with the exact v3 type. Preserve array dimensions and the documented indexing. Moving from v2 to v3 explains conversion and legacy file-storage rules.

Example

Setup

Add an Avax.cAvax Windows Forms control named `cAvax1`. Paste the procedures into the form class. Example setup and supporting files explains references, initialization and file paths.

Add these Windows Forms controls or components to the form. Fields referenced by Handles clauses must be declared WithEvents, as the Visual Studio designer normally does.

Name	Type
Command1	Button

VB.NET example

```
Private Sub Form1_Load(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs) Handles
    MyBase.Load
    Command1.Text = "GetFontName"
    cAvax1.StartAvax()
End Sub
Private Sub Form1_FormClosed(ByVal eventSender As System.Object, ByVal eventArgs As
```

```
    System.Windows.Forms.FormClosedEventArgs) Handles Me.FormClosed
    cAvax1.EndAvax()
End Sub
Private Sub Command1_Click(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs)
    Handles Command1.Click
    Dim Index As Integer
    Dim sFontName As String = String.Empty
    Dim lFontSet As Integer
    Call cAvax1.GetFontName(Index, sFontName, lFontSet)
    MsgBox("Font " & Format(Index) & " = " & sFontName & ", " & Format(lFontSet))
End Sub
```

GetFonts

cAvax method

Purpose

This function returns the total number, names and character sets of all the fonts

VB.NET declaration

```
Function GetFonts(ByRef sFonts As String(), ByRef lFontsSet As Integer()) As Integer
```

Parameters

Name	Type and passing	Meaning
sFonts	ByRef sFonts As String()	ByRef String() output; detached raw names with reserved index zero. Usable indices are 1 through the returned Integer count. Blank names inherit FontName.
lFontsSet	ByRef lFontsSet As Integer()	ByRef Integer() output aligned with sFonts; -1 inherits FontCharset, 0..255 are explicit Windows character sets.

Use the declaration for the overload you call. ByRef arguments can receive outputs; declare variables and arrays with the exact indicated type. Optional defaults are shown in the declaration.

Return value

Integer - Is the total number of the fonts

Numeric changes from v2

Update the following variables, arrays or event-data receivers when porting the corresponding v2 call. The declarations above show the current overloads.

Parameter or value	v2 type	v3 type
Return value	Short	Integer

For ByRef parameters, use a variable with the exact v3 type. Preserve array dimensions and the documented indexing. Moving from v2 to v3 explains conversion and legacy file-storage rules.

Example

Setup

Add an Avax.cAvax Windows Forms control named `cAvax1`. Paste the procedures into the form class. Example setup and supporting files explains references, initialization and file paths.

Add these Windows Forms controls or components to the form. Fields referenced by Handles clauses must be declared WithEvents, as the Visual Studio designer normally does.

Name	Type
Command1	Button

VB.NET example

```
Private Sub Form1_Load(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs) Handles
    MyBase.Load
    Command1.Text = "GetFonts"
    cAvax1.StartAvax()
End Sub
Private Sub Form1_FormClosed(ByVal eventSender As System.Object, ByVal eventArgs As
    System.Windows.Forms.FormClosedEventArgs) Handles Me.FormClosed
    cAvax1.EndAvax()
```

```
End Sub
Private Sub Command1_Click(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs)
    Handles Command1.Click
    Dim sFonts() As String = Nothing
    Dim lFontsSet() As Integer = Nothing
    Dim iCnt, iMax As Integer
    Dim sFontFile As String = String.Empty
    sFontFile = My.Application.Info.DirectoryPath & "\..\myFonts.fnt"
    Call cAvax1.LoadFonts(sFontFile)
    iMax = cAvax1.GetFonts(sFonts, lFontsSet)
    For iCnt = 1 To iMax
        Console.WriteLine(sFonts(iCnt) & "," & lFontsSet(iCnt))
    Next iCnt
End Sub
```

GetHandleKey

cAvax method

Purpose

Returns the key name of the object with the specific handle

VB.NET declaration

```
Function GetHandleKey(ByRef lHandle As Integer) As String
```

Parameters

Name	Type and passing	Meaning
lHandle	ByRef lHandle As Integer	Is the object's handle

Use the declaration for the overload you call. ByRef arguments can receive outputs; declare variables and arrays with the exact indicated type. Optional defaults are shown in the declaration.

Return value

String - Is the key name of the object

Example

Setup

Add an Avax.cAvax Windows Forms control named `cAvax1`. Paste the procedures into the form class. Example setup and supporting files explains references, initialization and file paths.

Add these Windows Forms controls or components to the form. Fields referenced by Handles clauses must be declared WithEvents, as the Visual Studio designer normally does.

Name	Type
Command1	Button

VB.NET example

```
Dim lHandle As Integer
Private Sub Form1_Load(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs) Handles
    MyBase.Load
    Command1.Text = "GetHandleKey"
    cAvax1.StartAvax()
    lHandle = cAvax1.Add_Line(0, 0, 0, 10, 10, 0)
    Call cAvax1.SetHandleKey(lHandle, "Line #1")
End Sub
Private Sub Form1_FormClosed(ByVal eventSender As System.Object, ByVal eventArgs As
    System.Windows.Forms.FormClosedEventArgs) Handles Me.FormClosed
    cAvax1.EndAvax()
End Sub
Private Sub Command1_Click(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs)
    Handles Command1.Click
    MsgBox(cAvax1.GetHandleKey(lHandle), MsgBoxStyle.Information, "GetHandleKey")
End Sub
```

GetHandlesArrByKeyArr

cAvax method

Purpose

Returns the number and handles of drawing objects whose keys match any entry in the supplied string array.

VB.NET declaration

```
Function GetHandlesArrByKeyArr(ByRef sKeys As String(), ByRef lHandles As Integer()) As Integer
```

Parameters

Name	Type and passing	Meaning
sKeys	ByRef sKeys As String()	One-based String() of exact keys; duplicate keys do not duplicate result handles. Index zero is ignored.
lHandles	ByRef lHandles As Integer()	One-based Integer() output; index zero reserved. Returned count equals Length-1; a successful search with no matches returns {0}.

Use the declaration for the overload you call. ByRef arguments can receive outputs; declare variables and arrays with the exact indicated type. Optional defaults are shown in the declaration.

Return value

Integer - Is the total number of the objects which have key names included in the string array sKeys()

Remarks

The number of objects can be greater than the number of key names in the string array sKeys()

Example

Setup

Add an Avax.cAvax Windows Forms control named `cAvax1`. Paste the procedures into the form class. Example setup and supporting files explains references, initialization and file paths.

Add these Windows Forms controls or components to the form. Fields referenced by Handles clauses must be declared WithEvents, as the Visual Studio designer normally does.

Name	Type
Command1	Button

VB.NET example

```
Private Sub Form1_Load(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs) Handles
    MyBase.Load
    Dim h As Integer
    Command1.Text = "GetHandlesArrByKeyArr"
    cAvax1.StartAvax()
    h = cAvax1.Add_Line(0, 0, 0, 15, 0, 0)
    Call cAvax1.SetHandleKey(h, "Line #1")
    h = cAvax1.Add_Line(0, 15, 0, 15, 15, 0)
    Call cAvax1.SetHandleKey(h, "Line #2")
    h = cAvax1.Add_Line(15, 0, 0, 15, 15, 0)
    Call cAvax1.SetHandleKey(h, "Line #3")
    h = cAvax1.Add_Line(0, 0, 0, 0, 15, 0)
    Call cAvax1.SetHandleKey(h, "Line #4")
End Sub
Private Sub Form1_FormClosed(ByVal eventSender As System.Object, ByVal eventArgs As
    System.Windows.Forms.FormClosedEventArgs) Handles Me.FormClosed
```

```
    cAvax1.EndAvax()
End Sub
Private Sub Command1_Click(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs)
    Handles Command1.Click
    Dim sKeys() As String = Nothing
    Dim sStr As String = String.Empty
    Dim lHandles() As Integer = Nothing
    Dim lMax, lCnt As Integer
    ReDim sKeys(4)
    sKeys(1) = "Line #1"
    sKeys(2) = "Line #2"
    sKeys(3) = "Line #3"
    sKeys(4) = "Line #4"
    lMax = cAvax1.GetHandlesArrByKeyArr(sKeys, lHandles)
    If lMax <> 0 Then
        For lCnt = 1 To lMax
            sStr = sStr & Str(lHandles(lCnt)) & " "
        Next lCnt
        MsgBox("Handles =" & sStr, MsgBoxStyle.Information, "GetHandlesArrByKeyArr")
    Else
        MsgBox("No handles found", MsgBoxStyle.Critical, "GetHandlesArrByKeyArr")
    End If
End Sub
```

GetHandlesByKey

cAvax method

Purpose

Returns the number and the handles of all the objects that have the specific key name

VB.NET declaration

```
Function GetHandlesByKey(ByRef sKey As String, ByRef lHandles As Integer()) As Integer
```

Parameters

Name	Type and passing	Meaning
sKey	ByRef sKey As String	Exact normalized key: trim and current-culture uppercase, without changing this ByRef input.
lHandles	ByRef lHandles As Integer()	One-based Integer() output; index zero reserved. Returned count equals Length-1; a successful search with no matches returns {0}.

Use the declaration for the overload you call. ByRef arguments can receive outputs; declare variables and arrays with the exact indicated type. Optional defaults are shown in the declaration.

Return value

Integer - Is the total number of the objects which have this key name

Example

Setup

Add an Avax.cAvax Windows Forms control named `cAvax1`. Paste the procedures into the form class. Example setup and supporting files explains references, initialization and file paths.

Add these Windows Forms controls or components to the form. Fields referenced by Handles clauses must be declared WithEvents, as the Visual Studio designer normally does.

Name	Type
Command1	Button

VB.NET example

```
Private Sub Form1_Load(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs) Handles
    MyBase.Load
    Dim h As Integer
    Command1.Text = "GetHandlesByKey"
    cAvax1.StartAvax()
    h = cAvax1.Add_Line(0, 0, 0, 15, 0, 0)
    Call cAvax1.SetHandleKey(h, "Line #1")
    h = cAvax1.Add_Line(0, 15, 0, 15, 15, 0)
    Call cAvax1.SetHandleKey(h, "Line #2")
    h = cAvax1.Add_Line(15, 0, 0, 15, 15, 0)
    Call cAvax1.SetHandleKey(h, "Line #3")
    h = cAvax1.Add_Line(0, 0, 0, 0, 15, 0)
    Call cAvax1.SetHandleKey(h, "Line #4")
End Sub
Private Sub Form1_FormClosed(ByVal eventSender As System.Object, ByVal eventArgs As
    System.Windows.Forms.FormClosedEventArgs) Handles Me.FormClosed
    cAvax1.EndAvax()
End Sub
Private Sub Command1_Click(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs)
    Handles Command1.Click
```

```
    Dim sKey As String = String.Empty
    Dim sStr As String = String.Empty
    Dim lHandles() As Integer = Nothing
    Dim lMax, lCnt As Integer
    sKey = "Line #3"
    lMax = cAvax1.GetHandlesByKey(sKey, lHandles)
    If lMax = 1 Then
        MsgBox("Handle =" & Str(lHandles(1)), MsgBoxStyle.Information, "GetHandlesByKey")
    ElseIf lMax > 1 Then
        For lCnt = 1 To lMax
            sStr = sStr & Str(lHandles(lCnt)) & " "
        Next lCnt
        MsgBox("Handles =" & sStr, MsgBoxStyle.Information, "GetHandlesByKey")
    Else
        MsgBox("No handles found", MsgBoxStyle.Critical, "GetHandlesByKey")
    End If
End Sub
```

GetHandlesUnderArea

cAvax method

Purpose

Returns with the number and the handles of the objects inside a specific parallelogram area

See Idle selection and spatial queries for automatic selection, filled-shape picking, one-based Integer query results, availability and Double-coordinate areas.

VB.NET declarations

```
Function GetHandlesUnderArea(ByRef avMyHandles As Integer(), ByRef xMin As Double, ByRef yMin As Double,
    ByRef xMax As Double, ByRef yMax As Double, ByRef fzSort As Boolean) As Integer

Function GetHandlesUnderArea(ByRef avMyHandles As Integer(), ByRef xMin As Double, ByRef yMin As Double,
    ByRef xMax As Double, ByRef yMax As Double) As Integer
```

Parameters

Name	Type and passing	Meaning
avMyHandles	ByRef avMyHandles As Integer()	Output Integer() containing the returned entity handles at indices 1 through the returned count. Index 0 is reserved.
xMin	ByRef xMin As Double	Is the x coordinate of the buttom left of the parallelogram
yMin	ByRef yMin As Double	Is the y coordinate of the buttom left of the parallelogram
xMax	ByRef xMax As Double	Is the x coordinate of the top right of the parallelogram
yMax	ByRef yMax As Double	Is the y coordinate of the top right of the parallelogram
fzSort	ByRef fzSort As Boolean	A Boolean expression that if it is true, then the objects inside the specific parallelogram area are sorted by their z order

Use the declaration for the overload you call. ByRef arguments can receive outputs; declare variables and arrays with the exact indicated type. Optional defaults are shown in the declaration.

Return value

Integer - Is the total number of the objects inside the specific parallelogram area

Numeric changes from v2

Update the following variables, arrays or event-data receivers when porting the corresponding v2 call. The declarations above show the current overloads.

Parameter or value	v2 type	v3 type
xMin	Single	Double
yMin	Single	Double
xMax	Single	Double
yMax	Single	Double

For ByRef parameters, use a variable with the exact v3 type. Preserve array dimensions and the documented indexing. Moving from v2 to v3 explains conversion and legacy file-storage rules.

Example

Setup

Add an Avax.cAvax Windows Forms control named `cAvax1`. Paste the procedures into the form class. Example setup and supporting files explains references, initialization and file paths.

Add these Windows Forms controls or components to the form. Fields referenced by Handles clauses must be declared WithEvents, as the Visual Studio designer normally does.

Name	Type
Command1	Button
Command2	Button
Label1	Label

VB.NET example

```
Dim xzMin, yzMin As Double
Dim xzMax, yzMax As Double
Private Sub CAvax1_ZoomChange(ByVal Sender As Object, ByVal e As Avax.cAvax.ZoomChangeEventArgs) Handles
    CAvax1.ZoomChange
    xzMin = e.xMin
    yzMin = e.yMin
    xzMax = e.xMax
    yzMax = e.yMax
    Call GetHandlesUnderZoomArea()
End Sub
Private Sub Form1_Load(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs) Handles
    MyBase.Load
    Command1.Text = "Zoom"
    Command2.Text = "Previous"
    cAvax1.StartAvax()
    Call cAvax1.Add_Line(0, 1, 0, 11, 10, 0)
    Call cAvax1.Add_Line(0, 12, 0, 12, 10, 0)
    Call cAvax1.Add_Line(8, 5, 0, 10, 10, 0)
    Call cAvax1.Add_Line(6, 5, 0, 5, 10, 0)
    Call cAvax1.Add_Line(7, 0, 0, 5, 7, 0)
    Call cAvax1.Add_Line(0, -1, 0, 8, 7, 0)
    Call cAvax1.Add_Line(9, 2, 0, 3, 9, 0)
    Call cAvax1.Add_Line(0, 0, 0, 10, 0, 0)
    Call GetHandlesUnderZoomArea()
End Sub
Private Sub Form1_FormClosed(ByVal eventSender As System.Object, ByVal eventArgs As
    System.Windows.Forms.FormClosedEventArgs) Handles Me.FormClosed
    cAvax1.EndAvax()
End Sub
Private Sub Command1_Click(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs)
    Handles Command1.Click
    CAvax1.Command = Avax.AvaxCommand.Zoom_c
End Sub
Private Sub Command2_Click(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs)
    Handles Command2.Click
    CAvax1.Command = Avax.AvaxCommand.Previous_c
End Sub
Sub GetHandlesUnderZoomArea()
    Dim lMax As Integer
    Dim [Handles]() As Integer = Nothing
    lMax = CAvax1.GetHandlesUnderArea([Handles], xzMin, yzMin, xzMax, yzMax)
    If lMax > 0 Then
        Label1.Text = Format(lMax) & " objects inside current zoom window."
    Else
        Label1.Text = "No objects inside current zoom window"
    End If
End Sub
```

GetHandlesUnderSelectCursor

cAvax method

Purpose

Returns with the number and the handles of the objects below the selection pointer

See Idle selection and spatial queries for automatic selection, filled-shape picking, one-based Integer query results, availability and Double-coordinate areas.

VB.NET declarations

```
Function GetHandlesUnderSelectCursor(ByRef avMyHandles As Integer(), ByRef fzSort As Boolean) As Integer

Function GetHandlesUnderSelectCursor(ByRef avMyHandles As Integer()) As Integer
```

Parameters

Name	Type and passing	Meaning
avMyHandles	ByRef avMyHandles As Integer()	Output Integer() containing the returned entity handles at indices 1 through the returned count. Index 0 is reserved.
fzSort	ByRef fzSort As Boolean	A Boolean that if it is true, then the objects below the selecion pointer are sorted by their z order

Use the declaration for the overload you call. ByRef arguments can receive outputs; declare variables and arrays with the exact indicated type. Optional defaults are shown in the declaration.

Return value

Integer - Is the total number of the objects below the selection pointer

Example

Setup

Add an Avax.cAvax Windows Forms control named `cAvax1`. Paste the procedures into the form class. Example setup and supporting files explains references, initialization and file paths.

Add these Windows Forms controls or components to the form. Fields referenced by Handles clauses must be declared WithEvents, as the Visual Studio designer normally does.

Name	Type
Label1	Label
ToolTip1	ToolTip

VB.NET example

```
Private Sub CAvax1_OnMouseMove(ByVal Sender As Object, ByVal e As Avax.cAvax.MouseMoveEventArgs) Handles
    CAvax1.OnMouseMove
    Dim sMsg As String = String.Empty
    Dim [Handles]() As Integer = Nothing
    Dim lCnt, lMax As Integer
    lMax = CAvax1.GetHandlesUnderSelectCursor([Handles])
    If lMax > 0 Then
        For lCnt = 1 To lMax
            sMsg = sMsg & Format([Handles](lCnt)) & ", "
        Next lCnt
        sMsg = Mid(sMsg, 1, Len(sMsg) - 2)
        ToolTip1.SetToolTip(CAvax1, sMsg)
        sMsg = "Handles of the objects under select cursor box:" & vbCrLf & sMsg
    Else
```

```
        ToolTip1.SetToolTip(CAvax1, "")
    End If
    Label1.Text = sMsg
End Sub
Private Sub Form1_Load(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs) Handles
    MyBase.Load
    cAvax1.StartAvax()
    Call cAvax1.Add_Line(0, 0, 0, 10, 10, 0)
    Call cAvax1.Add_Line(0, 10, 0, 10, 10, 0)
    Call cAvax1.Add_Line(10, 5, 0, 10, 10, 0)
    Call cAvax1.Add_Line(10, 5, 0, 5, 10, 0)
    Call cAvax1.Add_Rectangle(0, 0, 12, 10)
End Sub
Private Sub Form1_FormClosed(ByVal eventSender As System.Object, ByVal eventArgs As
    System.Windows.Forms.FormClosedEventArgs) Handles Me.FormClosed
    cAvax1.EndAvax()
End Sub
```

GetImageEffectDefinitions

Purpose

Reads the available effect kinds and their supported amounts. Use the returned descriptions to populate an effect selector and configure its value editor.

Declaration

```
Function GetImageEffectDefinitions() As AvaxImageEffectInfo()
```

Parameters and result

This method has no parameters. Returns a new zero-based array containing one AvaxImageEffectInfo description for each of the 17 effect kinds. Reading the catalog is available before starting a drawing. Changing a returned description does not change the catalog or an image.

See Effect descriptions and amount ranges for the individual values. Use a description's DefaultValue when constructing an effect with that kind; the ordinary AvaxImageEffect constructor does not automatically select the catalog default.

VB.NET example

```
Option Strict On
Public Module GetImageEffectDefinitionsExample
    Public Function CreateDefaultEffect(cad As Avax.cAvax,
                                        kind As Avax.AvaxImageEffectKind) As Avax.AvaxImageEffect
        For Each description As Avax.AvaxImageEffectInfo In cad.GetImageEffectDefinitions()
            If description.Kind = kind Then
                Return New Avax.AvaxImageEffect(description.Kind, description.DefaultValue)
            End If
        Next
        Return Nothing
    End Function
End Module
```

Related options and reference

GetImageEffects

Purpose

Reads an image's ordered effect list, including entries whose Enabled value is False. Applications can use this copy to populate an effect editor.

Declaration

```
Function GetImageEffects(ByVal handle As Integer) As AvaxImageEffect()
```

Parameters and result

Name	Meaning
handle	Integer handle of the image to inspect.

Returns a zero-based array of independent AvaxImageEffect values. Returns an empty array when no effects are stored or the handle is unavailable or unsupported. Reading is available while the control is disabled or editing is locked. Editing the array or its values does not edit the image; pass the proposed list to SetImageEffects to commit it.

See Effect value objects for Kind, Amount, Enabled and Clone. Malformed stored effect records can raise ArgumentException, FormatException or OverflowException; this getter does not convert those record-reading errors into an empty list.

VB.NET example

```
Option Strict On
Public Module GetImageEffectsExample
    Public Sub DescribeCurrentEffects(cad As Avax.cAvax, handle As Integer)
        Dim effects As Avax.AvaxImageEffect() = cad.GetImageEffects(handle)
        For Each effect As Avax.AvaxImageEffect In effects
            System.Diagnostics.Debug.WriteLine(effect.ToString())
        Next
    End Sub
End Module
```

Related options and reference

GetImageSource

Purpose

Copies the encoded image data stored in an image object, before its effect list is applied. Use this method to retrieve the source retained by the drawing.

Declaration

```
Function GetImageSource(ByVal handle As Integer) As Byte()
```

Parameters and result

Name	Meaning
handle	Integer image handle accepted by IsImageShape.

Returns a new Byte array containing the stored encoded source, or Nothing when the handle is unavailable or unsupported. The returned bytes retain their stored image format. Applying Crop or another effect leaves these source bytes unchanged. Editing the returned array changes only the caller's copy. Reading does not add an Undo entry and remains available while the control is disabled or editing is locked.

VB.NET example

```
Option Strict On
Public Module GetImageSourceExample
    Public Function SaveStoredSource(cad As Avax.cAvax, handle As Integer,
                                     filename As String) As Boolean
        Dim source As Byte() = cad.GetImageSource(handle)
        If source Is Nothing Then Return False
        ' Choose a filename extension matching the stored source image format.
        System.IO.File.WriteAllBytes(filename, source)
        Return True
    End Function
End Module
```

Related options and reference

GetItemByLayer

Purpose

Read whether an existing object follows its layer color and line style.

VB.NET declaration

```
Public Function GetItemByLayer(ByVal handle As Integer, ByRef colorByLayer As Boolean, ByRef
    lineStyleByLayer As Boolean) As Boolean
```

Parameters

handle: an object handle. colorByLayer and lineStyleByLayer: output Boolean variables.

Return value

True on success. False for an unavailable control or invalid handle; output variables are retained on failure.

Example

Call after StartAvax on the owning UI thread.

```
Option Strict On
Public Module Help_GetItemByLayer
    Public Sub Run(cad As Avax.cAvax)
        Dim handle As Integer = cad.AddLine(0.0, 0.0, 0.0, 10.0, 0.0, 0.0)
        Dim colorMode As Boolean, styleMode As Boolean
        Dim found As Boolean = cad.GetItemByLayer(handle, colorMode, styleMode)
    End Sub
End Module
```

Related options and reference

GetItemZOrder

Purpose

Reads the stored signed drawing-order value of an existing item. Use it to display order values or prepare a front-to-back list.

VB.NET declaration

```
Function GetItemZOrder(ByVal handle As Integer) As Integer
```

Parameters

Parameter	Meaning
handle	Integer handle of a root item in the current drawing.

Return value and behavior

Returns the stored Integer value. Zero is both a valid stored value and the neutral result for a missing handle or an unstarted/ended control; use a known valid handle when that distinction matters. The value is readable while the control is disabled. Reading it does not change selection, geometry, settings or history. Higher values are in front when ShapeZorderOn_p is enabled; equal values use handle order, with the higher handle in front.

VB.NET example

Pass a started control on its owning UI thread. For the reversal example, supply every current root once in the current front-to-back order, with slot zero reserved. Enable ShapeZorderOn_p when you want the stored order to determine the visible foreground.

```
Option Strict On
Imports Avax

Public Module ItemOrderReadExample
    Public Function ReadForDisplay(cad As cAvax, id As Integer) As String
        Return cad.GetItemZOrder(id).ToString(Globalization.CultureInfo.InvariantCulture)
    End Function
End Module
```

Related options and reference

GetKeyArrByHandlesArr

cAvax method

Purpose

Returns the key names that correspond to a specific handles array

VB.NET declaration

```
Function GetKeyArrByHandlesArr(ByRef lHandles As Integer(), ByRef sKeys As String()) As Boolean
```

Parameters

Name	Type and passing	Meaning
lHandles	ByRef lHandles As Integer()	One-based input IDs, retained in their original order. Unknown IDs yield Nothing slots.
sKeys	ByRef sKeys As String()	Detached one-based String() result with a slot for every logical input handle.

Use the declaration for the overload you call. ByRef arguments can receive outputs; declare variables and arrays with the exact indicated type. Optional defaults are shown in the declaration.

Return value

Boolean - True on success, false on error.

Example

Setup

Add an Avax.cAvax Windows Forms control named `cAvax1`. Paste the procedures into the form class. Example setup and supporting files explains references, initialization and file paths.

Add these Windows Forms controls or components to the form. Fields referenced by Handles clauses must be declared WithEvents, as the Visual Studio designer normally does.

Name	Type
Command1	Button

VB.NET example

```
Dim lHandles() As Integer = Nothing
Private Sub Form1_Load(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs) Handles
    MyBase.Load
    Command1.Text = "GetKeyArrByHandlesArr"
    cAvax1.StartAvax()
    ReDim lHandles(4)
    lHandles(1) = cAvax1.Add_Line(0, 0, 0, 15, 0, 0)
    Call cAvax1.SetHandleKey(lHandles(1), "Line #1")
    lHandles(2) = cAvax1.Add_Line(0, 15, 0, 15, 15, 0)
    Call cAvax1.SetHandleKey(lHandles(2), "Line #2")
    lHandles(3) = cAvax1.Add_Line(15, 0, 0, 15, 15, 0)
    Call cAvax1.SetHandleKey(lHandles(3), "Line #3")
    lHandles(4) = cAvax1.Add_Line(0, 0, 0, 0, 15, 0)
    Call cAvax1.SetHandleKey(lHandles(4), "Line #4")
End Sub
Private Sub Form1_FormClosed(ByVal eventSender As System.Object, ByVal eventArgs As
    System.Windows.Forms.FormClosedEventArgs) Handles Me.FormClosed
    cAvax1.EndAvax()
End Sub
Private Sub Command1_Click(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs)
```

```
    Handles Command1.Click
    Dim sKeys() As String = Nothing
    Dim sStr As String = String.Empty
    Dim lMax, lCnt As Integer
    If cAvax1.GetKeyArrByHandlesArr(lHandles, sKeys) = True Then
        lMax = UBound(sKeys)
        For lCnt = 1 To lMax
            sStr = sStr & sKeys(lCnt) & vbCrLf
        Next lCnt
        MsgBox("Keys =" & vbCrLf & sStr, MsgBoxStyle.Information, "GetKeyArrByHandlesArr")
    Else
        MsgBox("No keys found", MsgBoxStyle.Critical, "GetKeyArrByHandlesArr")
    End If
End Sub
```

GetLayerColor

Purpose

Read the color assigned to a layer.

VB.NET declaration

```
Public Function GetLayerColor(ByVal layer As Integer) As Integer
```

Parameters

layer: an existing zero-based layer index. Use a started control on its owning UI thread.

Return value

Integer OLE RGB value: red + green*256 + blue*65536.

Example

Call after StartAvax on the owning UI thread.

```
Option Strict On
Public Module Help_GetLayerColor
    Public Sub Run(cad As Avax.cAvax)
        Dim rgb As Integer = cad.GetLayerColor(0)
        Dim red As Integer = rgb And &HFF
    End Sub
End Module
```

Related options and reference

GetLayerLineStyle

Purpose

Read the line-style table index assigned to a layer.

VB.NET declaration

```
Public Function GetLayerLineStyle(ByVal layer As Integer) As Integer
```

Parameters

layer: an existing zero-based layer index. Use a started control on its owning UI thread.

Return value

0 for Continuous; otherwise the index of the stored definition in the drawing line-style table, or -1 if that definition is not registered.

Example

Call after StartAvax on the owning UI thread.

```
Option Strict On
Public Module Help_GetLayerLineStyle
    Public Sub Run(cad As Avax.cAvax)
        Dim styleIndex As Integer = cad.GetLayerLineStyle(0)
    End Sub
End Module
```

Related options and reference

GetLayers

cAvax method

Purpose

Gets the names and status for all the drawing layers

VB.NET declaration

```
Function GetLayers(ByRef sLayerNames As String(), ByRef iLayerStatus As Integer()) As Integer
```

Parameters

Name	Type and passing	Meaning
sLayerNames	ByRef sLayerNames As String()	The names of the layers
iLayerStatus	ByRef iLayerStatus As Integer()	Integer() indexed from zero. Independent flags: 1 hidden, 2 locked, 4 nonprinting; combine with Or (0..7). Original table indices are 0..255.

Use the declaration for the overload you call. ByRef arguments can receive outputs; declare variables and arrays with the exact indicated type. Optional defaults are shown in the declaration.

Return value

Integer - The number of the layers

Remarks

Every drawing has 256 layers (from 0 to 255)

Numeric changes from v2

Update the following variables, arrays or event-data receivers when porting the corresponding v2 call. The declarations above show the current overloads.

Parameter or value	v2 type	v3 type
iLayerStatus	Short()	Integer()
Return value	Short	Integer

For ByRef parameters, use a variable with the exact v3 type. Preserve array dimensions and the documented indexing. Moving from v2 to v3 explains conversion and legacy file-storage rules.

Example

Setup

Add an Avax.cAvax Windows Forms control named `cAvax1`. Paste the procedures into the form class. Example setup and supporting files explains references, initialization and file paths.

Add these Windows Forms controls or components to the form. Fields referenced by Handles clauses must be declared WithEvents, as the Visual Studio designer normally does.

Name	Type
Command1	Button
List1	ListBox

VB.NET example

```
Private Sub Form1_Load(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs) Handles
    MyBase.Load
```

```
    cAvax1.StartAvax()
End Sub
Private Sub Form1_FormClosed(ByVal eventSender As System.Object, ByVal eventArgs As
    System.Windows.Forms.FormClosedEventArgs) Handles Me.FormClosed
    cAvax1.EndAvax()
End Sub
Private Sub Command1_Click(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs)
    Handles Command1.Click
    Dim sLayerNames() As String = Nothing
    Dim iLayerStatus() As Integer = Nothing
    Dim Answer As Integer
    Dim iCnt As Integer
    Dim s1 As String = String.Empty
    Answer = cAvax1.GetLayers(sLayerNames, iLayerStatus)
    For iCnt = 1 To Answer
        If sLayerNames(iCnt - 1) = "" Then sLayerNames(iCnt - 1) = Format(iCnt - 1)
        If iLayerStatus(iCnt - 1) = 0 Then s1 = "Visible" Else s1 = "Not visible"
        List1.Items.Add(sLayerNames(iCnt - 1) & "/" & s1)
    Next iCnt
End Sub
```

GetLineStartPoint

cAvax method

Purpose

Reads an accepted XYZ construction point from the current command without supplying a new point.

See Line, Point and Rectangle commands for connected XYZ lines, the planar Point tool versus Add_Point, rectangular closed polylines, completion and host callbacks.

See Circle and Ellipse commands for center/radius, diameter and three-point circles, ellipse radii, Double precision and one-shot command completion.

See Arc commands and segmentation for the three arc pick orders, standalone completion, polyline continuation and arc resolution settings.

See Polyline, Polygon and Close commands for pending XYZ chains, exact closing edges, regular polygons, vertex counts, completion and complete VB.NET examples.

VB.NET declaration

```
Function GetLineStartPoint(ByRef x As Double, ByRef y As Double, ByRef z As Double) As Boolean
```

Parameters

Name	Type and passing	Meaning
x	ByRef x As Double	Is the x coordinate of the point
y	ByRef y As Double	Is the y coordinate of the point
z	ByRef z As Double	Is the z coordinate of the point

Use the declaration for the overload you call. ByRef arguments can receive outputs; declare variables and arrays with the exact indicated type. Optional defaults are shown in the declaration.

Return value

Boolean - True on success, false on error.

Remarks

For Line_c, Polyline_c, Polygon_c and FreehandPolyline_c the output is the last accepted point. Other supported active commands return their first accepted point. A started, enabled control with at least one accepted construction point is required. SelectV_c point-selection requests are excluded. Returns False without assigning outputs when no applicable point is available. Reading does not accept a new point or change the active command.

Numeric changes from v2

Update the following variables, arrays or event-data receivers when porting the corresponding v2 call. The declarations above show the current overloads.

Parameter or value	v2 type	v3 type
x	Single	Double
y	Single	Double
z	Single	Double

For ByRef parameters, use a variable with the exact v3 type. Preserve array dimensions and the documented indexing. Moving from v2 to v3 explains conversion and legacy file-storage rules.

Example

Setup

Add an Avax.cAvax Windows Forms control named `cAvax1`. Paste the procedures into the form class. Example setup and supporting files explains references, initialization and file paths.

VB.NET example

```
Private Sub Form1_Load(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs) Handles
    MyBase.Load
    Call CAvax1.StartAvax()
    CAvax1.Command = Avax.AvaxCommand.Line_c
End Sub
Private Sub Form1_FormClosed(ByVal eventSender As System.Object, ByVal eventArgs As
    System.Windows.Forms.FormClosedEventArgs) Handles Me.FormClosed
    Call cAvax1.EndAvax()
End Sub
Private Sub CAvax1_Click(ByVal sender As Object, ByVal e As System.EventArgs) Handles CAvax1.Click
    Dim x As Double
    Dim y As Double
    Dim z As Double
    Dim Answer As Boolean
    Dim lineon As Integer
    CAvax1.GetAvaxStatus(False, False, False, False, False, False, False, lineon, False, False, False,
    False, False, False, False, False)
    If lineon <> 0 Then
        Answer = CAvax1.GetLineStartPoint(x, y, z)
        If Answer = True Then MsgBox("x=" & x & Environment.NewLine & "y=" & y)
    End If
End Sub
```

GetLineStyle

cAvax method

Purpose

This function returns the linetype name and data for a specific line type index

VB.NET declaration

```
Function GetLineStyle(ByVal Index As Integer) As String
```

Parameters

Name	Type and passing	Meaning
Index	ByVal Index As Integer	Integer index (formerly Integer). Existing positive indices return the stored definition; all others return Nothing.

Use the declaration for the overload you call. ByRef arguments can receive outputs; declare variables and arrays with the exact indicated type. Optional defaults are shown in the declaration.

Return value

String - Is the data for the specific line type index

Numeric changes from v2

Update the following variables, arrays or event-data receivers when porting the corresponding v2 call. The declarations above show the current overloads.

Parameter or value	v2 type	v3 type
Index	Short	Integer

For ByRef parameters, use a variable with the exact v3 type. Preserve array dimensions and the documented indexing. Moving from v2 to v3 explains conversion and legacy file-storage rules.

Example

Setup

Add an Avax.cAvax Windows Forms control named `cAvax1`. Paste the procedures into the form class. Example setup and supporting files explains references, initialization and file paths.

Add these Windows Forms controls or components to the form. Fields referenced by Handles clauses must be declared WithEvents, as the Visual Studio designer normally does.

Name	Type
Command1	Button
Label1	Label

VB.NET example

```
Dim iLineStyle As Integer
Dim AvaxHandle As Integer
Private Sub Form1_Load(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs) Handles
    MyBase.Load
    Command1.Text = "ChangeLineStyle"
    cAvax1.StartAvax()
    iLineStyle = 0
    Label1.Text = cAvax1.GetLineStyle(iLineStyle)
    AvaxHandle = CAvax1.Add_Line(0, 0, 0, 25, 10, 0, Nothing, 5, -1, iLineStyle, -1)
```

```
End Sub
Private Sub Form1_FormClosed(ByVal eventSender As System.Object, ByVal eventArgs As
    System.Windows.Forms.FormClosedEventArgs) Handles Me.FormClosed
    cAvax1.EndAvax()
End Sub
Private Sub Command1_Click(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs)
    Handles Command1.Click
    Dim h(1) As Integer
    Dim Answer As Boolean
    iLineStyle = iLineStyle + 1
    If iLineStyle > 9 Then iLineStyle = 0
    h(1) = AvaxHandle
    Answer = cAvax1.ChangeLineStyle(h, iLineStyle)
    Label1.Text = cAvax1.GetLineStyle(iLineStyle)
End Sub
```

GetLineStyleNames

cAvax method

Purpose

This function returns the total number and the names of the line types

VB.NET declaration

```
Function GetLineStyleNames(ByRef sLineStyleNames As String()) As Integer
```

Parameters

Name	Type and passing	Meaning
sLineStyleNames	ByRef sLineStyleNames As String()	ByRef detached String() output of untrimmed name prefixes; reserved/unnamed slots are Nothing.

Use the declaration for the overload you call. ByRef arguments can receive outputs; declare variables and arrays with the exact indicated type. Optional defaults are shown in the declaration.

Return value

Integer - Is the total number of the line types

Numeric changes from v2

Update the following variables, arrays or event-data receivers when porting the corresponding v2 call. The declarations above show the current overloads.

Parameter or value	v2 type	v3 type
Return value	Short	Integer

For ByRef parameters, use a variable with the exact v3 type. Preserve array dimensions and the documented indexing. Moving from v2 to v3 explains conversion and legacy file-storage rules.

Example

Setup

Add an Avax.cAvax Windows Forms control named `cAvax1`. Paste the procedures into the form class. Example setup and supporting files explains references, initialization and file paths.

Add these Windows Forms controls or components to the form. Fields referenced by Handles clauses must be declared WithEvents, as the Visual Studio designer normally does.

Name	Type
Command1	Button

VB.NET example

```
Private Sub Form1_Load(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs) Handles
    MyBase.Load
    Command1.Text = "GetLineStyleNames"
    cAvax1.StartAvax()
End Sub
Private Sub Form1_FormClosed(ByVal eventSender As System.Object, ByVal eventArgs As
    System.Windows.Forms.FormClosedEventArgs) Handles Me.FormClosed
    cAvax1.EndAvax()
End Sub
Private Sub Command1_Click(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs)
    Handles Command1.Click
```

```
    Dim sLineStyleNames() As String = Nothing
    Dim lMax, lCnt As Integer
    Dim sMsg As String = String.Empty
    lMax = cAvax1.GetLineStyleNames(sLineStyleNames)
    For lCnt = 1 To lMax
        sMsg = sMsg & sLineStyleNames(lCnt) & vbCrLf
    Next lCnt
    MsgBox(sMsg, MsgBoxStyle.Information, "GetLineStyleNames")
End Sub
```

GetLineStyles

cAvax method

Purpose

This function returns the total number and the data of the line types

VB.NET declaration

```
Function GetLineStyles(ByRef sLineStyles As String()) As Integer
```

Parameters

Name	Type and passing	Meaning
sLineStyles	ByRef sLineStyles As String()	ByRef detached String() output, including reserved index zero; entries 1..returned Integer count are usable.

Use the declaration for the overload you call. ByRef arguments can receive outputs; declare variables and arrays with the exact indicated type. Optional defaults are shown in the declaration.

Return value

Integer - Is the total number of the line types

Example

Setup

Add an Avax.cAvax Windows Forms control named `cAvax1`. Paste the procedures into the form class. Example setup and supporting files explains references, initialization and file paths.

Add these Windows Forms controls or components to the form. Fields referenced by Handles clauses must be declared WithEvents, as the Visual Studio designer normally does.

Name	Type
Command1	Button

VB.NET example

```
Private Sub Form1_Load(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs) Handles
    MyBase.Load
    Command1.Text = "GetLineStyles"
    cAvax1.StartAvax()
End Sub
Private Sub Form1_FormClosed(ByVal eventSender As System.Object, ByVal eventArgs As
    System.Windows.Forms.FormClosedEventArgs) Handles Me.FormClosed
    cAvax1.EndAvax()
End Sub
Private Sub Command1_Click(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs)
    Handles Command1.Click
    Dim sLineStyles() As String = Nothing
    Dim lMax, lCnt As Integer
    Dim sMsg As String = String.Empty
    lMax = cAvax1.GetLineStyles(sLineStyles)
    For lCnt = 1 To lMax
        sMsg = sMsg & sLineStyles(lCnt) & vbCrLf
    Next lCnt
    MsgBox(sMsg, MsgBoxStyle.Information, "GetLineStyles")
End Sub
```

GetOSnapStatus

cAvax method

Purpose

Use this method if you want to get the OSnap value the user has set

VB.NET declaration

```
Function GetOSnapStatus() As AvaxOSnapType
```

Return value

Is the OSnap value

Remarks

Returns the pending one-shot object-snap mode selected by a snap command, as in R2. A matching point need not have been found. Original modes keep enum values 1 through 9; Center_o is the v3 extension with value 10. The automatic mask is a separate setting and is not returned by this getter. Use AutoOSnapType or UserOsnap event data to inspect a resolved candidate and its coordinates. A pending mode remains available while point acceptance is paused or edit-locked. Stopped, disabled and disposed controls return None_o; reenabling restores an unconsumed request. Cancellation, a replacement input command, a new drawing and restart clear the request. This read-only query does not resolve a snap, raise snap events or modify the drawing.

Example

Setup

Add an Avax.cAvax Windows Forms control named `cAvax1`. Paste the procedures into the form class. Example setup and supporting files explains references, initialization and file paths.

Add these Windows Forms controls or components to the form. Fields referenced by Handles clauses must be declared WithEvents, as the Visual Studio designer normally does.

Name	Type
Button1	Button
Button2	Button

VB.NET example

```
Private Sub Form1_Load(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs) Handles
    MyBase.Load
    Call CAvax1.StartAvax()
    Call CAvax1.Add_Line(0, 0, 10, 10, 10, 0)
    CAvax1.Command = Avax.AvaxCommand.Line_c
End Sub
Private Sub Form1_FormClosed(ByVal eventSender As System.Object, ByVal eventArgs As
    System.Windows.Forms.FormClosedEventArgs) Handles Me.FormClosed
    Call cAvax1.EndAvax()
End Sub
Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
    Button1.Click
    CAvax1.Command = Avax.AvaxCommand.Endpoint_c
End Sub
Private Sub Button2_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
    Button2.Click
    Dim Answer As Avax.AvaxOSnapType
    Answer = CAvax1.GetOSnapStatus
    MsgBox("OSnap Status=" & Answer)
End Sub
```

GetPalColor

cAvax method

Purpose

This function returns the RGB color associated with the position (index) of the current color palette

VB.NET declaration

```
Function GetPalColor(ByVal index0 As Integer) As Integer
```

Parameters

Name	Type and passing	Meaning
index0	ByVal index0 As Integer	Zero-based Integer index, promoted from Integer. -1 means background. See the method notes for legacy special indices and unverified selection/printing contexts.

Use the declaration for the overload you call. ByRef arguments can receive outputs; declare variables and arrays with the exact indicated type. Optional defaults are shown in the declaration.

Return value

Integer - Is the RGB color in the specific position of the current color palette_Remarks=

Remarks

Returns an effective display color as an OLE/BGR Integer. Ordinary indices use the drawing palette and CheckBackColorOn contrast; an invalid ordinary index falls back to palette entry zero. Reserved query values take precedence even when an imported palette has more than 256 entries: -1 and 256 return the background, 257 returns the grid color and 259 the cursor color with automatic background contrast. Value 258 uses the color of the most recently drawn visible object, including nested vector leaves; its initial color index is zero. Changing PenColor alone does not change that context. An invalid current-object reference falls back to the selection RGB color. Context is local to each control and background scene preparation does not replace its UI drawing context. During a paper operation,background queries use PrintPaperRGBColor and ordinary colors are contrasted against paper regardless of CheckBackColorOn. Screen background queries retain an OLE system-color identifier when applicable; contrast uses its resolved RGB channels. Queries do not modify the drawing or raise Change. Use GetPalette for the underlying raw color array,including raw entries 256 through 259. The current-object context follows screen painting, retained geometry pictures, adaptive detail tiles and vector/raster exports. Hidden objects and excluded selection-only objects do not replace it. Inside SetExternalLines, value258 describes the external reference whose coordinates are being requested, before the handler returns them. Reserved query values describe display roles; GetPalette returns the underlying resource-array entries, including expanded imported resource slots with those same numbers.

Numeric changes from v2

Update the following variables, arrays or event-data receivers when porting the corresponding v2 call. The declarations above show the current overloads.

Parameter or value	v2 type	v3 type
index0	Short	Integer

For ByRef parameters, use a variable with the exact v3 type. Preserve array dimensions and the documented indexing. Moving from v2 to v3 explains conversion and legacy file-storage rules.

Example

Setup

Add an Avax.cAvax Windows Forms control named `cAvax1`. Paste the procedures into the form class. Example setup and supporting files explains references, initialization and file paths.

Add these Windows Forms controls or components to the form. Fields referenced by Handles clauses must be declared WithEvents, as the Visual Studio designer normally does.

Name	Type
Command1	Button

VB.NET example

```
Private Sub Form1_Load(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs) Handles
    MyBase.Load
    Command1.Text = "GetPalColor"
    cAvax1.StartAvax()
End Sub
Private Sub Form1_FormClosed(ByVal eventSender As System.Object, ByVal eventArgs As
    System.Windows.Forms.FormClosedEventArgs) Handles Me.FormClosed
    cAvax1.EndAvax()
End Sub
Private Sub Command1_Click(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs)
    Handles Command1.Click
    Dim Index0 As Integer
    Index0 = 5
    MsgBox("Color " & Format(Index0) & " = " & CAvax1.GetPalColor(Index0), MsgBoxStyle.Information,
    "GetPalColor")
End Sub
```

GetPalette

cAvax method

Purpose

This function returns the total number and the RGB colors of the current color palette

VB.NET declaration

```
Function GetPalette(ByRef ClrPalette As Integer()) As Integer
```

Parameters

Name	Type and passing	Meaning
ClrPalette	ByRef ClrPalette As Integer()	Output: detached, zero-based Integer() containing raw OLE/BGR values; index zero is valid. The Integer return value is the total count.

Use the declaration for the overload you call. ByRef arguments can receive outputs; declare variables and arrays with the exact indicated type. Optional defaults are shown in the declaration.

Return value

Integer - Is the total number of the palette colors

Numeric changes from v2

Update the following variables, arrays or event-data receivers when porting the corresponding v2 call. The declarations above show the current overloads.

Parameter or value	v2 type	v3 type
Return value	Short	Integer

For ByRef parameters, use a variable with the exact v3 type. Preserve array dimensions and the documented indexing. Moving from v2 to v3 explains conversion and legacy file-storage rules.

Example

Setup

Add an Avax.cAvax Windows Forms control named `cAvax1`. Paste the procedures into the form class. Example setup and supporting files explains references, initialization and file paths.

Add these Windows Forms controls or components to the form. Fields referenced by Handles clauses must be declared WithEvents, as the Visual Studio designer normally does.

Name	Type
Command1	Button

VB.NET example

```
Private Sub Form1_Load(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs) Handles
    MyBase.Load
    Command1.Text = "GetPalette"
    cAvax1.StartAvax()
End Sub
Private Sub Form1_FormClosed(ByVal eventSender As System.Object, ByVal eventArgs As
    System.Windows.Forms.FormClosedEventArgs) Handles Me.FormClosed
    cAvax1.EndAvax()
End Sub
Private Sub Command1_Click(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs)
    Handles Command1.Click
```

```
    Dim ClrPalette() As Integer = Nothing
    Dim iMax, iCnt As Integer
    iMax = cAvax1.GetPalette(ClrPalette)
    For iCnt = 0 To iMax - 1
        Console.WriteLine(ClrPalette(iCnt))
    Next iCnt
End Sub
```

GetPaperSize

API reference

Returns the nominal dimensions and English description of a paper size. Dimensions are measured in millimeters.

Syntax

```
Public Function GetPaperSize(
    ByRef PaperSize As AvaxPaperSize,
    ByRef pWidthMM As Double,
    ByRef pHeightMM As Double,
    ByRef pDescription As String
) As Boolean
```

Parameters

Name	Type	Meaning
PaperSize	AvaxPaperSize	Input AvaxPaperSize value. Passed ByRef for compatibility, but unchanged on success and failure.
pWidthMM	Double	Output nominal width in millimeters as Double. Set only on success; zero for epsUser.
pHeightMM	Double	Output nominal height in millimeters as Double. Set only on success; zero for epsUser.
pDescription	String	Output English paper description. Set only on success; "User-defined" for epsUser.

Return value

`True` if the value is recognized; otherwise `False`. On failure, all four arguments retain their previous values. There is no fallback to A4 for this query.

Remarks

The method can be called before `StartAvax`. It does not query installed printers, print a document or change the drawing. Call it on the control's UI thread.

`epsUser` identifies user-defined paper and returns `0` for both dimensions. It does not return the dimensions stored in `PrintPaperX_MM_p` and `PrintPaperY_MM_p`. Use those properties when you need the configured custom print size.

Dimensions describe the nominal paper rectangle, before page orientation or printer margins. The descriptions remain in English regardless of the application's current language or numeric culture.

Example (VB.NET)

```
' Avax1 is the cAvax control on this Windows Forms form.
Dim paper As Avax.AvaxPaperSize = Avax.AvaxPaperSize.epsA3
Dim widthMm As Double
Dim heightMm As Double
Dim description As String = ""

If Avax1.GetPaperSize(PaperSize:=paper,
                      pWidthMM:=widthMm,
                      pHeightMM:=heightMm,
                      pDescription:=description) Then
    MessageBox.Show(description & ": " & widthMm.ToString() &
                    " x " & heightMm.ToString() & " mm")
End If
' Result: A3, width 297 mm, height 420 mm.
```

Paper sizes

Enumeration member	Value	Width (mm)	Height (mm)	Description
epsLetter	1	215.9	279.4	Letter
epsLetterSmall	2	215.9	279.4	Letter Small
epsTabloid	3	279.4	431.8	Tabloid
epsLedger	4	431.8	279.4	Ledger
epsLegal	5	215.9	355.6	Legal
epsStatement	6	139.7	215.9	Statement
epsExecutive	7	184.15	266.7	Executive
epsA3	8	297	420	A3
epsA4	9	210	297	A4
epsA4Small	10	210	297	A4 Small
epsA5	11	148	210	A5
epsB4	12	250	354	B4
epsB5	13	182	257	B5
epsFolio	14	215.9	330.2	Folio
epsQuarto	15	215	275	Quarto
eps10x14	16	254	355.6	10 in. x 14 in.
eps11x17	17	279.4	431.8	11 in. x 17 in.
epsNote	18	215.9	279.4	Note
epsEnv9	19	98.425	225.425	Envelope #9
epsEnv10	20	104.775	241.3	Envelope #10
epsEnv11	21	114.3	263.525	Envelope #11
epsEnv12	22	120.65	279.4	Envelope #12
epsEnv14	23	127	292.1	Envelope #14
epsCSheet	24	431.8	558.8	C size sheet
epsDSheet	25	558.8	863.6	D size sheet
epsESheet	26	863.6	1117.6	E size sheet
epsEnvDL	27	110	220	Envelope DL
epsEnvC3	28	324	458	Envelope C3
epsEnvC4	29	229	324	Envelope C4
epsEnvC5	30	162	229	Envelope C5
epsEnvC6	31	114	162	Envelope C6
epsEnvC65	32	114	229	Envelope C65
epsEnvB4	33	250	353	Envelope B4
epsEnvB5	34	176	250	Envelope B5
epsEnvB6	35	176	125	Envelope B6
epsEnvItaly	36	110	230	Envelope
epsenvmonarch	37	98.425	190.5	Envelope Monarch

Enumeration member	Value	Width (mm)	Height (mm)	Description
epsEnvPersonal	38	92.075	165.1	Envelope
epsFanfoldUS	39	377.825	279.4	U.S. Standard Fanfold
epsFanfoldStdGerman	40	215.9	304.8	German Standard Fanfold
epsFanfoldLglGerman	41	215.9	330.2	German Legal Fanfold
epsA0	272	841	1189	841 mm x 1189 mm
epsA1	271	594	841	594 mm x 841 mm
epsA2	66	420	594	420 mm x 594 mm
epsUser	256	0	0	User-defined

Migrating from v2

Declare the width and height variables as `Double` instead of `Single`. Parameter names, their `ByRef` calling convention, the Boolean result, and English descriptions are retained.

Executive is 184.15 × 266.7 mm and Envelope #12 is 120.65 × 279.4 mm. C, D and E sheet sizes retain their fractional millimeters. These correct the older table and agree with the dimensions used by `PrintToPDF`.

GetParametricShapeKind

Purpose

Identifies the parametric preset associated with an existing drawing object.

Declaration

```
Function GetParametricShapeKind(ByVal handle As Integer) As String
```

Parameters and result

`handle` is the object's Integer handle. Returns the preset Kind identifier, or an empty string if the object has no parametric recipe, the handle is unavailable, or the control is unstarted or disposed. Reading is available while the control is disabled. ConvertShapeToCurves, ChangeCompoundPath and ChangeShapeClosed clear the parametric association when they replace the preset with an independently edited path.

VB.NET example

```
Option Strict On
Public Module GetParametricShapeKindExample
    Public Function IsParametric(cad As Avax.cAvax, handle As Integer) As Boolean
        Dim kind As String = cad.GetParametricShapeKind(handle)
        If kind.Length = 0 Then Return False
        System.Diagnostics.Debug.WriteLine("Preset: " & kind)
        Return True
    End Function
End Module
```

Related options and reference

GetParametricShapes

Purpose

Reads the built-in shape catalog and its parameter descriptions so an application can populate a shape gallery and parameter editors. The catalog can be read before starting a drawing control.

Declaration

```
Function GetParametricShapes() As AvaxShapeDefinition()
```

Result

Returns a new array of AvaxShapeDefinition values. Pass each Kind to AddParametricShape and each parameter Name as a dictionary key. Use these identifiers exactly as returned. Editing returned descriptions changes only the caller's values. It returns the same 18 presets as ParametricShapeDefinitions. See Built-in shape catalog for the complete preset and parameter table.

AvaxShapeDefinition value	Type	Meaning
Kind	String	Preset identifier used by the control.
DisplayName	String	Caption for a gallery or menu.
Category	String	Suggested grouping for the user interface.
Parameters	AvaxShapeParameterInfo()	Parameters accepted by this preset. It can be empty.

AvaxShapeParameterInfo value	Type	Meaning
Name	String	Dictionary key and parameter identifier.
DisplayName	String	Human-readable parameter label.
Minimum, Maximum	Double	Inclusive supported value range.
DefaultValue	Double	Value used when the caller omits this parameter.
Increment	Double	Suggested editor step: 1 for an integer parameter, 0.01 otherwise. It does not round submitted values.
IsInteger	Boolean	True requires a whole number within the supported range.

Both data types have a public parameterless constructor. The control supplies populated descriptions; constructing or editing a description does not create a shape or register a new preset. A manually constructed definition has uninitialized String and Parameters properties until the caller supplies them. A manually constructed parameter description has zero numeric values and False IsInteger, except that Increment starts at 0.01.

VB.NET example

```
Option Strict On
Public Module GetParametricShapesExample
    Public Sub ListCatalog(cad As Avax.cAvax)
        For Each shape As Avax.AvaxShapeDefinition In cad.GetParametricShapes()
            System.Diagnostics.Debug.WriteLine(shape.Kind & ": " & shape.DisplayName)
            For Each parameter As Avax.AvaxShapeParameterInfo In shape.Parameters
                System.Diagnostics.Debug.WriteLine(parameter.Name & " = " &
                    parameter.DefaultValue & " [" & parameter.Minimum &
                    ", " & parameter.Maximum & "]")
            Next
        Next
    End Sub
End Module
```

Related options and reference

GetPathSegmentParameters

Purpose

Reads named numeric parameters for one segment without modifying the object.

Declaration

```
Function GetPathSegmentParameters(ByVal handle As Integer, ByVal figure As Integer, ByVal segment As
    Integer) As Dictionary(Of String, Double)
```

Parameters and result

`handle` identifies an object readable by GetCompoundPath. `figure` and `segment` are zero-based indices into Figures and that figure's Segments list. Segment 0 is the first segment; its endpoint is node 1. Returns an independent dictionary, or an empty dictionary for an absent/unreadable path or invalid indices. Changing this dictionary does not edit the drawing.

Segment kind	Returned keys
All kinds	EndX, EndY.
Bezier	Also Control1X, Control1Y, Control2X, Control2Y.
Arc	Also CenterX, CenterY, RadiusX, RadiusY, StartAngle, SweepAngle.

Coordinates and radii are Double drawing units. Angles are degrees and SweepAngle is signed. RadiusX and RadiusY are the lengths of the arc's U/V axis vectors, including for rotated or affine ellipses. Parameter names use the exact case shown. Use ChangePathSegmentParameter to commit a named change.

VB.NET example

```
Option Strict On
Public Module GetPathSegmentParametersExample
    Public Sub ShowFirstSegment(cad As Avax.cAvax, handle As Integer)
        Dim values As System.Collections.Generic.Dictionary(Of String, Double) =
            cad.GetPathSegmentParameters(handle, 0, 0)
        For Each item As System.Collections.Generic.KeyValuePair(Of String, Double) In values
            System.Diagnostics.Debug.WriteLine(item.Key & " = " & item.Value)
        Next
    End Sub
End Module
```

Related options and reference

GetPerimeter

cAvax method

Purpose

Returns the XY length of an open supported object or the perimeter of a closed one, in drawing units.

VB.NET declaration

```
Function GetPerimeter(ByRef lAvaxHandle As Integer) As Double
```

Parameters

Name	Type and passing	Meaning
lAvaxHandle	ByRef lAvaxHandle As Integer	Existing entity handle (Integer, passed ByRef for R2 source compatibility; not modified). Missing, zero and negative handles return zero without changing the drawing.

Use the declaration for the overload you call. ByRef arguments can receive outputs; declare variables and arrays with the exact indicated type. Optional defaults are shown in the declaration.

Return value

Double - Nonnegative length in XY drawing units, or 0 for unsupported, invalid or unavailable input.

Remarks

A line contributes its XY endpoint distance and a polyline contributes the lengths of its stored segments. A closed shape includes its closing side; an open shape remains open. Analytic ellipses/arcs use their transformed curved lengths. Compound-path figures are measured separately, with a closing side only for closed figures. Native hatch perimeter includes every boundary loop.

The result is independent of display zoom, pen width and Z elevation. Points, ordinary text/documents, blocks and unknown handles return 0. A started, enabled control is required. Invalid or out-of-range results report an error and return 0. The handle, drawing, selection and history are retained.

Numeric changes from v2

Update the following variables, arrays or event-data receivers when porting the corresponding v2 call. The declarations above show the current overloads.

Parameter or value	v2 type	v3 type
Return value	Single	Double

For ByRef parameters, use a variable with the exact v3 type. Preserve array dimensions and the documented indexing. Moving from v2 to v3 explains conversion and legacy file-storage rules.

Example

Setup

Add an Avax.cAvax Windows Forms control named `cAvax1`. Paste the procedures into the form class. Example setup and supporting files explains references, initialization and file paths.

Add these Windows Forms controls or components to the form. Fields referenced by Handles clauses must be declared WithEvents, as the Visual Studio designer normally does.

Name	Type
Command1	Button

VB.NET example

```
Dim lAvaxHandle As Integer
Private Sub Form1_Load(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs) Handles
    MyBase.Load
    cAvax1.StartAvax()
    lAvaxHandle = cAvax1.Add_Rectangle(0, 0, 10, 12)
    Call CAvax1.SetAvaxProperty(Avax.AvaxProp.DimScale_n, 5)
    Call cAvax1.Add_AllDimensions(2, 0)
End Sub
Private Sub Form1_FormClosed(ByVal eventSender As System.Object, ByVal eventArgs As
    System.Windows.Forms.FormClosedEventArgs) Handles Me.FormClosed
    cAvax1.EndAvax()
End Sub
Private Sub Command1_Click(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs)
    Handles Command1.Click
    Dim Answer As Double
    Answer = cAvax1.GetPerimeter(lAvaxHandle)
    MsgBox("Perimeter =" & Str(Answer))
End Sub
```

GetPictureSize

cAvax method

Purpose

Returns the width and the height (in drawing units) of the picture that is in a specific picture file

VB.NET declaration

```
Function GetPictureSize(ByRef sPictureFile As String, ByRef aWidth As Double, ByRef aHeight As Double) As
    Boolean
```

Parameters

Name	Type and passing	Meaning
sPictureFile	ByRef sPictureFile As String	Path to the picture file. Passed ByRef for compatibility; the path is not modified.
aWidth	ByRef aWidth As Double	Receives the width in drawing units as Double on success; retains its original value on failure.
aHeight	ByRef aHeight As Double	Receives the height in drawing units as Double on success; retains its original value on failure.

Use the declaration for the overload you call. ByRef arguments can receive outputs; declare variables and arrays with the exact indicated type. Optional defaults are shown in the declaration.

Return value

Boolean - True on success, false on error.

Remarks

Returns the image width and height in drawing units. Each dimension equals the source pixel count multiplied by 25.4, divided by the control display DPI for that axis and by PrintUnit_MM_p. The image file's embedded print resolution does not override the display DPI. The default PrintUnit_MM_p is 10; a nonpositive value uses that default. The method can read a file before StartAvax and while the control is disabled. The file path is preserved. Missing, invalid or unreadable images, a disposed control, and dimensions outside the finite Double range return False without changing either output. The input file is released before the method returns. Standard Windows image formats, WMF/EMF, icons, PCX and the first image of a supported animation are accepted. AVI dimensions come from the video stream header; no codec, playback or camera is needed. A top-down AVI frame has a negative stored height, which is returned as a positive height. A missing or malformed video header returns False.

Numeric changes from v2

Update the following variables, arrays or event-data receivers when porting the corresponding v2 call. The declarations above show the current overloads.

Parameter or value	v2 type	v3 type
aWidth	Single	Double
aHeight	Single	Double

For ByRef parameters, use a variable with the exact v3 type. Preserve array dimensions and the documented indexing. Moving from v2 to v3 explains conversion and legacy file-storage rules.

Example

Setup

Add an Avax.cAvax Windows Forms control named `cAvax1`. Paste the procedures into the form class. Example setup and supporting files explains references, initialization and file paths.

Add these Windows Forms controls or components to the form. Fields referenced by Handles clauses must be declared WithEvents, as the Visual Studio designer normally does.

Name	Type
Command1	Button

VB.NET example

```
Private Sub Form1_Load(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs) Handles
    MyBase.Load
    Command1.Text = "GetPictureSize"
    cAvax1.StartAvax()
End Sub
Private Sub Form1_FormClosed(ByVal eventSender As System.Object, ByVal eventArgs As
    System.Windows.Forms.FormClosedEventArgs) Handles Me.FormClosed
    cAvax1.EndAvax()
End Sub
Private Sub Command1_Click(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs)
    Handles Command1.Click
    Dim sPictureFile As String = String.Empty
    Dim aWidth As Double
    Dim aHeight As Double
    Dim sPictures(1) As String
    sPictureFile = My.Application.Info.DirectoryPath & "\..\Clouds1.bmp"
    Call cAvax1.SetAvaxProperty(Avax.AvaxProp.PrintUnit, MM, 1)
    If cAvax1.GetPictureSize(sPictureFile, aWidth, aHeight) = True Then
        MsgBox("Width=" & aWidth & vbCrLf & "Height=" & aHeight, MsgBoxStyle.Information,
    "GetPictureSize")
        sPictures(1) = sPictureFile
        Call cAvax1.Add_Picture(sPictures, 0, 0, aWidth, aHeight, 0, Nothing, -1, 8)
    End If
End Sub
```

GetProperties

cAvax method

Purpose

Reads object types, application metadata and geometry into detached one-based arrays. Use SetProperties to apply edits.

VB.NET declaration

```
Function GetProperties(ByRef avMyHandles As Integer(), ByRef iType As AvaxItemType(), ByRef vItemProperies
     As Object(), ByRef vData As Object()) As Boolean
```

Parameters

Name	Type and passing	Meaning
avMyHandles	ByRef avMyHandles As Integer()	One-based requested handles, with index 0 reserved. Returned geometry/metadata are detached copies.
iType	ByRef iType As AvaxItemType()	One-based AvaxItemType() matching the requested handles. SetProperties does not convert item types.
vItemProperies	ByRef vItemProperies As Object()	One-based Object() containing per-object string arrays; metadata and geometry are copied.
vData	ByRef vData As Object()	One-based Object() of per-item Double() arrays; see the Property geometry table for all exact layouts. Length includes reserved index 0, which is meaningful only for Curve_i. Block/ExLines_i expose six XYZ bounds. Data_i has no editable geometry.

Use the declaration for the overload you call. ByRef arguments can receive outputs; declare variables and arrays with the exact indicated type. Optional defaults are shown in the declaration.

Return value

Boolean - True on success, false on error.

Remarks

The input handles and outer output arrays use logical index 1; index 0 is reserved. Outputs contain item types, copied String() application metadata inside Object(), and per-object Double() geometry inside Object(). Use the Property geometry layouts topic to interpret each object type. Editing returned arrays does not change the drawing.

A started, enabled control is required. Empty requests or unknown handles return False. Outputs are published together only after the complete query succeeds. External line sources can be refreshed during the query; a callback that replaces the drawing or invalidates the control prevents stale outputs. No drawing edit or Undo operation is created.

Property geometry layouts

The Object() containers keep their declared type. Drawing-coordinate arrays stored inside them use Double(); use the documented per-object layout when reading or replacing those arrays.

Example

Setup

Add an Avax.cAvax Windows Forms control named `cAvax1`. Paste the procedures into the form class. Example setup and supporting files explains references, initialization and file paths.

Add these Windows Forms controls or components to the form. Fields referenced by Handles clauses must be declared WithEvents, as the Visual Studio designer normally does.

Name	Type
Command1	Button

VB.NET example

```
Private Sub Form1_Load(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs) Handles
    MyBase.Load
    Command1.Text = "Get/Set Object Properties"
    cAvax1.StartAvax()
    Call cAvax1.Add_Line(0, 0, 0, 10, 10, 0)
    Call cAvax1.Add_Line(0, 0, 0, 10, 10, 0)
End Sub
Private Sub Form1_FormClosed(ByVal eventSender As System.Object, ByVal eventArgs As
    System.Windows.Forms.FormClosedEventArgs) Handles Me.FormClosed
    cAvax1.EndAvax()
End Sub
Private Sub Command1_Click(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs)
    Handles Command1.Click
    Dim [Handles]() As Integer = Nothing
    Dim iType() As Avax.AvaxItemType = Nothing
    Dim vItemProperies() As Object = Nothing
    Dim vData() As Object = Nothing
    Dim lMax As Integer
    Dim lCnt As Integer
    Dim xyz() As Double = Nothing
    lMax = CAvax1.GetAllHandlesArr([Handles])
    If CAvax1.GetProperties([Handles], iType, vItemProperies, vData) = True Then
        For lCnt = 1 To lMax
            If iType(lCnt) = Avax.AvaxItemType.Line i Then
                xyz = vData(lCnt)
                xyz(1) = xyz(1) + 5
                xyz(2) = xyz(2) + 5
                xyz(4) = xyz(4) + 5
                xyz(5) = xyz(5) + 5
                vData(lCnt) = xyz
            End If
        Next lCnt
        Call CAvax1.SetProperties([Handles], iType, vItemProperies, vData)
    End If
End Sub
```

GetSelectedHandlesArr

cAvax method

Purpose

Returns the total number and the handles of the selected objects

VB.NET declarations

```
Function GetSelectedHandlesArr(ByRef avMyHandles As Integer(), ByRef fzSort As Boolean) As Integer

Function GetSelectedHandlesArr(ByRef avMyHandles As Integer()) As Integer
```

Parameters

Name	Type and passing	Meaning
avMyHandles	ByRef avMyHandles As Integer()	Output Integer() containing the returned entity handles at indices 1 through the returned count. Index 0 is reserved.
fzSort	ByRef fzSort As Boolean	A Boolean expression that if it is true, then the selected objects are sorted by their z order

Use the declaration for the overload you call. ByRef arguments can receive outputs; declare variables and arrays with the exact indicated type. Optional defaults are shown in the declaration.

Return value

Integer - Is the total number of the selected objects

Example

Setup

Add an Avax.cAvax Windows Forms control named `cAvax1`. Paste the procedures into the form class. Example setup and supporting files explains references, initialization and file paths.

Add these Windows Forms controls or components to the form. Fields referenced by Handles clauses must be declared WithEvents, as the Visual Studio designer normally does.

Name	Type
Command1	Button
Text1	TextBox
Label1	Label

VB.NET example

```
Private Sub Form1_Load(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs) Handles
    MyBase.Load
    Command1.Text = "SelectAll"
    Label1.Text = "Selected objects handles:"
    cAvax1.StartAvax()
    Call cAvax1.Add_Line(0, 0, 0, 10, 10, 0)
    Call cAvax1.Add_Line(0, 10, 0, 10, 0, 0)
    Call cAvax1.Add_Line(5, 0, 0, 5, 10, 0)
    Call cAvax1.Add_Line(0, 5, 0, 10, 5, 0)
    Call cAvax1.Add_Rectangle(0, 0, 10, 10)
End Sub
Private Sub Form1_FormClosed(ByVal eventSender As System.Object, ByVal eventArgs As
    System.Windows.Forms.FormClosedEventArgs) Handles Me.FormClosed
    cAvax1.EndAvax()
End Sub
```

```
Private Sub CAvax1_SelectChange(ByVal Sender As Object, ByVal e As Avax.cAvax.SelectChangeEventArgs)
    Handles CAvax1.SelectChange
    Dim [Handles]() As Integer = Nothing
    Dim lCnt, lMax As Integer
    Dim sText As String = String.Empty
    lMax = CAvax1.GetSelectedHandlesArr([Handles])
    For lCnt = 1 To lMax
        sText = sText & Str([Handles](lCnt)) & vbCrLf
    Next lCnt
    Text1.Text = sText
End Sub
Private Sub Command1_Click(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs)
    Handles Command1.Click
    CAvax1.Command = Avax.AvaxCommand.Select_All_c
End Sub
```

GetSelectedPoints

cAvax method

Purpose

Returns the coordinates and the number of points selected by the activation of the Select Points process with the method SelectPoints or the setting property Command=AvaxCommand.selectv_c

VB.NET declaration

```
Function GetSelectedPoints(ByRef x As Double(), ByRef y As Double(), ByRef z As Double()) As Integer
```

Parameters

Name	Type and passing	Meaning
x	ByRef x As Double()	An array for the coordinates x of the selected points
y	ByRef y As Double()	An array for the coordinates y of the selected points
z	ByRef z As Double()	An array for the coordinates z of the selected points

Use the declaration for the overload you call. ByRef arguments can receive outputs; declare variables and arrays with the exact indicated type. Optional defaults are shown in the declaration.

Return value

Integer - Is the number of the selected points

Numeric changes from v2

Update the following variables, arrays or event-data receivers when porting the corresponding v2 call. The declarations above show the current overloads.

Parameter or value	v2 type	v3 type
x	Single()	Double()
y	Single()	Double()
z	Single()	Double()

For ByRef parameters, use a variable with the exact v3 type. Preserve array dimensions and the documented indexing. Moving from v2 to v3 explains conversion and legacy file-storage rules.

Example

Setup

Add an Avax.cAvax Windows Forms control named `cAvax1`. Paste the procedures into the form class. Example setup and supporting files explains references, initialization and file paths.

Add these Windows Forms controls or components to the form. Fields referenced by Handles clauses must be declared WithEvents, as the Visual Studio designer normally does.

Name	Type
Label1	Label

VB.NET example

```
Private Sub CAvax1_OnMouseMove(ByVal Sender As Object, ByVal e As Avax.cAvax.MouseMoveEventArgs) Handles
    CAvax1.OnMouseMove
    Dim x1() As Double = Nothing
    Dim y1() As Double = Nothing
```

```
    Dim z1() As Double = Nothing
    Dim lMax As Integer
    lMax = CAvax1.GetSelectedPoints(x1, y1, z1)
    If lMax = 2 Then
        Call CAvax1.Draw_Line(x1(1), y1(1), x1(2), y1(2), 5, 2, 1)
    End If
End Sub
Private Sub CAvax1_SelectedPoints(ByVal Sender As Object, ByVal e As Avax.cAvax.SelectedPointsEventArgs)
    Handles CAvax1.SelectedPoints
    Select Case e.SelectId
        Case "Create triangle"
        If e.lPointNumber = 3 Then
            Call CAvax1.Add_Shape(e.x1, e.y1, e.z1, Nothing, 1, 3, 2, 0, "", 0, 0, -1, -1, -1, -1,
    Avax.AvaxPolylineType.Polyline_pt)
        End If
    End Select
End Sub
Private Sub Form1_Load(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs) Handles
    MyBase.Load
    Dim sSelectId As String = String.Empty
    Dim lPointsToSelect As Integer
    Dim ExistX() As Double = Nothing
    Dim ExistY() As Double = Nothing
    Dim Exist7() As Double = Nothing
    Dim fBoxOn As Boolean
    Dim fNoSnapAndOrtho As Boolean
    Dim Answer As Boolean
    Label1.Text = "Select points to make a triangle..."
    CAvax1.StartAvax()
    lPointsToSelect = 3
    ReDim ExistX(1)
    ReDim ExistY(1)
    ReDim Exist7(1)
    ExistX(1) = 3
    ExistY(1) = 3
    sSelectId = "Create triangle"
    Answer = CAvax1.SelectPoints(sSelectId, lPointsToSelect, ExistX, ExistY, Exist7, fBoxOn,
    fNoSnapAndOrtho)
End Sub
Private Sub Form1_FormClosed(ByVal eventSender As System.Object, ByVal eventArgs As
    System.Windows.Forms.FormClosedEventArgs) Handles Me.FormClosed
    CAvax1.EndAvax()
End Sub
```

GetSelectRegionBorders

cAvax method

Purpose

Returns the extents of the SelectRegion

VB.NET declaration

```
Function GetSelectRegionBorders(ByRef xMin As Double, ByRef yMin As Double, ByRef xMax As Double, ByRef
    yMax As Double) As Boolean
```

Parameters

Name	Type and passing	Meaning
xMin	ByRef xMin As Double	Is the x coordinate of the bottom left extend of the SelectRegion
yMin	ByRef yMin As Double	Is the y coordinate of the bottom left extend of the SelectRegion
xMax	ByRef xMax As Double	Is the x coordinate of the top right extend of the SelectRegion
yMax	ByRef yMax As Double	Is the y coordinate of the top right extend of the SelectRegion

Use the declaration for the overload you call. ByRef arguments can receive outputs; declare variables and arrays with the exact indicated type. Optional defaults are shown in the declaration.

Return value

Boolean - True on success, false on error.

Numeric changes from v2

Update the following variables, arrays or event-data receivers when porting the corresponding v2 call. The declarations above show the current overloads.

Parameter or value	v2 type	v3 type
xMin	Single	Double
yMin	Single	Double
xMax	Single	Double
yMax	Single	Double

For ByRef parameters, use a variable with the exact v3 type. Preserve array dimensions and the documented indexing. Moving from v2 to v3 explains conversion and legacy file-storage rules.

Example

Setup

Add an Avax.cAvax Windows Forms control named `cAvax1`. Paste the procedures into the form class. Example setup and supporting files explains references, initialization and file paths.

Add these Windows Forms controls or components to the form. Fields referenced by Handles clauses must be declared WithEvents, as the Visual Studio designer normally does.

Name	Type
Command1	Button

VB.NET example

```
Private Sub Form1_Load(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs) Handles
```

```
    MyBase.Load
    Command1.Text = "GetSelectRegionBorders"
    cAvax1.StartAvax()
    Call cAvax1.Add_Line(0, 0, 0, 10, 10, 0)
End Sub
Private Sub Form1_FormClosed(ByVal eventSender As System.Object, ByVal eventArgs As
    System.Windows.Forms.FormClosedEventArgs) Handles Me.FormClosed
    cAvax1.EndAvax()
End Sub
Private Sub Command1_Click(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs)
    Handles Command1.Click
    Dim sMsg As String = String.Empty
    Dim xMin As Double
    Dim yMin As Double
    Dim xMax As Double
    Dim yMax As Double
    cAvax1.Command = Avax.AvaxCommand.Select_All_c
    If cAvax1.GetSelectRegionBorders(xMin, yMin, xMax, yMax) = True Then
        sMsg = "xMin=" & Format(xMin) & vbCrLf
        sMsg = sMsg & "yMin=" & Format(yMin) & vbCrLf
        sMsg = sMsg & "xMax=" & Format(xMax) & vbCrLf
        sMsg = sMsg & "yMax=" & Format(yMax)
    End If
    MsgBox(sMsg, MsgBoxStyle.Information, "GetSelectRegionBorders")
End Sub
```

GetShapeClosed

Purpose

Reads an explicit shape-path closed state without assuming that every visually looped outline has such a state.

VB.NET declaration

```
Function GetShapeClosed(ByVal handle As Integer) As Nullable(Of Boolean)
```

Parameters

Parameter	Meaning
handle	Integer handle in the current drawing.

Return value and behavior

Returns Nullable(Of Boolean): True for closed, False for open, and Nothing when no explicit path state is available. Native arc/polyline shapes and shapes whose state was established with ChangeShapeClosed expose a state. Other object types, embedded controls, missing handles and unstarted/ended controls return Nothing. Disabled controls remain readable.

VB.NET example

Pass a started control on its owning UI thread. This is a complete standalone module.

```
Option Strict On
Imports Avax

Public Module EditableShapeClosedExample
    Public Function DescribePath(cad As cAvax, id As Integer) As String
        Dim closed As Boolean? = cad.GetShapeClosed(id)
        If Not closed.HasValue Then Return "No path state"
        Return If(closed.Value, "Closed", "Open")
    End Function
End Module
```

Related options and reference

GetShapeControlPoints

Purpose

Reads the current world positions and identifiers of editable control points on a parametric shape or compound path. Applications can display these positions and submit an edited point through ChangeShapeControlPoint.

Declaration

```
Function GetShapeControlPoints(ByVal handle As Integer) As AvaxShapeControlPoint()
```

Parameters and result

`handle` is the drawing handle. Returns an array of AvaxShapeControlPoint values, or an empty array when no supported points are available or the control is unstarted or disposed. Reading is available while disabled. Parametric shapes expose their noninteger recipe parameter handles; a preset without such parameters returns an empty array. Edit Points and other numeric values with ChangeShapeParameter. Compound paths and supported native vector outlines expose nodes and appropriate curve controls. Read the array again after each edit because the available points and their positions can change.

Value	Type	Meaning
Id	String	Identifier to pass back for this object. Obtain it from this query rather than constructing an identifier.
X, Y	Double	Current world coordinates in drawing units.
Description	String	Caption describing the node, tangent or shape parameter.
IsTangent	Boolean	True identifies a Bezier tangent control handle.

Changing the returned data objects does not edit the drawing. Their default constructor is available when an application needs its own copy of a displayed value. Use the control's editing method to commit a change.

VB.NET example

```
Option Strict On
Public Module GetShapeControlPointsExample
    Public Sub ListPoints(cad As Avax.cAvax, handle As Integer)
        For Each point As Avax.AvaxShapeControlPoint In cad.GetShapeControlPoints(handle)
            System.Diagnostics.Debug.WriteLine(point.Id & " (" &
                point.Description & "): " & point.X & ", " & point.Y)
        Next
    End Sub
End Module
```

Related options and reference

GetShapeDocumentText

Returns a document shape's original editable content and supplies its layout metrics. DocumentTextHeight is the font height; TextWidth is the occupied text width; DocumentWidth is the logical wrapping width; DocumentHeight is the laid-out content height. Angle is the rotation in degrees, normalized to the range 0 up to but excluding 360. Widths and heights are intrinsic drawing dimensions. On failure it returns Nothing and leaves the output variables unchanged.

```
Public Function GetShapeDocumentText(ByRef hDocumentShape As Integer,
                                     ByRef DocumentTextHeight As Double,
                                     ByRef TextWidth As Double,
                                     ByRef DocumentWidth As Double,
                                     ByRef DocumentHeight As Double,
                                     ByRef Angle As Double) As String
```

Numeric changes from v2

All font heights, widths, content heights and angles shown above are Double in v3; they were Single in v2. Handles remain Integer. Declare Double variables when passing these ByRef arguments. Native .avax files retain Double precision; legacy AVX storage still has its original numeric limits.

VB.NET example

Call CreateAndResize with an AVAX control after StartAvax has succeeded. The returned Integer() contains the two object handles at indices 1 and 2. Index 0 is reserved. This example creates an ordinary document and a rotated document shape, changes their dimensions and reads the results.

```
Option Strict On
Imports System

Public Module DocumentSizeExample
    Public Function CreateAndResize(cad As Avax.cAvax) As Integer()
        Dim document As Integer = cad.Add_Document(
            "First paragraph with several words" & vbCrLf & "Second paragraph",
            10.0, 100.0, 0.0, 100.0, 4.0)
        Dim shape As Integer = cad.Add_ShapeDocument(
            "<b>Formatted<b> document", 150.0, 100.0, 3.0,
            100.0, 4.0, 30.0, Nothing, 1, 7, 1, 0, 0)
        If document = 0 OrElse shape = 0 Then
            Throw New InvalidOperationException("Could not create the documents.")
        End If

        Dim fontHeight As Double = 6.0
        Dim pageWidth As Double = 40.0
        If Not cad.ChangeDocumentFontSize(document, fontHeight) OrElse
           Not cad.ChangeDocumentWidth(document, pageWidth) Then
            Throw New InvalidOperationException("Could not resize the document.")
        End If
        pageWidth = 60.0
        If Not cad.ChangeShapeDocumentFontSize(shape, fontHeight) OrElse
           Not cad.ChangeShapeDocumentWidth(shape, pageWidth) Then
            Throw New InvalidOperationException("Could not resize the document shape.")
        End If

        Dim actualHeight As Double = cad.GetDocumentFontSize(document)
        Dim actualWidth As Double = cad.GetDocumentWidth(document)
        Dim textWidth, contentHeight, angle As Double
        Dim content As String = cad.GetShapeDocumentText(
            shape, fontHeight, textWidth, pageWidth, contentHeight, angle)
        If actualHeight <> 6.0 OrElse actualWidth <> 40.0 OrElse
           content Is Nothing Then
            Throw New InvalidOperationException("Could not read the document dimensions.")
        End If
        Return New Integer() {0, document, shape}
    End Function
End Module
```

Related options and reference

GetShapeFill

cAvax method

Purpose

Use this method if you want to get a shape's fill properties

VB.NET declaration

```
Function GetShapeFill(ByRef hShape As Integer, ByRef Fill_iType As Byte, ByRef Fill_lHatchIndex As
    Integer, ByRef Fill_iInForeColor As Integer, ByRef Fill_iInBackColor As Integer, ByRef Fill_sPattern As
    String, ByRef Fill_zOrder As Integer, ByRef Fill_lFlags As Integer) As Boolean
```

Parameters

Name	Type and passing	Meaning
hShape	ByRef hShape As Integer	Input handle of an existing item in this control. A nonshape returns the R2 No_f descriptor. The handle is unchanged.
Fill_iType	ByRef Fill_iType As Byte	Output fill family code; 8 is shape text, 11 is a shape document.
Fill_lHatchIndex	ByRef Fill_lHatchIndex As Integer	Output hatch/style index belonging to the fill family.
Fill_iInForeColor	ByRef Fill_iInForeColor As Integer	Output fill foreground color; for a nonshape, its outline color.
Fill_iInBackColor	ByRef Fill_iInBackColor As Integer	Output fill background palette index.
Fill_sPattern	ByRef Fill_sPattern As String	Output opaque pattern data. Type 8/11 data reflects the current editable text/layout.
Fill_zOrder	ByRef Fill_zOrder As Integer	Output signed Integer stacking order; zero for a nonshape.
Fill_lFlags	ByRef Fill_lFlags As Integer	Output fill option bits; preserve them when reusing the descriptor.

Use the declaration for the overload you call. ByRef arguments can receive outputs; declare variables and arrays with the exact indicated type. Optional defaults are shown in the declaration.

Return value

Boolean - True on success, false on error.

Remarks

Reads an existing item's fill descriptor into the Fill_* ByRef outputs. Returns True on success. hShape is unchanged. StartAvax must have succeeded and the control must be enabled. A missing handle or unavailable control returns False without changing any output variable. A Shape_i returns its stored fill type, hatch index, foreground/background colors, pattern, zOrder and flags. Other existing item types also succeed, as in R2: they return No_f, foreground equal to the item's outline color, an empty pattern and zero for the other fields. The current drawing pen does not determine these outputs. For text shapes (Text_f, 8) and document shapes (Document_f, 11), the opaque pattern reflects the current editable text/layout, including newly created objects and subsequent text edits. An unchanged imported pattern retains its original bytes, including custom font weights and ancillary header fields. Preserve these bytes when passing the descriptor to another fill method. Legacy encoded numeric fields keep their file-format precision even though the coordinate API uses Double. Reading the fill does not change the object, selection, geometry, drawing revision or undo history. An invalid handle reports an error.

Numeric changes from v2

Update the following variables, arrays or event-data receivers when porting the corresponding v2 call. The declarations above show the current overloads.

Parameter or value	v2 type	v3 type
Fill_iInForeColor	Short	Integer
Fill_iInBackColor	Short	Integer

For ByRef parameters, use a variable with the exact v3 type. Preserve array dimensions and the documented indexing. Moving from v2 to v3 explains conversion and legacy file-storage rules.

Example

Setup

Add an Avax.cAvax Windows Forms control named `cAvax1`. Paste the procedures into the form class. Example setup and supporting files explains references, initialization and file paths.

Add these Windows Forms controls or components to the form. Fields referenced by Handles clauses must be declared WithEvents, as the Visual Studio designer normally does.

Name	Type
Button1	Button

VB.NET example

```
Dim hShape As Integer
Private Sub Form1_Load(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs) Handles
    MyBase.Load
    Call cAvax1.StartAvax()
    Call AddShape()
End Sub
Private Sub Form1_FormClosed(ByVal eventSender As System.Object, ByVal eventArgs As
    System.Windows.Forms.FormClosedEventArgs) Handles Me.FormClosed
    Call cAvax1.EndAvax()
End Sub
Sub AddShape()
    Dim xV() As Double = Nothing
    Dim yV() As Double = Nothing
    Dim zV() As Double = Nothing
    Dim vItemProperties As Object = Nothing
    Dim Fill iType As Byte
    Dim Fill lHatchIndex As Integer
    Dim Fill iInForeColor As Integer
    Dim Fill iInBackColor As Integer
    Dim Fill sPattern As String = String.Empty
    Dim Fill zOrder As Integer
    Dim Fill lFlags As Integer
    Dim iClr As Integer
    Dim iWidth As Integer
    Dim iStyle As Integer
    Dim iLayer As Integer
    Dim PolylineType As Avax.AvaxPolylineType
    ReDim xV(5)
    ReDim yV(5)
    ReDim zV(5)
    xV(1) = 2
    yV(1) = 2
    xV(2) = 2
    yV(2) = 12
    xV(3) = 12
    yV(3) = 12
    xV(4) = 12
    yV(4) = 2
    xV(5) = 10
    yV(5) = -1
    Fill.iType = Avax.AvaxFillType.Pattern_f
    Fill.lHatchIndex = 2
    iClr = 1
    iWidth = 2
    Fill.iInBackColor = 4
    hShape = cAvax1.Add_Shape(xV, yV, zV, vItemProperties, Fill.iType, Fill.lHatchIndex,
    Fill.iInForeColor, Fill.iInBackColor, Fill.sPattern, Fill.zOrder, Fill.lFlags, iClr, iWidth, iStyle,
    iLayer, PolylineType)
```

```
End Sub
Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
    Button1.Click
    Dim Fill_iType As Byte
    Dim Fill_lHatchIndex As Integer
    Dim Fill_iInForeColor As Integer
    Dim Fill_iInBackColor As Integer
    Dim Fill_sPattern As String = String.Empty
    Dim Fill_zOrder As Integer
    Dim Fill_lFlags As Integer
    Dim Answer as Boolean
    Answer = CAvax1.GetShapeFill(hShape, Fill_iType, Fill_lHatchIndex, Fill_iInForeColor,
    Fill_iInBackColor, Fill_sPattern, Fill_zOrder, Fill_lFlags)
    MsgBox("Fill_iType=" & Fill_iType & Environment.NewLine & "Fill_lHatchIndex=" & Fill_lHatchIndex &
    Environment.NewLine & "Fill_iInBackColor=" & Fill_iInBackColor)
End Sub
```

GetShapeParameters

Purpose

Reads the editable dimensions of an existing native shape. Use the returned names to decide which parameter fields to show in an application.

VB.NET declaration

```
Function GetShapeParameters(ByVal handle As Integer) As Dictionary(Of String, Double)
```

Parameters

Parameter	Meaning
handle	Integer handle in the current drawing.

Return value and behavior

Returns an independent Dictionary(Of String, Double). Missing, unsupported or freely node-edited geometry returns an empty dictionary. Recognized native R2 rectangles, circles, ellipses and regular polygons can also expose their dimensions. Editing the returned dictionary does not change the drawing. This read-only query is available while the control is disabled; an unstarted or ended control returns an empty dictionary.

Parameter names

Shape	Keys
Rectangle	ShapeWidth, ShapeHeight, CornerRadiusX, CornerRadiusY, CornerRadius. CornerRadius is NaN when the two radii differ.
Circle / ellipse	Radius for equal orthogonal axes; otherwise RadiusX and RadiusY.
Polygon	Radius or RadiusX/RadiusY, plus Vertices.
Arc / pie	Radius or RadiusX/RadiusY, plus StartAngle and SweepAngle.

VB.NET example

Pass a started control on its owning UI thread. This is a complete standalone module.

```
Option Strict On
Imports Avax

Public Module EditableShapeParametersExample
    Public Function ReadWidth(cad As cAvax, id As Integer) As Double?
        Dim values = cad.GetShapeParameters(id)
        Dim width As Double
        If values.TryGetValue("ShapeWidth", width) Then Return width
        Return Nothing
    End Function
End Module
```

Related options and reference

GetShapePicturePixels

cAvax method

Purpose

Use this method if you want to get an array of RGB colors from the pixels of a picture object

VB.NET declaration

```
Function GetShapePicturePixels(ByRef avHandle As Integer, ByRef lWidth As Integer, ByRef lHeight As
    Integer, ByRef RGBPixels As Integer(,)) As Boolean
```

Parameters

Name	Type and passing	Meaning
avHandle	ByRef avHandle As Integer	Handle of a picture shape, or a block whose first visible readable picture will be used. The input handle is not modified.
lWidth	ByRef lWidth As Integer	Receives the projected picture-frame width in pixels. It is also the first array dimension's upper bound.
lHeight	ByRef lHeight As Integer	Receives the projected picture-frame height in pixels. It is also the second array dimension's upper bound.
RGBPixels	ByRef RGBPixels As Integer(,)	Receives an Integer(lWidth, lHeight) array. Read colors at RGBPixels(x,y) for x=1..lWidth and y=1..lHeight. Index zero is reserved in both dimensions. The previous array remains unchanged on failure.

Use the declaration for the overload you call. ByRef arguments can receive outputs; declare variables and arrays with the exact indicated type. Optional defaults are shown in the declaration.

Return value

Boolean - True on success, false on error.

Remarks

Reads a picture shape into a two-dimensional Integer array of RGB colors. For a block handle, the first readable visible picture is used in child drawing order, including nested blocks. A hidden layer suppresses its objects and descendants; locking a layer or marking it nonprinting does not suppress this screen query. The dimensions follow the picture frame at the current zoom and ScreenXFactor_p; they are not necessarily the original image file dimensions. A rotated rectangular picture is sampled in its own frame before rotation. On success the array is allocated as RGBPixels(lWidth, lHeight), with real pixels at indices 1 through lWidth and 1 through lHeight. Row zero and column zero are reserved and contain zero. Pixel (1,1) is the top-left sample for either origin setting. Each value is a COLORREF: red + (green << 8) + (blue << 16), without an alpha channel. Transparent samples are combined with the control background color. WMF mapping mode is taken from the stored picture. Fixed metric, English and TWIPS mapping modes retain their native upward-positive Y direction. EMF vector content is rasterized at the requested frame size. Animated pictures use the current display frame when AnimationOn_p is True and the first frame otherwise. The control must be started and enabled. Invalid handles, other object types, unsupported picture data, zero-sized projected frames or dimensions over 8000 pixels return False and preserve all output values and the existing array reference. Editing locks and PauseAvax do not prevent the read. Returned arrays are independent copies; the query does not edit geometry, change the active command or create an Undo entry.

Example

Setup

Add an Avax.cAvax Windows Forms control named `cAvax1`. Paste the procedures into the form class. Example setup and supporting files explains references, initialization and file paths.

Add these Windows Forms controls or components to the form. Fields referenced by Handles clauses must be declared WithEvents, as the Visual Studio designer normally does.

Name	Type
Button1	Button

VB.NET example

```
Dim avHandle As Integer
Private Sub Form1_Load(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs) Handles
    MyBase.Load
    Call CAvax1.StartAvax()
    Call AddPicture()
End Sub
Private Sub Form1_FormClosed(ByVal eventSender As System.Object, ByVal eventArgs As
    System.Windows.Forms.FormClosedEventArgs) Handles Me.FormClosed
    Call cAvax1.EndAvax()
End Sub
Sub AddPicture()
    Dim sPictureFiles() As String = Nothing
    Dim xMin As Double
    Dim yMin As Double
    Dim xMax As Double
    Dim yMax As Double
    ReDim sPictureFiles(1)
    sPictureFiles(1) = My.Application.Info.DirectoryPath & "\..\Clouds1.bmp"
    xMin = 0
    yMin = 0
    xMax = 5
    yMax = 5
    avHandle = CAvax1.Add_Picture(sPictureFiles, xMin, yMin, xMax, yMax)
End Sub
Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
    Button1.Click
    Dim lWidth As Integer
    Dim lHeight As Integer
    Dim RGBPixels(,) As Integer = Nothing
    Dim Answer As Boolean
    Dim pixelsCount As Integer
    Answer = CAvax1.GetShapePicturePixels(avHandle, lWidth, lHeight, RGBPixels)
    If Answer = True Then
        pixelsCount = (RGBPixels.GetLength(0) - 1) * (RGBPixels.GetLength(1) - 1)
        MsgBox("Width=" & lWidth & Environment.NewLine & "Height=" & lHeight & Environment.NewLine &
    "Pixels count=" & pixelsCount)
    End If
End Sub
```

GetShapeText

cAvax method

Purpose

Returns the text of a ShapeText object with a specific handle

VB.NET declarations

```
Function GetShapeText(ByRef hTextShape As Integer) As String

Function GetShapeText(ByRef hTextShape As Integer, ByRef sFontName As String, ByRef fBold As Boolean
    ByRef fItalic As Boolean, ByRef fUnderline As Boolean, ByRef fStrikeOut As Boolean, ByRef lCharSet As
    Integer, ByRef TxtHeight As Double, ByRef TxtWidth As Double) As String

Function GetShapeText(ByRef hTextShape As Integer, ByRef sFontName As String, ByRef fBold As Boolean,
    ByRef fItalic As Boolean, ByRef fUnderline As Boolean, ByRef fStrikeOut As Boolean, ByRef lCharSet As
    Integer) As String
```

Parameters

Name	Type and passing	Meaning
hTextShape	ByRef hTextShape As Integer	Handle of a text-filled Shape_i object.
sFontName	ByRef sFontName As String	Embedded font name.
fBold	ByRef fBold As Boolean	Stored bold flag; False overwrites any previous True value.
fItalic	ByRef fItalic As Boolean	Stored italic flag.
fUnderline	ByRef fUnderline As Boolean	Stored underline flag.
fStrikeOut	ByRef fStrikeOut As Boolean	Stored strikeout flag.
lCharSet	ByRef lCharSet As Integer	Integer embedded character-set value.
TxtHeight	ByRef TxtHeight As Double	Double intrinsic font height.
TxtWidth	ByRef TxtWidth As Double	Double intrinsic text width, not world bounding-box width.

Use the declaration for the overload you call. ByRef arguments can receive outputs; declare variables and arrays with the exact indicated type. Optional defaults are shown in the declaration.

Return value

String - Is the text of the object

Remarks

Returns the exact stored text of a text-filled Shape_i (fill type 8). The short, original CHM seven-argument and full nine-argument overloads are available. The output arguments receive the embedded font name and charset, and all four style flags, including False values. TxtHeight and TxtWidth are intrinsic Double text-layout metrics: rotation does not replace them with axis-aligned bounding-box dimensions; scaling follows the transformed text axes. Embedded font information remains independent of later font-table changes. Ended or disabled controls and missing or wrong-type handles return Nothing without changing any ByRef output. Reading does not change the drawing, selection, current command or history.

Numeric changes from v2

Update the following variables, arrays or event-data receivers when porting the corresponding v2 call. The declarations above show the current overloads.

Parameter or value	v2 type	v3 type
TxtHeight	Single	Double

Parameter or value	v2 type	v3 type
TxtWidth	Single	Double

For ByRef parameters, use a variable with the exact v3 type. Preserve array dimensions and the documented indexing. Moving from v2 to v3 explains conversion and legacy file-storage rules.

Example

Setup

Add an Avax.cAvax Windows Forms control named `cAvax1`. Paste the procedures into the form class. Example setup and supporting files explains references, initialization and file paths.

Add these Windows Forms controls or components to the form. Fields referenced by Handles clauses must be declared WithEvents, as the Visual Studio designer normally does.

Name	Type
Command1	Button

VB.NET example

```
Dim hTextShape As Integer
Private Sub Form1_Load(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs) Handles
    MyBase.Load
    Command1.Text = "GetShapeText"
    cAvax1.StartAvax()
    Call AddShapeText()
End Sub
Private Sub Form1_FormClosed(ByVal eventSender As System.Object, ByVal eventArgs As
    System.Windows.Forms.FormClosedEventArgs) Handles Me.FormClosed
    cAvax1.EndAvax()
End Sub
Sub AddShapeText()
    Dim sText As String = String.Empty
    Dim iClr As Integer
    Dim fBold As Boolean
    Dim fItalic As Boolean
    sText = "Sample Text"
    fItalic = True
    fBold = True
    iClr = 1
    hTextShape = cAvax1.Add_ShapeText(sText, 0, 0, 6, Nothing, 0, -1, iClr, -1, -1, -1, fBold, fItalic,
    False, False, -1, 0, False)
End Sub
Private Sub Command1_Click(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs)
    Handles Command1.Click
    Dim sFontName As String = String.Empty
    Dim fBold As Boolean
    Dim fItalic As Boolean
    Dim fUnderline As Boolean
    Dim fStrikeOut As Boolean
    Dim lCharSet As Integer
    Dim sMsg As String = String.Empty
    sMsg = cAvax1.GetShapeText(hTextShape, sFontName, fBold, fItalic, fUnderline, fStrikeOut, lCharSet)
    sMsg = sMsg & vbCrLf & vbCrLf
    sMsg = sMsg & "FontName=" & sFontName & vbCrLf & "Bold=" & fBold & vbCrLf
    sMsg = sMsg & "Italic=" & fItalic & vbCrLf & "Underline=" & fUnderline & vbCrLf
    sMsg = sMsg & "StrikeOut=" & fStrikeOut & vbCrLf & "CharSet=" & lCharSet
    MsgBox(sMsg, MsgBoxStyle.Information, "GetShapeText")
End Sub
```

GetShapezOrder

cAvax method

Purpose

Returns the z order of a shape object with a specific handle

VB.NET declaration

```
Function GetShapezOrder(ByRef hShape As Integer) As Integer
```

Parameters

Name	Type and passing	Meaning
hShape	ByRef hShape As Integer	Input handle of an existing shape. Returns its signed Integer stacking order; other item types or an unavailable handle/control return zero. The handle is unchanged.

Use the declaration for the overload you call. ByRef arguments can receive outputs; declare variables and arrays with the exact indicated type. Optional defaults are shown in the declaration.

Return value

Integer - Is the z order of the shape

Remarks

Returns the stored stacking order of a Shape_i. Other existing item types return zero, as in R2. Requires a started, enabled control. Missing handles or an unavailable control return zero. hShape is unchanged, and the getter does not change the drawing or undo history.

zOrder is a signed Integer. When ShapeZorderOn_p is True, higher values draw in front of lower values; equal values retain handle order. When the property is False, rendering uses handle order. Use GetShapeFill when you need a Boolean success result as well as the order value.

Example

Setup

Add an Avax.cAvax Windows Forms control named `cAvax1`. Paste the procedures into the form class. Example setup and supporting files explains references, initialization and file paths.

Add these Windows Forms controls or components to the form. Fields referenced by Handles clauses must be declared WithEvents, as the Visual Studio designer normally does.

Name	Type
Command1	Button

VB.NET example

```
Dim hShape As Integer
Private Sub Form1_Load(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs) Handles
    MyBase.Load
    Command1.Text = "GetShapezOrder"
    cAvax1.StartAvax()
    Call AddShape()
End Sub
Private Sub Form1_FormClosed(ByVal eventSender As System.Object, ByVal eventArgs As
    System.Windows.Forms.FormClosedEventArgs) Handles Me.FormClosed
    cAvax1.EndAvax()
End Sub
Private Sub Command1_Click(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs)
```

```
    Handles Command1.Click
    MsgBox("zOrder=" & cAvax1.GetShapezOrder(hShape), MsgBoxStyle.Information, "GetShapezOrder")
End Sub
Sub AddShape()
    Dim xV() As Double = Nothing
    Dim yV() As Double = Nothing
    Dim zV() As Double = Nothing
    Dim vItemProperties As Object = Nothing
    Dim Fill iType As Byte
    Dim Fill lHatchIndex As Integer
    Dim Fill iInForeColor As Integer
    Dim Fill iInBackColor As Integer
    Dim Fill sPattern As String = String.Empty
    Dim Fill zOrder As Integer
    Dim Fill lFlags As Integer
    Dim iClr As Integer
    Dim iWidth As Integer
    Dim iStyle As Integer
    Dim iLayer As Integer
    Dim PolylineType As Avax.AvaxPolylineType
    ReDim xV(5)
    ReDim yV(5)
    ReDim zV(5)
    xV(1) = 2
    yV(1) = 2
    xV(2) = 2
    yV(2) = 12
    xV(3) = 12
    yV(3) = 12
    xV(4) = 12
    yV(4) = 2
    xV(5) = 10
    yV(5) = -1
    Fill iType = Avax.AvaxFillType.Pattern_f
    Fill lHatchIndex = 2
    iClr = 1
    iWidth = 2
    Fill iInBackColor = 4
    '---
    Fill zOrder = 15
    '---
    hShape = cAvax1.Add_Shape(xV, yV, zV, vItemProperties, Fill iType, Fill lHatchIndex,
    Fill iInForeColor, Fill iInBackColor, Fill sPattern, Fill zOrder, Fill lFlags, iClr, iWidth, iStyle,
    iLayer, PolylineType)
End Sub
```

GetStrokeWidth

Purpose

Reads the precise pen width stored on an item, including fractional widths set with ChangeStrokeWidth. Use it to display or retain an existing width without reducing it to an Integer.

VB.NET declaration

```
Function GetStrokeWidth(ByVal handle As Integer) As Double
```

Parameters

Parameter	Meaning
handle	Integer handle of an existing item in the current drawing.

Return value and behavior

Returns the precise Double value. Zero denotes an explicitly disabled drawing pen; it also represents an ordinary shape whose outline is disabled. For an item without a precise width, the result is its legacy Integer pen width, with a minimum of one. A group reports its own stored attribute, rather than a common value calculated from its children.

A missing handle or an unstarted/ended control returns Double.NaN. Test this result with Double.IsNaN. Reading is allowed while the control is disabled and does not change the drawing, selection or history. Native .avax saving retains fractional values. Borders of embedded Windows controls follow their control-specific border settings.

VB.NET example

Pass a started control and an item handle on the control's owning UI thread.

```
Option Strict On
Imports Avax
Imports System.Globalization

Public Module StrokeWidthReadExample
    Public Function ReadForDisplay(cad As cAvax, id As Integer) As String
        Dim width As Double = cad.GetStrokeWidth(id)
        If Double.IsNaN(width) Then Return "No item"
        Return width.ToString("0.###", CultureInfo.InvariantCulture)
    End Function
End Module
```

Related options and reference

GetText

cAvax method

Purpose

Returns the text of a text object with a specific handle

See DoAvaxAction and public value structures for direct polyline closure, attribute/fill/text/stretch/status fields, widened numeric declarations and complete examples.

VB.NET declaration

```
Function GetText(ByRef hTextItem As Integer) As String
```

Parameters

Name	Type and passing	Meaning
hTextItem	ByRef hTextItem As Integer	Is the handle of the text object

Use the declaration for the overload you call. ByRef arguments can receive outputs; declare variables and arrays with the exact indicated type. Optional defaults are shown in the declaration.

Return value

String - Is the text of the text object

Example

Setup

Add an Avax.cAvax Windows Forms control named `cAvax1`. Paste the procedures into the form class. Example setup and supporting files explains references, initialization and file paths.

Add these Windows Forms controls or components to the form. Fields referenced by Handles clauses must be declared WithEvents, as the Visual Studio designer normally does.

Name	Type
Command1	Button

VB.NET example

```
Dim hTextItem As Integer
Private Sub Form1_Load(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs) Handles
    MyBase.Load
    Command1.Text = "GetText"
    cAvax1.StartAvax()
    hTextItem = CAvax1.Add_Text("AVAX ActiveX control", 0, 5, 0, Nothing, 6, 1, 0, -1, -1, -1, -1, False,
    False, False, False, -1)
End Sub
Private Sub Form1_FormClosed(ByVal eventSender As System.Object, ByVal eventArgs As
    System.Windows.Forms.FormClosedEventArgs) Handles Me.FormClosed
    cAvax1.EndAvax()
End Sub
Private Sub Command1_Click(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs)
    Handles Command1.Click
    MsgBox(cAvax1.GetText(hTextItem))
End Sub
```

GetTextAttributes

cAvax method

Purpose

Returns stored ordinary text; the full overload also supplies its resolved font, charset and four style flags.

VB.NET declarations

```
Function GetTextAttributes(ByRef hTextItem As Integer, ByRef sFontName As String, ByRef fBold As Boolean,
    ByRef fItalic As Boolean, ByRef fUnderline As Boolean, ByRef fStrikeOut As Boolean, ByRef lCharSet As
    Integer) As String

Function GetTextAttributes(ByRef hTextItem As Integer) As String
```

Parameters

Name	Type and passing	Meaning
hTextItem	ByRef hTextItem As Integer	The Avax handle of the text object
sFontName	ByRef sFontName As String	The font name of the text
fBold	ByRef fBold As Boolean	A Boolean expression that if it is True then the text is bold
fItalic	ByRef fItalic As Boolean	A Boolean expression that if it is True then the text is italic
fUnderline	ByRef fUnderline As Boolean	A Boolean expression that if it is True then new text is underlined
fStrikeOut	ByRef fStrikeOut As Boolean	A Boolean expression that if it is True then new text is strikethrough
lCharSet	ByRef lCharSet As Integer	Is the character set number of the text

Use the declaration for the overload you call. ByRef arguments can receive outputs; declare variables and arrays with the exact indicated type. Optional defaults are shown in the declaration.

Return value

String - The text object's text

Remarks

The handle must identify an ordinary Text_i object. The short overload returns its stored text. The full overload also supplies the resolved font name and charset, and stored bold, italic, underline and strikeout flags. False output flags replace earlier True values. Positive font numbers follow the current font table; inherited or unavailable entries use FontName and FontCharset. An independently named imported text object retains its own font binding.

Use GetShapeText for text-filled shapes and the document APIs for documents. A started, enabled control is required. Missing or wrong-type handles return Nothing without assigning ByRef outputs. The method does not edit the drawing or command state.

Example

Setup

Add an Avax.cAvax Windows Forms control named `cAvax1`. Paste the procedures into the form class. Example setup and supporting files explains references, initialization and file paths.

Add these Windows Forms controls or components to the form. Fields referenced by Handles clauses must be declared WithEvents, as the Visual Studio designer normally does.

Name	Type
Command1	Button

VB.NET example

```
Dim hTextItem As Integer
Private Sub Form1_Load(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs) Handles
    MyBase.Load
    cAvax1.StartAvax()
    hTextItem = CAvax1.Add_Text("My Text", 0, 0, 0, Nothing, 5, 1, 0, -1, -1, -1, -1, True, False, False,
    False, -1)
End Sub
Private Sub Form1_FormClosed(ByVal eventSender As System.Object, ByVal eventArgs As
    System.Windows.Forms.FormClosedEventArgs) Handles Me.FormClosed
    cAvax1.EndAvax()
End Sub
Private Sub Command1_Click(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs)
    Handles Command1.Click
    Dim sFontName As String = String.Empty
    Dim fBold As Boolean
    Dim fItalic As Boolean
    Dim fUnderline As Boolean
    Dim fStrikeOut As Boolean
    Dim lCharSet As Integer
    Dim Answer As String = String.Empty
    Answer = cAvax1.GetTextAttributes(hTextItem, sFontName, fBold, fItalic, fUnderline, fStrikeOut,
    lCharSet)
    MsgBox("Text = " & Answer & vbCrLf & "Bold =" & Str(fBold))
End Sub
```

GetTextFontNumber

cAvax method

Purpose

Returns the font number of a text object with a specific handle

VB.NET declaration

```
Function GetTextFontNumber(ByRef hTextItem As Integer) As Integer
```

Parameters

Name	Type and passing	Meaning
hTextItem	ByRef hTextItem As Integer	Is the handle of the object

Use the declaration for the overload you call. ByRef arguments can receive outputs; declare variables and arrays with the exact indicated type. Optional defaults are shown in the declaration.

Return value

Integer - Is the font number of the text object

Numeric changes from v2

Update the following variables, arrays or event-data receivers when porting the corresponding v2 call. The declarations above show the current overloads.

Parameter or value	v2 type	v3 type
Return value	Short	Integer

For ByRef parameters, use a variable with the exact v3 type. Preserve array dimensions and the documented indexing. Moving from v2 to v3 explains conversion and legacy file-storage rules.

Example

Setup

Add an Avax.cAvax Windows Forms control named `cAvax1`. Paste the procedures into the form class. Example setup and supporting files explains references, initialization and file paths.

Add these Windows Forms controls or components to the form. Fields referenced by Handles clauses must be declared WithEvents, as the Visual Studio designer normally does.

Name	Type
Command1	Button

VB.NET example

```
Dim hTextItem As Integer
Private Sub Form1_Load(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs) Handles
    MyBase.Load
    Dim iFontNumber As Integer
    Command1.Text = "GetTextFontNumber"
    cAvax1.StartAvax()
    iFontNumber = 1
    hTextItem = CAvax1.Add_Text("Avax ActiveX", 0, 0, 0, Nothing, 10.0!, iFontNumber, 0, -1, -1, -1, -1,
    False, False, False, False, -1)
End Sub
Private Sub Form1_FormClosed(ByVal eventSender As System.Object, ByVal eventArgs As
    System.Windows.Forms.FormClosedEventArgs) Handles Me.FormClosed
```

```
    cAvax1.EndAvax()
End Sub
Private Sub Command1_Click(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs)
    Handles Command1.Click
    MsgBox(cAvax1.GetTextFontNumber(hTextItem), MsgBoxStyle.Information, "GetTextFontNumber")
End Sub
```

GetTextGeometry

cAvax method

Purpose

Use this method if you want to get the geometry of a text object

VB.NET declaration

```
Function GetTextGeometry(ByRef avHandle As Integer, ByRef x As Double, ByRef y As Double, ByRef dWidth As
    Double, ByRef dHeight As Double, ByRef dAngle As Double, ByRef lTextLines As Integer) As Boolean
```

Parameters

Name	Type and passing	Meaning
avHandle	ByRef avHandle As Integer	Handle of an ordinary Text_i object.
x	ByRef x As Double	Double insertion X.
y	ByRef y As Double	Double insertion Y.
dWidth	ByRef dWidth As Double	Double intrinsic text width.
dHeight	ByRef dHeight As Double	Double Double-line font height, not multiline or rotated bounding-box height.
dAngle	ByRef dAngle As Double	Stored Double text angle in degrees.
lTextLines	ByRef lTextLines As Integer	Integer logical line count. CRLF counts once; CR and LF are accepted. Empty text has one line.

Use the declaration for the overload you call. ByRef arguments can receive outputs; declare variables and arrays with the exact indicated type. Optional defaults are shown in the declaration.

Return value

Boolean - True on success, false on error.

Numeric changes from v2

Update the following variables, arrays or event-data receivers when porting the corresponding v2 call. The declarations above show the current overloads.

Parameter or value	v2 type	v3 type
x	Single	Double
y	Single	Double
dWidth	Single	Double
dHeight	Single	Double
dAngle	Single	Double

For ByRef parameters, use a variable with the exact v3 type. Preserve array dimensions and the documented indexing. Moving from v2 to v3 explains conversion and legacy file-storage rules.

Example

Setup

Add an Avax.cAvax Windows Forms control named `cAvax1`. Paste the procedures into the form class. Example setup and supporting files explains references, initialization and file paths.

Add these Windows Forms controls or components to the form. Fields referenced by Handles clauses must be declared WithEvents, as the Visual Studio designer normally does.

Name	Type
Button1	Button

VB.NET example

```
Dim avHandle As Integer
Private Sub Form1_Load(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs) Handles
    MyBase.Load
    Call CAvax1.StartAvax()
    Call AddText()
End Sub
Private Sub Form1_FormClosed(ByVal eventSender As System.Object, ByVal eventArgs As
    System.Windows.Forms.FormClosedEventArgs) Handles Me.FormClosed
    Call cAvax1.EndAvax()
End Sub
Private Sub AddText()
    Dim sText As String = String.Empty
    Dim x As Double
    Dim y As Double
    Dim z As Double
    Dim vItemProperties As Object = Nothing
    Dim sScale As Double
    Dim iFontNumber As Integer
    Dim Angle As Double
    Dim iClr As Integer
    Dim iWidth As Integer
    Dim iStyle As Integer
    Dim iLayer As Integer
    Dim fBold As Boolean
    Dim fItalic As Boolean
    Dim fUnderline As Boolean
    Dim fStrikeOut As Boolean
    Dim lCharSet As Integer
    'Definition of the text
    sText = "Hello !!!"
    'The scale
    sScale = 10
    'Start point of the text
    x = 1 : y = 4 : z = 0
    'Angle of the text (degrees)
    Angle = 30
    iClr = 6
    fBold = True
    fUnderline = True
    avHandle = CAvax1.Add_Text(sText, x, y, z, vItemProperties, sScale, iFontNumber, Angle, iClr, iWidth,
    iStyle, iLayer, fBold, fItalic, fUnderline, fStrikeOut, lCharSet)
End Sub
Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
    Button1.Click
    Dim x As Double
    Dim y As Double
    Dim dWidth As Double
    Dim dHeight As Double
    Dim dAngle As Double
    Dim lTextLines As Integer
    Dim Answer as Boolean
    Answer = CAvax1.GetTextGeometry(avHandle, x, y, dWidth, dHeight, dAngle, lTextLines)
    MsgBox("x=" & x & Environment.NewLine & "y=" & y & Environment.NewLine & "Width=" & dWidth &
    Environment.NewLine & "Height=" & dHeight & Environment.NewLine & "Angle=" & dAngle & Environment.NewLine
    & "TextLines=" & lTextLines)
End Sub
```

GetTextStyle

Purpose

Read an independent copy of a text shape's character appearance and spacing.

VB.NET declaration

```
Function GetTextStyle(ByVal handle As Integer) As AvaxTextStyle
```

Parameters

handle: a supported text-shape handle.

Return value

A detached style, or Nothing when the handle is not a supported text shape. Changing the returned object does not change the drawing until SetTextStyle or SetTextStyles succeeds.

Example

```
Option Strict On
Public Module Help_TextStyle_2
    Public Sub Run(cad As Avax.cAvax, handle As Integer)
        Dim style As Avax.AvaxTextStyle = cad.GetTextStyle(handle)
        If style IsNot Nothing Then
            style.Bold = True
            cad.SetTextStyle(handle, style)
        End If
    End Sub
End Module
```

Related options and reference

GetTextVectors

cAvax method

Purpose

Converts a string and font settings into glyph-outline line segments, without adding drawing objects.

VB.NET declaration

```
Function GetTextVectors(ByRef sText As String, ByRef sFontName As String, ByRef TextScale As Double, ByRef
    fBold As Boolean, ByRef fItalic As Boolean, ByRef x1 As Double(), ByRef y1 As Double(), ByRef x2 As
    Double(), ByRef y2 As Double()) As Integer
```

Parameters

Name	Type and passing	Meaning
sText	ByRef sText As String	Unicode text. Original ByRef input is unchanged. CR/LF line breaks and canonical combining accents are supported; invalid UTF-16 fails without replacing arrays.
sFontName	ByRef sFontName As String	Font family. Empty/whitespace selects current FontName; the original input string remains unchanged. Unavailable fonts use Windows font substitution.
TextScale	ByRef TextScale As Double	Double multiplier for TextInitHeight_p using R2 outline sizing and ascent normalization. It is not an em height by itself. Nonpositive finite values return 0; nonfinite values report an error.
fBold	ByRef fBold As Boolean	Explicit bold flag for the requested outline font.
fItalic	ByRef fItalic As Boolean	Explicit italic flag for the requested outline font.
x1	ByRef x1 As Double()	Output Double() start X coordinates at indices 1..return value; index 0 is reserved. Successful arrays are newly allocated.
y1	ByRef y1 As Double()	Output Double() start Y coordinates. First baseline is Y=0; positive Y is upward, matching R2 drawing coordinates.
x2	ByRef x2 As Double()	Output Double() end X coordinates, aligned with x1/y1/y2.
y2	ByRef y2 As Double()	Output Double() end Y coordinates. Invalid, empty-input or stopped calls preserve the previous caller arrays.

Use the declaration for the overload you call. ByRef arguments can receive outputs; declare variables and arrays with the exact indicated type. Optional defaults are shown in the declaration.

Return value

Integer - The number of returned outline segments, which is also the upper bound of each one-based output array. A result of 0 means no segments were returned.

Remarks

The four independent Double() outputs contain segment start and end XY coordinates at indices 1 through the Integer return value; index 0 is reserved. The first baseline starts at (0,0), positive Y is upward and subsequent lines run below the first. Separate glyph contours remain separate. TextScale multiplies TextInitHeight_p by 0.6; font-ascent normalization sets glyph size. PltTextQuality_p selects outline subdivision.

A blank font name selects FontName; explicit bold/italic arguments choose styles. Windows can substitute an unavailable font. The control must be started, but the query remains available when Enabled=False. It creates no drawing objects, Undo operation or Change/NewUserItems notification.

Empty input or a nonpositive finite scale returns 0 without replacing outputs. Whitespace has no outlines and returns four arrays containing only index 0. Nonfinite or unrepresentable geometry and invalid Unicode report an error and preserve the caller arrays.

Numeric changes from v2

Update the following variables, arrays or event-data receivers when porting the corresponding v2 call. The declarations above show the current overloads.

Parameter or value	v2 type	v3 type
TextScale	Single	Double
x1	Single()	Double()
y1	Single()	Double()
x2	Single()	Double()
y2	Single()	Double()

For ByRef parameters, use a variable with the exact v3 type. Preserve array dimensions and the documented indexing. Moving from v2 to v3 explains conversion and legacy file-storage rules.

Example

Setup

Add an Avax.cAvax Windows Forms control named `cAvax1`. Paste the procedures into the form class. Example setup and supporting files explains references, initialization and file paths.

Add these Windows Forms controls or components to the form. Fields referenced by Handles clauses must be declared WithEvents, as the Visual Studio designer normally does.

Name	Type
Button1	Button

VB.NET example

```
Private Sub Form1_Load(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs) Handles
    MyBase.Load
    Call cAvax1.StartAvax()
End Sub
Private Sub Form1_FormClosed(ByVal eventSender As System.Object, ByVal eventArgs As
    System.Windows.Forms.FormClosedEventArgs) Handles Me.FormClosed
    Call cAvax1.EndAvax()
End Sub
Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
    Button1.Click
    Dim sText As String = String.Empty
    Dim sFontName As String = String.Empty
    Dim TextScale As Double
    Dim fBold As Boolean
    Dim fItalic As Boolean
    Dim x1() As Double = Nothing
    Dim y1() As Double = Nothing
    Dim z1() As Double = Nothing
    Dim x2() As Double = Nothing
    Dim y2() As Double = Nothing
    Dim z2() As Double = Nothing
    Dim Answer As Integer
    sText = "Hello!"
    sFontName = "Arial"
    TextScale = 2
    fBold = True
    fItalic = True
    Answer = CAvax1.GetTextVectors(sText, sFontName, TextScale, fBold, fItalic, x1, y1, x2, y2)
    ReDim z1(x1.Length - 1), z2(x2.Length - 1)
    For i As Integer = 1 To x1.Length - 1
        CAvax1.Add_Line(x1(i), y1(i), 0, x2(i), y2(i), 0)
        CAvax1.Command = Avax.AvaxCommand.AutoLimits_c
    Next
```

```
End Sub
```

GetWorkspacePageBounds

Purpose

Return full-sheet or printable-area bounds in drawing coordinates, in either mode.

VB.NET declaration

```
Sub GetWorkspacePageBounds(ByRef xMin As Double, ByRef yMin As Double, ByRef xMax As Double, ByRef yMax As
     Double, Optional ByVal printableArea As Boolean = False)
```

Parameters

Parameter	Meaning
xMin	Output minimum X (Double variable).
yMin	Output minimum Y.
xMax	Output maximum X.
yMax	Output maximum Y.
printableArea	False (default): full sheet. True: inset by PageMarginMM.

Return value and behavior

No return value; writes the four ByRef outputs without changing the drawing.

Example

Call this procedure on the owning UI thread with a control already initialized by `StartAvax()`. Handles must belong to that control. File paths are supplied by the caller.

```
Option Strict On
Imports System
Imports Avax

Public Module Help_GetWorkspacePageBounds
    Public Sub Run(cad As cAvax)
        Dim xMin, yMin, xMax, yMax As Double
        cad.GetWorkspacePageBounds(xMin, yMin, xMax, yMax, printableArea:=True)
    End Sub
End Module
```

Related options and reference

InitProgramID

Purpose

Retains the product-initialization call used by existing AVAX applications. The v3 product identity and license settings are embedded and available automatically. Calling this method is optional and does not unlock the control, change a serial number or reset a drawing. It does not restore the product identity used by previous AVAX versions.

VB.NET declaration

```
Sub InitProgramID()
```

Parameters and return value

No parameters or return value. Repeated calls are safe. Initialize a drawing control with StartAvax; use Unlock for a process runtime serial number.

Example

```
Option Strict On
Imports Avax

Public Module Help_InitProgramID
    Public Function Initialize(cad As cAvax) As Boolean
        cad.InitProgramID()
        Return cad.StartAvax()
    End Function
End Module
```

Related options and reference

IsImageShape

Purpose

Checks whether a drawing handle identifies a stored still-bitmap shape that supports the image effect API.

Declaration

```
Function IsImageShape(ByVal handle As Integer) As Boolean
```

Parameters and result

Name	Meaning
handle	Integer handle of the drawing object to inspect.

Returns True for a supported bitmap shape with image data. Returns False for an unknown or unsupported object, a stored AVAX animation or embedded video, or an unstarted or disposed control. Reading remains available while the control is disabled or editing is locked. Hidden or frozen layer state does not prevent this query; applying effects requires an editable, visible target.

This is an object eligibility check. Image decoding takes place when rendering a preview or applying enabled effects.

VB.NET example

```
Option Strict On
Public Module IsImageShapeExample
    Public Function CanOfferImageEffects(cad As Avax.cAvax
                                         selectedHandle As Integer) As Boolean
        Return cad.IsImageShape(selectedHandle)
    End Function
End Module
```

Related options and reference

isSelected

cAvax method

Purpose

Informs if something is selected

See Selection commands and type filters for one-based handle arrays, point/window selection, modifier keys, type-mask values, and committed selection notifications.

VB.NET declaration

```
Function isSelected() As Boolean
```

Return value

Boolean - True if something is selected

Example

Setup

Add an Avax.cAvax Windows Forms control named `cAvax1`. Paste the procedures into the form class. Example setup and supporting files explains references, initialization and file paths.

Add these Windows Forms controls or components to the form. Fields referenced by Handles clauses must be declared WithEvents, as the Visual Studio designer normally does.

Name	Type
Command1	Button

VB.NET example

```
Private Sub Form1_Load(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs) Handles
    MyBase.Load
    Command1.Text = "IsSelected"
    cAvax1.StartAvax()
    Call cAvax1.Add_Rectangle(0, 0, 12, 10)
End Sub
Private Sub Form1_FormClosed(ByVal eventSender As System.Object, ByVal eventArgs As
    System.Windows.Forms.FormClosedEventArgs) Handles Me.FormClosed
    cAvax1.EndAvax()
End Sub
Private Sub Command1_Click(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs)
    Handles Command1.Click
    If cAvax1.IsSelected() = False Then
        MsgBox("Nothing is selected", MsgBoxStyle.Critical, "IsSelected")
    Else
        MsgBox("Something is selected!!!", MsgBoxStyle.Information, "IsSelected")
    End If
End Sub
```

IsSelectedItem

cAvax method

Purpose

Returns the select status for an object associated with a specific handle

See Selection commands and type filters for one-based handle arrays, point/window selection, modifier keys, type-mask values, and committed selection notifications.

VB.NET declaration

```
Function IsSelectedItem(ByRef Handle As Integer) As Boolean
```

Parameters

Name	Type and passing	Meaning
Handle	ByRef Handle As Integer	Is the object's handle

Use the declaration for the overload you call. ByRef arguments can receive outputs; declare variables and arrays with the exact indicated type. Optional defaults are shown in the declaration.

Return value

Boolean - True if the object is selected

Example

Setup

Add an Avax.cAvax Windows Forms control named cAvax1. Paste the procedures into the form class. Example setup and supporting files explains references, initialization and file paths.

Add these Windows Forms controls or components to the form. Fields referenced by Handles clauses must be declared WithEvents, as the Visual Studio designer normally does.

Name	Type
Command1	Button

VB.NET example

```
Dim hRect As Integer
Private Sub Form1_Load(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs) Handles
    MyBase.Load
    Command1.Text = "IsSelectedItem"
    cAvax1.StartAvax()
    hRect = cAvax1.Add_Rectangle(0, 0, 12, 10)
End Sub
Private Sub Form1_FormClosed(ByVal eventSender As System.Object, ByVal eventArgs As
    System.Windows.Forms.FormClosedEventArgs) Handles Me.FormClosed
    cAvax1.EndAvax()
End Sub
Private Sub Command1_Click(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs)
    Handles Command1.Click
    If cAvax1.IsSelectedItem(hRect) = False Then
        MsgBox("Rectangle is not selected", MsgBoxStyle.Critical, "IsSelectedItem")
    Else
        MsgBox("Rectangle is selected!!!", MsgBoxStyle.Information, "IsSelectedItem")
    End If
End Sub
```

IsTextShape

Purpose

Determine whether a handle identifies an editable text shape in this control.

VB.NET declaration

```
Function IsTextShape(ByVal handle As Integer) As Boolean
```

Parameters

handle: an object handle belonging to this drawing.

Return value

True for a supported text shape; False for another object, an unknown handle or an unavailable control.

Example

```
Option Strict On
Public Module Help_TextStyle_1
    Public Sub Run(cad As Avax.cAvax, handle As Integer)
        Dim editable As Boolean = cad.IsTextShape(handle)
    End Sub
End Module
```

Related options and reference

ItemType

Returns the type of the object identified by a handle passed by value.

```
Public Function ItemType(ByVal AvaxHandle As Integer) As Avax.AvaxItemType
```

Parameter	Meaning
AvaxHandle	The object's handle in this control. Passing it does not change the caller's variable.

Related options and reference

LineStyleIndexByName

cAvax method

Purpose

This function returns the line type index based on a specific line type name

VB.NET declaration

```
Function LineStyleIndexByName(ByRef sLineStyleName As String) As Integer
```

Parameters

Name	Type and passing	Meaning
sLineStyleName	ByRef sLineStyleName As String	ByRef input name, preserved. Matching trims whitespace and ignores case; missing names return Integer zero.

Use the declaration for the overload you call. ByRef arguments can receive outputs; declare variables and arrays with the exact indicated type. Optional defaults are shown in the declaration.

Return value

Integer - Is the line type index number

Numeric changes from v2

Update the following variables, arrays or event-data receivers when porting the corresponding v2 call. The declarations above show the current overloads.

Parameter or value	v2 type	v3 type
Return value	Short	Integer

For ByRef parameters, use a variable with the exact v3 type. Preserve array dimensions and the documented indexing. Moving from v2 to v3 explains conversion and legacy file-storage rules.

Example

Setup

Add an Avax.cAvax Windows Forms control named `cAvax1`. Paste the procedures into the form class. Example setup and supporting files explains references, initialization and file paths.

Add these Windows Forms controls or components to the form. Fields referenced by Handles clauses must be declared WithEvents, as the Visual Studio designer normally does.

Name	Type
Command1	Button

VB.NET example

```
Dim sLineStyleName As String = String.Empty
Private Sub Form1_Load(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs) Handles
    MyBase.Load
    Command1.Text = "LineStyleIndexByName"
    cAvax1.StartAvax()
    sLineStyleName = "HIDDEN"
    Call CAvax1.Add_Line(0, 0, 0, 10, 10, 0, Nothing, -1, -1, CAvax1.LineStyleIndexByName(sLineStyleName),
     -1)
End Sub
Private Sub Form1_FormClosed(ByVal eventSender As System.Object, ByVal eventArgs As
    System.Windows.Forms.FormClosedEventArgs) Handles Me.FormClosed
```

```
    cAvax1.EndAvax()
End Sub
Private Sub Command1_Click(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs)
    Handles Command1.Click
    MsgBox(sLineStyleName & "=" & cAvax1.LineStyleIndexByName(sLineStyleName))
End Sub
```

Load_HatchPattern

Purpose

Load one named PAT definition into a detached options object.

VB.NET declaration

```
Function Load_HatchPattern(ByVal path As String, ByVal patternName As String, Optional ByVal options As
    AvaxHatchOptions = Nothing) As AvaxHatchOptions
```

Parameters

Parameter	Meaning
path	Path to an existing PAT file.
patternName	Name of the pattern inside that file.
options	Optional starting options; Nothing starts from Get_HatchDefaults().

Return value and behavior

New validated options with Kind=Pattern and embedded PatternLines; Nothing if the file, named definition or resulting options are invalid. Scale and spacing must be positive, numeric settings finite and transparency from 0 through 90 percent. Previous PatternLines are replaced, so an incomplete old embedded definition can be replaced by a valid PAT definition. The supplied options object is not modified. Does not create or edit a drawing object. Apply through Add_Hatch or Set_Hatch.

Example

Call this procedure on the owning UI thread with a control already initialized by `StartAvax()`. Handles must belong to that control. File paths are supplied by the caller.

```
Option Strict On
Imports System
Imports Avax

Public Module Help_Load_HatchPattern
    Public Sub Run(cad As cAvax, handle As Integer, patFile As String, patternName As String)
        Dim data = cad.Get_Hatch(handle)
        If data Is Nothing Then Throw New ArgumentException("Native hatch required.")
        Dim options = cad.Load_HatchPattern(patFile, patternName, data.Options)
        If options Is Nothing Then Throw New InvalidOperationException("PAT load failed.")
        options.AngleDeg = 0
        options.Scale = 2
        data.Options = options
        If Not cad.Set_Hatch(handle, data) Then Throw New InvalidOperationException("Pattern update
    failed.")
    End Sub
End Module
```

Related options and reference

LoadACADLineStyles

cAvax method

Purpose

Loads line types from an Autocad external .LIN file

VB.NET declaration

```
Function LoadACADLineStyles(ByRef sACADLinFile As String) As Boolean
```

Parameters

Name	Type and passing	Meaning
sACADLinFile	ByRef sACADLinFile As String	ByRef LIN input path, unchanged. SHX files are resolved beside it or in the configured shape directory. Full-table failure preserves the existing resources.

Use the declaration for the overload you call. ByRef arguments can receive outputs; declare variables and arrays with the exact indicated type. Optional defaults are shown in the declaration.

Return value

Boolean - True on succesfull load, false on fail.

Remarks

Reads an AutoCAD LIN file and replaces the control's complete line-style table. Definitions retain file order and occupy indices 1 through 255; index zero remains continuous. Existing objects retain their numeric style indices, so reordering the table changes which definition those objects use. The source filename is passed ByRef and remains unchanged. Blank lines and semicolon comment lines are ignored. A-aligned patterns are accepted case-insensitively. Positive lengths draw, negative lengths leave gaps and zero denotes a dot; a pattern requires a finite positive total repeat length. Numbers use decimal points and may use Double scientific notation. An A-only continuous definition is represented by a unit continuous pattern. Complex patterns may contain quoted text and supported S/R/A/U/X/Y parameters or SHX symbols. Shape files are resolved beside the LIN file, then in the options directory configured with DirsSet. AutoCAD-86 shapes 1.0/1.1 programs and their referenced subshapes are embedded into the returned table, so saved line styles and drawings do not require the original SHX files. Missing or malformed dependencies, invalid patterns and tables exceeding 255 entries return False without replacing the previous table. A successful change publishes Change after the whole table is committed; an identical reload is a no-op. This method creates no drawing objects and preserves accepted command input. Resource loading remains available before startup and while stopped or disabled; drawing notifications require an active enabled control and are suppressed by PauseAvaxEvents. To repair an old missing or damaged definition, reload its valid original LIN/SHX resources with the original table order, or query one definition with CreateACADLineStyleString and assign it to its original index using SetLineStyle. The loader does not guess missing symbol data or remap existing objects by name.

Example

Setup

Add an Avax.cAvax Windows Forms control named `cAvax1`. Paste the procedures into the form class. Example setup and supporting files explains references, initialization and file paths.

Add these Windows Forms controls or components to the form. Fields referenced by Handles clauses must be declared WithEvents, as the Visual Studio designer normally does.

Name	Type
Command1	Button

VB.NET example

```
Private Sub Form1_Load(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs) Handles
    MyBase.Load
    Command1.Text = "LoadACADLineStyles"
```

```
    cAvax1.StartAvax()
End Sub
Private Sub Form1_FormClosed(ByVal eventSender As System.Object, ByVal eventArgs As
    System.Windows.Forms.FormClosedEventArgs) Handles Me.FormClosed
    cAvax1.EndAvax()
End Sub
Private Sub Command1_Click(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs)
    Handles Command1.Click
    Dim sACADLinFile As String = String.Empty
    sACADLinFile = My.Application.Info.DirectoryPath & "\..\acad1.lin"
    MsgBox(cAvax1.LoadACADLineStyles(sACADLinFile), MsgBoxStyle.Information, "LoadACADLineStyles")
End Sub
```

LoadAvaxPropertiesFromXML

cAvax method

Purpose

Use this method if you want to load an XML document file that contains the properties of another avax control

See Drawing coordinate limits for Double coordinate properties, individual versus complete work-area assignment, and transactional settings validation.

VB.NET declaration

```
Function LoadAvaxPropertiesFromXML(ByRef XMLProperties As System.Xml.XmlDocument) As Boolean
```

Parameters

Name	Type and passing	Meaning
XMLProperties	ByRef XMLProperties As System.Xml.XmlDocument	System.Xml.XmlDocument passed ByRef. Supply an R2 AVAX-Properties document or supported early-R3 numeric-ID document. The loader never replaces or edits this DOM. Nothing, unsupported version and malformed recognized values return False before applying the batch.

Use the declaration for the overload you call. ByRef arguments can receive outputs; declare variables and arrays with the exact indicated type. Optional defaults are shown in the declaration.

Return value

Boolean - True on success, false on error.

Remarks

Loads supported settings from a System.Xml.XmlDocument passed ByRef,without replacing or editing the caller DOM. Returns False for invalid input before changing settings,fonts,cursor,limits,drawing revision or Undo history. Uses the R2 AVAX-Properties/General/Properties settings schema with General Version=1.0. This version is separate from native .avax drawing version3. AvaxProp numbers use invariant Double/Integer values and Boolean -1/0. FillType retains its seven attributes; MouseDrawing contains one Line element per one-based segment,with seven invariant Double values. Settings XML contains control settings,not drawing objects or resource tables. Standalone properties outside AvaxProp are not part of this legacy schema. Accepts named R2 settings and early v3 numeric-ID AvaxProperties settings. Missing values retain current settings; unknown property names are ignored. Recognized duplicate entries,unsupported schema versions,nonfinite numbers,invalid ranges,fonts or cursors reject the complete batch. Bounds and font fields are applied coherently. Binary FillType_p Pattern attributes marked PatternEncoding="base64-utf16le" are decoded without losing NUL or unmatched UTF-16 surrogate units. Unknown encoding markers,invalid base64 or incomplete code units reject the batch;unmarked R2 Pattern attributes remain readable. Settings can be loaded while stopped or disabled;a disposed control returns False. The complete commit is visible to FontChanged/CursorChanged handlers. A handler may install newer settings or start another operation;the returning older load does not overwrite it. Host event-handler exceptions propagate after the coherent commit. An active drawing command retains its accepted input unless an applied property deliberately changes its input behavior.

Example

Setup

Add an Avax.cAvax Windows Forms control named `cAvax1`. Paste the procedures into the form class. Example setup and supporting files explains references, initialization and file paths.

Add these Windows Forms controls or components to the form. Fields referenced by Handles clauses must be declared WithEvents, as the Visual Studio designer normally does.

Name	Type
Button1	Button

VB.NET example

```
Private Sub Form1_Load(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs) Handles
    MyBase.Load
    Call cAvax1.StartAvax()
End Sub
Private Sub Form1_FormClosed(ByVal eventSender As System.Object, ByVal eventArgs As
    System.Windows.Forms.FormClosedEventArgs) Handles Me.FormClosed
    Call cAvax1.EndAvax()
End Sub
Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
    Button1.Click
    Dim xmlDoc As Xml.XmlDocument
    Dim Answer As Boolean
    xmlDoc = New Xml.XmlDocument
    xmlDoc.Load(My.Application.Info.DirectoryPath & "\..\AvaxProperties.xml")
    Answer = CAvax1.LoadAvaxPropertiesFromXML(xmlDoc)
End Sub
```

LoadFonts

cAvax method

Purpose

Loads the font table from a external AVAX font file

VB.NET declaration

```
Function LoadFonts(ByRef sFontFile As String) As Boolean
```

Parameters

Name	Type and passing	Meaning
sFontFile	ByRef sFontFile As String	ByRef existing FNT path, unchanged by the call. Empty font-name lines are meaningful. Invalid input fails atomically.

Use the declaration for the overload you call. ByRef arguments can receive outputs; declare variables and arrays with the exact indicated type. Optional defaults are shown in the declaration.

Return value

Boolean - True on successfull load, false on fail.

Example

Setup

Add an Avax.cAvax Windows Forms control named `cAvax1`. Paste the procedures into the form class. Example setup and supporting files explains references, initialization and file paths.

Add these Windows Forms controls or components to the form. Fields referenced by Handles clauses must be declared WithEvents, as the Visual Studio designer normally does.

Name	Type
Command1	Button

VB.NET example

```
Private Sub Form1_Load(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs) Handles
    MyBase.Load
    Command1.Text = "LoadFonts"
    cAvax1.StartAvax()
End Sub
Private Sub Form1_FormClosed(ByVal eventSender As System.Object, ByVal eventArgs As
    System.Windows.Forms.FormClosedEventArgs) Handles Me.FormClosed
    cAvax1.EndAvax()
End Sub
Private Sub Command1_Click(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs)
    Handles Command1.Click
    Dim sFontFile As String = String.Empty
    sFontFile = My.Application.Info.DirectoryPath & "\..\myFonts.fnt"
    MsgBox(cAvax1.LoadFonts(sFontFile), MsgBoxStyle.Information, "LoadFonts")
End Sub
```

LoadLineStyles

cAvax method

Purpose

Loads line types from an external AVAX line type file

VB.NET declaration

```
Function LoadLineStyles(ByRef sAVAXLinFile As String) As Boolean
```

Parameters

Name	Type and passing	Meaning
sAVAXLinFile	ByRef sAVAXLinFile As String	ByRef input path, unchanged. Reads complete AVAX-LINESTYLES tables with meaningful empty rows; failure is atomic.

Use the declaration for the overload you call. ByRef arguments can receive outputs; declare variables and arrays with the exact indicated type. Optional defaults are shown in the declaration.

Return value

Boolean - True on succesfull load, false on fail.

Example

Setup

Add an Avax.cAvax Windows Forms control named `cAvax1`. Paste the procedures into the form class. Example setup and supporting files explains references, initialization and file paths.

Add these Windows Forms controls or components to the form. Fields referenced by Handles clauses must be declared WithEvents, as the Visual Studio designer normally does.

Name	Type
Command1	Button
Text1	TextBox

VB.NET example

```
Private Sub Form1_Load(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs) Handles
    MyBase.Load
    Command1.Text = "Save/Load LineStyles"
    cAvax1.StartAvax()
End Sub
Private Sub Form1_FormClosed(ByVal eventSender As System.Object, ByVal eventArgs As
    System.Windows.Forms.FormClosedEventArgs) Handles Me.FormClosed
    cAvax1.EndAvax()
End Sub
Private Sub Command1_Click(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs)
    Handles Command1.Click
    Dim sAVAXLinFile As String = String.Empty
    Dim iFr As Integer
    Dim sTmp As String = String.Empty
    sAVAXLinFile = My.Application.Info.DirectoryPath & "\..\AvaxLineStyle.txt"
    If cAvax1.SaveLineStyles(sAVAXLinFile) = True Then
        If cAvax1.LoadLineStyles(sAVAXLinFile) = True Then
            iFr = FreeFile
            FileOpen(iFr, sAVAXLinFile, OpenMode.Binary)
            sTmp = Space(FileLen(sAVAXLinFile))
            FileGet(iFr, sTmp)
```

```
            FileClose(iFr)
            Text1.Text = sTmp
        End If
    End If
End Sub
```

LoadPalette

cAvax method

Purpose

This function loads a color palette from an external .PAL file

VB.NET declaration

```
Function LoadPalette(ByRef sAVAXPalFile As String) As Boolean
```

Parameters

Name	Type and passing	Meaning
sAVAXPalFile	ByRef sAVAXPalFile As String	Input path, preserved ByRef. Reads AVAX-PAL, JASC-PAL and early v3 raw Integer-per-line files; failure leaves the current palette intact.

Use the declaration for the overload you call. ByRef arguments can receive outputs; declare variables and arrays with the exact indicated type. Optional defaults are shown in the declaration.

Return value

Boolean - True on succesfull load, false on fail.

Example

Setup

Add an Avax.cAvax Windows Forms control named `cAvax1`. Paste the procedures into the form class. Example setup and supporting files explains references, initialization and file paths.

Add these Windows Forms controls or components to the form. Fields referenced by Handles clauses must be declared WithEvents, as the Visual Studio designer normally does.

Name	Type
Command1	Button

VB.NET example

```
Private Sub Form1_Load(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs) Handles
    MyBase.Load
    Command1.Text = "LoadPalette"
    cAvax1.StartAvax()
End Sub
Private Sub Form1_FormClosed(ByVal eventSender As System.Object, ByVal eventArgs As
    System.Windows.Forms.FormClosedEventArgs) Handles Me.FormClosed
    cAvax1.EndAvax()
End Sub
Private Sub Command1_Click(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs)
    Handles Command1.Click
    Dim sPalFile As String = String.Empty
    sPalFile = My.Application.Info.DirectoryPath & "\..\myPalette.pal"
    MsgBox(cAvax1.LoadPalette(sPalFile), MsgBoxStyle.Information, "LoadPalette")
End Sub
```

LongToSingle

cAvax method

Purpose

Converts window pixel coordinates to drawing coordinates. The historical method name is retained; the drawing-coordinate outputs in v3 are Double.

VB.NET declaration

```
Function LongToSingle(ByRef xL As Integer, ByRef yL As Integer, ByRef xS As Double, ByRef yS As Double) As
    Boolean
```

Parameters

Name	Type and passing	Meaning
xL	ByRef xL As Integer	Is the x window coordinate
yL	ByRef yL As Integer	Is the y window coordinate
xS	ByRef xS As Double	Is the x drawing coordinate
yS	ByRef yS As Double	Is the y drawing coordinate

Use the declaration for the overload you call. ByRef arguments can receive outputs; declare variables and arrays with the exact indicated type. Optional defaults are shown in the declaration.

Return value

Boolean - True on success, false on error.

Numeric changes from v2

Update the following variables, arrays or event-data receivers when porting the corresponding v2 call. The declarations above show the current overloads.

Parameter or value	v2 type	v3 type
xS	Single	Double
yS	Single	Double

For ByRef parameters, use a variable with the exact v3 type. Preserve array dimensions and the documented indexing. Moving from v2 to v3 explains conversion and legacy file-storage rules.

Example

Setup

Add an Avax.cAvax Windows Forms control named `cAvax1`. Paste the procedures into the form class. Example setup and supporting files explains references, initialization and file paths.

Add these Windows Forms controls or components to the form. Fields referenced by Handles clauses must be declared WithEvents, as the Visual Studio designer normally does.

Name	Type
Label1	Label

VB.NET example

```
Private Sub CAvax1_OnMouseMove(ByVal Sender As Object, ByVal e As Avax.cAvax.MouseMoveEventArgs) Handles
    CAvax1.OnMouseMove
    Dim xL, yL As Integer
```

```
    Dim xS, yS As Double
    Dim sMsg As String = String.Empty
    xL = e.X
    yL = e.Y
    If CAvax1.LongToSingle(xL, yL, xS, yS) = True Then
        sMsg = "x =" & Str(xS) & vbCrLf
        sMsg = sMsg & "y =" & Str(yS)
    End If
    Label1.Text = sMsg
End Sub
Private Sub Form1_Load(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs) Handles
    MyBase.Load
    cAvax1.StartAvax()
End Sub
Private Sub Form1_FormClosed(ByVal eventSender As System.Object, ByVal eventArgs As
    System.Windows.Forms.FormClosedEventArgs) Handles Me.FormClosed
    cAvax1.EndAvax()
End Sub
```

MaxItems

cAvax method

Purpose

Returns with the total number of objects in the drawing

VB.NET declaration

```
Function MaxItems() As Integer
```

Return value

Integer - the total number of objects in the drawing

Example

Setup

Add an Avax.cAvax Windows Forms control named `cAvax1`. Paste the procedures into the form class. Example setup and supporting files explains references, initialization and file paths.

Add these Windows Forms controls or components to the form. Fields referenced by Handles clauses must be declared WithEvents, as the Visual Studio designer normally does.

Name	Type
Command1	Button
Command2	Button

VB.NET example

```
Private Sub Command2_Click(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs)
    Handles Command2.Click
    CAvax1.Command = Avax.AvaxCommand.Circle_c
End Sub
Private Sub Form1_Load(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs) Handles
    MyBase.Load
    cAvax1.StartAvax()
End Sub
Private Sub Form1_FormClosed(ByVal eventSender As System.Object, ByVal eventArgs As
    System.Windows.Forms.FormClosedEventArgs) Handles Me.FormClosed
    cAvax1.EndAvax()
End Sub
Private Sub Command1_Click(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs)
    Handles Command1.Click
    Dim Answer As Integer
    Answer = cAvax1.MaxItems()
    MsgBox(Str(Answer), , "Max Items !")
End Sub
```

MaxSelectedItems

cAvax method

Purpose

Use this method if you want to get the number of the selected objects

See Selection commands and type filters for one-based handle arrays, point/window selection, modifier keys, type-mask values, and committed selection notifications.

VB.NET declaration

```
Function MaxSelectedItems() As Integer
```

Return value

Integer - Is the number of the selected objects

Example

Setup

Add an Avax.cAvax Windows Forms control named `cAvax1`. Paste the procedures into the form class. Example setup and supporting files explains references, initialization and file paths.

Add these Windows Forms controls or components to the form. Fields referenced by Handles clauses must be declared WithEvents, as the Visual Studio designer normally does.

Name	Type
Button1	Button
Button2	Button

VB.NET example

```
Private Sub Form1_Load(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs) Handles
    MyBase.Load
    Call CAvax1.StartAvax()
    Call CAvax1.Add_Line(0, 0, 0, 10, 10, 0)
    Call CAvax1.Add_Line(0, 20, 0, 10, 5, 0)
    Call CAvax1.Add_Rectangle(-5, -10, 10, 5)
    CAvax1.Command = Avax.AvaxCommand.AutoLimits_c
End Sub
Private Sub Form1_FormClosed(ByVal eventSender As System.Object, ByVal eventArgs As
    System.Windows.Forms.FormClosedEventArgs) Handles Me.FormClosed
    Call cAvax1.EndAvax()
End Sub
Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
    Button1.Click
    CAvax1.Command = Avax.AvaxCommand.WinSel_c
End Sub
Private Sub Button2_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
    Button2.Click
    Dim Answer As Integer
    Answer = CAvax1.MaxSelectedItems
    MsgBox("MaxSelectedItems=" & Answer)
End Sub
```

MirrorItems

cAvax method

Purpose

Reflects objects across an infinite XY line. By default it changes the originals; fMirrorStay=True retains them and returns copies.

VB.NET declarations

```
Function MirrorItems(ByRef avMyHandles As Integer(), ByRef mirX1 As Double, ByRef mirY1 As Double, ByRef
    mirX2 As Double, ByRef mirY2 As Double, ByRef fMirrorStay As Boolean) As Boolean

Function MirrorItems(ByRef avMyHandles As Integer(), ByRef mirX1 As Double, ByRef mirY1 As Double, ByRef
    mirX2 As Double, ByRef mirY2 As Double) As Boolean
```

Parameters

Name	Type and passing	Meaning
avMyHandles	ByRef avMyHandles As Integer()	Input Integer() of the entity handles to process. Logical entries start at index 1; index 0 is reserved.
mirX1	ByRef mirX1 As Double	Is the x coordinate of the first point of the mirror line
mirY1	ByRef mirY1 As Double	Is the y coordinate of the first point of the mirror line
mirX2	ByRef mirX2 As Double	Is the x coordinate of the second point of the mirror line
mirY2	ByRef mirY2 As Double	Is the y coordinate of the second point of the mirror line
fMirrorStay	ByRef fMirrorStay As Boolean	A Boolean expression that if it is true then the original objects are retained

Use the declaration for the overload you call. ByRef arguments can receive outputs; declare variables and arrays with the exact indicated type. Optional defaults are shown in the declaration.

Return value

Boolean - True on success, false on error.

Remarks

The two mirror points define an infinite line in XY; coincident points return False. The short overload uses fMirrorStay=False and reflects the existing objects. Any two distinct finite Double points define a usable axis without a drawing-unit tolerance; subnormal separation and spans larger than Double.MaxValue are handled during matrix construction. The reflected XY coordinates use a complete affine calculation with one final rounding. Z coordinates and object attributes remain with the objects.

With fMirrorStay=True, originals remain and the caller ByRef handle array receives the copied handles. The original array object is not overwritten. The reflection includes nested blocks, text/RTF frames, picture resources, persistent external references, editable geometry records and imported CAD objects. Native dimension definitions reflect while their annotation layout follows their dimension style.

Nonfinite axis values, a required matrix translation outside Double storage, or unrepresentable output reject the complete operation before drawing/history changes. The axis values, source objects, caller handles, current construction input and previous Redo remain available after a rejected request.

Pass a one-based Integer handle array; index 0 is ignored and duplicate handles are processed once. Geometry uses Double. The control must be started and enabled. PauseAvax and LockEdit govern interactive input and do not prohibit this direct call. Before events can change the proposed targets or parameters, or cancel the operation. Caller arrays are separate from event arrays. The complete batch is prepared before editing. Invalid handles, unrepresentable geometry or unavailable copy capacity return False without a partial edit. A successful batch is one Undo operation and raises Change without NewUserItems. PauseAvaxEvents suppresses notifications. Application event-handler exceptions propagate; they do not roll back an edit already committed. A callback that replaces the drawing or starts new input prevents the older operation from overwriting that state.

Numeric changes from v2

Update the following variables, arrays or event-data receivers when porting the corresponding v2 call. The declarations above show the current overloads.

Parameter or value	v2 type	v3 type
mirX1	Single	Double
mirY1	Single	Double
mirX2	Single	Double
mirY2	Single	Double

For ByRef parameters, use a variable with the exact v3 type. Preserve array dimensions and the documented indexing. Moving from v2 to v3 explains conversion and legacy file-storage rules.

Example

Setup

Add an Avax.cAvax Windows Forms control named `cAvax1`. Paste the procedures into the form class. Example setup and supporting files explains references, initialization and file paths.

Add these Windows Forms controls or components to the form. Fields referenced by Handles clauses must be declared WithEvents, as the Visual Studio designer normally does.

Name	Type
Command1	Button

VB.NET example

```
Private Sub Form1_Load(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs) Handles
    MyBase.Load
    cAvax1.StartAvax()
    Call cAvax1.Add_Line(0, 0, 0, 10, 10, 0)
    Call cAvax1.Add_Line(0, 10, 0, 10, 10, 0)
End Sub
Private Sub Form1_FormClosed(ByVal eventSender As System.Object, ByVal eventArgs As
    System.Windows.Forms.FormClosedEventArgs) Handles Me.FormClosed
    cAvax1.EndAvax()
End Sub
Private Sub Command1_Click(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs)
    Handles Command1.Click
    Dim [Handles]() As Integer = Nothing
    Dim mirX1 As Double
    Dim mirY1 As Double
    Dim mirX2 As Double
    Dim mirY2 As Double
    Dim fMirrorStay As Boolean
    Dim Answer As Boolean
    If CAvax1.GetAllHandlesArr([Handles]) > 0 Then
        mirX1 = 12
        mirY1 = 0
        mirX2 = 12
        mirY2 = 12
        fMirrorStay = True
        Answer = CAvax1.MirrorItems([Handles], mirX1, mirY1, mirX2, mirY2, fMirrorStay)
    End If
End Sub
```

modbutton.DrawAvaxButton and NonThemeStyle

API reference

modbutton.DrawAvaxButton

Paints a button surface and a centered, single-line caption into a Windows GDI device context. Use this method for application-owned buttons or other custom-painted controls. It draws immediately; it does not create a control, detect mouse input, change the current AVAX command or store a drawing object. The application chooses the style and current visual state and repaints when that state changes.

```
Function DrawAvaxButton(ByRef hDC As Integer, ByRef hwnd As Integer,
    ByRef BtnStyle As AvaxButtonStyle, ByRef BtnEstate As modbutton.isEstate,
    ByRef sCaption As String, ByRef lLeft As Integer, ByRef lTop As Integer,
    ByRef lWidth As Integer, ByRef lHeight As Integer,
    ByRef lFontColor As Integer, ByRef lFontHighlightColor As Integer,
    ByRef lBackColor As Integer, ByRef lHighlightColor As Integer,
    ByRef bUseCustomColors As Boolean, ByRef bEnabled As Boolean) As Boolean
```

Parameter	Meaning
hDC	Borrowed, valid GDI device context with the desired font selected. The legacy Integer carries the significant 32-bit Windows handle bits, including a possible sign bit. This is not an AVAX entity handle. Obtain and release the native HDC using its owning Graphics object or Windows API.
hwnd	Original compatibility window-handle argument. No window is created or modified, and rendering does not require a window; zero is accepted for bitmap painting.
BtnStyle	One of the ten AvaxButtonStyle values. Determines the surface, border and gradient design.
BtnEstate	A modbutton.isEstate value describing normal, hovered, pressed, disabled or default appearance. This is a drawing input, not a notification or automatic input tracker.
sCaption	Unicode caption. An empty string paints the button without text. The selected HDC font is used; text is centered and clipped to the button.
lLeft, lTop	Upper-left position in pixels in an ordinary MM_TEXT device context. They are not world coordinates in the AVAX drawing.
lWidth, lHeight	Positive width and height in pixels. Very small bounds are clipped and may have no visible interior.
lFontColor	Normal caption color as a Windows RGB/COLORREF Integer.
lFontHighlightColor	Caption color for enabled hot, pressed and default states. Disabled captions use the Windows disabled-text color.
lBackColor, lHighlightColor	Normal and highlighted surface RGB/COLORREF colors. Individual styles mix these with shading or fixed decorative colors; they are not a request for a uniform fill.
bUseCustomColors	True uses the supplied colors where supported by the selected design. False uses the Windows system colors and each style's built-in palette.
bEnabled	False forces disabled drawing regardless of BtnEstate. Some decorative background details also use the system button-face color when disabled. No input events are enabled or disabled by this painting call.

Return value and lifetime

True means the paint request completed, including a valid request fully outside the current clip. False indicates an invalid handle, nonpositive dimensions, an unknown style/state, unsafe coordinate arithmetic or failure to prepare the GDI drawing region. The caller retains ownership of its HDC and font. All arguments retain their values despite the legacy ByRef declarations. The caller's selected objects, clipping, text/background colors, background mode and current drawing point are restored.

The method preserves the caller's clipping and adds the button boundary. Captions support Greek and other characters available in the selected font. Call from the painting thread and do not use the same device context concurrently. Coordinates and handles already used Integer in R2; they remain Integer in v3. They do not follow the Single-to-Double change used for drawing geometry.

VB.NET example

Place the following module in a Windows Forms application referencing the v3 DLL. Call PaintButton from your Paint handler with e.Graphics, a pixel Rectangle, a caption, a style and your current pressed state. GetHdc and ReleaseHdc must be paired even if painting fails. Do not call other Graphics methods while its HDC is borrowed.

```
Option Strict On
Imports System
Imports System.Drawing
Imports Avax

Public Module ButtonRenderingExample
    ' Call from a Windows Forms Paint handler with e.Graphics and a pixel rectangle.
    Public Function PaintButton(graphics As Graphics, area As Rectangle, caption As String, style As
    AvaxButtonStyle, pressed As Boolean) As Boolean
        Dim nativeDc = graphics.GetHdc()
        Try
            ' The legacy Integer API retains the low 32 bits of a Windows GDI handle.
            Dim dc = BitConverter.ToInt32(BitConverter.GetBytes(nativeDc.ToInt64()), 0), window = 0
            Dim state = If(pressed, modbutton.isEstate.statePressed, modbutton.isEstate.stateNormal)
            Dim left = area.Left, top = area.Top, width = area.Width, height = area.Height
            Dim textColor = ColorTranslator.ToOle(Color.Black), hotText = textColor
            Dim background = ColorTranslator.ToOle(Color.LightGray), highlight =
    ColorTranslator.ToOle(Color.LightSkyBlue)
            Dim customColors = True, enabled = True
            Return modbutton.DrawAvaxButton(dc, window, style, state, caption, left, top, width, height,
    textColor, hotText, background, highlight, customColors, enabled)
        Finally
            graphics.ReleaseHdc(nativeDc)
        End Try
    End Function
End Module
```

modbutton.NonThemeStyle

```
Property NonThemeStyle As AvaxButtonStyle
```

Gets or sets the shared legacy style preference. Its initial value is Normal (0). It retains the assigned enumeration value for the application to read; DrawAvaxButton always uses its explicit BtnStyle argument and does not automatically consult this property. Setting it does not repaint a control or change Windows themes.

```
Imports Avax

Public Module ButtonPreferenceExample
    Public Sub UseStoredStyle()
        Dim previous As AvaxButtonStyle = modbutton.NonThemeStyle
        Try
            modbutton.NonThemeStyle = AvaxButtonStyle.OfficeXP
            Dim chosenStyle As AvaxButtonStyle = modbutton.NonThemeStyle
            ' Pass chosenStyle to PaintButton or DrawAvaxButton.
        Finally
            modbutton.NonThemeStyle = previous
        End Try
    End Sub
End Module
```

AvaxButtonStyle

Selects the button surface design. These values retain their R2 numbers. The names identify built-in designs and do not change the operating-system theme.

Name	Value	Design
Normal	0	Classic raised or sunken rectangular border.
Soft	1	Soft, shallow button edge.

Name	Value	Design
Flat	2	Flat surface with state-dependent edge.
Java	3	Beveled Java-style border.
OfficeXP	4	Office-style flat highlight.
WindowsXP	5	Rounded Windows XP-style surface.
Plastik	6	Shaded plastic-style surface.
Galaxy	7	Rounded shaded Galaxy design.
Keramik	8	Rounded shaded Keramik design.
MacOSX	9	Decorative glossy surface.

modbutton.isEstate

Describes the visual state supplied to DrawAvaxButton. The caller manages interaction and chooses the state before repainting.

Name	Value	Meaning
statenormal	1	Ordinary enabled button.
stateHot	2	Highlighted button, commonly while the pointer is over it.
statePressed	3	Button shown as pressed.
statedisabled	4	Disabled appearance.
stateDefaulted	5	Default-action emphasis where supported by the style.

MoveItems

cAvax method

Purpose

Moves the objects described in a specific handles array by a relative x and y distance

VB.NET declaration

```
Function MoveItems(ByRef avMyHandles As Integer(), ByRef xDist As Double, ByRef yDist As Double) As
    Boolean
```

Parameters

Name	Type and passing	Meaning
avMyHandles	ByRef avMyHandles As Integer()	Input Integer() of the entity handles to process. Logical entries start at index 1; index 0 is reserved.
xDist	ByRef xDist As Double	Is the relative x distance
yDist	ByRef yDist As Double	Is the relative y distance

Use the declaration for the overload you call. ByRef arguments can receive outputs; declare variables and arrays with the exact indicated type. Optional defaults are shown in the declaration.

Return value

Boolean - True on success, false on error.

Remarks

Moves the supplied objects by the Double XY displacement (xDist,yDist). Their handles and the caller input handle array are retained. Z coordinates, font sizes, pen attributes, user properties and owned picture resources remain with the objects.

The translation includes nested block children, text/RTF layouts, native geometry and persistent external references. Native dimensions retain text gaps, direction values and inherited scale. Editable-shape position records and hatch pattern origins move with the geometry; associative hatches follow their moved boundaries. The complete changing batch is one Undo/Redo operation.

Pass a one-based Integer handle array; index 0 is ignored and duplicate handles are processed once. Geometry uses Double. The control must be started and enabled. PauseAvax and LockEdit govern interactive input and do not prohibit this direct call. Before events can change the proposed targets or parameters, or cancel the operation. Caller arrays are separate from event arrays. The complete batch is prepared before editing. Invalid handles, unrepresentable geometry or unavailable copy capacity return False without a partial edit. A successful batch is one Undo operation and raises Change without NewUserItems. PauseAvaxEvents suppresses notifications. Application event-handler exceptions propagate; they do not roll back an edit already committed. A callback that replaces the drawing or starts new input prevents the older operation from overwriting that state.

Numeric changes from v2

Update the following variables, arrays or event-data receivers when porting the corresponding v2 call. The declarations above show the current overloads.

Parameter or value	v2 type	v3 type
xDist	Single	Double
yDist	Single	Double

For ByRef parameters, use a variable with the exact v3 type. Preserve array dimensions and the documented indexing. Moving from v2 to v3 explains conversion and legacy file-storage rules.

Example

Setup

Add an Avax.cAvax Windows Forms control named `cAvax1`. Paste the procedures into the form class. Example setup and supporting files explains references, initialization and file paths.

Add these Windows Forms controls or components to the form. Fields referenced by Handles clauses must be declared WithEvents, as the Visual Studio designer normally does.

Name	Type
Command1	Button

VB.NET example

```
Private Sub Form1_Load(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs) Handles
    MyBase.Load
    cAvax1.StartAvax()
    Call cAvax1.Add_Line(0, 0, 0, 10, 10, 0)
    Call cAvax1.Add_Line(0, 10, 0, 10, 10, 0)
End Sub
Private Sub Form1_FormClosed(ByVal eventSender As System.Object, ByVal eventArgs As
    System.Windows.Forms.FormClosedEventArgs) Handles Me.FormClosed
    cAvax1.EndAvax()
End Sub
Private Sub Command1_Click(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs)
    Handles Command1.Click
    Dim [Handles]() As Integer = Nothing
    Dim xDist As Double
    Dim yDist As Double
    Dim Answer As Boolean
    If CAvax1.GetAllHandlesArr([Handles]) > 0 Then
        xDist = 5
        yDist = 3
        Answer = CAvax1.MoveItems([Handles], xDist, yDist)
    End If
End Sub
```

MoveNode

cAvax method

Purpose

Moves matching drawing vertices to a new XYZ position, or translates a whole object when its anchor matches the search area.

VB.NET declarations

```
Function MoveNode(ByRef xOld As Double, ByRef yOld As Double, ByRef zOld As Double, ByRef xNew As Double,
    ByRef yNew As Double, ByRef zNew As Double) As Boolean

Function MoveNode(ByRef xOld As Double, ByRef yOld As Double, ByRef zOld As Double, ByRef xNew As Double,
    ByRef yNew As Double, ByRef zNew As Double, ByRef Breadth As Double) As Boolean
```

Parameters

Name	Type and passing	Meaning
xOld	ByRef xOld As Double	X coordinate of the search-box center, as Double.
yOld	ByRef yOld As Double	Y coordinate of the search-box center, as Double.
zOld	ByRef zOld As Double	Z coordinate of the search-box center, as Double. Elevation participates in matching.
xNew	ByRef xNew As Double	New X coordinate for matching vertices; whole objects use the displacement xNew-xOld.
yNew	ByRef yNew As Double	New Y coordinate for matching vertices; whole objects use the displacement yNew-yOld.
zNew	ByRef zNew As Double	New Z coordinate for matching vertices; whole objects use the displacement zNew-zOld.
Breadth	ByRef Breadth As Double	Nonnegative half-size of the search box on each coordinate axis, as Double. Boundaries are included. Zero, also used by the shorter overload, matches exact XYZ coordinates.

Use the declaration for the overload you call. ByRef arguments can receive outputs; declare variables and arrays with the exact indicated type. Optional defaults are shown in the declaration.

Return value

Boolean - True on success, false on error.

Remarks

Moves matching nodes throughout the drawing. For points, lines, polylines and shapes, each XYZ vertex inside the search box is replaced by (xNew, yNew, zNew). Breadth is the nonnegative half-size along each of X, Y and Z; boundaries are included. The six-argument overload uses Breadth=0, which requires exact coordinate equality. Ellipses and circular curves are matched by their transformed center, text by its insertion point, and blocks by the minimum corner of their public bounds. These whole objects are translated by (xNew-xOld, yNew-yOld, zNew-zOld), preserving their dimensions and any offset of their anchor within the search box. Other object kinds are unchanged. All live drawing objects are considered, including objects on hidden layers. StartAvax and Enabled=True are required. Returns True when changes are committed, or False for no matching change, invalid input or a rejected operation. Coordinates and Breadth must be finite Double values. A successful batch is one undo operation; caller selection, AutoRedraw and ContinueUndo are preserved. Whole-object moves raise BeforeMove separately for each matched object; handlers may change the target handles and XY displacement or cancel that whole-object move. A cancelled whole-object move leaves other matching vertices eligible. PauseAvaxEvents suppresses these callbacks. Invalid callback values or geometry reject the prepared batch without partial changes; if a callback edits or closes the drawing, the pending move stops and preserves the callback's own changes. For blocks containing external line references, the host geometry is refreshed before matching the lower bounding corner. Child text and document layouts, imported gradient frames and world-coordinate hatch origins move with the block; local hatch definitions keep their local origin. Large translations retain conic radii and affine axes, including when the intermediate displacement exceeds the Double range but the resulting stored geometry is finite. Invalid resulting geometry rejects the complete operation and leaves the drawing unchanged. A top-level ExLines_i reference is excluded from individual node editing, as in R2. Use MoveItems or StretchItems to transform the complete reference, or change the coordinates supplied by its SetExternalLines handler. Moving a containing block still moves its external children together with the block.

Numeric changes from v2

Update the following variables, arrays or event-data receivers when porting the corresponding v2 call. The declarations above show the current overloads.

Parameter or value	v2 type	v3 type
xOld	Single	Double
yOld	Single	Double
zOld	Single	Double
xNew	Single	Double
yNew	Single	Double
zNew	Single	Double
Breadth	Single	Double

For ByRef parameters, use a variable with the exact v3 type. Preserve array dimensions and the documented indexing. Moving from v2 to v3 explains conversion and legacy file-storage rules.

Example

Setup

Add an Avax.cAvax Windows Forms control named `cAvax1`. Paste the procedures into the form class. Example setup and supporting files explains references, initialization and file paths.

Add these Windows Forms controls or components to the form. Fields referenced by Handles clauses must be declared WithEvents, as the Visual Studio designer normally does.

Name	Type
Button1	Button

VB.NET example

```
Private Sub Form1_Load(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs) Handles
    MyBase.Load
    Call CAvax1.StartAvax()
    Call CAvax1.Add_Ellipse(10, 10, 0, 10, 5)
    Call CAvax1.Add_Ellipse(-10, -10, 0, 10, 5)
```

```
    CAvax1.Command = Avax.AvaxCommand.AutoLimits_c
End Sub
Private Sub Form1_FormClosed(ByVal eventSender As System.Object, ByVal eventArgs As
    System.Windows.Forms.FormClosedEventArgs) Handles Me.FormClosed
    Call cAvax1.EndAvax()
End Sub
Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
    Button1.Click
    Dim xOld As Double
    Dim yOld As Double
    Dim zOld As Double
    Dim xNew As Double
    Dim yNew As Double
    Dim zNew As Double
    Dim Breadth As Double
    Dim Answer As Boolean
    xOld = 10
    yOld = 10
    xNew = 15
    yNew = 15
    Breadth = 3
    Answer = cAvax1.MoveNode(xOld, yOld, zOld, xNew, yNew, zNew, Breadth)
End Sub
```

NearestPaletteIndexColor

cAvax method

Purpose

This function returns the index to the current color palette with a RGB color which matches or looks like the specific RGB color

VB.NET declaration

```
Function NearestPaletteIndexColor(ByRef RGBColor As Integer) As Integer
```

Parameters

Name	Type and passing	Meaning
RGBColor	ByRef RGBColor As Integer	Input OLE/BGR Integer, unchanged on return. -1 returns index zero without inserting a color. Other missing colors may be appended to the palette.

Use the declaration for the overload you call. ByRef arguments can receive outputs; declare variables and arrays with the exact indicated type. Optional defaults are shown in the declaration.

Return value

Integer - Is the nearest palette color index

Numeric changes from v2

Update the following variables, arrays or event-data receivers when porting the corresponding v2 call. The declarations above show the current overloads.

Parameter or value	v2 type	v3 type
Return value	Short	Integer

For ByRef parameters, use a variable with the exact v3 type. Preserve array dimensions and the documented indexing. Moving from v2 to v3 explains conversion and legacy file-storage rules.

Example

Setup

Add an Avax.cAvax Windows Forms control named `cAvax1`. Paste the procedures into the form class. Example setup and supporting files explains references, initialization and file paths.

Add these Windows Forms controls or components to the form. Fields referenced by Handles clauses must be declared WithEvents, as the Visual Studio designer normally does.

Name	Type
Command1	Button

VB.NET example

```
Private Sub Form1_Load(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs) Handles
    MyBase.Load
    Command1.Text = "Get NearestPaletteIndexColor"
    cAvax1.StartAvax()
End Sub
Private Sub Form1_FormClosed(ByVal eventSender As System.Object, ByVal eventArgs As
    System.Windows.Forms.FormClosedEventArgs) Handles Me.FormClosed
    cAvax1.EndAvax()
End Sub
Private Sub Command1_Click(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs)
```

```
    Handles Command1.Click
    Dim RGBColor As Integer
    RGBColor = System.Drawing.ColorTranslator.ToOle(System.Drawing.Color.Blue)
    MsgBox("Blue = color palette index " & Format(CAvax1.NearestPaletteIndexColor(RGBColor)),
    MsgBoxStyle.Information, "NearestPaletteIndexColor")
End Sub
```

OleDrag

Purpose

Starts a synchronous OLE copy drag for the selected drawing objects. The data includes complete geometry and resources in the native v3 drawing format. A legacy AVAX clipboard representation is also supplied when the selection can be represented by R2. Copying does not change the source drawing, its selection or its undo history.

VB.NET declaration

```
Sub OleDrag()
```

Parameters and return value

No parameters or return value. The control must be started and enabled, with a nonempty selection. Paused, locked, ended or disposed controls do not start a drag. The mouse position in the source drawing is the insertion anchor. The call returns when the OLE operation completes or is cancelled.

Drop behavior

A target AVAX control with OLEDropMode enabled can insert the native or legacy drawing at the target mouse position. It raises OnDragOver and OnDragDrop; host cancellation and state changes are honored before committing one undoable insertion. Other applications must understand one of the advertised AVAX data formats to accept these objects. The advertised data formats are `Gardos AVAX v3 Drawing` (native drawing bytes) and, when available, `AVAX Clipboard Data Version 1.1` (a MemoryStream containing the R2 representation). The operation advertises Copy, never Move, and does not change the Windows clipboard.

Example

Call from the control's owning UI thread, for example from a mouse gesture that starts a drag. selectedHandles uses the AVAX handle-array convention with a leading zero.

```
Option Strict On
Imports System
Imports Avax

Public Module Help_OleDrag
    Public Sub BeginSelectionDrag(cad As cAvax, selectedHandles As Integer())
        If Not cad.SelectItems(selectedHandles) Then
            Throw New ArgumentException("Select existing drawing objects before starting the drag.")
        End If
        cad.OleDrag()
    End Sub
End Module
```

Related options and reference

ParametricShapeDefinitions

Purpose

Reads the preset catalog without creating a drawing control. Use it to populate a shape gallery or parameter editor before a drawing is opened.

Declaration

```
Shared Function ParametricShapeDefinitions() As AvaxShapeDefinition()
```

Result

Returns a new array of preset definitions with new parameter descriptions on each call. These are detached descriptions: changing a returned value does not change the built-in catalog or register a new shape. It returns the same catalog as the instance method GetParametricShapes. See Built-in shape catalog for all current identifiers, bounds and defaults, and that instance method's result tables for the data members.

VB.NET example

```
Option Strict On
Public Module ParametricShapeDefinitionsExample
    Public Sub ListPresetNames()
        Dim catalog As Avax.AvaxShapeDefinition() =
            Avax.cAvax.ParametricShapeDefinitions()
        For Each definition As Avax.AvaxShapeDefinition In catalog
            System.Diagnostics.Debug.WriteLine(definition.Kind)
        Next
    End Sub
End Module
```

Related options and reference

Paste_Avax

cAvax method

Purpose

Imports into the drawing the contents of an Avax Clipboard bytes array

VB.NET declarations

```
Function Paste_Avax(ByRef x As Double, ByRef y As Double, ByRef AvaxClipboard As Byte(), ByRef vRetHandles
     As Object) As Boolean

Function Paste_Avax(ByRef x As Double, ByRef y As Double, ByRef AvaxClipboard As Byte()) As Boolean
```

Parameters

Name	Type and passing	Meaning
x	ByRef x As Double	Is the x coordinate of the insertion point
y	ByRef y As Double	Is the y coordinate of the insertion point
AvaxClipboard	ByRef AvaxClipboard As Byte()	An array of Bytes with the Avax clipboard data
vRetHandles	ByRef vRetHandles As Object	Returns an integer array with the handles of the new objects

Use the declaration for the overload you call. ByRef arguments can receive outputs; declare variables and arrays with the exact indicated type. Optional defaults are shown in the declaration.

Return value

Boolean - True on success, false on error.

Numeric changes from v2

Update the following variables, arrays or event-data receivers when porting the corresponding v2 call. The declarations above show the current overloads.

Parameter or value	v2 type	v3 type
x	Single	Double
y	Single	Double

For ByRef parameters, use a variable with the exact v3 type. Preserve array dimensions and the documented indexing. Moving from v2 to v3 explains conversion and legacy file-storage rules.

Example

Setup

Add an Avax.cAvax Windows Forms control named `cAvax1`. Paste the procedures into the form class. Example setup and supporting files explains references, initialization and file paths.

Add these Windows Forms controls or components to the form. Fields referenced by Handles clauses must be declared WithEvents, as the Visual Studio designer normally does.

Name	Type
Command1	Button

VB.NET example

```
Dim h(2) As Integer
Private Sub Form1_Load(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs) Handles
    MyBase.Load
```

```
    cAvax1.StartAvax()
    h(1) = cAvax1.Add_Line(0, 0, 0, 10, 10, 0)
    h(2) = cAvax1.Add_Line(0, 10, 0, 10, 0, 0)
End Sub
Private Sub Form1_FormClosed(ByVal eventSender As System.Object, ByVal eventArgs As
    System.Windows.Forms.FormClosedEventArgs) Handles Me.FormClosed
    cAvax1.EndAvax()
End Sub
Private Sub Command1_Click(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs)
    Handles Command1.Click
    Dim x As Double
    Dim y As Double
    Dim AvaxClipboard() As Byte = Nothing
    Dim vRetHandles As Object = Nothing
    Dim Answer As Boolean
    x = 5
    y = 5
    AvaxClipboard = cAvax1.Copy_Avax(h, x, y)
    x = 10
    y = 5
    Answer = cAvax1.Paste_Avax(x, y, AvaxClipboard, vRetHandles)
End Sub
```

PasteFromClipboard

cAvax method

Purpose

Imports into the drawing the contents of the Clipboard

VB.NET declarations

```
Function PasteFromClipboard(ByRef xBase As Double, ByRef yBase As Double, ByVal dx As Double, ByVal dy As
    Double, ByRef zOrder As Integer) As Integer

Function PasteFromClipboard(ByRef xBase As Double, ByRef yBase As Double, ByVal dx As Double, ByVal dy As
    Double, ByRef zOrder As Integer, ByRef vItemProperties As Object, ByRef iLayer As Integer, ByRef nMapMode
    As Integer) As Integer
```

Parameters

Name	Type and passing	Meaning
xBase	ByRef xBase As Double	Is the x coordinate of the insertion point
yBase	ByRef yBase As Double	Is the y coordinate of the insertion point
dx	ByVal dx As Double	Double in v3 (Double in R2). Requested image size in drawing units. When both dx and dy are zero, the image physical dimensions in millimetres are divided by PrintUnit_MM. WMF/EMF clipboard frames have 0.01 mm precision. The CHM example explicitly requests a 10 by 10 image.
dy	ByVal dy As Double	Double in v3 (Double in R2). Requested image size in drawing units. When both dx and dy are zero, the image physical dimensions in millimetres are divided by PrintUnit_MM. WMF/EMF clipboard frames have 0.01 mm precision. The CHM example explicitly requests a 10 by 10 image.
zOrder	ByRef zOrder As Integer	Is the new object's elevation in the whole drawing
vItemProperties	ByRef vItemProperties As Object	Is a string array with properties for the new object
iLayer	ByRef iLayer As Integer	Is the layer number
nMapMode	ByRef nMapMode As Integer	Is the API's mapmode

Use the declaration for the overload you call. ByRef arguments can receive outputs; declare variables and arrays with the exact indicated type. Optional defaults are shown in the declaration.

Return value

Integer - An Avax handle on success, zero on error.

Remarks

If dx=dy=0 then the program will calculate the width and height in accordance with the AVAX property AvaxProp.PrintUnit_MM_p

Numeric changes from v2

Update the following variables, arrays or event-data receivers when porting the corresponding v2 call. The declarations above show the current overloads.

Parameter or value	v2 type	v3 type
xBase	Single	Double
yBase	Single	Double
dx	Single	Double
dy	Single	Double

Parameter or value	v2 type	v3 type
iLayer	Short	Integer

For ByRef parameters, use a variable with the exact v3 type. Preserve array dimensions and the documented indexing. Moving from v2 to v3 explains conversion and legacy file-storage rules.

Example

Setup

Add an Avax.cAvax Windows Forms control named `cAvax1`. Paste the procedures into the form class. Example setup and supporting files explains references, initialization and file paths.

Add these Windows Forms controls or components to the form. Fields referenced by Handles clauses must be declared WithEvents, as the Visual Studio designer normally does.

Name	Type
Command1	Button

VB.NET example

```
Private Sub Form1_Load(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs) Handles
    MyBase.Load
    Command1.Text = "PasteFromClipboard"
    cAvax1.StartAvax()
End Sub
Private Sub Form1_FormClosed(ByVal eventSender As System.Object, ByVal eventArgs As
    System.Windows.Forms.FormClosedEventArgs) Handles Me.FormClosed
    cAvax1.EndAvax()
End Sub
Private Sub Command1_Click(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs)
    Handles Command1.Click
    Dim xBase As Double
    Dim yBase As Double
    Dim dx As Double
    Dim dy As Double
    Dim zOrder As Integer
    Dim nMapMode As Integer
    nMapMode = 8
    xBase = 0
    yBase = 0
    dx = 10
    dy = 10
    Call cAvax1.PasteFromClipboard(xBase, yBase, dx, dy, zOrder)
End Sub
```

PrintScreenToBMP

cAvax method

Purpose

Exports the drawing that is shown on the screen into a .BMP file

VB.NET declaration

```
Function PrintScreenToBMP(ByRef sBmpFile As String) As Boolean
```

Parameters

Name	Type and passing	Meaning
sBmpFile	ByRef sBmpFile As String	String BMP destination, unchanged ByRef. Image dimensions are the actual control client area; the screen frame is regenerated with normal paint callbacks unless events are paused. Invalid or cancelled capture preserves an existing file.

Use the declaration for the overload you call. ByRef arguments can receive outputs; declare variables and arrays with the exact indicated type. Optional defaults are shown in the declaration.

Return value

Boolean - True on success, false on error.

Example

Setup

Add an Avax.cAvax Windows Forms control named `cAvax1`. Paste the procedures into the form class. Example setup and supporting files explains references, initialization and file paths.

Add these Windows Forms controls or components to the form. Fields referenced by Handles clauses must be declared WithEvents, as the Visual Studio designer normally does.

Name	Type
Button1	Button

VB.NET example

```
Private Sub Form1_Load(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs) Handles
    MyBase.Load
    Call CAvax1.StartAvax()
    Call CAvax1.Add_Line(0, 5, 0, 10, 5, 0, Nothing, 1, 1, -1, -1)
    Call CAvax1.Add_Line(5, 0, 0, 5, 10, 0, Nothing, 1, 1, -1, -1)
    Call CAvax1.Add_Circle(5, 5, 0, 3, Nothing, 5, -1, -1, -1)
End Sub
Private Sub Form1_FormClosed(ByVal eventSender As System.Object, ByVal eventArgs As
    System.Windows.Forms.FormClosedEventArgs) Handles Me.FormClosed
    Call cAvax1.EndAvax()
End Sub
Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
    Button1.Click
    Dim sBmpFile As String = String.Empty
    Dim Answer As Boolean
    sBmpFile = My.Application.Info.DirectoryPath & "\..\temp.bmp"
    Answer = CAvax1.PrintScreenToBMP(sBmpFile)
End Sub
```

PrintToAPL

Purpose

Exports the drawing, or a specified drawing rectangle, to the legacy APL plot format. Vector geometry is clipped to the export bounds; text, images and fills are represented by plot lines.

Declarations

```
Function PrintToAPL(ByVal fileName As String) As Boolean

Function PrintToAPL(ByVal fileName As String, ByVal xMin As Double, ByVal yMin As Double, ByVal xMax As
    Double, ByVal yMax As Double) As Boolean
```

Parameters and result

Name	Meaning
fileName	Destination path. An .apl extension is conventional; this method selects APL encoding independently of the filename extension. Relative paths use the current working directory and the directory must exist.
xMin, yMin, xMax, yMax	Finite Double drawing coordinates of the export rectangle. The short overload uses drawing extents. Four zeros also request drawing extents; four minus-ones request the stored current view. Other reversed corners are normalized. A zero-width or zero-height rectangle is expanded by half a drawing unit on each side of that flat axis.

Returns True after output is committed, False when the operation cannot start or a handled error or cancellation prevents completion. The control must be started, enabled and undisposed, with output available and no other print/export operation active. Call on its owning thread.

Output coordinates and format limits

The APL rectangle starts at 0,0: emitted X/Y coordinates are relative to the requested bounds' minimum corner. Printable layer visibility applies. APL stores binary 20-byte records: four IEEE Single coordinates and two Int16 color/width attributes. The v3 arguments and clipping calculations use Double, then narrow at the legacy file boundary. Coordinates beyond Single range or attributes outside the legacy nonnegative Int16 range are rejected. APL carries plot records rather than editable object definitions and embedded resource payloads.

Failure and cancellation

The drawing filename, dirty state and Undo history are retained. Live external references are refreshed before export. A changed drawing/control context or AbortPrinting can cancel the pending output. The writer uses an owned temporary file and replaces the destination only after completion, preserving an existing file on a handled failure or cancellation. LastOperationError can provide a diagnostic after False; early rejection does not guarantee a fresh message.

VB.NET example: drawing extents

```
Option Strict On
Public Module PrintToAplDrawingExample
    Public Function WriteDrawingPlot(
        cad As Avax.cAvax, destination As String) As Boolean
        Dim saved As Boolean = cad.PrintToAPL(destination)
        If Not saved AndAlso Not String.IsNullOrEmpty(cad.LastOperationError) Then
            System.Diagnostics.Debug.WriteLine(cad.LastOperationError)
        End If
        Return saved
    End Function
End Module
```

VB.NET example: explicit bounds

```
Option Strict On
Public Module PrintToAplBoundsExample
    Public Function WritePlotArea(
        cad As Avax.cAvax, destination As String,
        left As Double, bottom As Double,
        right As Double, top As Double) As Boolean
        Dim saved As Boolean = cad.PrintToAPL(
```

```
            destination, left, bottom, right, top)
        If Not saved AndAlso Not String.IsNullOrEmpty(cad.LastOperationError) Then
            System.Diagnostics.Debug.WriteLine(cad.LastOperationError)
        End If
        Return saved
    End Function
End Module
```

Related options and reference

PrintToBMP

cAvax method

Purpose

Exports the drawing into a .BMP file

VB.NET declarations

```
Function PrintToBMP(ByRef sBmpFile As String, ByVal lWidth As Integer, ByVal lHeight As Integer, ByRef
    fSelected As Boolean, ByRef xMin As Double, ByRef yMin As Double, ByRef xMax As Double, ByRef yMax As
    Double, ByRef sScale As Double) As Boolean

Function PrintToBMP(ByRef sBmpFile As String, ByVal lWidth As Integer, ByVal lHeight As Integer) As
    Boolean
```

Parameters

Name	Type and passing	Meaning
sBmpFile	ByRef sBmpFile As String	String destination filename, unchanged ByRef. Output is actual BMP even if the extension differs. Relative paths are resolved before host callbacks; an existing file is replaced only after complete successful preparation.
lWidth	ByVal lWidth As Integer	Integer image width in pixels, exactly as written in the BMP header. Zero derives width from positive height and whole-drawing aspect ratio. Both zero fail; negative/overflow/allocation failures preserve existing output. Unlike the old implementation this is not width+1.
lHeight	ByVal lHeight As Integer	Integer image height in pixels, with the same rules as lWidth. Width and height stay ByVal. Fit includes the last pixel centre so bottom/right boundary lines remain visible.
fSelected	ByRef fSelected As Boolean	True exports only selected objects and fails if the selection is empty. Automatic extents still use the complete drawing as in R2. This ByRef flag is unchanged.
xMin	ByRef xMin As Double	Double left coordinate. Four zeros mean complete drawing extents; four minus-ones mean the current nominal view. Actual normalized extents are returned only on success.
yMin	ByRef yMin As Double	Double bottom coordinate. All extent coordinates must be finite and produce a representable rendering scale.
xMax	ByRef xMax As Double	Double right coordinate. Explicit reversed bounds normalize; exactly zero spans receive a usable usable nonzero frame (adjacent finite Double coordinates at large origins).
yMax	ByRef yMax As Double	Double top coordinate; corrects the CHM table's reference to an x coordinate. Failed preparation/file commit leaves all extent ByRefs unchanged.
sScale	ByRef sScale As Double	Double geometric multiplier about the output top-left. Finite values <=0 become 1, committed only on success. NaN/infinity/unrepresentable scale fail; scaling larger than one may intentionally clip geometry.

Use the declaration for the overload you call. ByRef arguments can receive outputs; declare variables and arrays with the exact indicated type. Optional defaults are shown in the declaration.

Return value

Boolean - True on success, false on error

Remarks

If all the xMin, yMin, xMax, yMax are zero, then the whole drawing will be printed

Numeric changes from v2

Update the following variables, arrays or event-data receivers when porting the corresponding v2 call. The declarations above show the current overloads.

Parameter or value	v2 type	v3 type
xMin	Single	Double
yMin	Single	Double
xMax	Single	Double
yMax	Single	Double
sScale	Single	Double

For ByRef parameters, use a variable with the exact v3 type. Preserve array dimensions and the documented indexing. Moving from v2 to v3 explains conversion and legacy file-storage rules.

Example

Setup

Add an Avax.cAvax Windows Forms control named `cAvax1`. Paste the procedures into the form class. Example setup and supporting files explains references, initialization and file paths.

Add these Windows Forms controls or components to the form. Fields referenced by Handles clauses must be declared WithEvents, as the Visual Studio designer normally does.

Name	Type
Command1	Button

VB.NET example

```
Private Sub Form1_Load(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs) Handles
    MyBase.Load
    Command1.Text = "PrintToBMP"
    cAvax1.StartAvax()
    Call cAvax1.Add_Line(0, 5, 0, 10, 5, 0, Nothing, 1, 1, -1, -1)
    Call cAvax1.Add_Line(5, 0, 0, 5, 10, 0, Nothing, 1, 1, -1, -1)
    Call cAvax1.Add_Circle(5, 5, 0, 3, Nothing, 5, -1, -1, -1)
End Sub
Private Sub Form1_FormClosed(ByVal eventSender As System.Object, ByVal eventArgs As
    System.Windows.Forms.FormClosedEventArgs) Handles Me.FormClosed
    cAvax1.EndAvax()
End Sub
Private Sub Command1_Click(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs)
    Handles Command1.Click
    Dim sBmpFile As String = String.Empty
    Dim lWidth As Integer
    Dim lHeight As Integer
    Dim fSelected As Boolean
    Dim xMin As Double
    Dim yMin As Double
    Dim xMax As Double
    Dim yMax As Double
    Dim sScale As Double
    Dim Answer As Boolean
    sBmpFile = My.Application.Info.DirectoryPath & "\..\temp.bmp"
    lWidth = 120
    lHeight = 120
    sScale = 1
    Answer = cAvax1.PrintToBMP(sBmpFile, lWidth, lHeight, fSelected, xMin, yMin, xMax, yMax, sScale)
End Sub
```

PrintToDWG

cAvax method

Purpose

Exports the drawing into a .DWG file

VB.NET declarations

```
Function PrintToDWG(ByRef sDWGFile As String) As Boolean

Function PrintToDWG(ByRef sDWGFile As String, ByRef fSelected As Boolean, ByRef xBase As Double, ByRef
    yBase As Double, ByRef zBase As Double, ByRef Version As AvaxACADVersion) As Boolean
```

Parameters

Name	Type and passing	Meaning
sDWGFile	ByRef sDWGFile As String	Is the name of the DWG file
fSelected	ByRef fSelected As Boolean	A Boolean expression that if it is true then only the selected objects are printed
xBase	ByRef xBase As Double	Is the x coordinate of the insertion point of the DWG file
yBase	ByRef yBase As Double	Is the y coordinate of the insertion point of the DWG file
zBase	ByRef zBase As Double	Is the z coordinate of the insertion point of the DWG file
Version	ByRef Version As AvaxACADVersion	An expression that defines the version of the DWG file

Use the declaration for the overload you call. ByRef arguments can receive outputs; declare variables and arrays with the exact indicated type. Optional defaults are shown in the declaration.

Return value

Boolean - True on success, false on error

Remarks

Writes binary DWG in AutoCAD 2000, 2004, 2010, 2013 or 2018 format. ACAD_10 requests use the nearest supported DWG version, AutoCAD 2000 (AC1015), and retain the caller Version argument. The short overload writes the whole drawing as AutoCAD 2018. The full overload optionally exports the selected objects and subtracts the Double xBase, yBase and zBase coordinates without moving the source drawing. StartAvax and an enabled control are required. Closed Solid_f shapes and the SOLID pattern in PatACAD_f retain their fill, including concave boundaries, holes and nested islands. Fill foreground and visible perimeter pen colors are independent; hidden borders do not produce triangle-edge strokes. Tilted triangles are stored in their own planes to retain their Z heights. Horizontal solid corners and perimeter vertices retain the supplied Double coordinates. AutoCAD 2000 uses the nearest indexed color; later supported versions retain RGB. Plain multiline Document text and imported dimension labels export as editable MTEXT, retaining Unicode, font, position, rotation and positive axis scales. OnlyLinesACADExportOn converts supported curves, paths, text and horizontal PAT fills to static linework while retaining solid fills and independent visible borders. VectorTextExportOn outlines supported text and native dimension labels. Transparent native hatches require native HATCH output with OnlyLinesACADExportOn disabled. Images, gradients, unsupported fill patterns, shaped text and formatted, sheared or reflected documents remain outside the editable-text export path. Errors return False and preserve an existing destination. AutoCAD 2007 output remains unsupported. The example keeps its original ACAD_10 request and writes AutoCAD 2000 through the compatibility mapping.

Numeric changes from v2

Update the following variables, arrays or event-data receivers when porting the corresponding v2 call. The declarations above show the current overloads.

Parameter or value	v2 type	v3 type
xBase	Single	Double

Parameter or value	v2 type	v3 type
yBase	Single	Double
zBase	Single	Double

For ByRef parameters, use a variable with the exact v3 type. Preserve array dimensions and the documented indexing. Moving from v2 to v3 explains conversion and legacy file-storage rules.

Example

Setup

Add an Avax.cAvax Windows Forms control named `cAvax1`. Paste the procedures into the form class. Example setup and supporting files explains references, initialization and file paths.

Add these Windows Forms controls or components to the form. Fields referenced by Handles clauses must be declared WithEvents, as the Visual Studio designer normally does.

Name	Type
Command1	Button

The example uses AutoCAD 2000, a supported DWG output version.

VB.NET example

```
Private Sub Form1_Load(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs) Handles
    MyBase.Load
    Command1.Text = "PrintToDWG"
    cAvax1.StartAvax()
    Call cAvax1.Add_Line(0, 5, 0, 10, 5, 0, Nothing, 1, 1, -1, -1)
    Call cAvax1.Add_Line(5, 0, 0, 5, 10, 0, Nothing, 1, 1, -1, -1)
    Call cAvax1.Add_Circle(5, 5, 0, 3, Nothing, 5, -1, -1, -1)
End Sub
Private Sub Form1_FormClosed(ByVal eventSender As System.Object, ByVal eventArgs As
    System.Windows.Forms.FormClosedEventArgs) Handles Me.FormClosed
    cAvax1.EndAvax()
End Sub
Private Sub Command1_Click(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs)
    Handles Command1.Click
    Dim sDWGFile As String = String.Empty
    Dim fSelected As Boolean
    Dim xBase As Double
    Dim yBase As Double
    Dim zBase As Double
    Dim Version As Avax.AvaxACADVersion
    Dim Answer As Boolean
    sDWGFile = My.Application.Info.DirectoryPath & "\..\temp.DWG"
    Version = Avax.AvaxACADVersion.ACAD_10
    xBase = 0
    yBase = 0
    zBase = 0
    fSelected = False
    Answer = cAvax1.PrintToDWG(sDWGFile, fSelected, xBase, yBase, zBase, Version)
End Sub
```

PrintToDXF

cAvax method

Purpose

Exports the drawing into a .DXF file

VB.NET declarations

```
Function PrintToDXF(ByRef sDxfFile As String, ByRef fSelected As Boolean, ByRef xBase As Double, ByRef
    yBase As Double, ByRef zBase As Double, ByRef Version As AvaxACADVersion) As Boolean

Function PrintToDXF(ByRef sDxfFile As String) As Boolean
```

Parameters

Name	Type and passing	Meaning
sDxfFile	ByRef sDxfFile As String	Is the name of the DXF file
fSelected	ByRef fSelected As Boolean	A Boolean expression that if it is true then only the selected objects are printed
xBase	ByRef xBase As Double	Is the x coordinate of the insertion point of the DXF file
yBase	ByRef yBase As Double	Is the y coordinate of the insertion point of the DXF file
zBase	ByRef zBase As Double	Is the z coordinate of the insertion point of the DXF file
Version	ByRef Version As AvaxACADVersion	An expression that defines the version of the DXF file

Use the declaration for the overload you call. ByRef arguments can receive outputs; declare variables and arrays with the exact indicated type. Optional defaults are shown in the declaration.

Return value

Boolean - True on success, false on error

Remarks

Exports the drawing or the selected objects without changing the source geometry. The full overload subtracts xBase, yBase and zBase from exported coordinates. The short overload uses ACAD_10. The full overload accepts the eight named R2 AvaxACADVersion members and the additional v3 numeric codes 8 through 12 described in AvaxACADVersion. The public enum retains its eight original names; select an additional format with CType(code, Avax.AvaxACADVersion). StartAvax and an enabled control are required. Closed shapes with Solid_f, or the SOLID pattern in PatACAD_f, are exported as filled SOLID triangles. Concave boundaries, holes and nested islands retain their fill rule. The fill uses its foreground color; the separate visible perimeter uses the pen color. A hidden border produces no triangle-edge strokes. OnlyLinesACADExportOn does not remove solid fills or independent visible borders. It expands supported horizontal PAT fills into clipped linework. VectorTextExportOn outlines supported text and exports native dimensions as static graphics so their labels are outlined too. Original triangle corners retain their Double XYZ values; intersections created by triangulation interpolate height. Images, gradients and fill patterns that cannot be expanded to supported CAD geometry report an error. Transparent native hatches require native HATCH output; OnlyLinesACADExportOn rejects that conversion without changing an existing destination. The destination is replaced only after successful generation; failure preserves an existing file and returns False.

Numeric changes from v2

Update the following variables, arrays or event-data receivers when porting the corresponding v2 call. The declarations above show the current overloads.

Parameter or value	v2 type	v3 type
xBase	Single	Double
yBase	Single	Double
zBase	Single	Double

For ByRef parameters, use a variable with the exact v3 type. Preserve array dimensions and the documented indexing. Moving from v2 to v3 explains conversion and legacy file-storage rules.

Example

Setup

Add an Avax.cAvax Windows Forms control named `cAvax1`. Paste the procedures into the form class. Example setup and supporting files explains references, initialization and file paths.

Add these Windows Forms controls or components to the form. Fields referenced by Handles clauses must be declared WithEvents, as the Visual Studio designer normally does.

Name	Type
Command1	Button

VB.NET example

```
Private Sub Form1_Load(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs) Handles
    MyBase.Load
    Command1.Text = "PrintToDXF"
    cAvax1.StartAvax()
    Call cAvax1.Add_Line(0, 5, 0, 10, 5, 0, Nothing, 1, 1, -1, -1)
    Call cAvax1.Add_Line(5, 0, 0, 5, 10, 0, Nothing, 1, 1, -1, -1)
    Call cAvax1.Add_Circle(5, 5, 0, 3, Nothing, 5, -1, -1, -1)
End Sub
Private Sub Form1_FormClosed(ByVal eventSender As System.Object, ByVal eventArgs As
    System.Windows.Forms.FormClosedEventArgs) Handles Me.FormClosed
    cAvax1.EndAvax()
End Sub
Private Sub Command1_Click(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs)
    Handles Command1.Click
    Dim sDxfFile As String = String.Empty
    Dim fSelected As Boolean
    Dim xBase As Double
    Dim yBase As Double
    Dim zBase As Double
    Dim Version As Avax.AvaxACADVersion
    Dim Answer As Boolean
    sDxfFile = My.Application.Info.DirectoryPath & "\..\temp.dxf"
    Version = Avax.AvaxACADVersion.ACAD_10
    xBase = 0
    yBase = 0
    zBase = 0
    fSelected = False
    Answer = cAvax1.PrintToDXF(sDxfFile, fSelected, xBase, yBase, zBase, Version)
End Sub
```

PrintToGIF

cAvax method

Purpose

Exports the drawing into a .GIF file

VB.NET declarations

```
Function PrintToGIF(ByRef sGIFFile As String, ByVal lWidth As Integer, ByVal lHeight As Integer, ByRef
    fSelected As Boolean, ByRef xMin As Double, ByRef yMin As Double, ByRef xMax As Double, ByRef yMax As
    Double, ByRef sScale As Double) As Boolean

Function PrintToGIF(ByRef sGIFFile As String, ByVal lWidth As Integer, ByVal lHeight As Integer) As
    Boolean
```

Parameters

Name	Type and passing	Meaning
sGIFFile	ByRef sGIFFile As String	String destination path, unchanged ByRef. Relative paths resolve before callbacks. The actual GIF encoder is used regardless of extension; preparation or commit failure preserves existing output.
lWidth	ByVal lWidth As Integer	Integer image width in pixels, exactly as written in the GIF header. Zero derives width from positive height and whole-drawing aspect ratio. Both zero fail; negative/overflow/allocation failures preserve existing output. Unlike the old implementation this is not width+1. The GIF format has 16-bit dimensions, so values above 65535 fail.
lHeight	ByVal lHeight As Integer	Integer image height in pixels, with the same rules as lWidth. Width and height stay ByVal. Fit includes the last pixel centre so bottom/right boundary lines remain visible. The same 65535 format limit applies.
fSelected	ByRef fSelected As Boolean	True exports only selected objects and fails if the selection is empty. Automatic extents still use the complete drawing as in R2. This ByRef flag is unchanged.
xMin	ByRef xMin As Double	Double left coordinate. Four zeros mean complete drawing extents; four minus-ones mean the current nominal view. Actual normalized extents are returned only on success.
yMin	ByRef yMin As Double	Double bottom coordinate. All extent coordinates must be finite and produce a representable rendering scale.
xMax	ByRef xMax As Double	Double right coordinate. Explicit reversed bounds normalize; exactly zero spans receive a usable usable nonzero frame (adjacent finite Double coordinates at large origins).
yMax	ByRef yMax As Double	Double top coordinate; corrects the CHM table's reference to an x coordinate. Failed preparation/file commit leaves all extent ByRefs unchanged.
sScale	ByRef sScale As Double	Double geometric multiplier about the output top-left. Finite values <=0 become 1, committed only on success. NaN/infinity/unrepresentable scale fail; scaling larger than one may intentionally clip geometry.

Use the declaration for the overload you call. ByRef arguments can receive outputs; declare variables and arrays with the exact indicated type. Optional defaults are shown in the declaration.

Return value

Boolean - True on success, false on error.

Numeric changes from v2

Update the following variables, arrays or event-data receivers when porting the corresponding v2 call. The declarations above show the current overloads.

Parameter or value	v2 type	v3 type
xMin	Single	Double
yMin	Single	Double
xMax	Single	Double
yMax	Single	Double
sScale	Single	Double

For ByRef parameters, use a variable with the exact v3 type. Preserve array dimensions and the documented indexing. Moving from v2 to v3 explains conversion and legacy file-storage rules.

Example

Setup

Add an Avax.cAvax Windows Forms control named `cAvax1`. Paste the procedures into the form class. Example setup and supporting files explains references, initialization and file paths.

Add these Windows Forms controls or components to the form. Fields referenced by Handles clauses must be declared WithEvents, as the Visual Studio designer normally does.

Name	Type
Button1	Button

VB.NET example

```
Private Sub Form1_Load(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs) Handles
    MyBase.Load
    Call cAvax1.StartAvax()
    Call cAvax1.Add_Line(0, 5, 0, 10, 5, 0, Nothing, 1, 1, -1, -1)
    Call cAvax1.Add_Line(5, 0, 0, 5, 10, 0, Nothing, 1, 1, -1, -1)
    Call cAvax1.Add_Circle(5, 5, 0, 3, Nothing, 5, -1, -1, -1)
End Sub
Private Sub Form1_FormClosed(ByVal eventSender As System.Object, ByVal eventArgs As
    System.Windows.Forms.FormClosedEventArgs) Handles Me.FormClosed
    Call cAvax1.EndAvax()
End Sub
Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
    Button1.Click
    Dim sGIFFile As String = String.Empty
    Dim lWidth As Integer
    Dim lHeight As Integer
    Dim fSelected As Boolean
    Dim xMin As Double
    Dim yMin As Double
    Dim xMax As Double
    Dim yMax As Double
    Dim sScale As Double
    Dim Answer As Boolean
    sGIFFile = My.Application.Info.DirectoryPath & "\..\temp.gif"
    lWidth = 120
    lHeight = 120
    sScale = 1
    Answer = cAvax1.PrintToGIF(sGIFFile, lWidth, lHeight, fSelected, xMin, yMin, xMax, yMax, sScale)
End Sub
```

PrintTohDC

cAvax method

Purpose

Prints the drawing into a hDC

VB.NET declarations

```
Function PrintTohDC(ByRef hDCToPrint As Integer, ByVal lWidth As Integer, ByVal lHeight As Integer) As
    Boolean

Function PrintTohDC(ByRef hDCToPrint As Integer, ByVal lWidth As Integer, ByVal lHeight As Integer, ByRef
    fSelected As Boolean, ByRef lLeft As Integer, ByRef lTop As Integer, ByRef xMin As Double, ByRef yMin As
    Double, ByRef xMax As Double, ByRef yMax As Double, ByRef sScale As Double) As Boolean
```

Parameters

Name	Type and passing	Meaning
hDCToPrint	ByRef hDCToPrint As Integer	Caller-owned valid output HDC. IntPtr is native-sized and recommended for x64; the legacy Integer overload remains supported. Pair Graphics.GetHdc with ReleaseHdc in Finally. The DLL preserves this handle.
lWidth	ByVal lWidth As Integer	Integer output width in the HDC logical coordinate system (pixels in the usual MM_TEXT mode). Zero derives width from height and whole-drawing aspect ratio. Both zero fail. Modern contexts support Integer dimensions subject to GDI limits; classic WMF recording retains 1..32767.
lHeight	ByVal lHeight As Integer	Integer output height, with the same rules as lWidth. A zero height is calculated from positive width. Dimensions remain ByVal.
fSelected	ByRef fSelected As Boolean	True draws only selected objects; an empty selection fails. Automatic bounds still fit the complete drawing as in R2. The flag is unchanged.
lLeft	ByRef lLeft As Integer	Integer offset of the output rectangle in the caller HDC coordinate space. Negative offsets are allowed when representable. The caller margin is unchanged.
lTop	ByRef lTop As Integer	Integer top offset of the output rectangle. Caller clip and mapping state are retained. Classic WMF recording requires signed 16-bit margins.
xMin	ByRef xMin As Double	Double left drawing coordinate. Four zeros request complete drawing bounds; four minus-ones request the current view. Normalized actual bounds commit only on success.
yMin	ByRef yMin As Double	Double bottom drawing coordinate. Nonfinite or unrepresentable bounds fail before touching output.
xMax	ByRef xMax As Double	Double right drawing coordinate. Explicit reversed corners normalize. Exactly zero spans receive a usable one-unit frame.
yMax	ByRef yMax As Double	Double top drawing coordinate; corrects the CHM table reference to an x coordinate. ByRef bounds are unchanged after preparation/validation failure.
sScale	ByRef sScale As Double	Double geometric scale about the output top-left. Nonpositive finite values normalize to 1 and commit ByRef on success; NaN/infinity or an unrepresentable rendering scale fail.

Use the declaration for the overload you call. ByRef arguments can receive outputs; declare variables and arrays with the exact indicated type. Optional defaults are shown in the declaration.

Return value

Boolean - True on success, false on error.

Remarks

If all the xMin, yMin, xMax and yMax are zero, then the whole drawing will be printed

Numeric changes from v2

Update the following variables, arrays or event-data receivers when porting the corresponding v2 call. The declarations above show the current overloads.

Parameter or value	v2 type	v3 type
xMin	Single	Double
yMin	Single	Double
xMax	Single	Double
yMax	Single	Double
sScale	Single	Double

For ByRef parameters, use a variable with the exact v3 type. Preserve array dimensions and the documented indexing. Moving from v2 to v3 explains conversion and legacy file-storage rules.

Example

Setup

Add an Avax.cAvax Windows Forms control named `cAvax1`. Paste the procedures into the form class. Example setup and supporting files explains references, initialization and file paths.

Add these Windows Forms controls or components to the form. Fields referenced by Handles clauses must be declared WithEvents, as the Visual Studio designer normally does.

Name	Type
Command1	Button
Picture1	PictureBox

VB.NET example

```
Private Sub Form1_Load(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs) Handles
    MyBase.Load
    Command1.Text = "PrintToHDC"
    cAvax1.StartAvax()
    Call CAvax1.Add_Line(0, 5, 0, 10, 5, 0, Nothing, 1, 1, -1, -1)
    Call CAvax1.Add_Line(5, 0, 0, 5, 10, 0, Nothing, 1, 1, -1, -1)
    Call CAvax1.Add_Circle(5, 5, 0, 3, Nothing, 5, -1, -1, -1)
End Sub
Private Sub Form1_FormClosed(ByVal eventSender As System.Object, ByVal eventArgs As
    System.Windows.Forms.FormClosedEventArgs) Handles Me.FormClosed
    cAvax1.EndAvax()
End Sub
Dim hDCToPrint As IntPtr
Private Sub Command1_Click(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs)
    Handles Command1.Click
    Dim lWidth As Integer
    Dim lHeight As Integer
    Dim fSelected As Boolean
    Dim lLeft As Integer
    Dim lTop As Integer
    Dim xMin As Double
    Dim yMin As Double
    Dim xMax As Double
    Dim yMax As Double
    Dim sScale As Double
    Dim Answer As Boolean
    Dim previousImage = Picture1.Image
```

```
    Dim image As New Bitmap(Picture1.ClientRectangle.Width, Picture1.ClientRectangle.Height)
    Try
        Using gr As Graphics = Graphics.FromImage(image)
            hDCToPrint = gr.GetHdc()
            Try
                lWidth = Picture1.ClientRectangle.Width
                lHeight = Picture1.ClientRectangle.Height
                Picture1.BackColor = cAvax1.BackColor
                Answer = cAvax1.PrintToHDC(hDCToPrint, lWidth, lHeight, fSelected, lLeft, lTop, xMin,
    yMin, xMax, yMax, sScale)
            Finally
                gr.ReleaseHdc(hDCToPrint)
                hDCToPrint = IntPtr.Zero
            End Try
        End Using
        Picture1.Image = image
        image = Nothing
        If previousImage IsNot Nothing Then previousImage.Dispose()
    Finally
        If image IsNot Nothing Then image.Dispose()
    End Try
End Sub
```

PrintToJPG

cAvax method

Purpose

Exports the drawing into a .JPG file

VB.NET declarations

```
Function PrintToJPG(ByRef sJPGFile As String, ByVal lWidth As Integer, ByVal lHeight As Integer, ByRef
    fSelected As Boolean, ByRef xMin As Double, ByRef yMin As Double, ByRef xMax As Double, ByRef yMax As
    Double, ByRef sScale As Double) As Boolean

Function PrintToJPG(ByRef sJPGFile As String, ByVal lWidth As Integer, ByVal lHeight As Integer) As
    Boolean
```

Parameters

Name	Type and passing	Meaning
sJPGFile	ByRef sJPGFile As String	String destination path, unchanged ByRef. Relative paths resolve before callbacks. The actual JPG encoder is used regardless of extension; preparation or commit failure preserves existing output.
lWidth	ByVal lWidth As Integer	Integer image width in pixels, exactly as written in the JPG header. Zero derives width from positive height and whole-drawing aspect ratio. Both zero fail; negative/overflow/allocation failures preserve existing output. Unlike the old implementation this is not width+1. The JPG format has 16-bit dimensions, so values above 65535 fail.
lHeight	ByVal lHeight As Integer	Integer image height in pixels, with the same rules as lWidth. Width and height stay ByVal. Fit includes the last pixel centre so bottom/right boundary lines remain visible. The same 65535 format limit applies.
fSelected	ByRef fSelected As Boolean	True exports only selected objects and fails if the selection is empty. Automatic extents still use the complete drawing as in R2. This ByRef flag is unchanged.
xMin	ByRef xMin As Double	Double left coordinate. Four zeros mean complete drawing extents; four minus-ones mean the current nominal view. Actual normalized extents are returned only on success.
yMin	ByRef yMin As Double	Double bottom coordinate. All extent coordinates must be finite and produce a representable rendering scale.
xMax	ByRef xMax As Double	Double right coordinate. Explicit reversed bounds normalize; exactly zero spans receive a usable usable nonzero frame (adjacent finite Double coordinates at large origins).
yMax	ByRef yMax As Double	Double top coordinate; corrects the CHM table's reference to an x coordinate. Failed preparation/file commit leaves all extent ByRefs unchanged.
sScale	ByRef sScale As Double	Double geometric multiplier about the output top-left. Finite values <=0 become 1, committed only on success. NaN/infinity/unrepresentable scale fail; scaling larger than one may intentionally clip geometry.

Use the declaration for the overload you call. ByRef arguments can receive outputs; declare variables and arrays with the exact indicated type. Optional defaults are shown in the declaration.

Return value

Boolean - True on success, false on error.

Remarks

If all the xMin, yMin, xMax, yMax are zero, then the whole drawing will be printed

Numeric changes from v2

Update the following variables, arrays or event-data receivers when porting the corresponding v2 call. The declarations above show the current overloads.

Parameter or value	v2 type	v3 type
xMin	Single	Double
yMin	Single	Double
xMax	Single	Double
yMax	Single	Double
sScale	Single	Double

For ByRef parameters, use a variable with the exact v3 type. Preserve array dimensions and the documented indexing. Moving from v2 to v3 explains conversion and legacy file-storage rules.

Example

Setup

Add an Avax.cAvax Windows Forms control named `cAvax1`. Paste the procedures into the form class. Example setup and supporting files explains references, initialization and file paths.

Add these Windows Forms controls or components to the form. Fields referenced by Handles clauses must be declared WithEvents, as the Visual Studio designer normally does.

Name	Type
Command1	Button

VB.NET example

```
Private Sub Form1_Load(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs) Handles
    MyBase.Load
    cAvax1.StartAvax()
    Call CAvax1.Add_Text("AVAX Vector ActiveX", 0, 0, 0, Nothing, 5, 1, 45, -1, -1, -1, -1, False, False,
    False, False, -1)
End Sub
Private Sub Form1_FormClosed(ByVal eventSender As System.Object, ByVal eventArgs As
    System.Windows.Forms.FormClosedEventArgs) Handles Me.FormClosed
    cAvax1.EndAvax()
End Sub
Private Sub Command1_Click(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs)
    Handles Command1.Click
    Dim sJPGFile As String = String.Empty
    Dim lWidth As Integer
    Dim lHeight As Integer
    Dim Answer As Boolean
    sJPGFile = My.Application.Info.DirectoryPath & "\..\temp.jpg"
    lWidth = 500
    lHeight = 500
    CAvax1.SetAvaxProperty(Avax.AvaxProp.JPGCompressQuality, 100)
    Answer = CAvax1.PrintToJPG(sJPGFile, lWidth, lHeight)
End Sub
```

PrintToPIC

cAvax method

Purpose

Exports the drawing into a .PIC file

VB.NET declarations

```
Function PrintToPIC(ByRef sPICFile As String) As Boolean

Function PrintToPIC(ByRef sPICFile As String, ByRef xMin As Double, ByRef yMin As Double, ByRef xMax As
    Double, ByRef yMax As Double) As Boolean
```

Parameters

Name	Type and passing	Meaning
sPICFile	ByRef sPICFile As String	String destination, unchanged ByRef. Resolved before callbacks; complete output commits atomically. Failures retain existing file bytes.
xMin	ByRef xMin As Double	Double ByRef. Four zeros use whole-drawing bounds; four minus-ones use the current view. Normalized resolved minimum X is returned only after successful commit.
yMin	ByRef yMin As Double	Double ByRef. Resolved minimum Y becomes the lower-left output origin. Failed export leaves the input unchanged.
xMax	ByRef xMax As Double	Double ByRef. Explicit reversed X corners normalize; an exactly zero span expands by 0.5 on each side. Nonfinite or unrepresentable ranges fail.
yMax	ByRef yMax As Double	Double ByRef. Explicit reversed Y corners normalize; an exactly zero span expands by 0.5 on each side. Nonzero tiny spans are retained. Returned bounds describe the crop in drawing units.

Use the declaration for the overload you call. ByRef arguments can receive outputs; declare variables and arrays with the exact indicated type. Optional defaults are shown in the declaration.

Return value

Boolean - True on success, false on error.

Remarks

If all the xMin, yMin, xMax, yMax are zero, then the whole drawing will be printed

Numeric changes from v2

Update the following variables, arrays or event-data receivers when porting the corresponding v2 call. The declarations above show the current overloads.

Parameter or value	v2 type	v3 type
xMin	Single	Double
yMin	Single	Double
xMax	Single	Double
yMax	Single	Double

For ByRef parameters, use a variable with the exact v3 type. Preserve array dimensions and the documented indexing. Moving from v2 to v3 explains conversion and legacy file-storage rules.

Example

Setup

Add an Avax.cAvax Windows Forms control named `cAvax1`. Paste the procedures into the form class. Example setup and supporting files explains references, initialization and file paths.

Add these Windows Forms controls or components to the form. Fields referenced by Handles clauses must be declared WithEvents, as the Visual Studio designer normally does.

Name	Type
Command1	Button

VB.NET example

```
Private Sub Form1_Load(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs) Handles
    MyBase.Load
    Command1.Text = "PrintToPIC"
    cAvax1.StartAvax()
    Call cAvax1.Add_Line(0, 5, 0, 10, 5, 0, Nothing, 1, 1, -1, -1)
    Call cAvax1.Add_Line(5, 0, 0, 5, 10, 0, Nothing, 1, 1, -1, -1)
    Call cAvax1.Add_Circle(5, 5, 0, 3, Nothing, 5, -1, -1, -1)
End Sub
Private Sub Form1_FormClosed(ByVal eventSender As System.Object, ByVal eventArgs As
    System.Windows.Forms.FormClosedEventArgs) Handles Me.FormClosed
    cAvax1.EndAvax()
End Sub
Private Sub Command1_Click(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs)
    Handles Command1.Click
    Dim sPICFile As String = String.Empty
    Dim xMin As Double
    Dim yMin As Double
    Dim xMax As Double
    Dim yMax As Double
    Dim Answer As Boolean
    sPICFile = My.Application.Info.DirectoryPath & "\..\temp.pic"
    Answer = cAvax1.PrintToPIC(sPICFile, xMin, yMin, xMax, yMax)
End Sub
```

PrintToPLT

cAvax method

Purpose

Exports the drawing into a .PLT file

VB.NET declarations

```
Function PrintToPLT(ByRef sPLTFile As String, ByRef xMin As Double, ByRef yMin As Double, ByRef xMax As
    Double, ByRef yMax As Double) As Boolean

Function PrintToPLT(ByRef sPLTFile As String) As Boolean
```

Parameters

Name	Type and passing	Meaning
sPLTFile	ByRef sPLTFile As String	String destination, unchanged ByRef. Resolved before callbacks; complete output commits atomically. Failures retain existing file bytes.
xMin	ByRef xMin As Double	Double ByRef. Four zeros use whole-drawing bounds; four minus-ones use the current view. Normalized resolved minimum X is returned only after successful commit.
yMin	ByRef yMin As Double	Double ByRef. Resolved minimum Y becomes the lower-left output origin. Failed export leaves the input unchanged.
xMax	ByRef xMax As Double	Double ByRef. Explicit reversed X corners normalize; an exactly zero span expands by 0.5 on each side. Nonfinite or unrepresentable ranges fail.
yMax	ByRef yMax As Double	Double ByRef. Explicit reversed Y corners normalize; an exactly zero span expands by 0.5 on each side. Nonzero tiny spans are retained. Returned bounds describe the crop in drawing units.

Use the declaration for the overload you call. ByRef arguments can receive outputs; declare variables and arrays with the exact indicated type. Optional defaults are shown in the declaration.

Return value

Boolean - True on success, false on error

Remarks

If all the xMin, yMin, xMax, yMax are zero, then the whole drawing will be printed

Numeric changes from v2

Update the following variables, arrays or event-data receivers when porting the corresponding v2 call. The declarations above show the current overloads.

Parameter or value	v2 type	v3 type
xMin	Single	Double
yMin	Single	Double
xMax	Single	Double
yMax	Single	Double

For ByRef parameters, use a variable with the exact v3 type. Preserve array dimensions and the documented indexing. Moving from v2 to v3 explains conversion and legacy file-storage rules.

Example

Setup

Add an Avax.cAvax Windows Forms control named `cAvax1`. Paste the procedures into the form class. Example setup and supporting files explains references, initialization and file paths.

Add these Windows Forms controls or components to the form. Fields referenced by Handles clauses must be declared WithEvents, as the Visual Studio designer normally does.

Name	Type
Command1	Button

VB.NET example

```
Private Sub Form1_Load(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs) Handles
    MyBase.Load
    Command1.Text = "PrintToPLT"
    cAvax1.StartAvax()
    Call CAvax1.Add_Line(0, 5, 0, 10, 5, 0, Nothing, 1, 1, -1, -1)
    Call CAvax1.Add_Line(5, 0, 0, 5, 10, 0, Nothing, 1, 1, -1, -1)
    Call CAvax1.Add_Circle(5, 5, 0, 3, Nothing, 5, -1, -1, -1)
End Sub
Private Sub Form1_FormClosed(ByVal eventSender As System.Object, ByVal eventArgs As
    System.Windows.Forms.FormClosedEventArgs) Handles Me.FormClosed
    cAvax1.EndAvax()
End Sub
Private Sub Command1_Click(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs)
    Handles Command1.Click
    Dim sPLTFile As String = String.Empty
    Dim xMin As Double
    Dim yMin As Double
    Dim xMax As Double
    Dim yMax As Double
    Dim Answer As Boolean
    sPLTFile = My.Application.Info.DirectoryPath & "\temp.plt"
    Call CAvax1.SetAvaxProperty(Avax.AvaxProp.PrintAllOn, True)
    Call CAvax1.SetAvaxProperty(Avax.AvaxProp.PrintUnit, MM, 10)
    Answer = cAvax1.PrintToPLT(sPLTFile, xMin, yMin, xMax, yMax)
End Sub
```

PrintToPNG

cAvax method

Purpose

Exports the drawing into a .PNG file

VB.NET declarations

```
Function PrintToPNG(ByRef sPNGFile As String, ByVal lWidth As Integer, ByVal lHeight As Integer) As
    Boolean

Function PrintToPNG(ByRef sPNGFile As String, ByVal lWidth As Integer, ByVal lHeight As Integer, ByRef
    fSelected As Boolean, ByRef xMin As Double, ByRef yMin As Double, ByRef xMax As Double, ByRef yMax As
    Double, ByRef sScale As Double) As Boolean

Function PrintToPNG(ByRef sPNGFile As String, ByVal lWidth As Integer, ByVal lHeight As Integer, ByRef
    fSelected As Boolean, ByRef xMin As Double, ByRef yMin As Double, ByRef xMax As Double, ByRef yMax As
    Double, ByRef sScale As Double, ByVal bTransparent As Boolean) As Boolean
```

Parameters

Name	Type and passing	Meaning
sPNGFile	ByRef sPNGFile As String	String destination path, unchanged ByRef. Relative paths resolve before callbacks. The actual PNG encoder is used regardless of extension; preparation or commit failure preserves existing output.
lWidth	ByVal lWidth As Integer	Integer image width in pixels, exactly as written in the PNG header. Zero derives width from positive height and whole-drawing aspect ratio. Both zero fail; negative/overflow/allocation failures preserve existing output. Unlike the old implementation this is not width+1.
lHeight	ByVal lHeight As Integer	Integer image height in pixels, with the same rules as lWidth. Width and height stay ByVal. Fit includes the last pixel centre so bottom/right boundary lines remain visible.
fSelected	ByRef fSelected As Boolean	True exports only selected objects and fails if the selection is empty. Automatic extents still use the complete drawing as in R2. This ByRef flag is unchanged.
xMin	ByRef xMin As Double	Double left coordinate. Four zeros mean complete drawing extents; four minus-ones mean the current nominal view. Actual normalized extents are returned only on success.
yMin	ByRef yMin As Double	Double bottom coordinate. All extent coordinates must be finite and produce a representable rendering scale.
xMax	ByRef xMax As Double	Double right coordinate. Explicit reversed bounds normalize; exactly zero spans receive a usable usable nonzero frame (adjacent finite Double coordinates at large origins).
yMax	ByRef yMax As Double	Double top coordinate; corrects the CHM table's reference to an x coordinate. Failed preparation/file commit leaves all extent ByRefs unchanged.
sScale	ByRef sScale As Double	Double geometric multiplier about the output top-left. Finite values <=0 become 1, committed only on success. NaN/infinity/unrepresentable scale fail; scaling larger than one may intentionally clip geometry.
bTransparent	ByVal bTransparent As Boolean	Boolean ByVal background color-key transparency. False is the default for Integer/nine-argument overloads; True removes all rendered pixels matching the captured BackColor, including pixels in embedded images. Conversion completes in memory before atomic saving.

Use the declaration for the overload you call. ByRef arguments can receive outputs; declare variables and arrays with the exact indicated type. Optional defaults are shown in the declaration.

Return value

Boolean - True on success, false on error.

Numeric changes from v2

Update the following variables, arrays or event-data receivers when porting the corresponding v2 call. The declarations above show the current overloads.

Parameter or value	v2 type	v3 type
xMin	Single	Double
yMin	Single	Double
xMax	Single	Double
yMax	Single	Double
sScale	Single	Double

For ByRef parameters, use a variable with the exact v3 type. Preserve array dimensions and the documented indexing. Moving from v2 to v3 explains conversion and legacy file-storage rules.

Example

Setup

Add an Avax.cAvax Windows Forms control named `cAvax1`. Paste the procedures into the form class. Example setup and supporting files explains references, initialization and file paths.

Add these Windows Forms controls or components to the form. Fields referenced by Handles clauses must be declared WithEvents, as the Visual Studio designer normally does.

Name	Type
Button1	Button

VB.NET example

```
Private Sub Form1_Load(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs) Handles
    MyBase.Load
    Call cAvax1.StartAvax()
    Call CAvax1.Add_Line(0, 5, 0, 10, 5, 0, Nothing, 1, 1, -1, -1)
    Call CAvax1.Add_Line(5, 0, 0, 5, 10, 0, Nothing, 1, 1, -1, -1)
    Call CAvax1.Add_Circle(5, 5, 0, 3, Nothing, 5, -1, -1, -1)
End Sub
Private Sub Form1_FormClosed(ByVal eventSender As System.Object, ByVal eventArgs As
    System.Windows.Forms.FormClosedEventArgs) Handles Me.FormClosed
    Call CAvax1.EndAvax()
End Sub
Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
    Button1.Click
    Dim sPNGFile As String = String.Empty
    Dim lWidth As Integer
    Dim lHeight As Integer
    Dim fSelected As Boolean
    Dim xMin As Double
    Dim yMin As Double
    Dim xMax As Double
    Dim yMax As Double
    Dim sScale As Double
    Dim Answer As Boolean
    sPNGFile = My.Application.Info.DirectoryPath & "\..\temp.png"
    lWidth = 120
    lHeight = 120
    sScale = 1
    Answer = CAvax1.PrintToPNG(sPNGFile, lWidth, lHeight, fSelected, xMin, yMin, xMax, yMax, sScale)
End Sub
```

PrintToPrinter

cAvax method

Purpose

Prints the drawing into a printer

VB.NET declarations

```
Function PrintToPrinter(ByRef sPrinterName As String, ByRef avPaperSize As AvaxPaperSize, ByRef UnitMM As
    Double) As Boolean

Function PrintToPrinter(ByRef sPrinterName As String, ByRef avPaperSize As AvaxPaperSize, ByRef UnitMM As
    Double, ByRef sBmpFiles As String()) As Boolean

Function PrintToPrinter(ByRef sPrinterName As String, ByRef avPaperSize As AvaxPaperSize, ByRef UnitMM As
    Double, ByRef xAvaxMin As Double, ByRef yAvaxMin As Double, ByRef xAvaxMax As Double, ByRef yAvaxMax As
    Double, ByRef fSelected As Boolean, ByRef fLandScape As Boolean, ByRef fExtends As Boolean, ByRef
    fPrinterDialogOn As Boolean) As Boolean

Function PrintToPrinter(ByRef sPrinterName As String, ByRef avPaperSize As AvaxPaperSize, ByRef UnitMM As
    Double, ByRef xAvaxMin As Double, ByRef yAvaxMin As Double, ByRef xAvaxMax As Double, ByRef yAvaxMax As
    Double, ByRef fSelected As Boolean, ByRef fLandScape As Boolean, ByRef fExtends As Boolean, ByRef
    fPrinterDialogOn As Boolean, ByRef sBmpFiles As String()) As Boolean
```

Parameters

Name	Type and passing	Meaning
sPrinterName	ByRef sPrinterName As String	String ByRef. Empty chooses the default printer. Also supports the original ChrW(1)-separated COUNTPAGES:, PRINTIMAGE: and PRINT: protocols described below. Written only on complete success.
avPaperSize	ByRef avPaperSize As AvaxPaperSize	AvaxPaperSize ByRef. All declared values use nominal physical dimensions; epsUser uses the custom millimetre properties. Unknown values become epsA4 after success.
UnitMM	ByRef UnitMM As Double	Double ByRef, millimetres per drawing unit. Finite nonpositive values request fit; successful calls return the resolved scale.
sBmpFiles	ByRef sBmpFiles As String()	String() ByRef. PRINTIMAGE returns one-based paths to owned completed BMP previews. Slot zero is unused; masked output preserves the old slot-one alias. Failure and non-image requests preserve the original array/reference.
xAvaxMin	ByRef xAvaxMin As Double	Double ByRef crop input, unchanged. Four zeros mean whole drawing; four minus ones mean current view; otherwise the four coordinates define an internally normalized crop.
yAvaxMin	ByRef yAvaxMin As Double	Is the y coordinate of the left bottom drawing extents that will be printed
xAvaxMax	ByRef xAvaxMax As Double	Is the x coordinate of the right top drawing extents that will be printed
yAvaxMax	ByRef yAvaxMax As Double	Is the y coordinate of the right top drawing extents that will be printed
fSelected	ByRef fSelected As Boolean	A Boolean expression that if it is true then only the selected objects are printed
fLandScape	ByRef fLandScape As Boolean	A Boolean expression that if it is true then the printer will print in landscape mode
fExtends	ByRef fExtends As Boolean	Boolean ByRef, unchanged. True fits a Double printable page; False uses UnitMM except that nonpositive UnitMM requests fit.
fPrinterDialogOn	ByRef fPrinterDialogOn As Boolean	Boolean ByRef, unchanged. True opens the normal printer dialog for a real print request; count/image requests stay noninteractive.

Use the declaration for the overload you call. ByRef arguments can receive outputs; declare variables and arrays with the exact indicated type. Optional defaults are shown in the declaration.

Return value

Boolean - True on success, false on error.

Remarks

This method can show a printer dialog or send output to a physical printer. Run its example only when you intend to print. The COUNTPAGES: and PRINTIMAGE: protocols provide page counting and bitmap previews without printing.

Numeric changes from v2

Update the following variables, arrays or event-data receivers when porting the corresponding v2 call. The declarations above show the current overloads.

Parameter or value	v2 type	v3 type
UnitMM	Single	Double
xAvaxMin	Single	Double
yAvaxMin	Single	Double
xAvaxMax	Single	Double
yAvaxMax	Single	Double

For ByRef parameters, use a variable with the exact v3 type. Preserve array dimensions and the documented indexing. Moving from v2 to v3 explains conversion and legacy file-storage rules.

Example

Setup

Add an Avax.cAvax Windows Forms control named `cAvax1`. Paste the procedures into the form class. Example setup and supporting files explains references, initialization and file paths.

Add these Windows Forms controls or components to the form. Fields referenced by Handles clauses must be declared WithEvents, as the Visual Studio designer normally does.

Name	Type
Command1	Button

VB.NET example

```
Private Sub Form1_Load(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs) Handles
    MyBase.Load
    Command1.Text = "PrintToPrinter"
    cAvax1.StartAvax()
    Call cAvax1.Add_Line(0, 5, 0, 10, 5, 0, Nothing, 1, 1, -1, -1)
    Call cAvax1.Add_Line(5, 0, 0, 5, 10, 0, Nothing, 1, 1, -1, -1)
    Call cAvax1.Add_Circle(5, 5, 0, 3, Nothing, 5, -1, -1, -1)
    Call cAvax1.Add_Text("Avax Vector ActiveX", 0, 11, 0, Nothing, 4, 1, 0, -1, -1, -1, -1, False, False,
    False, False, -1)
End Sub
Private Sub Form1_FormClosed(ByVal eventSender As System.Object, ByVal eventArgs As
    System.Windows.Forms.FormClosedEventArgs) Handles Me.FormClosed
    cAvax1.EndAvax()
End Sub
Private Sub Command1_Click(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs)
    Handles Command1.Click
    Dim sPrinterName As String = String.Empty
    Dim avPaperSize As Avax.AvaxPaperSize
    Dim UnitMM As Double
    Dim xAvaxMin As Double
```

```
    Dim yAvaxMin As Double
    Dim xAvaxMax As Double
    Dim yAvaxMax As Double
    Dim fSelected As Boolean
    Dim fLandScape As Boolean
    Dim fExtends As Boolean
    Dim fPrinterDialogOn As Boolean
    Dim Answer As Boolean
    On Error Resume Next
    sPrinterName = "" ' Default Printer
    avPaperSize = Avax.AvaxPaperSize.epsA4
    fLandScape = True
    fExtends = True
    fPrinterDialogOn = False
    Answer = CAvax1.PrintToPrinter(sPrinterName, avPaperSize, UnitMM, xAvaxMin, yAvaxMin, xAvaxMax,
    yAvaxMax, fSelected, fLandScape, fExtends, fPrinterDialogOn)
End Sub
```

PrintToRTF

cAvax method

Purpose

Exports a Windows metafile picture inside an RTF document; the full overload can emit only the picture group.

VB.NET declarations

```
Function PrintToRTF(ByRef sRTFFile As String, ByVal lWidth As Integer, ByVal lHeight As Integer) As
    Boolean

Function PrintToRTF(ByRef sRTFFile As String, ByVal lWidth As Integer, ByVal lHeight As Integer, ByRef
    fSelected As Boolean, ByRef xMin As Double, ByRef yMin As Double, ByRef xMax As Double, ByRef yMax As
    Double, ByRef sScale As Double, ByRef fRTFHeader As Boolean) As Boolean
```

Parameters

Name	Type and passing	Meaning
sRTFFile	ByRef sRTFFile As String	String destination filename, unchanged ByRef. Full RTF or picture-fragment bytes are written regardless of extension. The absolute destination is captured before callbacks; success replaces atomically and failure preserves existing bytes.
lWidth	ByVal lWidth As Integer	Integer printed width in logical WMF units, 1440 per inch. Range 1..32767, or zero to derive it from whole-drawing aspect ratio and positive height. The RTF physical goal equals the inclusive width+1 frame, capped at 32767. This fixes the old DPI-dependent PictureBox conversion; it is not a pixel count.
lHeight	ByVal lHeight As Integer	Integer printed height in the same logical WMF units, or zero to derive it from whole-drawing aspect ratio and positive width. Both zero fail; the same format limit and inclusive physical frame apply. Dimensions remain ByVal.
fSelected	ByRef fSelected As Boolean	True prints only selected objects. An empty selection returns False. All-zero crop arguments still use complete drawing bounds, as in R2.
xMin	ByRef xMin As Double	Double left drawing coordinate. Four zeros request whole-drawing bounds; four minus-ones request the current view. Actual normalized bounds are returned only on success.
yMin	ByRef yMin As Double	Double bottom drawing coordinate. Nonfinite bounds fail. Exactly zero drawing width or height receives a one-unit frame so lines/points can export.
xMax	ByRef xMax As Double	Double right drawing coordinate. Explicit reversed corners are normalized; output is clipped to the requested frame.
yMax	ByRef yMax As Double	Double top drawing coordinate. This corrects the original CHM table reference to an x coordinate. Returned ByRef only on success.
sScale	ByRef sScale As Double	Double geometric plot scale about the output top-left, clipped to the frame. It does not resize the picture goal. Nonpositive finite values normalize to 1 on success; NaN/infinity or unrepresentable scale fail without changing caller arguments.
fRTFHeader	ByRef fRTFHeader As Boolean	Boolean ByRef, unchanged. True writes a complete RTF document with A4 page settings and current font metadata. False writes only a {\pict\wmetafile8 ...} group suitable for embedding. The Integer overload defaults to True.

Use the declaration for the overload you call. ByRef arguments can receive outputs; declare variables and arrays with the exact indicated type. Optional defaults are shown in the declaration.

Return value

Boolean - True on success, false on error.

Remarks

The file contains a wmetafile8 picture. Simple geometry uses vector records; text, fills and other complex graphics can use bounded bitmap records within that metafile. The drawing is represented as a picture rather than editable RTF text. The short overload includes an RTF header; fRTFHeader=False emits only the picture group.

Width and height use the PrintToWMF physical frame: 1440 logical units per inch, with each dimension limited to 32767. The stored inclusive frame is the requested size plus one unit, capped at 32767. One zero dimension derives the other from the whole-drawing aspect ratio. Four zero crop coordinates mean the whole drawing; four -1 values mean the current view. Reversed corners are ordered. sScale changes the contents within the fixed output frame, not the RTF picture size. The RTF header uses A4 size/margins and the current font name/charset.

A started, enabled control is required. PrintingPage reports the one output page unless events are paused. AbortPrinting or a changed document/selection/lifecycle can cancel the export. The destination and ByRef bounds/scale are replaced only after complete successful generation. Invalid input, cancellation or file failure returns False and retains an existing destination.

Numeric changes from v2

Update the following variables, arrays or event-data receivers when porting the corresponding v2 call. The declarations above show the current overloads.

Parameter or value	v2 type	v3 type
xMin	Single	Double
yMin	Single	Double
xMax	Single	Double
yMax	Single	Double
sScale	Single	Double

For ByRef parameters, use a variable with the exact v3 type. Preserve array dimensions and the documented indexing. Moving from v2 to v3 explains conversion and legacy file-storage rules.

Example

Setup

Add an Avax.cAvax Windows Forms control named `cAvax1`. Paste the procedures into the form class. Example setup and supporting files explains references, initialization and file paths.

Add these Windows Forms controls or components to the form. Fields referenced by Handles clauses must be declared WithEvents, as the Visual Studio designer normally does.

Name	Type
Command1	Button

VB.NET example

```
Private Sub Form1_Load(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs) Handles
    MyBase.Load
    Command1.Text = "PrintToRTF"
    cAvax1.StartAvax()
    Call CAvax1.Add_Line(0, 5, 0, 10, 5, 0, Nothing, 1, 1, -1, -1)
    Call CAvax1.Add_Line(5, 0, 0, 5, 10, 0, Nothing, 1, 1, -1, -1)
    Call CAvax1.Add_Circle(5, 5, 0, 3, Nothing, 5, -1, -1, -1)
End Sub
Private Sub Form1_FormClosed(ByVal eventSender As System.Object, ByVal eventArgs As
    System.Windows.Forms.FormClosedEventArgs) Handles Me.FormClosed
    cAvax1.EndAvax()
End Sub
Private Sub Command1_Click(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs)
```

```
    Handles Command1.Click
    Dim sRTFFile As String = String.Empty
    Dim lWidth As Integer
    Dim lHeight As Integer
    Dim fSelected As Boolean
    Dim xMin As Double
    Dim yMin As Double
    Dim xMax As Double
    Dim yMax As Double
    Dim sScale As Double
    Dim fRTFHeader As Boolean
    Dim Answer As Boolean
    sRTFFile = My.Application.Info.DirectoryPath & "\..\temp.rtf"
    lWidth = 1200
    lHeight = 1200
    fRTFHeader = True
    Answer = cAvax1.PrintToRTF(sRTFFile, lWidth, lHeight, fSelected, xMin, yMin, xMax, yMax, sScale,
    fRTFHeader)
End Sub
```

PrintToSVG

cAvax method

Purpose

Exports the drawing into a .SVG file

VB.NET declarations

```
Function PrintToSVG(ByRef sSVGFile As String, ByVal WidthCm As Double, ByVal HeightCm As Double) As
    Boolean

Function PrintToSVG(ByRef sSVGFile As String, ByVal WidthCm As Double, ByVal HeightCm As Double, ByRef
    fSelected As Boolean, ByRef xMin As Double, ByRef yMin As Double, ByRef xMax As Double, ByRef yMax As
    Double) As Boolean
```

Parameters

Name	Type and passing	Meaning
sSVGFile	ByRef sSVGFile As String	String destination, unchanged ByRef. Absolute path is captured before provider callbacks. Successful complete output replaces atomically; failures retain old bytes.
WidthCm	ByVal WidthCm As Double	Double physical width in centimeters, ByVal. Zero derives width from positive HeightCm and resolved crop aspect. Must be finite, nonnegative and representable by supported SVG consumers; both dimensions zero fail.
HeightCm	ByVal HeightCm As Double	Double physical height in centimeters, ByVal. Zero derives height from positive WidthCm and resolved crop aspect. Default SVG aspect fitting retains geometry proportions if both explicit dimensions differ from the crop ratio.
fSelected	ByRef fSelected As Boolean	Boolean ByRef, unchanged. True exports only selected objects while retaining the requested full-drawing/crop frame. Empty selection fails; printing layer visibility still applies.
xMin	ByRef xMin As Double	Double lower-left X input, unchanged ByRef. Four zeros use whole-drawing bounds. The viewBox adds ten-percent border; reversed corners normalize internally.
yMin	ByRef yMin As Double	Double lower-left Y input, unchanged ByRef. OriginTopLeft_p determines SVG Y orientation consistently for both frame and geometry.
xMax	ByRef xMax As Double	Double upper-right X input, unchanged ByRef. A zero-width crop expands one unit to its positive side; nonfinite or unrepresentable extents fail.
yMax	ByRef yMax As Double	Double upper-right Y input, unchanged ByRef. A zero-height crop expands one unit to its positive side. Four minus-ones are an explicit crop, not the current-view sentinel used by some other print methods.

Use the declaration for the overload you call. ByRef arguments can receive outputs; declare variables and arrays with the exact indicated type. Optional defaults are shown in the declaration.

Return value

Boolean - True on success, false on error.

Remarks

Exports SVG 1.1 with explicit physical width/height in centimeters. Both overloads accept Double WidthCm/HeightCm ByVal. One zero dimension is inferred from the resolved crop aspect ratio; both zero, negative, nonfinite or unrepresentable dimensions return False. Positive dimensions above Single.MaxValue are rejected as unsupported by downstream renderers. All-zero crop uses the whole drawing even for selected-only output; explicit reversed corners normalize internally. Automatic bounds use actual geometry rather than spatial-index padding. A zero span expands by one drawing unit on its positive side, retaining the R2 SVG rule and safely extending it to explicit degenerate crops; where adding one rounds away, the next finite representable coordinate is used. Four minus-one crop values are literal coordinates. The SVG viewBox includes a ten-percent border on all sides; default xMidYMid meet preserves geometry aspect within the requested physical viewport. Dimensions, filename, selection flag and crop inputs are unchanged. OriginTopLeft_p consistently switches both geometry and frame Y direction, correcting the old off-frame top-left behavior. Per-object stroke widths are calculated from the actual fitted physical scale at 96 CSS pixels per inch, preserving distinct pen widths instead of the old FixSVG global string replacement and missing-dimension fallback. Very small, very large or distant coordinates are rebased/rescaled in the SVG geometry itself, including pens and text, to avoid renderer precision loss without changing physical placement. Optional data-avax-coordinate-scale and data-avax-origin-x/y attributes record that reversible coordinate mapping. Invalid/unrepresentable mappings fail. Every serialized numeric geometry value is checked after normalization, preventing Infinity/NaN path data from replacing a valid file. Hidden/nonprinting layers are omitted, locked layers remain printable, and selection is preserved; selected-only with no selected objects returns False. Started/Enabled/disposal and shared print ownership gates apply. External-line callbacks run before the export snapshot; lifecycle replacement/cancellation and recursive export cannot publish stale output. SVG does not raise PrintingPage. Relative destinations resolve before callbacks. Complete UTF-8 XML is flushed to an owned same-directory staging file and atomically replaces the destination; failure preserves previous file bytes and all caller ByRefs. Geometric paths, solid polygons and simple text have vector output. With VectorTextExportOn_p disabled, ordinary Windows text remains searchable native SVG text and retains its independent width and height, affine shear/reflection, rotation, font family, bold/italic style, Unicode and multiline placement. Underline and strikeout follow the same frame as explicit vector strokes. Native font advances and requested text lengths preserve the drawing width; a reader uses its installed font family, so glyph rasterization can differ between readers. Finite text metrics are normalized before their final Double scaling, avoiding intermediate overflow for very small or large geometry. With VectorTextExportOn_p enabled and PltTextQuality_p set to 0, 1 or 2, Windows text uses filled sampled glyph contours from the printing font, retaining counters, independent width/height, affine placement, multiline baselines, bold/italic, underline and strikeout. These paths do not depend on fonts installed in the SVG reader. Original Unicode remains in SVG titles. Sampling is performed in canonical font units before the Double world transform; small, large and distant drawing coordinates do not reduce the glyph outline to world-coordinate Single precision. When vector export is enabled, quality -1 selects the R2 DOS stroke-font profile for ordinary text. It uses the actual S0..S8.OPT glyph design, with S1 for indices above 8, the current TextInitWidth_p/TextInitHeight_p spacing ratio and a nominal-font-height line advance. Underline, strikeout, independent width/height, full affine placement and the top-left convention are retained. Rich documents keep their chosen Windows run fonts. If no legacy font resource is available, the Windows-font fallback uses normal contour sampling. Automatic bounds and explicit crop candidates use the real legacy glyph and decoration ink. Very small or large scales and character advances are evaluated without intermediate font-ratio overflow; a final unrepresentable coordinate returns False without replacing the destination. Automatic frames with an extreme aspect ratio make their fitted physical viewport explicit so a resolvable stroke is not lost. Pictures embed their image pixels as PNG data URIs, including their image effects and affine placement. Shape fills and line styles use the same drawing resources as the printing renderer: native linear gradients remain vector gradients, hatch boundaries and holes remain vector paths, and numeric dash runs and complex line symbols retain their ink. Bitmap patterns embed their tile pixels rather than rasterizing the whole drawing. Shape opacity and native hatch transparency are represented by SVG compositing groups. Resource identifiers are unique within the exported document. Recorder-backed fills and line styles require a positive finite CSS-pixel viewport within the Single drawing-backend range; an unsupported extent returns False without replacing the destination. Rich documents serialize their actual positioned Windows text runs, font/color/style resources, vector table edges and rules, and embedded picture resources. Text and table pages remain vector content; pictures alone use image data. Page clipping follows the RichEdit layout while separately rendered rules and controls retain their own ordering. Rich text and document frames retain shear, reflection and the top-left convention. Temporary font/palette tables do not alter the drawing. Explicit first highlight or shading is retained; the matte-dependent control/page background remains transparent. Rich Windows run fonts remain active when ordinary Windows fonts are disabled. Shaped TrueType runs, including contextual scripts, right-to-left text, combining marks and supplementary Unicode, retain scalable glyph outlines and their recorded run placement. The source Unicode is retained in the rich document SVG title; outlined glyphs are not editable text elements. Curves use unhinted font geometry so they scale without inheriting a temporary screen pixel grid. Raster font hinting can differ slightly from bitmap printing. This SVG rendering support does not change the separate editable-import restrictions of Add_RTF. Other specialized imported-resource presentation retains its separate rendering paths. When ordinary Windows fonts are disabled, ordinary text uses the actual S0..S8.OPT stroke-font resources even with VectorTextExportOn_p disabled. The selected font and its fallback follow the drawing renderer. OEM Greek characters, stored character advances, 1.25-height line spacing and the full text frame are retained as vector strokes; a Windows font is not substituted. Explicit Windows-font text and rich documents keep their separate font rules. Point symbols are exported as vector ink using the current fixed/minimum size and cap preferences. Explicit round/butt caps and round joins are retained on stroked paths; connected polyline segments remain joined. Symbols crossing the output frame are included even when their insertion point lies outside it. Plain

text shapes and plain document shapes retain their actual four-corner frame, fitted multiline layout, leading/trailing blank lines, wrapped words, font attributes and rotations, shear and reflections. Visible text-shape borders retain their line style. These framed glyphs are exported as vector outlines in both VectorTextExportOn_p modes so the fitted layout does not depend on fonts installed in the SVG reader. The original Unicode text is retained in the group title. Path export does not change VectorTextPresentationOn_p or other stored drawing preferences. Plain ordinary Windows documents retain their stored field width and wrapped or explicit rows. With VectorTextExportOn_p False, their rows remain native SVG text; True produces filled glyph outlines. Both modes retain the full document matrix, including shear, reflection, rotation and top-left origin placement, with source Unicode in the title. Bold, italic, underline and strikeout follow the selected drawing font. Legacy-font ordinary documents retain their drawing stroke-font mode rather than substituting Windows text. Missing legacy resources use the drawing Windows-font fallback.

Numeric changes from v2

Update the following variables, arrays or event-data receivers when porting the corresponding v2 call. The declarations above show the current overloads.

Parameter or value	v2 type	v3 type
WidthCm	Single	Double
HeightCm	Single	Double
xMin	Single	Double
yMin	Single	Double
xMax	Single	Double
yMax	Single	Double

For ByRef parameters, use a variable with the exact v3 type. Preserve array dimensions and the documented indexing. Moving from v2 to v3 explains conversion and legacy file-storage rules.

Example

Setup

Add an Avax.cAvax Windows Forms control named `cAvax1`. Paste the procedures into the form class. Example setup and supporting files explains references, initialization and file paths.

Add these Windows Forms controls or components to the form. Fields referenced by Handles clauses must be declared WithEvents, as the Visual Studio designer normally does.

Name	Type
Command1	Button

VB.NET example

```
Private Sub Form1_Load(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs) Handles
    MyBase.Load
    Command1.Text = "PrintToSVG"
    cAvax1.StartAvax()
    Call CAvax1.Add_Line(0, 5, 0, 10, 5, 0, Nothing, 1, 1, -1, -1)
    Call CAvax1.Add_Line(5, 0, 0, 5, 10, 0, Nothing, 1, 1, -1, -1)
    Call CAvax1.Add_Circle(5, 5, 0, 3, Nothing, 5, -1, -1, -1)
End Sub
Private Sub Form1_FormClosed(ByVal eventSender As System.Object, ByVal eventArgs As
    System.Windows.Forms.FormClosedEventArgs) Handles Me.FormClosed
    cAvax1.EndAvax()
End Sub
Private Sub Command1_Click(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs)
    Handles Command1.Click
    Dim sSVGFile As String = String.Empty
    Dim WidthCM As Double
    Dim HeightCM As Double
    Dim fSelected As Boolean
    Dim xMin As Double
    Dim yMin As Double
```

```
    Dim xMax As Double
    Dim yMax As Double
    Dim Answer As Boolean
    sSVGFile = My.Application.Info.DirectoryPath & "\..\temp.svg"
    WidthCM = 5
    HeightCM = 5
    Answer = cAvax1.PrintToSVG(sSVGFile, WidthCM, HeightCM, fSelected, xMin, yMin, xMax, yMax)
End Sub
```

PrintToSWF

cAvax method

Purpose

Exports the drawing into a .SWF (Shockwave Flash) file

VB.NET declarations

```
Function PrintToSWF(ByRef sSwfFile As String, ByRef lWidth As Integer, ByRef lHeight As Integer, ByRef
    fSelected As Boolean) As Boolean

Function PrintToSWF(ByRef sSwfFile As String, ByRef lWidth As Integer, ByRef lHeight As Integer, ByRef
    lFrameRate As Integer, ByRef fSelected As Boolean, ByRef xMin As Double, ByRef yMin As Double, ByRef xMax
    As Double, ByRef yMax As Double, ByRef Alpha As Integer) As Boolean
```

Parameters

Name	Type and passing	Meaning
sSwfFile	ByRef sSwfFile As String	Destination SWF path. An existing file is replaced only after complete successful generation.
lWidth	ByRef lWidth As Integer	Stage width in pixels; positive Integer within the SWF coordinate range. Raster content also has bitmap-format and pixel-buffer limits.
lHeight	ByRef lHeight As Integer	Stage height in pixels. Aspect ratio is preserved, with centered unused margins.
fSelected	ByRef fSelected As Boolean	True exports only the current selection; an empty selection produces a valid empty frame.
lFrameRate	ByRef lFrameRate As Integer	Frames per second, from 1 through 255. The result is a static one-frame movie. The Integer overload uses 24.
xMin	ByRef xMin As Double	Left drawing coordinate, as Double. If all four extent arguments are zero, use the bounds of the exported objects.
yMin	ByRef yMin As Double	Bottom drawing coordinate, as Double. Explicit extents must have positive width and height.
xMax	ByRef xMax As Double	Right drawing coordinate, as Double. Geometry outside the requested drawing rectangle is clipped.
yMax	ByRef yMax As Double	Top drawing coordinate, as Double. This corrects the original CHM table's mistaken reference to an x coordinate.
Alpha	ByRef Alpha As Integer	Opacity percentage from 0 (transparent) to 100 (opaque), promoted from Integer to Integer. The Integer overload uses 100.

Use the declaration for the overload you call. ByRef arguments can receive outputs; declare variables and arrays with the exact indicated type. Optional defaults are shown in the declaration.

Return value

Boolean - True on success, false on error.

Numeric changes from v2

Update the following variables, arrays or event-data receivers when porting the corresponding v2 call. The declarations above show the current overloads.

Parameter or value	v2 type	v3 type
xMin	Single	Double
yMin	Single	Double

Parameter or value	v2 type	v3 type
xMax	Single	Double
yMax	Single	Double
Alpha	Short	Integer

For ByRef parameters, use a variable with the exact v3 type. Preserve array dimensions and the documented indexing. Moving from v2 to v3 explains conversion and legacy file-storage rules.

Example

Setup

Add an Avax.cAvax Windows Forms control named `cAvax1`. Paste the procedures into the form class. Example setup and supporting files explains references, initialization and file paths.

Add these Windows Forms controls or components to the form. Fields referenced by Handles clauses must be declared WithEvents, as the Visual Studio designer normally does.

Name	Type
Button1	Button

VB.NET example

```
Private Sub Form1_Load(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs) Handles
    MyBase.Load
    Call cAvax1.StartAvax()
    Call cAvax1.Add_Line(0, 5, 0, 10, 5, 0, Nothing, 1, 1, -1, -1)
    Call cAvax1.Add_Line(5, 0, 0, 5, 10, 0, Nothing, 1, 1, -1, -1)
    Call cAvax1.Add_Circle(5, 5, 0, 3, Nothing, 5, -1, -1, -1)
End Sub
Private Sub Form1_FormClosed(ByVal eventSender As System.Object, ByVal eventArgs As
    System.Windows.Forms.FormClosedEventArgs) Handles Me.FormClosed
    Call cAvax1.EndAvax()
End Sub
Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
    Button1.Click
    Dim sSwfFile As String = String.Empty
    Dim lWidth As Integer
    Dim lHeight As Integer
    Dim lFrameRate As Integer
    Dim fSelected As Boolean
    Dim xMin As Double
    Dim yMin As Double
    Dim xMax As Double
    Dim yMax As Double
    Dim Alpha As Integer
    Dim Answer As Boolean
    sSwfFile = My.Application.Info.DirectoryPath & "\..\temp.swf"
    lWidth = 1200
    lHeight = 1200
    xMin = 0
    yMin = 0
    xMax = 0
    yMax = 0
    Alpha = 100
    lFrameRate = 24
    Answer = cAvax1.PrintToSWF(sSwfFile, lWidth, lHeight, lFrameRate, fSelected, xMin, yMin, xMax, yMax,
    Alpha)
End Sub
```

PrintToTIF

cAvax method

Purpose

Exports the drawing into a .TIF file

VB.NET declarations

```
Function PrintToTIF(ByRef sTIFFile As String, ByVal lWidth As Integer, ByVal lHeight As Integer, ByRef
    fSelected As Boolean, ByRef xMin As Double, ByRef yMin As Double, ByRef xMax As Double, ByRef yMax As
    Double, ByRef sScale As Double) As Boolean

Function PrintToTIF(ByRef sTIFFile As String, ByVal lWidth As Integer, ByVal lHeight As Integer) As
    Boolean
```

Parameters

Name	Type and passing	Meaning
sTIFFile	ByRef sTIFFile As String	String destination path, unchanged ByRef. Relative paths resolve before callbacks. The actual TIF encoder is used regardless of extension; preparation or commit failure preserves existing output.
lWidth	ByVal lWidth As Integer	Integer image width in pixels, exactly as written in the TIF header. Zero derives width from positive height and whole-drawing aspect ratio. Both zero fail; negative/overflow/allocation failures preserve existing output. Unlike the old implementation this is not width+1.
lHeight	ByVal lHeight As Integer	Integer image height in pixels, with the same rules as lWidth. Width and height stay ByVal. Fit includes the last pixel centre so bottom/right boundary lines remain visible.
fSelected	ByRef fSelected As Boolean	True exports only selected objects and fails if the selection is empty. Automatic extents still use the complete drawing as in R2. This ByRef flag is unchanged.
xMin	ByRef xMin As Double	Double left coordinate. Four zeros mean complete drawing extents; four minus-ones mean the current nominal view. Actual normalized extents are returned only on success.
yMin	ByRef yMin As Double	Double bottom coordinate. All extent coordinates must be finite and produce a representable rendering scale.
xMax	ByRef xMax As Double	Double right coordinate. Explicit reversed bounds normalize; exactly zero spans receive a usable usable nonzero frame (adjacent finite Double coordinates at large origins).
yMax	ByRef yMax As Double	Double top coordinate; corrects the CHM table's reference to an x coordinate. Failed preparation/file commit leaves all extent ByRefs unchanged.
sScale	ByRef sScale As Double	Double geometric multiplier about the output top-left. Finite values <=0 become 1, committed only on success. NaN/infinity/unrepresentable scale fail; scaling larger than one may intentionally clip geometry.

Use the declaration for the overload you call. ByRef arguments can receive outputs; declare variables and arrays with the exact indicated type. Optional defaults are shown in the declaration.

Return value

Boolean - True on success, false on error.

Numeric changes from v2

Update the following variables, arrays or event-data receivers when porting the corresponding v2 call. The declarations above show the current overloads.

Parameter or value	v2 type	v3 type
xMin	Single	Double
yMin	Single	Double
xMax	Single	Double
yMax	Single	Double
sScale	Single	Double

For ByRef parameters, use a variable with the exact v3 type. Preserve array dimensions and the documented indexing. Moving from v2 to v3 explains conversion and legacy file-storage rules.

Example

Setup

Add an Avax.cAvax Windows Forms control named `cAvax1`. Paste the procedures into the form class. Example setup and supporting files explains references, initialization and file paths.

Add these Windows Forms controls or components to the form. Fields referenced by Handles clauses must be declared WithEvents, as the Visual Studio designer normally does.

Name	Type
Button1	Button

VB.NET example

```
Private Sub Form1_Load(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs) Handles
    MyBase.Load
    Call cAvax1.StartAvax()
    Call cAvax1.Add_Line(0, 5, 0, 10, 5, 0, Nothing, 1, 1, -1, -1)
    Call cAvax1.Add_Line(5, 0, 0, 5, 10, 0, Nothing, 1, 1, -1, -1)
    Call cAvax1.Add_Circle(5, 5, 0, 3, Nothing, 5, -1, -1, -1)
End Sub
Private Sub Form1_FormClosed(ByVal eventSender As System.Object, ByVal eventArgs As
    System.Windows.Forms.FormClosedEventArgs) Handles Me.FormClosed
    Call cAvax1.EndAvax()
End Sub
Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
    Button1.Click
    Dim sTIFFile As String = String.Empty
    Dim lWidth As Integer
    Dim lHeight As Integer
    Dim fSelected As Boolean
    Dim xMin As Double
    Dim yMin As Double
    Dim xMax As Double
    Dim yMax As Double
    Dim sScale As Double
    Dim Answer As Boolean
    sTIFFile = My.Application.Info.DirectoryPath & "\..\temp.tif"
    lWidth = 120
    lHeight = 120
    sScale = 1
    Answer = cAvax1.PrintToTIF(sTIFFile, lWidth, lHeight, fSelected, xMin, yMin, xMax, yMax, sScale)
End Sub
```

PrintToWMF

cAvax method

Purpose

Exports the drawing or selected objects to a Windows metafile, optionally using supplied bounds and scale.

VB.NET declarations

```
Function PrintToWMF(ByRef sMetaFile As String, ByVal lWidth As Integer, ByVal lHeight As Integer, ByRef
    fSelected As Boolean, ByRef xMin As Double, ByRef yMin As Double, ByRef xMax As Double, ByRef yMax As
    Double, ByRef sScale As Double) As Boolean

Function PrintToWMF(ByRef sMetaFile As String, ByVal lWidth As Integer, ByVal lHeight As Integer) As
    Boolean
```

Parameters

Name	Type and passing	Meaning
sMetaFile	ByRef sMetaFile As String	Destination path. Relative paths are fixed at call entry. Success replaces the destination atomically; failure preserves an existing file. The caller filename is unchanged.
lWidth	ByVal lWidth As Integer	Integer width, 1..32767 logical WMF units. Zero calculates this dimension from the whole drawing aspect ratio when height is positive. The placeable file uses 1440 units per inch and the historical inclusive plus-one frame, capped at 32767.
lHeight	ByVal lHeight As Integer	Integer height, 1..32767 logical WMF units. Zero calculates this dimension from the whole drawing aspect ratio when width is positive. Both zero fail. Dimensions remain ByVal.
fSelected	ByRef fSelected As Boolean	True prints only selected objects. An empty selection returns False. All-zero crop arguments still use complete drawing bounds, as in R2.
xMin	ByRef xMin As Double	Double left drawing coordinate. Four zeros request whole-drawing bounds; four minus-ones request the current view. Actual normalized bounds are returned only on success.
yMin	ByRef yMin As Double	Double bottom drawing coordinate. Nonfinite bounds fail. Exactly zero drawing width or height receives a one-unit frame so lines/points can export.
xMax	ByRef xMax As Double	Double right drawing coordinate. Explicit reversed corners are normalized; output is clipped to the requested frame.
yMax	ByRef yMax As Double	Double top drawing coordinate. This corrects the original CHM table reference to an x coordinate. Returned ByRef only on success.
sScale	ByRef sScale As Double	Double geometric scale about the output top-left. Nonpositive finite values normalize to 1 and are returned ByRef on success. NaN/infinity fail without changing caller arguments.

Use the declaration for the overload you call. ByRef arguments can receive outputs; declare variables and arrays with the exact indicated type. Optional defaults are shown in the declaration.

Return value

Boolean - True on success, false on error.

Remarks

Exports a placeable Windows metafile. Width and height are Integer logical units at 1440 units per inch, each limited to 32767. The stored inclusive frame is the requested size plus one unit, capped at 32767. The full overload supplies Double bounds and scale and can restrict output to selected objects. One zero dimension derives the other from whole-drawing aspect ratio. Four zero crop coordinates mean the whole drawing; four -1 values mean the current view.

Simple geometry uses vector records. Text, images, fills and other complex graphics can use bounded bitmap records. PrintPaperRGBColor_p controls foreground contrast and the background inside bitmap regions. For white paper use &HFFFFFF, which renders white drawing colors as black. BackColor controls only the editing canvas.

A started, enabled control is required. PrintingPage reports the one output page unless events are paused. AbortPrinting or a changed document/selection/lifecycle can cancel output. Invalid input, cancellation or file failure returns False and retains the destination. ByRef bounds/scale are published only after successful completion.

Numeric changes from v2

Update the following variables, arrays or event-data receivers when porting the corresponding v2 call. The declarations above show the current overloads.

Parameter or value	v2 type	v3 type
xMin	Single	Double
yMin	Single	Double
xMax	Single	Double
yMax	Single	Double
sScale	Single	Double

For ByRef parameters, use a variable with the exact v3 type. Preserve array dimensions and the documented indexing. Moving from v2 to v3 explains conversion and legacy file-storage rules.

Example

Setup

Add an Avax.cAvax Windows Forms control named `cAvax1`. Paste the procedures into the form class. Example setup and supporting files explains references, initialization and file paths.

Add these Windows Forms controls or components to the form. Fields referenced by Handles clauses must be declared WithEvents, as the Visual Studio designer normally does.

Name	Type
Command1	Button

VB.NET example

```
Private Sub Form1_Load(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs) Handles
    MyBase.Load
    Command1.Text = "PrintToWMF"
    cAvax1.StartAvax()
    Call CAvax1.Add_Line(0, 5, 0, 10, 5, 0, Nothing, 1, 1, -1, -1)
    Call CAvax1.Add_Line(5, 0, 0, 5, 10, 0, Nothing, 1, 1, -1, -1)
    Call CAvax1.Add_Circle(5, 5, 0, 3, Nothing, 5, -1, -1, -1)
End Sub
Private Sub Form1_FormClosed(ByVal eventSender As System.Object, ByVal eventArgs As
    System.Windows.Forms.FormClosedEventArgs) Handles Me.FormClosed
    cAvax1.EndAvax()
End Sub
Private Sub Command1_Click(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs)
    Handles Command1.Click
    Dim sMetaFile As String = String.Empty
    Dim lWidth As Integer
    Dim lHeight As Integer
    Dim fSelected As Boolean
```

```
    Dim xMin As Double
    Dim yMin As Double
    Dim xMax As Double
    Dim yMax As Double
    Dim sScale As Double
    Dim Answer As Boolean
    sMetaFile = My.Application.Info.DirectoryPath & "\..\temp.wmf"
    lWidth = 1200
    lHeight = 1200
    sScale = 1
    Answer = cAvax1.PrintToWMF(sMetaFile, lWidth, lHeight, fSelected, xMin, yMin, xMax, yMax, sScale)
End Sub
```

PrintToXML

cAvax method

Purpose

Exports the drawing into a .XML file

VB.NET declaration

```
Function PrintToXML(ByRef sXMLFile As String) As Boolean
```

Parameters

Name	Type and passing	Meaning
sXMLFile	ByRef sXMLFile As String	String destination passed ByRef, unchanged on success and failure. A relative path resolves before external providers. Empty/invalid paths fail. Existing output is replaced only after complete XML serialization and flush.

Use the declaration for the overload you call. ByRef arguments can receive outputs; declare variables and arrays with the exact indicated type. Optional defaults are shown in the declaration.

Return value

Boolean - True on success, false on error.

Remarks

Exports the complete drawing to the filename passed ByRef and returns True after the file has been committed. The filename variable, selection and current drawing command are retained. Failure returns False and leaves an existing destination intact. The control must be started and enabled. The output includes hidden and nonprinting layers. Entity records, layers, colors, line styles, fonts and images use the R2 AVAX XML schema, with General Version=1.0. A native supplement retains additional v3 details; it is ignored if the surrounding legacy XML is edited. External-line references are refreshed before the snapshot; their resolved geometry is available to R2 readers. XMLencoding_p selects the output encoding; an empty setting uses UTF-8. Geometry uses invariant decimal numbers with Double precision. Unicode text and properties retain their content and line endings. Characters unavailable in a supported legacy encoding use XML numeric references. Characters forbidden by XML, such as NUL, cause export to fail. No PrintingPage event is raised. A recursive export is rejected, and cancellation or document replacement during preparation abandons stale output. Plain text-filled shapes follow the current OriginTopLeft_p setting in the legacy representation, including text inside nested blocks. Use the same origin mode in the receiving R2 control to retain the displayed orientation. Exporting does not reorder the source drawing geometry.

Example

Setup

Add an Avax.cAvax Windows Forms control named `cAvax1`. Paste the procedures into the form class. Example setup and supporting files explains references, initialization and file paths.

Add these Windows Forms controls or components to the form. Fields referenced by Handles clauses must be declared WithEvents, as the Visual Studio designer normally does.

Name	Type
Button1	Button

VB.NET example

```
Private Sub Form1_Load(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs) Handles
    MyBase.Load
    Call CAvax1.StartAvax()
    Call CAvax1.Add_Line(0, 5, 0, 10, 5, 0, Nothing, 1, 1, -1, -1)
    Call CAvax1.Add_Line(5, 0, 0, 5, 10, 0, Nothing, 1, 1, -1, -1)
    Call CAvax1.Add_Circle(5, 5, 0, 3, Nothing, 5, -1, -1, -1)
```

```
End Sub
Private Sub Form1_FormClosed(ByVal eventSender As System.Object, ByVal eventArgs As
    System.Windows.Forms.FormClosedEventArgs) Handles Me.FormClosed
    Call cAvax1.EndAvax()
End Sub
Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
    Button1.Click
    Dim sXMLFile As String = String.Empty
    Dim Answer As Boolean
    sXMLFile = My.Application.Info.DirectoryPath & "\..\temp.xml"
    Answer = cAvax1.PrintToXML(sXMLFile)
End Sub
```

PrintWorkspaceToImage

Purpose

Exports the current workspace paper page as a white-background raster image. The page uses its configured physical size, orientation, drawing scale and printable margins in either Canvas or Page workspace mode.

Declaration

```
Function PrintWorkspaceToImage(ByVal fileName As String, ByVal pixelWidth As Integer) As Boolean
```

Parameters and result

Name	Meaning
fileName	Destination path. The extension selects PNG (.png), JPEG (.jpg or .jpeg), BMP (.bmp), GIF (.gif) or TIFF (.tif or .tiff). A relative path uses the current working directory; its directory must exist.
pixelWidth	Output width in pixels, from 1 through 16384. Zero does not request automatic sizing. Height is the ceiling of pixelWidth × paper height / paper width after orientation; it must also be from 1 through 16384 and the image must contain no more than 64,000,000 pixels.

Returns True after the destination has been committed, False when output cannot start or a handled error or cancellation prevents completion. The method is synchronous and requires an existing started, enabled, undisposed control with output available and no other print/export operation active.

Page rendering

The output follows the current workspace page, including PageMarginMM clipping. The background is white, printable layer visibility applies, and editor decorations such as selection handles and rulers are excluded. JPEG uses JPGCompressQuality_p. The filename and active drawing's saved-state/history are retained. Configure the page through the existing workspace properties before calling; RenderWorkspacePage supplies a bitmap when the host needs in-memory page rendering instead.

Failure and cancellation

An unsupported extension, invalid dimensions or an unrepresentable rendering scale prevents output. Live external references are refreshed before rendering. A drawing or control-context change detected while producing the image abandons it. AbortPrinting requests cooperative cancellation; checks occur at operation boundaries. Rendering and encoding use an owned temporary destination, so a handled failure or cancellation preserves an existing file. Inspect LastOperationError after False when a diagnostic is available; an early rejection can leave it empty or unchanged.

VB.NET example

```
Option Strict On
Public Module PrintWorkspaceToImageExample
    ' The host has already configured the workspace paper and scale.
    Public Function WritePageImage(
        cad As Avax.cAvax, destination As String,
        pixelWidth As Integer) As Boolean
        Dim saved As Boolean = cad.PrintWorkspaceToImage(destination, pixelWidth)
        If Not saved Then
            Dim message As String = cad.LastOperationError
            If Not String.IsNullOrEmpty(message) Then
                System.Diagnostics.Debug.WriteLine(message)
            End If
        End If
        Return saved
    End Function
End Module
```

Related options and reference

PrintWorkspaceToPDF

Purpose

Export the single workspace sheet at its physical size and stored scale, in either mode.

VB.NET declaration

```
Function PrintWorkspaceToPDF(ByVal fileName As String) As Boolean
```

Parameters

Parameter	Meaning
fileName	Destination PDF path supplied by the application.

Return value and behavior

True on successful export; False on failure. Clips to margins; does not automatically fit objects outside the page.

Example

Call this procedure on the owning UI thread with a control already initialized by `StartAvax()`. Handles must belong to that control. File paths are supplied by the caller.

```
Option Strict On
Imports System
Imports Avax

Public Module Help_PrintWorkspaceToPDF
    Public Sub Run(cad As cAvax, pdfPath As String)
        If Not cad.PrintWorkspaceToPDF(pdfPath) Then Throw New InvalidOperationException("Workspace PDF
    export failed.")
    End Sub
End Module
```

Related options and reference

PrintWorkspaceToPrinter

Purpose

Print the single workspace sheet at the configured scale.

VB.NET declaration

```
Function PrintWorkspaceToPrinter(Optional ByVal printerName As String = "", Optional ByVal showDialog As
    Boolean = True) As Boolean
```

Parameters

Parameter	Meaning
printerName	Printer name; empty default uses the existing printer selection behavior.
showDialog	True (default) shows the printer dialog. False uses supplied/current printer settings directly.

Return value and behavior

Boolean success result; cancellation or failure returns False. Select matching paper and orientation in the driver; no automatic scale change to fit a different sheet.

Example

Call this procedure on the owning UI thread with a control already initialized by `StartAvax()`. Handles must belong to that control. File paths are supplied by the caller.

```
Option Strict On
Imports System
Imports Avax

Public Module Help_PrintWorkspaceToPrinter
    Public Sub Run(cad As cAvax)
        Dim printed As Boolean = cad.PrintWorkspaceToPrinter(showDialog:=True)
        If Not printed Then Return
    End Sub
End Module
```

Related options and reference

PrintWorkspaceToRTF

Purpose

Exports the workspace sheet as a metafile picture inside an RTF document. Drawing objects are represented by that picture; the method does not convert them into editable RTF text.

VB.NET declaration

```
Public Function PrintWorkspaceToRTF(ByVal fileName As String, ByVal pixelWidth As Integer) As Boolean
```

Parameters

Name	Meaning
fileName	Required writable destination RTF path; no default and no Save dialog.
pixelWidth	Required layout/reference picture width from 1 through 16384. Drawing height follows the oriented paper aspect ratio, rounded upward. It must be at most 16384, and width × height must be at most 64,000,000. There is no default width.

Result and behavior

Returns True when the RTF file is written successfully. An invalid width or computed height, unavailable control, invalid destination, cancellation or handled export failure returns False. Call after `StartAvax()` with an enabled, live control.

The embedded picture retains the metafile's inclusive frame: its RTF goal size is `(pixelWidth + 1)` × `(computedHeight + 1)` twips. A 400 × 200 drawing extent therefore gives a 401 × 201 twip picture goal. Changing the requested extent changes that physical picture size; the RTF document and picture do not automatically acquire the workspace paper dimensions.

The picture uses the workspace page bounds and printable-area clipping in either Canvas or Page mode. It follows PageBackgroundPrintExport and PageBackgroundColor: include the stored paper color with True, or white paper with False. Editor labels, guides and shadows are excluded.

Example

Pass a started control and a destination path chosen by the application.

```
Option Strict On
Imports System
Imports Avax

Public Module Help_PrintWorkspaceToRTF
    Public Sub Run(cad As cAvax, rtfPath As String)
        If Not cad.PrintWorkspaceToRTF(rtfPath, 1200) Then
            Throw New InvalidOperationException("Workspace RTF export failed.")
        End If
    End Sub
End Module
```

Related options and reference

PrintWorkspaceToSVG

Purpose

Exports the configured workspace sheet to SVG at its stored physical paper size, orientation and drawing scale. It uses the workspace in either Canvas or Page mode and clips drawing content to the printable area.

VB.NET declaration

```
Public Function PrintWorkspaceToSVG(ByVal fileName As String) As Boolean
```

Parameter

Name	Meaning
fileName	Required destination SVG file path. Supply a valid writable path; the method has no default file name and does not open a Save dialog.

Result and behavior

Returns True when the SVG file is written successfully. An unavailable control, invalid destination, cancelled operation or handled export failure returns False. Call after `StartAvax()` with an enabled, live control. The configured page bounds and `PageMarginMM` determine the output; the method does not automatically fit all drawing objects onto the sheet.

With PageBackgroundPrintExport set to True, SVG contains a full-sheet rectangle filled with PageBackgroundColor. With False, that background rectangle is omitted. Editor labels, margin guides and shadows are excluded.

Example

Pass a started control and a destination path chosen by the application.

```
Option Strict On
Imports System
Imports Avax

Public Module Help_PrintWorkspaceToSVG
    Public Sub Run(cad As cAvax, svgPath As String)
        If Not cad.PrintWorkspaceToSVG(svgPath) Then
            Throw New InvalidOperationException("Workspace SVG export failed.")
        End If
    End Sub
End Module
```

Related options and reference

PrintWorkspaceToWMF

Purpose

Exports the workspace sheet to a Windows Metafile (WMF), computing its drawing height from the configured paper aspect ratio. It uses the workspace layout and printable-area clipping in either Canvas or Page mode.

VB.NET declaration

```
Public Function PrintWorkspaceToWMF(ByVal fileName As String, ByVal pixelWidth As Integer) As Boolean
```

Parameters

Name	Meaning
fileName	Required writable destination WMF path; no default and no Save dialog.
pixelWidth	Required layout/reference width from 1 through 16384. There is no default. Drawing height is rounded upward from the oriented paper aspect ratio; it must also be at most 16384, and width × height must be at most 64,000,000.

Result and behavior

Returns True on successful file output. An invalid width or computed height, unavailable control, invalid destination, cancellation or handled export failure returns False. Call after `StartAvax()` with an enabled, live control. This method exports the workspace sheet rather than the current selection or screen zoom.

The placeable WMF uses an inclusive frame of `(pixelWidth + 1)` × `(computedHeight + 1)` twips, at 1440 twips per inch. For example, a 400 × 200 drawing extent produces a 401 × 201 twip frame. Despite the parameter name, the saved physical frame is not a pixel measurement or automatically the physical paper size; it depends on the requested extent.

The paper fill follows PageBackgroundColor when PageBackgroundPrintExport is True; False uses white paper. Page margins clip drawing content. Editor captions, guides and shadows are excluded.

Example

Pass a started control and a destination path chosen by the application.

```
Option Strict On
Imports System
Imports Avax

Public Module Help_PrintWorkspaceToWMF
    Public Sub Run(cad As cAvax, wmfPath As String)
        If Not cad.PrintWorkspaceToWMF(wmfPath, 1200) Then
            Throw New InvalidOperationException("Workspace WMF export failed.")
        End If
    End Sub
End Module
```

Related options and reference

RedrawItems

cAvax method

Purpose

Redraws the objects described in a specific handles array

VB.NET declarations

```
Function RedrawItems(ByRef avMyHandles As Integer()) As Boolean

Function RedrawItems(ByRef avMyHandles As Integer(), ByRef iClr As Integer, ByRef ItemType As
    AvaxItemType) As Boolean
```

Parameters

Name	Type and passing	Meaning
avMyHandles	ByRef avMyHandles As Integer()	One-based Integer array, unchanged. Nothing or zero/one element means all objects. Slot zero is ignored.
iClr	ByRef iClr As Integer	Integer display color; -1 uses each object's stored color. Stored properties are unchanged.
ItemType	ByRef ItemType As AvaxItemType	AvaxItemType filter; Any_i includes every type.

Use the declaration for the overload you call. ByRef arguments can receive outputs; declare variables and arrays with the exact indicated type. Optional defaults are shown in the declaration.

Return value

Boolean - True on success, false on error.

Numeric changes from v2

Update the following variables, arrays or event-data receivers when porting the corresponding v2 call. The declarations above show the current overloads.

Parameter or value	v2 type	v3 type
iClr	Short	Integer

For ByRef parameters, use a variable with the exact v3 type. Preserve array dimensions and the documented indexing. Moving from v2 to v3 explains conversion and legacy file-storage rules.

Example

Setup

Add an Avax.cAvax Windows Forms control named `cAvax1`. Paste the procedures into the form class. Example setup and supporting files explains references, initialization and file paths.

Add these Windows Forms controls or components to the form. Fields referenced by Handles clauses must be declared WithEvents, as the Visual Studio designer normally does.

Name	Type
Command1	Button

VB.NET example

```
Private Sub Form1_Load(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs) Handles
    MyBase.Load
    Dim iClr As Integer
    Command1.Text = "RedrawItems"
```

```
    cAvax1.StartAvax()
    iClr = 1
    Call CAvax1.Add_Line(0, 0, 0, 10, 10, 0, Nothing, iClr, -1, -1, -1)
    Call CAvax1.Add_Line(0, 10, 0, 10, 10, 0, Nothing, iClr, -1, -1, -1)
End Sub
Private Sub Form1_FormClosed(ByVal eventSender As System.Object, ByVal eventArgs As
    System.Windows.Forms.FormClosedEventArgs) Handles Me.FormClosed
    cAvax1.EndAvax()
End Sub
Private Sub Command1_Click(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs)
    Handles Command1.Click
    Dim [Handles]() As Integer = Nothing
    Dim iClr As Integer
    Dim ItemType As Avax.AvaxItemType
    Dim Answer As Boolean
    If CAvax1.GetAllHandlesArr([Handles]) > 0 Then
        iClr = 2 'Redraw objects with an other color
        Answer = CAvax1.RedrawItems([Handles], iClr, ItemType)
    End If
End Sub
```

Refresh

```
Public Overrides Sub Refresh()
```

Explicitly repaints the drawing and viewport. Unlike automatic presentation, this request also works when AutoRedraw is False. It preserves the active drawing command, accepted construction points, selection and Undo/Redo history. Normal paint events are raised unless PauseAvaxEvents is True.

A visible control is refreshed before the call returns. A hidden control retains the request for its next paint. A request made during BeginUpdate waits until display suspension ends; one made inside a paint callback is deferred to a later presentation to avoid recursive painting.

Related options and reference

Regen

```
Public Sub Regen()
```

Regenerates the drawing presentation, including external drawing references and embedded-control display values, and requests a refresh. Use it when cached display content needs to be rebuilt. The control must have a started drawing. It follows the same display-suspension and AutoRedraw rules as Refresh.

External background and tile files and control picture files are loaded again, including replacements that keep the same file name, length and timestamp. Embedded-control providers update the displayed values without rewriting the stored document or adding an Undo operation. PauseAvaxEvents suppresses those providers. Application event handlers may make their own drawing changes; their exceptions propagate to the caller.

The active command, its accepted Double coordinates and the current selection are preserved. A provider that ends the control or installs a new drawing owns that new state. Before StartAvax, after EndAvax and after disposal, this method has no effect.

VB.NET example

Call this after changing the application's external drawing data or replacing cached image files.

```
Option Strict On
Public Module RegenerateDrawingExample
    Public Sub RefreshDrawing(cad As Avax.cAvax)
        cad.Regen()
    End Sub
End Module
```

Related options and reference

RegenControlsZoomBitmap

```
Public Sub RegenControlsZoomBitmap()
```

Regenerates the presentation of embedded controls and repaints the drawing. Host events that supply control captions, text, values and other display data are consulted again. The control must have a started drawing. A request made while presentation is suspended is applied when EndUpdate resumes it.

This also refreshes live fields created by AfterPaint. Stored drawing objects, their source text and the Undo history are preserved unless the application's callbacks edit them. Changed or missing control picture files are reloaded. PauseAvaxEvents suppresses data-provider callbacks. A provider exception propagates, and a later call can retry the refresh.

With nested BeginUpdate calls, control-provider requests and the drawing presentation wait for the outermost EndUpdate. Explicit regeneration works with AutoRedraw=False. Calls require the UI thread and a started drawing; inactive, ended or disposed controls are left unchanged. In v3 this method shares the full regeneration path with Regen, so external drawing references and background resources are refreshed too.

VB.NET example

Call this outside a control data-provider event, after updating the values returned by the application.

```
Option Strict On
Public Module RegenerateControlsExample
    Public Sub RefreshControlDisplay(cad As Avax.cAvax)
        cad.RegenControlsZoomBitmap()
    End Sub
End Module
```

All five methods have no parameters and no return value. Their calling signatures are unchanged in v3. Coordinates used by the drawing methods in the example are Double; object handles remain Integer. Invoke these methods on the control's UI thread.

Related options and reference

RenderImagePreview

Purpose

Renders an image into a detached PNG byte array for an application's preview window. A proposed list can be previewed without changing the drawing or its Undo history.

Declaration

```
Function RenderImagePreview(ByVal handle As Integer,
                            Optional ByVal maximumDimension As Integer = 640,
                            Optional ByVal effects As AvaxImageEffect() = Nothing) As Byte()
```

Parameters

Name	Meaning
handle	Integer handle of a supported stored bitmap shape.
maximumDimension	Maximum preview width or height in pixels, from 1 through 16384 inclusive. Default 640. Aspect ratio is retained; smaller source images are not enlarged.
effects	Nothing uses the image's stored list. An empty array requests an unfiltered source preview. A supplied list replaces the stored list only for this preview and follows the same kind, amount and 32-entry validation rules as SetImageEffects.

Result and errors

Returns a new Byte array containing the encoded PNG, or Nothing for an unavailable image, invalid preview size, invalid list or decoding/rendering failure. Reading remains available while the control is disabled or editing is locked, including images on hidden or frozen layers. The rendered preview canvas must contain at most 32,000,000 pixels.

Validation, source decoding and rendering failures return Nothing. An unavailable image handle or a maximum dimension below one also returns Nothing. Changing the returned bytes does not change a cached preview, source image or drawing.

Blur and Pixelate scale their pixel-based amounts for the preview resolution. Crop keeps the preview canvas size and resamples its centred inner area. To obtain the retained encoded source itself, use GetImageSource.

VB.NET example

```
Option Strict On
Public Module RenderImagePreviewExample
    Public Function CurrentAppearance(cad As Avax.cAvax,
                                      handle As Integer) As Byte()
        Return cad.RenderImagePreview(handle)
    End Function
    Public Function UnfilteredAppearance(cad As Avax.cAvax,
                                         handle As Integer) As Byte()
        Return cad.RenderImagePreview(handle, 640, New Avax.AvaxImageEffect() {})
    End Function
    Public Function ProposedAppearance(cad As Avax.cAvax,
                                       handle As Integer) As Byte()
        Dim proposed As Avax.AvaxImageEffect() = {
            New Avax.AvaxImageEffect(Avax.AvaxImageEffectKind.Exposure, 0.5),
            New Avax.AvaxImageEffect(Avax.AvaxImageEffectKind.Crop, 10.0)
        }
        Return cad.RenderImagePreview(handle, 512, proposed)
    End Function
End Module
```

Related options and reference

RenderImagePreviewAsync

API reference

Purpose

Renders a still image preview on a worker thread and returns its PNG bytes through a Task. The calling thread captures the image and effect values first, allowing a Windows Forms application to await the preview while continuing to process input.

Declaration

```
Function RenderImagePreviewAsync(ByVal handle As Integer,
                                 Optional ByVal maximumDimension As Integer = 640,
                                 Optional ByVal effects As AvaxImageEffect() = Nothing,
                                 Optional ByVal cancellationToken As System.Threading.CancellationToken =
    Nothing) As Task(Of Byte())
```

The return type is `System.Threading.Tasks.Task(Of Byte())`. Omitting cancellationToken uses a token without cancellation, equivalent to CancellationToken.None.

Parameters

Name	Meaning
handle	Integer handle of a supported still-bitmap shape in the current drawing. The control must be started and undisposed.
maximumDimension	Maximum preview width or height in pixels, from 1 through 16384 inclusive. Default 640. Aspect ratio is retained and smaller source images are not enlarged. The rendered preview canvas supports at most 32,000,000 pixels.
effects	Nothing captures the image's current stored effect list. An empty array requests an unfiltered preview. A supplied zero-based array provides an alternative list for this preview. It follows the same limits and amount ranges as SetImageEffects, including at most 32 entries and whole-number amounts for Pixelate and Posterize.
cancellationToken	A token supplied by the caller to cancel this preview request. Use a CancellationTokenSource when an application needs to cancel a superseded request.

Result and ownership

Awaiting a successful task returns a new Byte array containing the encoded PNG preview. The returned array belongs to the caller; changing it cannot alter the stored source image or another preview. Previewing leaves the image's source bytes, stored effects, geometry and Undo history unchanged.

An unknown or unsupported image handle, an unstarted or disposed control, or maximumDimension below 1 produces a completed task whose result is Nothing. A pre-canceled token takes precedence and produces a canceled task. Reading previews is available while the control is disabled or editing is locked, including supported images on hidden or frozen layers.

Threading and captured values

Call this method on the thread that owns the drawing control. The method reads the current image and validates and copies the requested effect values before returning the task. Pixel processing and PNG encoding then run on a worker thread without accessing the control.

Changes to the caller's effect array or its objects after the call returns do not change that request. Later drawing replacement, image edits or stored-effect changes leave the captured request intact. The completed bytes can therefore describe an earlier image state. Cancel requests that have been superseded before using their results in a preview window.

Closing or disposing the control after capture does not automatically cancel an existing request. The application owns its cancellation token source. A normal Await from a Windows Forms event handler resumes through that handler's synchronization context, allowing the caller to update its preview UI.

Cancellation

An already canceled token returns a canceled task immediately. Otherwise cancellation is cooperative: the renderer checks the token at processing boundaries and before returning the final bytes. A source decode, individual effect or PNG encoding step already in progress can finish before cancellation is observed. Awaiting a canceled task raises OperationCanceledException or its TaskCanceledException subtype.

Errors

Invalid supplied effect entries, nonfinite or out-of-range amounts, and invalid effect counts can raise ArgumentException synchronously when the method is called. Reading malformed stored effect values can also raise ArgumentException, FormatException or OverflowException before a task is returned.

Worker-side errors fault the returned task and are raised when it is awaited. These include ArgumentOutOfRangeException for maximumDimension above 16384, ArgumentException for a source that cannot be decoded or an oversized preview canvas, and rendering or encoding failures. Handle errors around both the call and Await.

For the synchronous preview method and the individual effect meanings, see RenderImagePreview and AvaxImageEffectKind.

VB.NET example

```
Option Strict On
Public Module RenderImagePreviewAsyncExample
    ' Call from the UI thread with an existing started drawing control.
    ' The caller owns the token source and can cancel a superseded request.
    Public Async Function CreateProposedPreviewAsync(
        cad As Avax.cAvax,
        imageHandle As Integer,
        cancellationToken As System.Threading.CancellationToken
    ) As System.Threading.Tasks.Task(Of Byte())
        Dim proposed As Avax.AvaxImageEffect() = {
            New Avax.AvaxImageEffect(Avax.AvaxImageEffectKind.Exposure, 0.5),
            New Avax.AvaxImageEffect(Avax.AvaxImageEffectKind.Crop, 10.0)
        }
        Return Await cad.RenderImagePreviewAsync(
            imageHandle, 640, proposed, cancellationToken)
    End Function
End Module
```

RenderParametricShapeIcon

Purpose

Creates a small transparent PNG preview of a preset for an application's shape gallery. It draws the preset's default parameters using fixed preview colors and does not create a drawing object.

Declaration

```
Shared Function RenderParametricShapeIcon(ByVal kind As String, Optional ByVal size As Integer = 32) As
    Byte()
```

Parameters and result

Name	Type	Meaning
kind	String	Case-sensitive Kind returned by either catalog method.
size	Integer	Width and height in pixels, from 16 to 256 inclusive. The default is 32.

Returns the encoded PNG bytes for a square image. The method is Shared and does not require a control instance or an active drawing. An unsupported Kind throws ArgumentException; an out-of-range size throws ArgumentOutOfRangeException. Decode the bytes when assigning the preview to a Windows Forms image control. If the decoding stream is closed, keep a separate Bitmap copy as in the example.

VB.NET example

```
Option Strict On
Public Module RenderParametricShapeIconExample
    Public Function CreateStarIcon() As System.Drawing.Bitmap
        Dim png As Byte() = Avax.cAvax.RenderParametricShapeIcon("Star", 48)
        Using stream As New System.IO.MemoryStream(png),
              decoded As System.Drawing.Image = System.Drawing.Image.FromStream(stream)
            Return New System.Drawing.Bitmap(decoded)
        End Using
    End Function
End Module
```

Related options and reference

RenderWorkspacePage

Purpose

Render a paper preview clipped to the printable area, without editor labels/shadow/guides. It uses PageBackgroundColor when PageBackgroundPrintExport is True, and white paper when False.

VB.NET declaration

```
Function RenderWorkspacePage(ByVal pixelWidth As Integer) As System.Drawing.Bitmap
```

Parameters

Parameter	Meaning
pixelWidth	Image width 1-16384 pixels. Aspect ratio follows paper; computed height must also be <=16384 and total pixels <=64,000,000.

Return value and behavior

A System.Drawing.Bitmap owned by the caller; dispose it. Its height is the paper aspect ratio rounded upward to a whole pixel. Width and height must each be at most 16384, with at most 64,000,000 pixels in total; invalid dimensions throw ArgumentOutOfRangeException before allocation.

Requires StartAvax. External-data providers may update geometry and page settings; the preview uses the accepted final settings and validates the resulting dimensions again. If a provider replaces or clears the drawing, ends or restarts the control, disables it, or changes event dispatch, the operation throws InvalidOperationException and returns no bitmap. Provider exceptions propagate unchanged. Newer callback settings and drawings are retained.

Some extreme physical paper sizes or drawing-to-paper magnifications exceed the native renderer range even when the bitmap dimensions are valid. Nonempty output in that range throws ArgumentException instead of returning an incorrectly blank preview. The returned bitmap has independent pixels and remains valid after the control is disposed.

Example

Call this procedure on the owning UI thread with a control already initialized by `StartAvax()`. Handles must belong to that control. File paths are supplied by the caller.

```
Option Strict On
Imports System
Imports Avax

Public Module Help_RenderWorkspacePage
    Public Sub Run(cad As cAvax)
        Using preview As System.Drawing.Bitmap = cad.RenderWorkspacePage(1200)
            Dim previewHeight As Integer = preview.Height
            ' Display or save inside this Using block, or transfer ownership explicitly.
        End Using
    End Sub
End Module
```

Related options and reference

Reset_DimensionTextPosition

Purpose

Restore automatic text positioning using the effective dimension style.

VB.NET declaration

```
Function Reset_DimensionTextPosition(ByVal handle As Integer) As Boolean
```

Parameters

Parameter	Meaning
handle	Handle of an existing native dimension in this control.

Return value and behavior

True on success; False on failure. Keeps the dimension handle and supports Undo/Redo.

Example

Call this procedure on the owning UI thread with a control already initialized by `StartAvax()`. Handles must belong to that control. File paths are supplied by the caller.

```
Option Strict On
Imports System
Imports Avax

Public Module Help_Reset_DimensionTextPosition
    Public Sub Run(cad As cAvax, handle As Integer)
        If Not cad.Reset_DimensionTextPosition(handle) Then Throw New InvalidOperationException("Text
    reset failed.")
    End Sub
End Module
```

Related options and reference

ResetImageEffects

Purpose

Removes all enabled and disabled effects from one image, restoring its unfiltered appearance from the retained source image.

Declaration

```
Function ResetImageEffects(ByVal handle As Integer) As Boolean
```

Parameters and result

Name	Meaning
handle	Integer handle of the image to reset.

Returns the result of SetImageEffects with an empty effect array. Its started/enabled/unlocked and supported visible-target requirements apply. A successful reset participates in normal Undo/Redo and preserves the stored source bytes, image position and outline. It also reverses the displayed result of Crop.

VB.NET example

```
Option Strict On
Public Module ResetImageEffectsExample
    Public Function RestoreUnfilteredImage(cad As Avax.cAvax,
                                           handle As Integer) As Boolean
        Return cad.ResetImageEffects(handle)
    End Function
End Module
```

Related options and reference

ReverseCompoundPath

Purpose

Reverses the traversal direction of every figure of a path while retaining its geometric outline.

Declaration

```
Function ReverseCompoundPath(ByVal handle As Integer) As Boolean
```

Parameters and result

`handle` identifies an object readable by GetCompoundPath. Returns True for an accepted edit, False if no path can be read or the object cannot be edited. Each figure begins at its former final endpoint; its segment order reverses. Bezier controls swap, arc sweep changes sign and node modes follow their corresponding nodes. Figure order, Closed and EvenOddFill are retained. Re-read segment indices and control-point identifiers after reversing. The object retains its handle and drawing attributes; a parametric recipe is removed and the edit participates in Undo/Redo.

VB.NET example

```
Option Strict On
Public Module ReverseCompoundPathExample
    Public Function AddReversedPath(cad As Avax.cAvax) As Integer
        Dim path As New Avax.AvaxPathData
        Dim figure As New Avax.AvaxPathFigure
        figure.Segments.Add(Avax.AvaxPathSegment.LineTo(20.0, 0.0))
        figure.Segments.Add(Avax.AvaxPathSegment.LineTo(20.0, 10.0))
        path.Figures.Add(figure)
        Dim handle As Integer = cad.AddCompoundPath(path)
        If handle = 0 OrElse Not cad.ReverseCompoundPath(handle) Then Return 0
        Dim reversed As Avax.AvaxPathData = cad.GetCompoundPath(handle)
        System.Diagnostics.Debug.WriteLine(reversed.Figures(0).X)
        Return handle
    End Function
End Module
```

Related options and reference

RotateItems

cAvax method

Purpose

Rotates objects counterclockwise about an XY base point, using degrees. The Stay overload can retain originals and return copies.

VB.NET declarations

```
Function RotateItems(ByRef avMyHandles As Integer(), ByRef RotX As Double, ByRef RotY As Double, ByRef
    theta As Double, ByRef fRotateStay As Boolean) As Boolean

Function RotateItems(ByRef avMyHandles As Integer(), ByRef RotX As Double, ByRef RotY As Double, ByRef
    theta As Double) As Boolean
```

Parameters

Name	Type and passing	Meaning
avMyHandles	ByRef avMyHandles As Integer()	Input Integer() of the entity handles to process. Logical entries start at index 1; index 0 is reserved.
RotX	ByRef RotX As Double	Is the x coordinate of the rotation base point
RotY	ByRef RotY As Double	Is the y coordinate of the rotation base point
theta	ByRef theta As Double	Is the rotation angle
fRotateStay	ByRef fRotateStay As Boolean	A Boolean expression that if it is true then the original objects are retained

Use the declaration for the overload you call. ByRef arguments can receive outputs; declare variables and arrays with the exact indicated type. Optional defaults are shown in the declaration.

Return value

Boolean - True on success, false on error.

Remarks

Angles are counterclockwise degrees about (RotX, RotY). The short overload uses fRotateStay=False and changes the existing objects. Finite degree values are reduced modulo 360 before conversion to radians; exact quarter and half turns use exact zero/unit coefficients. Pivot offsets and complete affine point expressions are rounded once at their Double storage boundaries. A required matrix offset or resulting geometry outside the finite Double range rejects the complete batch.

With fRotateStay=True, originals remain and the caller ByRef handle array receives the copied handles. The original array object is not overwritten. Z coordinates, object attributes, embedded resources and nested content are retained. Live external geometry retains the transformation when regenerated; native dimensions and hatches update their definitions. Document text follows FixedDocumentTextSizeOn_p: True keeps its drawing-unit font height, while False transforms the font with the document frame.

Pass a one-based Integer handle array; index 0 is ignored and duplicate handles are processed once. Geometry uses Double. The control must be started and enabled. PauseAvax and LockEdit govern interactive input and do not prohibit this direct call. Before events can change the proposed targets or parameters, or cancel the operation. Caller arrays are separate from event arrays. The complete batch is prepared before editing. Invalid handles, unrepresentable geometry or unavailable copy capacity return False without a partial edit. A successful batch is one Undo operation and raises Change without NewUserItems. PauseAvaxEvents suppresses notifications. Application event-handler exceptions propagate; they do not roll back an edit already committed. A callback that replaces the drawing or starts new input prevents the older operation from overwriting that state.

Numeric changes from v2

Update the following variables, arrays or event-data receivers when porting the corresponding v2 call. The declarations above show the current overloads.

Parameter or value	v2 type	v3 type
RotX	Single	Double

Parameter or value	v2 type	v3 type
RotY	Single	Double
theta	Single	Double

For ByRef parameters, use a variable with the exact v3 type. Preserve array dimensions and the documented indexing. Moving from v2 to v3 explains conversion and legacy file-storage rules.

Example

Setup

Add an Avax.cAvax Windows Forms control named `cAvax1`. Paste the procedures into the form class. Example setup and supporting files explains references, initialization and file paths.

Add these Windows Forms controls or components to the form. Fields referenced by Handles clauses must be declared WithEvents, as the Visual Studio designer normally does.

Name	Type
Command1	Button

VB.NET example

```
Private Sub Form1_Load(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs) Handles
    MyBase.Load
    cAvax1.StartAvax()
    Call cAvax1.Add_Line(0, 0, 0, 10, 10, 0)
    Call cAvax1.Add_Line(0, 10, 0, 10, 10, 0)
End Sub
Private Sub Form1_FormClosed(ByVal eventSender As System.Object, ByVal eventArgs As
    System.Windows.Forms.FormClosedEventArgs) Handles Me.FormClosed
    cAvax1.EndAvax()
End Sub
Private Sub Command1_Click(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs)
    Handles Command1.Click
    Dim [Handles]() As Integer = Nothing
    Dim RotX As Double
    Dim RotY As Double
    Dim theta As Double
    Dim fRotateStay As Boolean
    Dim Answer As Boolean
    If CAvax1.GetAllHandlesArr([Handles]) > 0 Then
        RotX = 10
        RotY = 10
        theta = 60
        fRotateStay = True
        Answer = CAvax1.RotateItems([Handles], RotX, RotY, theta, fRotateStay)
    End If
End Sub
```

SaveAs

cAvax method

Purpose

Saves the current drawing under the specified file name. Use the native .avax extension for a version 3 drawing; .avx is the legacy interchange format.

VB.NET declaration

```
Function SaveAs(ByVal sNewFile As String) As Boolean
```

Parameters

Name	Type and passing	Meaning
sNewFile	ByVal sNewFile As String	Is the new drawing file name

Use the declaration for the overload you call. ByRef arguments can receive outputs; declare variables and arrays with the exact indicated type. Optional defaults are shown in the declaration.

Return value

Boolean - True on success, false on error.

Remarks

Saves the current drawing and returns True after the destination has been written successfully. Returns False for invalid paths or data, write failures, unavailable controls or a superseded save. Validation and I/O failures report an error. Exceptions thrown by a host Progress handler propagate to the caller. Relative paths use the data directory configured by DirsSet. A name without an extension receives .avx. Use .avax to retain native Double precision; .avx and .blk use the R2 binary formats and reject values that cannot be represented. The native format records its own version number. A reader rejects a file whose format version is newer than that reader supports; the file-format version is separate from the DLL version. The complete snapshot is captured before the first Progress callback. Edits made during callbacks remain in the live drawing and keep it modified; they are not mixed into that snapshot. Temporary output is read back before replacing an existing destination. A failed save therefore retains the previous file. A newer file request or document replacement cancels the pending operation. Plain text-filled shapes follow the current OriginTopLeft_p setting in the legacy representation, including text inside nested blocks. Use the same origin mode in the receiving R2 control to retain the displayed orientation. Exporting does not reorder the source drawing geometry. Native .avax files preserve imported polyline centreline and width vectors together. Legacy AVX/BLK files represent these widths with filled geometry in a block. Native XML data retains the editable profile; its R2-compatible presentation contains the filled geometry. Native format 8 stores retained CAD polyline vertices, bulges, widths and their plane when present. Files without that information continue to use the minimum required earlier format. A runtime supporting only format 7 or earlier cannot open a format 8 drawing.

Example

Setup

Add an Avax.cAvax Windows Forms control named `cAvax1`. Paste the procedures into the form class. Example setup and supporting files explains references, initialization and file paths.

Add these Windows Forms controls or components to the form. Fields referenced by Handles clauses must be declared WithEvents, as the Visual Studio designer normally does.

Name	Type
Command1	Button

VB.NET example

```
Private Sub Form1_Load(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs) Handles
    MyBase.Load
    Command1.Text = "SaveAs"
```

```
    cAvax1.StartAvax()
    Call cAvax1.Add_Rectangle(0, 0, 10, 10)
    Call cAvax1.Add_Rectangle(1, 1, 8, 8)
    Call cAvax1.Add_Rectangle(2, 2, 6, 6)
End Sub
Private Sub Form1_FormClosed(ByVal eventSender As System.Object, ByVal eventArgs As
    System.Windows.Forms.FormClosedEventArgs) Handles Me.FormClosed
    cAvax1.EndAvax()
End Sub
Private Sub Command1_Click(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs)
    Handles Command1.Click
    Dim sNewFile As String = String.Empty
    Dim Answer As Boolean
    sNewFile = My.Application.Info.DirectoryPath & "\new.avx"
    Answer = cAvax1.SaveAs(sNewFile)
End Sub
```

SaveAvaxPropertiesToXML

cAvax method

Purpose

Use this method if you want to get the properties of avax control as a XML document

See Drawing coordinate limits for Double coordinate properties, individual versus complete work-area assignment, and transactional settings validation.

VB.NET declaration

```
Function SaveAvaxPropertiesToXML() As System.Xml.XmlDocument
```

Return value

Xml.XmlDocument - Is the XML document

Remarks

Returns an independently owned System.Xml.XmlDocument containing the current AvaxProp settings,or Nothing when serialization is unavailable or a setting cannot be represented. Uses the R2 AVAX-Properties/General/Properties settings schema with General Version=1.0. This version is separate from native .avax drawing version3. AvaxProp numbers use invariant Double/Integer values and Boolean -1/0. FillType retains its seven attributes; MouseDrawing contains one Line element per one-based segment,with seven invariant Double values. Settings XML contains control settings,not drawing objects or resource tables. Standalone properties outside AvaxProp are not part of this legacy schema. The document can be saved to a file,Stream or TextWriter. Unicode,entity text,tabs,line breaks and whitespace-only strings round-trip without trimming. XMLencoding_p controls file/stream encoding; an empty value uses UTF-8. Streams and TextWriters remain owned and open by the caller. Save(String) stages and flushes the complete output before atomically replacing the destination; failure preserves the existing file and throws the normal save exception. Binary legacy hatch Pattern strings are preserved as exact UTF-16 code units: when Pattern contains XML-invalid characters,the FillType_p element has PatternEncoding="base64-utf16le" and a base64 Pattern attribute. Printable Pattern text retains the original unmarked R2 representation. Encoded binary patterns require the v3 loader. Other XML-invalid string settings are rejected rather than silently changed. Settings can be serialized while stopped or disabled;a disposed control returns Nothing.

Example

Setup

Add an Avax.cAvax Windows Forms control named `cAvax1`. Paste the procedures into the form class. Example setup and supporting files explains references, initialization and file paths.

Add these Windows Forms controls or components to the form. Fields referenced by Handles clauses must be declared WithEvents, as the Visual Studio designer normally does.

Name	Type
Button1	Button

VB.NET example

```
Private Sub Form1_Load(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs) Handles
    MyBase.Load
    Call cAvax1.StartAvax()
End Sub
Private Sub Form1_FormClosed(ByVal eventSender As System.Object, ByVal eventArgs As
    System.Windows.Forms.FormClosedEventArgs) Handles Me.FormClosed
    Call cAvax1.EndAvax()
End Sub
Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
    Button1.Click
    Dim xmlDoc As Xml.XmlDocument
    xmlDoc = cAvax1.SaveAvaxPropertiesToXML
    xmlDoc.Save(My.Application.Info.DirectoryPath & "\test.xml")
```

```
End Sub
```

SaveFonts

cAvax method

Purpose

Saves the font table to a external font file

VB.NET declaration

```
Function SaveFonts(ByRef sFontFile As String) As Boolean
```

Parameters

Name	Type and passing	Meaning
sFontFile	ByRef sFontFile As String	ByRef destination FNT path, unchanged by the call. Existing output is replaced only after successful flush and readback verification. ANSI when lossless, otherwise the Unicode v3 form.

Use the declaration for the overload you call. ByRef arguments can receive outputs; declare variables and arrays with the exact indicated type. Optional defaults are shown in the declaration.

Return value

Boolean - True on success, false on error.

Example

Setup

Add an Avax.cAvax Windows Forms control named `cAvax1`. Paste the procedures into the form class. Example setup and supporting files explains references, initialization and file paths.

Add these Windows Forms controls or components to the form. Fields referenced by Handles clauses must be declared WithEvents, as the Visual Studio designer normally does.

Name	Type
Command1	Button
Text1	TextBox

VB.NET example

```
Private Sub Form1_Load(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs) Handles
    MyBase.Load
    Command1.Text = "SaveFonts"
    cAvax1.StartAvax()
End Sub
Private Sub Form1_FormClosed(ByVal eventSender As System.Object, ByVal eventArgs As
    System.Windows.Forms.FormClosedEventArgs) Handles Me.FormClosed
    cAvax1.EndAvax()
End Sub
Private Sub Command1_Click(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs)
    Handles Command1.Click
    Dim sFontFile As String = String.Empty
    Dim iFr As Integer
    Dim sTmp As String = String.Empty
    Call SetScreenFonts()
    sFontFile = My.Application.Info.DirectoryPath & "\myFonts.fnt"
    If cAvax1.SaveFonts(sFontFile) = True Then
        iFr = FreeFile
        sTmp = Space(FileLen(sFontFile))
        FileOpen(iFr, sFontFile, OpenMode.Binary)
```

```
        FileGet(iFr, sTmp)
        Text1.Text = sTmp
        FileClose(iFr)
    End If
    Command1.Enabled = False
End Sub
Function SetScreenFonts() As Boolean
    Dim sFonts() As String = Nothing
    Dim lFontsSet() As Integer = Nothing
    Dim iCnt, lMax As Integer
    lMax = System.Drawing.FontFamily.Families.Length
    If lMax > 256 Then lMax = 256
    ReDim sFonts(lMax)
    ReDim lFontsSet(lMax)
    For iCnt = 0 To lMax - 1
        sFonts(iCnt + 1) = System.Drawing.FontFamily.Families(iCnt).Name
        lFontsSet(iCnt + 1) = 0
    Next iCnt
    SetScreenFonts = cAvax1.SetFonts(sFonts, lFontsSet)
End Function
```

SaveLineStyles

cAvax method

Purpose

Saves the current line types data to an external file

VB.NET declaration

```
Function SaveLineStyles(ByRef sAVAXLinFile As String) As Boolean
```

Parameters

Name	Type and passing	Meaning
sAVAXLinFile	ByRef sAVAXLinFile As String	ByRef destination path, unchanged. Native ANSI when lossless, otherwise BOM Unicode; replacement occurs only after flush and successful readback.

Use the declaration for the overload you call. ByRef arguments can receive outputs; declare variables and arrays with the exact indicated type. Optional defaults are shown in the declaration.

Return value

Boolean - True on success, false on error.

Example

Setup

Add an Avax.cAvax Windows Forms control named `cAvax1`. Paste the procedures into the form class. Example setup and supporting files explains references, initialization and file paths.

Add these Windows Forms controls or components to the form. Fields referenced by Handles clauses must be declared WithEvents, as the Visual Studio designer normally does.

Name	Type
Command1	Button
Text1	TextBox

VB.NET example

```
Private Sub Form1_Load(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs) Handles
    MyBase.Load
    Command1.Text = "Save/Load LineStyles"
    cAvax1.StartAvax()
End Sub
Private Sub Form1_FormClosed(ByVal eventSender As System.Object, ByVal eventArgs As
    System.Windows.Forms.FormClosedEventArgs) Handles Me.FormClosed
    cAvax1.EndAvax()
End Sub
Private Sub Command1_Click(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs)
    Handles Command1.Click
    Dim sAVAXLinFile As String = String.Empty
    Dim iFr As Integer
    Dim sTmp As String = String.Empty
    sAVAXLinFile = My.Application.Info.DirectoryPath & "\AvaxLineStyle.txt"
    If cAvax1.SaveLineStyles(sAVAXLinFile) = True Then
        If cAvax1.LoadLineStyles(sAVAXLinFile) = True Then
            iFr = FreeFile
            FileOpen(iFr, sAVAXLinFile, OpenMode.Binary)
            sTmp = Space(FileLen(sAVAXLinFile))
```

```
            FileGet(iFr, sTmp)
            FileClose(iFr)
            Text1.Text = sTmp
        End If
    End If
End Sub
```

SavePalette

cAvax method

Purpose

Saves the current palette to an external palette file

VB.NET declaration

```
Function SavePalette(ByRef sPalFile As String) As Boolean
```

Parameters

Name	Type and passing	Meaning
sPalFile	ByRef sPalFile As String	Output PAL path, preserved ByRef. Existing bytes are replaced only after the complete RGB table is written and verified.

Use the declaration for the overload you call. ByRef arguments can receive outputs; declare variables and arrays with the exact indicated type. Optional defaults are shown in the declaration.

Return value

Boolean - True on success, false on error.

Example

Setup

Add an Avax.cAvax Windows Forms control named `cAvax1`. Paste the procedures into the form class. Example setup and supporting files explains references, initialization and file paths.

Add these Windows Forms controls or components to the form. Fields referenced by Handles clauses must be declared WithEvents, as the Visual Studio designer normally does.

Name	Type
Command1	Button
Text1	TextBox

VB.NET example

```
Private Sub Form1_Load(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs) Handles
    MyBase.Load
    Command1.Text = "SavePalette"
    cAvax1.StartAvax()
End Sub
Private Sub Form1_FormClosed(ByVal eventSender As System.Object, ByVal eventArgs As
    System.Windows.Forms.FormClosedEventArgs) Handles Me.FormClosed
    cAvax1.EndAvax()
End Sub
Private Sub Command1_Click(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs)
    Handles Command1.Click
    Dim sPalFile As String = String.Empty
    Dim iFr As Integer
    Dim sTmp As String = String.Empty
    sPalFile = My.Application.Info.DirectoryPath & "\myPalette.pal"
    If cAvax1.SavePalette(sPalFile) = True Then
        iFr = FreeFile
        sTmp = Space(FileLen(sPalFile))
        FileOpen(iFr, sPalFile, OpenMode.Binary)
        FileGet(iFr, sTmp)
        Text1.Text = sTmp
```

```
        FileClose(iFr)
    End If
End Sub
```

ScaleItems

cAvax method

Purpose

Scales objects independently in X and Y about a base point. The Stay overload can retain originals and return copies.

VB.NET declarations

```
Function ScaleItems(ByRef avMyHandles As Integer(), ByRef SclBaseX As Double, ByRef SclBaseY As Double,
    ByRef ScaleX As Double, ByRef ScaleY As Double) As Boolean

Function ScaleItems(ByRef avMyHandles As Integer(), ByRef SclBaseX As Double, ByRef SclBaseY As Double,
    ByRef ScaleX As Double, ByRef ScaleY As Double, ByRef fScaleStay As Boolean) As Boolean
```

Parameters

Name	Type and passing	Meaning
avMyHandles	ByRef avMyHandles As Integer()	Input Integer() of the entity handles to process. Logical entries start at index 1; index 0 is reserved.
SclBaseX	ByRef SclBaseX As Double	Is the x coordinate of the scale base point
SclBaseY	ByRef SclBaseY As Double	Is the y coordinate of the scale base point
ScaleX	ByRef ScaleX As Double	Is the scale x factor
ScaleY	ByRef ScaleY As Double	Is the scale y factor
fScaleStay	ByRef fScaleStay As Boolean	A Boolean expression that if it is true then the original objects are retained

Use the declaration for the overload you call. ByRef arguments can receive outputs; declare variables and arrays with the exact indicated type. Optional defaults are shown in the declaration.

Return value

Boolean - True on success, false on error.

Remarks

SclBaseX and SclBaseY define the fixed base point. X and Y scale factors are independent; negative factors reflect. A zero factor inherits the other factor, while two zeros return False. Pivot offsets are calculated from the complete expressions baseX-baseX*ScaleX and baseY-baseY*ScaleY, with one rounding to Double. A required matrix offset or resulting geometry outside the finite Double range rejects the complete batch.

The short overload uses fScaleStay=False. With fScaleStay=True, originals remain and the caller ByRef handle array receives the copied handles. The original array object is not overwritten. Z coordinates, object attributes, embedded resources and nested content are retained. Live external geometry retains the transformation when regenerated; native dimensions and hatches update their definitions. Document text follows FixedDocumentTextSizeOn_p: True keeps its drawing-unit font height, while False transforms the font with the document frame.

Pass a one-based Integer handle array; index 0 is ignored and duplicate handles are processed once. Geometry uses Double. The control must be started and enabled. PauseAvax and LockEdit govern interactive input and do not prohibit this direct call. Before events can change the proposed targets or parameters, or cancel the operation. Caller arrays are separate from event arrays. The complete batch is prepared before editing. Invalid handles, unrepresentable geometry or unavailable copy capacity return False without a partial edit. A successful batch is one Undo operation and raises Change without NewUserItems. PauseAvaxEvents suppresses notifications. Application event-handler exceptions propagate; they do not roll back an edit already committed. A callback that replaces the drawing or starts new input prevents the older operation from overwriting that state.

Numeric changes from v2

Update the following variables, arrays or event-data receivers when porting the corresponding v2 call. The declarations above show the current overloads.

Parameter or value	v2 type	v3 type
SclBaseX	Single	Double
SclBaseY	Single	Double
ScaleX	Single	Double
ScaleY	Single	Double

For ByRef parameters, use a variable with the exact v3 type. Preserve array dimensions and the documented indexing. Moving from v2 to v3 explains conversion and legacy file-storage rules.

Example

Setup

Add an Avax.cAvax Windows Forms control named `cAvax1`. Paste the procedures into the form class. Example setup and supporting files explains references, initialization and file paths.

Add these Windows Forms controls or components to the form. Fields referenced by Handles clauses must be declared WithEvents, as the Visual Studio designer normally does.

Name	Type
Command1	Button

VB.NET example

```
Private Sub Form1_Load(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs) Handles
    MyBase.Load
    cAvax1.StartAvax()
    Call cAvax1.Add_Line(0, 0, 0, 10, 10, 0)
    Call cAvax1.Add_Line(0, 10, 0, 10, 10, 0)
End Sub
Private Sub Form1_FormClosed(ByVal eventSender As System.Object, ByVal eventArgs As
    System.Windows.Forms.FormClosedEventArgs) Handles Me.FormClosed
    cAvax1.EndAvax()
End Sub
Private Sub Command1_Click(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs)
    Handles Command1.Click
    Dim [Handles]() As Integer = Nothing
    Dim sclx1 As Double
    Dim scly1 As Double
    Dim sclx2 As Double
    Dim scly2 As Double
    Dim fScaleStay As Boolean
    Dim Answer As Boolean
    If CAvax1.GetAllHandlesArr([Handles]) > 0 Then
        sclx1 = 0
        scly1 = 0
        sclx2 = 1.5
        scly2 = 1.5
        fScaleStay = True
        Answer = CAvax1.ScaleItems([Handles], sclx1, scly1, sclx2, scly2, fScaleStay)
    End If
End Sub
```

SelectItems

cAvax method

Purpose

Selects objects with handles described in a specific handles array

See Selection commands and type filters for one-based handle arrays, point/window selection, modifier keys, type-mask values, and committed selection notifications.

See Immediate actions and selection borders for direct mode changes, filtered selection, one overall Border, outputs, callbacks and accepted Polyline input.

VB.NET declarations

```
Function SelectItems(ByRef avMyHandles As Integer()) As Boolean

Function SelectItems(ByRef avMyHandles As Integer(), ByRef ItemType As AvaxItemType) As Boolean
```

Parameters

Name	Type and passing	Meaning
avMyHandles	ByRef avMyHandles As Integer()	ByRef one-based Integer array, dummy slot 0. Nothing or length 1 expands to all root handles; length 0 clears selection. Explicit arrays stay intact; duplicates/missing handles are skipped.
ItemType	ByRef ItemType As AvaxItemType	An expression that defines the type of the objects

Use the declaration for the overload you call. ByRef arguments can receive outputs; declare variables and arrays with the exact indicated type. Optional defaults are shown in the declaration.

Return value

Boolean - True on success, false on error.

Example

Setup

Add an Avax.cAvax Windows Forms control named `cAvax1`. Paste the procedures into the form class. Example setup and supporting files explains references, initialization and file paths.

Add these Windows Forms controls or components to the form. Fields referenced by Handles clauses must be declared WithEvents, as the Visual Studio designer normally does.

Name	Type
Command1	Button

VB.NET example

```
Dim h1 As Integer
Dim h2 As Integer
Private Sub Form1_Load(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs) Handles
    MyBase.Load
    cAvax1.StartAvax()
    h1 = cAvax1.Add_Line(0, 0, 0, 10, 10, 0)
    h2 = cAvax1.Add_Line(0, 10, 0, 10, 0, 0)
End Sub
Private Sub Form1_FormClosed(ByVal eventSender As System.Object, ByVal eventArgs As
    System.Windows.Forms.FormClosedEventArgs) Handles Me.FormClosed
    cAvax1.EndAvax()
End Sub
Private Sub Command1_Click(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs)
```

```
    Handles Command1.Click
    Dim [Handles]() As Integer = Nothing
    ReDim [Handles](2)
    [Handles](1) = h1
    [Handles](2) = h2
    Call CAvax1.SelectItems([Handles])
End Sub
```

SelectPoints

cAvax method

Purpose

Activates the Select Points process with a specific key and the number of the points to be selected

VB.NET declarations

```
Function SelectPoints(ByRef sSelectId As String, ByRef lPointsToSelect As Integer, ByRef ExistX As
    Double(), ByRef ExistY As Double(), ByRef ExistZ As Double(), ByRef fBoxOn As Boolean, ByRef
    fNoSnapAndOrtho As Boolean) As Boolean

Function SelectPoints(ByRef sSelectId As String, ByRef lPointsToSelect As Integer, ByRef ExistX As
    Double(), ByRef ExistY As Double(), ByRef ExistZ As Double()) As Boolean
```

Parameters

Name	Type and passing	Meaning
sSelectId	ByRef sSelectId As String	Is the key name you want to come in the event SelectedPoints after the selection of a point
lPointsToSelect	ByRef lPointsToSelect As Integer	Is the the total number of the points that you want to select
ExistX	ByRef ExistX As Double()	An array for the x coordinates of the points that you want to be preselected
ExistY	ByRef ExistY As Double()	An array for the y coordinates of the points that you want to be preselected
ExistZ	ByRef ExistZ As Double()	An array for the z coordinates of the points that you want to be preselected
fBoxOn	ByRef fBoxOn As Boolean	A Boolean expression that defines if in the case where we want two points, the second point will be a parallelogram apex
fNoSnapAndOrtho	ByRef fNoSnapAndOrtho As Boolean	A Boolean parameter that determines if Ortho and Snap are enabled during the selection of points

Use the declaration for the overload you call. ByRef arguments can receive outputs; declare variables and arrays with the exact indicated type. Optional defaults are shown in the declaration.

Return value

Boolean - True on success, false on error.

Remarks

Starts an identified request for a total number of points, including preselected seeds. XYZ arrays are one-based Double arrays: logical points start at index 1 and index 0 is reserved. Seeds are copied; later caller edits do not change the request. The short overload uses fBoxOn=False and fNoSnapAndOrtho=True. With fNoSnapAndOrtho=False, grid and Ortho constrain accepted coordinates; True bypasses those constraints while explicit object snaps remain available. fBoxOn applies only to a two-point request and previews a rectangle with both diagonals. All-Nothing seeds start an empty request; valid X/Y arrays with omitted Z use zero Z values. Targets must be positive and exceed the seed count. Invalid, mismatched or nonfinite seeds and inactive, disabled, disposed, paused or edit-locked controls return False without replacing existing input. Each accepted point raises SelectedPoints with cumulative coordinates. Final completion clears the active request before notification. Grid/Snap/Ortho aids retain the pending request; Escape, cancellation, a replacement drafting tool or a new drawing discards it without an accepted-point notification. A callback may replace the command, request or drawing; returning input from the previous operation does not overwrite the replacement. The method collects coordinates without creating drawing entities or adding Undo records.

Numeric changes from v2

Update the following variables, arrays or event-data receivers when porting the corresponding v2 call. The declarations above show the current overloads.

Parameter or value	v2 type	v3 type
ExistX	Single()	Double()
ExistY	Single()	Double()
ExistZ	Single()	Double()

For ByRef parameters, use a variable with the exact v3 type. Preserve array dimensions and the documented indexing. Moving from v2 to v3 explains conversion and legacy file-storage rules.

Example

Setup

Add an Avax.cAvax Windows Forms control named `cAvax1`. Paste the procedures into the form class. Example setup and supporting files explains references, initialization and file paths.

Add these Windows Forms controls or components to the form. Fields referenced by Handles clauses must be declared WithEvents, as the Visual Studio designer normally does.

Name	Type
Command1	Button

VB.NET example

```
Private Sub CAvax1_SelectedPoints(ByVal Sender As Object, ByVal e As Avax.cAvax.SelectedPointsEventArgs)
    Handles CAvax1.SelectedPoints
    Select Case e.SelectId
        Case "Create triangle"
        If e.lPointNumber = 3 Then
            Call CAvax1.Add_Shape(e.x1, e.y1, e.z1, Nothing, 1, 3, 2, 0, "", 0, 0, -1, -1, -1, -1,
    Avax.AvaxPolylineType.Polyline, pt)
        End If
    End Select
End Sub
Private Sub Form1_Load(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs) Handles
    MyBase.Load
    Command1.Text = "Create triangle"
    cAvax1.StartAvax()
End Sub
Private Sub Form1_FormClosed(ByVal eventSender As System.Object, ByVal eventArgs As
    System.Windows.Forms.FormClosedEventArgs) Handles Me.FormClosed
    cAvax1.EndAvax()
End Sub
Private Sub Command1_Click(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs)
    Handles Command1.Click
    Dim sSelectId As String = String.Empty
    Dim lPointsToSelect As Integer
    Dim ExistX() As Double = Nothing
    Dim ExistY() As Double = Nothing
    Dim ExistZ() As Double = Nothing
    Dim fBoxOn As Boolean
    Dim fNoSnapAndOrtho As Boolean
    Dim Answer As Boolean
    lPointsToSelect = 3
    ReDim ExistX(1)
    ReDim ExistY(1)
    ReDim ExistZ(1)
    ExistX(1) = 3
    ExistY(1) = 3
    sSelectId = "Create triangle"
```

```
    Answer = cAvax1.SelectPoints(sSelectId, lPointsToSelect, ExistX, ExistY, ExistZ, fBoxOn,
    fNoSnapAndOrtho)
End Sub
```

Set_Dimension

Purpose

Validate and replace a native dimension definition while keeping its handle.

VB.NET declaration

```
Function Set_Dimension(ByVal handle As Integer, ByVal data As AvaxDimensionData) As Boolean
```

Parameters

Parameter	Meaning
handle	Handle of an existing native dimension in this control.
data	Complete detached definition, including Style and Options. The method copies it.

Return value and behavior

True on success; False on invalid input. Both the complete named Style and the effective combination of Style, StyleOverrides and TextHeight must be valid. An invalid base style is rejected even when a per-object override would hide its invalid value. Failure changes neither the dimension nor the style table. Successful replacement is one undoable change. If the supplied named style differs from an existing style with the same name, a unique style name is assigned; other dimensions keep their style. Use Set_DimensionStyle(style) to update a named style for all its dimensions. ChangeColor/ChangeWidth/ChangeLayer/ChangeLineStyle edit the common object appearance; Options pen values are creation inputs.

Example

Call this procedure on the owning UI thread with a control already initialized by `StartAvax()`. Handles must belong to that control. File paths are supplied by the caller.

```
Option Strict On
Imports System
Imports Avax

Public Module Help_Set_Dimension
    Public Sub Run(cad As cAvax, handle As Integer)
        Dim data = cad.Get_Dimension(handle)
        If data Is Nothing Then Throw New ArgumentException("Native dimension required.")
        data.Options.TextOverride = "Length = <>"
        If Not cad.Set_Dimension(handle, data) Then Throw New InvalidOperationException("Dimension update
    failed.")
    End Sub
End Module
```

Related options and reference

Set_DimensionStyle

Purpose

Register/update a named style, or assign an existing named style to one dimension.

VB.NET declaration

```
Function Set_DimensionStyle(ByVal handle As Integer, ByVal styleName As String) As Boolean

Function Set_DimensionStyle(ByVal style As AvaxDimensionStyle) As Boolean
```

Parameters

Parameter	Meaning
handle	Handle of an existing native dimension in this control.
styleName	An existing style name, or AVAX_STANDARD, to assign to the supplied handle.
style	Complete named style to register. Dimensions using this name are regenerated in one undo operation.

Return value and behavior

True on success; False on failure. Per-object StyleOverrides and positive TextHeight remain effective; clear the relevant overrides through Set_Dimension when style inheritance is desired.

Example

Call this procedure on the owning UI thread with a control already initialized by `StartAvax()`. Handles must belong to that control. File paths are supplied by the caller.

```
Option Strict On
Imports System
Imports Avax

Public Module Help_Set_DimensionStyle
    Public Sub Run(cad As cAvax, handle As Integer)
        Dim style = cad.Get_DimensionStyle()
        style.Name = "MY_DIM"
        style.TextHeight = 3
        style.ArrowSize = 2
        If Not cad.Set_DimensionStyle(style) Then Throw New InvalidOperationException("Style registration
    failed.")
        If Not cad.Set_DimensionStyle(handle, style.Name) Then Throw New InvalidOperationException("Style
    assignment failed.")
    End Sub
End Module
```

Related options and reference

Set_DimensionTextPosition

Purpose

Move the dimension text to explicit drawing coordinates and enable manual placement.

VB.NET declaration

```
Function Set_DimensionTextPosition(ByVal handle As Integer, ByVal tx As Double, ByVal ty As Double) As
    Boolean
```

Parameters

Parameter	Meaning
handle	Handle of an existing native dimension in this control.
tx	Text-position X in drawing units.
ty	Text-position Y in drawing units.

Return value and behavior

True on success; False on failure. The current TextMoveMode governs line/leader behavior. The change is undoable.

Example

Call this procedure on the owning UI thread with a control already initialized by `StartAvax()`. Handles must belong to that control. File paths are supplied by the caller.

```
Option Strict On
Imports System
Imports Avax

Public Module Help_Set_DimensionTextPosition
    Public Sub Run(cad As cAvax, handle As Integer)
        If Not cad.Set_DimensionTextPosition(handle, 60, 40) Then Throw New
    InvalidOperationException("Text move failed.")
    End Sub
End Module
```

Related options and reference

Set_Hatch

Purpose

Validate and replace a hatch definition without changing its handle.

VB.NET declaration

```
Function Set_Hatch(ByVal handle As Integer, ByVal data As AvaxHatchData) As Boolean
```

Parameters

Parameter	Meaning
handle	Handle of an existing native hatch in this control.
data	Complete detached data with Options and valid closed Loops. Copied on success.

Return value and behavior

True on success; False on failure. Records one undoable edit. Changing elevation independently disconnects association. Reconnecting sources should use Set_HatchBoundaries.

Example

Call this procedure on the owning UI thread with a control already initialized by `StartAvax()`. Handles must belong to that control. File paths are supplied by the caller.

```
Option Strict On
Imports System
Imports Avax

Public Module Help_Set_Hatch
    Public Sub Run(cad As cAvax, handle As Integer)
        Dim data = cad.Get_Hatch(handle)
        If data Is Nothing Then Throw New ArgumentException("Native hatch required.")
        data.Options.Transparency = 25
        data.Options.DoublePattern = True
        If Not cad.Set_Hatch(handle, data) Then Throw New InvalidOperationException("Hatch update
    failed.")
    End Sub
End Module
```

Related options and reference

Set_HatchBoundaries

Purpose

Replace boundary loops from source objects and optionally establish association.

VB.NET declaration

```
Function Set_HatchBoundaries(ByVal handle As Integer, ByVal boundaryHandles As Integer(), Optional ByVal
    associative As Boolean = True) As Boolean
```

Parameters

Parameter	Meaning
handle	Handle of an existing native hatch in this control.
boundaryHandles	New boundary handles in this drawing; normal zero-based array.
associative	True (default) keeps links to the new sources. False creates independent boundary geometry.

Return value and behavior

True on success; False on invalid sources. Rebuilds boundaries and takes their elevation in one undoable edit.

Example

Call this procedure on the owning UI thread with a control already initialized by `StartAvax()`. Handles must belong to that control. File paths are supplied by the caller.

```
Option Strict On
Imports System
Imports Avax

Public Module Help_Set_HatchBoundaries
    Public Sub Run(cad As cAvax, handle As Integer, boundaries As Integer())
        If Not cad.Set_HatchBoundaries(handle, boundaries, associative:=True) Then Throw New
    InvalidOperationException("Boundary update failed.")
    End Sub
End Module
```

Related options and reference

SetAvaxProperty

cAvax method

Purpose

Sets the value of a specific AVAX property (AvaxProp)

See Drawing coordinate limits for Double coordinate properties, individual versus complete work-area assignment, and transactional settings validation.

See Idle selection and spatial queries for automatic selection, filled-shape picking, one-based Integer query results, availability and Double-coordinate areas.

See Arc commands and segmentation for the three arc pick orders, standalone completion, polyline continuation and arc resolution settings.

VB.NET declaration

```
Function SetAvaxProperty(ByRef iProp As AvaxProp, ByRef v As Object) As Boolean
```

Parameters

Name	Type and passing	Meaning
iProp	ByRef iProp As AvaxProp	The AvaxProp identifier to update. See AvaxProp for the available parameters.
v	ByRef v As Object	The new value in the type documented for that parameter. Use Double for measurements and Integer for former Integer values. When using Option Strict On, pass a typed variable as (value) or use an Object variable for this ByRef Object argument.

Use the declaration for the overload you call. ByRef arguments can receive outputs; declare variables and arrays with the exact indicated type. Optional defaults are shown in the declaration.

Return value

Boolean - True on success, false on error.

Remarks

Sets the parameter identified by an AvaxProp member using its documented value type. Direct properties and their AvaxProp equivalents share the same setting. Use Double values for measurements and Integer values for former Short parameters. For a custom selection cursor, assign SelectCursorMousePointer_p = 99 and SelectCursorFile_p to an existing .cur file; SelectCursorHideOn_p hides the CAD pickbox while keeping the custom Windows pointer available. A cursor file that cannot be loaded raises an exception and leaves the accepted cursor and filename intact. See AvaxProp for the available parameters.

Example

Setup

Add an Avax.cAvax Windows Forms control named `cAvax1`. Paste the procedures into the form class. Example setup and supporting files explains references, initialization and file paths.

Add these Windows Forms controls or components to the form. Fields referenced by Handles clauses must be declared WithEvents, as the Visual Studio designer normally does.

Name	Type
Command1	Button

VB.NET example

```
Private Sub Form1_Load(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs) Handles
    MyBase.Load
```

```
    Command1.Text = "SetAvaxProperty"
    cAvax1.StartAvax()
End Sub
Private Sub Form1_FormClosed(ByVal eventSender As System.Object, ByVal eventArgs As
    System.Windows.Forms.FormClosedEventArgs) Handles Me.FormClosed
    cAvax1.EndAvax()
End Sub
Private Sub Command1_Click(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs)
    Handles Command1.Click
    Dim vbCustom As Integer = 99 '(&H63)
    Call CAvax1.SetAvaxProperty(Avax.AvaxProp.SelectCursorHideOn, True)
    Call CAvax1.SetAvaxProperty(Avax.AvaxProp.SelectCursorMousePointer, vbCustom)
    Call CAvax1.SetAvaxProperty(Avax.AvaxProp.SelectCursorFile, My.Application.Info.DirectoryPath &
    "\..\box11.cur")
End Sub
```

SetBoundsAvax

cAvax method

Purpose

Sets the total XYZ drawing work area and fits the view to it, without changing object coordinates.

See Drawing coordinate limits for Double coordinate properties, individual versus complete work-area assignment, and transactional settings validation.

VB.NET declaration

```
Function SetBoundsAvax(ByRef xMin As Double, ByRef yMin As Double, ByRef zMin As Double, ByRef xMax As
    Double, ByRef yMax As Double, ByRef zMax As Double) As Boolean
```

Parameters

Name	Type and passing	Meaning
xMin	ByRef xMin As Double	Input xMin coordinate in Double world units. Must be finite; XY spans must be nonzero. Z may be planar.
yMin	ByRef yMin As Double	Input yMin coordinate in Double world units. Must be finite; XY spans must be nonzero. Z may be planar.
zMin	ByRef zMin As Double	Input zMin coordinate in Double world units. Must be finite; XY spans must be nonzero. Z may be planar.
xMax	ByRef xMax As Double	Input xMax coordinate in Double world units. Must be finite; XY spans must be nonzero. Z may be planar.
yMax	ByRef yMax As Double	Input yMax coordinate in Double world units. Must be finite; XY spans must be nonzero. Z may be planar.
zMax	ByRef zMax As Double	Input zMax coordinate in Double world units. Must be finite; XY spans must be nonzero. Z may be planar.

Use the declaration for the overload you call. ByRef arguments can receive outputs; declare variables and arrays with the exact indicated type. Optional defaults are shown in the declaration.

Return value

Boolean - True on success, false on error

Remarks

Supply finite XYZ coordinates with nonzero XY spans. Reversed corners are ordered, and the upper/right XY extent is expanded to the drawable canvas aspect ratio. The effective work area becomes the view. Arguments are retained; GetBoundsAvax and GetBoundsZoom return the effective bounds. The All_c command displays this area.

Native AVAX and legacy AVX retain work-area XYZ limits. Set OnLoadAutoLimitsOn=False to retain saved bounds on loading. A started, enabled control is required. Invalid input returns False without changing state. Cached drawing bounds and object geometry are retained.

Numeric changes from v2

Update the following variables, arrays or event-data receivers when porting the corresponding v2 call. The declarations above show the current overloads.

Parameter or value	v2 type	v3 type
xMin	Single	Double
yMin	Single	Double
zMin	Single	Double

Parameter or value	v2 type	v3 type
xMax	Single	Double
yMax	Single	Double
zMax	Single	Double

For ByRef parameters, use a variable with the exact v3 type. Preserve array dimensions and the documented indexing. Moving from v2 to v3 explains conversion and legacy file-storage rules.

Example

Setup

Add an Avax.cAvax Windows Forms control named `cAvax1`. Paste the procedures into the form class. Example setup and supporting files explains references, initialization and file paths.

Add these Windows Forms controls or components to the form. Fields referenced by Handles clauses must be declared WithEvents, as the Visual Studio designer normally does.

Name	Type
Command1	Button

VB.NET example

```
Private Sub Form1_Load(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs) Handles
    MyBase.Load
    cAvax1.StartAvax()
    Call cAvax1.Add_Line(-8, -8, 0, 45, 45, 0)
End Sub
Private Sub Form1_FormClosed(ByVal eventSender As System.Object, ByVal eventArgs As
    System.Windows.Forms.FormClosedEventArgs) Handles Me.FormClosed
    cAvax1.EndAvax()
End Sub
Private Sub Command1_Click(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs)
    Handles Command1.Click
    Dim xMin As Double
    Dim yMin As Double
    Dim zMin As Double
    Dim xMax As Double
    Dim yMax As Double
    Dim zMax As Double
    xMin = -20
    yMin = -20
    zMin = -20
    xMax = 80
    yMax = 80
    zMax = 80
    If cAvax1.SetBoundsAvax(xMin, yMin, zMin, xMax, yMax, zMax) = True Then
        CAvax1.Command = Avax.AvaxCommand.All_c
    End If
End Sub
```

SetBoundsDrawing

cAvax method

Purpose

Assigns cached XYZ drawing bounds and expands the work area if needed, without transforming objects.

VB.NET declaration

```
Function SetBoundsDrawing(ByRef xMin As Double, ByRef yMin As Double, ByRef zMin As Double, ByRef xMax As
    Double, ByRef yMax As Double, ByRef zMax As Double) As Boolean
```

Parameters

Name	Type and passing	Meaning
xMin	ByRef xMin As Double	Input xMin coordinate in Double world units. Must be finite; XY spans must be nonzero. Z may be planar.
yMin	ByRef yMin As Double	Input yMin coordinate in Double world units. Must be finite; XY spans must be nonzero. Z may be planar.
zMin	ByRef zMin As Double	Input zMin coordinate in Double world units. Must be finite; XY spans must be nonzero. Z may be planar.
xMax	ByRef xMax As Double	Input xMax coordinate in Double world units. Must be finite; XY spans must be nonzero. Z may be planar.
yMax	ByRef yMax As Double	Input yMax coordinate in Double world units. Must be finite; XY spans must be nonzero. Z may be planar.
zMax	ByRef zMax As Double	Input zMax coordinate in Double world units. Must be finite; XY spans must be nonzero. Z may be planar.

Use the declaration for the overload you call. ByRef arguments can receive outputs; declare variables and arrays with the exact indicated type. Optional defaults are shown in the declaration.

Return value

Boolean - True on success, false on error

Remarks

Coordinates must be finite with nonzero XY spans. Reversed corners are ordered internally; the supplied arguments are retained. The total XYZ work area expands when necessary. Expanded XY work bounds fit the canvas aspect ratio and become the view; a snapshot contained in the work area retains the current view.

Extends_c fits this cached rectangle. CalcBoundsDrawing replaces it with bounds calculated from geometry. Later entity changes do not automatically change the snapshot. Native AVAX and legacy AVX retain the snapshot; set OnLoadAutoLimitsOn=False to retain it when loading. The control must be started and enabled. Invalid input returns False without changing bounds. No entity edit or Undo operation is created.

Numeric changes from v2

Update the following variables, arrays or event-data receivers when porting the corresponding v2 call. The declarations above show the current overloads.

Parameter or value	v2 type	v3 type
xMin	Single	Double
yMin	Single	Double
zMin	Single	Double
xMax	Single	Double

Parameter or value	v2 type	v3 type
yMax	Single	Double
zMax	Single	Double

For ByRef parameters, use a variable with the exact v3 type. Preserve array dimensions and the documented indexing. Moving from v2 to v3 explains conversion and legacy file-storage rules.

Example

Setup

Add an Avax.cAvax Windows Forms control named `cAvax1`. Paste the procedures into the form class. Example setup and supporting files explains references, initialization and file paths.

Add these Windows Forms controls or components to the form. Fields referenced by Handles clauses must be declared WithEvents, as the Visual Studio designer normally does.

Name	Type
Command1	Button

VB.NET example

```
Private Sub Form1_Load(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs) Handles
    MyBase.Load
    Command1.Text = "SetBoundsDrawing"
    cAvax1.StartAvax()
    Call cAvax1.Add_Rectangle(0, 0, 10, 10)
End Sub
Private Sub Form1_FormClosed(ByVal eventSender As System.Object, ByVal eventArgs As
    System.Windows.Forms.FormClosedEventArgs) Handles Me.FormClosed
    cAvax1.EndAvax()
End Sub
Private Sub Command1_Click(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs)
    Handles Command1.Click
    Dim xMin As Double
    Dim yMin As Double
    Dim zMin As Double
    Dim xMax As Double
    Dim yMax As Double
    Dim zMax As Double
    xMin = -2
    yMin = -2
    zMin = -2
    xMax = 12
    yMax = 12
    zMax = 12
    If cAvax1.SetBoundsDrawing(xMin, yMin, zMin, xMax, yMax, zMax) = True Then
        CAvax1.Command = Avax.AvaxCommand.Extends_c
    End If
End Sub
```

SetBoundsStretch

cAvax method

Purpose

Define the parallelogram of the stretch window border before the use of Command=AvaxCommand.Stretch_c

VB.NET declaration

```
Function SetBoundsStretch(ByRef xMin As Double, ByRef yMin As Double, ByRef xMax As Double, ByRef yMax As
    Double) As Boolean
```

Parameters

Name	Type and passing	Meaning
xMin	ByRef xMin As Double	Finite Double coordinate of the stretch window; input preserved. Both XY spans must be nonzero.
yMin	ByRef yMin As Double	Finite Double coordinate of the stretch window; input preserved. Both XY spans must be nonzero.
xMax	ByRef xMax As Double	Finite Double coordinate of the stretch window; input preserved. Both XY spans must be nonzero.
yMax	ByRef yMax As Double	Finite Double coordinate of the stretch window; input preserved. Both XY spans must be nonzero.

Use the declaration for the overload you call. ByRef arguments can receive outputs; declare variables and arrays with the exact indicated type. Optional defaults are shown in the declaration.

Return value

Boolean - True on success, false on error.

Numeric changes from v2

Update the following variables, arrays or event-data receivers when porting the corresponding v2 call. The declarations above show the current overloads.

Parameter or value	v2 type	v3 type
xMin	Single	Double
yMin	Single	Double
xMax	Single	Double
yMax	Single	Double

For ByRef parameters, use a variable with the exact v3 type. Preserve array dimensions and the documented indexing. Moving from v2 to v3 explains conversion and legacy file-storage rules.

Example

Setup

Add an Avax.cAvax Windows Forms control named `cAvax1`. Paste the procedures into the form class. Example setup and supporting files explains references, initialization and file paths.

Add these Windows Forms controls or components to the form. Fields referenced by Handles clauses must be declared WithEvents, as the Visual Studio designer normally does.

Name	Type
Command1	Button

VB.NET example

```
Dim entityHandles(1) As Integer
Private Sub Form1_Load(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs) Handles
    MyBase.Load
    Command1.Text = "Move the bottom left point of the line"
    cAvax1.StartAvax()
    entityHandles(1) = CAvax1.Add_Line(1, 1, 0, 12, 12, 0)
End Sub
Private Sub Form1_FormClosed(ByVal eventSender As System.Object, ByVal eventArgs As
    System.Windows.Forms.FormClosedEventArgs) Handles Me.FormClosed
    cAvax1.EndAvax()
End Sub
Private Sub Command1_Click(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs)
    Handles Command1.Click
    Dim xMin As Double
    Dim yMin As Double
    Dim xMax As Double
    Dim yMax As Double
    Dim Answer As Boolean
    xMin = 0
    yMin = 0
    xMax = 2
    yMax = 2
    Call CAvax1.SelectItems(entityHandles)
    Answer = cAvax1.SetBoundsStretch(xMin, yMin, xMax, yMax)
    CAvax1.Command = Avax.AvaxCommand.Stretch_c
    Command1.Enabled = False
End Sub
```

SetBoundsZoom

cAvax method

Purpose

Changes the view to show a supplied XY rectangle, adjusted to the viewport aspect ratio, without transforming objects.

VB.NET declaration

```
Function SetBoundsZoom(ByRef zxMin As Double, ByRef zyMin As Double, ByRef zxMax As Double, ByRef zyMax As
     Double) As Boolean
```

Parameters

Name	Type and passing	Meaning
zxMin	ByRef zxMin As Double	Requested XY view coordinate (Double, ByRef); receives the aspect-normalized and work-area-constrained effective value on success only.
zyMin	ByRef zyMin As Double	Requested XY view coordinate (Double, ByRef); receives the aspect-normalized and work-area-constrained effective value on success only.
zxMax	ByRef zxMax As Double	Requested XY view coordinate (Double, ByRef); receives the aspect-normalized and work-area-constrained effective value on success only.
zyMax	ByRef zyMax As Double	Requested XY view coordinate (Double, ByRef); receives the aspect-normalized and work-area-constrained effective value on success only.

Use the declaration for the overload you call. ByRef arguments can receive outputs; declare variables and arrays with the exact indicated type. Optional defaults are shown in the declaration.

Return value

Boolean - True on success, false on error.

Remarks

Reversed corners are ordered. The rectangle fits the canvas aspect ratio from its lower-left corner and navigation is constrained to the total work area. All four ByRef arguments receive the effective view after success. The operation records one zoom-history entry and raises ZoomChange. A started, enabled control is required. Invalid, nonfinite or zero-span input returns False without changing view, outputs or history.

Numeric changes from v2

Update the following variables, arrays or event-data receivers when porting the corresponding v2 call. The declarations above show the current overloads.

Parameter or value	v2 type	v3 type
zxMin	Single	Double
zyMin	Single	Double
zxMax	Single	Double
zyMax	Single	Double

For ByRef parameters, use a variable with the exact v3 type. Preserve array dimensions and the documented indexing. Moving from v2 to v3 explains conversion and legacy file-storage rules.

Example

Setup

Add an Avax.cAvax Windows Forms control named `cAvax1`. Paste the procedures into the form class. Example setup and supporting files explains references, initialization and file paths.

Add these Windows Forms controls or components to the form. Fields referenced by Handles clauses must be declared WithEvents, as the Visual Studio designer normally does.

Name	Type
Command1	Button

VB.NET example

```
Private Sub Form1_Load(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs) Handles
    MyBase.Load
    Command1.Text = "SetBoundsZoom"
    cAvax1.StartAvax()
    Call cAvax1.Add_Rectangle(0, 0, 12, 10)
End Sub
Private Sub Form1_FormClosed(ByVal eventSender As System.Object, ByVal eventArgs As
    System.Windows.Forms.FormClosedEventArgs) Handles Me.FormClosed
    cAvax1.EndAvax()
End Sub
Private Sub Command1_Click(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs)
    Handles Command1.Click
    Dim zxMin As Double
    Dim zyMin As Double
    Dim zxMax As Double
    Dim zyMax As Double
    Dim Answer As Boolean
    If cAvax1.GetBoundsZoom(zxMin, zyMin, zxMax, zyMax) = True Then
        zxMin = zxMin - 15
        zyMin = zyMin - 15
        zxMax = zxMax + 15
        zyMax = zyMax + 15
        Answer = cAvax1.SetBoundsZoom(zxMin, zyMin, zxMax, zyMax)
        CAvax1.Command = Avax.AvaxCommand.Redraw_c
    End If
End Sub
```

SetCoordinate

cAvax method

Purpose

Supplies the next XYZ construction point to the active interactive command in drawing coordinates.

See Line, Point and Rectangle commands for connected XYZ lines, the planar Point tool versus Add_Point, rectangular closed polylines, completion and host callbacks.

See Circle and Ellipse commands for center/radius, diameter and three-point circles, ellipse radii, Double precision and one-shot command completion.

See Arc commands and segmentation for the three arc pick orders, standalone completion, polyline continuation and arc resolution settings.

See Polyline, Polygon and Close commands for pending XYZ chains, exact closing edges, regular polygons, vertex counts, completion and complete VB.NET examples.

VB.NET declaration

```
Function SetCoordinate(ByRef x As Double, ByRef y As Double, ByRef z As Double) As Boolean
```

Parameters

Name	Type and passing	Meaning
x	ByRef x As Double	Is the x coordinate of the starting point
y	ByRef y As Double	Is the y coordinate of the starting point
z	ByRef z As Double	Is the z coordinate of the starting point

Use the declaration for the overload you call. ByRef arguments can receive outputs; declare variables and arrays with the exact indicated type. Optional defaults are shown in the declaration.

Return value

Boolean - True on success, false on error.

Remarks

Use successive calls to supply all points required by the selected command. A True result means the point was accepted, even when the command still needs more points. The ByRef Double input variables are retained. A final accepted point can create geometry and raise normal command notifications.

Stopped, disabled, paused, edit-locked or idle controls return False. Nonfinite coordinates, unresolved required snaps or invalid final geometry return False; earlier accepted points remain available for correction. Application event-handler exceptions propagate after committed creation. If a handler starts another command or replaces the drawing, the returning older call retains that newer input state.

Numeric changes from v2

Update the following variables, arrays or event-data receivers when porting the corresponding v2 call. The declarations above show the current overloads.

Parameter or value	v2 type	v3 type
x	Single	Double
y	Single	Double
z	Single	Double

For ByRef parameters, use a variable with the exact v3 type. Preserve array dimensions and the documented indexing. Moving from v2 to v3 explains conversion and legacy file-storage rules.

Example

Setup

Add an Avax.cAvax Windows Forms control named `cAvax1`. Paste the procedures into the form class. Example setup and supporting files explains references, initialization and file paths.

Add these Windows Forms controls or components to the form. Fields referenced by Handles clauses must be declared WithEvents, as the Visual Studio designer normally does.

Name	Type
Button1	Button

VB.NET example

```
Private Sub Form1_Load(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs) Handles
    MyBase.Load
    Call cAvax1.StartAvax()
End Sub
Private Sub Form1_FormClosed(ByVal eventSender As System.Object, ByVal eventArgs As
    System.Windows.Forms.FormClosedEventArgs) Handles Me.FormClosed
    Call cAvax1.EndAvax()
End Sub
Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
    Button1.Click
    Dim x As Double
    Dim y As Double
    Dim z As Double
    Dim Answer As Boolean
    CAvax1.Command = Avax.AvaxCommand.Line_c
    x = 10
    y = 10
    z = 10
    Answer = CAvax1.SetCoordinate(x, y, z)
End Sub
```

SetEditablePointSelectionPreview

Purpose

Chooses the temporary outline shown during an active SelectPoints request. The application remains responsible for creating an object after receiving the selected points.

VB.NET declaration

```
Function SetEditablePointSelectionPreview(ByVal kind As String, Optional ByVal vertices As Integer = 6) As
    Boolean
```

Parameters

Parameter	Meaning
kind	Case-sensitive preview name: Rectangle, Circle, Ellipse, Polygon, Arc, Closed arc, Pie or Line.
vertices	Polygon side count, from 3 to 10000; ignored for the other preview kinds.

Return value and behavior

Returns True when accepted. Requires a started, enabled, unpaused control and an active SelectPoints request: two points for ordinary shapes, or three start/end/centre points for Arc or Closed arc. Closed arc adds an implicit closing edge to the same arc outline. A rejected kind, point count or polygon count preserves the existing preview. Before enough points are available, no outline is shown; the first arc stage uses a line. Completing or cancelling the request clears the preview. Choosing a preview does not add a drawing object or alter the accepted coordinates.

VB.NET example

Pass a started control on its owning UI thread. This is a complete standalone module.

```
Option Strict On
Imports Avax

Public Module EditableShapePreviewExample
    Public Function BeginRectangle(cad As cAvax) As Boolean
        If Not cad.SelectPoints("rectangle", 2, Nothing, Nothing, Nothing) Then Return False
        Return cad.SetEditablePointSelectionPreview("Rectangle")
    End Function
End Module
```

Related options and reference

SetFontName

cAvax method

Purpose

Sets the font name and the character set for the specific font table index

VB.NET declarations

```
Sub SetFontName(ByVal Index As Integer, ByVal sFontName As String, ByVal lFontSet As Integer)

Sub SetFontName(ByVal Index As Integer, ByVal sFontName As String)
```

Parameters

Name	Type and passing	Meaning
Index	ByVal Index As Integer	Existing Integer table index, 1..GetFonts(). Does not expand the table.
sFontName	ByVal sFontName As String	Font family name; blank or the exact current FontName uses inheritance. Names cannot contain CR, LF or NUL.
lFontSet	ByVal lFontSet As Integer	Integer charset -1 (inherit) or 0..255. The overload without this parameter uses -1. The current FontCharset is normalized to -1.

Use the declaration for the overload you call. ByRef arguments can receive outputs; declare variables and arrays with the exact indicated type. Optional defaults are shown in the declaration.

Result

This procedure has no return value. Any outputs are supplied through the documented ByRef arguments.

Numeric changes from v2

Update the following variables, arrays or event-data receivers when porting the corresponding v2 call. The declarations above show the current overloads.

Parameter or value	v2 type	v3 type
Index	Short	Integer

For ByRef parameters, use a variable with the exact v3 type. Preserve array dimensions and the documented indexing. Moving from v2 to v3 explains conversion and legacy file-storage rules.

Example

Setup

Add an Avax.cAvax Windows Forms control named `cAvax1`. Paste the procedures into the form class. Example setup and supporting files explains references, initialization and file paths.

Add these Windows Forms controls or components to the form. Fields referenced by Handles clauses must be declared WithEvents, as the Visual Studio designer normally does.

Name	Type
Command1	Button

VB.NET example

```
Private Sub Form1_Load(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs) Handles
    MyBase.Load
    Command1.Text = "SetFontName"
    cAvax1.StartAvax()
End Sub
```

```
Private Sub Form1_FormClosed(ByVal eventSender As System.Object, ByVal eventArgs As
    System.Windows.Forms.FormClosedEventArgs) Handles Me.FormClosed
    cAvax1.EndAvax()
End Sub
Private Sub Command1_Click(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs)
    Handles Command1.Click
    Dim Index As Integer
    Dim sFontName As String = String.Empty
    Dim lFontSet As Integer
    Index = 1
    sFontName = "Arial"
    lFontSet = 0
    Call cAvax1.SetFontName(Index, sFontName, lFontSet)
    Call cAvax1.GetFontName(Index, sFontName, lFontSet)
    MsgBox(sFontName & Str(lFontSet), MsgBoxStyle.Information)
End Sub
```

SetFonts

cAvax method

Purpose

Defines all the font names and their associated font character sets

VB.NET declaration

```
Function SetFonts(ByRef sFonts As String(), ByRef lFontsSet As Integer()) As Boolean
```

Parameters

Name	Type and passing	Meaning
sFonts	ByRef sFonts As String()	ByRef String() input containing reserved slot zero and 1..256 usable names. The input reference is unchanged; the control owns a copy. Blank names inherit FontName.
lFontsSet	ByRef lFontsSet As Integer()	ByRef Integer() input of exactly the same length as sFonts. Entries 1..count must be -1 or 0..255; the control owns a copy.

Use the declaration for the overload you call. ByRef arguments can receive outputs; declare variables and arrays with the exact indicated type. Optional defaults are shown in the declaration.

Return value

Boolean - True on success, false on error

Example

Setup

Add an Avax.cAvax Windows Forms control named `cAvax1`. Paste the procedures into the form class. Example setup and supporting files explains references, initialization and file paths.

Add these Windows Forms controls or components to the form. Fields referenced by Handles clauses must be declared WithEvents, as the Visual Studio designer normally does.

Name	Type
Command1	Button

VB.NET example

```
Private Sub Form1_Load(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs) Handles
    MyBase.Load
    Command1.Text = "SetFonts"
    cAvax1.StartAvax()
End Sub
Private Sub Form1_FormClosed(ByVal eventSender As System.Object, ByVal eventArgs As
    System.Windows.Forms.FormClosedEventArgs) Handles Me.FormClosed
    cAvax1.EndAvax()
End Sub
Private Sub Command1_Click(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs)
    Handles Command1.Click
    Dim sFonts() As String = Nothing
    Dim lFontsSet() As Integer = Nothing
    Dim iCnt, lMax As Integer
    lMax = System.Drawing.FontFamily.Families.Length
    If lMax > 256 Then lMax = 256
    ReDim sFonts(lMax)
    ReDim lFontsSet(lMax)
    For iCnt = 0 To lMax - 1
```

```
        sFonts(iCnt + 1) = System.Drawing.FontFamily.Families(iCnt).Name
        lFontsSet(iCnt + 1) = 0
    Next iCnt
    MsgBox(cAvax1.SetFonts(sFonts, lFontsSet), MsgBoxStyle.Information, "SetFonts")
End Sub
```

SetHandleKey

cAvax method

Purpose

Sets the key name of the object with the specific handle

VB.NET declaration

```
Function SetHandleKey(ByRef lHandle As Integer, ByRef sKey As String) As Boolean
```

Parameters

Name	Type and passing	Meaning
lHandle	ByRef lHandle As Integer	Existing drawing handle; unchanged by this method.
sKey	ByRef sKey As String	Exact key text to store; Nothing clears it. Spaces/case are preserved.

Use the declaration for the overload you call. ByRef arguments can receive outputs; declare variables and arrays with the exact indicated type. Optional defaults are shown in the declaration.

Return value

Boolean - True on success, false on error.

Example

Setup

Add an Avax.cAvax Windows Forms control named `cAvax1`. Paste the procedures into the form class. Example setup and supporting files explains references, initialization and file paths.

Add these Windows Forms controls or components to the form. Fields referenced by Handles clauses must be declared WithEvents, as the Visual Studio designer normally does.

Name	Type
Command1	Button

VB.NET example

```
Dim lHandle As Integer
Private Sub Form1_Load(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs) Handles
    MyBase.Load
    Command1.Text = "GetHandleKey"
    cAvax1.StartAvax()
    lHandle = cAvax1.Add_Line(0, 0, 0, 10, 10, 0)
    Call cAvax1.SetHandleKey(lHandle, "Line #1")
End Sub
Private Sub Form1_FormClosed(ByVal eventSender As System.Object, ByVal eventArgs As
    System.Windows.Forms.FormClosedEventArgs) Handles Me.FormClosed
    cAvax1.EndAvax()
End Sub
Private Sub Command1_Click(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs)
    Handles Command1.Click
    MsgBox(cAvax1.GetHandleKey(lHandle), MsgBoxStyle.Information, "GetHandleKey")
End Sub
```

SetImageEffects

Purpose

Replaces the ordered effect list of one image or several images. The same list is applied to every target in the batch overload.

Declarations

```
Function SetImageEffects(ByVal handle As Integer,
                         ByVal effects As AvaxImageEffect()) As Boolean

Function SetImageEffects(ByVal ids As Integer(),
                         ByVal effects As AvaxImageEffect()) As Boolean
```

Parameters

Name	Meaning
handle	Integer handle of one image to edit.
ids	A handle array with reserved element 0 and target handles in elements 1 through Length - 1. Include at least one target. Duplicate handles are processed once.
effects	An ordinary zero-based array of 0 through 32 non-Nothing AvaxImageEffect values. Array order is processing order. An empty array removes the effects; Nothing is invalid.

Result and editing rules

Returns True when the edit is accepted. The control must be started, enabled, undisposed and unlocked. All targets must be existing supported still-bitmap shapes on editable visible layers. An unknown handle, unsupported target or hidden/frozen target rejects the batch.

Each Kind must be defined and each finite Double Amount must fall within its inclusive supported range. Pixelate and Posterize require whole-number amounts. Disabled entries must also contain valid kinds and amounts. Multiple entries can use the same kind. Entries with Enabled = False are retained but skipped during rendering.

When at least one effect is enabled, every target source must decode successfully and contain at most 32,000,000 pixels. Input and target preparation completes before target images are replaced. Changed targets participate together in normal Undo/Redo. The control stores its own effect values, so later changes to the caller's array or objects do not change the drawing.

Effects preserve the stored source bytes and drawing geometry. Crop resamples a centred inner image area into the same pixel canvas; it keeps the image object's position and outline. ResetImageEffects restores its unfiltered appearance.

Invalid effect values, source decoding failures, rejected target preparation and unavailable controls return False. Exceptions raised by host event handlers during a drawing edit can propagate to the caller.

VB.NET example: one image

```
Option Strict On
Public Module SetImageEffectsExample
    Public Function AdjustImage(cad As Avax.cAvax, handle As Integer) As Boolean
        Dim effects As Avax.AvaxImageEffect() = {
            New Avax.AvaxImageEffect(Avax.AvaxImageEffectKind.Brightness, 12.0),
            New Avax.AvaxImageEffect(Avax.AvaxImageEffectKind.Contrast, 20.0),
            New Avax.AvaxImageEffect(Avax.AvaxImageEffectKind.Gamma, 1.2),
            New Avax.AvaxImageEffect(Avax.AvaxImageEffectKind.Crop, 5.5)
        }
        Return cad.SetImageEffects(handle, effects)
    End Function
End Module
```

VB.NET example: a batch

```
Option Strict On
Public Module SetImageEffectsBatchExample
    Public Function AdjustTwoImages(cad As Avax.cAvax, firstHandle As Integer,
                                    secondHandle As Integer) As Boolean
        Dim ids As Integer() = {0, firstHandle, secondHandle}
```

```
        Dim effects As Avax.AvaxImageEffect() = {
            New Avax.AvaxImageEffect(Avax.AvaxImageEffectKind.Exposure, 0.5),
            New Avax.AvaxImageEffect(Avax.AvaxImageEffectKind.Saturation, 15.0)
        }
        Return cad.SetImageEffects(ids, effects)
    End Function
End Module
```

Related options and reference

SetItemByLayer

Purpose

Set independent layer-color and layer-line-style inheritance for one existing object.

VB.NET declaration

```
Public Function SetItemByLayer(ByVal handle As Integer, ByVal colorByLayer As Boolean, ByVal
    lineStyleByLayer As Boolean) As Boolean
```

Parameters

handle: a drawing object handle. colorByLayer and lineStyleByLayer: True follows the layer; False retains the current resolved appearance.

Return value

True on success, otherwise False. A successful edit can be undone.

Example

Call after StartAvax on the owning UI thread.

```
Option Strict On
Public Module Help_SetItemByLayer
    Public Sub Run(cad As Avax.cAvax)
        Dim handle As Integer = cad.Add_Line(0.0, 0.0, 0.0, 10.0, 0.0, 0.0)
        Dim changed As Boolean = cad.SetItemByLayer(handle, True, True)
    End Sub
End Module
```

Related options and reference

SetItemsByLayer

Purpose

Apply the same inheritance choices atomically to several drawing objects in one Undo step.

VB.NET declaration

```
Public Function SetItemsByLayer(ByVal ids As Integer(), ByVal colorByLayer As Boolean, ByVal
    lineStyleByLayer As Boolean) As Boolean
```

Parameters

ids: positive object handles; a legacy leading zero is accepted, zeros are ignored and duplicates are removed. Negative or unknown handles are invalid. colorByLayer and lineStyleByLayer select the independent modes.

Return value

True if the entire batch succeeds; False if the control, handles or prepared geometry are invalid. No partial edit is committed.

Example

Call after StartAvax on the owning UI thread.

```
Option Strict On
Public Module Help_SetItemsByLayer
    Public Sub Run(cad As Avax.cAvax)
        Dim first As Integer = cad.AddLine(0.0, 0.0, 0.0, 10.0, 0.0, 0.0)
        Dim second As Integer = cad.AddLine(0.0, 5.0, 0.0, 10.0, 5.0, 0.0)
        Dim changed As Boolean = cad.SetItemsByLayer(New Integer() {0, first, second}, True, True)
    End Sub
End Module
```

Related options and reference

SetLayerColor

Purpose

Change a layer color and refresh the objects that inherit it, including block contents.

VB.NET declaration

```
Public Function SetLayerColor(ByVal layer As Integer, ByVal color As Integer) As Boolean
```

Parameters

layer: an existing zero-based layer index. color: explicit OLE RGB from 0 to &HFFFFFF; red occupies the low byte, as with GetPalColor.

Return value

True on success; False on an invalid layer/color or unavailable control. The layer and affected objects form one undoable edit.

Example

Call after StartAvax on the owning UI thread.

```
Option Strict On
Public Module Help_SetLayerColor
    Public Sub Run(cad As Avax.cAvax)
        Dim changed As Boolean = cad.SetLayerColor(0, &HFF) ' Red in OLE RGB
    End Sub
End Module
```

Related options and reference

SetLayerLineStyle

Purpose

Change a layer line style and refresh the objects that inherit it.

VB.NET declaration

```
Public Function SetLayerLineStyle(ByVal layer As Integer, ByVal lineStyle As Integer) As Boolean
```

Parameters

layer: an existing zero-based layer index. lineStyle: a valid index in the drawing line-style table; 0 means Continuous.

Return value

True on success; False on an invalid index or unavailable control. The layer and affected objects form one undoable edit.

Example

Call after StartAvax on the owning UI thread.

```
Option Strict On
Public Module Help_SetLayerLineStyle
    Public Sub Run(cad As Avax.cAvax)
        Dim changed As Boolean = cad.SetLayerLineStyle(0, 0) ' Continuous
    End Sub
End Module
```

Related options and reference

SetLayers

cAvax method

Purpose

Sets the names and status for all the drawing layers

VB.NET declaration

```
Function SetLayers(ByRef sLayerNames As String(), ByRef iLayerStatus As Integer()) As Boolean
```

Parameters

Name	Type and passing	Meaning
sLayerNames	ByRef sLayerNames As String()	The names of the layers
iLayerStatus	ByRef iLayerStatus As Integer()	Integer() indexed from zero. Independent flags: 1 hidden, 2 locked, 4 nonprinting; combine with Or (0..7). Original table indices are 0..255.

Use the declaration for the overload you call. ByRef arguments can receive outputs; declare variables and arrays with the exact indicated type. Optional defaults are shown in the declaration.

Return value

Boolean - True on success, false on error.

Remarks

Every drawing has 256 layers (from 0 to 255)

Numeric changes from v2

Update the following variables, arrays or event-data receivers when porting the corresponding v2 call. The declarations above show the current overloads.

Parameter or value	v2 type	v3 type
iLayerStatus	Short()	Integer()

For ByRef parameters, use a variable with the exact v3 type. Preserve array dimensions and the documented indexing. Moving from v2 to v3 explains conversion and legacy file-storage rules.

Example

Setup

Add an Avax.cAvax Windows Forms control named `cAvax1`. Paste the procedures into the form class. Example setup and supporting files explains references, initialization and file paths.

Add these Windows Forms controls or components to the form. Fields referenced by Handles clauses must be declared WithEvents, as the Visual Studio designer normally does.

Name	Type
Command1	Button

VB.NET example

```
Private Sub Form1_Load(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs) Handles
    MyBase.Load
    cAvax1.StartAvax()
    Call cAvax1.Add_Line(0, 10, 0, 10, 10, 0)
    Call CAvax1.Add_Line(0, 0, 0, 10, 10, 0, Nothing, -1, -1, -1, 1)
```

```
End Sub
Private Sub Form1_FormClosed(ByVal eventSender As System.Object, ByVal eventArgs As
    System.Windows.Forms.FormClosedEventArgs) Handles Me.FormClosed
    cAvax1.EndAvax()
End Sub
Private Sub Command1_Click(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs)
    Handles Command1.Click
    Dim sLayerNames() As String = Nothing
    Dim iLayerStatus() As Integer = Nothing
    Dim Answer As Boolean
    Call cAvax1.GetLayers(sLayerNames, iLayerStatus)
    sLayerNames(1) = "Layer 1"
    iLayerStatus(1) = 1
    Answer = cAvax1.SetLayers(sLayerNames, iLayerStatus)
    Call cAvax1.GetLayers(sLayerNames, iLayerStatus)
    MsgBox(sLayerNames(1) & " is not visible")
    iLayerStatus(1) = 0
    Answer = cAvax1.SetLayers(sLayerNames, iLayerStatus)
    MsgBox(sLayerNames(1) & " is visible")
End Sub
```

SetLineStyle

cAvax method

Purpose

This function sets the line type name and data for the specific line type index

VB.NET declaration

```
Function SetLineStyle(ByVal Index As Integer, ByVal sLineStyle As String) As Boolean
```

Parameters

Name	Type and passing	Meaning
Index	ByVal Index As Integer	Existing Integer table index in 1..GetLineStyles(). No automatic growth.
sLineStyle	ByVal sLineStyle As String	Named LIN definition: positive lengths draw, negative lengths skip, zero denotes a dot. Supports quoted text and embedded shape records; unresolved external SHX references must be imported first.

Use the declaration for the overload you call. ByRef arguments can receive outputs; declare variables and arrays with the exact indicated type. Optional defaults are shown in the declaration.

Return value

Boolean - True on successfull input, false on fail

Numeric changes from v2

Update the following variables, arrays or event-data receivers when porting the corresponding v2 call. The declarations above show the current overloads.

Parameter or value	v2 type	v3 type
Index	Short	Integer

For ByRef parameters, use a variable with the exact v3 type. Preserve array dimensions and the documented indexing. Moving from v2 to v3 explains conversion and legacy file-storage rules.

Example

Setup

Add an Avax.cAvax Windows Forms control named `cAvax1`. Paste the procedures into the form class. Example setup and supporting files explains references, initialization and file paths.

Add these Windows Forms controls or components to the form. Fields referenced by Handles clauses must be declared WithEvents, as the Visual Studio designer normally does.

Name	Type
Command1	Button

VB.NET example

```
Private Sub Form1_Load(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs) Handles
    MyBase.Load
    Command1.Text = "SetLineStyle"
    cAvax1.StartAvax()
End Sub
Private Sub Form1_FormClosed(ByVal eventSender As System.Object, ByVal eventArgs As
    System.Windows.Forms.FormClosedEventArgs) Handles Me.FormClosed
    cAvax1.EndAvax()
```

```
End Sub
Private Sub Command1_Click(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs)
    Handles Command1.Click
    Dim Index As Integer
    Dim sLineStyle As String = String.Empty
    sLineStyle = "MyStyle, 1, -0.5, 0.5, -0.25, 0.25, -0.125"
    Index = 7
    If cAvax1.SetLineStyle(Index, sLineStyle) = True Then
        Call CAvax1.Add_Line(0, 0, 0, 25, 12, 0, Nothing, -1, -1, 7, -1)
    End If
    Command1.Enabled = False
End Sub
```

SetLineStyles

cAvax method

Purpose

This function assigns all the line type names and data

VB.NET declaration

```
Function SetLineStyles(ByRef sLineStyles As String()) As Boolean
```

Parameters

Name	Type and passing	Meaning
sLineStyles	ByRef sLineStyles As String()	ByRef input String() with reserved slot zero and 1..255 entries. The reference is preserved and contents are copied. Bulk blank entries retain their positions; invalid input leaves the current table intact.

Use the declaration for the overload you call. ByRef arguments can receive outputs; declare variables and arrays with the exact indicated type. Optional defaults are shown in the declaration.

Return value

Boolean - True on success, false on error

Example

Setup

Add an Avax.cAvax Windows Forms control named `cAvax1`. Paste the procedures into the form class. Example setup and supporting files explains references, initialization and file paths.

Add these Windows Forms controls or components to the form. Fields referenced by Handles clauses must be declared WithEvents, as the Visual Studio designer normally does.

Name	Type
Command1	Button

VB.NET example

```
Private Sub Form1_Load(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs) Handles
    MyBase.Load
    Command1.Text = "GetLineStyles"
    cAvax1.StartAvax()
End Sub
Private Sub Form1_FormClosed(ByVal eventSender As System.Object, ByVal eventArgs As
    System.Windows.Forms.FormClosedEventArgs) Handles Me.FormClosed
    cAvax1.EndAvax()
End Sub
Private Sub Command1_Click(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs)
    Handles Command1.Click
    Dim sLineStyles() As String = Nothing
    Dim lMax As Integer
    Dim sMsg As String = String.Empty
    Dim lCnt As Integer
    lMax = cAvax1.GetLineStyles(sLineStyles)
    sLineStyles(1) = "MyStyle1, 1, -0.5, 0.5, -0.25, 0.25, -0.125"
    sLineStyles(2) = "MyStyle2, 1, -1.5, 1.5, -1.25, 1.25, -1.125"
    Call cAvax1.SetLineStyles(sLineStyles)
    lMax = cAvax1.GetLineStyles(sLineStyles)
    For lCnt = 1 To lMax
        sMsg = sMsg & sLineStyles(lCnt) & vbCrLf
```

```
    Next lCnt
    MsgBox(sMsg, MsgBoxStyle.Information, "Current LineStyles")
End Sub
```

SetPalColor

cAvax method

Purpose

Assigns an RGB color to an index in the current palette.

VB.NET declaration

```
Function SetPalColor(ByVal index0 As Integer, ByVal clr As Integer) As Boolean
```

Parameters

Name	Type and passing	Meaning
index0	ByVal index0 As Integer	Existing zero-based Integer index; invalid indices return False without changing the palette.
clr	ByVal clr As Integer	Raw OLE/BGR Integer color: red is bits 0..7, green 8..15, blue 16..23.

Use the declaration for the overload you call. ByRef arguments can receive outputs; declare variables and arrays with the exact indicated type. Optional defaults are shown in the declaration.

Return value

Boolean - True on success, false on error.

Numeric changes from v2

Update the following variables, arrays or event-data receivers when porting the corresponding v2 call. The declarations above show the current overloads.

Parameter or value	v2 type	v3 type
index0	Short	Integer

For ByRef parameters, use a variable with the exact v3 type. Preserve array dimensions and the documented indexing. Moving from v2 to v3 explains conversion and legacy file-storage rules.

Example

Setup

Add an Avax.cAvax Windows Forms control named `cAvax1`. Paste the procedures into the form class. Example setup and supporting files explains references, initialization and file paths.

Add these Windows Forms controls or components to the form. Fields referenced by Handles clauses must be declared WithEvents, as the Visual Studio designer normally does.

Name	Type
Command1	Button

VB.NET example

```
Dim iClr As Integer
Private Sub Form1_Load(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs) Handles
    MyBase.Load
    Command1.Text = "SetPalColor"
    cAvax1.StartAvax()
    iClr = 0
    Call CAvax1.Add_Line(0, 0, 0, 10, 10, 0, Nothing, iClr, 2, -1, -1)
End Sub
Private Sub Form1_FormClosed(ByVal eventSender As System.Object, ByVal eventArgs As
```

```
    System.Windows.Forms.FormClosedEventArgs) Handles Me.FormClosed
    cAvax1.EndAvax()
End Sub
Private Sub Command1_Click(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs)
    Handles Command1.Click
    Dim RGBColor As Integer
    RGBColor = System.Drawing.ColorTranslator.ToOle(System.Drawing.Color.Magenta)
    If cAvax1.SetPalColor(iClr, RGBColor) = True Then
        'The color of the line was changed
    End If
End Sub
```

SetPalette

cAvax method

Purpose

Assigns all the RGB colors of the current color palette

VB.NET declaration

```
Function SetPalette(ByRef ClrPalette As Integer()) As Boolean
```

Parameters

Name	Type and passing	Meaning
ClrPalette	ByRef ClrPalette As Integer()	Input: zero-based Integer() with 16..256 colors. The control copies it and preserves the caller reference. Invalid input does not replace the current table.

Use the declaration for the overload you call. ByRef arguments can receive outputs; declare variables and arrays with the exact indicated type. Optional defaults are shown in the declaration.

Return value

Boolean - True on success, false on error.

Example

Setup

Add an Avax.cAvax Windows Forms control named `cAvax1`. Paste the procedures into the form class. Example setup and supporting files explains references, initialization and file paths.

Add these Windows Forms controls or components to the form. Fields referenced by Handles clauses must be declared WithEvents, as the Visual Studio designer normally does.

Name	Type
Command1	Button

VB.NET example

```
Private Sub Form1_Load(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs) Handles
    MyBase.Load
    Command1.Text = "SetPalette"
    cAvax1.StartAvax()
    Dim y As Double
    For y = 0 To 15
        Call CAvax1.Add_Line(0, y, 0, 15, y, 0, Nothing, CInt(y), 3, -1, -1)
    Next y
End Sub
Private Sub Form1_FormClosed(ByVal eventSender As System.Object, ByVal eventArgs As
    System.Windows.Forms.FormClosedEventArgs) Handles Me.FormClosed
    cAvax1.EndAvax()
End Sub
Private Sub Command1_Click(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs)
    Handles Command1.Click
    Dim ClrPalette() As Integer = Nothing
    Dim Answer As Boolean
    Dim iCnt As Integer
    ReDim ClrPalette(15)
    For iCnt = 0 To 15
        ClrPalette(iCnt) = QBColor(iCnt)
    Next iCnt
```

```
    Answer = cAvax1.SetPalette(ClrPalette)
End Sub
```

SetPathNodeMode

Purpose

Sets the tangent policy used when a paired Bezier control handle is moved at a path node.

Declaration

```
Function SetPathNodeMode(ByVal handle As Integer, ByVal figureIndex As Integer, ByVal nodeIndex As
    Integer, ByVal mode As AvaxPathNodeMode) As Boolean
```

Parameters and result

Name	Meaning
handle	Object with a compound path.
figureIndex	Zero-based index in Figures.
nodeIndex	0 identifies the figure's start. A positive index identifies the endpoint of Segments(nodeIndex - 1).
mode	One of the AvaxPathNodeMode values below.

Value	Meaning
Corner = 0	Control handles move independently.
Smooth = 1	Paired handles stay collinear; the opposite handle retains its length.
Symmetric = 2	Paired handles stay collinear and have equal lengths.

Returns False for invalid indices, an unsupported mode or an object that cannot be edited. The mode applies to paired Bezier handles; it does not change a line or arc into a Bezier segment. Like ChangeCompoundPath, this edit removes a parametric recipe.

VB.NET example

```
Option Strict On
Public Module SetPathNodeModeExample
    Public Function AddSmoothJoin(cad As Avax.cAvax) As Integer
        Dim path As New Avax.AvaxPathData
        Dim figure As New Avax.AvaxPathFigure
        figure.Segments.Add(Avax.AvaxPathSegment.BezierTo(
            10.0, 20.0, 20.0, 20.0, 30.0, 0.0))
        figure.Segments.Add(Avax.AvaxPathSegment.BezierTo(
            40.0, -20.0, 50.0, -20.0, 60.0, 0.0))
        path.Figures.Add(figure)
        Dim handle As Integer = cad.AddCompoundPath(path)
        If handle = 0 Then Return 0
        If Not cad.SetPathNodeMode(handle, 0, 1,
            Avax.AvaxPathNodeMode.Smooth) Then Return 0
        Return handle
    End Function
End Module
```

Related options and reference

SetPointSelectionAspectRatio

API reference

Purpose

Constrains the X/Y width-to-height ratio of the current two-point selection request. It is useful when the host asks for two corners of a proportioned placement rectangle.

Declaration

```
Function SetPointSelectionAspectRatio(ByVal ratio As Double) As Boolean
```

Parameter and result

`ratio` is a finite Double width / height in drawing X/Y units. A positive value enables the constraint; 0 restores free proportions. For example, 4.0 / 3.0 requests a four-to-three rectangle. Negative, NaN and infinite values are rejected.

Returns True when the active request accepts the setting, False if there is no current SelectPoints request for exactly two points or the control is unstarted, disabled, disposed, paused or locked for editing. An invalid value leaves the previous setting unchanged. Start the existing control, begin the two-point request, then apply the ratio.

Point and preview behavior

After the first point is accepted, the control expands the shorter required dimension to obtain the requested ratio, retaining the pointer's X/Y direction relative to the first point. The ratio describes world drawing coordinates; ScreenXFactor remains a separate display scaling setting. Preview and point acceptance use the constrained X/Y values; the input Z values retain the normal SelectPoints semantics. Changing the ratio refreshes the current preview.

The constraint belongs to this request and ends when the request completes or is cancelled/replaced. It does not resize an existing drawing object or change the workspace page. Command = Cancel_c cancels the point request. Assigning a ratio of 0 through this method removes the constraint while retaining the current request. If the constrained rectangle cannot be represented with finite, distinct Double coordinates, the provisional point collapses to the anchor instead of producing nonfinite placement coordinates.

VB.NET example

```
Option Strict On
Public Module SetPointSelectionAspectRatioExample
    ' Supply the existing started control and a positive width/height ratio.
    Public Function BeginProportionedBox(
        cad As Avax.cAvax, widthOverHeight As Double) As Boolean
        Dim requestId As String = "Placement box"
        Dim count As Integer = 2
        Dim x As Double() = {0.0}
        Dim y As Double() = {0.0}
        Dim z As Double() = {0.0}
        Dim showBox As Boolean = True
        Dim freeOfGridAndOrtho As Boolean = True
        If Not cad.SelectPoints(requestId, count, x, y, z,
            showBox, freeOfGridAndOrtho) Then Return False
        Return cad.SetPointSelectionAspectRatio(widthOverHeight)
    End Function
End Module
```

SetPointSelectionPreview

API reference

Purpose

Chooses the temporary shape outline displayed while the user supplies points to an active SelectPoints request. The preview does not add an object. Handle the SelectedPoints event to use the accepted coordinates and create the application's final object.

VB.NET declaration

```
Function SetPointSelectionPreview(ByVal shape As AvaxShapeType, Optional ByVal polygonSides As Integer =
    6) As Boolean
```

Parameters

Parameter	Meaning
shape	For a two-point request: Rectangle_s, Ellipse_s, Circle_s, RegularPolygon_s or Line_s. Arc_s requires a three-point request. Any_s restores the original generic SelectPoints feedback.
polygonSides	Regular polygon side count, from 3 through 10000; default 6. Ignored for other preview shapes.

Return value and behavior

Returns True when the preview choice is accepted. Requires a started, enabled and unpaused control with an active SelectPoints request. A mismatched point count, unsupported enum value or invalid polygon side count returns False and retains the current preview, including a preview previously chosen with SetEditablePointSelectionPreview.

Rectangle and ellipse use opposite corners. Circle and regular polygon use the first point as centre and the cursor distance as radius. Line uses the first point and the cursor position as endpoints. Arc uses start, end and centre in that order; before an end point is accepted, its feedback is a line. A zero-radius curved draft is omitted. Before the first accepted point there is no shape outline.

Any_s restores the request's existing line or box-and-diagonal feedback. A successful enum choice replaces an earlier editable-kind preview. Completing or cancelling the request clears the preview; accepted Double coordinates, Z values and the request ID are retained. Preview changes create no drawing record or Undo entry. Call the method on the control's owning UI thread.

For named previews such as Pie or Closed arc, see SetEditablePointSelectionPreview.

VB.NET example

Pass a started control. This starts a two-point request with ellipse feedback; receive its points through SelectedPoints.

```
Option Strict On
Imports Avax

Public Module PointSelectionPreviewExample
    Public Function BeginEllipse(cad As cAvax) As Boolean
        If Not cad.SelectPoints("ellipse", 2, Nothing, Nothing, Nothing) Then Return False
        Return cad.SetPointSelectionPreview(AvaxShapeType.Ellipse_s)
    End Function
End Module
```

SetProperties

cAvax method

Purpose

Updates metadata and geometry of existing objects from one-based arrays, retaining their handles and types.

VB.NET declaration

```
Function SetProperties(ByRef avMyHandles As Integer(), ByRef iType As AvaxItemType(), ByRef vItemProperies
     As Object(), ByRef vData As Object()) As Boolean
```

Parameters

Name	Type and passing	Meaning
avMyHandles	ByRef avMyHandles As Integer()	One-based distinct object handles, with index 0 reserved. Objects keep their existing handles after a successful update.
iType	ByRef iType As AvaxItemType()	One-based AvaxItemType() matching the requested handles. SetProperties does not convert item types.
vItemProperies	ByRef vItemProperies As Object()	One-based Object() containing per-object string arrays; metadata and geometry are copied.
vData	ByRef vData As Object()	One-based Object() of per-item Double() arrays; see the Property geometry table for all exact layouts. Length includes reserved index 0, which is meaningful only for Curve_i. Block/ExLines_i expose six XYZ bounds. Data_i has no editable geometry.

Use the declaration for the overload you call. ByRef arguments can receive outputs; declare variables and arrays with the exact indicated type. Optional defaults are shown in the declaration.

Return value

Boolean - True on success, false on error.

Remarks

All outer arrays must match the one-based handle array. Handles must be valid and distinct; the item type must match the existing object. Each geometry entry is the per-type Double() record described in Property geometry layouts. Application metadata is copied; independent object keys are retained.

Every replacement is validated before any object is changed. A started, enabled control is required. Invalid arrays, handles, types or geometry return False without a partial batch. A successful batch is one Undo operation and one Change notification. Application event-handler exceptions propagate after committed editing.

Property geometry layouts

The Object() containers keep their declared type. Drawing-coordinate arrays stored inside them use Double(); use the documented per-object layout when reading or replacing those arrays.

Example

Setup

Add an Avax.cAvax Windows Forms control named `cAvax1`. Paste the procedures into the form class. Example setup and supporting files explains references, initialization and file paths.

Add these Windows Forms controls or components to the form. Fields referenced by Handles clauses must be declared WithEvents, as the Visual Studio designer normally does.

Name	Type
Command1	Button

VB.NET example

```
Private Sub Form1_Load(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs) Handles
    MyBase.Load
    Command1.Text = "Get/Set Object Properties"
    cAvax1.StartAvax()
    Call cAvax1.Add_Line(0, 0, 0, 10, 10, 0)
    Call cAvax1.Add_Line(0, 0, 0, 10, 10, 0)
End Sub
Private Sub Form1_FormClosed(ByVal eventSender As System.Object, ByVal eventArgs As
    System.Windows.Forms.FormClosedEventArgs) Handles Me.FormClosed
    cAvax1.EndAvax()
End Sub
Private Sub Command1_Click(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs)
    Handles Command1.Click
    Dim [Handles]() As Integer = Nothing
    Dim iType() As Avax.AvaxItemType = Nothing
    Dim vItemProperies() As Object = Nothing
    Dim vData() As Object = Nothing
    Dim lMax As Integer
    Dim lCnt As Integer
    Dim xyz() As Double = Nothing
    lMax = CAvax1.GetAllHandlesArr([Handles])
    If CAvax1.GetProperties([Handles], iType, vItemProperies, vData) = True Then
        For lCnt = 1 To lMax
            If iType(lCnt) = Avax.AvaxItemType.Line i Then
                xyz = vData(lCnt)
                xyz(1) = xyz(1) + 5
                xyz(2) = xyz(2) + 5
                xyz(4) = xyz(4) + 5
                xyz(5) = xyz(5) + 5
                vData(lCnt) = xyz
            End If
        Next lCnt
        Call CAvax1.SetProperties([Handles], iType, vItemProperies, vData)
    End If
End Sub
```

SetSelectFilter

cAvax method

Purpose

Define before the selection what objects will be selected based on a specific filter

See Selection commands and type filters for one-based handle arrays, point/window selection, modifier keys, type-mask values, and committed selection notifications.

See Immediate actions and selection borders for direct mode changes, filtered selection, one overall Border, outputs, callbacks and accepted Polyline input.

VB.NET declaration

```
Function SetSelectFilter(ByRef lFilter As Integer) As Boolean
```

Parameters

Name	Type and passing	Meaning
lFilter	ByRef lFilter As Integer	Is the specific filter

Use the declaration for the overload you call. ByRef arguments can receive outputs; declare variables and arrays with the exact indicated type. Optional defaults are shown in the declaration.

Return value

Boolean - True on success, false on error.

Example

Setup

Add an Avax.cAvax Windows Forms control named `cAvax1`. Paste the procedures into the form class. Example setup and supporting files explains references, initialization and file paths.

Add these Windows Forms controls or components to the form. Fields referenced by Handles clauses must be declared WithEvents, as the Visual Studio designer normally does.

Name	Type
Label1	Label

VB.NET example

```
Private Sub CAvax1_BeforeClick(ByVal Sender As Object, ByVal e As Avax.cAvax.BeforeClickEventArgs) Handles
     CAvax1.BeforeClick
    Dim lFilter As Integer
    If e.ClickType = Avax.AvaxClickType.Select ct Then
        'Don't select the line
        lFilter = (2 ^ (Avax.AvaxItemType.Polyline i - 1)) Or (2 ^ (Avax.AvaxItemType.Ellipse i - 1))
        Call CAvax1.SetSelectFilter(lFilter)
    End If
End Sub
Private Sub Form1_Load(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs) Handles
    MyBase.Load
    Label1.Text = "Try to select the line"
    cAvax1.StartAvax()
    Call cAvax1.Add_Line(2, 2, 0, 8, 8, 0)
    Call cAvax1.Add_Rectangle(0, 0, 10, 10)
    Call cAvax1.Add_Circle(5, 5, 0, 6)
End Sub
Private Sub Form1_FormClosed(ByVal eventSender As System.Object, ByVal eventArgs As
```

```
    System.Windows.Forms.FormClosedEventArgs) Handles Me.FormClosed
    cAvax1.EndAvax()
End Sub
```

SetSelectRegionBoundingBox

cAvax method

Purpose

Define a parallelogram region where changes to the SelectRegion window are permitted

VB.NET declaration

```
Function SetSelectRegionBoundingBox(ByRef xMin As Double, ByRef yMin As Double, ByRef xMax As Double,
    ByRef yMax As Double) As Boolean
```

Parameters

Name	Type and passing	Meaning
xMin	ByRef xMin As Double	Is the x coordinate of the bottom left extend of the SelectRegion window
yMin	ByRef yMin As Double	Is the y coordinate of the bottom left extend of the SelectRegion window
xMax	ByRef xMax As Double	Is the x coordinate of the top right extend of the SelectRegion window
yMax	ByRef yMax As Double	Is the y coordinate of the top right extend of the SelectRegion window

Use the declaration for the overload you call. ByRef arguments can receive outputs; declare variables and arrays with the exact indicated type. Optional defaults are shown in the declaration.

Return value

Boolean - True on success, false on error.

Remarks

Defines the permitted drag rectangle for the selection region. It does not replace the selected objects' bounds or transform their geometry. The control must be started and enabled, SelectRegionOn_p must be True, and at least one object must be selected.

Coordinates are Double drawing units. Reversed corners are normalized internally without changing the four ByRef arguments. Coincident opposite corners clear the drag constraint. Nonfinite coordinates or a rectangle whose span exceeds the finite Double range return False and retain the previous constraint.

Installing a constraint leaves selection, geometry and Undo history unchanged. GetSelectRegionBorders continues to report the actual selected region. Dragging validates the resulting displayed bounds, including fixed/minimum pixel sizes and physical pen width. Host changes to targets or parameters are checked against the current limits before committing. The same checks apply to nested/imported geometry and native dimensions/hatches.

Numeric changes from v2

Update the following variables, arrays or event-data receivers when porting the corresponding v2 call. The declarations above show the current overloads.

Parameter or value	v2 type	v3 type
xMin	Single	Double
yMin	Single	Double
xMax	Single	Double
yMax	Single	Double

For ByRef parameters, use a variable with the exact v3 type. Preserve array dimensions and the documented indexing. Moving from v2 to v3 explains conversion and legacy file-storage rules.

Example

Setup

Add an Avax.cAvax Windows Forms control named `cAvax1`. Paste the procedures into the form class. Example setup and supporting files explains references, initialization and file paths.

VB.NET example

```
Private Sub Form1_Load(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs) Handles
    MyBase.Load
    Dim h(1) As Integer
    cAvax1.StartAvax()
    Call cAvax1.Add_Rectangle(0, 0, 10, 10)
    h(1) = cAvax1.Add_Line(4, 4, 0, 8, 8, 0)
    Call cAvax1.SelectItems(h)
    Call cAvax1.SetSelectRegionBoundingBox(0, 0, 10, 10)
End Sub
Private Sub Form1_FormClosed(ByVal eventSender As System.Object, ByVal eventArgs As
    System.Windows.Forms.FormClosedEventArgs) Handles Me.FormClosed
    cAvax1.EndAvax()
End Sub
```

SetSVGPalette

cAvax method

Purpose

Assigns the SVG's RGB colors to the current color palette

VB.NET declaration

```
Function SetSVGPalette() As Boolean
```

Return value

Boolean - True on success, false on error.

Example

Setup

Add an Avax.cAvax Windows Forms control named `cAvax1`. Paste the procedures into the form class. Example setup and supporting files explains references, initialization and file paths.

Add these Windows Forms controls or components to the form. Fields referenced by Handles clauses must be declared WithEvents, as the Visual Studio designer normally does.

Name	Type
Command1	Button

VB.NET example

```
Private Sub Form1_Load(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs) Handles
    MyBase.Load
    Dim x As Double
    Dim iClr As Integer
    Command1.Text = "SetSVGPalette"
    cAvax1.StartAvax()
    cAvax1.AutoRedraw = False
    For x = 0 To 146
        iClr = x
        Call CAvax1.Add_Line(x, 0, 0, x, 100, 0, Nothing, iClr, 3, -1, -1)
    Next x
    CAvax1.Command = Avax.AvaxCommand.AutoLimits_c
    cAvax1.AutoRedraw = True
End Sub
Private Sub Form1_FormClosed(ByVal eventSender As System.Object, ByVal eventArgs As
    System.Windows.Forms.FormClosedEventArgs) Handles Me.FormClosed
    cAvax1.EndAvax()
End Sub
Private Sub Command1_Click(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs)
    Handles Command1.Click
    Dim Answer As Boolean
    Answer = cAvax1.SetSVGPalette()
End Sub
```

SetTextStyle

Purpose

Apply a validated style to one text shape as an undoable edit, retaining its handle and editable content.

VB.NET declaration

```
Function SetTextStyle(ByVal handle As Integer, ByVal style As AvaxTextStyle) As Boolean
```

Parameters

handle: a visible, editable text shape. style: the style to copy and apply.

Return value

True on success. False on an invalid style, an unsupported object, locked editing or an unavailable control; a failed preparation leaves the drawing unchanged.

Example

```
Option Strict On
Public Module Help_TextStyle_3
    Public Sub Run(cad As Avax.cAvax, handle As Integer)
        Dim style As Avax.AvaxTextStyle = cad.GetTextStyle(handle)
        If style IsNot Nothing Then
            style.FillEnabled = False
            style.OutlineColor = System.Drawing.Color.Blue
            style.OutlineWidth = 0.15
            If Not cad.SetTextStyle(handle, style) Then
                Throw New System.InvalidOperationException(cad.LastOperationError)
            End If
        End If
    End Sub
End Module
```

Related options and reference

SetTextStyles

Purpose

Apply independent styles to several text shapes atomically in one Undo step.

VB.NET declaration

```
Function SetTextStyles(ByVal ids As Integer(), ByVal styles As AvaxTextStyle()) As Boolean
```

Parameters

ids: a handle array with index 0 reserved. styles: a zero-based array containing exactly one non-null style per distinct handle in ids(1) onward. All handles must belong to visible, editable text shapes.

Return value

True when every style is applied. False when any entry is invalid or editing is unavailable; no partial edit is committed.

Example

```
Option Strict On
Public Module Help_TextStyle_4
    Public Sub Run(cad As Avax.cAvax, firstHandle As Integer, secondHandle As Integer)
        Dim first As Avax.AvaxTextStyle = cad.GetTextStyle(firstHandle)
        Dim second As Avax.AvaxTextStyle = cad.GetTextStyle(secondHandle)
        If first IsNot Nothing AndAlso second IsNot Nothing Then
            first.Bold = True
            second.Italic = True
            cad.SetTextStyles(New Integer() {0, firstHandle, secondHandle},
                              New Avax.AvaxTextStyle() {first, second})
        End If
    End Sub
End Module
```

Related options and reference

SingleToLong

cAvax method

Purpose

Converts Double drawing coordinates to integer window pixel coordinates. The historical method name is retained for source compatibility.

VB.NET declaration

```
Function SingleToLong(ByRef xS As Double, ByRef yS As Double, ByRef xL As Integer, ByRef yL As Integer) As
    Boolean
```

Parameters

Name	Type and passing	Meaning
xS	ByRef xS As Double	Is the drawing coordinate x
yS	ByRef yS As Double	Is the drawing coordinate y
xL	ByRef xL As Integer	Is the window coordinate x
yL	ByRef yL As Integer	Is the window coordinate y

Use the declaration for the overload you call. ByRef arguments can receive outputs; declare variables and arrays with the exact indicated type. Optional defaults are shown in the declaration.

Return value

Boolean - True on success, false on error.

Numeric changes from v2

Update the following variables, arrays or event-data receivers when porting the corresponding v2 call. The declarations above show the current overloads.

Parameter or value	v2 type	v3 type
xS	Single	Double
yS	Single	Double

For ByRef parameters, use a variable with the exact v3 type. Preserve array dimensions and the documented indexing. Moving from v2 to v3 explains conversion and legacy file-storage rules.

Example

Setup

Add an Avax.cAvax Windows Forms control named `cAvax1`. Paste the procedures into the form class. Example setup and supporting files explains references, initialization and file paths.

Add these Windows Forms controls or components to the form. Fields referenced by Handles clauses must be declared WithEvents, as the Visual Studio designer normally does.

Name	Type
Label1	Label

VB.NET example

```
Private Sub CAvax1_OnMouseMove(ByVal Sender As Object, ByVal e As Avax.cAvax.MouseMoveEventArgs) Handles
    CAvax1.OnMouseMove
    Dim xMinSel, yMinSel As Double
```

```
    Dim xMaxSel, yMaxSel As Double
    Dim xL1, yL1 As Integer
    Dim xL2, yL2 As Integer
    Dim xC, yC As Integer
    Dim sMsg As String = String.Empty
    If cAvax1.GetBoundsSelectBox(xMinSel, yMinSel, xMaxSel, yMaxSel) = True Then
        Call cAvax1.SingleToLong(xMinSel, yMinSel, xL1, yL1)
        Call cAvax1.SingleToLong(xMaxSel, yMaxSel, xL2, yL2)
        xC = xL1 + (xL2 - xL1) / 2
        yC = yL1 + (yL2 - yL1) / 2
    End If
    If e.x = xC And e.y = yC Then
        sMsg = "The center of the select box cursor has the same "
        sMsg = sMsg & "window coordinates with the mouse pointer"
    End If
    Label1.Text = sMsg
End Sub
Private Sub Form1_Load(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs) Handles
    MyBase.Load
    cAvax1.StartAvax()
End Sub
Private Sub Form1_FormClosed(ByVal eventSender As System.Object, ByVal eventArgs As
    System.Windows.Forms.FormClosedEventArgs) Handles Me.FormClosed
    cAvax1.EndAvax()
End Sub
```

SnapToStepSet

cAvax method

Purpose

Sets the horizontal and vertical snap spacing in drawing units. Positive values on both axes enable snapping; a zero on either axis disables it.

VB.NET declaration

```
Function SnapToStepSet(ByRef stepX As Double, ByRef stepY As Double) As Boolean
```

Parameters

Name	Type and passing	Meaning
stepX	ByRef stepX As Double	Is the spacing of the x snap movements
stepY	ByRef stepY As Double	Is the spacing of the y snap movements

Use the declaration for the overload you call. ByRef arguments can receive outputs; declare variables and arrays with the exact indicated type. Optional defaults are shown in the declaration.

Return value

Boolean - True on success, false on error.

Remarks

Call StartAvax first and keep the control enabled. stepX and stepY are nonnegative finite Double values in drawing units. Returns False for a negative or nonfinite value, or an unstarted, ended, disabled or disposed control; a rejected call retains both previous steps and the snap status. Returns True when accepted, including a zero step. A zero on either axis disables snapping. Positive steps enable snapping on both axes. The arguments are not modified. Changing the steps keeps the current drawing command and its accepted points; it does not add a drawing undo record. Read SnapToStepX, SnapToStepY and SnapToStepStatus to obtain the current configuration.

Numeric changes from v2

Update the following variables, arrays or event-data receivers when porting the corresponding v2 call. The declarations above show the current overloads.

Parameter or value	v2 type	v3 type
stepX	Single	Double
stepY	Single	Double

For ByRef parameters, use a variable with the exact v3 type. Preserve array dimensions and the documented indexing. Moving from v2 to v3 explains conversion and legacy file-storage rules.

Example

Setup

Add an Avax.cAvax Windows Forms control named `cAvax1`. Paste the procedures into the form class. Example setup and supporting files explains references, initialization and file paths.

Add these Windows Forms controls or components to the form. Fields referenced by Handles clauses must be declared WithEvents, as the Visual Studio designer normally does.

Name	Type
Command1	Button

Name	Type
Command2	Button
Text1	TextBox

VB.NET example

```
Private Sub Command2_Click(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs)
    Handles Command2.Click
    CAvax1.Command = Avax.AvaxCommand.Circle_c
End Sub
Private Sub Form1_Load(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs) Handles
    MyBase.Load
    cAvax1.StartAvax()
End Sub
Private Sub Form1_FormClosed(ByVal eventSender As System.Object, ByVal eventArgs As
    System.Windows.Forms.FormClosedEventArgs) Handles Me.FormClosed
    cAvax1.EndAvax()
End Sub
Private Sub Command1_Click(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs)
    Handles Command1.Click
    Dim stepX As Double
    Dim stepY As Double
    Dim Answer As Boolean
    Dim ii As Integer
    Dim txt As String = String.Empty
    stepX = Val(Text1.Text)
    stepY = stepX
    If stepX <= 0 Then
        ii = MsgBox("Snap step is not valid, Is set to 5 Units", MsgBoxStyle.Exclamation, "SnapSet")
        Answer = cAvax1.SnapToStepSet(5, 5)
    Else
        txt = "Snap is set to " & Str(stepX) & " units"
        ii = MsgBox(txt, , "SnapSet")
        Answer = cAvax1.SnapToStepSet(stepX, stepY)
    End If
End Sub
```

SplitPathSegment

Purpose

Divides one line, arc or cubic Bezier segment into two segments with the same geometric path.

Declaration

```
Function SplitPathSegment(ByVal handle As Integer, ByVal figure As Integer, ByVal segment As Integer,
    Optional ByVal t As Double = 0.5) As Boolean
```

Parameters and result

`handle` identifies the path. `figure` and `segment` are zero-based indices. `t` is a finite parameter strictly between 0 and 1; default 0.5. It is a fraction of line traversal, arc sweep or Bezier parameter, not a general curve-length fraction. The original entry becomes the first part; the second is inserted at segment + 1. Later segment and node indices therefore increase. The new joining node uses Smooth mode.

Returns True for an accepted edit, False for invalid indices/parameters or an object that cannot be edited. The object handle and figure closure are retained. An accepted edit removes any parametric recipe and participates in Undo/Redo.

VB.NET example

```
Option Strict On
Public Module SplitPathSegmentExample
    Public Function AddSplitCubic(cad As Avax.cAvax) As Integer
        Dim path As New Avax.AvaxPathData
        Dim figure As New Avax.AvaxPathFigure
        figure.Segments.Add(Avax.AvaxPathSegment.BezierTo(
            10.0, 20.0, 20.0, 20.0, 30.0, 0.0))
        path.Figures.Add(figure)
        Dim handle As Integer = cad.AddCompoundPath(path)
        If handle = 0 OrElse Not cad.SplitPathSegment(handle, 0, 0, 0.5) Then Return 0
        System.Diagnostics.Debug.WriteLine(cad.GetCompoundPath(handle).Figures(0).Segments.Count)
        Return handle
    End Function
End Module
```

Related options and reference

StartAvax

cAvax method

Purpose

Initializes the control and opens or creates a drawing. With no file name, the control creates an owned temporary drawing. Call this before using the drawing API, and call EndAvax when the owning form closes.

VB.NET declarations

```
Function StartAvax() As Boolean

Function StartAvax(ByRef sFileName As String) As Boolean
```

Parameters

Name	Type and passing	Meaning
sFileName	ByRef sFileName As String	String ByRef input, unchanged. Empty/Nothing creates a fresh owned temporary drawing. Otherwise load/create the resolved named drawing; relative paths use DirData, and an absent extension means .avx.

Use the declaration for the overload you call. ByRef arguments can receive outputs; declare variables and arrays with the exact indicated type. Optional defaults are shown in the declaration.

Return value

Boolean - True on success, false on error.

Remarks

An empty file name creates a temporary drawing. A named drawing is resolved using DirData; an omitted extension is treated as .avx. Check the Boolean result before adding entities. Keep calls on the control's owning Windows Forms thread.

Example

Setup

Add an Avax.cAvax Windows Forms control named `cAvax1`. Paste the procedures into the form class. Example setup and supporting files explains references, initialization and file paths.

VB.NET example

```
Private Sub Form1_Load(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs) Handles
    MyBase.Load
    Dim sFileName As String = String.Empty
    sFileName = ""
    Call cAvax1.StartAvax(sFileName)
End Sub
Private Sub Form1_FormClosed(ByVal eventSender As System.Object, ByVal eventArgs As
    System.Windows.Forms.FormClosedEventArgs) Handles Me.FormClosed
    cAvax1.EndAvax()
End Sub
```

StretchItems

cAvax method

Purpose

Stretches the objects described in a specific handles array by the same proportion that a specific parallelogram has been changed from its original coordinates to new changed coordinates

VB.NET declaration

```
Function StretchItems(ByRef avMyHandles As Integer(), ByRef xOldMin As Double, ByRef yOldMin As Double,
    ByRef xOldMax As Double, ByRef yOldMax As Double, ByRef xNewMin As Double, ByRef yNewMin As Double, ByRef
    xNewMax As Double, ByRef yNewMax As Double) As Boolean
```

Parameters

Name	Type and passing	Meaning
avMyHandles	ByRef avMyHandles As Integer()	Input Integer() of the entity handles to process. Logical entries start at index 1; index 0 is reserved.
xOldMin	ByRef xOldMin As Double	Is the x original coordinate of the bottom left extend of the parallelogram
yOldMin	ByRef yOldMin As Double	Is the y original coordinate of the bottom left extend of the parallelogram
xOldMax	ByRef xOldMax As Double	Is the x original coordinate of the top right extend of the parallelogram
yOldMax	ByRef yOldMax As Double	Is the y original coordinate of the top right extend of the parallelogram
xNewMin	ByRef xNewMin As Double	Is the x new changed coordinate of the bottom left extend of the parallelogram
yNewMin	ByRef yNewMin As Double	Is the y new changed coordinate of the bottom left extend of the parallelogram
xNewMax	ByRef xNewMax As Double	Is the x new changed coordinate of the top right extend of the parallelogram
yNewMax	ByRef yNewMax As Double	Is the x new changed coordinate of the top right extend of the parallelogram

Use the declaration for the overload you call. ByRef arguments can receive outputs; declare variables and arrays with the exact indicated type. Optional defaults are shown in the declaration.

Return value

Boolean - True on success, false on error.

Remarks

Maps the original XY rectangle to the new XY rectangle and applies the same proportions to every requested object. For each nondegenerate axis, the scale is (newMax-newMin)/(oldMax-oldMin); the translation is newMin-oldMin*scale. Crossed bounds produce a negative scale. The source objects need not be inside the reference rectangle. This method transforms complete objects; interactive window stretching has a separate vertex-selection contract.

An exactly zero old span can translate when the new span is also zero, using scale 1; it cannot acquire a nonzero extent. An intentional zero new span can collapse supported geometry. A nonzero new span whose ratio would round to zero is rejected, so a numeric underflow cannot silently flatten the drawing. Unchanged rectangles return False without an edit.

All eight bounds must be finite Double values. Complete span differences and pivot offsets are evaluated before rounding, allowing valid transformations even when an intermediate subtraction or product would overflow. A required coefficient, offset or resulting geometry outside the supported Double representation returns False without changing the drawing, caller handles, accepted construction input or previous Redo.

A translation raises BeforeMove; other maps raise BeforeTransform. These handlers can replace targets or parameters, or cancel the operation. The existing handles and Z coordinates are retained, including embedded resources and nested content. Live external

geometry retains the map when regenerated. Document font sizing follows FixedDocumentTextSizeOn_p.

Pass a one-based Integer handle array; index 0 is ignored and duplicate handles are processed once. Geometry uses Double. The control must be started and enabled. PauseAvax and LockEdit govern interactive input and do not prohibit this direct call. Before events can change the proposed targets or parameters, or cancel the operation. Caller arrays are separate from event arrays. The complete batch is prepared before editing. Invalid handles, unrepresentable geometry or unavailable copy capacity return False without a partial edit. A successful batch is one Undo operation and raises Change without NewUserItems. PauseAvaxEvents suppresses notifications. Application event-handler exceptions propagate; they do not roll back an edit already committed. A callback that replaces the drawing or starts new input prevents the older operation from overwriting that state.

Numeric changes from v2

Update the following variables, arrays or event-data receivers when porting the corresponding v2 call. The declarations above show the current overloads.

Parameter or value	v2 type	v3 type
xOldMin	Single	Double
yOldMin	Single	Double
xOldMax	Single	Double
yOldMax	Single	Double
xNewMin	Single	Double
yNewMin	Single	Double
xNewMax	Single	Double
yNewMax	Single	Double

For ByRef parameters, use a variable with the exact v3 type. Preserve array dimensions and the documented indexing. Moving from v2 to v3 explains conversion and legacy file-storage rules.

Example

Setup

Add an Avax.cAvax Windows Forms control named `cAvax1`. Paste the procedures into the form class. Example setup and supporting files explains references, initialization and file paths.

Add these Windows Forms controls or components to the form. Fields referenced by Handles clauses must be declared WithEvents, as the Visual Studio designer normally does.

Name	Type
Command1	Button

VB.NET example

```
Private Sub Form1_Load(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs) Handles
    MyBase.Load
    cAvax1.StartAvax()
    Call cAvax1.Add_Line(0, 0, 0, 3, 3, 0)
    Call cAvax1.Add_Line(0, 3, 0, 3, 3, 0)
End Sub
Private Sub Form1_FormClosed(ByVal eventSender As System.Object, ByVal eventArgs As
    System.Windows.Forms.FormClosedEventArgs) Handles Me.FormClosed
    cAvax1.EndAvax()
End Sub
Private Sub Command1_Click(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs)
    Handles Command1.Click
    Dim [Handles]() As Integer = Nothing
    Dim xOldMin As Double
    Dim yOldMin As Double
    Dim xOldMax As Double
    Dim yOldMax As Double
```

```
    Dim xNewMin As Double
    Dim yNewMin As Double
    Dim xNewMax As Double
    Dim yNewMax As Double
    Dim z1, z2 As Double
    Dim Answer As Boolean
    Call cAvax1.CalcBoundsDrawing()
    Call cAvax1.GetBoundsDrawing(xOldMin, yOldMin, z1, xOldMax, yOldMax, z2)
    If CAvax1.GetAllHandlesArr([Handles]) > 0 Then
        xNewMin = xOldMin + 5
        yNewMin = yOldMin + 5
        xNewMax = xOldMax + 9
        yNewMax = yOldMax + 9
        Answer = CAvax1.StretchItems([Handles], xOldMin, yOldMin, xOldMax, yOldMax, xNewMin, yNewMin,
    xNewMax, yNewMax)
    End If
End Sub
```

TransformItems

cAvax method

Purpose

Applies a one-based 3-by-3 Double affine matrix in XY. The Stay overload can retain originals and return transformed copies.

VB.NET declarations

```
Function TransformItems(ByRef avMyHandles As Integer(), ByRef TransArr33 As Double(,)) As Boolean

Function TransformItems(ByRef avMyHandles As Integer(), ByRef TransArr33 As Double(,), ByRef
    fTransformStay As Boolean) As Boolean
```

Parameters

Name	Type and passing	Meaning
avMyHandles	ByRef avMyHandles As Integer()	Input Integer() of the entity handles to process. Logical entries start at index 1; index 0 is reserved.
TransArr33	ByRef TransArr33 As Double(,)	A 3 by 3 Double matrix for the XY affine transformation. Use [a,b,0; c,d,0; tx,ty,1], so x'=a*x+c*y+tx and y'=b*x+d*y+ty.
fTransformStay	ByRef fTransformStay As Boolean	A Boolean expression that if it is true then the original objects are retained

Use the declaration for the overload you call. ByRef arguments can receive outputs; declare variables and arrays with the exact indicated type. Optional defaults are shown in the declaration.

Return value

Boolean - True on success, false on error.

Remarks

Use a Double(3,3) CLR array whose active matrix indices are 1 through 3; row and column 0 are reserved when present. An actual CLR array with lower bounds 1 is also accepted if both dimensions include indices 1 through 3. Padding outside these indices is ignored. The row-vector convention is xNew=x*m(1,1)+y*m(2,1)+m(3,1), yNew=x*m(1,2)+y*m(2,2)+m(3,2). All nine active values must be finite. The third column does not affect XY, and there is no division by W; Z coordinates remain unchanged. Use an affine third column of (0,0,1) for clarity. Finite other values are accepted with the same XY meaning for every object type and subsequent transformation.

The caller matrix is retained unchanged, including its array identity and padding. BeforeTransform receives an isolated matrix and can edit or replace it, replace targets, or cancel. Complete coordinate and stored-frame expressions round once to Double; unrepresentable output rejects the complete batch.

The short overload transforms existing objects. With fTransformStay=True, originals remain and the caller ByRef handle array receives copied handles. Nested geometry, embedded resources, native dimensions/hatches and live external references follow the same map. Document font sizing follows FixedDocumentTextSizeOn_p.

Pass a one-based Integer handle array; index 0 is ignored and duplicate handles are processed once. Geometry uses Double. The control must be started and enabled. PauseAvax and LockEdit govern interactive input and do not prohibit this direct call. Before events can change the proposed targets or parameters, or cancel the operation. Caller arrays are separate from event arrays. The complete batch is prepared before editing. Invalid handles, unrepresentable geometry or unavailable copy capacity return False without a partial edit. A successful batch is one Undo operation and raises Change without NewUserItems. PauseAvaxEvents suppresses notifications. Application event-handler exceptions propagate; they do not roll back an edit already committed. A callback that replaces the drawing or starts new input prevents the older operation from overwriting that state.

Numeric changes from v2

Update the following variables, arrays or event-data receivers when porting the corresponding v2 call. The declarations above show the current overloads.

Parameter or value	v2 type	v3 type
TransArr33	Single(,)	Double(,)

For ByRef parameters, use a variable with the exact v3 type. Preserve array dimensions and the documented indexing. Moving from v2 to v3 explains conversion and legacy file-storage rules.

Example

Setup

Add an Avax.cAvax Windows Forms control named `cAvax1`. Paste the procedures into the form class. Example setup and supporting files explains references, initialization and file paths.

Add these Windows Forms controls or components to the form. Fields referenced by Handles clauses must be declared WithEvents, as the Visual Studio designer normally does.

Name	Type
Command1	Button

VB.NET example

```
Private Sub Form1_Load(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs) Handles
    MyBase.Load
    cAvax1.StartAvax()
    Call cAvax1.Add_Line(0, 0, 0, 10, 0, 0)
    Call cAvax1.Add_Line(0, 10, 0, 0, 0, 0)
End Sub
Private Sub Form1_FormClosed(ByVal eventSender As System.Object, ByVal eventArgs As
    System.Windows.Forms.FormClosedEventArgs) Handles Me.FormClosed
    cAvax1.EndAvax()
End Sub
Private Sub Command1_Click(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs)
    Handles Command1.Click
    Static i As Integer
    Static x As Double
    Dim [Handles]() As Integer = Nothing

    Dim TransArr33(,) As Double = Nothing
    Dim fTransformStay As Boolean
    Dim Answer As Boolean

    fTransformStay = True
    i = i + 1
    x = x + i * 10
    If CAvax1.GetAllHandlesArr([Handles]) > 0 Then
        ReDim TransArr33(3, 3)
        ' Reflect about the line through (x, 0) and (x + 10, 10).
        ' Row-vector matrix: newX = oldY + x; newY = oldX - x.
        TransArr33(1, 2) = 1
        TransArr33(2, 1) = 1
        TransArr33(3, 1) = x
        TransArr33(3, 2) = -x
        TransArr33(3, 3) = 1
        Answer = CAvax1.TransformItems([Handles], TransArr33, fTransformStay)
    End If
    CAvax1.Command = Avax.AvaxCommand.AutoLimits_C
End Sub
```

UnitX

cAvax method

Purpose

Returns the width of one pixel (in drawing units) of the current zoom

VB.NET declaration

```
Function UnitX() As Double
```

Return value

Double - Is the width of one pixel into a drawing unit of the current zoom

Numeric changes from v2

Update the following variables, arrays or event-data receivers when porting the corresponding v2 call. The declarations above show the current overloads.

Parameter or value	v2 type	v3 type
Return value	Single	Double

For ByRef parameters, use a variable with the exact v3 type. Preserve array dimensions and the documented indexing. Moving from v2 to v3 explains conversion and legacy file-storage rules.

Example

Setup

Add an Avax.cAvax Windows Forms control named `cAvax1`. Paste the procedures into the form class. Example setup and supporting files explains references, initialization and file paths.

Add these Windows Forms controls or components to the form. Fields referenced by Handles clauses must be declared WithEvents, as the Visual Studio designer normally does.

Name	Type
Command1	Button
Command2	Button
Label1	Label

VB.NET example

```
Private Sub Form1_Load(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs) Handles
    MyBase.Load
    cAvax1.StartAvax()
End Sub
Private Sub Form1_FormClosed(ByVal eventSender As System.Object, ByVal eventArgs As
    System.Windows.Forms.FormClosedEventArgs) Handles Me.FormClosed
    cAvax1.EndAvax()
End Sub
Private Sub Command2_Click(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs)
    Handles Command2.Click
    CAvax1.Command = Avax.AvaxCommand.Zoom_c
End Sub
Private Sub Command1_Click(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs)
    Handles Command1.Click
    Dim x, y As Double
    x = cAvax1.UnitX()
    y = cAvax1.UnitY()
    Label1.Text = "UNIT X = " & Format(x, "####.#######") & " UNIT Y = " & Format(y, "####.#######")
```

```
End Sub
```

UnitY

cAvax method

Purpose

Returns the height of one pixel (in drawing units) of the current zoom

VB.NET declaration

```
Function UnitY() As Double
```

Return value

Double -Is the heigth of one pixel into a drawing unit of the current zoom

Numeric changes from v2

Update the following variables, arrays or event-data receivers when porting the corresponding v2 call. The declarations above show the current overloads.

Parameter or value	v2 type	v3 type
Return value	Single	Double

For ByRef parameters, use a variable with the exact v3 type. Preserve array dimensions and the documented indexing. Moving from v2 to v3 explains conversion and legacy file-storage rules.

Example

Setup

Add an Avax.cAvax Windows Forms control named `cAvax1`. Paste the procedures into the form class. Example setup and supporting files explains references, initialization and file paths.

Add these Windows Forms controls or components to the form. Fields referenced by Handles clauses must be declared WithEvents, as the Visual Studio designer normally does.

Name	Type
Command1	Button
Command2	Button
Label1	Label

VB.NET example

```
Private Sub Form1_Load(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs) Handles
    MyBase.Load
    cAvax1.StartAvax()
End Sub
Private Sub Form1_FormClosed(ByVal eventSender As System.Object, ByVal eventArgs As
    System.Windows.Forms.FormClosedEventArgs) Handles Me.FormClosed
    cAvax1.EndAvax()
End Sub
Private Sub Command2_Click(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs)
    Handles Command2.Click
    CAvax1.Command = Avax.AvaxCommand.Zoom_c
End Sub
Private Sub Command1_Click(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs)
    Handles Command1.Click
    Dim x, y As Double
    x = cAvax1.UnitX()
    y = cAvax1.UnitY()
    Label1.Text = "UNIT X = " & Format(x, "####.#######") & " UNIT Y = " & Format(y, "####.#######")
```

```
End Sub
```

Unlock

Purpose

Validates a v3 runtime serial number and applies it to the current process. Existing controls and controls created later use that serial unless they have an individual UserSerialNumber value. This method does not contact an activation server or write a persistent activation record. Interactive activation uses the separate v3 activation window.

VB.NET declaration

```
Shared Function Unlock(ByVal SN As String) As Boolean
```

Parameters

Parameter	Meaning
SN	A runtime serial issued for AVAX Vector .NET v3. Leading/trailing whitespace and letter case are normalized. Nothing, malformed values and serials for previous product versions are invalid.

Return value and behavior

True for a valid v3 serial; False otherwise. An invalid replacement clears the process runtime serial and restores evaluation behavior for controls that inherit it. A per-control UserSerialNumber remains separate. Call Unlock once during application startup and check the Boolean result.

Example

```
Option Strict On
Imports Avax

Public Module Help_Unlock
    Public Function Enable(runtimeSerial As String) As Boolean
        Return cAvax.Unlock(runtimeSerial)
    End Function
End Module
```

Related options and reference

ZoomIn

cAvax method

Purpose

Use this method if you want to zoom in the drawing

VB.NET declaration

```
Sub ZoomIn()
```

Result

This procedure has no return value. Any outputs are supplied through the documented ByRef arguments.

Remarks

Zooms in around the pointer when it is inside the drawing canvas, or around the canvas center otherwise. Uses the ZoomFactor_p divisor: each visible span is multiplied by 1-2/divisor. The default divisor 10 gives 80% of the previous span; 25 gives 92%. Honors the current Y origin, horizontal display factor, rulers, scroll bars and work-area limits. Closes an active magnifier. Preserves drawing geometry, selection and the current command, including points already entered. Records useful Previous/Undo zoom history and publishes ZoomChange after an actual change. Recursive ZoomIn/ZoomOut calls from that notification are ignored. Inactive, disabled and disposed controls, an unchanged scale and windows outside representable Double coordinates produce no zoom. Complete window coordinates retain Double precision, including subnormal spans; they are not rounded through a one-pixel unit. Exceptions raised by a ZoomChange handler propagate unchanged, and a later zoom remains available.

Example

Setup

Add an Avax.cAvax Windows Forms control named `cAvax1`. Paste the procedures into the form class. Example setup and supporting files explains references, initialization and file paths.

Add these Windows Forms controls or components to the form. Fields referenced by Handles clauses must be declared WithEvents, as the Visual Studio designer normally does.

Name	Type
Button1	Button
Button2	Button

VB.NET example

```
Private Sub Form1_Load(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs) Handles
    MyBase.Load
    Call CAvax1.StartAvax()
    Call CAvax1.Add_Line(0, 5, 0, 10, 5, 0, Nothing, 1, 1, -1, -1)
    Call CAvax1.Add_Line(5, 0, 0, 5, 10, 0, Nothing, 1, 1, -1, -1)
    Call CAvax1.Add_Circle(5, 5, 0, 3, Nothing, 5, -1, -1, -1)
    CAvax1.Command = Avax.AvaxCommand.AutoLimits_c
End Sub
Private Sub Form1_FormClosed(ByVal eventSender As System.Object, ByVal eventArgs As
    System.Windows.Forms.FormClosedEventArgs) Handles Me.FormClosed
    Call cAvax1.EndAvax()
End Sub
Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
    Button1.Click
    Call CAvax1.ZoomIn()
End Sub
Private Sub Button2_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
    Button2.Click
    Call CAvax1.ZoomOut()
End Sub
```

ZoomOut

cAvax method

Purpose

Use this method if you want to zoom out the drawing

VB.NET declaration

```
Sub ZoomOut()
```

Result

This procedure has no return value. Any outputs are supplied through the documented ByRef arguments.

Remarks

Zooms out around the pointer inside the drawing canvas, or around the canvas center otherwise. Uses the ZoomFactor_p divisor: each visible span is multiplied by 1+2/divisor. The default divisor 10 gives 120% of the previous span. Honors the current Y origin, horizontal display factor, rulers and scroll bars. Clamps to the work area instead of using unrelated Previous history when a limit is reached; touching one edge does not prevent zooming out. At the complete work area it is a no-op. Closes an active magnifier. Preserves geometry, selection and accepted command points, records useful Previous/Undo zoom history, and publishes ZoomChange after an actual change. Recursive ZoomIn/ZoomOut calls from that notification are ignored. Inactive, disabled and disposed controls, an unchanged scale and windows outside representable Double coordinates produce no zoom. Complete window coordinates retain Double precision, including subnormal spans. Exceptions raised by a ZoomChange handler propagate unchanged, and a later zoom remains available.

Example

Setup

Add an Avax.cAvax Windows Forms control named `cAvax1`. Paste the procedures into the form class. Example setup and supporting files explains references, initialization and file paths.

Add these Windows Forms controls or components to the form. Fields referenced by Handles clauses must be declared WithEvents, as the Visual Studio designer normally does.

Name	Type
Button1	Button
Button2	Button

VB.NET example

```
Private Sub Form1_Load(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs) Handles
    MyBase.Load
    Call CAvax1.StartAvax()
    Call CAvax1.Add_Line(0, 5, 0, 10, 5, 0, Nothing, 1, 1, -1, -1)
    Call CAvax1.Add_Line(5, 0, 0, 5, 10, 0, Nothing, 1, 1, -1, -1)
    Call CAvax1.Add_Circle(5, 5, 0, 3, Nothing, 5, -1, -1, -1)
    CAvax1.Command = Avax.AvaxCommand.AutoLimits_c
End Sub
Private Sub Form1_FormClosed(ByVal eventSender As System.Object, ByVal eventArgs As
    System.Windows.Forms.FormClosedEventArgs) Handles Me.FormClosed
    Call cAvax1.EndAvax()
End Sub
Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
    Button1.Click
    Call CAvax1.ZoomIn()
End Sub
Private Sub Button2_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
    Button2.Click
    Call CAvax1.ZoomOut()
```

```
End Sub
```

ZoomWorkspacePage

Purpose

Fit the view around the sheet, with room for its label.

VB.NET declaration

```
Sub ZoomWorkspacePage()
```

Parameters

None.

Return value and behavior

No return value. Changes only the view zoom; preserves PageScale and geometry. Requires a started control; stopped or disposed controls ignore this call. Accounts for ScreenXFactor and either screen origin, and records the view for Previous. An unrepresentable view throws ArgumentException before changing the view or publishing ZoomChange.

Example

Call this procedure on the owning UI thread with a control already initialized by `StartAvax()`. Handles must belong to that control. File paths are supplied by the caller.

```
Option Strict On
Imports System
Imports Avax

Public Module Help_ZoomWorkspacePage
    Public Sub Run(cad As cAvax)
        cad.WorkspaceMode = AvaxWorkspaceMode.Page
        cad.ZoomWorkspacePage()
    End Sub
End Module
```

Related options and reference

Properties

Use the linked topic list in Contents or the PDF bookmarks to locate a member or guide.

Properties

Select a topic for its purpose, declaration, argument descriptions and example.

- AutoRedraw
- Command
- ContinueUndo
- filename
- FontBold
- FontCharset
- FontItalic
- FontName
- FontSize
- FontStrikethrough
- FontUnderline
- hdc
- hwnd
- IsChanged
- Language
- LastHandle
- LockEdit
- PauseAvax
- PauseAvaxEvents
- Pen3DElevetion
- Pen3DHeight
- PenColor
- PenFontNumber
- PenFontScale
- PenLayer
- PenLineStyle
- PenLineStyleScale
- PenPointRadius
- PenPointStyle
- PenWidth
- ScrollBars
- SnapToStepStatus
- SnapToStepX
- SnapToStepY
- UndoRecorderOn
- UserName
- UserSerialNumber

- XMLDoc

Additional members

- AvaxDimensionStyle.AngularMeasurementScale
- AvaxDimensionStyleOverrides.AngularMeasurementScale
- AvaxTextStyle.Alignment
- AvaxTextStyle.Bold
- AvaxTextStyle.FillColor
- AvaxTextStyle.FillEnabled
- AvaxTextStyle.FontHeight
- AvaxTextStyle.FontName
- AvaxTextStyle.Italic
- AvaxTextStyle.LetterSpacing
- AvaxTextStyle.LineSpacing
- AvaxTextStyle.Opacity
- AvaxTextStyle.OutlineColor
- AvaxTextStyle.OutlineWidth
- AvaxTextStyle.Strikethrough
- AvaxTextStyle.Underline
- AvaxTextStyle.WordSpacing
- BackColor
- BorderStyle
- CommandFilePicker
- CompoundPathClosed
- CompoundPathSegmentKind
- DefaultExtension
- DrawingUnitMM
- Enabled
- ForeColor
- HasWorkspaceSettings
- IsCreatingCompoundPath
- LastImportWarnings
- LastOperationError
- OriginVisible
- PageBackgroundColor
- PageBackgroundPrintExport
- PageCustomHeightMM
- PageCustomWidthMM
- PageHeightMM
- PageMarginMM
- PageMarginsVisible

- PageNumber
- PageOrientation
- PageOriginX
- PageOriginY
- PagePaperSize
- PagePrintUnitMM
- PageScale
- PageShadowVisible
- PageTitle
- PageWidthMM
- ParameterDialogInput
- PenColorByLayer
- PenLineStyleByLayer
- PreserveSelectionAppearance
- ResolvedObjectSnap
- RulerX
- RulerY
- WorkspaceMode
- modbutton.NonThemeStyle

AutoRedraw

cAvax property

Purpose

Gets or sets automatic display updates. Set False while making a batch of drawing changes, then restore True to display the result. This does not move the operations to a background thread.

VB.NET declaration

```
Property AutoRedraw As Boolean
```

Property value

Type	Access	Meaning
Boolean	Read/write	A Boolean expression that defines if the property is True or False.

Example

Setup

Add an Avax.cAvax Windows Forms control named `cAvax1`. Paste the procedures into the form class. Example setup and supporting files explains references, initialization and file paths.

Add these Windows Forms controls or components to the form. Fields referenced by Handles clauses must be declared WithEvents, as the Visual Studio designer normally does.

Name	Type
Command1	Button
Command2	Button

VB.NET example

```
Private Sub Form1_Load(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs) Handles
    MyBase.Load
    cAvax1.StartAvax()
End Sub
Private Sub Form1_FormClosed(ByVal eventSender As System.Object, ByVal eventArgs As
    System.Windows.Forms.FormClosedEventArgs) Handles Me.FormClosed
    cAvax1.EndAvax()
End Sub
Private Sub Command1_Click(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs)
    Handles Command1.Click
    Dim xStart, yStart As Double
    Dim iClr As Integer
    xStart = 0
    yStart = -2
    iClr = 1
    cAvax1.AutoRedraw = False
    Call AddLines(xStart, yStart, iClr)
    cAvax1.AutoRedraw = True
End Sub
Private Sub Command2_Click(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs)
    Handles Command2.Click
    Dim xStart, yStart As Double
    Dim iClr As Integer
    xStart = 15
    yStart = -2
    iClr = 5
    Call AddLines(xStart, yStart, iClr)
```

```
End Sub
Private Sub AddLines(ByRef xStart As Double, ByRef yStart As Double, ByRef iClr As Integer)
    Dim x1 As Double
    Dim y1 As Double
    Dim z1 As Double
    Dim x2 As Double
    Dim y2 As Double
    Dim z2 As Double
    Dim vItemProperties As Object = Nothing
    Dim iWidth As Integer
    Dim iStyle As Integer
    Dim iLayer As Integer
    Dim AvaxHandle As Integer
    Dim i As Integer
    x1 = xStart : y1 = yStart : z1 = 0
    x2 = xStart + 10 : y2 = yStart + 10 : z2 = 0
    For i = 1 To 10
        AvaxHandle = cAvax1.Add_Line(x1, y1, z1, x2, y2, z2, vItemProperties, iClr, iWidth, iStyle,
    iLayer)
        y1 = y1 + 1
        y2 = y2 + 1
    Next i
End Sub
```

Native dimension methods and options

API reference

Each native dimension is one `AvaxItemType.Dimension_i` object with one Integer handle, stored geometry, text placement and style. All methods below are members of `Avax.cAvax`; the receiving control supplies the drawing, so there is no separate Design argument. The existing `Add_Dim` and `Add_AngularDim` keep their original behavior.

Coordinates and lengths use drawing units in the XY plane; creation uses the current pen elevation. Angles use degrees. Add methods return a positive handle or 0 on failure. Invalid input must not be treated as a created object. Use the return values shown below; the examples require only the public DLL API.

Defaults and precedence

Interactive native dimension commands copy the existing AVAX Dim properties. API callers opt into the same snapshot with `Get_DimensionDefaults(kind)`. Omitting `options` in an Add_Dim_* call uses the named/default native style (AVAX_STANDARD), not a snapshot of the legacy Dim properties. The order is named style, non-Nothing StyleOverrides, then a positive textH argument. A textH of 0 inherits the effective style height. OverallScale applies to appearance sizes; MeasurementScale changes the displayed linear value; AngularMeasurementScale changes the displayed angular value before unit conversion. Both accept signed and zero finite factors and leave the geometry unchanged.

Legacy default	Native value
DimTextScale / DimTextFont / DimScale	TextHeight / FontName / OverallScale
DimTextAbove / DimTextHorizontal / DimTextVerticalToLine	Vertical position / inside and outside horizontal text / 90-degree text rotation offset
DimArrowSize / DimArrowType / DimTickSize	Arrow size and text gap / arrow blocks / tick size
DimOutExtLines	ExtensionBeyondDimLine; ExtensionOffset starts at 0. DimInExtLines measures from a different reference and is not copied.
DimDecimal / DimUnitFormat / DimFractionalTick	Linear precision (0-8) / units / fractional precision
DimAngularUnitFormat / DimFactor / AngleFactor	Angular units (including GMS) / MeasurementScale / AngularMeasurementScale
DimDefText or AngleDefText	MeasurementTemplate suffix for the requested kind.

Native formatting follows the CAD unit and precision conventions. A native Fractional value such as `14 1/2` has no automatic inch mark; MeasurementTemplate can add a desired suffix. Native degrees/minutes/seconds use the CAD angular precision convention: precision 2 displays a right angle as `90°00'`. The existing Add_Dim and Add_AngularDim methods retain their R2 text conventions, including the fractional inch mark and the legacy degree/minute/second separators. Get_DimensionDefaults copies the corresponding unit selections and parameters into the native style.

Add_Dim_Linear

Add_Dim_Aligned

Add_Dim_Angular

Add_Dim_Radial

Add_Dim_Diametric

Get_DimensionDefaults

Get_Dimension

Get_DimensionMeasurement

Set_Dimension

Set_DimensionTextPosition

Reset_DimensionTextPosition

Get_DimensionStyle

Set_DimensionStyle

AvaxDimensionOptions

Property / type	Meaning and default
Color As Nullable(Of Integer)	Optional creation palette index; Nothing uses current pen.
Layer As Nullable(Of Integer)	Optional creation layer index; Nothing uses current pen.
LineSpacingFactor As Double	Default 1; valid range 0.25-4.
LineSpacingStyle As AvaxDimLineSpacing	Default AtLeast; Exact is also supported.
LineStyle As Nullable(Of Integer)	Optional creation line-style index; Nothing uses current pen.
LineWidth As Nullable(Of Integer)	Optional nonnegative creation width; Nothing uses current pen.
StyleName As String	Nothing/blank uses AVAX_STANDARD; otherwise the style must be registered.
StyleOverrides As AvaxDimensionStyleOverrides	Nothing inherits the named style. Only supplied fields override it; 0 and False are explicit overrides.
TextAttachment As AvaxDimTextAttachment	Default MiddleCenter (5); attachment values 1-9 are listed below.
TextOverride As String	Nothing/empty uses measurement; <> inserts it in custom text; one space hides text. Supports newlines, \P, %%c, %%d and %%p.
TextRotationOffsetDeg As Nullable(Of Double)	Nothing uses the normal text direction; otherwise an offset in degrees (DXF group 53).

AvaxDimensionData

A detached definition of one editable dimension. New initializes Kind to Linear, ExtensionAngleDeg to 90, other numeric/point values to 0, LeaderLength to Nothing and independent default Style and Options objects. Clone copies every value and detaches Style, Options and nested StyleOverrides. Edit the returned definition, then call Set_Dimension to commit it atomically. Style and Options must both be present when committed.

Property / type	Meaning and default
AngleDeg As Double	Linear measurement-axis or radial/diametric direction, in degrees. Aligned dimensions derive their axis from FirstPoint and SecondPoint.
Center As AvaxDimensionPoint	Angular vertex or circle center.
DefinitionPoint As AvaxDimensionPoint	Dimension-line/arc placement point.
Elevation As Double	Common XY-plane elevation of the definition and all generated geometry. Default: 0. Stored as Double, independently of the XY point values.
ExtensionAngleDeg As Double	Extension-line direction relative to the linear/aligned dimension axis; default 90 degrees. A parallel direction is invalid. Oblique directions change the extension geometry without changing the measured projection/distance.
FirstPoint As AvaxDimensionPoint	First measured point, or point on the first angular ray.
HorizontalDirectionDeg As Double	Horizontal reference direction in degrees, used when horizontal dimension text is enabled. Finite angles may include complete turns; generated text is adjusted for readability. DXF/DWG preserve the reference.
IsTextPositionManual As Boolean	False for automatic placement; True uses TextPosition.
Kind As AvaxDimensionKind	Linear, Aligned, Angular, Radial or Diametric.
LeaderLength As Nullable(Of Double)	Optional radius/diameter leader length in drawing units. A positive value requests automatic outside text with that leader distance, unless ForceTextInside is True. Nothing derives the outside distance from the effective arrow size when fitting requires outside text. Explicit 0 removes that distance and allows inside fitting. A supplied value must be finite and nonnegative; manual text placement takes precedence.

Property / type	Meaning and default
Measurement As Double	Read-only geometry measurement before style display multipliers: Linear projects the measured points onto AngleDeg; Aligned measures their distance; Angular returns the sector selected by DefinitionPoint in degrees; Radial returns Radius and Diametric returns twice Radius.
Options As AvaxDimensionOptions	Per-object text and style overrides.
Radius As Double	Positive circular radius.
SecondPoint As AvaxDimensionPoint	Second measured point, or point on the second angular ray.
Style As AvaxDimensionStyle	Complete named style snapshot.
TextHeight As Double	0 inherits effective style; positive overrides it before OverallScale.
TextPosition As AvaxDimensionPoint	Text position in XY; used with IsTextPositionManual=True.
ResolvedDimensionColor As Integer	Control-maintained document-palette index for dimension lines, arrows and center marks. After color resolution, -1 means ACI 0: inherit the owning dimension's object color. Explicit ACI and ByLayer colors use resolved palette indices.
ResolvedExtensionColor As Integer	Control-maintained document-palette index for extension lines. The reference remains valid even when the dimension kind has no extension lines or those lines are hidden. Default -1.
ResolvedTextColor As Integer	Control-maintained document-palette index for dimension text and its optional frame, including hidden text. Default -1.

The Resolved…Color values are palette indices, not ACI numbers. After control resolution, -1 denotes ACI 0 and the child graphics inherit the owning dimension's object color; ACI 1-255 and ByLayer ACI 256 resolve to document-palette entries. Use the style or StyleOverrides color properties to choose colors. Get_Dimension returns a detached copy; editing its resolved indices does not alter the drawing. Set_Dimension recomputes them from the effective style and the owning layer. Native save/open and block insertion maintain the references when palettes are merged.

`AvaxDimensionPoint` has Double properties X and Y and constructor `New AvaxDimensionPoint(x, y)`. Definitions, options and styles expose `Clone()`; Get methods already return detached copies.

AvaxDimensionStyle

Property / type	Meaning and default
AlternatePrecision As Integer	DIMALTD: precision 0-8. Default: 2.
AlternateScale As Double	DIMALTF: positive alternate-unit multiplier. Default: 25.4.
AlternateTemplate As String	DIMAPOST: alternate text with <> placeholder. Default: "[<>]".
AlternateUnits As Boolean	DIMALT: enable alternate units. Default: False.
AngularMeasurementScale As Double	Finite angular measurement multiplier, applied to degrees before converting units. Signed and zero values are supported. Default: 1. Geometry is unchanged.
AngularPrecision As Integer	DIMADEC: precision 0-8. Default: 2.
AngularUnits As AvaxDimAngularUnits	DIMAUNIT: angular display units. Default: AvaxDimAngularUnits.Degrees.
AngularZeroSuppression As Integer	DIMAZIN bit flags, 0-3. Default: 0.
ArrowBlock1 As String	DIMBLK1: first arrow; empty/_CLOSEDFILLED, _OPEN, _CLOSED, _OBLIQUE, _DOT, _NONE. Default: "".
ArrowBlock2 As String	DIMBLK2: second arrow, same names as ArrowBlock1. Default: "".
ArrowSize As Double	DIMASZ: nonnegative arrow size, multiplied by OverallScale. Also reserves space for arrows during automatic fitting. When TickSize is positive, ArrowSize has no effect on non-angular glyphs, fitting, text placement or default leader distance. Angular dimensions continue to use ArrowSize. Default: 2.5.
CenterMarkSize As Double	DIMCEN: circular center mark size. Default: 0.
DecimalSeparator As String	DIMDSEP: one printable decimal separator for primary, tolerance and alternate-unit text. Default: ".".
DimensionLineColor As Integer	DIMCLRD: dimension lines, arrows and center marks. ACI 0 inherits the owning dimension's object color; 1-255 selects an explicit CAD indexed color; 256 uses its retained layer color. A native layer without an imported CAD color defaults to ACI 7. Default: 0.
DimensionLineWeight As Integer	DIMLWD: dimension-line weight in hundredths of a millimeter: 25 means 0.25 mm. 0 is a visible hairline, -1 is ByLayer, -2 is ByBlock, and -3 uses the control CAD default of 0.25 mm. Explicit native values 0 through 211 are accepted. Default: -2.
DimLineExtension As Double	DIMDLE: dimension-line extension. Default: 0.
DrawDimLineInside As Boolean	DIMTOFL: draw inside line. Default: True.
ExtensionBeyondDimLine As Double	DIMEXE: extension beyond dimension line. Default: 1.25.
ExtensionLineColor As Integer	DIMCLRE: extension lines. ACI 0 inherits the owning dimension's object color; 1-255 selects an explicit CAD indexed color; 256 uses its retained layer color, with ACI 7 as the native-layer default. Default: 0.
ExtensionLineWeight As Integer	DIMLWE: separate extension-line weight, with the same physical units and inheritance values as DimensionLineWeight. Does not replace the dimension-line setting. Default: -2.
ExtensionOffset As Double	DIMEXO: offset from measured point. Default: 0.625.
FitMode As AvaxDimFit	DIMATFIT: controls automatic text and arrow placement when their combined size does not fit the available dimension line or angular arc. Applies to linear, aligned, angular, radial and diametric dimensions. See AvaxDimFit below for each policy. ForceTextInside takes precedence over automatic text placement; an explicit manual text position remains unchanged. Default: AvaxDimFit.BestFit.
FontName As String	Font family used for dimension glyph measurement, drawing and the CAD text resource. Default: "Arial".
ForceTextInside As Boolean	DIMTIX: keep automatically placed text between the measured limits even if it is wider than the available space. Arrows move outside when the combined text and arrows do not fit. This also takes precedence over a positive radius/diameter LeaderLength. It does not replace an explicitly assigned manual text position. Default: False.
GenerateTolerances As Boolean	DIMTOL: enable tolerances. Default: False.
LimitsGeneration As Boolean	DIMLIM: display upper and lower limits in the configured linear or angular unit format. Limits take precedence if GenerateTolerances is also True. Default: False.
LinearPrecision As Integer	DIMDEC: precision 0-8. Default: 2.

Property / type	Meaning and default
LinearUnits As AvaxDimLinearUnits	DIMLUNIT: linear number format. Default: AvaxDimLinearUnits.Decimal.
MeasurementScale As Double	Finite linear measurement multiplier. Signed and zero values are supported. Default: 1. Geometry is unchanged.
MeasurementTemplate As String	DIMPOST: <> placeholder for measured text. Default: "<>".
MinusTolerance As Double	DIMTM: nonnegative lower tolerance. Default: 0.
Name As String	Style-table name. Default: "AVAX_STANDARD".
OverallScale As Double	DIMSCALE: positive appearance scale. Default: 1.
PlusTolerance As Double	DIMTP: nonnegative upper tolerance. Default: 0.
Rounding As Double	DIMRND: nonnegative linear rounding increment; 0 disables. Finite increments below the stored Double resolution leave the value unchanged. Default: 0.
SuppressDimLine1 As Boolean	DIMSD1: hide first dimension-line part. Default: False.
SuppressDimLine2 As Boolean	DIMSD2: hide second dimension-line part. Default: False.
SuppressExtension1 As Boolean	DIMSE1: hide first extension. Default: False.
SuppressExtension2 As Boolean	DIMSE2: hide second extension. Default: False.
SuppressOutsideDimLines As Boolean	DIMSOXD: suppress nonfitting arrowheads when ForceTextInside is True. Has no effect when ForceTextInside is False or when text and arrows fit together. Dimension-line extensions and text leaders are retained. Default: False.
TextColor As Integer	DIMCLRT: dimension text and its optional frame. ACI 0 inherits the owning dimension's object color; 1-255 selects an explicit CAD indexed color; 256 uses its retained layer color, with ACI 7 as the native-layer default. Default: 0.
TextGap As Double	DIMGAP: clearance around text, multiplied by OverallScale. A negative value uses its absolute magnitude for clearance and adds a rectangular frame around the text; 0 removes the clearance. Default: 0.625.
TextHeight As Double	DIMTXT: positive text height before OverallScale. A positive creation textH takes precedence over the named style and StyleOverrides; textH = 0 inherits the effective style height. Default: 2.5.
TextHorizontalPosition As AvaxDimTextHorizontal	DIMJUST: automatic text justification for linear, aligned and angular dimensions. Centered uses the line or selected arc midpoint; the other values select text beside or along either extension line. First/second refer to FirstPoint/SecondPoint, even when the selected angular sector begins at the other ray. Inside fitting can center text when the requested edge clearance does not fit. Over-extension labels do not consume text space between arrowheads. Explicit manual text coordinates take precedence. This setting does not reposition radius/diameter text. Default: AvaxDimTextHorizontal.Centered.
TextInsideHorizontal As Boolean	DIMTIH: True aligns text inside the extension lines to HorizontalDirectionDeg; False follows the dimension direction, adjusted for readability. TextRotationOffsetDeg is applied afterward. Applies to automatic and manual placement. Default: False.
TextMoveMode As AvaxDimTextMove	DIMTMOVE: governs manual text placement. MoveLineWithText moves a linear/aligned dimension line, changes the angular arc radius, or turns a radius/diameter axis toward the text without changing the measurement. AddLeader keeps the line/axis and joins displaced text to it; a connector hidden by the text clearance is omitted. FreeWithoutLeader keeps the line/axis and adds no manual-text leader. For automatically displaced angular text, AddLeader also joins the text to the dimension arc. A connector inside the text clearance is omitted, including an automatic linear leader already covered by the line extension. Reset_DimensionTextPosition restores automatic fitting on the current line/axis. Default: AvaxDimTextMove.FreeWithoutLeader.
TextOutsideHorizontal As Boolean	DIMTOH: True aligns text outside the extension lines to HorizontalDirectionDeg; False follows the dimension direction, adjusted for readability. TextRotationOffsetDeg is applied afterward. Applies to automatic and manual placement. Default: True.
TextStyleName As String	DIMTXSTY: name of the CAD text resource associated with FontName. Export reuses matching resources; if the name already identifies another font, a numeric suffix preserves both fonts. Default: "AVAX_DIM_TEXT".

Property / type	Meaning and default
TextVerticalPosition As AvaxDimTextVertical	DIMTAD: automatic centered, above, outside or below placement, using the scaled text height and absolute TextGap. Outside chooses the side away from the first defining point (the center for a circular dimension). Above centers inside forced-horizontal text on a tilted dimension line. Manual text coordinates take precedence. JIS is retained for CAD exchange; native JIS placement currently uses the above-line layout. Default: AvaxDimTextVertical.Above.
TickSize As Double	DIMTSZ: nonnegative oblique-stroke size for linear/aligned, radius and diameter dimensions. A positive value replaces the selected arrow blocks, including _NONE; OverallScale multiplies its size. Zero uses the selected arrows. Angular dimensions use ArrowBlock1/2 and ArrowSize regardless of TickSize. Default: 0.
TolerancePrecision As Integer	DIMTDEC: precision 0-8 for tolerance numbers and upper/lower limit labels. Default: 2.
ToleranceScale As Double	DIMTFAC: positive tolerance-text scale. Default: 0.7.
ZeroSuppression As Integer	DIMZIN flags 0-15. For engineering/architectural feet and inches, the low two bits select: 0 suppresses zero feet and inches; 1 includes both; 2 includes zero feet; 3 includes zero inches. Decimal flags 4 and 8 suppress leading and trailing zeros; 12 combines both. Default: 0.

Component colors and layers

The three color properties use CAD ACI codes, not indices into the control's customizable palette. ACI 0 follows the owning dimension's object color. ACI 256 follows its layer independently, including when that layer was imported with a true-color RGB value. Imported indexed and true-color layer identities remain distinct even when their RGB values match. A native/R2 layer with no retained CAD appearance uses ACI 7. Changing a dimension's layer recomputes its ByLayer components; changing its object color affects ACI 0 components. The stored style or override remains 0 or 256 rather than being replaced by the resolved palette index.

Layer appearance accompanies the retained drawing resources through native save/open, Undo/Redo and block insertion, including nested resource merges. Conflicting imported layer appearances keep separate layer resources. CAD exchange preserves indexed versus true-color identity where the chosen format supports it; an older format without true-color support uses the nearest ACI representation. Preview resolves component colors in a temporary resource context, leaving the live palette, drawing revision and Undo history unchanged.

Dimension lineweights

Explicit CAD lineweights retain their paper thickness when the drawing is zoomed, scaled or printed. ByLayer uses the owning dimension layer; a layer without an explicit CAD weight uses 0.25 mm. ByBlock inherits the owning object stroke. An object created with the legacy pen APIs keeps that pen convention, including PixelWidth_MM_p; an imported CAD owner inherits its CAD weight. A CAD hairline remains visible and is distinct from a zero legacy stroke, which suppresses the outline.

DimensionLineWeight and ExtensionLineWeight in StyleOverrides are Nullable(Of Integer). Nothing inherits the named style; zero is an explicit hairline. Native storage retains supplied integer codes. DXF/DWG retain standard CAD codes and map a nonstandard explicit value to the nearest supported CAD lineweight, choosing the smaller value on a tie. Native and legacy block storage retain the resolved appearance when dimension graphics are flattened.

AvaxDimensionStyleOverrides

The same style properties except Name are available as optional overrides. Numeric, enum and Boolean properties are nullable; string properties use Nothing for inheritance. New overrides start entirely unspecified. Clear a property to Nothing to inherit again; zero and False do not clear an override.

AvaxDimensionKind

Member	Value
Aligned	1
Angular	5
Diametric	3
Linear	0
Radial	4

AvaxDimTextAttachment

Member	Value
BottomCenter	8
BottomLeft	7
BottomRight	9
MiddleCenter	5
MiddleLeft	4
MiddleRight	6
TopCenter	2
TopLeft	1
TopRight	3

AvaxDimLineSpacing

Member	Value
AtLeast	1
Exact	2

AvaxDimTextMove

Member	Value	Purpose
AddLeader	1	Move text independently and connect it to the fixed dimension line/arc. Applies inside and outside circular dimensions.
FreeWithoutLeader	2	Move text independently without a leader.
MoveLineWithText	0	Move the dimension line/arc or turn the circular dimension axis with the text. The measurement is unchanged. Text at the angular/circular center is invalid because it cannot define a nonzero arc radius or direction.

AvaxDimFit

When the text, its clearance and the arrows fit together, all four policies place them inside. Otherwise the selected policy applies. For angular dimensions, available space is measured along the dimension arc; for a radius or diameter, it is the corresponding radial or diametric span. ForceTextInside overrides automatic text placement. These choices preserve the raw measurement and measured geometry.

Member	Value	Purpose when combined space is insufficient
BothOutside	0	Move both text and arrows outside.
ArrowsFirst	1	Keep arrows inside if they fit by themselves, placing text outside. If the arrows cannot fit, move both outside.
TextFirst	2	Keep text inside if it fits by itself, placing arrows outside. If the text cannot fit, move both outside.
BestFit	3	Keep text inside when it fits by itself. Otherwise keep arrows inside when they fit; if neither fits, move both outside.

Named styles and per-dimension StyleOverrides support the same policies; explicit False overrides an inherited ForceTextInside value. The settings are stored in AVAX, DXF and DWG files.

AvaxDimTextVertical

Member	Value	Purpose
Centered	0	Center the text on the dimension line or arc.
Above	1	Place the glyph box above the line with TextGap clearance. Inside text forced horizontal on a tilted dimension line stays centered on that line.
Outside	2	Choose the side farther from the first defining point; for radius/diameter dimensions use the defining center.
JIS	3	Japanese Industrial Standards code for CAD exchange. Native display currently uses the above-line layout; full JIS-specific placement is pending.
Below	4	Place the glyph box below the line with TextGap clearance.

AvaxDimTextHorizontal

Member	Value	Purpose
Centered	0	Center the text between extension lines, or at the midpoint of the selected angular arc.
FirstExtension	1	Place text beside the first extension line or angular ray. When it fits inside, its glyph edge clears that line by twice the effective arrow size plus TextGap. Fitting can center the text inside or place it outside this extension.
SecondExtension	2	Place text beside the second extension line or angular ray, using the same clearance and fitting rules.
OverFirstExtension	3	Place text beyond and aligned with the first extension endpoint, following the actual signed extension direction and keeping the text readable. Applies to linear/aligned dimensions and angular rays.
OverSecondExtension	4	Place text beyond and aligned with the second extension endpoint. Arrow fitting remains independent of the width of an over-extension label.

AvaxDimLinearUnits

Member	Value
Architectural	4
Decimal	2
Engineering	3
Fractional	5
Scientific	1

AvaxDimAngularUnits

Member	Value
Degrees	0
DegreesMinutesSeconds	1
Gradians	2
Radians	3
Surveyor	4
GradiansMinutesSeconds	5

Measurement formatting

AvaxDimensionStyle.AngularMeasurementScale

`Property AngularMeasurementScale As Double` multiplies the measured angle in degrees before formatting. Default: 1. Set it on a named style with Set_DimensionStyle. Finite negative and zero values are accepted; geometry and Get_DimensionMeasurement remain unchanged.

AvaxDimensionStyleOverrides.AngularMeasurementScale

`Property AngularMeasurementScale As Double?` overrides the named value for one dimension. Nothing inherits the named style; zero is an explicit factor. Get_DimensionDefaults copies AngleFactor_p into this property. Set_Dimension applies edits to a detached definition and regenerates its label.

AvaxDimAngularUnits.GradiansMinutesSeconds

Value 5. Displays gradians with 100 minutes per gradian and 100 seconds per minute. Seconds round to the nearest whole second with carry. Get_DimensionDefaults maps the legacy DimAngularUnitFormat_p value 3 to this mode. Native DMS retains its DXF precision convention: 0 means degrees, 1-2 includes minutes, and 3-8 includes seconds.

All measurement factors reject NaN and infinity. An unrepresentable scaled value fails without replacing the existing dimension. TextHeight, OverallScale and other geometric scale constraints remain unchanged.

AVAX, AVX, BLK and XML retain the editable dimension definition. AVAX format 7 or later is selected when angular scaling, GMS or nonpositive linear factors are present, including unused named styles. DXF/DWG retains standard DIMENSION geometry, a display block and the correctly formatted explicit label. AVAX formatting metadata restores the live factor and original text template on reimport. Other CAD applications display that explicit label; after changing geometry there, they cannot automatically recalculate AVAX-specific formatting. On reimport, AVAX uses the edited CAD geometry and recalculates the original template unless the external application changed the explicit text. Removing the metadata loses the AVAX-specific settings.

VB.NET example

Call with an initialized control on its owning UI thread. The measured angle remains 90 degrees; the label is 150 gradians, zero minutes and zero seconds, followed by the survey suffix.

```
Option Strict On
Imports System
Imports Avax

Public Module Help_NativeMeasurementFormatting
    Public Function Run(cad As cAvax) As Integer
        Dim options = cad.Get_DimensionDefaults(AvaxDimensionKind.Angular)
        options.StyleOverrides.AngularMeasurementScale = 1.5R
        options.StyleOverrides.AngularUnits = AvaxDimAngularUnits.GradiansMinutesSeconds
        options.StyleOverrides.MeasurementTemplate = "<> survey"
        Dim handle As Integer = cad.Add_Dim_Angular(0, 0, 10, 0, 0, 10, 5, 5, 0, options:=options)
        If handle = 0 Then Throw New InvalidOperationException("Dimension creation failed.")
        Return handle
    End Function
End Module
```

Editing, storage and exchange

Selection, move, rotate, mirror, scale, copy, delete, Undo and Redo act on one dimension. Radius, diameter and angular dimensions require uniform scaling; nonuniform scaling or shear is rejected because the defining circle would become an ellipse. Linear and aligned dimensions support nonsingular affine transformations. Reference points are stored in the dimension; there is no automatic DIMASSOC connection to other line/circle handles. Use Set_Dimension or supported grips/stretch to change them. In AVAX-CAD R3 the five native dimension commands appear with the drawing shapes, and selection opens their editable properties.

DXF/DWG export writes DIMENSION entities with DIMSTYLE, overrides and anonymous display blocks. Use DXF R13 or later; 2018 is suitable for drawings containing both native dimensions and native hatches. OnlyLinesACADExportOn exports static graphics instead of DIMENSION. VectorTextExportOn also exports a native dimension as static graphics so its label is outlined; the source dimension remains editable and unchanged. Supported imported dimensions remain editable; unsupported dimension kinds, tilted planes or custom styles/arrows use the available display block. Native .avax reads formats 1-8. Ordinary files use format 5; ribbon widths require format 6; extended dimension formatting requires format 7; retained original CAD polyline paths require format 8. Older readers reject newer formats, and this build rejects formats greater than 8. Use matching runtime files.

BackColor

Purpose

Gets or sets the canvas background color using a Windows Forms Color value. The drawing repaints against the new background without changing entity coordinates, palette colors or the saved changed flag.

Default: `Color.White`.

VB.NET declaration

```
Property BackColor As System.Drawing.Color
```

Example

```
Option Strict On
Imports Avax
Imports System.Drawing

Public Module Help_Display_BackColor
    Public Sub Run(cad As cAvax)
        cad.BackColor = Color.Black
    End Sub
End Module
```

Related options and reference

BorderStyle

Purpose

Gets or sets the border using the legacy Integer interface: 0 = no border, 1 = fixed single border, 2 = fixed 3D border. The corresponding Windows Forms border is updated immediately.

Default: `0`.

VB.NET declaration

```
Property BorderStyle As Integer
```

Example

```
Option Strict On
Imports Avax
Imports System.Drawing

Public Module Help_Display_BorderStyle
    Public Sub Run(cad As cAvax)
        cad.BorderStyle = 1
    End Sub
End Module
```

Related options and reference

Command

cAvax property

Purpose

Gets or starts a built-in drawing, selection, editing or view command. Use an AvaxCommand value to choose the operation.

See AvaxCommand for the available commands and their values.

See legacy cursor and refresh commands for the mAvax values that preserve the current drawing command.

See Line, Point and Rectangle commands for connected XYZ lines, the planar Point tool versus Add_Point, rectangular closed polylines, completion and host callbacks.

See Circle and Ellipse commands for center/radius, diameter and three-point circles, ellipse radii, Double precision and one-shot command completion.

See Arc commands and segmentation for the three arc pick orders, standalone completion, polyline continuation and arc resolution settings.

See Polyline, Polygon and Close commands for pending XYZ chains, exact closing edges, regular polygons, vertex counts, completion and complete VB.NET examples.

See Immediate actions and selection borders for direct mode changes, filtered selection, one overall Border, outputs, callbacks and accepted Polyline input.

VB.NET declaration

```
Property Command As AvaxCommand
```

Property value

Type	Access	Meaning
AvaxCommand	Read/write	Is the ready made AVAX command

Remarks

Assign an AvaxCommand value to start a drawing, selection, editing or view command. The getter returns the current command. See AvaxCommand for the complete list of command names and values. Supply construction points with the mouse or SetCoordinate; use Cancel_c to abandon unfinished input. Grid and orthogonal-mode switches preserve the drawing command in progress. Change_Layer_c, Change_Color_c, Change_Width_c, Change_Ltype_c, Change_Ptype_c, Change_z_c, Change_h_c and Change_Font_c open a parameter dialog. With selected objects, they change only visible, unlocked objects of an applicable type; without a selection, they set the corresponding pen default for newly created objects. An inapplicable selection leaves both the drawing and pen unchanged. Layer and line-style numbers range from 0 to 255; numeric dialog values use a decimal point. Change_Text_c and Calc_Text_c open the text editor for selected text and document objects. The accepted text is applied to all eligible targets while retaining each target's bold, italic, underline, strikeout and character set. Change_Font_c changes their font. Each accepted multi-object change is one undoable operation. Cancel leaves the objects unchanged. Text creation and these attribute-edit commands are ignored while the control is stopped, disabled, paused or edit-locked, preserving unfinished construction. If a host event handler changes the drawing, selection, command or editing state while a dialog is open, its stale answer is discarded. Offset_c retains the endpoint widths of open polyline segments at new miter joins. A segment with no XY direction cannot be offset in the drawing plane and is rejected without modifying the drawing. Explode_c keeps nonzero widths as individual one-segment polylines. Pedit_c joins source segments in their existing order and retains each occurrence's endpoint widths, including coincident or reversed segments. Exact zero-length edges are omitted without shifting the remaining width profiles. Taxis_c keeps each segment's full width vectors when ordering or reversing it. Coincident or contained centrelines with different width profiles are retained. P_Break_c splits each source occurrence independently, interpolating its own endpoint widths at each cut. File and parameter dialogs belong to the drawing and command that opened them. If the host replaces the drawing, starts another command or ends the control while a dialog is open, its obsolete answer is discarded. Dxf_Out_c and Dwg_Out_c require a selection and wait for a picked XYZ base point before writing the file; use SetCoordinate to supply the point programmatically. Cancel preserves the destination. A write failure retains the export request for retry; editing the drawing or changing the selection before choosing the base point invalidates that request. Insert_c, Insert_Avx_c, Dxf_In_c and Dwg_In_c apply the full XYZ coordinate supplied through SetCoordinate. Scale and angle affect X/Y; the insertion Z is added to source elevations. Invalid preparation leaves no pending insertion, while an invalid placement point can be corrected without reloading the source. The legacy mAvax.Refresh_m (202) and mAvax.HideCursor_m (204) values refresh/clear transient presentation while preserving the current command and accepted points. Assign them with CType(value, AvaxCommand). Selection add/subtract state is independent of the current tool. SelectSub_c enables subtraction and SelectAdd_c restores addition. Ordinary Select_c, Cancel, drawing/view commands and selection windows retain this state. SelectAddOn is 0 for subtract and 1 for add; both point clicks and windows follow the same value. A new drawing resets it to add. Trim and Extend force add when starting target selection from valid selected boundaries. Border_c creates one combined, closed bounds frame for the selected objects, preserving the source objects and selection. With no selection, enter points and press Enter to complete the frame. See Border_c in AvaxCommand for the direct-action notification contract.

Example

Setup

Add an Avax.cAvax Windows Forms control named `cAvax1`. Paste the procedures into the form class. Example setup and supporting files explains references, initialization and file paths.

Add these Windows Forms controls or components to the form. Fields referenced by Handles clauses must be declared WithEvents, as the Visual Studio designer normally does.

Name	Type
Combo1	ComboBox

VB.NET example

```
Private Sub Combo1_SelectedIndexChanged(ByVal eventSender As System.Object, ByVal eventArgs As
    System.EventArgs) Handles Combo1.SelectedIndexChanged
    Select Case Combo1.Text
        Case "Cancel"
        CAvax1.Command = Avax.AvaxCommand.Cancel_c
        Case "Zoom"
        CAvax1.Command = Avax.AvaxCommand.Zoom_c
        Case "Previous"
        CAvax1.Command = Avax.AvaxCommand.Previous_c
        Case "Line"
        CAvax1.Command = Avax.AvaxCommand.Line_c
        Case "Circle"
        CAvax1.Command = Avax.AvaxCommand.Circle_c
```

```
        Case "Ellipse"
        CAvax1.Command = Avax.AvaxCommand.Ellipse_c
    End Select
End Sub
Private Sub Form1_Load(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs) Handles
    MyBase.Load
    cAvax1.StartAvax()
    Combo1.Items.Add("Cancel")
    Combo1.Items.Add("Zoom")
    Combo1.Items.Add("Previous")
    Combo1.Items.Add("Line")
    Combo1.Items.Add("Circle")
    Combo1.Items.Add("Ellipse")
    Combo1.Text = ""
End Sub
Private Sub Form1_FormClosed(ByVal eventSender As System.Object, ByVal eventArgs As
    System.Windows.Forms.FormClosedEventArgs) Handles Me.FormClosed
    cAvax1.EndAvax()
End Sub
```

CommandFilePicker

API reference

Purpose and default

Lets the application supply the file path when a native command requests an Open or Save file dialog. The callback defaults to `Nothing`, which retains the built-in file dialog and its `DialogEnabled_p` policy.

VB.NET declaration

```
Public Property CommandFilePicker As Func(Of Boolean, String, String, String)
```

Callback arguments and result

Value	Meaning
Boolean argument	`True` requests a Save destination; `False` requests a file to open or insert.
First String argument	Localized dialog title, such as the title for saving a drawing or inserting a block.
Second String argument	Windows Forms file filter, using description and pattern pairs separated by `|`. A pattern can contain several extensions separated by `;`.
Return value	The selected file path. Return `Nothing` or an empty string to cancel this file request.

Validation and behavior

The callback runs synchronously during the command. Invoke commands on the control's owning UI thread and return after the application's file selection is complete. A supplied callback takes precedence over the built-in dialog, including when `DialogEnabled_p` is `False`. Cancellation does not open a fallback dialog.

The returned path goes to the command's existing file operation. The host picker supplies its own extension selection, overwrite confirmation and other dialog features. Use an extension appropriate for that command; for example, `Save_As_c` saves the native drawing and should receive an `.avax` path. The command's output permissions, input validation and document ownership checks still apply. A result belonging to an earlier drawing or command is discarded if the callback replaces the drawing, changes the active command or ends the control.

File prompts using this callback include opening and saving drawings, inserting blocks and CAD drawings, creating blocks, exporting DXF, DWG and legacy plot files, selecting a block label, and saving an icon. A command may require a selection or further coordinate input before completing its operation. Methods accepting an explicit filename use their own filename parameter. For example, call `PrintToSVG` or `PrintWorkspaceToSVG` with the chosen SVG path.

Lifetime

This is a per-control application callback. It is not stored in drawing files or settings XML. Restore a previous callback, or assign `Nothing`, when the override is no longer needed. The property has no success return value; file commands keep their normal completion behavior. Subscribe to OperationFailed to receive errors reported by native operations.

Example

Pass a started, enabled control on its owning UI thread and a writable full `.avax` path chosen by the host. This temporary picker supplies the Save As destination and restores the previous callback afterward.

```
Option Strict On
Imports System
Imports Avax

Public Module Help_CommandFilePicker
    Public Sub Run(cad As cAvax, drawingFile As String)
        Dim previous = cad.CommandFilePicker
        Try
            cad.CommandFilePicker = Function(save, title, filter)
                                        If save Then Return drawingFile
                                        Return Nothing
                                    End Function
            cad.Command = AvaxCommand.Save_As_c
        Finally
            cad.CommandFilePicker = previous
```

```
        End Try
    End Sub
End Module
```

Related members

Command, AvaxCommand, ParameterDialogInput, OperationFailed.

CompoundPathClosed

Purpose

Controls whether the current compound-path figure closes back to its starting node.

Declaration

```
Property CompoundPathClosed As Boolean
```

Property value

True closes the figure; False leaves it open. BeginCompoundPath sets the value from its optional closed argument, whose default is False. Assigning this property also updates the figure already being entered. Closure supplies a connection from the final endpoint to the start; the application need not repeat the first coordinate. The property does not change a previously committed object; edit that object's AvaxPathFigure.Closed through GetCompoundPath and ChangeCompoundPath instead.

VB.NET example

```
Option Strict On
Public Module CompoundPathClosedExample
    Public Function DrawClosedOutline(cad As Avax.cAvax) As Integer
        If Not cad.BeginCompoundPath() Then Return 0
        cad.SetCoordinate(0.0, 0.0, 0.0)
        cad.SetCoordinate(20.0, 0.0, 0.0)
        cad.SetCoordinate(20.0, 10.0, 0.0)
        cad.SetCoordinate(0.0, 10.0, 0.0)
        cad.CompoundPathClosed = True
        Return cad.FinishCompoundPath()
    End Function
End Module
```

Related options and reference

CompoundPathSegmentKind

Purpose

Chooses the kind of the next segment entered in the current compound-path session.

Declaration

```
Property CompoundPathSegmentKind As AvaxPathSegmentKind
```

Property value

Value	Required points after the previous endpoint
Line = 0	Endpoint.
Arc = 1	Intermediate point, then endpoint.
Bezier = 2	First control, second control, then endpoint.

BeginCompoundPath resets this value to Line. Assigning the property, even to its current value, discards any pending controls of an incomplete segment and retains all completed segments. It does not convert a completed segment; use ChangePathSegmentKind for that operation. An undefined enumeration value throws ArgumentOutOfRangeException.

VB.NET example

```
Option Strict On
Public Module CompoundPathSegmentKindExample
    Public Function DrawCubic(cad As Avax.cAvax) As Integer
        If Not cad.BeginCompoundPath() Then Return 0
        cad.SetCoordinate(0.0, 0.0, 0.0)
        cad.CompoundPathSegmentKind = Avax.AvaxPathSegmentKind.Bezier
        cad.SetCoordinate(10.0, 20.0, 0.0)
        cad.SetCoordinate(20.0, 20.0, 0.0)
        cad.SetCoordinate(30.0, 0.0, 0.0)
        Return cad.FinishCompoundPath()
    End Function
End Module
```

Related options and reference

ContinueUndo

cAvax property

Purpose

When this property is True then the program adds to the undo list all subsequent commands. Upon the execution of the last command you must set this property to False. If the AvaxCommand.Undo_c is executed with the use of property Command then all the previous commands in the undo list are undone

VB.NET declaration

```
Property ContinueUndo As Boolean
```

Property value

Type	Access	Meaning
Boolean	Read/write	A Boolean expression that defines if the ContinueUndo is True or False

Example

Setup

Add an Avax.cAvax Windows Forms control named `cAvax1`. Paste the procedures into the form class. Example setup and supporting files explains references, initialization and file paths.

Add these Windows Forms controls or components to the form. Fields referenced by Handles clauses must be declared WithEvents, as the Visual Studio designer normally does.

Name	Type
Command1	Button
Command2	Button
Command3	Button

VB.NET example

```
Private Sub Form1_Load(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs) Handles
    MyBase.Load
    Command1.Text = "Undo"
    Command2.Text = "Add 3 Lines (3 Undos)"
    Command3.Text = "Add 3 Lines (1 Undo)"
    cAvax1.StartAvax()
End Sub
Private Sub Form1_FormClosed(ByVal eventSender As System.Object, ByVal eventArgs As
    System.Windows.Forms.FormClosedEventArgs) Handles Me.FormClosed
    cAvax1.EndAvax()
End Sub
Private Sub Command1_Click(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs)
    Handles Command1.Click
    CAvax1.Command = Avax.AvaxCommand.Undo_c
End Sub
Private Sub Command2_Click(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs)
    Handles Command2.Click
    'Add 3 Lines (3 Undos)
    Call cAvax1.Add_Line(0, 0, 0, 10, 10, 0)
    Call cAvax1.Add_Line(0, 10, 0, 10, 10, 0)
    Call cAvax1.Add_Line(10, 0, 0, 10, 10, 0)
End Sub
Private Sub Command3_Click(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs)
    Handles Command3.Click
    'Add 3 Lines (1 Undo)
```

```
    Call CAvax1.Add_Line(2, 0, 0, 2, 10, 0, Nothing, 5, -1, -1, -1)
    CAvax1.ContinueUndo = True
    Call CAvax1.Add_Line(2, 10, 0, 0, 2, 0, Nothing, 5, -1, -1, -1)
    Call CAvax1.Add_Line(10, 2, 0, 2, 2, 0, Nothing, 5, -1, -1, -1)
    CAvax1.ContinueUndo = False
End Sub
```

DefaultExtension

API reference

Purpose

Chooses the format for the next new drawing and for extensionless names passed to StartAvax. This property was added in v3. Set it to avx when an application needs new files in the legacy R2 format.

Declaration

```
Public Property DefaultExtension As String
```

Value

The default is avax, the native v3 drawing format. The other supported value is avx, the legacy drawing format. Values are trimmed and normalized to lower case; an optional leading dot is accepted. An unsupported value, an empty string or Nothing raises ArgumentException and retains the preceding value.

Behavior

Set this property before creating the next drawing. Changing it preserves the current drawing's format. Opening an existing file or successfully saving with an explicit extension selects that drawing's format. Ordinary Save and extensionless SaveAs use the current format. Save As initializes its extension and filter from that format; cancellation and failure retain it.

VB.NET example

```
Option Strict On
Imports Avax

Public Module DefaultExtensionExample
    Public Function CreateLegacyDrawing(cad As cAvax, baseName As String) As Boolean
        cad.DefaultExtension = "avx"
        Return cad.StartAvax(baseName)
    End Function
End Module
```

DrawingUnitMM

Purpose and value

Finite positive physical millimeters per coordinate unit; default 1000 (meters).

VB.NET declaration

```
Property DrawingUnitMM As Double
```

Example

Call this procedure on the owning UI thread with a control already initialized by `StartAvax()`. Handles must belong to that control. File paths are supplied by the caller.

```
Option Strict On
Imports System
Imports Avax

Public Module Help_DrawingUnitMM
    Public Sub Run(cad As cAvax)
        cad.DrawingUnitMM = 1000
    End Sub
End Module
```

Related options and reference

Enabled

Purpose

Gets or sets whether the control accepts normal user interaction. Set False to disable its Windows Forms input and True to restore it. This does not discard the current drawing. IsChanged remains accessible for a started disabled control.

Default: `True`.

VB.NET declaration

```
Property Enabled As Boolean
```

Example

```
Option Strict On
Imports Avax
Imports System.Drawing

Public Module Help_Display_Enabled
    Public Sub Run(cad As cAvax)
        cad.Enabled = True
    End Sub
End Module
```

Related options and reference

filename

cAvax property

Purpose

Is the file name of the drawing. The control goes into effect when a file name is given or with the command StartAvax

VB.NET declaration

```
Property filename As String
```

Property value

Type	Access	Meaning
String	Read/write	A String expression that defines the File name

Example

Setup

Add an Avax.cAvax Windows Forms control named `cAvax1`. Paste the procedures into the form class. Example setup and supporting files explains references, initialization and file paths.

Add these Windows Forms controls or components to the form. Fields referenced by Handles clauses must be declared WithEvents, as the Visual Studio designer normally does.

Name	Type
Command1	Button

VB.NET example

```
Private Sub Form1_Load(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs) Handles
    MyBase.Load
    Command1.Text = "Set AVAX Filename"
    cAvax1.StartAvax()
End Sub
Private Sub Form1_FormClosed(ByVal eventSender As System.Object, ByVal eventArgs As
    System.Windows.Forms.FormClosedEventArgs) Handles Me.FormClosed
    cAvax1.EndAvax()
End Sub
Private Sub Command1_Click(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs)
    Handles Command1.Click
    cAvax1.FileName = My.Application.Info.DirectoryPath & "\Sample.AVX"
    Me.Text = cAvax1.FileName
End Sub
```

FontBold

cAvax property

Purpose

Sets or returns the current Font Bold status

VB.NET declaration

```
Property FontBold As Boolean
```

Property value

Type	Access	Meaning
Boolean	Read/write	A Boolean expression that if it is true then the fonts are bold.

Example

Setup

Add an Avax.cAvax Windows Forms control named `cAvax1`. Paste the procedures into the form class. Example setup and supporting files explains references, initialization and file paths.

Add these Windows Forms controls or components to the form. Fields referenced by Handles clauses must be declared WithEvents, as the Visual Studio designer normally does.

Name	Type
Command1	Button
Check1	CheckBox

VB.NET example

```
Private Sub Check1_CheckStateChanged(ByVal eventSender As System.Object, ByVal eventArgs As
    System.EventArgs) Handles Check1.CheckStateChanged
    If Check1.CheckState = 1 Then
        CAvax1.FontBold = True
    Else
        CAvax1.FontBold = False
    End If
End Sub
Private Sub Form1_Load(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs) Handles
    MyBase.Load
    CAvax1.StartAvax()
    Call CAvax1.SetAvaxProperty(Avax.AvaxProp.TextScale n, 5.0!)
End Sub
Private Sub Form1_FormClosed(ByVal eventSender As System.Object, ByVal eventArgs As
    System.Windows.Forms.FormClosedEventArgs) Handles Me.FormClosed
    cAvax1.EndAvax()
End Sub
Private Sub Command1_Click(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs)
    Handles Command1.Click
    CAvax1.Command = Avax.AvaxCommand.Text c
End Sub
```

FontCharset

cAvax property

Purpose

Sets or returns the current Font Charset value

VB.NET declaration

```
Property FontCharset As Integer
```

Property value

Type	Access	Meaning
Integer	Read/write	A numeric expression that identifies the Font Character Set.

Example

Setup

Add an Avax.cAvax Windows Forms control named `cAvax1`. Paste the procedures into the form class. Example setup and supporting files explains references, initialization and file paths.

Add these Windows Forms controls or components to the form. Fields referenced by Handles clauses must be declared WithEvents, as the Visual Studio designer normally does.

Name	Type
Combo1	ComboBox
Combo2	ComboBox
Label1	Label
Label2	Label

VB.NET example

```
Private Sub Form1_Load(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs) Handles
    MyBase.Load
    Label1.Text = "Default Font Name="
    Label2.Text = "Default Font Character Set="
    cAvax1.StartAvax()
    Call CAvax1.Add_Text("AVAX ActiveX Control", 0, 5, 0, Nothing, 10.0!, 1, 0, -1, -1, -1, -1, False,
    False, False, False, -1)
    Call GetScreenFonts()
End Sub
Private Sub Form1_FormClosed(ByVal eventSender As System.Object, ByVal eventArgs As
    System.Windows.Forms.FormClosedEventArgs) Handles Me.FormClosed
    cAvax1.EndAvax()
End Sub
Private Sub Combo1_SelectedIndexChanged(ByVal eventSender As System.Object, ByVal eventArgs As
    System.EventArgs) Handles Combo1.SelectedIndexChanged
    CAvax1.FontName = Combo1.Text
End Sub
Private Sub Combo2_SelectedIndexChanged(ByVal eventSender As System.Object, ByVal eventArgs As
    System.EventArgs) Handles Combo2.SelectedIndexChanged
    CAvax1.FontCharset = Val(Combo2.Text)
End Sub
Public Sub GetScreenFonts()
    Dim lFont As Integer
    Dim lMax As Integer
    Dim i As Integer
```

```
    lMax = System.Drawing.FontFamily.Families.Length
    For lFont = 0 To lMax - 1
        Combo1.Items.Add(System.Drawing.FontFamily.Families(lFont).Name)
        If lFont = 0 Then
            Combo1.Text = System.Drawing.FontFamily.Families(lFont).Name
            CAvax1.FontName = Combo1.Text
        End If
    Next lFont
    For i = 0 To 255
        Combo2.Items.Add(Format(i))
    Next i
    Combo2.Text = "0"
    cAvax1.FontCharset = 0
End Sub
```

FontItalic

cAvax property

Purpose

Sets or returns the current Font Italic status

VB.NET declaration

```
Property FontItalic As Boolean
```

Property value

Type	Access	Meaning
Boolean	Read/write	A Boolean expression that if it is true then the fonts are italic.

Example

Setup

Add an Avax.cAvax Windows Forms control named `cAvax1`. Paste the procedures into the form class. Example setup and supporting files explains references, initialization and file paths.

Add these Windows Forms controls or components to the form. Fields referenced by Handles clauses must be declared WithEvents, as the Visual Studio designer normally does.

Name	Type
Command1	Button
Check1	CheckBox

VB.NET example

```
Private Sub Check1_CheckStateChanged(ByVal eventSender As System.Object, ByVal eventArgs As
    System.EventArgs) Handles Check1.CheckStateChanged
    If Check1.CheckState = 1 Then
        CAvax1.FontItalic = True
    Else
        CAvax1.FontItalic = False
    End If
End Sub
Private Sub Form1_Load(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs) Handles
    MyBase.Load
    cAvax1.StartAvax()
    Call CAvax1.SetAvaxProperty(Avax.AvaxProp.TextScale_n, 5.0!)
End Sub
Private Sub Form1_FormClosed(ByVal eventSender As System.Object, ByVal eventArgs As
    System.Windows.Forms.FormClosedEventArgs) Handles Me.FormClosed
    cAvax1.EndAvax()
End Sub
Private Sub Command1_Click(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs)
    Handles Command1.Click
    CAvax1.Command = Avax.AvaxCommand.Text_c
End Sub
```

FontName

cAvax property

Purpose

This property contains the default font that the AVAX uses when a Font Table is not set

VB.NET declaration

```
Property FontName As String
```

Property value

Type	Access	Meaning
String	Read/write	A String expression that defines the font name

Example

Setup

Add an Avax.cAvax Windows Forms control named `cAvax1`. Paste the procedures into the form class. Example setup and supporting files explains references, initialization and file paths.

Add these Windows Forms controls or components to the form. Fields referenced by Handles clauses must be declared WithEvents, as the Visual Studio designer normally does.

Name	Type
Combo1	ComboBox
Combo2	ComboBox
Label1	Label
Label2	Label

VB.NET example

```
Private Sub Form1_Load(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs) Handles
    MyBase.Load
    Label1.Text = "Default Font Name="
    Label2.Text = "Default Font Character Set="
    cAvax1.StartAvax()
    Call CAvax1.Add_Text("AVAX ActiveX Control", 0, 5, 0, Nothing, 10.0!, 1, 0, -1, -1, -1, -1, False,
    False, False, False, -1)
    Call GetScreenFonts()
End Sub
Private Sub Form1_FormClosed(ByVal eventSender As System.Object, ByVal eventArgs As
    System.Windows.Forms.FormClosedEventArgs) Handles Me.FormClosed
    cAvax1.EndAvax()
End Sub
Private Sub Combo1_SelectedIndexChanged(ByVal eventSender As System.Object, ByVal eventArgs As
    System.EventArgs) Handles Combo1.SelectedIndexChanged
    CAvax1.FontName = Combo1.Text
End Sub
Private Sub Combo2_SelectedIndexChanged(ByVal eventSender As System.Object, ByVal eventArgs As
    System.EventArgs) Handles Combo2.SelectedIndexChanged
    CAvax1.FontCharset = Val(Combo2.Text)
End Sub
Public Sub GetScreenFonts()
    Dim lFont As Integer
    Dim lMax As Integer
    Dim i As Integer
```

```
    lMax = System.Drawing.FontFamily.Families.Length
    For lFont = 0 To lMax - 1
        Combo1.Items.Add(System.Drawing.FontFamily.Families(lFont).Name)
        If lFont = 0 Then
            Combo1.Text = System.Drawing.FontFamily.Families(lFont).Name
            CAvax1.FontName = Combo1.Text
        End If
    Next lFont
    For i = 0 To 255
        Combo2.Items.Add(Format(i))
    Next i
    Combo2.Text = "0"
    cAvax1.FontCharset = 0
End Sub
```

FontSize

cAvax property

Purpose

Gets or sets the fallback Windows font size in points. Text entities also have their own drawing-space height; this property is not a replacement for the height argument of Add_Text.

VB.NET declaration

```
Property FontSize As Integer
```

Property value

Type	Access	Meaning
Integer	Read/write	Gets or sets the fallback Windows font size in points. Text entities also have their own drawing-space height; this property is not a replacement for the height argument of Add_Text.

Remarks

Nonpositive values use 8 points. Values that cannot round-trip through the underlying Windows font size are rejected.

Example

Setup

Add an Avax.cAvax Windows Forms control named `cAvax1`. Paste the procedures into the form class. Example setup and supporting files explains references, initialization and file paths.

Add these Windows Forms controls or components to the form. Fields referenced by Handles clauses must be declared WithEvents, as the Visual Studio designer normally does.

Name	Type
Command1	Button

VB.NET example

```
Private Sub Form1_Load(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs) Handles
    MyBase.Load
    Command1.Text = "Get FontSize"
    cAvax1.StartAvax()
End Sub
Private Sub Form1_FormClosed(ByVal eventSender As System.Object, ByVal eventArgs As
    System.Windows.Forms.FormClosedEventArgs) Handles Me.FormClosed
    cAvax1.EndAvax()
End Sub
Private Sub Command1_Click(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs)
    Handles Command1.Click
    MsgBox(cAvax1.Font.SizeInPoints, MsgBoxStyle.Information, "FontSize")
End Sub
```

FontStrikethrough

cAvax property

Purpose

Sets or returns the current Font Strikethrough status

VB.NET declaration

```
Property FontStrikethrough As Boolean
```

Property value

Type	Access	Meaning
Boolean	Read/write	A Boolean expression that if it is true then the Font is strikethrough

Example

Setup

Add an Avax.cAvax Windows Forms control named `cAvax1`. Paste the procedures into the form class. Example setup and supporting files explains references, initialization and file paths.

Add these Windows Forms controls or components to the form. Fields referenced by Handles clauses must be declared WithEvents, as the Visual Studio designer normally does.

Name	Type
Command1	Button
Check1	CheckBox

VB.NET example

```
Private Sub Check1_CheckStateChanged(ByVal eventSender As System.Object, ByVal eventArgs As
    System.EventArgs) Handles Check1.CheckStateChanged
    If Check1.CheckState = 1 Then
        CAvax1.FontStrikethrough = True
    Else
        CAvax1.FontStrikethrough = False
    End If
End Sub
Private Sub FrmFontStrikeThrough_Load(ByVal eventSender As System.Object, ByVal eventArgs As
    System.EventArgs) Handles MyBase.Load
    cAvax1.StartAvax()
    Call CAvax1.SetAvaxProperty(Avax.AvaxProp.TextScale_n, 5.0!)
End Sub
Private Sub FrmFontStrikeThrough_FormClosed(ByVal eventSender As System.Object, ByVal eventArgs As
    System.Windows.Forms.FormClosedEventArgs) Handles Me.FormClosed
    cAvax1.EndAvax()
End Sub
Private Sub Command1_Click(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs)
    Handles Command1.Click
    CAvax1.Command = Avax.AvaxCommand.Text_c
End Sub
```

FontUnderline

cAvax property

Purpose

Sets or returns the current Font Underline status

VB.NET declaration

```
Property FontUnderline As Boolean
```

Property value

Type	Access	Meaning
Boolean	Read/write	A Boolean expression that if it is true then the fonts are underline

Example

Setup

Add an Avax.cAvax Windows Forms control named `cAvax1`. Paste the procedures into the form class. Example setup and supporting files explains references, initialization and file paths.

Add these Windows Forms controls or components to the form. Fields referenced by Handles clauses must be declared WithEvents, as the Visual Studio designer normally does.

Name	Type
Command1	Button
Check1	CheckBox

VB.NET example

```
Private Sub Check1_CheckStateChanged(ByVal eventSender As System.Object, ByVal eventArgs As
    System.EventArgs) Handles Check1.CheckStateChanged
    If Check1.CheckState = 1 Then
        CAvax1.FontUnderline = True
    Else
        CAvax1.FontUnderline = False
    End If
End Sub
Private Sub Form1_Load(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs) Handles
    MyBase.Load
    cAvax1.StartAvax()
    Call CAvax1.SetAvaxProperty(Avax.AvaxProp.TextScale n, 5.0!)
End Sub
Private Sub Form1_FormClosed(ByVal eventSender As System.Object, ByVal eventArgs As
    System.Windows.Forms.FormClosedEventArgs) Handles Me.FormClosed
    cAvax1.EndAvax()
End Sub
Private Sub Command1_Click(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs)
    Handles Command1.Click
    CAvax1.Command = Avax.AvaxCommand.Text c
End Sub
```

ForeColor

Purpose

Gets or sets the Windows Forms foreground Color value of the control. Drawing entity colors use the palette indices supplied to Add_* methods or the current PenColor. Assigning this property preserves the existing drawing.

Default: `Color.Black`.

VB.NET declaration

```
Property ForeColor As System.Drawing.Color
```

Example

```
Option Strict On
Imports Avax
Imports System.Drawing

Public Module Help_Display_ForeColor
    Public Sub Run(cad As cAvax)
        cad.ForeColor = Color.Gold
    End Sub
End Module
```

Related options and reference

HasWorkspaceSettings

Purpose and value

Read-only Boolean: drawing contains explicit stored workspace settings. An unchanged default assignment need not create settings.

VB.NET declaration

```
ReadOnly Property HasWorkspaceSettings As Boolean
```

Example

Call this procedure on the owning UI thread with a control already initialized by `StartAvax()`. Handles must belong to that control. File paths are supplied by the caller.

```
Option Strict On
Imports System
Imports Avax

Public Module Help_HasWorkspaceSettings
    Public Sub Run(cad As cAvax)
        Dim value As Boolean = cad.HasWorkspaceSettings
    End Sub
End Module
```

Invalid workspace property values throw an argument exception before changing the drawing. Related settings must leave a nonempty, representable printable area. Successful edits participate in Undo/Redo; changes are local to this drawing.

Related options and reference

hdc

cAvax property

Purpose

Returns the hDC of the AVAX control

VB.NET declaration

```
ReadOnly Property hdc As Integer
```

Property value

Type	Access	Meaning
Integer	Read-only	A numeric expression that identifies the Hdc of the AVAX window.

Example

Setup

Add an Avax.cAvax Windows Forms control named `cAvax1`. Paste the procedures into the form class. Example setup and supporting files explains references, initialization and file paths.

Add these Windows Forms controls or components to the form. Fields referenced by Handles clauses must be declared WithEvents, as the Visual Studio designer normally does.

Name	Type
Command1	Button

VB.NET example

```
Private Sub Form1_Load(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs) Handles
    MyBase.Load
    cAvax1.StartAvax()
End Sub
Private Sub Form1_FormClosed(ByVal eventSender As System.Object, ByVal eventArgs As
    System.Windows.Forms.FormClosedEventArgs) Handles Me.FormClosed
    cAvax1.EndAvax()
End Sub
Private Sub Command1_Click(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs)
    Handles Command1.Click
    MsgBox("hDC = " & Format(CAvax1.hdc), MsgBoxStyle.Information, "AVAX")
End Sub
```

hwnd

cAvax property

Purpose

Returns the hWnd of the AVAX window

VB.NET declaration

```
ReadOnly Property hwnd As Integer
```

Property value

Type	Access	Meaning
Integer	Read-only	A numeric expression that identifies the hWnd of the AVAX window

Example

Setup

Add an Avax.cAvax Windows Forms control named `cAvax1`. Paste the procedures into the form class. Example setup and supporting files explains references, initialization and file paths.

Add these Windows Forms controls or components to the form. Fields referenced by Handles clauses must be declared WithEvents, as the Visual Studio designer normally does.

Name	Type
Command1	Button

VB.NET example

```
Private Sub Form1_Load(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs) Handles
    MyBase.Load
    cAvax1.StartAvax()
End Sub
Private Sub Form1_FormClosed(ByVal eventSender As System.Object, ByVal eventArgs As
    System.Windows.Forms.FormClosedEventArgs) Handles Me.FormClosed
    cAvax1.EndAvax()
End Sub
Private Sub Command1_Click(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs)
    Handles Command1.Click
    MsgBox("hWnd = " & Format(CAvax1.hwnd), MsgBoxStyle.Information, "AVAX")
End Sub
```

IsChanged

cAvax property

Purpose

Informs that the drawing has changed since it has been opened

VB.NET declaration

```
Property IsChanged As Boolean
```

Property value

Type	Access	Meaning
Boolean	Read/write	A Boolean expression that if it is True then the drawing has been changed since it has been opened and if it is False then the drawing has not been changed since it has been opened.

Remarks

Reports whether the current started drawing contains unsaved changes. Geometry edits set this flag; a successful native save or reopen clears it. Assign False after your own successful save operation, or True when your application needs to mark the drawing modified. The property does not save the drawing or discard undo history. Before StartAvax and after EndAvax the getter returns False and assignments have no effect. Enabled = False does not disable this property for a started control. In the example, the Yes and No branches only permit closing: add your save code at the indicated comment.

Example

Setup

Add an Avax.cAvax Windows Forms control named `cAvax1`. Paste the procedures into the form class. Example setup and supporting files explains references, initialization and file paths.

VB.NET example

```
Private Sub Form1_Load(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs) Handles
    MyBase.Load
    cAvax1.StartAvax()
    Call cAvax1.Add_Line(0, 0, 0, 10, 10, 0)
End Sub
Private Sub Form1_FormClosing(ByVal eventSender As System.Object, ByVal eventArgs As
    System.Windows.Forms.FormClosingEventArgs) Handles Me.FormClosing
    Dim Cancel As Boolean = eventArgs.Cancel
    Dim UnloadMode As System.Windows.Forms.CloseReason = eventArgs.CloseReason
    Dim sMsg As String = String.Empty
    Dim iAns As Integer
    If cAvax1.IsChanged = True Then
        sMsg = "Do you want to save the changes of the current drawing?"
        iAns = MsgBox(sMsg, MsgBoxStyle.Exclamation + MsgBoxStyle.YesNoCancel, "Avax Vector ActiveX")
        Select Case iAns
            Case MsgBoxResult.No
            Cancel = 0
            'Exit without save...
            Case MsgBoxResult.Yes
            Cancel = 0
            'Save the drawing...
            Case MsgBoxResult.Cancel
            Cancel = 1
        End Select
    End If
    eventArgs.Cancel = Cancel
End Sub
```

```
Private Sub Form1_FormClosed(ByVal eventSender As System.Object, ByVal eventArgs As
    System.Windows.Forms.FormClosedEventArgs) Handles Me.FormClosed
    cAvax1.EndAvax()
End Sub
```

IsCreatingCompoundPath

Purpose

Reports whether a compound-path input session currently exists.

Declaration

```
ReadOnly Property IsCreatingCompoundPath As Boolean
```

Property value

True after BeginCompoundPath has started a session, including before any points are supplied; False when no such session exists. It reports a draft, not whether an existing drawing object is a compound path. An unsuccessful FinishCompoundPath with too few points can leave the session active. Cancellation or successful ending/replacement of the drawing clears the draft.

VB.NET example

```
Option Strict On
Public Module IsCreatingCompoundPathExample
    Public Sub CancelCurrentDraft(cad As Avax.cAvax)
        If cad.IsCreatingCompoundPath Then
            cad.Command = Avax.AvaxCommand.Cancel_c
        End If
    End Sub
End Module
```

Related options and reference

Language

cAvax property

Purpose

Gets or sets the language for control dialogs, prompts and messages, using catalogs in the options directory.

VB.NET declaration

```
Property Language As AvaxLanguage
```

Property value

Type	Access	Meaning
AvaxLanguage	Read/write	Gets or sets the language for control dialogs, prompts and messages, using catalogs in the options directory.

Remarks

English is the default. AvaxLanguage country codes are English=1, Greek=30, Italian=39, French=33 and German=49. The options directory selected by DirsSet contains MSG1.OPT for Greek, MSG2.OPT for English, MSG3.OPT for Italian, MSG4.OPT for French and MSG5.OPT for German. Each control owns its catalog. Assigning the same language reloads it; changing the options directory also reloads it. Missing or malformed optional translations fall back to English.

The catalogs translate control prompts, dialogs, type descriptions and legacy messages. Application menus and messages remain supplied by the host. Greek selects the Greek legacy-vector-font mapping and direct Language assignment selects ISO-8859-7 XML encoding. Language_p and settings XML retain an explicit XMLencoding_p. Changing language retains the active command and accepted points.

Example

Setup

Add an Avax.cAvax Windows Forms control named `cAvax1`. Paste the procedures into the form class. Example setup and supporting files explains references, initialization and file paths.

VB.NET example

```
Private Sub Form1_Load(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs) Handles
    MyBase.Load
    CAvax1.Language = Avax.AvaxLanguage.English
    cAvax1.StartAvax()
End Sub
Private Sub Form1_FormClosed(ByVal eventSender As System.Object, ByVal eventArgs As
    System.Windows.Forms.FormClosedEventArgs) Handles Me.FormClosed
    cAvax1.EndAvax()
End Sub
```

LastHandle

cAvax property

Purpose

Returns the highest handle allocated in the current drawing, even if that object has since been deleted.

VB.NET declaration

```
ReadOnly Property LastHandle As Integer
```

Property value

Type	Access	Meaning
Integer	Read-only	Returns the highest handle allocated in the current drawing, even if that object has since been deleted.

Remarks

The value is 0 before any objects are created. Deleting an object does not reduce the allocation sequence, so this number can identify an object that no longer exists. Use ItemType or GetAvaxItemType to check whether a handle is live. New objects receive new numbers. DestroyDrawing, a new drawing and EndAvax reset the sequence. The value remains readable while the control is disabled, without changing drawing or history.

Example

Setup

Add an Avax.cAvax Windows Forms control named `cAvax1`. Paste the procedures into the form class. Example setup and supporting files explains references, initialization and file paths.

Add these Windows Forms controls or components to the form. Fields referenced by Handles clauses must be declared WithEvents, as the Visual Studio designer normally does.

Name	Type
Command1	Button

VB.NET example

```
Private Sub Form1_Load(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs) Handles
    MyBase.Load
    cAvax1.StartAvax()
    Call cAvax1.Add_Line(0, 0, 0, 10, 10, 0)
End Sub
Private Sub Form1_FormClosed(ByVal eventSender As System.Object, ByVal eventArgs As
    System.Windows.Forms.FormClosedEventArgs) Handles Me.FormClosed
    cAvax1.EndAvax()
End Sub
Private Sub Command1_Click(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs)
    Handles Command1.Click
    MsgBox("LastHandle=" & Format(CAvax1.LastHandle), MsgBoxStyle.Information, "LastHandle")
End Sub
```

LastImportWarnings

API reference

Purpose

Reads warnings collected by the most recent DXF/DWG import attempt or placement preview. An import can succeed while unsupported objects are omitted or their appearance is simplified. Use these messages to explain the result to the application user.

Declaration

```
Public ReadOnly Property LastImportWarnings As String()
```

Value

An array of warning messages, or an empty array when no warnings are available. Each read returns a separate array; changing its elements does not modify the control's messages. A DXF/DWG import attempt clears the preceding warning list before it reads the file. Installing another drawing replaces its current warning context.

This property is a v3 addition. Read it after the import call or from ImportWarningsAvailable. A warning is distinct from an import failure; check the import method's Boolean result as well.

VB.NET example

```
Option Strict On
Public Module ReadImportWarningsExample
    Public Sub ShowWarnings(cad As Avax.cAvax)
        For Each message As String In cad.LastImportWarnings
            System.Diagnostics.Debug.WriteLine(message)
        Next
    End Sub
End Module
```

LastOperationError

Purpose

Reads the control's current operation diagnostic, including messages recorded by exporters that return False.

Declaration

```
ReadOnly Property LastOperationError As String
```

Property value

A String containing the stored diagnostic, or an empty string when no message is available. Reading this property leaves its value unchanged. Check the operation's Boolean result first: a False result does not guarantee that a new message was recorded. For example, a control that cannot start an output operation can return False without replacing an earlier diagnostic. Successful operations commonly clear the diagnostic, but the property alone is not a success indicator.

VB.NET example

```
Option Strict On
Public Module LastOperationErrorExample
    Public Sub LogCurrentDiagnostic(cad As Avax.cAvax)
        Dim message As String = cad.LastOperationError
        If Not String.IsNullOrEmpty(message) Then
            System.Diagnostics.Debug.WriteLine(message)
        End If
    End Sub
End Module
```

Related options and reference

LockEdit

cAvax property

Purpose

Locks drawing editing

VB.NET declaration

```
Property LockEdit As Boolean
```

Property value

Type	Access	Meaning
Boolean	Read/write	A Boolean expression that if it is True the locks the drawing editing.

Remarks

Boolean, False by default. True prevents interactive drawing, selection, grip editing and interactive Undo while allowing navigation and programmatic drawing methods. Enabling the lock cancels pending selection, freehand and grip gestures immediately, including a lock enabled and then disabled from a mouse callback. Releasing an old mouse gesture cannot apply stale edits. Ordinary accepted command points remain available after unlocking. The property changes editing policy without changing stored geometry or history.

Example

Setup

Add an Avax.cAvax Windows Forms control named `cAvax1`. Paste the procedures into the form class. Example setup and supporting files explains references, initialization and file paths.

Add these Windows Forms controls or components to the form. Fields referenced by Handles clauses must be declared WithEvents, as the Visual Studio designer normally does.

Name	Type
Command1	Button

VB.NET example

```
Private Sub Form1_Load(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs) Handles
    MyBase.Load
    Command1.Text = "Lock edit"
    cAvax1.StartAvax()
    Call cAvax1.Add_Line(0, 0, 0, 10, 10, 0)
End Sub
Private Sub Form1_FormClosed(ByVal eventSender As System.Object, ByVal eventArgs As
    System.Windows.Forms.FormClosedEventArgs) Handles Me.FormClosed
    cAvax1.EndAvax()
End Sub
Private Sub Command1_Click(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs)
    Handles Command1.Click
    If cAvax1.LockEdit = True Then
        cAvax1.LockEdit = False
        Command1.Text = "Lock edit"
    Else
        cAvax1.LockEdit = True
        Command1.Text = "Unlock edit"
    End If
End Sub
```

OriginVisible

Purpose

Shows a symbol at drawing coordinates (0, 0) when that point lies inside the view. Its screen size stays fixed while zooming. The symbol moves with the world origin during pan and stays hidden when zero is outside the view. It is a view overlay and is excluded from exported drawing images.

Default: `False`.

VB.NET declaration

```
Property OriginVisible As Boolean
```

Example

```
Option Strict On
Imports Avax
Imports System.Drawing

Public Module Help_Display_OriginVisible
    Public Sub Run(cad As cAvax)
        cad.OriginVisible = True
    End Sub
End Module
```

Related options and reference

PageBackgroundColor

Purpose and value

Gets or sets the paper fill color shown in Page workspace mode. The default is `System.Drawing.Color.White`. Canvas mode uses the control's `BackColor`.

VB.NET declaration

```
Public Property PageBackgroundColor As System.Drawing.Color
```

Validation and behavior

The color must be opaque: its alpha channel must be 255. Assigning a transparent or partially transparent color, including `Color.Empty`, throws `ArgumentException` and retains the previous workspace settings. Changing the color does not move or resize drawing objects or the sheet.

The editor displays this color in Page mode even when PageBackgroundPrintExport is False. Workspace-page output uses the color when that property is True. Assign a color on the control's owning UI thread.

Persistence and notifications

The value belongs to the current drawing's workspace and is saved in native `.avax` files. Drawings without the newer background fields use white. A changed value participates in Undo/Redo and the WorkspaceChanged event; assigning the current value does not create another change.

Example

Call this procedure on the owning UI thread after `StartAvax()`.

```
Option Strict On
Imports System.Drawing
Imports Avax

Public Module Help_PageBackgroundColor
    Public Sub Run(cad As cAvax)
        cad.WorkspaceMode = AvaxWorkspaceMode.Page
        cad.PageBackgroundColor = Color.FromArgb(255, 245, 248, 252)
        Dim paperColor As Color = cad.PageBackgroundColor
    End Sub
End Module
```

Related options and reference

PageBackgroundPrintExport

Purpose and value

Gets or sets whether workspace-page printing and visual output include the stored paper color. The default is `True`.

VB.NET declaration

```
Public Property PageBackgroundPrintExport As Boolean
```

Output behavior

With True, workspace-page output uses PageBackgroundColor. With False, the page stays colored in the editor, while RenderWorkspacePage and workspace-page raster, PDF and printer output use white paper. Workspace-page SVG output omits its background rectangle.

This setting applies to operations that use the workspace sheet, including PrintWorkspaceToPDF and PrintWorkspaceToPrinter. It does not change drawing-object colors, paper dimensions, scale or margins. It accepts either Boolean value; the same live-control and UI-thread requirements as the other workspace properties apply.

Persistence and notifications

The value is saved with the drawing's workspace in native `.avax` files. Older drawings without this field use True. A changed value participates in Undo/Redo and WorkspaceChanged; assigning the current value does not create another change. Use native `.avax` to retain page settings; DXF/DWG output does not create workspace layouts from them.

Example

Call this procedure on the owning UI thread after `StartAvax()`. The existing editor paper color is retained.

```
Option Strict On
Imports Avax

Public Module Help_PageBackgroundPrintExport
    Public Sub Run(cad As cAvax)
        cad.PageBackgroundPrintExport = False
        Dim includePaperColor As Boolean = cad.PageBackgroundPrintExport
    End Sub
End Module
```

Related options and reference

PageCustomHeightMM

Purpose and value

Finite positive custom paper height in mm; default 297, used with epsUser.

VB.NET declaration

```
Property PageCustomHeightMM As Double
```

Example

Call this procedure on the owning UI thread with a control already initialized by `StartAvax()`. Handles must belong to that control. File paths are supplied by the caller.

```
Option Strict On
Imports System
Imports Avax

Public Module Help_PageCustomHeightMM
    Public Sub Run(cad As cAvax)
        cad.PageCustomHeightMM = 400
    End Sub
End Module
```

Related options and reference

PageCustomWidthMM

Purpose and value

Finite positive custom paper width in mm; default 210, used with epsUser.

VB.NET declaration

```
Property PageCustomWidthMM As Double
```

Example

Call this procedure on the owning UI thread with a control already initialized by `StartAvax()`. Handles must belong to that control. File paths are supplied by the caller.

```
Option Strict On
Imports System
Imports Avax

Public Module Help_PageCustomWidthMM
    Public Sub Run(cad As cAvax)
        cad.PageCustomWidthMM = 300
    End Sub
End Module
```

Related options and reference

PageHeightMM

Purpose and value

Read-only oriented paper height in physical mm.

VB.NET declaration

```
ReadOnly Property PageHeightMM As Double
```

Example

Call this procedure on the owning UI thread with a control already initialized by `StartAvax()`. Handles must belong to that control. File paths are supplied by the caller.

```
Option Strict On
Imports System
Imports Avax

Public Module Help_PageHeightMM
    Public Sub Run(cad As cAvax)
        Dim value As Double = cad.PageHeightMM
    End Sub
End Module
```

Related options and reference

PageMarginMM

Purpose and value

Equal margin on all four sides in mm; default 10, finite and nonnegative. Twice the margin must be less than the shorter paper side.

VB.NET declaration

```
Property PageMarginMM As Double
```

Example

Call this procedure on the owning UI thread with a control already initialized by `StartAvax()`. Handles must belong to that control. File paths are supplied by the caller.

```
Option Strict On
Imports System
Imports Avax

Public Module Help_PageMarginMM
    Public Sub Run(cad As cAvax)
        cad.PageMarginMM = 12
    End Sub
End Module
```

Related options and reference

PageMarginsVisible

Purpose and value

Default True. Controls the dashed editing guide; printing still clips to the configured margin when hidden.

VB.NET declaration

```
Property PageMarginsVisible As Boolean
```

Example

Call this procedure on the owning UI thread with a control already initialized by `StartAvax()`. Handles must belong to that control. File paths are supplied by the caller.

```
Option Strict On
Imports System
Imports Avax

Public Module Help_PageMarginsVisible
    Public Sub Run(cad As cAvax)
        cad.PageMarginsVisible = False
    End Sub
End Module
```

Related options and reference

PageNumber

Purpose and value

Gets or sets the number displayed in the workspace sheet label. It does not select a different drawing page or add printed drawing text. The value must be at least 1; the default is 1.

VB.NET declaration

```
Property PageNumber As Integer
```

Example

Call this procedure on the owning UI thread with a control already initialized by `StartAvax()`. Handles must belong to that control. File paths are supplied by the caller.

```
Option Strict On
Imports System
Imports Avax

Public Module Help_PageNumber
    Public Sub Run(cad As cAvax)
        cad.PageNumber = 2
    End Sub
End Module
```

Related options and reference

PageOrientation

Purpose and value

Default Portrait. Portrait puts the short side horizontally; Landscape puts the long side horizontally.

VB.NET declaration

```
Property PageOrientation As AvaxPageOrientation
```

Example

Call this procedure on the owning UI thread with a control already initialized by `StartAvax()`. Handles must belong to that control. File paths are supplied by the caller.

```
Option Strict On
Imports System
Imports Avax

Public Module Help_PageOrientation
    Public Sub Run(cad As cAvax)
        cad.PageOrientation = AvaxPageOrientation.Landscape
    End Sub
End Module
```

Related options and reference

PageOriginX

Purpose and value

Finite minimum X of the full sheet in drawing coordinates; default 0.

VB.NET declaration

```
Property PageOriginX As Double
```

Example

Call this procedure on the owning UI thread with a control already initialized by `StartAvax()`. Handles must belong to that control. File paths are supplied by the caller.

```
Option Strict On
Imports System
Imports Avax

Public Module Help_PageOriginX
    Public Sub Run(cad As cAvax)
        cad.PageOriginX = 5
    End Sub
End Module
```

Related options and reference

PageOriginY

Purpose and value

Finite minimum Y of the full sheet in drawing coordinates; default 0.

VB.NET declaration

```
Property PageOriginY As Double
```

Example

Call this procedure on the owning UI thread with a control already initialized by `StartAvax()`. Handles must belong to that control. File paths are supplied by the caller.

```
Option Strict On
Imports System
Imports Avax

Public Module Help_PageOriginY
    Public Sub Run(cad As cAvax)
        cad.PageOriginY = 10
    End Sub
End Module
```

Related options and reference

PagePaperSize

Purpose and value

Existing AvaxPaperSize; default epsA4. epsUser uses the custom dimensions.

VB.NET declaration

```
Property PagePaperSize As AvaxPaperSize
```

Example

Call this procedure on the owning UI thread with a control already initialized by `StartAvax()`. Handles must belong to that control. File paths are supplied by the caller.

```
Option Strict On
Imports System
Imports Avax

Public Module Help_PagePaperSize
    Public Sub Run(cad As cAvax)
        cad.PagePaperSize = AvaxPaperSize.epsA4
    End Sub
End Module
```

Related options and reference

PagePrintUnitMM

Purpose and value

Read-only DrawingUnitMM / PageScale: printed mm per coordinate unit.

VB.NET declaration

```
ReadOnly Property PagePrintUnitMM As Double
```

Example

Call this procedure on the owning UI thread with a control already initialized by `StartAvax()`. Handles must belong to that control. File paths are supplied by the caller.

```
Option Strict On
Imports System
Imports Avax

Public Module Help_PagePrintUnitMM
    Public Sub Run(cad As cAvax)
        Dim value As Double = cad.PagePrintUnitMM
    End Sub
End Module
```

Related options and reference

PageScale

Purpose and value

Finite positive scale denominator; default 100. Changes paper coverage, not geometry.

VB.NET declaration

```
Property PageScale As Double
```

Example

Call this procedure on the owning UI thread with a control already initialized by `StartAvax()`. Handles must belong to that control. File paths are supplied by the caller.

```
Option Strict On
Imports System
Imports Avax

Public Module Help_PageScale
    Public Sub Run(cad As cAvax)
        cad.PageScale = 100
    End Sub
End Module
```

Related options and reference

PageShadowVisible

Purpose and value

Default True. Controls the paper shadow in the editor, excluded from output.

VB.NET declaration

```
Property PageShadowVisible As Boolean
```

Example

Call this procedure on the owning UI thread with a control already initialized by `StartAvax()`. Handles must belong to that control. File paths are supplied by the caller.

```
Option Strict On
Imports System
Imports Avax

Public Module Help_PageShadowVisible
    Public Sub Run(cad As cAvax)
        cad.PageShadowVisible = False
    End Sub
End Module
```

Related options and reference

PageTitle

Purpose and value

Gets or sets the title displayed beside the page number in the workspace sheet label. The label is an editing decoration and is not automatically printed as drawing text. The title can contain at most 512 characters; the default is empty, and Nothing becomes an empty string.

VB.NET declaration

```
Property PageTitle As String
```

Example

Call this procedure on the owning UI thread with a control already initialized by `StartAvax()`. Handles must belong to that control. File paths are supplied by the caller.

```
Option Strict On
Imports System
Imports Avax

Public Module Help_PageTitle
    Public Sub Run(cad As cAvax)
        cad.PageTitle = "Ground floor"
    End Sub
End Module
```

Related options and reference

PageWidthMM

Purpose and value

Read-only oriented paper width in physical mm.

VB.NET declaration

```
ReadOnly Property PageWidthMM As Double
```

Example

Call this procedure on the owning UI thread with a control already initialized by `StartAvax()`. Handles must belong to that control. File paths are supplied by the caller.

```
Option Strict On
Imports System
Imports Avax

Public Module Help_PageWidthMM
    Public Sub Run(cad As cAvax)
        Dim value As Double = cad.PageWidthMM
    End Sub
End Module
```

Related options and reference

ParameterDialogInput

Purpose and default

Lets the application supply its own editor or values when a native command requests numeric parameters. The callback is optional and defaults to Nothing. With Nothing, the command retains its built-in behavior and `DialogEnabled_p` policy.

VB.NET declaration

```
Public Property ParameterDialogInput As Func(Of String, String(), Double(), Tuple(Of Boolean, Double()))
```

Callback arguments and result

Value	Meaning
First String argument	Localized dialog title.
String() argument	Detached, zero-based array of localized field labels.
Double() argument	Detached, zero-based initial values. Each value corresponds to the label at the same index.
Return value	Return Tuple.Create(True, replacementValues) to accept. Item2 must be a non-Nothing Double array with exactly the same length as the initial values. Return Nothing or a tuple with Item1=False to cancel.

Validation and behavior

The callback runs synchronously on the thread invoking the command. Use the control's owning UI thread and return only after the application's input is complete. Accepted values are copied and must pass that command's existing validation; for Polygon_c, the count must be a finite whole number from 3 through 60. A missing, wrong-length or invalid result cancels the parameter request and does not fall back to another dialog.

A supplied callback takes precedence over the built-in numeric dialog, including when `DialogEnabled_p` is False. Commands still apply their normal lifecycle and editing checks. Titles and labels may be translated; their numeric fields keep the same zero-based order.

Lifetime

This is a per-control application callback, not a drawing setting. It is not stored in native drawing files or workspace settings. Restore a previous callback or assign Nothing when the application no longer needs the override.

Example

Pass a started, enabled control on its owning UI thread. This temporary callback supplies six vertices to Polygon_c, then the procedure supplies its center and radius point. The previous callback is restored.

```
Option Strict On
Imports System
Imports Avax

Public Module Help_ParameterDialogInput
    Public Sub Run(cad As cAvax)
        Dim previous = cad.ParameterDialogInput
        Try
            cad.ParameterDialogInput = Function(title, labels, initial)
                                           If initial.Length <> 1 Then Return Nothing
                                           Return Tuple.Create(True, New Double() {6.0R})
                                       End Function
            cad.Command = AvaxCommand.Polygon_c
            If cad.Command <> AvaxCommand.Polygon_c OrElse
                Not cad.SetCoordinate(0, 0, 0) OrElse
                Not cad.SetCoordinate(10, 0, 0) Then
                Throw New InvalidOperationException("Polygon input was not accepted.")
            End If
        Finally
            cad.ParameterDialogInput = previous
        End Try
    End Sub
End Module
```

Related options and reference

PauseAvax

cAvax property

Purpose

Pauses normal interactive drawing input and navigation while retaining the active command and points; direct APIs remain available.

VB.NET declaration

```
Property PauseAvax As Boolean
```

Property value

Type	Access	Meaning
Boolean	Read/write	Pauses normal interactive drawing input and navigation while retaining the active command and points; direct APIs remain available.

Remarks

The default is False. True suspends default interactive point acceptance, wheel navigation, command completion and keyboard shortcuts. It cancels unfinished pointer gestures but retains the active command and accepted construction points. False resumes input from those points. The drawing remains visible and direct APIs remain available. Host mouse/key notifications remain independent; use PauseAvaxEvents to suppress AVAX notifications.

Example

Setup

Add an Avax.cAvax Windows Forms control named `cAvax1`. Paste the procedures into the form class. Example setup and supporting files explains references, initialization and file paths.

Add these Windows Forms controls or components to the form. Fields referenced by Handles clauses must be declared WithEvents, as the Visual Studio designer normally does.

Name	Type
Command1	Button
Check1	CheckBox

VB.NET example

```
Private Sub Check1_CheckStateChanged(ByVal eventSender As System.Object, ByVal eventArgs As
    System.EventArgs) Handles Check1.CheckStateChanged
    If Check1.CheckState = 1 Then
        CAvax1.PauseAvax = True
    Else
        CAvax1.PauseAvax = False
    End If
End Sub
Private Sub Form1_Load(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs) Handles
    MyBase.Load
    Command1.Text = "Add a circle"
    Check1.Text = "PauseAVAX"
    cAvax1.StartAvax()
End Sub
Private Sub Form1_FormClosed(ByVal eventSender As System.Object, ByVal eventArgs As
    System.Windows.Forms.FormClosedEventArgs) Handles Me.FormClosed
    cAvax1.EndAvax()
End Sub
Private Sub Command1_Click(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs)
    Handles Command1.Click
```

```
    CAvax1.Command = Avax.AvaxCommand.Circle_c
End Sub
```

PauseAvaxEvents

cAvax property

Purpose

When this property is True then AVAX stops sending events

VB.NET declaration

```
Property PauseAvaxEvents As Boolean
```

Property value

Type	Access	Meaning
Boolean	Read/write	A Boolean expression that if it is true then the AVAX events are paused.

Example

Setup

Add an Avax.cAvax Windows Forms control named cAvax1. Paste the procedures into the form class. Example setup and supporting files explains references, initialization and file paths.

Add these Windows Forms controls or components to the form. Fields referenced by Handles clauses must be declared WithEvents, as the Visual Studio designer normally does.

Name	Type
Check1	CheckBox
Label1	Label

VB.NET example

```
Private Sub CAvax1_BeforeClick1(ByVal Sender As Object, ByVal e As Avax.cAvax.BeforeClickEventArgs)
    Handles CAvax1.BeforeClick
    MsgBox("BeforeClick", MsgBoxStyle.Information, "Event BeforeClick")
End Sub
Private Sub Check1_CheckStateChanged(ByVal eventSender As System.Object, ByVal eventArgs As
    System.EventArgs) Handles Check1.CheckStateChanged
    If Check1.CheckState = 1 Then
        CAvax1.PauseAvaxEvents = True
    Else
        CAvax1.PauseAvaxEvents = False
    End If
End Sub
Private Sub Form1_Load(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs) Handles
    MyBase.Load
    Label1.Text = "Click on the AVAX window"
    Check1.Text = "PauseAvaxEvents"
    cAvax1.StartAvax()
End Sub
Private Sub Form1_FormClosed(ByVal eventSender As System.Object, ByVal eventArgs As
    System.Windows.Forms.FormClosedEventArgs) Handles Me.FormClosed
    cAvax1.EndAvax()
End Sub
```

Pen3DElevetion

cAvax property

Purpose

Sets or returns a value that determines the pen's elevation.

VB.NET declaration

```
Property Pen3DElevetion As Double
```

Property value

Type	Access	Meaning
Double	Read/write	A numeric expression that identifies the pen elevation

Remarks

Gets or sets the current pen elevation as a Double. The spelling Pen3DElevetion is retained for source compatibility. Newly created entities copy this elevation; existing entities retain their own values. Use Plane_On_c to mark subsequent entities as plane/extrusion entities and Plane_Off_c to turn that mode off. Endpoint Z coordinates remain independent. Elevation3D_p accesses the same pen value, and native .avax files preserve the full Double elevation.

Numeric changes from v2

Update the following variables, arrays or event-data receivers when porting the corresponding v2 call. The declarations above show the current overloads.

Parameter or value	v2 type	v3 type
Property value	Single	Double

For ByRef parameters, use a variable with the exact v3 type. Preserve array dimensions and the documented indexing. Moving from v2 to v3 explains conversion and legacy file-storage rules.

Example

Setup

Add an Avax.cAvax Windows Forms control named `cAvax1`. Paste the procedures into the form class. Example setup and supporting files explains references, initialization and file paths.

VB.NET example

```
Private Sub Form1_Load(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs) Handles
    MyBase.Load
    cAvax1.StartAvax()
    cAvax1.Pen3DElevetion = 0
    cAvax1.Pen3DHeight = 3.1
    CAvax1.Command = Avax.AvaxCommand.Plane_On_c
    Call cAvax1.Add_Line(0, 0, 0, 12, 0, 0)
    CAvax1.Command = Avax.AvaxCommand.Plane_Off_c
    'This line is a 3d face
End Sub
Private Sub Form1_FormClosed(ByVal eventSender As System.Object, ByVal eventArgs As
    System.Windows.Forms.FormClosedEventArgs) Handles Me.FormClosed
    cAvax1.EndAvax()
End Sub
```

Pen3DHeight

cAvax property

Purpose

Sets or returns a value that determines the pen's height.

VB.NET declaration

```
Property Pen3DHeight As Double
```

Property value

Type	Access	Meaning
Double	Read/write	A numeric expression that identifies the pen height.

Remarks

Gets or sets the current pen extrusion height as a Double. Newly created entities copy the value; changing the pen does not alter existing entities. Plane_On_c marks subsequent entities as planes and Plane_Off_c ends that mode. Signed heights are supported. Height3D_p accesses the same setting, and native .avax files preserve its Double precision. The original example creates a line with height 3.1 while plane mode is enabled.

Numeric changes from v2

Update the following variables, arrays or event-data receivers when porting the corresponding v2 call. The declarations above show the current overloads.

Parameter or value	v2 type	v3 type
Property value	Single	Double

For ByRef parameters, use a variable with the exact v3 type. Preserve array dimensions and the documented indexing. Moving from v2 to v3 explains conversion and legacy file-storage rules.

Example

Setup

Add an Avax.cAvax Windows Forms control named `cAvax1`. Paste the procedures into the form class. Example setup and supporting files explains references, initialization and file paths.

VB.NET example

```
Private Sub Form1_Load(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs) Handles
    MyBase.Load
    cAvax1.StartAvax()
    cAvax1.Pen3DElevetion = 0
    cAvax1.Pen3DHeight = 3.1
    CAvax1.Command = Avax.AvaxCommand.Plane_On_c
    Call cAvax1.Add_Line(0, 0, 0, 12, 0, 0)
    CAvax1.Command = Avax.AvaxCommand.Plane_Off_c
    'This line is a 3d face
End Sub
Private Sub Form1_FormClosed(ByVal eventSender As System.Object, ByVal eventArgs As
    System.Windows.Forms.FormClosedEventArgs) Handles Me.FormClosed
    cAvax1.EndAvax()
End Sub
```

PenColor

cAvax property

Purpose

Sets or returns a value that determines the pen's color (number from 0 to 255).

VB.NET declaration

```
Property PenColor As Integer
```

Property value

Type	Access	Meaning
Integer	Read/write	A numeric expression that identifies the pen color.

Remarks

Gets or sets the current color palette index, from 0 to 255, as an Integer. A creation method that uses the current-color sentinel -1 copies PenColor into the new entity. Explicit color arguments override the pen. Changing PenColor does not recolor existing entities. ColorPenIndex_p accesses the same per-control setting. The example selects palette indices 0 through 15 and draws a new line for each choice.

Numeric changes from v2

Update the following variables, arrays or event-data receivers when porting the corresponding v2 call. The declarations above show the current overloads.

Parameter or value	v2 type	v3 type
Property value	Short	Integer

For ByRef parameters, use a variable with the exact v3 type. Preserve array dimensions and the documented indexing. Moving from v2 to v3 explains conversion and legacy file-storage rules.

Example

Setup

Add an Avax.cAvax Windows Forms control named `cAvax1`. Paste the procedures into the form class. Example setup and supporting files explains references, initialization and file paths.

Add these Windows Forms controls or components to the form. Fields referenced by Handles clauses must be declared WithEvents, as the Visual Studio designer normally does.

Name	Type
Command1	Button
Combo1	ComboBox

VB.NET example

```
Private Sub Combo1_SelectedIndexChanged(ByVal eventSender As System.Object, ByVal eventArgs As
    System.EventArgs) Handles Combo1.SelectedIndexChanged
    CAvax1.pencolor = Val(Combo1.Text)
End Sub
Private Sub Form1_Load(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs) Handles
    MyBase.Load
    Dim i As Integer
    cAvax1.StartAvax()
    For i = 0 To 15
```

```
        Combo1.Items.Add(CStr(i))
    Next i
    Combo1.Text = "0"
End Sub
Private Sub Form1_FormClosed(ByVal eventSender As System.Object, ByVal eventArgs As
    System.Windows.Forms.FormClosedEventArgs) Handles Me.FormClosed
    cAvax1.EndAvax()
End Sub
Private Sub Command1_Click(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs)
    Handles Command1.Click
    Static v0 As Double
    Call CAvax1.Add_Line(0, v0, 0, 20, v0, 0, Nothing, -1, 2, 0, 0)
    v0 = v0 + 2
End Sub
```

PenColorByLayer

Purpose

Choose whether new objects inherit their layer color.

VB.NET declaration

```
Public Property PenColorByLayer As Boolean
```

Default False. True makes new objects follow their layer color when no explicit color is supplied, or the creation color argument is -1. Assigning PenColor selects an explicit color and clears this property. Existing objects are unchanged.

Example

```
Option Strict On
Public Module Help_PenColorByLayer
    Public Sub Run(cad As Avax.cAvax)
        cad.PenColorByLayer = True
        Dim handle As Integer = cad.Add_Line(0.0, 0.0, 0.0, 10.0, 0.0, 0.0)
    End Sub
End Module
```

Related options and reference

PenFontNumber

cAvax property

Purpose

Sets or returns a value that determines the pen's font number.

VB.NET declaration

```
Property PenFontNumber As Integer
```

Property value

Type	Access	Meaning
Integer	Read/write	A numeric expression that identifies the font number.

Numeric changes from v2

Update the following variables, arrays or event-data receivers when porting the corresponding v2 call. The declarations above show the current overloads.

Parameter or value	v2 type	v3 type
Property value	Short	Integer

For ByRef parameters, use a variable with the exact v3 type. Preserve array dimensions and the documented indexing. Moving from v2 to v3 explains conversion and legacy file-storage rules.

Example

Setup

Add an Avax.cAvax Windows Forms control named `cAvax1`. Paste the procedures into the form class. Example setup and supporting files explains references, initialization and file paths.

Add these Windows Forms controls or components to the form. Fields referenced by Handles clauses must be declared WithEvents, as the Visual Studio designer normally does.

Name	Type
Command1	Button
Combo1	ComboBox
Label1	Label

VB.NET example

```
Dim iFontNumber As Integer
Private Sub Form1_Load(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs) Handles
    MyBase.Load
    Dim lMax As Integer
    Label1.Text = "PenFontNumber"
    Command1.Text = "Add a text"
    lMax = System.Drawing.FontFamily.Families.Length
    If lMax > 255 Then lMax = 255
    Dim ScreenFonts(lMax) As String
    Dim lSets(lMax) As Integer
    For iFontNumber = 1 To lMax
        Combo1.Items.Add(Trim(Str(iFontNumber)))
        ScreenFonts(iFontNumber) = System.Drawing.FontFamily.Families(iFontNumber - 1).Name
    Next iFontNumber
    iFontNumber = 1
```

```
    Combo1.Text = "1"
    cAvax1.StartAvax()
    Call CAvax1.SetAvaxProperty(Avax.AvaxProp.TextScale_n, 5.0!)
    Call cAvax1.SetFonts(ScreenFonts, lSets)
    Call Combo1_SelectedIndexChanged(Combo1, New System.EventArgs())
End Sub
Private Sub Form1_FormClosed(ByVal eventSender As System.Object, ByVal eventArgs As
    System.Windows.Forms.FormClosedEventArgs) Handles Me.FormClosed
    cAvax1.EndAvax()
End Sub
Private Sub Combo1_SelectedIndexChanged(ByVal eventSender As System.Object, ByVal eventArgs As
    System.EventArgs) Handles Combo1.SelectedIndexChanged
    Dim sText As String = String.Empty
    Call CAvax1.DeleteAllItems()
    iFontNumber = Val(Combo1.Text)
    CAvax1.PenFontNumber = iFontNumber
    sText = "Current FontNumber=" & Format(iFontNumber) & vbCrLf & "Add a text now..."
    Call CAvax1.Add_Text(sText, 0, 4, 0, Nothing, 5.0!, iFontNumber, 0, -1, -1, -1, -1, False, False,
    False, False, -1)
End Sub
Private Sub Combo1_KeyPress(ByVal eventSender As System.Object, ByVal eventArgs As
    System.Windows.Forms.KeyPressEventArgs) Handles Combo1.KeyPress
    Dim KeyAscii As Integer = Asc(eventArgs.KeyChar)
    If KeyAscii = System.Windows.Forms.Keys.Return Then Call Combo1_SelectedIndexChanged(Combo1, New
    System.EventArgs())
    eventArgs.KeyChar = Chr(KeyAscii)
    If KeyAscii = 0 Then
        eventArgs.Handled = True
    End If
End Sub
Private Sub Command1_Click(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs)
    Handles Command1.Click
    CAvax1.Command = Avax.AvaxCommand.Text_c
End Sub
```

PenFontScale

cAvax property

Purpose

Sets or returns a value that determines the pen's Font Scale factor.

VB.NET declaration

```
Property PenFontScale As Double
```

Property value

Type	Access	Meaning
Double	Read/write	A numeric expression that identifies the pen's Font Scale factor.

Remarks

Gets or sets the current text pen scale as a Double multiplier. The default is 1. A nonpositive finite value restores 1; NaN and infinity throw ArgumentOutOfRangeException and leave the previous value unchanged. GetAvaxProperty/SetAvaxProperty with AvaxProp.TextScale_p access the same value. The interactive Text command multiplies this scale by TextInitHeight_p (default 0.3 drawing units). Changing the pen affects subsequent interactive text, not existing objects or the four-argument Add_Text overload. It preserves the active command, accepted points, drawing undo history and modified state. Each control has its own pen. Settings XML includes the value at Double precision; opening a drawing retains the receiving control's pen setting. Use the full Add_Text overload to supply an explicit creation scale.

Numeric changes from v2

Update the following variables, arrays or event-data receivers when porting the corresponding v2 call. The declarations above show the current overloads.

Parameter or value	v2 type	v3 type
Property value	Single	Double

For ByRef parameters, use a variable with the exact v3 type. Preserve array dimensions and the documented indexing. Moving from v2 to v3 explains conversion and legacy file-storage rules.

Example

Setup

Add an Avax.cAvax Windows Forms control named `cAvax1`. Paste the procedures into the form class. Example setup and supporting files explains references, initialization and file paths.

Add these Windows Forms controls or components to the form. Fields referenced by Handles clauses must be declared WithEvents, as the Visual Studio designer normally does.

Name	Type
Command1	Button

VB.NET example

```
Private Sub Form1_Load(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs) Handles
    MyBase.Load
    Command1.Text = "Add a text"
    cAvax1.StartAvax()
    cAvax1.PenFontScale = 20
End Sub
Private Sub Form1_FormClosed(ByVal eventSender As System.Object, ByVal eventArgs As
    System.Windows.Forms.FormClosedEventArgs) Handles Me.FormClosed
```

```
    cAvax1.EndAvax()
End Sub
Private Sub Command1_Click(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs)
    Handles Command1.Click
    CAvax1.Command = Avax.AvaxCommand.Text_c
End Sub
```

PenLayer

cAvax property

Purpose

Sets or returns a value that determines the pen's layer.

VB.NET declaration

```
Property PenLayer As Integer
```

Property value

Type	Access	Meaning
Integer	Read/write	A numeric expression that identifies the pen's layer.

Numeric changes from v2

Update the following variables, arrays or event-data receivers when porting the corresponding v2 call. The declarations above show the current overloads.

Parameter or value	v2 type	v3 type
Property value	Short	Integer

For ByRef parameters, use a variable with the exact v3 type. Preserve array dimensions and the documented indexing. Moving from v2 to v3 explains conversion and legacy file-storage rules.

Example

Setup

Add an Avax.cAvax Windows Forms control named cAvax1. Paste the procedures into the form class. Example setup and supporting files explains references, initialization and file paths.

Add these Windows Forms controls or components to the form. Fields referenced by Handles clauses must be declared WithEvents, as the Visual Studio designer normally does.

Name	Type
Command1	Button
Combo1	ComboBox

VB.NET example

```
Private Sub Combo1_SelectedIndexChanged(ByVal eventSender As System.Object, ByVal eventArgs As
    System.EventArgs) Handles Combo1.SelectedIndexChanged
    CAvax1.PenLayer = Val(Combo1.Text)
End Sub
Private Sub Form1_Load(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs) Handles
    MyBase.Load
    Dim i As Integer
    cAvax1.StartAvax()
    For i = 0 To 15
        Combo1.Items.Add(CStr(i))
    Next i
    Combo1.Text = "0"
End Sub
Private Sub Form1_FormClosed(ByVal eventSender As System.Object, ByVal eventArgs As
    System.Windows.Forms.FormClosedEventArgs) Handles Me.FormClosed
    cAvax1.EndAvax()
End Sub
```

```
Private Sub Command1_Click(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs)
    Handles Command1.Click
    Static v0 As Double
    Call CAvax1.Add_Line(0, v0, 0, 20, v0, 0, Nothing, -1, 2, -1, -1)
    v0 = v0 + 2
End Sub
```

PenLineStyle

cAvax property

Purpose

Sets or returns a value that determines the pen's line style.

VB.NET declaration

```
Property PenLineStyle As Integer
```

Property value

Type	Access	Meaning
Integer	Read/write	A numeric expression that identifies the pen's line style

Numeric changes from v2

Update the following variables, arrays or event-data receivers when porting the corresponding v2 call. The declarations above show the current overloads.

Parameter or value	v2 type	v3 type
Property value	Short	Integer

For ByRef parameters, use a variable with the exact v3 type. Preserve array dimensions and the documented indexing. Moving from v2 to v3 explains conversion and legacy file-storage rules.

Example

Setup

Add an Avax.cAvax Windows Forms control named `cAvax1`. Paste the procedures into the form class. Example setup and supporting files explains references, initialization and file paths.

Add these Windows Forms controls or components to the form. Fields referenced by Handles clauses must be declared WithEvents, as the Visual Studio designer normally does.

Name	Type
Command1	Button
Combo1	ComboBox

VB.NET example

```
Private Sub Combo1_SelectedIndexChanged(ByVal eventSender As System.Object, ByVal eventArgs As
    System.EventArgs) Handles Combo1.SelectedIndexChanged
    CAvax1.PenLineStyle = Val(Combo1.Text)
End Sub
Private Sub Form1_Load(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs) Handles
    MyBase.Load
    Dim i As Integer
    cAvax1.StartAvax()
    For i = 0 To 10
        Combo1.Items.Add(CStr(i))
    Next i
    Combo1.Text = "0"
End Sub
Private Sub Form1_FormClosed(ByVal eventSender As System.Object, ByVal eventArgs As
    System.Windows.Forms.FormClosedEventArgs) Handles Me.FormClosed
    cAvax1.EndAvax()
End Sub
```

```
Private Sub Command1_Click(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs)
    Handles Command1.Click
    Static v0 As Double
    Call cAvax1.Add_Line(0, v0, 0, 30, v0, 0)
    v0 = v0 + 2
End Sub
```

PenLineStyleByLayer

Purpose

Choose whether new objects inherit their layer line style.

VB.NET declaration

```
Public Property PenLineStyleByLayer As Boolean
```

Default False. True makes new objects follow their layer line style when no explicit style is supplied, or the creation style argument is -1. Assigning PenLineStyle clears this property. Point symbols and text alignment retain their own style conventions.

Example

```
Option Strict On
Public Module Help_PenLineStyleByLayer
    Public Sub Run(cad As Avax.cAvax)
        cad.PenLineStyleByLayer = True
        Dim handle As Integer = cad.Add_Line(0.0, 10.0, 0.0, 10.0, 10.0, 0.0)
    End Sub
End Module
```

Related options and reference

PenLineStyleScale

cAvax property

Purpose

Sets or returns a value that determines the pen's linetype scale factor.

VB.NET declaration

```
Property PenLineStyleScale As Double
```

Property value

Type	Access	Meaning
Double	Read/write	A numeric expression that identifies the linetype scale factor.

Numeric changes from v2

Update the following variables, arrays or event-data receivers when porting the corresponding v2 call. The declarations above show the current overloads.

Parameter or value	v2 type	v3 type
Property value	Single	Double

For ByRef parameters, use a variable with the exact v3 type. Preserve array dimensions and the documented indexing. Moving from v2 to v3 explains conversion and legacy file-storage rules.

Example

Setup

Add an Avax.cAvax Windows Forms control named `cAvax1`. Paste the procedures into the form class. Example setup and supporting files explains references, initialization and file paths.

Add these Windows Forms controls or components to the form. Fields referenced by Handles clauses must be declared WithEvents, as the Visual Studio designer normally does.

Name	Type
Combo1	ComboBox
Label1	Label

VB.NET example

```
Private Sub Form1_Load(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs) Handles
    MyBase.Load
    Dim scl As Double
    Label1.Text = "PenLineStyleScale"
    Combo1.Items.Add("0.5")
    For scl = 1! To 2.5! Step 0.2!
        Combo1.Items.Add(Trim(Str(scl)))
    Next scl
    Combo1.Items.Add("5")
    Combo1.Text = "1"
    cAvax1.StartAvax()
    Call CAvax1.Add_Line(0, 0, 0, 22, 0, 0, Nothing, -1, -1, 1, -1)
End Sub
Private Sub Form1_FormClosed(ByVal eventSender As System.Object, ByVal eventArgs As
    System.Windows.Forms.FormClosedEventArgs) Handles Me.FormClosed
    cAvax1.EndAvax()
End Sub
```

```
Private Sub Combo1_SelectedIndexChanged(ByVal eventSender As System.Object, ByVal eventArgs As
    System.EventArgs) Handles Combo1.SelectedIndexChanged
    CAvax1.PenLineStyleScale = Val(Combo1.Text)
End Sub
Private Sub Combo1_KeyPress(ByVal eventSender As System.Object, ByVal eventArgs As
    System.Windows.Forms.KeyPressEventArgs) Handles Combo1.KeyPress
    Dim KeyAscii As Integer = Asc(eventArgs.KeyChar)
    If KeyAscii = System.Windows.Forms.Keys.Return Then
        cAvax1.PenLineStyleScale = Val(Combo1.Text)
    End If
    eventArgs.KeyChar = Chr(KeyAscii)
    If KeyAscii = 0 Then
        eventArgs.Handled = True
    End If
End Sub
```

PenPointRadius

cAvax property

Purpose

Gets or sets the radius used to display styled point symbols, in drawing units. This is not the tessellation count for circles or arcs.

VB.NET declaration

```
Property PenPointRadius As Double
```

Property value

Type	Access	Meaning
Double	Read/write	Gets or sets the radius used to display styled point symbols, in drawing units. This is not the tessellation count for circles or arcs.

Remarks

The value must be finite. A nonpositive value restores the default radius of 0.2 drawing units.

Numeric changes from v2

Update the following variables, arrays or event-data receivers when porting the corresponding v2 call. The declarations above show the current overloads.

Parameter or value	v2 type	v3 type
Property value	Single	Double

For ByRef parameters, use a variable with the exact v3 type. Preserve array dimensions and the documented indexing. Moving from v2 to v3 explains conversion and legacy file-storage rules.

Example

Setup

Add an Avax.cAvax Windows Forms control named `cAvax1`. Paste the procedures into the form class. Example setup and supporting files explains references, initialization and file paths.

Add these Windows Forms controls or components to the form. Fields referenced by Handles clauses must be declared WithEvents, as the Visual Studio designer normally does.

Name	Type
Command1	Button

VB.NET example

```
Private Sub Form1_Load(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs) Handles
    MyBase.Load
    Command1.Text = "Add point"
    cAvax1.StartAvax()
    cAvax1.PenPointRadius = 1
    cAvax1.PenPointStyle = 5
End Sub
Private Sub Form1_FormClosed(ByVal eventSender As System.Object, ByVal eventArgs As
    System.Windows.Forms.FormClosedEventArgs) Handles Me.FormClosed
    cAvax1.EndAvax()
End Sub
Private Sub Command1_Click(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs)
    Handles Command1.Click
```

```
    Call cAvax1.Add_Point(0, 0, 0)
End Sub
```

PenPointStyle

cAvax property

Purpose

Sets or returns a value that determines the point style.

VB.NET declaration

```
Property PenPointStyle As Integer
```

Property value

Type	Access	Meaning
Integer	Read/write	A numeric expression that identifies the point style.

Numeric changes from v2

Update the following variables, arrays or event-data receivers when porting the corresponding v2 call. The declarations above show the current overloads.

Parameter or value	v2 type	v3 type
Property value	Short	Integer

For ByRef parameters, use a variable with the exact v3 type. Preserve array dimensions and the documented indexing. Moving from v2 to v3 explains conversion and legacy file-storage rules.

Example

Setup

Add an Avax.cAvax Windows Forms control named `cAvax1`. Paste the procedures into the form class. Example setup and supporting files explains references, initialization and file paths.

Add these Windows Forms controls or components to the form. Fields referenced by Handles clauses must be declared WithEvents, as the Visual Studio designer normally does.

Name	Type
Command1	Button

VB.NET example

```
Private Sub Form1_Load(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs) Handles
    MyBase.Load
    Command1.Text = "Add point"
    cAvax1.StartAvax()
    cAvax1.PenPointRadius = 1
    cAvax1.PenPointStyle = 5
End Sub
Private Sub Form1_FormClosed(ByVal eventSender As System.Object, ByVal eventArgs As
    System.Windows.Forms.FormClosedEventArgs) Handles Me.FormClosed
    cAvax1.EndAvax()
End Sub
Private Sub Command1_Click(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs)
    Handles Command1.Click
    Call cAvax1.Add_Point(0, 0, 0)
End Sub
```

PenWidth

cAvax property

Purpose

Sets or returns a value that determines the pen's width.

VB.NET declaration

```
Property PenWidth As Integer
```

Property value

Type	Access	Meaning
Integer	Read/write	A numeric expression that identifies the pen's width

Numeric changes from v2

Update the following variables, arrays or event-data receivers when porting the corresponding v2 call. The declarations above show the current overloads.

Parameter or value	v2 type	v3 type
Property value	Short	Integer

For ByRef parameters, use a variable with the exact v3 type. Preserve array dimensions and the documented indexing. Moving from v2 to v3 explains conversion and legacy file-storage rules.

Example

Setup

Add an Avax.cAvax Windows Forms control named `cAvax1`. Paste the procedures into the form class. Example setup and supporting files explains references, initialization and file paths.

Add these Windows Forms controls or components to the form. Fields referenced by Handles clauses must be declared WithEvents, as the Visual Studio designer normally does.

Name	Type
Command1	Button
Combo1	ComboBox

VB.NET example

```
Private Sub Combo1_SelectedIndexChanged(ByVal eventSender As System.Object, ByVal eventArgs As
    System.EventArgs) Handles Combo1.SelectedIndexChanged
    CAvax1.PenWidth = Val(Combo1.Text)
End Sub
Private Sub Form1_Load(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs) Handles
    MyBase.Load
    Dim i As Integer
    cAvax1.StartAvax()
    For i = 1 To 10
        Combo1.Items.Add(CStr(i))
    Next i
    Combo1.Text = "1"
End Sub
Private Sub Form1_FormClosed(ByVal eventSender As System.Object, ByVal eventArgs As
    System.Windows.Forms.FormClosedEventArgs) Handles Me.FormClosed
    cAvax1.EndAvax()
End Sub
```

```
Private Sub Command1_Click(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs)
    Handles Command1.Click
    Static v0 As Double
    Call cAvax1.Add_Line(0, v0, 0, 20, v0, 0)
    v0 = v0 + 2
End Sub
```

PreserveSelectionAppearance

API reference

Purpose

Shows selected drawing objects with their original colours and fills while retaining the selection frame and grips. Use this setting when the application must display selected artwork in its ordinary appearance.

VB.NET declaration

```
Property PreserveSelectionAppearance As Boolean
```

Value and behavior

The default is False, which keeps the legacy selection highlight. True preserves object appearance during screen painting. Selection membership, selection events, grip interaction, object geometry and drawing attributes are unchanged. The setting also applies when a prepared scene or a retained drawing raster is used; changing it invalidates the existing geometry frame for the next presentation.

This is a setting of each control instance, rather than an item or drawing-file attribute. It can be set before StartAvax and remains configured after EndAvax. A disposed control ignores later assignments. Assign it on the control's owning UI thread. It creates no drawing edit or Undo entry. Normal redraw/update settings determine when the next screen frame is presented. Ordinary drawing exports already use the objects' own appearance independently of selection highlighting.

VB.NET example

Call this procedure with the control whose selection you want to display in its original colours.

```
Option Strict On
Imports Avax

Public Module SelectionAppearanceExample
    Public Sub KeepOriginalColours(cad As cAvax)
        cad.PreserveSelectionAppearance = True
    End Sub
End Module
```

ResolvedObjectSnap

API reference

Purpose

Reads the type of the currently displayed object-snap candidate. Use this property to report the snap resolved while the user moves the pointer over a drawing.

Declaration

```
ReadOnly Property ResolvedObjectSnap As AvaxOSnapType
```

Property value

Returns the candidate's AvaxOSnapType value while its marker is current. Returns None_o when no marker is available or the control is unstarted, disabled or disposed. Reading the property does not search the drawing, consume a construction point or change a pending snap request. Pointer movement and drafting operations update the candidate.

GetOSnapStatus reads the pending one-shot R2 snap request. ResolvedObjectSnap reads the candidate currently resolved by object snapping; these values can differ. It is a single candidate type, not the AutoOSnap_p bitmask.

VB.NET example

```
Option Strict On
Public Module ResolvedObjectSnapExample
    ' Call from the host's drawing-pointer/status update handler.
    Public Function SnapCaption(cad As Avax.cAvax) As String
        Dim candidate As Avax.AvaxOSnapType = cad.ResolvedObjectSnap
        If candidate = Avax.AvaxOSnapType.None_o Then Return ""
        Return candidate.ToString()
    End Function
End Module
```

RulerX

Purpose

Shows or hides the horizontal X ruler. It can be enabled independently of RulerY. Coordinate conversion methods use pixels relative to the drawing area, excluding rulers and scroll bars. Changing visibility preserves the active drawing command and its accepted points.

Default: `False`.

VB.NET declaration

```
Property RulerX As Boolean
```

Example

```
Option Strict On
Imports Avax
Imports System.Drawing

Public Module Help_Display_RulerX
    Public Sub Run(cad As cAvax)
        cad.RulerX = True
    End Sub
End Module
```

Related options and reference

RulerY

Purpose

Shows or hides the vertical Y ruler. RulerX and RulerY may be used separately or together. The ruler follows drawing coordinates during pan and zoom. Changing visibility preserves the drawing geometry and the active command.

Default: `False`.

VB.NET declaration

```
Property RulerY As Boolean
```

Example

```
Option Strict On
Imports Avax
Imports System.Drawing

Public Module Help_Display_RulerY
    Public Sub Run(cad As cAvax)
        cad.RulerY = True
    End Sub
End Module
```

Related options and reference

ScrollBars

cAvax property

Purpose

Returns or sets a value indicating whether object AVAX has horizontal or vertical scroll bars

VB.NET declaration

```
Property ScrollBars As AvaxScrollBarsStatus
```

Property value

Type	Access	Meaning
AvaxScrollBarsStatus	Read/write	An expression that defines the scroll bars status

Declared default: `0`.

Remarks

Displays NoBars, Horizontal, Vertical or Both using AvaxScrollBarsStatus; the default is NoBars. The native scroll bars pan the drawing on their respective axes. Changing their visibility adjusts the drawing area while preserving its center and world units per pixel. A completed thumb gesture can be restored with Previous_c; when ZoomUndo_On_c is enabled it also adds one Zoom_u record to normal Undo/Redo. Scrolling and scrollbar layout preserve model geometry.

Example

Setup

Add an Avax.cAvax Windows Forms control named `cAvax1`. Paste the procedures into the form class. Example setup and supporting files explains references, initialization and file paths.

Add these Windows Forms controls or components to the form. Fields referenced by Handles clauses must be declared WithEvents, as the Visual Studio designer normally does.

Name	Type
Combo1	ComboBox

VB.NET example

```
Private Sub Combo1_SelectedIndexChanged(ByVal eventSender As System.Object, ByVal eventArgs As
    System.EventArgs) Handles Combo1.SelectedIndexChanged
    Select Case Combo1.Text
        Case "Horizontal"
        CAvax1.ScrollBars = Avax.AvaxScrollBarsStatus.horizontal
        Case "Vertical"
        CAvax1.ScrollBars = Avax.AvaxScrollBarsStatus.Vertical
        Case "Both"
        CAvax1.ScrollBars = Avax.AvaxScrollBarsStatus.Both
        Case "No scrollbars"
        CAvax1.ScrollBars = Avax.AvaxScrollBarsStatus.NoBars
    End Select
End Sub
Private Sub Form1_Load(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs) Handles
    MyBase.Load
    cAvax1.StartAvax()
    Combo1.Items.Insert(0, "Horizontal")
    Combo1.Items.Insert(1, "Vertical")
    Combo1.Items.Insert(2, "Both")
    Combo1.Items.Insert(3, "No scrollbars")
    Combo1.Text = "Both"
End Sub
```

```
Private Sub Form1_FormClosed(ByVal eventSender As System.Object, ByVal eventArgs As
    System.Windows.Forms.FormClosedEventArgs) Handles Me.FormClosed
    cAvax1.EndAvax()
End Sub
```

SnapToStepStatus

cAvax property

Purpose

Returns whether snap to a regular coordinate step is enabled. Use SnapToStepSet or the corresponding AvaxProp setting to change it.

VB.NET declaration

```
ReadOnly Property SnapToStepStatus As Boolean
```

Property value

Type	Access	Meaning
Boolean	Read-only	Returns whether snap to a regular coordinate step is enabled. Use SnapToStepSet or the corresponding AvaxProp setting to change it.

Remarks

Read-only. True means grid-step snapping is enabled; False means it is disabled. Use SnapToStepSet to set the spacing and enable or disable snapping. This property does not control grid visibility or object snaps. It corresponds to GetAvaxProperty(AvaxProp.SnapOn_p). The original v2 declaration was read-only despite the read/write wording in its help.

Example

Setup

Add an Avax.cAvax Windows Forms control named `cAvax1`. Paste the procedures into the form class. Example setup and supporting files explains references, initialization and file paths.

Add these Windows Forms controls or components to the form. Fields referenced by Handles clauses must be declared WithEvents, as the Visual Studio designer normally does.

Name	Type
Command1	Button

VB.NET example

```
Private Sub Command2_Click()
    CAvax1.Command = Avax.AvaxCommand.Circle_c
End Sub
Private Sub Form1_Load(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs) Handles
    MyBase.Load
    cAvax1.StartAvax()
End Sub
Private Sub Form1_FormClosed(ByVal eventSender As System.Object, ByVal eventArgs As
    System.Windows.Forms.FormClosedEventArgs) Handles Me.FormClosed
    cAvax1.EndAvax()
End Sub
Private Sub Command1_Click(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs)
    Handles Command1.Click
    Dim ii As Integer
    Dim txt As String = String.Empty
    Dim stepX As Double
    Dim stepY As Double
    Dim Answer As Boolean
    Dim PropValue As Boolean
    PropValue = cAvax1.SnapToStepStatus
    txt = "Snap status is :" & Str(PropValue)
    ii = MsgBox(txt)
    ' Set snap steps
```

```
    stepX = 5 : stepY = 5
    Answer = cAvax1.SnapToStepSet(stepX, stepY)
    PropValue = cAvax1.SnapToStepStatus
    txt = "Now, The snap status is :" & Str(PropValue)
    ii = MsgBox(txt)
End Sub
```

SnapToStepX

cAvax property

Purpose

Returns the horizontal snap step in drawing units. Use SnapToStepSet to configure the step.

VB.NET declaration

```
ReadOnly Property SnapToStepX As Double
```

Property value

Type	Access	Meaning
Double	Read-only	Returns the horizontal snap step in drawing units. Use SnapToStepSet to configure the step.

Remarks

Read-only Double value in drawing units. Use SnapToStepSet to change both steps. A zero spacing on either axis disables snapping; when snapping is disabled the stored spacing is still returned. This property corresponds to GetAvaxProperty(AvaxProp.SnapStepX_p). The original v2 declaration was read-only despite the read/write wording in its help.

Numeric changes from v2

Update the following variables, arrays or event-data receivers when porting the corresponding v2 call. The declarations above show the current overloads.

Parameter or value	v2 type	v3 type
Property value	Single	Double

For ByRef parameters, use a variable with the exact v3 type. Preserve array dimensions and the documented indexing. Moving from v2 to v3 explains conversion and legacy file-storage rules.

Example

Setup

Add an Avax.cAvax Windows Forms control named `cAvax1`. Paste the procedures into the form class. Example setup and supporting files explains references, initialization and file paths.

Add these Windows Forms controls or components to the form. Fields referenced by Handles clauses must be declared WithEvents, as the Visual Studio designer normally does.

Name	Type
Command1	Button

VB.NET example

```
Private Sub Form1_Load(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs) Handles
    MyBase.Load
    cAvax1.StartAvax()
End Sub
Private Sub Form1_FormClosed(ByVal eventSender As System.Object, ByVal eventArgs As
    System.Windows.Forms.FormClosedEventArgs) Handles Me.FormClosed
    cAvax1.EndAvax()
End Sub
Private Sub Command1_Click(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs)
    Handles Command1.Click
    Call cAvax1.SnapToStepSet(5, 10)
```

```
    MsgBox("SnapToStepX=" & cAvax1.SnapToStepX & vbCrLf & "SnapToStepY=" & cAvax1.SnapToStepY,
    MsgBoxStyle.Information, "SnapToStepX-SnapToStepY")
End Sub
```

SnapToStepY

cAvax property

Purpose

Returns the vertical snap step in drawing units. Use SnapToStepSet to configure the step.

VB.NET declaration

```
ReadOnly Property SnapToStepY As Double
```

Property value

Type	Access	Meaning
Double	Read-only	Returns the vertical snap step in drawing units. Use SnapToStepSet to configure the step.

Remarks

Read-only Double value in drawing units. Use SnapToStepSet to change both steps. A zero spacing on either axis disables snapping; when snapping is disabled the stored spacing is still returned. This property corresponds to GetAvaxProperty(AvaxProp.SnapStepY_p). The original v2 declaration was read-only despite the read/write wording in its help.

Numeric changes from v2

Update the following variables, arrays or event-data receivers when porting the corresponding v2 call. The declarations above show the current overloads.

Parameter or value	v2 type	v3 type
Property value	Single	Double

For ByRef parameters, use a variable with the exact v3 type. Preserve array dimensions and the documented indexing. Moving from v2 to v3 explains conversion and legacy file-storage rules.

Example

Setup

Add an Avax.cAvax Windows Forms control named `cAvax1`. Paste the procedures into the form class. Example setup and supporting files explains references, initialization and file paths.

Add these Windows Forms controls or components to the form. Fields referenced by Handles clauses must be declared WithEvents, as the Visual Studio designer normally does.

Name	Type
Command1	Button

VB.NET example

```
Private Sub Form1_Load(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs) Handles
    MyBase.Load
    cAvax1.StartAvax()
End Sub
Private Sub Form1_FormClosed(ByVal eventSender As System.Object, ByVal eventArgs As
    System.Windows.Forms.FormClosedEventArgs) Handles Me.FormClosed
    cAvax1.EndAvax()
End Sub
Private Sub Command1_Click(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs)
    Handles Command1.Click
    Call cAvax1.SnapToStepSet(5, 10)
    MsgBox("SnapToStepX=" & cAvax1.SnapToStepX & vbCrLf & "SnapToStepY=" & cAvax1.SnapToStepY,
```

```
    MsgBoxStyle.Information, "SnapToStepX-SnapToStepY")
End Sub
```

UndoRecorderOn

cAvax property

Purpose

Sets or returns the Undo Recorder status

VB.NET declaration

```
Property UndoRecorderOn As Boolean
```

Property value

Type	Access	Meaning
Boolean	Read/write	A Boolean expression that defines the Undo Recorder status

Example

Setup

Add an Avax.cAvax Windows Forms control named `cAvax1`. Paste the procedures into the form class. Example setup and supporting files explains references, initialization and file paths.

Add these Windows Forms controls or components to the form. Fields referenced by Handles clauses must be declared WithEvents, as the Visual Studio designer normally does.

Name	Type
Command1	Button

VB.NET example

```
Private Sub Form1_Load(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs) Handles
    MyBase.Load
    Command1.Text = "Undo"
    cAvax1.StartAvax()
    '----
    cAvax1.UndoRecorderOn = True
    Call cAvax1.Add_Line(0, 0, 0, 10, 10, 0) '--+
    Call cAvax1.Add_Line(10, 0, 0, 0, 10, 0) '  |-> 1 undo
    Call cAvax1.Add_Line(0, 5, 0, 10, 5, 0) '--+
    cAvax1.UndoRecorderOn = False
    '----
    Call cAvax1.Add_Circle(5, 5, 0, 8) '-> 1 undo
    '----
    Call cAvax1.Add_Circle(5, 5, 0, 6) '-> 1 undo
    '----
End Sub
Private Sub Form1_FormClosed(ByVal eventSender As System.Object, ByVal eventArgs As
    System.Windows.Forms.FormClosedEventArgs) Handles Me.FormClosed
    cAvax1.EndAvax()
End Sub
Private Sub Command1_Click(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs)
    Handles Command1.Click
    CAvax1.Command = Avax.AvaxCommand.Undo_c
End Sub
```

UserName

cAvax property

Purpose

Gets or sets the user name stored on this control instance for the application licensing workflow.

VB.NET declaration

```
Property UserName As String
```

Property value

Type	Access	Meaning
String	Read/write	Gets or sets the user name stored on this control instance for the application licensing workflow.

Remarks

The initial value is an empty String. Assignment retains the supplied text, including case, surrounding whitespace, Unicode and Nothing. The value belongs to this control and is retained across StartAvax, EndAvax and drawing replacement. The property is hidden in the designer Properties window; an assigned name can be serialized as an ordinary String property.

Example

Setup

Add an Avax.cAvax Windows Forms control named `cAvax1`. Paste the procedures into the form class. Example setup and supporting files explains references, initialization and file paths.

Add these Windows Forms controls or components to the form. Fields referenced by Handles clauses must be declared WithEvents, as the Visual Studio designer normally does.

Name	Type
Command1	Button

VB.NET example

```
Private Sub Form1_Load(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs) Handles
    MyBase.Load
    cAvax1.StartAvax()
    Command1.Text = "Get User Name"
End Sub
Private Sub Form1_FormClosed(ByVal eventSender As System.Object, ByVal eventArgs As
    System.Windows.Forms.FormClosedEventArgs) Handles Me.FormClosed
    cAvax1.EndAvax()
End Sub
Private Sub Command1_Click(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs)
    Handles Command1.Click
    MsgBox(CAvax1.UserName())
End Sub
```

UserSerialNumber

cAvax property

Purpose

Gets or sets this control runtime serial number; explicit assignment overrides the process key supplied by Unlock.

VB.NET declaration

```
Property UserSerialNumber As String
```

Property value

Type	Access	Meaning
String	Read/write	Gets or sets this control runtime serial number; explicit assignment overrides the process key supplied by Unlock.

Remarks

Until an explicit per-control assignment is made, the getter follows the process runtime key accepted by Unlock. An explicit value overrides that process key for this control and is retained across drawing start/end. Assignment trims surrounding whitespace and converts letters to uppercase before validation. Invalid text remains readable but does not grant licensing. Nothing or an empty String creates an explicit empty override. Assign runtime keys at application startup; this property is excluded from form-designer serialization. Version 3 requires a version 3 license.

Example

Setup

Add an Avax.cAvax Windows Forms control named `cAvax1`. Paste the procedures into the form class. Example setup and supporting files explains references, initialization and file paths.

Add these Windows Forms controls or components to the form. Fields referenced by Handles clauses must be declared WithEvents, as the Visual Studio designer normally does.

Name	Type
Command1	Button

VB.NET example

```
Private Sub Form1_Load(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs) Handles
    MyBase.Load
    cAvax1.StartAvax()
    Command1.Text = "Get User Serial Number"
End Sub
Private Sub Form1_FormClosed(ByVal eventSender As System.Object, ByVal eventArgs As
    System.Windows.Forms.FormClosedEventArgs) Handles Me.FormClosed
    cAvax1.EndAvax()
End Sub
Private Sub Command1_Click(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs)
    Handles Command1.Click
    MsgBox(CAvax1.UserSerialNumber())
End Sub
```

WorkspaceMode

Purpose and value

Default Canvas. Canvas=0 hides the paper; Page=1 shows it. Both share the same scale and settings.

VB.NET declaration

```
Property WorkspaceMode As AvaxWorkspaceMode
```

Example

Call this procedure on the owning UI thread with a control already initialized by `StartAvax()`. Handles must belong to that control. File paths are supplied by the caller.

```
Option Strict On
Imports System
Imports Avax

Public Module Help_WorkspaceMode
    Public Sub Run(cad As cAvax)
        cad.WorkspaceMode = AvaxWorkspaceMode.Page
    End Sub
End Module
```

Related options and reference

XMLDoc

cAvax property

Purpose

Returns an XML representation of the current drawing. To insert a drawing from an XML file, use Add_XML.

VB.NET declaration

```
ReadOnly Property XMLDoc As System.Xml.XmlDocument
```

Property value

Type	Access	Meaning
System.Xml.XmlDocument	Read-only	A System.Xml.XmlDocument expression that defines the XML document that contains the entire information of the drawing

Example

Setup

Add an Avax.cAvax Windows Forms control named `cAvax1`. Paste the procedures into the form class. Example setup and supporting files explains references, initialization and file paths.

Add these Windows Forms controls or components to the form. Fields referenced by Handles clauses must be declared WithEvents, as the Visual Studio designer normally does.

Name	Type
Button1	Button

VB.NET example

```
Private Sub Form1_Load(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs) Handles
    MyBase.Load
    Call CAvax1.StartAvax()
    Call CAvax1.Add_Line(0, 5, 0, 10, 5, 0, Nothing, 1, 1, -1, -1)
    Call CAvax1.Add_Line(5, 0, 0, 5, 10, 0, Nothing, 1, 1, -1, -1)
    Call CAvax1.Add_Circle(5, 5, 0, 3, Nothing, 5, -1, -1, -1)
End Sub
Private Sub Form1_FormClosed(ByVal eventSender As System.Object, ByVal eventArgs As
    System.Windows.Forms.FormClosedEventArgs) Handles Me.FormClosed
    Call cAvax1.EndAvax()
End Sub
Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
    Button1.Click
    Dim doc As Xml.XmlDocument
    Dim xmlFile As String = String.Empty
    doc = CAvax1.XMLDoc
    xmlFile = My.Application.Info.DirectoryPath & "\..\temp.xml"
    doc.Save(xmlFile)
    System.Diagnostics.Process.Start(xmlFile)
End Sub
```

Events

Use the linked topic list in Contents or the PDF bookmarks to locate a member or guide.

Events

Select a topic for its purpose, declaration, argument descriptions and example.

- AutoOSnapType
- BeforeClick
- BeforeCopy
- BeforeMirror
- BeforeMove
- BeforeRotate
- BeforeScale
- BeforeTransform
- Change
- DblClick
- DrawingRepainted
- EnterFocus
- ExitFocus
- MouseIn
- MouseOut
- MouseStop
- NewUserItems
- OnClick
- OnKeyDown
- OnKeyPress
- OnKeyUp
- OnMouseDown
- OnMouseMove
- OnMouseUp
- OnPaint
- OnResize
- PrintingPage
- SelectChange
- SelectedPoints
- SelectRegionStretch
- SetExternalLines
- UserOsnap
- ZoomChange

Additional members

- AfterPaint
- AvaxControlBehavior
- AvaxControlCaption

- AvaxControlChange
- AvaxControlClick
- AvaxControlDblClick
- AvaxControlErased
- AvaxControlGotFocus
- AvaxControlKeyDown
- AvaxControlKeyPress
- AvaxControlKeyUp
- AvaxControlList
- AvaxControlLostFocus
- AvaxControlMouseDown
- AvaxControlMouseMove
- AvaxControlMouseUp
- AvaxControlPictureFile
- AvaxControlText
- AvaxControlTooltip
- AvaxControlUnErased
- AvaxControlValidate
- AvaxControlValue
- ImportWarningsAvailable
- OnDragDrop
- OnDragOver
- OnGiveFeedback
- OperationFailed
- Progress
- Redo
- Undo
- WorkspaceChanged

OLE and paint completion events

API reference

These events belong to the AVAX drawing control. Create one example handler per control after StartAvax, retain it in the host form, and dispose it when detaching the host. Event handlers run on the owning Windows Forms thread.

OLE drawing drops accept complete native v3 data and the legacy AVAX clipboard format without using the Windows clipboard. See OleDrag for an outgoing selection-copy drag. The default file-drop importer supports AVAX drawings and blocks, DXF/DWG, PLT/HPG, SHP, SVG, XML, TXT, RTF and PCX. Bitmap, metafile, animated-image, AAF and AVI files are collected into one embedded picture sequence, following the R2 multi-picture convention. Existing drawing content is preserved and the batch is committed as one reversible action. AVI playback and sound are described under Add_AVI.

AfterPaint

Purpose

Raised after the control has painted the drawing and the OnPaint overlay for a complete frame. It has no parameters. Use it to update host status associated with the displayed drawing. PauseAvaxEvents suppresses it. Ending or replacing the drawing, or changing event suppression during an earlier paint callback, stops obsolete trailing paint notifications.

VB.NET declaration

```
Event AfterPaint()
```

Example

```
Option Strict On
Imports System
Imports System.Windows.Forms
Imports Avax

Public NotInheritable Class Help_AfterPaint
    Implements IDisposable
    Private ReadOnly source As cAvax
    Public Property PaintedFrames As Integer
    Public Sub New(cad As cAvax)
        If cad Is Nothing Then Throw New ArgumentNullException(NameOf(cad))
        source = cad
        AddHandler source.AfterPaint, AddressOf OnEvent
    End Sub
    Private Sub OnEvent()
        PaintedFrames += 1
    End Sub
    Public Sub Dispose() Implements IDisposable.Dispose
        RemoveHandler source.AfterPaint, AddressOf OnEvent
    End Sub
End Class
```

OnDragOver

Purpose

Reports a native OLE drag entering, moving over or leaving the drawing. State is 0 for enter, 2 for over and 1 for leave. X/Y are screen pixels stored as Double; Button and Shift are Integer bit masks. Return an Effect from DragDropEffects, or set fCancel to reject the drag. The resulting native effect is restricted to the source allowed-effect mask. OLEDropMode_p = 0 disables acceptance and callbacks. An unstarted, ended or disabled control does not publish this notification.

VB.NET declaration

```
Event OnDragOver(ByVal Sender As Object, ByVal e As cAvax.DragOverEventArgs)
```

Example

```
Option Strict On
Imports System
Imports System.Windows.Forms
```

```
Imports Avax

Public NotInheritable Class Help_OnDragOver
    Implements IDisposable
    Private ReadOnly source As cAvax
    Public Sub New(cad As cAvax)
        If cad Is Nothing Then Throw New ArgumentNullException(NameOf(cad))
        source = cad
        AddHandler source.OnDragOver, AddressOf OnEvent
    End Sub
    Private Sub OnEvent(sender As Object, e As cAvax.DragOverEventArgs)
        If e.Data IsNot Nothing AndAlso (e.Data.GetDataPresent(DataFormats.FileDrop) OrElse
            e.Data.GetDataPresent("Gardos AVAX v3 Drawing", False) OrElse
            e.Data.GetDataPresent("AVAX Clipboard Data Version 1.1", False)) Then
            e.Effect = CInt(DragDropEffects.Copy)
        Else
            e.Effect = CInt(DragDropEffects.None)
        End If
    End Sub
    Public Sub Dispose() Implements IDisposable.Dispose
        RemoveHandler source.OnDragOver, AddressOf OnEvent
    End Sub
End Class
```

OnDragDrop

Purpose

Reports an OLE drop before the default import. Data contains the DataObject, X/Y are screen pixel coordinates as Double, and Button/Shift are Integer masks. Set fCancel to cancel, or return replacement Data to choose other files or an AVAX drawing payload. The default handler converts the screen point into drawing coordinates and inserts the native/legacy drawing payload or imports all supported files into the current drawing. It preserves existing objects, filename, workspace and prior work instead of opening/replacing the drawing. One Undo removes the complete drop. A bad or unsupported file rejects the whole batch before insertion. LockEdit or PauseAvax prevents default editing. A callback that replaces/restarts/edits the drawing, changes the active command or disables drops abandons stale default processing. PauseAvaxEvents suppresses notifications while allowing an otherwise valid default import.

VB.NET declaration

```
Event OnDragDrop(ByVal Sender As Object, ByVal e As cAvax.DragDropEventArgs)
```

Example

```
Option Strict On
Imports System
Imports System.Windows.Forms
Imports Avax

Public NotInheritable Class Help_OnDragDrop
    Implements IDisposable
    Private ReadOnly source As cAvax
    Public Property CancelDrop As Boolean
    Public Property LastScreenX As Double
    Public Property LastScreenY As Double
    Public Sub New(cad As cAvax)
        If cad Is Nothing Then Throw New ArgumentNullException(NameOf(cad))
        source = cad
        AddHandler source.OnDragDrop, AddressOf OnEvent
    End Sub
    Private Sub OnEvent(sender As Object, e As cAvax.DragDropEventArgs)
        LastScreenX = e.x : LastScreenY = e.y
        e.fCancel = CancelDrop
    End Sub
    Public Sub Dispose() Implements IDisposable.Dispose
        RemoveHandler source.OnDragDrop, AddressOf OnEvent
    End Sub
End Class
```

OnGiveFeedback

Purpose

Provides cursor feedback during a native OLE drag. Effect is an Integer DragDropEffects value. Set DefaultCursors to False when supplying your own Cursor.Current; True uses the standard drag cursor. fCancel also suppresses default cursor selection. This feedback event does not itself start or cancel an OLE gesture. It is suppressed before StartAvax, after EndAvax, when disabled and by PauseAvaxEvents.

VB.NET declaration

```
Event OnGiveFeedback(ByVal Sender As Object, ByVal e As cAvax.GiveFeedbackEventArgs)
```

Example

```
Option Strict On
Imports System
Imports System.Windows.Forms
Imports Avax

Public NotInheritable Class Help_OnGiveFeedback
    Implements IDisposable
    Private ReadOnly source As cAvax
    Public Sub New(cad As cAvax)
        If cad Is Nothing Then Throw New ArgumentNullException(NameOf(cad))
        source = cad
        AddHandler source.OnGiveFeedback, AddressOf OnEvent
    End Sub
    Private Sub OnEvent(sender As Object, e As cAvax.GiveFeedbackEventArgs)
        e.DefaultCursors = False
        Cursor.Current = Cursors.Hand
    End Sub
    Public Sub Dispose() Implements IDisposable.Dispose
        RemoveHandler source.OnGiveFeedback, AddressOf OnEvent
    End Sub
End Class
```

AutoOSnapType

cAvax event

Purpose

Reports the automatic object snap selected by the control, so the host can display or respond to the snap type.

VB.NET declaration

```
Event AutoOSnapType(ByVal Sender As Object, ByVal e As cAvax.AutoOSnapTypeEventArgs)
```

Parameters

Name	Type and passing	Meaning
Sender	ByVal Sender As Object	Is the object that raises the event
e	ByVal e As AutoOSnapTypeEventArgs	Event data. The fields are described below; writable cancellation or output fields are read by the control when the handler returns.

Use the declaration for the overload you call. ByRef arguments can receive outputs; declare variables and arrays with the exact indicated type. Optional defaults are shown in the declaration.

Event data

Member	Declaration	Meaning
e.fCancel	fCancel As Boolean	If set true then no OSnap happens
e.OSnapType	OSnapType As AvaxOSnapType	Is the type of the OSnap that took place
e.x	x As Double	Is the x coordinate of the point that OSnap took place
e.y	y As Double	Is the y coordinate of the point that OSnap took place

An event has no return value. Handle it with a procedure whose arguments match the declaration. Where a cancellation flag or output array is documented, assign it on the supplied event data object.

Remarks

Raised synchronously when a non-user object-snap candidate is resolved. OSnapType identifies the candidate; x and y are writable Double coordinates, and fCancel=True rejects that candidate. Subscribers share one event-data object in .NET subscription order, so a later subscriber sees earlier edits and may change them. A valid enum type and finite coordinate overrides are honored; None_o, an invalid type or nonfinite coordinates produce no resolved candidate. Changing a tangent-first coordinate or type removes its deferred curve constraint. User_o detection raises UserOsnap instead. PauseAvaxEvents suppresses this notification while ordinary snapping continues. A handler exception propagates to a programmatic caller and prevents later subscribers from running; the resolution/publication guards are released so a later retry can succeed. GetOSnapStatus reports a requested one-shot mode, not this candidate. If a handler changes the pending mode, automatic mask, aperture, geometry, drawing, view, axis convention, horizontal factor or control/input lifetime, the old result is abandoned and cannot consume newer state or accept a stale coordinate. Editing only this event's result fields applies to the current candidate normally.

Numeric changes from v2

Update the following variables, arrays or event-data receivers when porting the corresponding v2 call. The declarations above show the current overloads.

Parameter or value	v2 type	v3 type
e.x	Single	Double
e.y	Single	Double

For ByRef parameters, use a variable with the exact v3 type. Preserve array dimensions and the documented indexing. Moving from v2 to v3 explains conversion and legacy file-storage rules.

Example

Setup

Add an Avax.cAvax Windows Forms control named `cAvax1`. Paste the procedures into the form class. Example setup and supporting files explains references, initialization and file paths.

Add these Windows Forms controls or components to the form. Fields referenced by Handles clauses must be declared WithEvents, as the Visual Studio designer normally does.

Name	Type
Button1	Button

VB.NET example

```
Private Sub Form1_Load(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs) Handles
    MyBase.Load
    Call cAvax1.StartAvax()
    Dim AutoOsnapStatus As Integer
    AutoOsnapStatus = Avax.AvaxOSnapType.Endpoint_o
    Call cAvax1.SetAvaxProperty(Avax.AvaxProp.AutoOSnap_n, AutoOsnapStatus)
    Call cAvax1.Add_Line(0, 0, 10, 10, 10, 0)
End Sub
Private Sub Form1_FormClosed(ByVal eventSender As System.Object, ByVal eventArgs As
    System.Windows.Forms.FormClosedEventArgs) Handles Me.FormClosed
    Call cAvax1.EndAvax()
End Sub
Private Sub cAvax1_AutoOSnapType(Sender As System.Object, e As Avax.cAvax.AutoOSnapTypeEventArgs) Handles
    cAvax1.AutoOSnapType
    If e.OSnapType = Avax.AvaxOSnapType.Endpoint_o Then
        e.fCancel = True
    End If
End Sub
Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
    Button1.Click
    CAvax1.Command = Avax.AvaxCommand.Line_c
End Sub
Private Sub CAvax1_MouseUp(ByVal sender As Object, ByVal e As System.Windows.Forms.MouseEventArgs) Handles
     CAvax1.MouseUp
    If e.Button = Windows.Forms.MouseButtons.Right Then
        CAvax1.Command = Avax.AvaxCommand.Cancel_c
    End If
End Sub
```

Using event handlers

Embedded control events

API reference

These events belong to the AVAX drawing control and are raised for controls embedded in drawing objects and inline documents. Keep the original ByRef handler declarations when migrating source. Former Short values are Integer and former Single coordinates are Double. Use ParentHandle and Index to identify the drawing field; a reused native editor is not its identity.

PauseAvaxEvents suppresses these notifications. Examples use one disposable handler instance per drawing control: create the handler after StartAvax, retain it in the host form, and dispose it when the host detaches. Event code runs on the owning Windows Forms thread.

AvaxControlBehavior

Purpose

Supplies visibility, enabled and locked state when an embedded control is refreshed. Set bVisible to False to hide it. A hidden control stops requesting its remaining caption, text, value, list, picture and tooltip data during that refresh.

VB.NET declaration

```
Event AvaxControlBehavior(ByRef eType As AvaxControlType, ByRef ParentHandle As Integer, ByRef Index As
    Integer, ByRef sName As String, ByRef sTag As String, ByRef lStatus As AvaxControlStatus, ByRef bVisible
    As Boolean, ByRef bEnabled As Boolean, ByRef bLocked As Boolean)
```

Parameters

Name	Type	Meaning
eType	AvaxControlType	The type of the embedded control that caused this callback.
ParentHandle	Integer	Integer identity of the containing drawing object or application-assigned transient parent.
Index	Integer	Child control index within the parent. It is separate from a one-based registry slot; ordinary inline documents start their child indices at 0.
sName	String	Name assigned to the control in its drawing data.
sTag	String	Application tag assigned to the control.
lStatus	AvaxControlStatus	AvaxControlStatus snapshot for this callback.
bVisible	Boolean	Boolean visibility supplied by the host. False hides the control and stops its remaining data requests in this refresh.
bEnabled	Boolean	Boolean input-enabled state supplied by the host.
bLocked	Boolean	Boolean editing-lock state supplied by the host; True prevents normal editing of that field.

Example

```
Option Strict On
Imports System
Imports Avax

Public NotInheritable Class Help_AvaxControlBehavior
    Implements IDisposable
    Private ReadOnly source As cAvax
    Public Property LastParentHandle As Integer
    Public Property LastIndex As Integer
    Public Sub New(cad As cAvax)
        If cad Is Nothing Then Throw New ArgumentNullException(NameOf(cad))
        source = cad
        AddHandler source.AvaxControlBehavior, AddressOf OnEvent
    End Sub
    Private Sub OnEvent(ByRef [eType] As AvaxControlType, ByRef [ParentHandle] As Integer, ByRef [Index]
    As Integer, ByRef [sName] As String, ByRef [sTag] As String, ByRef [lStatus] As AvaxControlStatus, ByRef
```

```
    [bVisible] As Boolean, ByRef [bEnabled] As Boolean, ByRef [bLocked] As Boolean)
        LastParentHandle = ParentHandle
        LastIndex = Index
        If sTag = "read-only" Then bLocked = True
    End Sub
    Public Sub Dispose() Implements IDisposable.Dispose
        RemoveHandler source.AvaxControlBehavior, AddressOf OnEvent
    End Sub
End Class
```

AvaxControlCaption

Purpose

Supplies the caption for a button, label, checkbox or option control. Replace sCaption to display application data. Text fields and combo boxes use AvaxControlText; a picture control uses AvaxControlPictureFile.

VB.NET declaration

```
Event AvaxControlCaption(ByRef eType As AvaxControlType, ByRef ParentHandle As Integer, ByRef Index As
    Integer, ByRef sName As String, ByRef sTag As String, ByRef lStatus As AvaxControlStatus, ByRef sCaption
    As String)
```

Parameters

Name	Type	Meaning
eType	AvaxControlType	The type of the embedded control that caused this callback.
ParentHandle	Integer	Integer identity of the containing drawing object or application-assigned transient parent.
Index	Integer	Child control index within the parent. It is separate from a one-based registry slot; ordinary inline documents start their child indices at 0.
sName	String	Name assigned to the control in its drawing data.
sTag	String	Application tag assigned to the control.
lStatus	AvaxControlStatus	AvaxControlStatus snapshot for this callback.
sCaption	String	Replace this String with the caption to display.

Example

```
Option Strict On
Imports System
Imports Avax

Public NotInheritable Class Help_AvaxControlCaption
    Implements IDisposable
    Private ReadOnly source As cAvax
    Public Property LastParentHandle As Integer
    Public Property LastIndex As Integer
    Public Sub New(cad As cAvax)
        If cad Is Nothing Then Throw New ArgumentNullException(NameOf(cad))
        source = cad
        AddHandler source.AvaxControlCaption, AddressOf OnEvent
    End Sub
    Private Sub OnEvent(ByRef [eType] As AvaxControlType, ByRef [ParentHandle] As Integer, ByRef [Index]
    As Integer, ByRef [sName] As String, ByRef [sTag] As String, ByRef [lStatus] As AvaxControlStatus, ByRef
    [sCaption] As String)
        LastParentHandle = ParentHandle
        LastIndex = Index
        sCaption = "Execute"
    End Sub
    Public Sub Dispose() Implements IDisposable.Dispose
        RemoveHandler source.AvaxControlCaption, AddressOf OnEvent
    End Sub
```

```
End Class
```

AvaxControlChange

Purpose

Reports an edit to the text of an embedded field or combo box. sNewText contains the proposed text. The handler may replace it; the returned text is applied to the native editor and drawing without recursively publishing the same correction.

VB.NET declaration

```
Event AvaxControlChange(ByRef eType As AvaxControlType, ByRef ParentHandle As Integer, ByRef Index As
    Integer, ByRef sName As String, ByRef sTag As String, ByRef lStatus As AvaxControlStatus, ByRef sNewText
    As String)
```

Parameters

Name	Type	Meaning
eType	AvaxControlType	The type of the embedded control that caused this callback.
ParentHandle	Integer	Integer identity of the containing drawing object or application-assigned transient parent.
Index	Integer	Child control index within the parent. It is separate from a one-based registry slot; ordinary inline documents start their child indices at 0.
sName	String	Name assigned to the control in its drawing data.
sTag	String	Application tag assigned to the control.
lStatus	AvaxControlStatus	AvaxControlStatus snapshot for this callback.
sNewText	String	Text supplied by the control or data request. Replace it with the value your application wants the editor to use.

Example

```
Option Strict On
Imports System
Imports Avax

Public NotInheritable Class Help_AvaxControlChange
    Implements IDisposable
    Private ReadOnly source As cAvax
    Public Property LastParentHandle As Integer
    Public Property LastIndex As Integer
    Public Property LastText As String
    Public Sub New(cad As cAvax)
        If cad Is Nothing Then Throw New ArgumentNullException(NameOf(cad))
        source = cad
        AddHandler source.AvaxControlChange, AddressOf OnEvent
    End Sub
    Private Sub OnEvent(ByRef [eType] As AvaxControlType, ByRef [ParentHandle] As Integer, ByRef [Index]
    As Integer, ByRef [sName] As String, ByRef [sTag] As String, ByRef [lStatus] As AvaxControlStatus, ByRef
    [sNewText] As String)
        LastParentHandle = ParentHandle
        LastIndex = Index
        sNewText = If(sNewText, "").Trim()
        LastText = sNewText
    End Sub
    Public Sub Dispose() Implements IDisposable.Dispose
        RemoveHandler source.AvaxControlChange, AddressOf OnEvent
    End Sub
End Class
```

AvaxControlClick

Purpose

Reports a click on an embedded control. ParentHandle and Index identify the control in the drawing. ListIndex is 0 for ordinary clicks and is the one-based selected row for a committed combo-box choice.

VB.NET declaration

```
Event AvaxControlClick(ByRef eType As AvaxControlType, ByRef ParentHandle As Integer, ByRef Index As
    Integer, ByRef ListIndex As Integer, ByRef sName As String, ByRef sTag As String, ByRef lStatus As
    AvaxControlStatus)
```

Parameters

Name	Type	Meaning
eType	AvaxControlType	The type of the embedded control that caused this callback.
ParentHandle	Integer	Integer identity of the containing drawing object or application-assigned transient parent.
Index	Integer	Child control index within the parent. It is separate from a one-based registry slot; ordinary inline documents start their child indices at 0.
ListIndex	Integer	0 for an ordinary click; a committed combo-box choice reports its one-based row index.
sName	String	Name assigned to the control in its drawing data.
sTag	String	Application tag assigned to the control.
lStatus	AvaxControlStatus	AvaxControlStatus snapshot for this callback.

Example

```
Option Strict On
Imports System
Imports Avax

Public NotInheritable Class Help_AvaxControlClick
    Implements IDisposable
    Private ReadOnly source As cAvax
    Public Property LastParentHandle As Integer
    Public Property LastIndex As Integer
    Public Property LastListIndex As Integer
    Public Sub New(cad As cAvax)
        If cad Is Nothing Then Throw New ArgumentNullException(NameOf(cad))
        source = cad
        AddHandler source.AvaxControlClick, AddressOf OnEvent
    End Sub
    Private Sub OnEvent(ByRef [eType] As AvaxControlType, ByRef [ParentHandle] As Integer, ByRef [Index]
    As Integer, ByRef [ListIndex] As Integer, ByRef [sName] As String, ByRef [sTag] As String, ByRef [lStatus]
     As AvaxControlStatus)
        LastParentHandle = ParentHandle
        LastIndex = Index
        LastListIndex = ListIndex
    End Sub
    Public Sub Dispose() Implements IDisposable.Dispose
        RemoveHandler source.AvaxControlClick, AddressOf OnEvent
    End Sub
End Class
```

AvaxControlDblClick

Purpose

Reports a double click on an embedded control. Use the parent/child identity to choose an application action. ListIndex follows the same convention as AvaxControlClick.

VB.NET declaration

```
Event AvaxControlDblClick(ByRef eType As AvaxControlType, ByRef ParentHandle As Integer, ByRef Index As
    Integer, ByRef ListIndex As Integer, ByRef sName As String, ByRef sTag As String, ByRef lStatus As
    AvaxControlStatus)
```

Parameters

Name	Type	Meaning
eType	AvaxControlType	The type of the embedded control that caused this callback.
ParentHandle	Integer	Integer identity of the containing drawing object or application-assigned transient parent.
Index	Integer	Child control index within the parent. It is separate from a one-based registry slot; ordinary inline documents start their child indices at 0.
ListIndex	Integer	0 for an ordinary click; a committed combo-box choice reports its one-based row index.
sName	String	Name assigned to the control in its drawing data.
sTag	String	Application tag assigned to the control.
lStatus	AvaxControlStatus	AvaxControlStatus snapshot for this callback.

Example

```
Option Strict On
Imports System
Imports Avax

Public NotInheritable Class Help_AvaxControlDblClick
    Implements IDisposable
    Private ReadOnly source As cAvax
    Public Property LastParentHandle As Integer
    Public Property LastIndex As Integer
    Public Property LastListIndex As Integer
    Public Sub New(cad As cAvax)
        If cad Is Nothing Then Throw New ArgumentNullException(NameOf(cad))
        source = cad
        AddHandler source.AvaxControlDblClick, AddressOf OnEvent
    End Sub
    Private Sub OnEvent(ByRef [eType] As AvaxControlType, ByRef [ParentHandle] As Integer, ByRef [Index]
    As Integer, ByRef [ListIndex] As Integer, ByRef [sName] As String, ByRef [sTag] As String, ByRef [lStatus]
     As AvaxControlStatus)
        LastParentHandle = ParentHandle
        LastIndex = Index
        LastListIndex = ListIndex
    End Sub
    Public Sub Dispose() Implements IDisposable.Dispose
        RemoveHandler source.AvaxControlDblClick, AddressOf OnEvent
    End Sub
End Class
```

AvaxControlErased

Purpose

Reports that a parent drawing object containing embedded controls has been removed. The parent is reported once even when it owns several children. Retired native children stop publishing input events.

VB.NET declaration

```
Event AvaxControlErased(ByRef ParentHandle As Integer)
```

Parameters

Name	Type	Meaning
ParentHandle	Integer	Integer identity of the containing drawing object or application-assigned transient parent.

Example

```
Option Strict On
Imports System
Imports Avax

Public NotInheritable Class Help_AvaxControlErased
    Implements IDisposable
    Private ReadOnly source As cAvax
    Public Property LastParentHandle As Integer
    Public Sub New(cad As cAvax)
        If cad Is Nothing Then Throw New ArgumentNullException(NameOf(cad))
        source = cad
        AddHandler source.AvaxControlErased, AddressOf OnEvent
    End Sub
    Private Sub OnEvent(ByRef [ParentHandle] As Integer)
        LastParentHandle = ParentHandle
    End Sub
    Public Sub Dispose() Implements IDisposable.Dispose
        RemoveHandler source.AvaxControlErased, AddressOf OnEvent
    End Sub
End Class
```

AvaxControlGotFocus

Purpose

Reports entry into an embedded editor. The identity belongs to the drawing control and child field, even when a native editor instance is reused for another field.

VB.NET declaration

```
Event AvaxControlGotFocus(ByRef eType As AvaxControlType, ByRef ParentHandle As Integer, ByRef Index As
    Integer, ByRef sName As String, ByRef sTag As String, ByRef lStatus As AvaxControlStatus)
```

Parameters

Name	Type	Meaning
eType	AvaxControlType	The type of the embedded control that caused this callback.
ParentHandle	Integer	Integer identity of the containing drawing object or application-assigned transient parent.
Index	Integer	Child control index within the parent. It is separate from a one-based registry slot; ordinary inline documents start their child indices at 0.
sName	String	Name assigned to the control in its drawing data.
sTag	String	Application tag assigned to the control.
lStatus	AvaxControlStatus	AvaxControlStatus snapshot for this callback.

Example

```
Option Strict On
Imports System
Imports Avax

Public NotInheritable Class Help_AvaxControlGotFocus
    Implements IDisposable
    Private ReadOnly source As cAvax
```

```
    Public Property LastParentHandle As Integer
    Public Property LastIndex As Integer
    Public Property LastName As String
    Public Sub New(cad As cAvax)
        If cad Is Nothing Then Throw New ArgumentNullException(NameOf(cad))
        source = cad
        AddHandler source.AvaxControlGotFocus, AddressOf OnEvent
    End Sub
    Private Sub OnEvent(ByRef [eType] As AvaxControlType, ByRef [ParentHandle] As Integer, ByRef [Index]
    As Integer, ByRef [sName] As String, ByRef [sTag] As String, ByRef [lStatus] As AvaxControlStatus)
        LastParentHandle = ParentHandle
        LastIndex = Index
        LastName = sName
    End Sub
    Public Sub Dispose() Implements IDisposable.Dispose
        RemoveHandler source.AvaxControlGotFocus, AddressOf OnEvent
    End Sub
End Class
```

AvaxControlKeyDown

Purpose

Reports a virtual-key press in an embedded editor. Shift is captured from that key message: Shift=1, Ctrl=2, Alt=4, combined by addition/bitwise Or. Set KeyCode to 0 to consume the key and suppress its following character.

VB.NET declaration

```
Event AvaxControlKeyDown(ByRef eType As AvaxControlType, ByRef ParentHandle As Integer, ByRef Index As
    Integer, ByRef sName As String, ByRef sTag As String, ByRef lStatus As AvaxControlStatus, ByRef KeyCode As
     Integer, ByRef Shift As Integer)
```

Parameters

Name	Type	Meaning
eType	AvaxControlType	The type of the embedded control that caused this callback.
ParentHandle	Integer	Integer identity of the containing drawing object or application-assigned transient parent.
Index	Integer	Child control index within the parent. It is separate from a one-based registry slot; ordinary inline documents start their child indices at 0.
sName	String	Name assigned to the control in its drawing data.
sTag	String	Application tag assigned to the control.
lStatus	AvaxControlStatus	AvaxControlStatus snapshot for this callback.
KeyCode	Integer	Integer Windows virtual-key code. Set to 0 to consume the notification.
Shift	Integer	Integer modifier mask: Shift=1, Ctrl=2, Alt=4. Keyboard events use the modifiers of that key message.

Example

```
Option Strict On
Imports System
Imports Avax

Public NotInheritable Class Help_AvaxControlKeyDown
    Implements IDisposable
    Private ReadOnly source As cAvax
    Public Property LastParentHandle As Integer
    Public Property LastIndex As Integer
    Public Sub New(cad As cAvax)
        If cad Is Nothing Then Throw New ArgumentNullException(NameOf(cad))
        source = cad
        AddHandler source.AvaxControlKeyDown, AddressOf OnEvent
```

```
    End Sub
    Private Sub OnEvent(ByRef [eType] As AvaxControlType, ByRef [ParentHandle] As Integer, ByRef [Index]
    As Integer, ByRef [sName] As String, ByRef [sTag] As String, ByRef [lStatus] As AvaxControlStatus, ByRef
    [KeyCode] As Integer, ByRef [Shift] As Integer)
        LastParentHandle = ParentHandle
        LastIndex = Index
        If KeyCode = CInt(System.Windows.Forms.Keys.Escape) Then KeyCode = 0
    End Sub
    Public Sub Dispose() Implements IDisposable.Dispose
        RemoveHandler source.AvaxControlKeyDown, AddressOf OnEvent
    End Sub
End Class
```

AvaxControlKeyPress

Purpose

Reports a UTF-16 character in an embedded editor. KeyAscii is the Integer value produced by AscW; it may be negative for code units above U+7FFF. Use ChrW to interpret it. Set it to 0 to consume the character, or assign another valid AscW value to replace the character. Greek text is preserved.

VB.NET declaration

```
Event AvaxControlKeyPress(ByRef eType As AvaxControlType, ByRef ParentHandle As Integer, ByRef Index As
    Integer, ByRef sName As String, ByRef sTag As String, ByRef lStatus As AvaxControlStatus, ByRef KeyAscii
    As Integer)
```

Parameters

Name	Type	Meaning
eType	AvaxControlType	The type of the embedded control that caused this callback.
ParentHandle	Integer	Integer identity of the containing drawing object or application-assigned transient parent.
Index	Integer	Child control index within the parent. It is separate from a one-based registry slot; ordinary inline documents start their child indices at 0.
sName	String	Name assigned to the control in its drawing data.
sTag	String	Application tag assigned to the control.
lStatus	AvaxControlStatus	AvaxControlStatus snapshot for this callback.
KeyAscii	Integer	Integer signed AscW UTF-16 code unit. Return 0 to consume it or another valid code unit to replace it.

Example

```
Option Strict On
Imports System
Imports Avax

Public NotInheritable Class Help_AvaxControlKeyPress
    Implements IDisposable
    Private ReadOnly source As cAvax
    Public Property LastParentHandle As Integer
    Public Property LastIndex As Integer
    Public Sub New(cad As cAvax)
        If cad Is Nothing Then Throw New ArgumentNullException(NameOf(cad))
        source = cad
        AddHandler source.AvaxControlKeyPress, AddressOf OnEvent
    End Sub
    Private Sub OnEvent(ByRef [eType] As AvaxControlType, ByRef [ParentHandle] As Integer, ByRef [Index]
    As Integer, ByRef [sName] As String, ByRef [sTag] As String, ByRef [lStatus] As AvaxControlStatus, ByRef
    [KeyAscii] As Integer)
        LastParentHandle = ParentHandle
        LastIndex = Index
```

```
        If KeyAscii = 9 Then KeyAscii = 0
    End Sub
    Public Sub Dispose() Implements IDisposable.Dispose
        RemoveHandler source.AvaxControlKeyPress, AddressOf OnEvent
    End Sub
End Class
```

AvaxControlKeyUp

Purpose

Reports a virtual-key release in an embedded editor. Modifier bits come from the corresponding key message. Set KeyCode to 0 to mark the native key notification handled.

VB.NET declaration

```
Event AvaxControlKeyUp(ByRef eType As AvaxControlType, ByRef ParentHandle As Integer, ByRef Index As
    Integer, ByRef sName As String, ByRef sTag As String, ByRef lStatus As AvaxControlStatus, ByRef KeyCode As
     Integer, ByRef Shift As Integer)
```

Parameters

Name	Type	Meaning
eType	AvaxControlType	The type of the embedded control that caused this callback.
ParentHandle	Integer	Integer identity of the containing drawing object or application-assigned transient parent.
Index	Integer	Child control index within the parent. It is separate from a one-based registry slot; ordinary inline documents start their child indices at 0.
sName	String	Name assigned to the control in its drawing data.
sTag	String	Application tag assigned to the control.
lStatus	AvaxControlStatus	AvaxControlStatus snapshot for this callback.
KeyCode	Integer	Integer Windows virtual-key code. Set to 0 to consume the notification.
Shift	Integer	Integer modifier mask: Shift=1, Ctrl=2, Alt=4. Keyboard events use the modifiers of that key message.

Example

```
Option Strict On
Imports System
Imports Avax

Public NotInheritable Class Help_AvaxControlKeyUp
    Implements IDisposable
    Private ReadOnly source As cAvax
    Public Property LastParentHandle As Integer
    Public Property LastIndex As Integer
    Public Property LastKeyCode As Integer
    Public Sub New(cad As cAvax)
        If cad Is Nothing Then Throw New ArgumentNullException(NameOf(cad))
        source = cad
        AddHandler source.AvaxControlKeyUp, AddressOf OnEvent
    End Sub
    Private Sub OnEvent(ByRef [eType] As AvaxControlType, ByRef [ParentHandle] As Integer, ByRef [Index]
    As Integer, ByRef [sName] As String, ByRef [sTag] As String, ByRef [lStatus] As AvaxControlStatus, ByRef
    [KeyCode] As Integer, ByRef [Shift] As Integer)
        LastParentHandle = ParentHandle
        LastIndex = Index
        LastKeyCode = KeyCode
    End Sub
    Public Sub Dispose() Implements IDisposable.Dispose
        RemoveHandler source.AvaxControlKeyUp, AddressOf OnEvent
    End Sub
```

```
End Class
```

AvaxControlList

Purpose

Supplies the entries for a combo box. Replace sList with a one-dimensional Array whose logical entries start at index 1. Element 0, when present, is unused. Entries retain their order and may contain Unicode text.

VB.NET declaration

```
Event AvaxControlList(ByRef eType As AvaxControlType, ByRef ParentHandle As Integer, ByRef Index As
    Integer, ByRef sName As String, ByRef sTag As String, ByRef lStatus As AvaxControlStatus, ByRef sList As
    System.Array)
```

Parameters

Name	Type	Meaning
eType	AvaxControlType	The type of the embedded control that caused this callback.
ParentHandle	Integer	Integer identity of the containing drawing object or application-assigned transient parent.
Index	Integer	Child control index within the parent. It is separate from a one-based registry slot; ordinary inline documents start their child indices at 0.
sName	String	Name assigned to the control in its drawing data.
sTag	String	Application tag assigned to the control.
lStatus	AvaxControlStatus	AvaxControlStatus snapshot for this callback.
sList	System.Array	One-dimensional Array of list entries at logical indices 1 through N; slot 0 is unused when present.

Example

```
Option Strict On
Imports System
Imports Avax

Public NotInheritable Class Help_AvaxControlList
    Implements IDisposable
    Private ReadOnly source As cAvax
    Public Property LastParentHandle As Integer
    Public Property LastIndex As Integer
    Public Sub New(cad As cAvax)
        If cad Is Nothing Then Throw New ArgumentNullException(NameOf(cad))
        source = cad
        AddHandler source.AvaxControlList, AddressOf OnEvent
    End Sub
    Private Sub OnEvent(ByRef [eType] As AvaxControlType, ByRef [ParentHandle] As Integer, ByRef [Index]
    As Integer, ByRef [sName] As String, ByRef [sTag] As String, ByRef [lStatus] As AvaxControlStatus, ByRef
    [sList] As System.Array)
        LastParentHandle = ParentHandle
        LastIndex = Index
        sList = New String() {Nothing, "First", "Δεύτερο"}
    End Sub
    Public Sub Dispose() Implements IDisposable.Dispose
        RemoveHandler source.AvaxControlList, AddressOf OnEvent
    End Sub
End Class
```

AvaxControlLostFocus

Purpose

Reports exit from an embedded editor. Use the drawing identity to commit application state or update the host interface.

VB.NET declaration

```
Event AvaxControlLostFocus(ByRef eType As AvaxControlType, ByRef ParentHandle As Integer, ByRef Index As
    Integer, ByRef sName As String, ByRef sTag As String, ByRef lStatus As AvaxControlStatus)
```

Parameters

Name	Type	Meaning
eType	AvaxControlType	The type of the embedded control that caused this callback.
ParentHandle	Integer	Integer identity of the containing drawing object or application-assigned transient parent.
Index	Integer	Child control index within the parent. It is separate from a one-based registry slot; ordinary inline documents start their child indices at 0.
sName	String	Name assigned to the control in its drawing data.
sTag	String	Application tag assigned to the control.
lStatus	AvaxControlStatus	AvaxControlStatus snapshot for this callback.

Example

```
Option Strict On
Imports System
Imports Avax

Public NotInheritable Class Help_AvaxControlLostFocus
    Implements IDisposable
    Private ReadOnly source As cAvax
    Public Property LastParentHandle As Integer
    Public Property LastIndex As Integer
    Public Property LastName As String
    Public Sub New(cad As cAvax)
        If cad Is Nothing Then Throw New ArgumentNullException(NameOf(cad))
        source = cad
        AddHandler source.AvaxControlLostFocus, AddressOf OnEvent
    End Sub
    Private Sub OnEvent(ByRef [eType] As AvaxControlType, ByRef [ParentHandle] As Integer, ByRef [Index]
    As Integer, ByRef [sName] As String, ByRef [sTag] As String, ByRef [lStatus] As AvaxControlStatus)
        LastParentHandle = ParentHandle
        LastIndex = Index
        LastName = sName
    End Sub
    Public Sub Dispose() Implements IDisposable.Dispose
        RemoveHandler source.AvaxControlLostFocus, AddressOf OnEvent
    End Sub
End Class
```

AvaxControlMouseDown

Purpose

Reports a mouse-button press over an embedded control. X and Y are Double pixel coordinates relative to the drawing area, excluding its rulers and scroll bars. They are not world/drawing units. Use ScreenToWorld when drawing coordinates are needed.

VB.NET declaration

```
Event AvaxControlMouseDown(ByRef eType As AvaxControlType, ByRef ParentHandle As Integer, ByRef Index As
    Integer, ByRef sName As String, ByRef sTag As String, ByRef lStatus As AvaxControlStatus, ByRef Button As
    Integer, ByRef Shift As Integer, ByRef X As Double, ByRef Y As Double)
```

Parameters

Name	Type	Meaning
eType	AvaxControlType	The type of the embedded control that caused this callback.
ParentHandle	Integer	Integer identity of the containing drawing object or application-assigned transient parent.
Index	Integer	Child control index within the parent. It is separate from a one-based registry slot; ordinary inline documents start their child indices at 0.
sName	String	Name assigned to the control in its drawing data.
sTag	String	Application tag assigned to the control.
lStatus	AvaxControlStatus	AvaxControlStatus snapshot for this callback.
Button	Integer	Integer button mask: left=1, right=2, middle=4; bits may be combined.
Shift	Integer	Integer modifier mask: Shift=1, Ctrl=2, Alt=4. Keyboard events use the modifiers of that key message.
X	Double	Double X pixel coordinate relative to the drawing area, not world units.
Y	Double	Double Y pixel coordinate relative to the drawing area, not world units.

Example

```
Option Strict On
Imports System
Imports Avax

Public NotInheritable Class Help_AvaxControlMouseDown
    Implements IDisposable
    Private ReadOnly source As cAvax
    Public Property LastParentHandle As Integer
    Public Property LastIndex As Integer
    Public Property LastX As Double
    Public Property LastY As Double
    Public Property LastButtons As Integer
    Public Sub New(cad As cAvax)
        If cad Is Nothing Then Throw New ArgumentNullException(NameOf(cad))
        source = cad
        AddHandler source.AvaxControlMouseDown, AddressOf OnEvent
    End Sub
    Private Sub OnEvent(ByRef [eType] As AvaxControlType, ByRef [ParentHandle] As Integer, ByRef [Index]
    As Integer, ByRef [sName] As String, ByRef [sTag] As String, ByRef [lStatus] As AvaxControlStatus, ByRef
    [Button] As Integer, ByRef [Shift] As Integer, ByRef [X] As Double, ByRef [Y] As Double)
        LastParentHandle = ParentHandle
        LastIndex = Index
        LastX = X : LastY = Y : LastButtons = Button
    End Sub
    Public Sub Dispose() Implements IDisposable.Dispose
        RemoveHandler source.AvaxControlMouseDown, AddressOf OnEvent
    End Sub
End Class
```

AvaxControlMouseMove

Purpose

Reports pointer movement over an embedded control. X and Y use the same drawing-area pixel coordinate system as the main control mouse events, including when zoom, origin orientation or rulers change.

VB.NET declaration

```
Event AvaxControlMouseMove(ByRef eType As AvaxControlType, ByRef ParentHandle As Integer, ByRef Index As
    Integer, ByRef sName As String, ByRef sTag As String, ByRef lStatus As AvaxControlStatus, ByRef Button As
    Integer, ByRef Shift As Integer, ByRef X As Double, ByRef Y As Double)
```

Parameters

Name	Type	Meaning
eType	AvaxControlType	The type of the embedded control that caused this callback.
ParentHandle	Integer	Integer identity of the containing drawing object or application-assigned transient parent.
Index	Integer	Child control index within the parent. It is separate from a one-based registry slot; ordinary inline documents start their child indices at 0.
sName	String	Name assigned to the control in its drawing data.
sTag	String	Application tag assigned to the control.
lStatus	AvaxControlStatus	AvaxControlStatus snapshot for this callback.
Button	Integer	Integer button mask: left=1, right=2, middle=4; bits may be combined.
Shift	Integer	Integer modifier mask: Shift=1, Ctrl=2, Alt=4. Keyboard events use the modifiers of that key message.
X	Double	Double X pixel coordinate relative to the drawing area, not world units.
Y	Double	Double Y pixel coordinate relative to the drawing area, not world units.

Example

```
Option Strict On
Imports System
Imports Avax

Public NotInheritable Class Help_AvaxControlMouseMove
    Implements IDisposable
    Private ReadOnly source As cAvax
    Public Property LastParentHandle As Integer
    Public Property LastIndex As Integer
    Public Property LastX As Double
    Public Property LastY As Double
    Public Property LastButtons As Integer
    Public Sub New(cad As cAvax)
        If cad Is Nothing Then Throw New ArgumentNullException(NameOf(cad))
        source = cad
        AddHandler source.AvaxControlMouseMove, AddressOf OnEvent
    End Sub
    Private Sub OnEvent(ByRef [eType] As AvaxControlType, ByRef [ParentHandle] As Integer, ByRef [Index]
    As Integer, ByRef [sName] As String, ByRef [sTag] As String, ByRef [lStatus] As AvaxControlStatus, ByRef
    [Button] As Integer, ByRef [Shift] As Integer, ByRef [X] As Double, ByRef [Y] As Double)
        LastParentHandle = ParentHandle
        LastIndex = Index
        LastX = X : LastY = Y : LastButtons = Button
    End Sub
    Public Sub Dispose() Implements IDisposable.Dispose
        RemoveHandler source.AvaxControlMouseMove, AddressOf OnEvent
    End Sub
End Class
```

AvaxControlMouseUp

Purpose

Reports a mouse-button release over an embedded control. Button and Shift are Integer bit masks; X and Y are Double drawing-area pixels.

VB.NET declaration

```
Event AvaxControlMouseUp(ByRef eType As AvaxControlType, ByRef ParentHandle As Integer, ByRef Index As
    Integer, ByRef sName As String, ByRef sTag As String, ByRef lStatus As AvaxControlStatus, ByRef Button As
    Integer, ByRef Shift As Integer, ByRef X As Double, ByRef Y As Double)
```

Parameters

Name	Type	Meaning
eType	AvaxControlType	The type of the embedded control that caused this callback.
ParentHandle	Integer	Integer identity of the containing drawing object or application-assigned transient parent.
Index	Integer	Child control index within the parent. It is separate from a one-based registry slot; ordinary inline documents start their child indices at 0.
sName	String	Name assigned to the control in its drawing data.
sTag	String	Application tag assigned to the control.
lStatus	AvaxControlStatus	AvaxControlStatus snapshot for this callback.
Button	Integer	Integer button mask: left=1, right=2, middle=4; bits may be combined.
Shift	Integer	Integer modifier mask: Shift=1, Ctrl=2, Alt=4. Keyboard events use the modifiers of that key message.
X	Double	Double X pixel coordinate relative to the drawing area, not world units.
Y	Double	Double Y pixel coordinate relative to the drawing area, not world units.

Example

```
Option Strict On
Imports System
Imports Avax

Public NotInheritable Class Help_AvaxControlMouseUp
    Implements IDisposable
    Private ReadOnly source As cAvax
    Public Property LastParentHandle As Integer
    Public Property LastIndex As Integer
    Public Property LastX As Double
    Public Property LastY As Double
    Public Property LastButtons As Integer
    Public Sub New(cad As cAvax)
        If cad Is Nothing Then Throw New ArgumentNullException(NameOf(cad))
        source = cad
        AddHandler source.AvaxControlMouseUp, AddressOf OnEvent
    End Sub
    Private Sub OnEvent(ByRef [eType] As AvaxControlType, ByRef [ParentHandle] As Integer, ByRef [Index]
    As Integer, ByRef [sName] As String, ByRef [sTag] As String, ByRef [lStatus] As AvaxControlStatus, ByRef
    [Button] As Integer, ByRef [Shift] As Integer, ByRef [X] As Double, ByRef [Y] As Double)
        LastParentHandle = ParentHandle
        LastIndex = Index
        LastX = X : LastY = Y : LastButtons = Button
    End Sub
    Public Sub Dispose() Implements IDisposable.Dispose
        RemoveHandler source.AvaxControlMouseUp, AddressOf OnEvent
    End Sub
End Class
```

AvaxControlPictureFile

Purpose

Supplies the selected picture file for an embedded Button or PictureBox. Normal, highlighted, pressed and disabled states use their corresponding pattern fields; an empty state field falls back to the normal picture. Return a readable image filename in sPictureFile. The example keeps this path in a provider instance belonging to one drawing control. For a PictureBox, the initial filename is selected from picture, DownPicture, HighlightedPicture or DisablePicture according to the current state; an empty alternative falls back to picture. A returned replacement updates the selected state image and preserves the normal picture. Press/release and enable/disable transitions update the native image immediately.

VB.NET declaration

```
Event AvaxControlPictureFile(ByRef eType As AvaxControlType, ByRef ParentHandle As Integer, ByRef Index As
     Integer, ByRef sName As String, ByRef sTag As String, ByRef lStatus As AvaxControlStatus, ByRef
    sPictureFile As String)
```

Parameters

Name	Type	Meaning
eType	AvaxControlType	The type of the embedded control that caused this callback.
ParentHandle	Integer	Integer identity of the containing drawing object or application-assigned transient parent.
Index	Integer	Child control index within the parent. It is separate from a one-based registry slot; ordinary inline documents start their child indices at 0.
sName	String	Name assigned to the control in its drawing data.
sTag	String	Application tag assigned to the control.
lStatus	AvaxControlStatus	AvaxControlStatus snapshot for this callback.
sPictureFile	String	Readable filename of the image to display in the picture control.

Example

```
Option Strict On
Imports System
Imports Avax

Public NotInheritable Class Help_AvaxControlPictureFile
    Implements IDisposable
    Private ReadOnly source As cAvax
    Public Property LastParentHandle As Integer
    Public Property LastIndex As Integer
    Private ReadOnly picturePath As String
    Public Sub New(cad As cAvax, imagePath As String)
        If cad Is Nothing Then Throw New ArgumentNullException(NameOf(cad))
        source = cad
        picturePath = imagePath
        AddHandler source.AvaxControlPictureFile, AddressOf OnEvent
    End Sub
    Private Sub OnEvent(ByRef [eType] As AvaxControlType, ByRef [ParentHandle] As Integer, ByRef [Index]
    As Integer, ByRef [sName] As String, ByRef [sTag] As String, ByRef [lStatus] As AvaxControlStatus, ByRef
    [sPictureFile] As String)
        LastParentHandle = ParentHandle
        LastIndex = Index
        sPictureFile = picturePath
    End Sub
    Public Sub Dispose() Implements IDisposable.Dispose
        RemoveHandler source.AvaxControlPictureFile, AddressOf OnEvent
    End Sub
End Class
```

AvaxControlText

Purpose

Supplies the displayed text of an embedded text field or combo box. Replace sNewText with application data; iDecimals supplies its decimal-setting context. Refresh applies the returned text while suppressing an internal change-notification loop.

VB.NET declaration

```
Event AvaxControlText(ByRef eType As AvaxControlType, ByRef ParentHandle As Integer, ByRef Index As
    Integer, ByRef sName As String, ByRef sTag As String, ByRef lStatus As AvaxControlStatus, ByRef iDecimals
    As Integer, ByRef sNewText As String)
```

Parameters

Name	Type	Meaning
eType	AvaxControlType	The type of the embedded control that caused this callback.
ParentHandle	Integer	Integer identity of the containing drawing object or application-assigned transient parent.
Index	Integer	Child control index within the parent. It is separate from a one-based registry slot; ordinary inline documents start their child indices at 0.
sName	String	Name assigned to the control in its drawing data.
sTag	String	Application tag assigned to the control.
lStatus	AvaxControlStatus	AvaxControlStatus snapshot for this callback.
iDecimals	Integer	Integer decimal-setting context for the field. Use it when implementing application-specific numeric validation or display formatting.
sNewText	String	Text supplied by the control or data request. Replace it with the value your application wants the editor to use.

Example

```
Option Strict On
Imports System
Imports Avax

Public NotInheritable Class Help_AvaxControlText
    Implements IDisposable
    Private ReadOnly source As cAvax
    Public Property LastParentHandle As Integer
    Public Property LastIndex As Integer
    Public Sub New(cad As cAvax)
        If cad Is Nothing Then Throw New ArgumentNullException(NameOf(cad))
        source = cad
        AddHandler source.AvaxControlText, AddressOf OnEvent
    End Sub
    Private Sub OnEvent(ByRef [eType] As AvaxControlType, ByRef [ParentHandle] As Integer, ByRef [Index]
    As Integer, ByRef [sName] As String, ByRef [sTag] As String, ByRef [lStatus] As AvaxControlStatus, ByRef
    [iDecimals] As Integer, ByRef [sNewText] As String)
        LastParentHandle = ParentHandle
        LastIndex = Index
        sNewText = "Ready"
    End Sub
    Public Sub Dispose() Implements IDisposable.Dispose
        RemoveHandler source.AvaxControlText, AddressOf OnEvent
    End Sub
End Class
```

AvaxControlTooltip

Purpose

Supplies the tooltip text associated with an embedded control. Replace sTooltipText with the help text to show for that control.

VB.NET declaration

```
Event AvaxControlTooltip(ByRef eType As AvaxControlType, ByRef ParentHandle As Integer, ByRef Index As
    Integer, ByRef sName As String, ByRef sTag As String, ByRef lStatus As AvaxControlStatus, ByRef
    sTooltipText As String)
```

Parameters

Name	Type	Meaning
eType	AvaxControlType	The type of the embedded control that caused this callback.
ParentHandle	Integer	Integer identity of the containing drawing object or application-assigned transient parent.
Index	Integer	Child control index within the parent. It is separate from a one-based registry slot; ordinary inline documents start their child indices at 0.
sName	String	Name assigned to the control in its drawing data.
sTag	String	Application tag assigned to the control.
lStatus	AvaxControlStatus	AvaxControlStatus snapshot for this callback.
sTooltipText	String	Tooltip String supplied by the host.

Example

```
Option Strict On
Imports System
Imports Avax

Public NotInheritable Class Help_AvaxControlTooltip
    Implements IDisposable
    Private ReadOnly source As cAvax
    Public Property LastParentHandle As Integer
    Public Property LastIndex As Integer
    Public Sub New(cad As cAvax)
        If cad Is Nothing Then Throw New ArgumentNullException(NameOf(cad))
        source = cad
        AddHandler source.AvaxControlTooltip, AddressOf OnEvent
    End Sub
    Private Sub OnEvent(ByRef [eType] As AvaxControlType, ByRef [ParentHandle] As Integer, ByRef [Index]
    As Integer, ByRef [sName] As String, ByRef [sTag] As String, ByRef [lStatus] As AvaxControlStatus, ByRef
    [sTooltipText] As String)
        LastParentHandle = ParentHandle
        LastIndex = Index
        sTooltipText = "Help for " & sName
    End Sub
    Public Sub Dispose() Implements IDisposable.Dispose
        RemoveHandler source.AvaxControlTooltip, AddressOf OnEvent
    End Sub
End Class
```

AvaxControlUnErased

Purpose

Reports restoration of a parent containing embedded controls, for example by Undo. Query the parent again to obtain its current child controls.

VB.NET declaration

```
Event AvaxControlUnErased(ByRef ParentHandle As Integer)
```

Parameters

Name	Type	Meaning
ParentHandle	Integer	Integer identity of the containing drawing object or application-assigned transient parent.

Example

```
Option Strict On
```

```
Imports System
Imports Avax

Public NotInheritable Class Help_AvaxControlUnErased
    Implements IDisposable
    Private ReadOnly source As cAvax
    Public Property LastParentHandle As Integer
    Public Sub New(cad As cAvax)
        If cad Is Nothing Then Throw New ArgumentNullException(NameOf(cad))
        source = cad
        AddHandler source.AvaxControlUnErased, AddressOf OnEvent
    End Sub
    Private Sub OnEvent(ByRef [ParentHandle] As Integer)
        LastParentHandle = ParentHandle
    End Sub
    Public Sub Dispose() Implements IDisposable.Dispose
        RemoveHandler source.AvaxControlUnErased, AddressOf OnEvent
    End Sub
End Class
```

AvaxControlValidate

Purpose

Provides the text when an embedded editor validates, including a combo-box selection. Inspect iDecimals and replace sNewText with the accepted or normalized value. This legacy event has no Cancel parameter; applications that reject a value can return the previous accepted text.

VB.NET declaration

```
Event AvaxControlValidate(ByRef eType As AvaxControlType, ByRef ParentHandle As Integer, ByRef Index As
    Integer, ByRef sName As String, ByRef sTag As String, ByRef lStatus As AvaxControlStatus, ByRef iDecimals
    As Integer, ByRef sNewText As String)
```

Parameters

Name	Type	Meaning
eType	AvaxControlType	The type of the embedded control that caused this callback.
ParentHandle	Integer	Integer identity of the containing drawing object or application-assigned transient parent.
Index	Integer	Child control index within the parent. It is separate from a one-based registry slot; ordinary inline documents start their child indices at 0.
sName	String	Name assigned to the control in its drawing data.
sTag	String	Application tag assigned to the control.
lStatus	AvaxControlStatus	AvaxControlStatus snapshot for this callback.
iDecimals	Integer	Integer decimal-setting context for the field. Use it when implementing application-specific numeric validation or display formatting.
sNewText	String	Text supplied by the control or data request. Replace it with the value your application wants the editor to use.

Example

```
Option Strict On
Imports System
Imports Avax

Public NotInheritable Class Help_AvaxControlValidate
    Implements IDisposable
    Private ReadOnly source As cAvax
    Public Property LastParentHandle As Integer
    Public Property LastIndex As Integer
```

```
    Public Property LastText As String
    Public Sub New(cad As cAvax)
        If cad Is Nothing Then Throw New ArgumentNullException(NameOf(cad))
        source = cad
        AddHandler source.AvaxControlValidate, AddressOf OnEvent
    End Sub
    Private Sub OnEvent(ByRef [eType] As AvaxControlType, ByRef [ParentHandle] As Integer, ByRef [Index]
    As Integer, ByRef [sName] As String, ByRef [sTag] As String, ByRef [lStatus] As AvaxControlStatus, ByRef
    [iDecimals] As Integer, ByRef [sNewText] As String)
        LastParentHandle = ParentHandle
        LastIndex = Index
        sNewText = If(sNewText, "").Trim()
        LastText = sNewText
    End Sub
    Public Sub Dispose() Implements IDisposable.Dispose
        RemoveHandler source.AvaxControlValidate, AddressOf OnEvent
    End Sub
End Class
```

AvaxControlValue

Purpose

Supplies the current value for a checkbox or option control. Zero means unchecked; a nonzero Double value means checked. Return the application state in dValue.

VB.NET declaration

```
Event AvaxControlValue(ByRef eType As AvaxControlType, ByRef ParentHandle As Integer, ByRef Index As
    Integer, ByRef sName As String, ByRef sTag As String, ByRef lStatus As AvaxControlStatus, ByRef dValue As
    Double)
```

Parameters

Name	Type	Meaning
eType	AvaxControlType	The type of the embedded control that caused this callback.
ParentHandle	Integer	Integer identity of the containing drawing object or application-assigned transient parent.
Index	Integer	Child control index within the parent. It is separate from a one-based registry slot; ordinary inline documents start their child indices at 0.
sName	String	Name assigned to the control in its drawing data.
sTag	String	Application tag assigned to the control.
lStatus	AvaxControlStatus	AvaxControlStatus snapshot for this callback.
dValue	Double	Double checkbox/option value: 0 is unchecked, nonzero is checked.

Example

```
Option Strict On
Imports System
Imports Avax

Public NotInheritable Class Help_AvaxControlValue
    Implements IDisposable
    Private ReadOnly source As cAvax
    Public Property LastParentHandle As Integer
    Public Property LastIndex As Integer
    Public Sub New(cad As cAvax)
        If cad Is Nothing Then Throw New ArgumentNullException(NameOf(cad))
        source = cad
        AddHandler source.AvaxControlValue, AddressOf OnEvent
    End Sub
    Private Sub OnEvent(ByRef [eType] As AvaxControlType, ByRef [ParentHandle] As Integer, ByRef [Index]
```

```
    As Integer, ByRef [sName] As String, ByRef [sTag] As String, ByRef [lStatus] As AvaxControlStatus, ByRef
    [dValue] As Double)
        LastParentHandle = ParentHandle
        LastIndex = Index
        dValue = 1.0
    End Sub
    Public Sub Dispose() Implements IDisposable.Dispose
        RemoveHandler source.AvaxControlValue, AddressOf OnEvent
    End Sub
End Class
```

BeforeClick

cAvax event

Purpose

Occurs when the user presses and then releases a mouse button in the drawing

VB.NET declaration

```
Event BeforeClick(ByVal Sender As Object, ByVal e As cAvax.BeforeClickEventArgs)
```

Parameters

Name	Type and passing	Meaning
Sender	ByVal Sender As Object	Is the object that raises the event
e	ByVal e As BeforeClickEventArgs	Event data. The fields are described below; writable cancellation or output fields are read by the control when the handler returns.

Use the declaration for the overload you call. ByRef arguments can receive outputs; declare variables and arrays with the exact indicated type. Optional defaults are shown in the declaration.

Event data

Member	Declaration	Meaning
e.Button	Button As Integer	Mouse-button bits: left=1, right=2, middle=4.
e.ClickType	ClickType As AvaxClickType	AvaxClickType describing the pending drawing operation.
e.fCancel	fCancel As Boolean	Set True to reject this drawing pick.
e.Shift	Shift As Integer	Modifier bits: Shift=1, Ctrl=2, Alt=4.
e.x	x As Integer	Client X coordinate in pixels; Integer.
e.y	y As Integer	Client Y coordinate in pixels; Integer.

An event has no return value. Handle it with a procedure whose arguments match the declaration. Where a cancellation flag or output array is documented, assign it on the supplied event data object.

Remarks

Occurs before a drawing pick is accepted. ClickType identifies the operation: Point_ct for a point request, LineVertex_ct for construction vertices, SelectWindowCorner_ct for a selection window, ZoomWindowCorner_ct for zoom, Select_ct for object selection, and None_ct when no operation applies. Coordinates are client pixels. Set fCancel=True to reject the pick, or replace x/y to change its location. If the handler pauses or ends the control, replaces the drawing or starts another command, the superseded pick is abandoned. PauseAvaxEvents suppresses the notification.

Numeric changes from v2

Update the following variables, arrays or event-data receivers when porting the corresponding v2 call. The declarations above show the current overloads.

Parameter or value	v2 type	v3 type
e.Button	Short	Integer
e.Shift	Short	Integer

For ByRef parameters, use a variable with the exact v3 type. Preserve array dimensions and the documented indexing. Moving from v2 to v3 explains conversion and legacy file-storage rules.

Example

Setup

Add an Avax.cAvax Windows Forms control named cAvax1. Paste the procedures into the form class. Example setup and supporting files explains references, initialization and file paths.

VB.NET example

```
Private Sub Form1_Load(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs) Handles
    MyBase.Load
    cAvax1.StartAvax()
    Call cAvax1.Add_Line(1, 1, 0, 10, 10, 0)
End Sub
Private Sub Form1_FormClosed(ByVal eventSender As System.Object, ByVal eventArgs As
    System.Windows.Forms.FormClosedEventArgs) Handles Me.FormClosed
    cAvax1.EndAvax()
End Sub
Private Sub CAvax1_BeforeClick(ByVal Sender As Object, ByVal e As Avax.cAvax.BeforeClickEventArgs) Handles
     CAvax1.BeforeClick
    Dim sStr As String = String.Empty
    Dim sMsg As String = String.Empty
    Dim h() As Integer = Nothing
    Select Case e.ClickType
        Case Avax.AvaxClickType.LineVertex ct
        sStr = "to give a line vertex"
        Case Avax.AvaxClickType.None ct
        sStr = "for no reason"
        Case Avax.AvaxClickType.Point ct
        sStr = "make a point"
        Case Avax.AvaxClickType.Select ct
        sStr = "select an object"
        Case Avax.AvaxClickType.SelectWindowCorner ct
        sStr = "select a window corner"
        Case Avax.AvaxClickType.ZoomWindowCorner ct
        sStr = "select a zoom window corner"
    End Select
    sMsg = "You have clicked on the drawing area to " & sStr & vbCrLf & vbCrLf
    If CAvax1.GetHandlesUnderSelectCursor(h) <> 0 Then
        sMsg = sMsg & "Under the select cursor is an object." & vbCrLf
        sMsg = sMsg & "Do you want to select this object?" & vbCrLf
        If MsgBox(sMsg, MsgBoxStyle.YesNo Or MsgBoxStyle.Question, "") = MsgBoxResult.No Then
            e.fCancel = True
        End If
    Else
        sMsg = sMsg & "There is nothing under the select cursor." & vbCrLf
        MsgBox(sMsg, MsgBoxStyle.Information, "")
    End If
End Sub
```

Using event handlers

BeforeCopy

cAvax event

Purpose

This event occurs before some objects are copied to a relative new location

VB.NET declaration

```
Event BeforeCopy(ByVal Sender As Object, ByVal e As cAvax.BeforeCopyEventArgs)
```

Parameters

Name	Type and passing	Meaning
Sender	ByVal Sender As Object	Is the object that raises the event
e	ByVal e As BeforeCopyEventArgs	Event data. The fields are described below; writable cancellation or output fields are read by the control when the handler returns.

Use the declaration for the overload you call. ByRef arguments can receive outputs; declare variables and arrays with the exact indicated type. Optional defaults are shown in the declaration.

Event data

Member	Declaration	Meaning
e.cDx	cDx As Double	Is the relative x new location
e.cDy	cDy As Double	Is the relative y new location
e.fCancel	fCancel As Boolean	This event trigger can be canceled by changing this parameter to true
e.lHandles	lHandles As Integer()	The handles of the objects to be copy

An event has no return value. Handle it with a procedure whose arguments match the declaration. Where a cancellation flag or output array is documented, assign it on the supplied event data object.

Remarks

Occurs before CopyItems creates copies, including the Copy command, direct DoAvaxAction copies and each destination cell of an Array command. cDx and cDy are the proposed world-coordinate displacements and may be changed. In an array, fCancel skips the current cell; CancelArray stops the remaining cells. Earlier committed cells remain available to subsequent callbacks and are retained if a later cell fails. The array groups its history and publishes its final creation/change notifications at completion. The sender is the control performing the edit. lHandles is a one-based handle array; element zero is unused. Handlers may replace lHandles or edit the operation parameters. Duplicate target handles are applied once. All subscribers receive the same writable event object in subscription order; the final fCancel value determines cancellation. Event input arrays are detached from the method caller's arrays. A cancelled or invalid edit changes no target geometry. PauseAvaxEvents suppresses the notification while allowing the programmatic operation. Stopped, disabled or disposed controls perform no edit and raise no event. A handler that replaces or clears the drawing, ends the control or starts a different command prevents the pending operation from using stale input. Exceptions raised by a handler propagate from direct method and SetCoordinate calls; independent edits already made by the handler remain in the drawing. PauseAvax and LockEdit restrict interactive editing; direct editing methods remain available.

Numeric changes from v2

Update the following variables, arrays or event-data receivers when porting the corresponding v2 call. The declarations above show the current overloads.

Parameter or value	v2 type	v3 type
e.cDx	Single	Double

Parameter or value	v2 type	v3 type
e.cDy	Single	Double

For ByRef parameters, use a variable with the exact v3 type. Preserve array dimensions and the documented indexing. Moving from v2 to v3 explains conversion and legacy file-storage rules.

Example

Setup

Add an Avax.cAvax Windows Forms control named `cAvax1`. Paste the procedures into the form class. Example setup and supporting files explains references, initialization and file paths.

Add these Windows Forms controls or components to the form. Fields referenced by Handles clauses must be declared WithEvents, as the Visual Studio designer normally does.

Name	Type
Command1	Button

VB.NET example

```
Private Sub Command1_Click(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs)
    Handles Command1.Click
    CAvax1.Command = Avax.AvaxCommand.SelectAll_c
    CAvax1.Command = Avax.AvaxCommand.Copy_c
End Sub
Private Sub Form1_Load(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs) Handles
    MyBase.Load
    cAvax1.StartAvax()
    Call cAvax1.AddLine(0, 0, 0, 12, 0, 0)
    Call cAvax1.AddLine(1, 1, 0, 12, 0, 0)
End Sub
Private Sub Form1_FormClosed(ByVal eventSender As System.Object, ByVal eventArgs As
    System.Windows.Forms.FormClosedEventArgs) Handles Me.FormClosed
    cAvax1.EndAvax()
End Sub
Private Sub CAvax1_BeforeCopy(ByVal Sender As Object, ByVal e As Avax.cAvax.BeforeCopyEventArgs) Handles
    CAvax1.BeforeCopy
    If MsgBox("Do you want to copy the objects", MsgBoxStyle.Question Or MsgBoxStyle.YesNo, "Event
    BeforeCopy") = MsgBoxResult.No Then
        e.fCancel = True
    End If
End Sub
```

Using event handlers

BeforeMirror

cAvax event

Purpose

This event occurs before some objects are mirror copied by a specific mirror line

VB.NET declaration

```
Event BeforeMirror(ByVal Sender As Object, ByVal e As cAvax.BeforeMirrorEventArgs)
```

Parameters

Name	Type and passing	Meaning
Sender	ByVal Sender As Object	Is the object that raises the event
e	ByVal e As BeforeMirrorEventArgs	Event data. The fields are described below; writable cancellation or output fields are read by the control when the handler returns.

Use the declaration for the overload you call. ByRef arguments can receive outputs; declare variables and arrays with the exact indicated type. Optional defaults are shown in the declaration.

Event data

Member	Declaration	Meaning
e.fCancel	fCancel As Boolean	A Boolean expression that if it is True then this event is canceled
e.lHandles	lHandles As Integer()	The handles of the objects to be mirror copy
e.mirX1	mirX1 As Double	Is the x coordinate of the first point of the mirror line
e.mirX2	mirX2 As Double	Is the x coordinate of the second point of the mirror line
e.mirY1	mirY1 As Double	Is the y coordinate of the first point of the mirror line
e.mirY2	mirY2 As Double	Is the y coordinate of the second point of the mirror line

An event has no return value. Handle it with a procedure whose arguments match the declaration. Where a cancellation flag or output array is documented, assign it on the supplied event data object.

Remarks

Occurs before MirrorItems, the Mirror command or direct DoAvaxAction mirroring reflects objects. mirX1, mirY1, mirX2 and mirY2 define the world-coordinate mirror axis. Handlers may change either endpoint. Coincident endpoints do not define a valid mirror axis. The selected operation determines whether the result replaces the originals or creates copies. The sender is the control performing the edit. lHandles is a one-based handle array; element zero is unused. Handlers may replace lHandles or edit the operation parameters. Duplicate target handles are applied once. All subscribers receive the same writable event object in subscription order; the final fCancel value determines cancellation. Event input arrays are detached from the method caller's arrays. A cancelled or invalid edit changes no target geometry. PauseAvaxEvents suppresses the notification while allowing the programmatic operation. Stopped, disabled or disposed controls perform no edit and raise no event. A handler that replaces or clears the drawing, ends the control or starts a different command prevents the pending operation from using stale input. Exceptions raised by a handler propagate from direct method and SetCoordinate calls; independent edits already made by the handler remain in the drawing. PauseAvax and LockEdit restrict interactive editing; direct editing methods remain available.

Numeric changes from v2

Update the following variables, arrays or event-data receivers when porting the corresponding v2 call. The declarations above show the current overloads.

Parameter or value	v2 type	v3 type
e.mirX1	Single	Double
e.mirX2	Single	Double
e.mirY1	Single	Double
e.mirY2	Single	Double

For ByRef parameters, use a variable with the exact v3 type. Preserve array dimensions and the documented indexing. Moving from v2 to v3 explains conversion and legacy file-storage rules.

Example

Setup

Add an Avax.cAvax Windows Forms control named `cAvax1`. Paste the procedures into the form class. Example setup and supporting files explains references, initialization and file paths.

Add these Windows Forms controls or components to the form. Fields referenced by Handles clauses must be declared WithEvents, as the Visual Studio designer normally does.

Name	Type
Command1	Button

VB.NET example

```
Private Sub Command1_Click(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs)
    Handles Command1.Click
    CAvax1.Command = Avax.AvaxCommand.Select_All_c
    CAvax1.Command = Avax.AvaxCommand.Mirror_c
End Sub
Private Sub Form1_Load(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs) Handles
    MyBase.Load
    cAvax1.StartAvax()
    Call cAvax1.Add_Line(0, 0, 0, 12, 0, 0)
    Call cAvax1.Add_Line(1, 1, 0, 12, 0, 0)
End Sub
Private Sub Form1_FormClosed(ByVal eventSender As System.Object, ByVal eventArgs As
    System.Windows.Forms.FormClosedEventArgs) Handles Me.FormClosed
    cAvax1.EndAvax()
End Sub
Private Sub cAvax1_BeforeMirror(ByVal Sender As Object, ByVal e As Avax.cAvax.BeforeMirrorEventArgs)
    Handles CAvax1.BeforeMirror
    If MsgBox("Do you want to mirror copy the objects", MsgBoxStyle.Question Or MsgBoxStyle.YesNo, "Event
    BeforeCopy") = MsgBoxResult.No Then
        e.fCancel = True
    End If
End Sub
```

Using event handlers

BeforeMove

cAvax event

Purpose

This event occurs before some objects are moved to a relative new location

VB.NET declaration

```
Event BeforeMove(ByVal Sender As Object, ByVal e As cAvax.BeforeMoveEventArgs)
```

Parameters

Name	Type and passing	Meaning
Sender	ByVal Sender As Object	Is the object that raises the event
e	ByVal e As BeforeMoveEventArgs	Event data. The fields are described below; writable cancellation or output fields are read by the control when the handler returns.

Use the declaration for the overload you call. ByRef arguments can receive outputs; declare variables and arrays with the exact indicated type. Optional defaults are shown in the declaration.

Event data

Member	Declaration	Meaning
e.fCancel	fCancel As Boolean	A Boolean expression that if it is True then this event is canceled
e.lHandles	lHandles As Integer()	The handles of the objects to be move
e.mDx	mDx As Double	the relative x new location
e.mDy	mDy As Double	the relative y new location

An event has no return value. Handle it with a procedure whose arguments match the declaration. Where a cancellation flag or output array is documented, assign it on the supplied event data object.

Remarks

Occurs before MoveItems or the Move command translates objects, before the translation branch of StretchItems and the selection region, and before AlignItem or whole-object MoveNode translation. Vertex-only MoveNode replacements do not raise this event. mDx and mDy are the proposed world-coordinate displacements and may be changed. AlignItem and MoveNode also abandon their prepared edit when the handler changes the underlying drawing, preserving that independent change. The sender is the control performing the edit. lHandles is a one-based handle array; element zero is unused. Handlers may replace lHandles or edit the operation parameters. Duplicate target handles are applied once. All subscribers receive the same writable event object in subscription order; the final fCancel value determines cancellation. Event input arrays are detached from the method caller's arrays. A cancelled or invalid edit changes no target geometry. PauseAvaxEvents suppresses the notification while allowing the programmatic operation. Stopped, disabled or disposed controls perform no edit and raise no event. A handler that replaces or clears the drawing, ends the control or starts a different command prevents the pending operation from using stale input. Exceptions raised by a handler propagate from direct method and SetCoordinate calls; independent edits already made by the handler remain in the drawing. PauseAvax and LockEdit restrict interactive editing; direct editing methods remain available.

Numeric changes from v2

Update the following variables, arrays or event-data receivers when porting the corresponding v2 call. The declarations above show the current overloads.

Parameter or value	v2 type	v3 type
e.mDx	Single	Double

Parameter or value	v2 type	v3 type
e.mDy	Single	Double

For ByRef parameters, use a variable with the exact v3 type. Preserve array dimensions and the documented indexing. Moving from v2 to v3 explains conversion and legacy file-storage rules.

Example

Setup

Add an Avax.cAvax Windows Forms control named `cAvax1`. Paste the procedures into the form class. Example setup and supporting files explains references, initialization and file paths.

Add these Windows Forms controls or components to the form. Fields referenced by Handles clauses must be declared WithEvents, as the Visual Studio designer normally does.

Name	Type
Command1	Button

VB.NET example

```
Private Sub Command1_Click(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs)
    Handles Command1.Click
    CAvax1.Command = Avax.AvaxCommand.Select_All_c
    CAvax1.Command = Avax.AvaxCommand.Move_c
End Sub
Private Sub Form1_Load(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs) Handles
    MyBase.Load
    cAvax1.StartAvax()
    Call cAvax1.Add_Line(0, 0, 0, 12, 0, 0)
    Call cAvax1.Add_Line(1, 1, 0, 12, 0, 0)
End Sub
Private Sub Form1_FormClosed(ByVal eventSender As System.Object, ByVal eventArgs As
    System.Windows.Forms.FormClosedEventArgs) Handles Me.FormClosed
    cAvax1.EndAvax()
End Sub
Private Sub CAvax1_BeforeMove(ByVal Sender As Object, ByVal e As Avax.cAvax.BeforeMoveEventArgs) Handles
    CAvax1.BeforeMove
    If MsgBox("Do you want to move the objects", MsgBoxStyle.Question Or MsgBoxStyle.YesNo, "Event
    BeforeCopy") = MsgBoxResult.No Then
        e.fCancel = True
    End If
End Sub
```

Using event handlers

BeforeRotate

cAvax event

Purpose

This event occurs before some objects are rotated with a rotation center point and a rotation angle

VB.NET declaration

```
Event BeforeRotate(ByVal Sender As Object, ByVal e As cAvax.BeforeRotateEventArgs)
```

Parameters

Name	Type and passing	Meaning
Sender	ByVal Sender As Object	Is the object that raises the event
e	ByVal e As BeforeRotateEventArgs	Event data. The fields are described below; writable cancellation or output fields are read by the control when the handler returns.

Use the declaration for the overload you call. ByRef arguments can receive outputs; declare variables and arrays with the exact indicated type. Optional defaults are shown in the declaration.

Event data

Member	Declaration	Meaning
e.fCancel	fCancel As Boolean	A Boolean expression that if it is True then this event is canceled
e.lHandles	lHandles As Integer()	The handles of the objects to be rotate
e.RotAngle	RotAngle As Double	Is the rotation angle
e.rx	rx As Double	Is the coordinate x of the rotation center point
e.ry	ry As Double	Is the coordinate y of the rotation center point

An event has no return value. Handle it with a procedure whose arguments match the declaration. Where a cancellation flag or output array is documented, assign it on the supplied event data object.

Remarks

Occurs before RotateItems, the Rotate command, direct DoAvaxAction rotation or the selection region rotation handle applies a rotation. RotAngle is in degrees; rx and ry identify the world-coordinate rotation center. Handlers may change the center and angle. The selected operation determines whether the result replaces the originals or creates copies. The sender is the control performing the edit. lHandles is a one-based handle array; element zero is unused. Handlers may replace lHandles or edit the operation parameters. Duplicate target handles are applied once. All subscribers receive the same writable event object in subscription order; the final fCancel value determines cancellation. Event input arrays are detached from the method caller's arrays. A cancelled or invalid edit changes no target geometry. PauseAvaxEvents suppresses the notification while allowing the programmatic operation. Stopped, disabled or disposed controls perform no edit and raise no event. A handler that replaces or clears the drawing, ends the control or starts a different command prevents the pending operation from using stale input. Exceptions raised by a handler propagate from direct method and SetCoordinate calls; independent edits already made by the handler remain in the drawing. PauseAvax and LockEdit restrict interactive editing; direct editing methods remain available.

Numeric changes from v2

Update the following variables, arrays or event-data receivers when porting the corresponding v2 call. The declarations above show the current overloads.

Parameter or value	v2 type	v3 type
e.RotAngle	Single	Double

Parameter or value	v2 type	v3 type
e.rx	Single	Double
e.ry	Single	Double

For ByRef parameters, use a variable with the exact v3 type. Preserve array dimensions and the documented indexing. Moving from v2 to v3 explains conversion and legacy file-storage rules.

Example

Setup

Add an Avax.cAvax Windows Forms control named `cAvax1`. Paste the procedures into the form class. Example setup and supporting files explains references, initialization and file paths.

Add these Windows Forms controls or components to the form. Fields referenced by Handles clauses must be declared WithEvents, as the Visual Studio designer normally does.

Name	Type
Command1	Button

VB.NET example

```
Private Sub Command1_Click(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs)
    Handles Command1.Click
    CAvax1.Command = Avax.AvaxCommand.Select_All_c
    CAvax1.Command = Avax.AvaxCommand.Rotate_c
End Sub
Private Sub Form1_Load(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs) Handles
    MyBase.Load
    cAvax1.StartAvax()
    Call cAvax1.Add_Line(0, 0, 0, 12, 0, 0)
    Call cAvax1.Add_Line(1, 1, 0, 12, 0, 0)
End Sub
Private Sub Form1_FormClosed(ByVal eventSender As System.Object, ByVal eventArgs As
    System.Windows.Forms.FormClosedEventArgs) Handles Me.FormClosed
    cAvax1.EndAvax()
End Sub
Private Sub CAvax1_BeforeRotate(ByVal Sender As Object, ByVal e As Avax.cAvax.BeforeRotateEventArgs)
    Handles CAvax1.BeforeRotate
    If MsgBox("Do you want to rotate the objects", MsgBoxStyle.Question Or MsgBoxStyle.YesNo, "Event
    BeforeCopy") = MsgBoxResult.No Then
        e.fCancel = True
    End If
End Sub
```

Using event handlers

BeforeScale

cAvax event

Purpose

This event occurs before some objects are scaled with a scale base point and a scale magnifying factor

VB.NET declaration

```
Event BeforeScale(ByVal Sender As Object, ByVal e As cAvax.BeforeScaleEventArgs)
```

Parameters

Name	Type and passing	Meaning
Sender	ByVal Sender As Object	Is the object that raises the event
e	ByVal e As BeforeScaleEventArgs	Event data. The fields are described below; writable cancellation or output fields are read by the control when the handler returns.

Use the declaration for the overload you call. ByRef arguments can receive outputs; declare variables and arrays with the exact indicated type. Optional defaults are shown in the declaration.

Event data

Member	Declaration	Meaning
e.fCancel	fCancel As Boolean	A Boolean expression that if it is True then this event is canceled
e.lHandles	lHandles As Integer()	The handles of the objects to be scaled
e.ScaleX	ScaleX As Double	Is the scale magnifying factor for the x coordinate
e.ScaleY	ScaleY As Double	Is the scale magnifying factor for the y coordinate
e.sx	sx As Double	Is the coordinate x of the scale base point
e.sY	sY As Double	Is the coordinate y of the scale base point

An event has no return value. Handle it with a procedure whose arguments match the declaration. Where a cancellation flag or output array is documented, assign it on the supplied event data object.

Remarks

Occurs before ScaleItems, the Scale command or direct DoAvaxAction scaling applies scale factors. ScaleX and ScaleY are the proposed factors; sx and sY identify the world-coordinate scale origin. Handlers may change all four values. A zero input axis factor is normalized from the other axis before the event; a pair of zero factors is rejected. The selected operation determines whether the result replaces the originals or creates copies. The sender is the control performing the edit. lHandles is a one-based handle array; element zero is unused. Handlers may replace lHandles or edit the operation parameters. Duplicate target handles are applied once. All subscribers receive the same writable event object in subscription order; the final fCancel value determines cancellation. Event input arrays are detached from the method caller's arrays. A cancelled or invalid edit changes no target geometry. PauseAvaxEvents suppresses the notification while allowing the programmatic operation. Stopped, disabled or disposed controls perform no edit and raise no event. A handler that replaces or clears the drawing, ends the control or starts a different command prevents the pending operation from using stale input. Exceptions raised by a handler propagate from direct method and SetCoordinate calls; independent edits already made by the handler remain in the drawing. PauseAvax and LockEdit restrict interactive editing; direct editing methods remain available.

Numeric changes from v2

Update the following variables, arrays or event-data receivers when porting the corresponding v2 call. The declarations above show the current overloads.

Parameter or value	v2 type	v3 type
e.ScaleX	Single	Double
e.ScaleY	Single	Double
e.sx	Single	Double
e.sY	Single	Double

For ByRef parameters, use a variable with the exact v3 type. Preserve array dimensions and the documented indexing. Moving from v2 to v3 explains conversion and legacy file-storage rules.

Example

Setup

Add an Avax.cAvax Windows Forms control named `cAvax1`. Paste the procedures into the form class. Example setup and supporting files explains references, initialization and file paths.

Add these Windows Forms controls or components to the form. Fields referenced by Handles clauses must be declared WithEvents, as the Visual Studio designer normally does.

Name	Type
Command1	Button

VB.NET example

```
Private Sub Command1_Click(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs)
    Handles Command1.Click
    CAvax1.Command = Avax.AvaxCommand.Select_All_c
    CAvax1.Command = Avax.AvaxCommand.Scale_c
End Sub
Private Sub Form1_Load(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs) Handles
    MyBase.Load
    cAvax1.StartAvax()
    Call cAvax1.Add_Line(0, 0, 0, 12, 0, 0)
    Call cAvax1.Add_Line(1, 1, 0, 12, 0, 0)
End Sub
Private Sub Form1_FormClosed(ByVal eventSender As System.Object, ByVal eventArgs As
    System.Windows.Forms.FormClosedEventArgs) Handles Me.FormClosed
    cAvax1.EndAvax()
End Sub
Private Sub CAvax1_BeforeScale(ByVal Sender As Object, ByVal e As Avax.cAvax.BeforeScaleEventArgs) Handles
     CAvax1.BeforeScale
    If MsgBox("Do you want to scale the objects", MsgBoxStyle.Question Or MsgBoxStyle.YesNo, "Event
    BeforeCopy") = MsgBoxResult.No Then
        e.fCancel = True
    End If
End Sub
```

Using event handlers

BeforeTransform

cAvax event

Purpose

This event occurs before some objects are transformed based on a 3x3 transformation array

VB.NET declaration

```
Event BeforeTransform(ByVal Sender As Object, ByVal e As cAvax.BeforeTransformEventArgs)
```

Parameters

Name	Type and passing	Meaning
Sender	ByVal Sender As Object	Is the object that raises the event
e	ByVal e As BeforeTransformEventArgs	Event data. The fields are described below; writable cancellation or output fields are read by the control when the handler returns.

Use the declaration for the overload you call. ByRef arguments can receive outputs; declare variables and arrays with the exact indicated type. Optional defaults are shown in the declaration.

Event data

Member	Declaration	Meaning
e.fCancel	fCancel As Boolean	A Boolean expression that if it is True then this event is canceled
e.lHandles	lHandles As Integer()	The handles of the objects to be transformed
e.TransArray3x3	TransArray3x3 As Double(,)	A 3x3 transformation array of Singles

An event has no return value. Handle it with a procedure whose arguments match the declaration. Where a cancellation flag or output array is documented, assign it on the supplied event data object.

Remarks

Occurs before TransformItems or direct DoAvaxAction applies an affine matrix, and before the scaling branch of StretchItems or the selection region applies its matrix. TransArray3x3 uses indices 1 through 3 in both dimensions of a Double(3,3) array; element zero in either dimension is unused. Handlers may edit or replace the matrix. A missing, undersized, nonfinite or otherwise unusable result is rejected before mutation. The supplied operation determines whether the result replaces the originals or creates copies. The sender is the control performing the edit. lHandles is a one-based handle array; element zero is unused. Handlers may replace lHandles or edit the operation parameters. Duplicate target handles are applied once. All subscribers receive the same writable event object in subscription order; the final fCancel value determines cancellation. Event input arrays are detached from the method caller's arrays. A cancelled or invalid edit changes no target geometry. PauseAvaxEvents suppresses the notification while allowing the programmatic operation. Stopped, disabled or disposed controls perform no edit and raise no event. A handler that replaces or clears the drawing, ends the control or starts a different command prevents the pending operation from using stale input. Exceptions raised by a handler propagate from direct method and SetCoordinate calls; independent edits already made by the handler remain in the drawing. PauseAvax and LockEdit restrict interactive editing; direct editing methods remain available.

Numeric changes from v2

Update the following variables, arrays or event-data receivers when porting the corresponding v2 call. The declarations above show the current overloads.

Parameter or value	v2 type	v3 type
e.TransArray3x3	Single(,)	Double(,)

For ByRef parameters, use a variable with the exact v3 type. Preserve array dimensions and the documented indexing. Moving from v2 to v3 explains conversion and legacy file-storage rules.

Example

Setup

Add an Avax.cAvax Windows Forms control named cAvax1. Paste the procedures into the form class. Example setup and supporting files explains references, initialization and file paths.

Add these Windows Forms controls or components to the form. Fields referenced by Handles clauses must be declared WithEvents, as the Visual Studio designer normally does.

Name	Type
Command1	Button

VB.NET example

```
Private Sub Command1_Click(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs)
    Handles Command1.Click
    Dim h() As Integer = Nothing
    Dim TransArr33(3, 3) As Double
    TransArr33(1, 1) = 1
    TransArr33(2, 2) = 1
    TransArr33(3, 3) = 1
    If cAvax1.GetAllHandlesArr(h) > 0 Then
        TransArr33(3, 1) = 5
        TransArr33(3, 2) = 3
        Call cAvax1.TransformItems(h, TransArr33)
    End If
End Sub
Private Sub Form1_Load(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs) Handles
    MyBase.Load
    cAvax1.StartAvax()
    Call cAvax1.Add_Line(0, 0, 0, 12, 0, 0)
    Call cAvax1.Add_Line(1, 1, 0, 12, 0, 0)
End Sub
Private Sub Form1_FormClosed(ByVal eventSender As System.Object, ByVal eventArgs As
    System.Windows.Forms.FormClosedEventArgs) Handles Me.FormClosed
    cAvax1.EndAvax()
End Sub
Private Sub CAvax1_BeforeTransform(ByVal Sender As Object, ByVal e As Avax.cAvax.BeforeTransformEventArgs)
     Handles CAvax1.BeforeTransform
    If MsgBox("Do you want to transform the objects", MsgBoxStyle.Question Or MsgBoxStyle.YesNo, "Event
    BeforeCopy") = MsgBoxResult.No Then
        e.fCancel = True
    End If
End Sub
```

Using event handlers

Change

cAvax event

Purpose

Occurs when a change is made in the drawing. That can be the creation of a new object, the deletion of an object, etc.

VB.NET declaration

```
Event Change()
```

An event has no return value. Handle it with a procedure whose arguments match the declaration. Where a cancellation flag or output array is documented, assign it on the supplied event data object.

Remarks

Occurs after a drawing edit or drawing-resource change is committed. The handler can inspect the changed drawing and its complete Undo state, start another command or make another edit. Model Undo/Redo report Change; a pure view change does not. Client-handler exceptions propagate after the completed edit without replaying its notification. Ordinary programmatic primitives and CopyItems publish Change without NewUserItems. Filled-shape helpers and Add_ShapeDocument normally announce NewUserItems before Change; Add_Document publishes Change before its explicit creation notification, as in R2. BeginUpdate and EndUpdate suspend presentation; they do not defer or combine normal API Change notifications. PauseAvaxEvents suppresses notifications without replay on resume. GetAvaxEvents can flush pending external changes after active construction finishes; an idle call does not invent changes. A preceding creation handler that stops/disables/disposes the control, replaces its document or changes event-pause state prevents a stale trailing Change.

Example

Setup

Add an Avax.cAvax Windows Forms control named `cAvax1`. Paste the procedures into the form class. Example setup and supporting files explains references, initialization and file paths.

Add these Windows Forms controls or components to the form. Fields referenced by Handles clauses must be declared WithEvents, as the Visual Studio designer normally does.

Name	Type
Command1	Button

VB.NET example

```
Private Sub Form1_Load(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs) Handles
    MyBase.Load
    cAvax1.StartAvax()
End Sub
Private Sub Form1_FormClosed(ByVal eventSender As System.Object, ByVal eventArgs As
    System.Windows.Forms.FormClosedEventArgs) Handles Me.FormClosed
    cAvax1.EndAvax()
End Sub
Private Sub Command1_Click(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs)
    Handles Command1.Click
    Call cAvax1.Add_Line(0, 0, 0, 12, 0, 0)
End Sub
Private Sub cAvax1_Change() Handles CAvax1.Change
    MsgBox("Drawing has changed.", MsgBoxStyle.Information, "Event Change")
End Sub
```

Using event handlers

DblClick

cAvax event

Purpose

Occurs when the user presses and releases a mouse button and then presses and releases it again over AVAX

VB.NET declaration

```
Event DblClick(ByVal Sender As Object, ByVal e As System.EventArgs)
```

Parameters

Name	Type and passing	Meaning
Sender	ByVal Sender As Object	Is the object that raises the event
e	ByVal e As System.EventArgs	Is the object that contains the event data

Use the declaration for the overload you call. ByRef arguments can receive outputs; declare variables and arrays with the exact indicated type. Optional defaults are shown in the declaration.

An event has no return value. Handle it with a procedure whose arguments match the declaration. Where a cancellation flag or output array is documented, assign it on the supplied event data object.

Remarks

Occurs when Windows Forms reports a double-click on the drawing surface. Sender is the AVAX control and e is EventArgs. A double-click can also complete an active polyline. PauseAvaxEvents suppresses the notification without changing the drawing-command setting.

Example

Setup

Add an Avax.cAvax Windows Forms control named `cAvax1`. Paste the procedures into the form class. Example setup and supporting files explains references, initialization and file paths.

Add these Windows Forms controls or components to the form. Fields referenced by Handles clauses must be declared WithEvents, as the Visual Studio designer normally does.

Name	Type
Label1	Label

VB.NET example

```
Private Sub Form1_Load(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs) Handles
    MyBase.Load
    cAvax1.StartAvax()
    Label1.Text = "Try a double click on Avax window"
End Sub
Private Sub Form1_FormClosed(ByVal eventSender As System.Object, ByVal eventArgs As
    System.Windows.Forms.FormClosedEventArgs) Handles Me.FormClosed
    cAvax1.EndAvax()
End Sub
Private Sub cAvax1_DblClick(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs)
    Handles cAvax1.DblClick
    MsgBox("DblClick on Avax window", MsgBoxStyle.Information, "Event DblClick")
End Sub
```

Using event handlers

DrawingRepainted

cAvax event

Purpose

This event occurs when the current drawing in the window is repainted and returns the handles of the objects that have been repainted

VB.NET declaration

```
Event DrawingRepainted(ByVal Sender As Object, ByVal e As cAvax.DrawingRepaintedEventArgs)
```

Parameters

Name	Type and passing	Meaning
Sender	ByVal Sender As Object	Is the object that raises the event
e	ByVal e As DrawingRepaintedEventArgs	Event data. The fields are described below; writable cancellation or output fields are read by the control when the handler returns.

Use the declaration for the overload you call. ByRef arguments can receive outputs; declare variables and arrays with the exact indicated type. Optional defaults are shown in the declaration.

Event data

Member	Declaration	Meaning
e.pHandles	pHandles As Integer()	One-based Integer array of visible top-level handles for this frame; element 0 is unused. This is not the current selection.

An event has no return value. Handle it with a procedure whose arguments match the declaration. Where a cancellation flag or output array is documented, assign it on the supplied event data object.

Remarks

Reports the top-level drawing objects visible in the frame being painted. Read e.pHandles as a one-based Integer array: element 0 is unused. Hidden layers and off-screen objects are excluded. This is a frame snapshot, not the selection array; modifying it does not change object identities or the next frame. The event precedes OnPaint and AfterPaint. PauseAvaxEvents suppresses these notifications.

Example

Setup

Add an Avax.cAvax Windows Forms control named `cAvax1`. Paste the procedures into the form class. Example setup and supporting files explains references, initialization and file paths.

Add these Windows Forms controls or components to the form. Fields referenced by Handles clauses must be declared WithEvents, as the Visual Studio designer normally does.

Name	Type
Command1	Button

VB.NET example

```
Private Sub CAvax1_DrawingRepainted(ByVal Sender As Object, ByVal e As
    Avax.cAvax.DrawingRepaintedEventArgs) Handles CAvax1.DrawingRepainted
    Dim sMsg As String = String.Empty
    Dim i As Integer
    sMsg = "Handles of the Objects that was Repainted:" & vbCrLf
    For i = 1 To UBound(e.pHandles)
```

```
        sMsg = sMsg & Str(e.pHandles(i)) & ", "
    Next i
    If Microsoft.VisualBasic.Strings.Right(sMsg, 2) = ", " Then
        sMsg = Mid(sMsg, 1, Len(sMsg) - 2)
    End If
    MsgBox(sMsg, MsgBoxStyle.Information, "Event DrawingRepainted")
End Sub
Private Sub Command1_Click(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs)
    Handles Command1.Click
    CAvax1.Command = Avax.AvaxCommand.Redraw_c
End Sub
Private Sub Form1_Load(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs) Handles
    MyBase.Load
    cAvax1.StartAvax()
    cAvax1.PauseAvaxEvents = True
    Call cAvax1.Add_Line(0, 0, 0, 10, 10, 0)
    Call cAvax1.Add_Line(0, 0, 0, 0, 10, 0)
    cAvax1.PauseAvaxEvents = False
End Sub
Private Sub Form1_FormClosed(ByVal eventSender As System.Object, ByVal eventArgs As
    System.Windows.Forms.FormClosedEventArgs) Handles Me.FormClosed
    cAvax1.EndAvax()
End Sub
```

Using event handlers

EnterFocus

cAvax event

Purpose

Occurs when focus enters the object AVAX

VB.NET declaration

```
Event EnterFocus(ByVal Sender As Object, ByVal e As System.EventArgs)
```

Parameters

Name	Type and passing	Meaning
Sender	ByVal Sender As Object	Is the object that raises the event
e	ByVal e As System.EventArgs	Is the object that contains the event data

Use the declaration for the overload you call. ByRef arguments can receive outputs; declare variables and arrays with the exact indicated type. Optional defaults are shown in the declaration.

An event has no return value. Handle it with a procedure whose arguments match the declaration. Where a cancellation flag or output array is documented, assign it on the supplied event data object.

Remarks

Occurs when keyboard focus enters the drawing surface. Sender is the AVAX control and e is EventArgs. Calling Focus on a visible, enabled AVAX control transfers focus into its drawing area. The example moves focus from a form button to AVAX. PauseAvaxEvents suppresses this notification.

Example

Setup

Add an Avax.cAvax Windows Forms control named `cAvax1`. Paste the procedures into the form class. Example setup and supporting files explains references, initialization and file paths.

Add these Windows Forms controls or components to the form. Fields referenced by Handles clauses must be declared WithEvents, as the Visual Studio designer normally does.

Name	Type
Command1	Button

VB.NET example

```
Private Sub cAvax1_EnterFocus(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs)
    Handles cAvax1.EnterFocus
    MsgBox("EnterFocus...", MsgBoxStyle.Information, "Event EnterFocus")
End Sub
Private Sub Command1_Click(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs)
    Handles Command1.Click
    cAvax1.Focus()
End Sub
Private Sub Form1_Load(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs) Handles
    MyBase.Load
    cAvax1.StartAvax()
End Sub
Private Sub Form1_FormClosed(ByVal eventSender As System.Object, ByVal eventArgs As
    System.Windows.Forms.FormClosedEventArgs) Handles Me.FormClosed
    cAvax1.EndAvax()
End Sub
```

Using event handlers

ExitFocus

cAvax event

Purpose

Occurs when focus leaves the object AVAX

VB.NET declaration

```
Event ExitFocus(ByVal Sender As Object, ByVal e As System.EventArgs)
```

Parameters

Name	Type and passing	Meaning
Sender	ByVal Sender As Object	Is the object that raises the event
e	ByVal e As System.EventArgs	Is the object that contains the event data

Use the declaration for the overload you call. ByRef arguments can receive outputs; declare variables and arrays with the exact indicated type. Optional defaults are shown in the declaration.

An event has no return value. Handle it with a procedure whose arguments match the declaration. Where a cancellation flag or output array is documented, assign it on the supplied event data object.

Remarks

Occurs when keyboard focus leaves the drawing surface, for example when the user selects another form control. Sender is the AVAX control and e is EventArgs. PauseAvaxEvents suppresses this notification.

Example

Setup

Add an Avax.cAvax Windows Forms control named `cAvax1`. Paste the procedures into the form class. Example setup and supporting files explains references, initialization and file paths.

Add these Windows Forms controls or components to the form. Fields referenced by Handles clauses must be declared WithEvents, as the Visual Studio designer normally does.

Name	Type
Command1	Button

VB.NET example

```
Private Sub FrmExitFocus_Load(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs)
    Handles MyBase.Load
    cAvax1.StartAvax()
End Sub
Private Sub FrmExitFocus_FormClosed(ByVal eventSender As System.Object, ByVal eventArgs As
    System.Windows.Forms.FormClosedEventArgs) Handles Me.FormClosed
    cAvax1.EndAvax()
End Sub
Private Sub cAvax1_ExitFocus(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs)
    Handles cAvax1.ExitFocus
    MsgBox("ExitFocus", MsgBoxStyle.Information, "")
End Sub
Private Sub Command1_Click(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs)
    Handles Command1.Click
End Sub
```

Using event handlers

ImportWarningsAvailable

API reference

ImportWarningsAvailable

Purpose

Notifies the host application that a successful DXF/DWG import or placement preview produced warnings. Read LastImportWarnings to obtain their text. The control leaves presentation of those messages to the application.

Declaration

```
Public Event ImportWarningsAvailable As System.EventHandler
```

Event arguments

sender is the AVAX control. e is EventArgs.Empty; the messages are read from LastImportWarnings. Notifications require a started, enabled control and are suppressed by PauseAvaxEvents. If a drawing-change handler has already installed another warning context, the superseded import does not announce its old warnings. Exceptions from this event handler propagate to the calling import or command.

VB.NET example

```
Option Strict On
Public Module ImportWarningNotificationExample
    Public Sub Subscribe(cad As Avax.cAvax)
        AddHandler cad.ImportWarningsAvailable,
            Sub(sender As Object, e As System.EventArgs)
                For Each message As String In cad.LastImportWarnings
                    System.Diagnostics.Debug.WriteLine(message)
                Next
            End Sub
    End Sub
End Module
```

MouseIn

cAvax event

Purpose

Occurs when the mouse pointer is going in the drawing area

VB.NET declaration

```
Event MouseIn()
```

An event has no return value. Handle it with a procedure whose arguments match the declaration. Where a cancellation flag or output array is documented, assign it on the supplied event data object.

Remarks

Occurs when the pointer enters the drawing surface. This event has no arguments. The example changes BackColor to white; its MouseOut handler changes it to red. PauseAvaxEvents suppresses these notifications.

Example

Setup

Add an Avax.cAvax Windows Forms control named `cAvax1`. Paste the procedures into the form class. Example setup and supporting files explains references, initialization and file paths.

VB.NET example

```
Private Sub Form1_Load(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs) Handles
    MyBase.Load
    cAvax1.StartAvax()
End Sub
Private Sub Form1_FormClosed(ByVal eventSender As System.Object, ByVal eventArgs As
    System.Windows.Forms.FormClosedEventArgs) Handles Me.FormClosed
    cAvax1.EndAvax()
End Sub
Private Sub cAvax1_MouseIn() Handles CAvax1.MouseIn
    CAvax1.BackColor = System.Drawing.Color.White
End Sub
Private Sub cAvax1_MouseOut() Handles CAvax1.MouseOut
    CAvax1.BackColor = System.Drawing.Color.Red
End Sub
```

Using event handlers

MouseOut

cAvax event

Purpose

Occurs when the mouse pointer is going out of the drawing area

VB.NET declaration

```
Event MouseOut()
```

An event has no return value. Handle it with a procedure whose arguments match the declaration. Where a cancellation flag or output array is documented, assign it on the supplied event data object.

Remarks

Occurs when the pointer leaves the drawing surface. This event has no arguments. A pending MouseStop notification is canceled on leaving. The example changes BackColor to red. PauseAvaxEvents suppresses the notification.

Example

Setup

Add an Avax.cAvax Windows Forms control named `cAvax1`. Paste the procedures into the form class. Example setup and supporting files explains references, initialization and file paths.

VB.NET example

```
Private Sub Form1_Load(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs) Handles
    MyBase.Load
    cAvax1.StartAvax()
End Sub
Private Sub Form1_FormClosed(ByVal eventSender As System.Object, ByVal eventArgs As
    System.Windows.Forms.FormClosedEventArgs) Handles Me.FormClosed
    cAvax1.EndAvax()
End Sub
Private Sub cAvax1_MouseIn() Handles CAvax1.MouseIn
    CAvax1.BackColor = System.Drawing.Color.White
End Sub
Private Sub cAvax1_MouseOut() Handles CAvax1.MouseOut
    CAvax1.BackColor = System.Drawing.Color.Red
End Sub
```

Using event handlers

MouseStop

cAvax event

Purpose

Occurs when the mouse pointer in the drawing area stops moving

VB.NET declaration

```
Event MouseStop()
```

An event has no return value. Handle it with a procedure whose arguments match the declaration. Where a cancellation flag or output array is documented, assign it on the supplied event data object.

Remarks

Occurs once when the pointer briefly stops moving inside the drawing surface. A new movement starts the inactivity interval again. Leaving the drawing cancels the pending notification; no MouseStop is sent for the previous movement outside the surface. This event has no arguments. PauseAvax and PauseAvaxEvents suppress it.

Example

Setup

Add an Avax.cAvax Windows Forms control named `cAvax1`. Paste the procedures into the form class. Example setup and supporting files explains references, initialization and file paths.

Add these Windows Forms controls or components to the form. Fields referenced by Handles clauses must be declared WithEvents, as the Visual Studio designer normally does.

Name	Type
Label1	Label

VB.NET example

```
Private Sub Form1_Load(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs) Handles
    MyBase.Load
    cAvax1.StartAvax()
End Sub
Private Sub Form1_FormClosed(ByVal eventSender As System.Object, ByVal eventArgs As
    System.Windows.Forms.FormClosedEventArgs) Handles Me.FormClosed
    cAvax1.EndAvax()
End Sub
Private Sub CAvax1_OnMouseMove(ByVal Sender As Object, ByVal e As Avax.cAvax.MouseMoveEventArgs) Handles
    CAvax1.OnMouseMove
    Label1.Text = "Mouse is moving..."
End Sub
Private Sub cAvax1_MouseStop() Handles CAvax1.MouseStop
    Label1.Text = "Mouse is not moving..."
End Sub
```

Using event handlers

NewUserItems

cAvax event

Purpose

Reports handles created by completed interactive actions or explicit creation helpers that announce their new objects.

VB.NET declaration

```
Event NewUserItems(ByVal Sender As Object, ByVal e As cAvax.NewUserItemsEventArgs)
```

Parameters

Name	Type and passing	Meaning
Sender	ByVal Sender As Object	Is the object that raises the event
e	ByVal e As NewUserItemsEventArgs	Event data. The fields are described below; writable cancellation or output fields are read by the control when the handler returns.

Use the declaration for the overload you call. ByRef arguments can receive outputs; declare variables and arrays with the exact indicated type. Optional defaults are shown in the declaration.

Event data

Member	Declaration	Meaning
e.NewHandles	NewHandles As Integer()	One-based Integer array of newly added object handles; element 0 is unused.

An event has no return value. Handle it with a procedure whose arguments match the declaration. Where a cancellation flag or output array is documented, assign it on the supplied event data object.

Remarks

NewHandles is a fresh one-based Integer() with index 0 unused. The complete objects and their reversible state exist before notification. Subscribers share the event-args object in subscription order; changing its array changes only the notification, not the drawing.

Ordinary programmatic primitives, dimension methods, transform copies, offset copies and ordinary file imports return their results through their own return values or output arguments. Explicit announcing shape, picture and document helpers can also raise this event. Add_Document announces after Change; Add_ShapeDocument and other filled-shape helpers announce before Change. Supported helpers accept fireNewItemsEvent=False. Add_ShapeText follows ordinary text creation.

PauseAvaxEvents suppresses notices without replay. Replacing or clearing the drawing, stopping/disabling the control or removing announced objects prevents stale notices. Application handler exceptions propagate after commit and do not replay a drained notice. A handler can start new commands or edit the drawing without the returning notification clearing that state.

Example

Setup

Add an Avax.cAvax Windows Forms control named `cAvax1`. Paste the procedures into the form class. Example setup and supporting files explains references, initialization and file paths.

Add these Windows Forms controls or components to the form. Fields referenced by Handles clauses must be declared WithEvents, as the Visual Studio designer normally does.

Name	Type
Command1	Button

VB.NET example

```
Private Sub Command1_Click(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs)
    Handles Command1.Click
    CAvax1.Command = Avax.AvaxCommand.Circle_c
End Sub
Private Sub Form1_Load(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs) Handles
    MyBase.Load
    Command1.Text = "Add a circle"
    cAvax1.StartAvax()
End Sub
Private Sub Form1_FormClosed(ByVal eventSender As System.Object, ByVal eventArgs As
    System.Windows.Forms.FormClosedEventArgs) Handles Me.FormClosed
    cAvax1.EndAvax()
End Sub
Private Sub CAvax1_NewUserItems(ByVal Sender As Object, ByVal e As Avax.cAvax.NewUserItemsEventArgs)
    Handles CAvax1.NewUserItems
    MsgBox("The end user has add a circle into the current drawing!!!", MsgBoxStyle.Information, "Event
    NewUserItems")
End Sub
```

Using event handlers

OnClick

cAvax event

Purpose

Occurs when the user presses and then releases a mouse button over AVAX

VB.NET declaration

```
Event OnClick(ByVal Sender As Object, ByVal e As System.EventArgs)
```

Parameters

Name	Type and passing	Meaning
Sender	ByVal Sender As Object	Is the object that raises the event
e	ByVal e As System.EventArgs	Is the object that contains the event data

Use the declaration for the overload you call. ByRef arguments can receive outputs; declare variables and arrays with the exact indicated type. Optional defaults are shown in the declaration.

An event has no return value. Handle it with a procedure whose arguments match the declaration. Where a cancellation flag or output array is documented, assign it on the supplied event data object.

Remarks

Occurs when Windows Forms reports a completed mouse click over the drawing surface, after a press and release. It is separate from OnMouseDown and from BeforeClick, which controls whether a drawing pick is accepted. Sender is the AVAX control and e is EventArgs. Notifications require a started, enabled control and are suppressed by PauseAvaxEvents or PauseAvax.

Example

Setup

Add an Avax.cAvax Windows Forms control named `cAvax1`. Paste the procedures into the form class. Example setup and supporting files explains references, initialization and file paths.

Add these Windows Forms controls or components to the form. Fields referenced by Handles clauses must be declared WithEvents, as the Visual Studio designer normally does.

Name	Type
Label1	Label

VB.NET example

```
Private Sub Form1_Load(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs) Handles
    MyBase.Load
    cAvax1.StartAvax()
    Label1.Text = "Click on the drawing area..."
End Sub
Private Sub Form1_FormClosed(ByVal eventSender As System.Object, ByVal eventArgs As
    System.Windows.Forms.FormClosedEventArgs) Handles Me.FormClosed
    cAvax1.EndAvax()
End Sub
Private Sub cAvax1_OnClick(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs)
    Handles CAvax1.OnClick
    MsgBox("OnClick", MsgBoxStyle.Information, "Event OnClick")
End Sub
```

Using event handlers

OnKeyDown

cAvax event

Purpose

Occurs when the user presses a key from the keyboard when AVAX has the focus

VB.NET declaration

```
Event OnKeyDown(ByVal Sender As Object, ByVal e As cAvax.KeyDownEventArgs)
```

Parameters

Name	Type and passing	Meaning
Sender	ByVal Sender As Object	Is the object that raises the event
e	ByVal e As KeyDownEventArgs	Event data. The fields are described below; writable cancellation or output fields are read by the control when the handler returns.

Use the declaration for the overload you call. ByRef arguments can receive outputs; declare variables and arrays with the exact indicated type. Optional defaults are shown in the declaration.

Event data

Member	Declaration	Meaning
e.KeyCode	KeyCode As Integer	Windows virtual-key code; Integer. Assign 0 to consume this key notification.
e.Shift	Shift As Integer	Modifier bits captured in the key message: Shift=1, Ctrl=2, Alt=4.

An event has no return value. Handle it with a procedure whose arguments match the declaration. Where a cancellation flag or output array is documented, assign it on the supplied event data object.

Remarks

Occurs when a key is pressed while the drawing surface has keyboard focus. KeyCode is the Windows virtual-key code as Integer. Shift is the modifier mask captured in that key message: 1=Shift, 2=Ctrl, 4=Alt; combine bits when several modifiers are held. Set KeyCode=0 to consume the key and suppress the control's default key handling. PauseAvaxEvents suppresses the notification.

Numeric changes from v2

Update the following variables, arrays or event-data receivers when porting the corresponding v2 call. The declarations above show the current overloads.

Parameter or value	v2 type	v3 type
e.KeyCode	Short	Integer
e.Shift	Short	Integer

For ByRef parameters, use a variable with the exact v3 type. Preserve array dimensions and the documented indexing. Moving from v2 to v3 explains conversion and legacy file-storage rules.

Example

Setup

Add an Avax.cAvax Windows Forms control named `cAvax1`. Paste the procedures into the form class. Example setup and supporting files explains references, initialization and file paths.

VB.NET example

```
Private Sub Form1_Load(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs) Handles
    MyBase.Load
    CAvax1.StartAvax()
End Sub
Private Sub Form1_FormClosed(ByVal eventSender As System.Object, ByVal eventArgs As
    System.Windows.Forms.FormClosedEventArgs) Handles Me.FormClosed
    cAvax1.EndAvax()
End Sub
Private Sub CAvax1_OnKeyDown(ByVal Sender As Object, ByVal e As Avax.cAvax.KeyDownEventArgs) Handles
    CAvax1.OnKeyDown
    MsgBox("Shift = " & e.Shift & vbCrLf & "KeyCode =" & e.KeyCode, MsgBoxStyle.Information,
    "cAvax1_OnKeyDown")
End Sub
```

Using event handlers

OnKeyPress

cAvax event

Purpose

Occurs when the user presses and releases a key from the keyboard when AVAX has the focus

VB.NET declaration

```
Event OnKeyPress(ByVal Sender As Object, ByVal e As cAvax.KeyPressEventArgs)
```

Parameters

Name	Type and passing	Meaning
Sender	ByVal Sender As Object	Is the object that raises the event
e	ByVal e As KeyPressEventArgs	Event data. The fields are described below; writable cancellation or output fields are read by the control when the handler returns.

Use the declaration for the overload you call. ByRef arguments can receive outputs; declare variables and arrays with the exact indicated type. Optional defaults are shown in the declaration.

Event data

Member	Declaration	Meaning
e.KeyAscii	KeyAscii As Integer	UTF-16 character code; Integer. Assign another code to replace the character or 0 to consume it.

An event has no return value. Handle it with a procedure whose arguments match the declaration. Where a cancellation flag or output array is documented, assign it on the supplied event data object.

Remarks

Occurs when a character is entered while the drawing surface has keyboard focus. Despite its historical name, KeyAscii is an Integer UTF-16 character code in v3, preserving Greek and other Unicode text. Assign another character code to replace the character, or assign 0 to consume it. Characters outside the Basic Multilingual Plane arrive as two UTF-16 surrogate code units. PauseAvaxEvents suppresses the notification.

Numeric changes from v2

Update the following variables, arrays or event-data receivers when porting the corresponding v2 call. The declarations above show the current overloads.

Parameter or value	v2 type	v3 type
e.KeyAscii	Short	Integer

For ByRef parameters, use a variable with the exact v3 type. Preserve array dimensions and the documented indexing. Moving from v2 to v3 explains conversion and legacy file-storage rules.

Example

Setup

Add an Avax.cAvax Windows Forms control named `cAvax1`. Paste the procedures into the form class. Example setup and supporting files explains references, initialization and file paths.

VB.NET example

```
Private Sub Form1_Load(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs) Handles
    MyBase.Load
    CAvax1.StartAvax()
```

```
End Sub
Private Sub Form1_FormClosed(ByVal eventSender As System.Object, ByVal eventArgs As
    System.Windows.Forms.FormClosedEventArgs) Handles Me.FormClosed
    cAvax1.EndAvax()
End Sub
Private Sub CAvax1_OnKeyPress(ByVal Sender As Object, ByVal e As Avax.cAvax.KeyPressEventArgs) Handles
    CAvax1.OnKeyPress
    MsgBox("KeyAscii = " & e.KeyAscii, MsgBoxStyle.Information, "cAvax1_OnKeyPress")
End Sub
```

Using event handlers

OnKeyUp

cAvax event

Purpose

Occurs when the user releases a key from the keyboard when AVAX has the focus

VB.NET declaration

```
Event OnKeyUp(ByVal Sender As Object, ByVal e As cAvax.KeyUpEventArgs)
```

Parameters

Name	Type and passing	Meaning
Sender	ByVal Sender As Object	Is the object that raises the event
e	ByVal e As KeyUpEventArgs	Event data. The fields are described below; writable cancellation or output fields are read by the control when the handler returns.

Use the declaration for the overload you call. ByRef arguments can receive outputs; declare variables and arrays with the exact indicated type. Optional defaults are shown in the declaration.

Event data

Member	Declaration	Meaning
e.KeyCode	KeyCode As Integer	Windows virtual-key code; Integer. Assign 0 to consume this key notification.
e.Shift	Shift As Integer	Modifier bits captured in the key message: Shift=1, Ctrl=2, Alt=4.

An event has no return value. Handle it with a procedure whose arguments match the declaration. Where a cancellation flag or output array is documented, assign it on the supplied event data object.

Remarks

Occurs when a key is released while the drawing surface has keyboard focus. KeyCode is the Windows virtual-key code as Integer. Shift uses the modifiers captured in that key message: 1=Shift, 2=Ctrl and 4=Alt. Set KeyCode=0 to mark the key-up message as handled. PauseAvaxEvents suppresses the notification.

Numeric changes from v2

Update the following variables, arrays or event-data receivers when porting the corresponding v2 call. The declarations above show the current overloads.

Parameter or value	v2 type	v3 type
e.KeyCode	Short	Integer
e.Shift	Short	Integer

For ByRef parameters, use a variable with the exact v3 type. Preserve array dimensions and the documented indexing. Moving from v2 to v3 explains conversion and legacy file-storage rules.

Example

Setup

Add an Avax.cAvax Windows Forms control named `cAvax1`. Paste the procedures into the form class. Example setup and supporting files explains references, initialization and file paths.

VB.NET example

```
Private Sub Form1_Load(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs) Handles
    MyBase.Load
    CAvax1.StartAvax()
End Sub
Private Sub Form1_FormClosed(ByVal eventSender As System.Object, ByVal eventArgs As
    System.Windows.Forms.FormClosedEventArgs) Handles Me.FormClosed
    cAvax1.EndAvax()
End Sub
Private Sub CAvax1_OnKeyUp(ByVal Sender As Object, ByVal e As Avax.cAvax.KeyUpEventArgs) Handles
    CAvax1.OnKeyUp
    MsgBox("Shift = " & e.Shift & vbCrLf & "KeyCode =" & e.KeyCode, MsgBoxStyle.Information,
    "cAvax1_OnKeyUp")
End Sub
```

Using event handlers

OnMouseDown

cAvax event

Purpose

Occur when the user presses a mouse button over AVAX

VB.NET declaration

```
Event OnMouseDown(ByVal Sender As Object, ByVal e As cAvax.MouseDownEventArgs)
```

Parameters

Name	Type and passing	Meaning
Sender	ByVal Sender As Object	Is the object that raises the event
e	ByVal e As MouseDownEventArgs	Event data. The fields are described below; writable cancellation or output fields are read by the control when the handler returns.

Use the declaration for the overload you call. ByRef arguments can receive outputs; declare variables and arrays with the exact indicated type. Optional defaults are shown in the declaration.

Event data

Member	Declaration	Meaning
e.Button	Button As Integer	Mouse-button bits: left=1, right=2, middle=4.
e.Shift	Shift As Integer	Modifier bits: Shift=1, Ctrl=2, Alt=4.
e.x	x As Double	Client X coordinate in pixels; Double.
e.y	y As Double	Client Y coordinate in pixels; Double.

An event has no return value. Handle it with a procedure whose arguments match the declaration. Where a cancellation flag or output array is documented, assign it on the supplied event data object.

Remarks

Occurs when a mouse button is pressed on the drawing surface. Button uses the legacy bit mask: 1=left, 2=right, 4=middle. Shift uses 1=Shift, 2=Ctrl and 4=Alt. x and y are Double client-pixel coordinates, not drawing coordinates. Sender is the AVAX control. PauseAvaxEvents suppresses this notification.

Numeric changes from v2

Update the following variables, arrays or event-data receivers when porting the corresponding v2 call. The declarations above show the current overloads.

Parameter or value	v2 type	v3 type
e.Button	Short	Integer
e.Shift	Short	Integer
e.x	Single	Double
e.y	Single	Double

For ByRef parameters, use a variable with the exact v3 type. Preserve array dimensions and the documented indexing. Moving from v2 to v3 explains conversion and legacy file-storage rules.

Example

Setup

Add an Avax.cAvax Windows Forms control named cAvax1. Paste the procedures into the form class. Example setup and supporting files explains references, initialization and file paths.

VB.NET example

```
Private Sub Form1_Load(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs) Handles
    MyBase.Load
    cAvax1.StartAvax()
End Sub
Private Sub Form1_FormClosed(ByVal eventSender As System.Object, ByVal eventArgs As
    System.Windows.Forms.FormClosedEventArgs) Handles Me.FormClosed
    cAvax1.EndAvax()
End Sub
Private Sub CAvax1_OnMouseDown(ByVal Sender As Object, ByVal e As Avax.cAvax.MouseDownEventArgs) Handles
    CAvax1.OnMouseDown
    Dim ii As Integer
    Select Case e.Button
        Case 1
        ii = MsgBox("You have pressed down the first mouse button !")
        Case 2
        ii = MsgBox("You have pressed down the second mouse button !")
    End Select
End Sub
```

Using event handlers

OnMouseMove

cAvax event

Purpose

Occurs when the user moves the mouse over AVAX

VB.NET declaration

```
Event OnMouseMove(ByVal Sender As Object, ByVal e As cAvax.MouseMoveEventArgs)
```

Parameters

Name	Type and passing	Meaning
Sender	ByVal Sender As Object	Is the object that raises the event
e	ByVal e As MouseMoveEventArgs	Event data. The fields are described below; writable cancellation or output fields are read by the control when the handler returns.

Use the declaration for the overload you call. ByRef arguments can receive outputs; declare variables and arrays with the exact indicated type. Optional defaults are shown in the declaration.

Event data

Member	Declaration	Meaning
e.Button	Button As Integer	Mouse-button bits: left=1, right=2, middle=4.
e.Shift	Shift As Integer	Modifier bits: Shift=1, Ctrl=2, Alt=4.
e.x	x As Double	Client X coordinate in pixels; Double.
e.y	y As Double	Client Y coordinate in pixels; Double.

An event has no return value. Handle it with a procedure whose arguments match the declaration. Where a cancellation flag or output array is documented, assign it on the supplied event data object.

Remarks

Occurs when the pointer moves over the drawing surface. Button and Shift contain the mouse-button and keyboard-modifier bit masks. x and y are Double client-pixel coordinates. Use CurrentXY when drawing coordinates are needed. The example displays a moving message and uses MouseStop to show when movement pauses. PauseAvaxEvents suppresses the notification.

Numeric changes from v2

Update the following variables, arrays or event-data receivers when porting the corresponding v2 call. The declarations above show the current overloads.

Parameter or value	v2 type	v3 type
e.Button	Short	Integer
e.Shift	Short	Integer
e.x	Single	Double
e.y	Single	Double

For ByRef parameters, use a variable with the exact v3 type. Preserve array dimensions and the documented indexing. Moving from v2 to v3 explains conversion and legacy file-storage rules.

Example

Setup

Add an Avax.cAvax Windows Forms control named `cAvax1`. Paste the procedures into the form class. Example setup and supporting files explains references, initialization and file paths.

Add these Windows Forms controls or components to the form. Fields referenced by Handles clauses must be declared WithEvents, as the Visual Studio designer normally does.

Name	Type
Label1	Label

VB.NET example

```
Private Sub Form1_Load(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs) Handles
    MyBase.Load
    cAvax1.StartAvax()
End Sub
Private Sub Form1_FormClosed(ByVal eventSender As System.Object, ByVal eventArgs As
    System.Windows.Forms.FormClosedEventArgs) Handles Me.FormClosed
    cAvax1.EndAvax()
End Sub
Private Sub CAvax1_MouseStop() Handles CAvax1.MouseStop
    Label1.Text = "Mouse is not moving..."
End Sub
Private Sub CAvax1_OnMouseMove(ByVal Sender As Object, ByVal e As Avax.cAvax.MouseMoveEventArgs) Handles
    CAvax1.OnMouseMove
    Label1.Text = "Mouse is moving..."
End Sub
```

Using event handlers

OnMouseUp

cAvax event

Purpose

Occur when the user releases a mouse button over AVAX

VB.NET declaration

```
Event OnMouseUp(ByVal Sender As Object, ByVal e As cAvax.MouseUpEventArgs)
```

Parameters

Name	Type and passing	Meaning
Sender	ByVal Sender As Object	Is the object that raises the event
e	ByVal e As MouseUpEventArgs	Event data. The fields are described below; writable cancellation or output fields are read by the control when the handler returns.

Use the declaration for the overload you call. ByRef arguments can receive outputs; declare variables and arrays with the exact indicated type. Optional defaults are shown in the declaration.

Event data

Member	Declaration	Meaning
e.Button	Button As Integer	Mouse-button bits: left=1, right=2, middle=4.
e.Shift	Shift As Integer	Modifier bits: Shift=1, Ctrl=2, Alt=4.
e.x	x As Double	Client X coordinate in pixels; Double.
e.y	y As Double	Client Y coordinate in pixels; Double.

An event has no return value. Handle it with a procedure whose arguments match the declaration. Where a cancellation flag or output array is documented, assign it on the supplied event data object.

Remarks

Occurs when a mouse button is released on the drawing surface. Button uses 1=left, 2=right and 4=middle; Shift uses 1=Shift, 2=Ctrl and 4=Alt. x and y are Double client-pixel coordinates. Sender is the AVAX control. PauseAvaxEvents suppresses the notification.

Numeric changes from v2

Update the following variables, arrays or event-data receivers when porting the corresponding v2 call. The declarations above show the current overloads.

Parameter or value	v2 type	v3 type
e.Button	Short	Integer
e.Shift	Short	Integer
e.x	Single	Double
e.y	Single	Double

For ByRef parameters, use a variable with the exact v3 type. Preserve array dimensions and the documented indexing. Moving from v2 to v3 explains conversion and legacy file-storage rules.

Example

Setup

Add an Avax.cAvax Windows Forms control named cAvax1. Paste the procedures into the form class. Example setup and supporting files explains references, initialization and file paths.

VB.NET example

```
Private Sub Form1_Load(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs) Handles
    MyBase.Load
    cAvax1.StartAvax()
End Sub
Private Sub Form1_FormClosed(ByVal eventSender As System.Object, ByVal eventArgs As
    System.Windows.Forms.FormClosedEventArgs) Handles Me.FormClosed
    cAvax1.EndAvax()
End Sub
Private Sub CAvax1_OnMouseUp(ByVal Sender As Object, ByVal e As Avax.cAvax.MouseUpEventArgs) Handles
    CAvax1.OnMouseUp
    Dim ii As Integer
    Select Case e.Button
        Case 1
        ii = MsgBox("You have released the first mouse button !")
        Case 2
        ii = MsgBox("You have released the second mouse button !")
    End Select
End Sub
```

Using event handlers

OnPaint

cAvax event

Purpose

This occurs when the AVAX Paint event fires and the drawing repaints

VB.NET declaration

```
Event OnPaint(ByVal Sender As Object, ByVal e As System.Windows.Forms.PaintEventArgs)
```

Parameters

Name	Type and passing	Meaning
Sender	ByVal Sender As Object	Is the object that raises the event
e	ByVal e As System.Windows.Forms.PaintEventArgs	Is the object that contains the event data

Use the declaration for the overload you call. ByRef arguments can receive outputs; declare variables and arrays with the exact indicated type. Optional defaults are shown in the declaration.

An event has no return value. Handle it with a procedure whose arguments match the declaration. Where a cancellation flag or output array is documented, assign it on the supplied event data object.

Remarks

Supplies a Windows Forms PaintEventArgs whose Graphics draws an overlay on the drawing canvas. ClipRectangle is expressed in pixels for this frame. DrawingRepainted is raised before OnPaint, and AfterPaint follows it. Paint using e.Graphics only during this callback; the control owns its lifetime. PauseAvaxEvents suppresses the callback.

Example

Setup

Add an Avax.cAvax Windows Forms control named `cAvax1`. Paste the procedures into the form class. Example setup and supporting files explains references, initialization and file paths.

VB.NET example

```
Private Sub Form1_Load(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs) Handles
    MyBase.Load
    cAvax1.StartAvax()
End Sub
Private Sub Form1_FormClosed(ByVal eventSender As System.Object, ByVal eventArgs As
    System.Windows.Forms.FormClosedEventArgs) Handles Me.FormClosed
    cAvax1.EndAvax()
End Sub
Private Sub CAvax1_OnPaint(ByVal Sender As Object, ByVal e As System.Windows.Forms.PaintEventArgs) Handles
     CAvax1.OnPaint
    MsgBox("OnPaint", MsgBoxStyle.Information, "Event OnPaint")
End Sub
```

Using event handlers

OnResize

cAvax event

Purpose

This occurs when the AVAX Resize event fires

VB.NET declaration

```
Event OnResize(ByVal Sender As Object, ByVal e As System.EventArgs)
```

Parameters

Name	Type and passing	Meaning
Sender	ByVal Sender As Object	Is the object that raises the event
e	ByVal e As System.EventArgs	Is the object that contains the event data

Use the declaration for the overload you call. ByRef arguments can receive outputs; declare variables and arrays with the exact indicated type. Optional defaults are shown in the declaration.

An event has no return value. Handle it with a procedure whose arguments match the declaration. Where a cancellation flag or output array is documented, assign it on the supplied event data object.

Remarks

Raised when a started control changes size, after updating its drawing viewport. Sender is the drawing control and e is an ordinary EventArgs. Before StartAvax and after EndAvax no OnResize notification is published. Started disabled controls retain the R2 resize notification. PauseAvaxEvents suppresses it. A drawing/lifecycle change or event-suppression cycle inside ZoomChange abandons the obsolete trailing resize event.

Example

Setup

Add an Avax.cAvax Windows Forms control named `cAvax1`. Paste the procedures into the form class. Example setup and supporting files explains references, initialization and file paths.

Add these Windows Forms controls or components to the form. Fields referenced by Handles clauses must be declared WithEvents, as the Visual Studio designer normally does.

Name	Type
Command1	Button

VB.NET example

```
Private Sub Form1_Load(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs) Handles
    MyBase.Load
    Command1.Text = "Resize Avax"
    cAvax1.StartAvax()
End Sub
Private Sub Form1_FormClosed(ByVal eventSender As System.Object, ByVal eventArgs As
    System.Windows.Forms.FormClosedEventArgs) Handles Me.FormClosed
    cAvax1.EndAvax()
End Sub
Private Sub CAvax1_OnResize(ByVal Sender As Object, ByVal e As System.EventArgs) Handles CAvax1.OnResize
    MsgBox("OnResize", MsgBoxStyle.Information, "Event OnResize")
End Sub
Private Sub Command1_Click(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs)
    Handles Command1.Click
    Call cAvax1.SetBounds(0, 0, Me.Width - 0.5 * Command1.Height, Me.Height - 3 * Command1.Height)
    Command1.Enabled = False
```

```
End Sub
```

Using event handlers

OperationFailed

API reference

Purpose

Notifies the application when a native operation reports an error. It also covers interactive commands whose Command assignment cannot return a success value. Use the event to display a message or record the original exception in the application's error log.

VB.NET declaration

```
Public Event OperationFailed As System.EventHandler(Of cAvax.OperationFailedEventArgs)
```

Event parameters

Parameter	Meaning
sender As Object	The control reporting the error.
e As OperationFailedEventArgs	Event data containing the original exception in `ErrorException`.

Behavior

Notification occurs synchronously when the operation reports its caught error. The event does not cancel, retry or repair that operation, and the failing method retains its own return-value or exception contract. A method that rejects an input without reporting an error need not raise this event. Built-in error notifications and the original exception's message can use different display languages; `ErrorException` preserves the original exception object.

Keep the handler brief and handle failures in the application's logging or display code. The reporting path does not catch subscriber exceptions around the event invocation. Subscribe while the application needs notifications and remove the same handler during cleanup. Subscriptions are runtime state and are not stored with a drawing.

Event data

The event uses `OperationFailedEventArgs`, a public, non-inheritable nested type derived from `System.EventArgs`. It has the following constructor and read-only property.

Constructor

```
Sub New(ByVal ex As System.Exception)
```

Creates event data containing `ex`. The constructor keeps the supplied exception reference; it does not copy or wrap the exception. It has no return value. Passing `Nothing` is accepted and produces an `ErrorException` value of `Nothing`.

Exception property

```
ReadOnly Property ErrorException As System.Exception
```

Returns the exception supplied to the constructor. Read its `Message`, type, inner exception or stack trace as needed. The property cannot be assigned after construction.

Example

Pass the application logger to `Subscribe` and retain the returned handler. Call `Unsubscribe` with that handler during form cleanup. `LogSample` shows the constructor and read-only property with an application-created exception; constructing event data does not raise the control's event.

```
Option Strict On
Imports System
Imports Avax

Public Module Help_OperationFailed
    Public Function Subscribe(cad As cAvax, log As Action(Of Exception)) As EventHandler(Of
    cAvax.OperationFailedEventArgs)
        Dim handler As EventHandler(Of cAvax.OperationFailedEventArgs) =
            Sub(sender As Object, e As cAvax.OperationFailedEventArgs)
                If e.ErrorException IsNot Nothing Then log(e.ErrorException)
            End Sub
        AddHandler cad.OperationFailed, handler
        Return handler
```

```
    End Function

    Public Sub Unsubscribe(cad As cAvax, handler As EventHandler(Of cAvax.OperationFailedEventArgs))
        RemoveHandler cad.OperationFailed, handler
    End Sub

    Public Sub LogSample(log As Action(Of Exception))
        Dim problem As New InvalidOperationException("Example operation failed.")
        Dim details As New cAvax.OperationFailedEventArgs(problem)
        log(details.ErrorException)
    End Sub
End Module
```

Related members

Command, CommandFilePicker.

PrintingPage

cAvax event

Purpose

This event occurs when a page is being printed and returns the number of the current printing page and the number of the total pages to be print

VB.NET declaration

```
Event PrintingPage(ByVal Sender As Object, ByVal e As cAvax.PrintingPageEventArgs)
```

Parameters

Name	Type and passing	Meaning
Sender	ByVal Sender As Object	Is the object that raises the event
e	ByVal e As PrintingPageEventArgs	Event data. The fields are described below; writable cancellation or output fields are read by the control when the handler returns.

Use the declaration for the overload you call. ByRef arguments can receive outputs; declare variables and arrays with the exact indicated type. Optional defaults are shown in the declaration.

Event data

Member	Declaration	Meaning
e.lMaxPage	lMaxPage As Integer	Is the total number of pages printed
e.lNowPage	lNowPage As Integer	Is the page number of the current page being printed

An event has no return value. Handle it with a procedure whose arguments match the declaration. Where a cancellation flag or output array is documented, assign it on the supplied event data object.

Example

Setup

Add an Avax.cAvax Windows Forms control named `cAvax1`. Paste the procedures into the form class. Example setup and supporting files explains references, initialization and file paths.

Add these Windows Forms controls or components to the form. Fields referenced by Handles clauses must be declared WithEvents, as the Visual Studio designer normally does.

Name	Type
Command1	Button
Label1	Label

VB.NET example

```
Private Sub Form1_Load(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs) Handles
    MyBase.Load
    Command1.Text = "Print"
    Label1.Text = ""
    cAvax1.StartAvax()
    Call cAvax1.Add_Line(0, 0, 0, 35, 35, 0)
    Call cAvax1.Add_Line(0, 35, 0, 35, 35, 0)
    Call cAvax1.Add_Line(35, 35, 0, 35, 0, 0)
    Call cAvax1.Add_Line(0, 35, 0, 35, 0, 0)
    Call cAvax1.Add_Line(0, 0, 0, 35, 0, 0)
    Call cAvax1.Add_Line(0, 0, 0, 0, 35, 0)
    CAvax1.Command = Avax.AvaxCommand.AutoLimits_c
```

```
End Sub
Private Sub Form1_FormClosed(ByVal eventSender As System.Object, ByVal eventArgs As
    System.Windows.Forms.FormClosedEventArgs) Handles Me.FormClosed
    cAvax1.EndAvax()
End Sub
Private Sub Command1_Click(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs)
    Handles Command1.Click
    Call CAvax1.SetAvaxProperty(Avax.AvaxProp.PrintAllOn, True)
    Call CAvax1.SetAvaxProperty(Avax.AvaxProp.PrintUnit_MM, 10)
    CAvax1.Command = Avax.AvaxCommand.Print_c
End Sub
Private Sub CAvax1_PrintingPage(ByVal Sender As Object, ByVal e As Avax.cAvax.PrintingPageEventArgs)
    Handles CAvax1.PrintingPage
    Label1.Text = "Printing page " & Format(e.lNowPage) & " from " & Format(e.lMaxPage)
End Sub
```

Using event handlers

Progress

API reference

Progress

Reports operation progress while the control processes drawing data. Handle the event to update a progress bar or status display.

Declaration

```
Public Event Progress(ByVal ProgressType As AvaxProgress, ByVal Percent As Double)
```

Parameters

Parameter	Purpose
ProgressType	Operation category. See AvaxProgress for the names and Integer values.
Percent	Completion percentage from 0 through 100. This is Double in v3; use Double for event-handler variables formerly declared Single.

Behavior

Call the control on its owning UI thread. Bounds calculations report 0 at the start, increasing intermediate percentages, and 100 after committing their result. Updates are coalesced to approximately one percentage point while retaining fractional Double values. Empty scans still report start and completion. AutoLimits_c and FitToAvax report Autolimits_p. Initial automatic fitting during loading belongs to Load_p.

PauseAvaxEvents suppresses notifications while allowing calculation to finish. If a handler changes the drawing, geometry, explicit bounds, view, active command or lifecycle, the superseded bounds scan stops without overwriting that state or sending stale completion. A nested bounds operation owns its own result. Exceptions thrown by the handler propagate to the caller.

VB.NET example

Pass an initialized control and a reporting callback. The example fits the view and detaches its event handler even if the reporting callback throws.

```
Option Strict On
Imports System
Imports Avax

Public Module BoundsProgressExample
    ' Call on the owning UI thread after StartAvax.
    Public Sub FitWithProgress(cad As cAvax, report As Action(Of Double))
        Dim handler As cAvax.ProgressEventHandler =
            Sub(kind As AvaxProgress, percent As Double)
                If kind = AvaxProgress.Autolimits_p Then report(percent)
            End Sub
        AddHandler cad.Progress, handler
        Try
            cad.FitToAvax()
        Finally
            RemoveHandler cad.Progress, handler
        End Try
    End Sub
End Module
```

View and rendering progress

Zoom_p accompanies changed viewport commits. Extends_p accompanies fitting to stored drawing extents; CalcExtends_c first reports FindMinMax_p while recalculating them. Changed views report 0 before commit and 100 after the new view is committed. Interactive pan, dynamic-zoom and scrollbar gestures report their final committed operation. Unchanged views add no fictitious progress.

Redraw_p reports actual drawing-frame production, including explicit Redraw/RePaint and Refresh/Regen requests. Direct vector rendering coalesces intermediate updates. Prepared scenes report a rendering batch; retained geometry reports its constant replay work. Frozen screen replay and display suspension emit no fictitious redraw work. Background cache preparation can defer rendering updates until a frame is produced. Explicit redraw renders the drawing even with automatic rendering disabled and without a prepared cache.

A redraw reports 100 after publishing its complete frame. If an earlier progress callback supersedes the drawing, command, view, selection, projection or refresh request, its incomplete old frame is discarded. UI painting during another progress callback does not cancel a parent bounds calculation. Host progress exceptions propagate on both CPU and GPU rendering paths.

Observe view operations

Retain the subscription while the form is active, since rendering can occur after the method that requested it returns. Call Unsubscribe during form cleanup.

```
Option Strict On
Imports System
Imports Avax

Public Module ViewProgressExample
    ' Subscribe after StartAvax and retain the returned handler until the form closes.
    Public Function Subscribe(cad As cAvax, report As Action(Of AvaxProgress, Double)) As
    cAvax.ProgressEventHandler
        Dim handler As cAvax.ProgressEventHandler =
            Sub(kind As AvaxProgress, percent As Double)
                If kind = AvaxProgress.Redraw_p OrElse kind = AvaxProgress.Extends_p OrElse kind =
    AvaxProgress.Zoom_p Then
                    report(kind, percent)
                End If
            End Sub
        AddHandler cad.Progress, handler
        Return handler
    End Function

    Public Sub Unsubscribe(cad As cAvax, handler As cAvax.ProgressEventHandler)
        RemoveHandler cad.Progress, handler
    End Sub
End Module
```

Redo

API reference

Redo

Notifies the host that Undo has made a Redo action available. Use it to enable a Redo button or update its caption.

Declaration

```
Public Event Redo(ByVal RedoType As AvaxUndo)
```

Parameters

Parameter	Purpose
RedoType	The R2-compatible notification kind of the available Redo action. See AvaxUndo.

Behavior

A successful Undo publishes Redo availability after restoring the complete previous state. The event announces the resulting available action. A successful Redo publishes Undo availability instead.

Original edit	Redo event argument after Undo	CurrentRedo type after Undo
Create_u	Unerase_u	Erase_u
Erase_u	Erase_u	Unerase_u
Other action kinds, including Zoom_u	Original kind	Original kind

The event describes the offered action; CurrentRedo retains the historical inverse-record labels shown above. Keep these distinct when displaying a command caption.

Call the control on its owning UI thread. The complete reversible state is committed before the event, so a handler can query history, execute the available command, clear history, replace the drawing or begin another edit. The returning operation retains the state established by the handler. An exception from a handler propagates to the caller after the completed edit; the committed history remains available.

PauseAvaxEvents suppresses notifications without discarding history. Resuming events and calling GetAvaxEvents do not replay suppressed notifications. A control that has been stopped, disabled or disposed does not publish a pending availability event. If an earlier Change or ZoomChange handler clears or replaces the announced record, the old notification is skipped.

Subscribers run in normal .NET registration order and receive the same ByVal enum argument. An earlier subscriber may change history, so query the control when current availability is needed. Clearing history does not synthesize an event with None_u; query CurrentUndo and CurrentRedo after a clear.

Use CurrentRedo to read current Redo availability. Execute it with Command = AvaxCommand.Redo_c. A new recorded edit invalidates the old Redo branch.

VB.NET example

Subscribe after StartAvax, retain the returned handler while the form is active, and call Unsubscribe during form cleanup. The callback receives the notification kind and the current record count and query kind.

```
Option Strict On
Imports System
Imports Avax

Public Module RedoAvailabilityExample
    Public Function Subscribe(cad As cAvax, show As Action(Of AvaxUndo, Integer, AvaxUndo)) As
    cAvax.RedoEventHandler
        Dim handler As cAvax.RedoEventHandler =
            Sub(kind As AvaxUndo)
```

```
                Dim queryKind As AvaxUndo
                Dim count As Integer = cad.CurrentRedo(queryKind)
                show(kind, count, queryKind)
            End Sub
        AddHandler cad.Redo, handler
        Return handler
    End Function

    Public Sub Unsubscribe(cad As cAvax, handler As cAvax.RedoEventHandler)
        RemoveHandler cad.Redo, handler
    End Sub
End Module
```

SelectChange

cAvax event

Purpose

This event occurs when a new selection or unselection is done and returns the handle of the last selected object

See Selection commands and type filters for one-based handle arrays, point/window selection, modifier keys, type-mask values, and committed selection notifications.

VB.NET declaration

```
Event SelectChange(ByVal Sender As Object, ByVal e As cAvax.SelectChangeEventArgs)
```

Parameters

Name	Type and passing	Meaning
Sender	ByVal Sender As Object	Is the object that raises the event
e	ByVal e As SelectChangeEventArgs	Event data. The fields are described below; writable cancellation or output fields are read by the control when the handler returns.

Use the declaration for the overload you call. ByRef arguments can receive outputs; declare variables and arrays with the exact indicated type. Optional defaults are shown in the declaration.

Event data

Member	Declaration	Meaning
e.LastSelectedHandle	LastSelectedHandle As Integer	Is the handle of the last selected object

An event has no return value. Handle it with a procedure whose arguments match the declaration. Where a cancellation flag or output array is documented, assign it on the supplied event data object.

Example

Setup

Add an Avax.cAvax Windows Forms control named `cAvax1`. Paste the procedures into the form class. Example setup and supporting files explains references, initialization and file paths.

Add these Windows Forms controls or components to the form. Fields referenced by Handles clauses must be declared WithEvents, as the Visual Studio designer normally does.

Name	Type
Label1	Label

VB.NET example

```
Private Sub cAvax1_SelectChange(ByVal Sender As Object, ByVal e As Avax.cAvax.SelectChangeEventArgs)
    Handles cAvax1.SelectChange
    If e.LastSelectedHandle <> 0 Then
        MsgBox("The line was selected", MsgBoxStyle.Information, "Event SelectChange")
    Else
        MsgBox("The line was unselected", MsgBoxStyle.Information, "Event SelectChange")
    End If
End Sub
Private Sub Form1_Load(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs) Handles
    MyBase.Load
    Label1.Text = "Select the line..."
    cAvax1.StartAvax()
    Call cAvax1.Add_Line(0, 0, 0, 10, 10, 0)
```

```
End Sub
Private Sub Form1_FormClosed(ByVal eventSender As System.Object, ByVal eventArgs As
    System.Windows.Forms.FormClosedEventArgs) Handles Me.FormClosed
    cAvax1.EndAvax()
End Sub
```

Using event handlers

SelectedPoints

cAvax event

Purpose

This event is related with the command SelectPoints which has been used in the drawing to select points with a specific key name. When a selection of every point is executed this event is triggered

VB.NET declaration

```
Event SelectedPoints(ByVal Sender As Object, ByVal e As cAvax.SelectedPointsEventArgs)
```

Parameters

Name	Type and passing	Meaning
Sender	ByVal Sender As Object	Is the object that raises the event
e	ByVal e As SelectedPointsEventArgs	Event data. The fields are described below; writable cancellation or output fields are read by the control when the handler returns.

Use the declaration for the overload you call. ByRef arguments can receive outputs; declare variables and arrays with the exact indicated type. Optional defaults are shown in the declaration.

Event data

Member	Declaration	Meaning
e.lPointNumber	lPointNumber As Integer	Is the number of the last selected point
e.SelectId	SelectId As String	A String expression that defines the key name of the Select Points process
e.x1	x1 As Double()	An array of Singles for the x coordinates of the selected points
e.y1	y1 As Double()	An array of Singles for the y coordinates of the selected points
e.z1	z1 As Double()	An array of Singles for the z coordinates of the selected points

An event has no return value. Handle it with a procedure whose arguments match the declaration. Where a cancellation flag or output array is documented, assign it on the supplied event data object.

Remarks

Raised synchronously after each accepted point in a SelectPoints request. SelectId identifies the request; lPointNumber is the cumulative Integer point count including seeds. x1, y1 and z1 are one-based Double snapshots containing every accepted coordinate, including Z. During an intermediate notification, GetSelectedPoints returns the active points. The final point clears the request before raising this event, so GetSelectedPoints then returns 0; copy the final event arrays if they are needed later. Earlier snapshots remain valid when more points are accepted. Mutating event fields or arrays does not change internal request data. Handlers share the supplied event-data object in normal .NET subscription order. A handler may start a new request or drafting command, replace the drawing, end the control or accept another point; the returning old notification does not clear that newer state. A handler exception does not roll back an already accepted point. PauseAvaxEvents suppresses notifications while collection and completion continue; suppressed notifications are not replayed. Mouse-driven delivery runs on the Windows Forms UI thread, and programmatic acceptance raises the event on its calling thread.

Numeric changes from v2

Update the following variables, arrays or event-data receivers when porting the corresponding v2 call. The declarations above show the current overloads.

Parameter or value	v2 type	v3 type
e.x1	Single()	Double()
e.y1	Single()	Double()
e.z1	Single()	Double()

For ByRef parameters, use a variable with the exact v3 type. Preserve array dimensions and the documented indexing. Moving from v2 to v3 explains conversion and legacy file-storage rules.

Example

Setup

Add an Avax.cAvax Windows Forms control named `cAvax1`. Paste the procedures into the form class. Example setup and supporting files explains references, initialization and file paths.

Add these Windows Forms controls or components to the form. Fields referenced by Handles clauses must be declared WithEvents, as the Visual Studio designer normally does.

Name	Type
Command1	Button

VB.NET example

```
Private Sub cAvax1_SelectedPoints(ByVal Sender As Object, ByVal e As Avax.cAvax.SelectedPointsEventArgs)
    Handles cAvax1.SelectedPoints
    Select Case e.SelectId
        Case "Create triangle"
        If e.lPointNumber = 3 Then
            Call cAvax1.Add_Shape(e.x1, e.y1, e.z1, Nothing, 1, 3, 2, 0, "", 0, 0, -1, -1, -1, -1,
    Avax.AvaxPolylineType.Polyline_pt)
        End If
    End Select
End Sub
Private Sub Form1_Load(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs) Handles
    MyBase.Load
    Command1.Text = "Create triangle"
    cAvax1.StartAvax()
End Sub
Private Sub Form1_FormClosed(ByVal eventSender As System.Object, ByVal eventArgs As
    System.Windows.Forms.FormClosedEventArgs) Handles Me.FormClosed
    cAvax1.EndAvax()
End Sub
Private Sub Command1_Click(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs)
    Handles Command1.Click
    Dim sSelectId As String = String.Empty
    Dim lPointsToSelect As Integer
    Dim ExistX() As Double = Nothing
    Dim ExistY() As Double = Nothing
    Dim ExistZ() As Double = Nothing
    Dim fBoxOn As Boolean
    Dim fNoSnapAndOrtho As Boolean
    Dim Answer As Boolean
    lPointsToSelect = 3
    ReDim ExistX(1)
    ReDim ExistY(1)
    ReDim ExistZ(1)
    ExistX(1) = 3
    ExistY(1) = 3
    sSelectId = "Create triangle"
    Answer = cAvax1.SelectPoints(sSelectId, lPointsToSelect, ExistX, ExistY, ExistZ, fBoxOn,
    fNoSnapAndOrtho)
End Sub
```

Using event handlers

SelectRegionStretch

cAvax event

Purpose

This event occurs when the SelectRegion window is changing and returns the handles of the selected objects that will be transformed

VB.NET declaration

```
Event SelectRegionStretch(ByVal Sender As Object, ByVal e As cAvax.SelectRegionStretchEventArgs)
```

Parameters

Name	Type and passing	Meaning
Sender	ByVal Sender As Object	Is the object that raises the event
e	ByVal e As SelectRegionStretchEventArgs	Event data. The fields are described below; writable cancellation or output fields are read by the control when the handler returns.

Use the declaration for the overload you call. ByRef arguments can receive outputs; declare variables and arrays with the exact indicated type. Optional defaults are shown in the declaration.

Event data

Member	Declaration	Meaning
e.fCancel	fCancel As Boolean	This event trigger can be canceled by changing this parameter to true
e.StretchHandles	StretchHandles As Integer()	An array of Integers for the handles of the selected objects
e.xNewMax	xNewMax As Double	Is the x new changed coordinate of the top right extend of the SelectRegion window
e.xNewMin	xNewMin As Double	Is the x new changed coordinate of the bottom left extend of the SelectRegion window
e.xOldMax	xOldMax As Double	Is the x original coordinate of the top right extend of the SelectRegion window
e.xOldMin	xOldMin As Double	Is the x original coordinate of the bottom left extend of the SelectRegion window
e.yNewMax	yNewMax As Double	Is the x new changed coordinate of the top right extend of the SelectRegion window
e.yNewMin	yNewMin As Double	Is the y new changed coordinate of the bottom left extend of the SelectRegion window
e.yOldMax	yOldMax As Double	Is the y original coordinate of the top right extend of the SelectRegion window
e.yOldMin	yOldMin As Double	Is the y original coordinate of the bottom left extend of the SelectRegion window

An event has no return value. Handle it with a procedure whose arguments match the declaration. Where a cancellation flag or output array is documented, assign it on the supplied event data object.

Remarks

Raised when a changed selection-region move or resize is committed, before object geometry changes. Sender is the control. StretchHandles is a one-based Integer array; the eight old/new rectangle coordinates are Double. Subscribers share the same mutable event object. They may edit the coordinates, replace the handle array, or set fCancel=True.

An unchanged center-grip proposal is a pure translation, followed by BeforeMove. Resizing or intentionally changed rectangle proportions uses BeforeTransform. A single line's corner grip moves its corresponding endpoint and can change its slope; its opposite endpoint and both Z values are retained. The rotation grip uses BeforeRotate instead of SelectRegionStretch.

Changed targets and parameters are prepared and validated together before editing, including displayed drag limits, nested resources and live external geometry. Success is one Undo/Redo transaction and raises one Change notification. A cancelled or rejected gesture adds no pending geometry or history entry; independent callback edits are retained. Representable small Double rectangles retain active grips and screen-relative motion; exactly zero-size axes keep their explicit degenerate behavior.

Starting another command or replacing/ending the drawing in a callback prevents the older pending gesture from committing. Transparent drafting aids such as Ortho and Grid retain the current gesture. PauseAvaxEvents suppresses the event. Use SetSelectRegionBoundingBox to constrain permitted displayed bounds without changing the selected geometry.

Numeric changes from v2

Update the following variables, arrays or event-data receivers when porting the corresponding v2 call. The declarations above show the current overloads.

Parameter or value	v2 type	v3 type
e.xNewMax	Single	Double
e.xNewMin	Single	Double
e.xOldMax	Single	Double
e.xOldMin	Single	Double
e.yNewMax	Single	Double
e.yNewMin	Single	Double
e.yOldMax	Single	Double
e.yOldMin	Single	Double

For ByRef parameters, use a variable with the exact v3 type. Preserve array dimensions and the documented indexing. Moving from v2 to v3 explains conversion and legacy file-storage rules.

Example

Setup

Add an Avax.cAvax Windows Forms control named `cAvax1`. Paste the procedures into the form class. Example setup and supporting files explains references, initialization and file paths.

Add these Windows Forms controls or components to the form. Fields referenced by Handles clauses must be declared WithEvents, as the Visual Studio designer normally does.

Name	Type
Label1	Label

VB.NET example

```
Private Sub Form1_Load(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs) Handles
    MyBase.Load
    Label1.Text = "Try to stretch or move the line..."
    cAvax1.StartAvax()
    Call cAvax1.Add_Line(0, 0, 0, 10, 10, 0)
    CAvax1.Command = Avax.AvaxCommand.Select_All_c
End Sub
Private Sub Form1_FormClosed(ByVal eventSender As System.Object, ByVal eventArgs As
    System.Windows.Forms.FormClosedEventArgs) Handles Me.FormClosed
    cAvax1.EndAvax()
End Sub
Private Sub CAvax1_SelectRegionStretch(ByVal Sender As Object, ByVal e As
    Avax.cAvax.SelectRegionStretchEventArgs) Handles CAvax1.SelectRegionStretch
    e.fCancel = True
```

```
End Sub
```

Using event handlers

SetExternalLines

cAvax event

Purpose

Requests line geometry for an external source name. Supply six equal-length one-based Double arrays of XYZ endpoints.

VB.NET declaration

```
Event SetExternalLines(ByVal Sender As Object, ByVal e As cAvax.SetExternalLinesEventArgs)
```

Parameters

Name	Type and passing	Meaning
Sender	ByVal Sender As Object	Is the object that raises the event
e	ByVal e As SetExternalLinesEventArgs	Event data. The fields are described below; writable cancellation or output fields are read by the control when the handler returns.

Use the declaration for the overload you call. ByRef arguments can receive outputs; declare variables and arrays with the exact indicated type. Optional defaults are shown in the declaration.

Event data

Member	Declaration	Meaning
e.sName	sName As String	A String expression that defines the key name of the external lines
e.x1	x1 As Double()	An array of Singles for the x coordinates of the first points of the external lines
e.x2	x2 As Double()	An array of Singles for the x coordinates of the second points of the external lines
e.y1	y1 As Double()	An array of Singles for the y coordinates of the first points of the external lines
e.y2	y2 As Double()	An array of Singles for the y coordinates of the second points of the external lines
e.z1	z1 As Double()	An array of Singles for the z coordinates of the first points of the external lines
e.z2	z2 As Double()	An array of Singles for the z coordinates of the second points of the external lines

An event has no return value. Handle it with a procedure whose arguments match the declaration. Where a cancellation flag or output array is documented, assign it on the supplied event data object.

Remarks

The event is synchronous on the calling control thread. Fill x1/y1/z1/x2/y2/z2 with equal-length Double arrays; indices 1 through Length-1 are actual line segments and index 0 is reserved. For an existing reference, six Nothing arrays retain cached geometry; one-element reserved-only arrays clear the linework while retaining the reference. Partial or malformed arrays report an error and retain a valid cache.

Creation, regeneration, opening, fitting, bounds queries, rendering and CAD export can request the source. Do not depend on an exact callback count. Supply source-space coordinates; the control applies the reference stored transform. A callback that replaces the drawing or edits/deletes the reference prevents an older refresh from overwriting that state.

Application exceptions propagate from creation and regeneration. DXF/DWG exporters report resolver failures and return False while retaining an existing destination. PauseAvaxEvents suppresses the resolver and uses available cached geometry.

Numeric changes from v2

Update the following variables, arrays or event-data receivers when porting the corresponding v2 call. The declarations above show the current overloads.

Parameter or value	v2 type	v3 type
e.x1	Single()	Double()
e.x2	Single()	Double()
e.y1	Single()	Double()
e.y2	Single()	Double()
e.z1	Single()	Double()
e.z2	Single()	Double()

For ByRef parameters, use a variable with the exact v3 type. Preserve array dimensions and the documented indexing. Moving from v2 to v3 explains conversion and legacy file-storage rules.

Example

Setup

Add an Avax.cAvax Windows Forms control named `cAvax1`. Paste the procedures into the form class. Example setup and supporting files explains references, initialization and file paths.

Add these Windows Forms controls or components to the form. Fields referenced by Handles clauses must be declared WithEvents, as the Visual Studio designer normally does.

Name	Type
Command1	Button

Copy the supplied v3 `map-double.dat` from the SDK examples beside your application executable. The example reads its coordinates with BinaryReader.ReadDouble and supplies Double arrays to SetExternalLines.

VB.NET example

```
Dim x1_Map() As Double = Nothing
Dim y1_Map() As Double = Nothing
Dim z1_Map() As Double = Nothing
Dim x2_Map() As Double = Nothing
Dim y2_Map() As Double = Nothing
Dim z2_Map() As Double = Nothing
Sub ReadMapData()
    Static loaded As Boolean
    If loaded Then Return
    Dim mapFile As String = System.IO.Path.Combine(System.AppDomain.CurrentDomain.BaseDirectory,
    "map-double.dat")
    Using reader As New System.IO.BinaryReader(System.IO.File.OpenRead(mapFile))
        Dim lineCount As Integer = reader.ReadInt32()
        If lineCount < 0 OrElse lineCount >= Integer.MaxValue OrElse reader.BaseStream.Length <> 4L + 32L
    * (CLng(lineCount) + 1L) Then
            Throw New System.IO.InvalidDataException("Invalid map-double.dat length or line count.")
        End If
        Dim coordinates(3)() As Double
        For axis As Integer = 0 To 3
            ReDim coordinates(axis)(lineCount)
            For i As Integer = 0 To lineCount
                Dim value As Double = reader.ReadDouble()
                If i > 0 AndAlso (Double.IsNaN(value) OrElse Double.IsInfinity(value)) Then
                    Throw New System.IO.InvalidDataException("Non-finite map-double.dat coordinate.")
                End If
                coordinates(axis)(i) = value
            Next
        Next
        x1_Map = coordinates(0) : y1_Map = coordinates(1)
```

```
        x2_Map = coordinates(2) : y2_Map = coordinates(3)
        ReDim z1_Map(lineCount), z2_Map(lineCount)
    End Using
    loaded = True
End Sub
Private Sub Command1_Click(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs)
    Handles Command1.Click
    Call ReadMapData()
    Call cAvax1.Add_ExternalLines("My_Map")
    CAvax1.Command = Avax.AvaxCommand.AutoLimits_c
End Sub
Private Sub Form1_Load(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs) Handles
    MyBase.Load
    Command1.Text = "Add_ExternalLines"
    cAvax1.StartAvax()
End Sub
Private Sub Form1_FormClosed(ByVal eventSender As System.Object, ByVal eventArgs As
    System.Windows.Forms.FormClosedEventArgs) Handles Me.FormClosed
    cAvax1.EndAvax()
End Sub
Private Sub CAvax1_SetExternalLines(ByVal Sender As Object, ByVal e As
    Avax.cAvax.SetExternalLinesEventArgs) Handles CAvax1.SetExternalLines
    Select Case e.sName
        Case "My_Map"
        e.x1 = x1_Map
        e.y1 = y1_Map
        e.z1 = z1_Map
        e.x2 = x2_Map
        e.y2 = y2_Map
        e.z2 = z2_Map
    End Select
End Sub
```

Using event handlers

Undo

API reference

Undo

Notifies the host that a committed edit is available for Undo. Use it to enable an Undo button or update its caption.

Declaration

```
Public Event Undo(ByVal UndoType As AvaxUndo)
```

Parameters

Parameter	Purpose
UndoType	The original action kind of the available Undo record. See AvaxUndo.

Behavior

A recorded edit, a nonempty UndoRecorderOn group commit and a successful Redo publish Undo availability. This event announces an available record; it does not mean that an Undo command has just executed. Recording defers the notification until the group closes. Closing an empty group sends no event. Recorded viewport changes use Zoom_u when ZoomUndo_On_c is enabled.

Call the control on its owning UI thread. The complete reversible state is committed before the event, so a handler can query history, execute the available command, clear history, replace the drawing or begin another edit. The returning operation retains the state established by the handler. An exception from a handler propagates to the caller after the completed edit; the committed history remains available.

PauseAvaxEvents suppresses notifications without discarding history. Resuming events and calling GetAvaxEvents do not replay suppressed notifications. A control that has been stopped, disabled or disposed does not publish a pending availability event. If an earlier Change or ZoomChange handler clears or replaces the announced record, the old notification is skipped.

Subscribers run in normal .NET registration order and receive the same ByVal enum argument. An earlier subscriber may change history, so query the control when current availability is needed. Clearing history does not synthesize an event with None_u; query CurrentUndo and CurrentRedo after a clear.

Use CurrentUndo to read the available record and its type. Execute it with Command = AvaxCommand.Undo_c. After Undo, the Redo event announces the resulting Redo availability.

VB.NET example

Subscribe after StartAvax, retain the returned handler while the form is active, and call Unsubscribe during form cleanup. The callback receives the notification kind and the current record count.

```
Option Strict On
Imports System
Imports Avax

Public Module UndoAvailabilityExample
    Public Function Subscribe(cad As cAvax, show As Action(Of AvaxUndo, Integer)) As
    cAvax.UndoEventHandler
        Dim handler As cAvax.UndoEventHandler =
            Sub(kind As AvaxUndo)
                Dim nextKind As AvaxUndo
                Dim count As Integer = cad.CurrentUndo(nextKind)
                show(kind, count)
            End Sub
        AddHandler cad.Undo, handler
        Return handler
    End Function

    Public Sub Unsubscribe(cad As cAvax, handler As cAvax.UndoEventHandler)
        RemoveHandler cad.Undo, handler
```

```
    End Sub
End Module
```

UserOsnap

cAvax event

Purpose

This occurs when value UserOSnap_c is given to Command property and user clicks on an object while adding a new one.

VB.NET declaration

```
Event UserOsnap(ByVal Sender As Object, ByVal e As cAvax.UserOsnapEventArgs)
```

Parameters

Name	Type and passing	Meaning
Sender	ByVal Sender As Object	Is the object that raises the event
e	ByVal e As UserOsnapEventArgs	Event data. The fields are described below; writable cancellation or output fields are read by the control when the handler returns.

Use the declaration for the overload you call. ByRef arguments can receive outputs; declare variables and arrays with the exact indicated type. Optional defaults are shown in the declaration.

Event data

Member	Declaration	Meaning
e.fNoResult	fNoResult As Boolean	If set true then the NewX and NewY are ignored
e.NewX	NewX As Double	Is the x coordinate of the snap point the user sets
e.NewY	NewY As Double	Is the y coordinate of the snap point the user sets
e.SelectedHandle	SelectedHandle As Integer	A numeric expression that identifies the handle of the object the user clicks on
e.selx1m	selx1m As Double	Is the x coordinate of the first point of the object the user clicks on
e.selx2m	selx2m As Double	Is the x coordinate of the second point of the object the user clicks on
e.sely1m	sely1m As Double	Is the y coordinate of the first point of the object the user clicks on
e.sely2m	sely2m As Double	Is the y coordinate of the second point of the object the user clicks on

An event has no return value. Handle it with a procedure whose arguments match the declaration. Where a cancellation flag or output array is documented, assign it on the supplied event data object.

Remarks

Supplies a host-defined snap point for a picked drawing object. The sender is this control,and handlers run synchronously on the caller's control thread. Handle identifies the picked root object;xMin,yMin,xMax,yMax describe the world-coordinate search bounds. NewX and NewY are initialized to the current input position,and fNoResult defaults toFalse. The payload has mutable Integer Handle,six Double coordinate fields and Boolean fNoResult. Subscribers share the same payload in invocation order. Return finite NewX/NewY and leave fNoResult=False to supply a point;set it True to reject. A custom point may lie outside the search bounds and does not invoke AutoOSnapType. The accepted point retains the drawing operation's normal Z/elevation semantics. No eligible object or subscriber leaves an explicit UserOsnap_c request unresolved rather than choosing an uninitialized origin. Paused events also leave the request available for retry. Newer commands,snap modes,documents or control lifetimes started by a callback take ownership;the old answer cannot overwrite them. Host exceptions propagate without being converted to successful point input.

Numeric changes from v2

Update the following variables, arrays or event-data receivers when porting the corresponding v2 call. The declarations above show the current overloads.

Parameter or value	v2 type	v3 type
e.NewX	Single	Double
e.NewY	Single	Double
e.selx1m	Single	Double
e.selx2m	Single	Double
e.sely1m	Single	Double
e.sely2m	Single	Double

For ByRef parameters, use a variable with the exact v3 type. Preserve array dimensions and the documented indexing. Moving from v2 to v3 explains conversion and legacy file-storage rules.

Example

Setup

Add an Avax.cAvax Windows Forms control named `cAvax1`. Paste the procedures into the form class. Example setup and supporting files explains references, initialization and file paths.

Add these Windows Forms controls or components to the form. Fields referenced by Handles clauses must be declared WithEvents, as the Visual Studio designer normally does.

Name	Type
Button1	Button

VB.NET example

```
Dim avHandle1 As Integer
Dim avHandle2 As Integer
Dim avHandle3 As Integer
Private Sub Form1_Load(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs) Handles
    MyBase.Load
    Call CAvax1.StartAvax()
    avHandle1 = CAvax1.Add_Line(0, 0, 0, 10, 0, 0)
    avHandle2 = CAvax1.Add_Line(0, 5, 0, 10, 5, 0)
    avHandle3 = CAvax1.Add_Line(0, 10, 0, 10, 10, 0)
    CAvax1.Command = Avax.AvaxCommand.Line_c
End Sub
Private Sub Form1_FormClosed(ByVal eventSender As System.Object, ByVal eventArgs As
    System.Windows.Forms.FormClosedEventArgs) Handles Me.FormClosed
    Call cAvax1.EndAvax()
End Sub
Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
    Button1.Click
    CAvax1.Command = Avax.AvaxCommand.UserOsnap_c
End Sub
Private Sub cAvax1_UserOsnap(ByVal Sender As System.Object, ByVal e As Avax.cAvax.UserOsnapEventArgs)
    Handles CAvax1.UserOsnap
    Select Case e.SelectedHandle
        Case avHandle1
        e.NewX = 7.5
        e.NewY = 0
        Case avHandle2
        e.NewX = 2.5
        e.NewY = 5
        Case avHandle3
        e.NewX = 5
        e.NewY = 10
```

```
    End Select
End Sub
```

Using event handlers

Canvas / Page workspace

API reference

Each control/drawing has one shared Canvas / Page workspace. Canvas hides the paper and Page shows it. Switching modes preserves object coordinates, zoom and page scale. The sheet, margins, shadow, title and page-number label are view decorations, not CAD objects; the title/number are not automatically printed as drawing text.

Scale and drawing units

`PageScale` is the denominator: 100 means 1:100. `DrawingUnitMM` is real millimeters per drawing-coordinate unit: 1000 for meters, 10 for centimeters, 1 for millimeters. It does not convert existing coordinates. Printed mm per drawing unit = DrawingUnitMM / PageScale. Paper size in drawing units = paper mm × PageScale / DrawingUnitMM. For A4 portrait, meters and 1:100, the sheet spans 21 × 29.7 drawing units and a 5-unit line prints 50 mm long. Zoom does not alter printing scale.

WorkspaceMode

PageNumber

PageTitle

PageScale

DrawingUnitMM

PagePaperSize

PageOrientation

PageCustomWidthMM

PageCustomHeightMM

PageOriginX

PageOriginY

PageMarginMM

PageMarginsVisible

PageShadowVisible

PageBackgroundColor

PageBackgroundPrintExport

PageWidthMM

PageHeightMM

PagePrintUnitMM

HasWorkspaceSettings

CenterWorkspacePage

ZoomWorkspacePage

GetWorkspacePageBounds

RenderWorkspacePage

PrintWorkspaceToPDF

PrintWorkspaceToPrinter

PrintWorkspaceToSVG

PrintWorkspaceToWMF

PrintWorkspaceToRTF

WorkspaceChanged

Purpose

Raised after workspace settings change, including Undo/Redo and opening a different drawing. Use it to refresh property displays. The event sender is this cAvax control; EventArgs carries no additional values, so read the properties again.

```
Event WorkspaceChanged(ByVal sender As Object, ByVal e As System.EventArgs)
```

Example

```
Option Strict On
Imports System
Imports Avax
Public Module Help_WorkspaceChanged
    Public Sub Attach(cad As cAvax)
        AddHandler cad.WorkspaceChanged, AddressOf OnWorkspaceChanged
    End Sub
    Public Sub Detach(cad As cAvax)
        RemoveHandler cad.WorkspaceChanged, AddressOf OnWorkspaceChanged
    End Sub
    Private Sub OnWorkspaceChanged(sender As Object, e As EventArgs)
        Dim cad = DirectCast(sender, cAvax)
        Dim caption As String = cad.PageTitle & "  1:" & cad.PageScale.ToString()
        ' Assign caption to the host application's workspace label.
    End Sub
End Module
```

AvaxWorkspaceMode

Member	Value
Canvas	0
Page	1

AvaxPageOrientation

Member	Value
Landscape	1
Portrait	0

AVAX-CAD R3, persistence and printing

Use Workspace → Canvas / Page, paper size, orientation and Scale 1:. Properties opens page number/title, units, paper position, margins, shadow and custom paper dimensions in the right panel. Selecting a drawing object shows its properties; press Workspace Properties again for the page. Center paper changes placement, Zoom page changes zoom, and Preview / Print uses the printable content. Objects outside the sheet stay visible in the editor but do not print outside the printable region.

Save/Save As to native .avax stores this drawing's workspace in the optional $R3Workspace section. Old drawings without it open in Canvas with defaults. DXF/DWG do not create layouts from these settings; use native .avax to retain them. When explicit workspace settings exist, the CAD shell's normal Print/Preview/Export PDF uses the workspace sheet in either mode. Existing explicit PrintToPDF/PrintToPrinter overloads retain their own behavior; call the new methods to use workspace settings.

ZoomChange

cAvax event

Purpose

This event occurs when the current zoomed view is changed and returns with the coordinates of the new window viewport

VB.NET declaration

```
Event ZoomChange(ByVal Sender As Object, ByVal e As cAvax.ZoomChangeEventArgs)
```

Parameters

Name	Type and passing	Meaning
Sender	ByVal Sender As Object	Is the object that raises the event
e	ByVal e As ZoomChangeEventArgs	Event data. The fields are described below; writable cancellation or output fields are read by the control when the handler returns.

Use the declaration for the overload you call. ByRef arguments can receive outputs; declare variables and arrays with the exact indicated type. Optional defaults are shown in the declaration.

Event data

Member	Declaration	Meaning
e.xMax	xMax As Double	Committed view maximum X in drawing coordinates; Double.
e.xMin	xMin As Double	Committed view minimum X in drawing coordinates; Double.
e.yMax	yMax As Double	Committed view maximum Y in drawing coordinates; Double.
e.yMin	yMin As Double	Committed view minimum Y in drawing coordinates; Double.

An event has no return value. Handle it with a procedure whose arguments match the declaration. Where a cancellation flag or output array is documented, assign it on the supplied event data object.

Remarks

Notifies the application of an updated viewport. Sender is the AVAX control; xMin,yMin,xMax,yMax contain its committed nominal window in Double drawing coordinates. Step/wheel/window zoom,pan/dynamic zoom,scrollbars,view-history restoration,fitting,workspace navigation,resize,chrome changes and document assignment use this event. Some explicit resize or refresh paths can notify even when the numeric bounds are unchanged. PauseAvaxEvents suppresses notifications. Handlers share the same mutable event-argument instance,but changing its fields does not rewrite the actual view; later notifications use fresh data. Interactive window zoom consumes its accepted corners before publishing,so the handler can query the completed window. A handler may install a newer command or drawing. Exceptions preserve their identity; native Windows message callbacks follow normal WinForms exception routing,including Application.ThreadException. ZoomIn/ZoomOut retain their step-zoom recursion guard.

Numeric changes from v2

Update the following variables, arrays or event-data receivers when porting the corresponding v2 call. The declarations above show the current overloads.

Parameter or value	v2 type	v3 type
e.xMax	Single	Double
e.xMin	Single	Double

Parameter or value	v2 type	v3 type
e.yMax	Single	Double
e.yMin	Single	Double

For ByRef parameters, use a variable with the exact v3 type. Preserve array dimensions and the documented indexing. Moving from v2 to v3 explains conversion and legacy file-storage rules.

Example

Setup

Add an Avax.cAvax Windows Forms control named `cAvax1`. Paste the procedures into the form class. Example setup and supporting files explains references, initialization and file paths.

Add these Windows Forms controls or components to the form. Fields referenced by Handles clauses must be declared WithEvents, as the Visual Studio designer normally does.

Name	Type
Command1	Button
Command2	Button

VB.NET example

```
Private Sub CAvax1_ZoomChange(ByVal Sender As Object, ByVal e As Avax.cAvax.ZoomChangeEventArgs) Handles
    CAvax1.ZoomChange
    Dim sMsg As String = String.Empty
    sMsg = "xMin=" & e.xMin & vbCrLf & "yMin=" & e.yMin & vbCrLf & "xMax=" & e.xMax & vbCrLf & "yMax=" &
    e.yMax
    MsgBox(sMsg, MsgBoxStyle.Information, "ZoomChange")
End Sub
Private Sub Command1_Click(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs)
    Handles Command1.Click
    CAvax1.Command = Avax.AvaxCommand.Zoom_c
End Sub
Private Sub Command2_Click(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs)
    Handles Command2.Click
    CAvax1.Command = Avax.AvaxCommand.Previous_c
End Sub
Private Sub Form1_Load(ByVal eventSender As System.Object, ByVal eventArgs As System.EventArgs) Handles
    MyBase.Load
    cAvax1.StartAvax()
    Call cAvax1.Add_Line(0, 0, 0, 12, 12, 0)
End Sub
Private Sub Form1_FormClosed(ByVal eventSender As System.Object, ByVal eventArgs As
    System.Windows.Forms.FormClosedEventArgs) Handles Me.FormClosed
    cAvax1.EndAvax()
End Sub
```

Using event handlers

Enumerations

Use the linked topic list in Contents or the PDF bookmarks to locate a member or guide.

Enumerations

Select a topic for its purpose, declaration, argument descriptions and example.

- AvaxACADVersion
- AvaxAlignConstants
- AvaxClickType
- AvaxCommand
- AvaxFillType
- AvaxItemType
- AvaxLanguage
- AvaxOSnapType
- AvaxPaperSize
- AvaxPolygonsComp
- AvaxPolylineType
- AvaxProp
- AvaxScrollBarsStatus
- AvaxShapeType
- AvaxUndo

Additional members

- AvaxButtonStyle
- AvaxCommand.Ang_c
- AvaxCommand.Center_c
- AvaxCommand.Triangle_c
- AvaxDimAngularUnits
- AvaxDimensionKind
- AvaxDimFit
- AvaxDimLinearUnits
- AvaxDimLineSpacing
- AvaxDimTextAttachment
- AvaxDimTextHorizontal
- AvaxDimTextMove
- AvaxDimTextVertical
- AvaxHatchIslandStyle
- AvaxHatchKind
- AvaxImageEffectKind
- AvaxOSnapType.Center_o
- AvaxPageOrientation
- AvaxPathNodeMode
- AvaxPathSegmentKind
- AvaxProgress

- AvaxTextAlignment
- AvaxWorkspaceMode
- modbutton.isEstate

AvaxACADVersion

Selects a target file format, not a bit mask. The eight named members and their Integer values retain the R2 contract. PrintToDXF accepts every named value. DWG writing accepts ACAD_2000 and the additional supported v3 codes described below; earlier named versions are rejected without replacing the destination.

Parameter	Description	Value
ACAD_25	ACAD Version 2.5	0
ACAD_26	ACAD Version 2.6	1
ACAD_9	ACAD Version 9	2
ACAD_10	ACAD Version 10	3
ACAD_11	ACAD Version 11	4
ACAD_13	ACAD Version 13	5
ACAD_14	ACAD Version 14	6
ACAD_2000	ACAD Version 2000	7

The DXF header identifiers for the named members, in their numeric order 0 through 7, are AC1002, AC1003, AC1004, AC1006, AC1009, AC1012, AC1014 and AC1015.

Additional v3 target codes

Integer code	AutoCAD release	DXF header	DWG writing
8	2004	AC1018	Supported
9	2007	AC1021	Not supported
10	2010	AC1024	Supported
11	2013	AC1027	Supported
12	2018	AC1032	Supported

These are additional numeric target codes, not named public enum members. Pass CType(code, Avax.AvaxACADVersion) to the full export overload. The DXF command reads ACADVersion_p; explicit full-method Version arguments take precedence, and the short PrintToDXF overload always uses ACAD_10. The default ACADVersion_p value is Integer 3. Property XML retains the selected value. An invalid target fails during export and leaves an existing destination intact.

See PrintToDXF and PrintToDWG for purposes, declarations and inline examples. Choosing a version does not enable every entity type in that format; consult the export method restrictions.

AvaxAlignConstants

Selects how an object is aligned within its target rectangle.

Parameter	Description	Value
None_a	No Align	0
Top_a	Align to the Top	1
Bottom_a	Align to the Bottom	2
Left_a	Align to the Left	4
Right_a	Align to the Right	8
Center_a	Align to the Center	16
Middle_a	Align to the Middle	32

AvaxClickType

Identifies the interactive operation that is consuming a drawing click.

Parameter	Description	Value
None_ct	None	0
Select_ct	Select	1
ZoomWindowCorner_ct	Corner of the zoom window	2
SelectWindowCorner_ct	Corner of the select window	3
LineVertex_ct	Line's vertex	4
Point_ct	Point's vertex	5

AvaxCommand

Selects a built-in command through the Command property.

For bitmap export, call PrintToBMP. The BMP_Out_c name shown in older help is not a declared AvaxCommand value.

Parameter	Description	Value
None_c	No command	0
Cancel_c	Cancels all actions and unselects all selected objects if there are any	1
Zoom_c	Zooms to display an area specified by two opposite corners of a rectangular window	2
All_c	Zooms to display all objects of the drawing	3
Select_c	Enables the select action. So user cad add or remove objects (one object every time) from the selection set, using a small box which replaces the crosshair on the graphics cursor.	4
Line_c	Creates straight line segments	5
Polyline_c	Creates polylines	6
Point_c	Creates Points	7
Endpoint_c	Snaps to the closest endpoint of a line, polyline, arc etc.	8
Midpoint_c	Snaps to the midpoint of a line, polyline, arc, etc. or to the center of a circle	9
Nearest_c	Snaps to the closest point of a line, polyline, arc, etc.	10
Vertical_c	Snaps to a point perpendicular to a line, polyline, arc, etc.	11
Intersec_c	Snaps to a intersection of a line, polyline, arc, etc.	12
Rotate_c	Rotates the selected objects	13
Mirror_c	Creates a mirror copy of selected drawing objects about a a temporary line given from the user	14
Scale_c	Enlarges or reduces selected drawing objects by a temporary line given from the user	15
Move_c	Displaces selected drawing objects a specified distance in a specified direction given from the user by a temporary line	16
WinSel_c	Selects all drawing objects inside a rectangle defined by two points	21
Unselect_c	Unselects all selected objects if there are any	22
Erase_c	Removes all selected objects from the drawing	23
Undo_c	Reverses the effects of commands. Avax can reverse the last 3000 commands	24
Redo_c	Reverses the effects of the previous UNDO command	25
Previous_c	Restore the last view in the independent previous-view list, which retains up to 50 entries. The list is populated even when ZoomUndo recording is off. When ZoomUndo_On_c is enabled, the Previous operation is also reversible through normal Undo/Redo as Zoom_u. Assigning a new drawing clears previous views.	26
InWinSel_c	Selects all drawing objects completely inside a rectangle defined by two points	27
SelectAdd_c	Switches to the "Add" selection method in which selected objects can be added to the selection set	28
SelectSub_c	Switches to the "Sub" selection method in which selected objects can be removed from the selection set	29
Redraw_c	Refreshes the display of the current viewport	31
Extends_c	Zooms to display the drawing extents	32
Copy_c	Duplicates selected drawing objects a specified distance in a specified direction given from the user by a temporary line	33

Parameter	Description	Value
Save_c	Saves the current drawing	34
Exit_c	Teminates the control	35
Extend_c	Extends a line to meet another line	36
Fillet_c	Fillets the edges of selected lines	37
Trim_c	Trims lines at a cutting edge defined by other already selected lines	38
Break_c	Erases part of already selected line or splits this line in two by a temporary line given from the user	39
Block_c	Creates a block definition from selected objects	40
Insert_c	Inserts an AVAX block or drawing (.blk, .avax or .avx). Choose the file, scale and rotation angle, then specify its placement point. The preview and placed geometry use the same transform. The result is a block with retained child metadata and one Undo entry. Cancel discards the pending insertion. Scale and angle transform X/Y about the source origin; source Z values are retained. SetCoordinate(x, y, z) adds the supplied XYZ placement offset, so z is added to each imported elevation without being multiplied by the XY scale. Empty/corrupt source files and unrepresentable transforms are rejected before a pending insertion is published. A failed placement leaves the drawing unchanged and keeps the valid source available for a corrected point.	41
Ellipse_c	Creates an ellipse	42
Arc_c	Creates an arc	43
Text_c	Opens the text editor, then places the entered text at the next mouse or SetCoordinate point. Uses the current font, style flags and text scale. CR, LF and CRLF line breaks are supported. Preview does not change the drawing; successful placement creates one undoable text object. Cancel leaves no object. If the size or placement exceeds the finite coordinate range, the command retains the text so the user can correct the scale or point and retry.	44
FastText_On_c	Switches to the "FastText" drawing method in which Avax draws the bounding box of a text and not the real text object	46
FastText_Off_c	Switches off the "FastText" drawing method	47
Polygon_c	Creates an equilateral closed polyline	48
Rectangle_c	Creates a rectangular closed polyline	49
AutoLimits_c	Increses the drawing limits by the AvaxProp.DownLeftFactor_p and AvaxProp.UpRightFactor_p factors	50
ZoomUndo_On_c	Enable adding subsequent changed viewport bounds to normal Undo/Redo as Zoom_u. Recording is off by default. This does not enable or disable the independent Previous_c list or bitmap caching.	51
ZoomUndo_Off_c	Stop adding subsequent viewport changes to normal Undo/Redo. Existing undo records and the independent Previous_c history remain available. Bitmap caching is controlled separately by ZoomMemoryOn_p.	52
Explode_c	Breaks compound selected objects into their component objects	54
Pedit_c	Create a polyline by the already selected lines	55
Node_c	Snaps to a point	56
Stretch_c	Moves or stretches already selected lines or polylines	57
Ortho_On_c	Constrains cursor movement to the horizontal or vertical directions	58
Ortho_Off_c	Switches off the horizontal or vertical cursor movement	59
Array_c	Creates multiple copies of already selected objects	60
Offset_c	Creates parallel lines at a specified distance	61
Set_c	Opens a dialog box in which user can give the pen width, color, layer, etc.	63

Parameter	Description	Value
Tangent_c	Snaps to the tangent of a circle and ellipse	64
Divide_c	Places evenly spaced point objects along the length or perimeter of a selected object	65
Quadrant_c	Snaps to a quadrant point of a circle and ellipse	66
Measure_c	Estimates the length of line or polyline	67
Dxf_In_c	Chooses a DXF file, requests scale and angle, and waits for its placement point. Imported entities are placed as individual objects in one undoable operation. Cancel discards the pending insertion. Scale and angle transform X/Y about the source origin; source Z values are retained. SetCoordinate(x, y, z) adds the supplied XYZ placement offset, so z is added to each imported elevation without being multiplied by the XY scale. Empty/corrupt source files and unrepresentable transforms are rejected before a pending insertion is published. A failed placement leaves the drawing unchanged and keeps the valid source available for a corrected point.	68
Shell_c	Dos Shell	69
Save_Load_c	Saves the current drawing and reloads the saved file, regenerating its content and clearing the drawing undo buffer. Reload occurs only after a successful save and only while the saved document and command remain current. An edit made by a save-completion handler is retained instead of being replaced by the saved snapshot.	70
Load_c	Requests an AVAX drawing (.avax, .avx or .blk) and replaces the current drawing only after parsing and any required save confirmation succeed. Cancellation or invalid input preserves the existing drawing.	71
Circle3p_c	Creates a circle by three points given from the user	72
SelectV_c	Selects vertices to make a temporary polygon	74
Area_c	Estimates the area of a selected closed polyline or temporary polygon	75
Calc_Text_c	Unused	76
LastSv_c	Selects the last temporary polygon	77
MakeSv_c	Creates a temporary polygon from a polyline object	78
Dim_c	Creates the dimension of a selected line	79
Ang_c	Creates the angle of two selected lines	80
Print_c	Switches to print to printer	81
NoText_On_c	Switches to the "NoText" drawing method in which Avax does not draw the text objects	82
NoText_Off_c	Switches off the "NoText" drawing method	83
CalcExtends_c	Calculates the Extends of the current drawing	84
Select_All_c	Selects all the drawing objects	85
Info_c	V3: displays read-only information for the selected objects: handle/type, layer, color, width/style, bounds, key, text, custom properties and child count. No dialog is shown without a selection or when DialogEnabled_p is False. Listed as unused in the R2 manual.	86
Grid_c	Configure independent X/Y grid spacing and enable the grid when accepted. Disabled dialogs or cancellation leave settings unchanged.	87
Grid_On_c	Displays a grid (line or dot)	88
Grid_Off_c	Switches "Grid" off	89
Snap_c	Configure independent X/Y coordinate snap intervals and enable snapping when accepted. Disabled dialogs or cancellation leave settings unchanged.	90
Change_Text_c	Changes the selected text	91
Parameter	**Description**	**Value**
Help_c	V3: opens read-only drawing-control help when DialogEnabled_p is True. Loads Help/Avax-CAD-GRE.rtf or Help/Avax-CAD-ENG.rtf from the configured options directory,	92

Parameter	Description	Value
	according to Language; uses built-in instructions when customized help is absent, too short, invalid or unreadable. Closing it preserves a command started while the dialog was open. Listed as unused in the R2 manual.	
Thaw_c	Thaws the current Layer	93
Freeze_c	Freezes the current Layer	94
Change_Layer_c	Changes the selected object's Layer	95
Change_Color_c	Changes the selected object's Color	96
Change_Width_c	Changes the selected object's Width	97
Change_Ltype_c	Changes the selected object's Line type	98
Change_Ptype_c	Changes the selected object's point type	99
Change_z_c	Changes the selected object's 3D elevation	100
Change_h_c	Changes the selected object's 3D height	101
Dxf_Out_c	Exports the selected objects to a DXF file relative to a picked XYZ base point. Select at least one object, choose the destination, then click the base point or supply it with SetCoordinate(Double, Double, Double). The ACADVersion_p property chooses the DXF version. No file is written until a base point is accepted. Cancel leaves the destination untouched. A failed write retains the request for retry; changing the drawing or selection before the point is accepted cancels the pending export. Export does not modify the drawing or its undo history.	102
Circle_c	Creates a Circle by center and radius	103
Fasttext_c	Switches to the "FastText" drawing method to be "on" when it is "off" or to be "off" when it is "on"	104
Ortho_c	Switches "Ortho" to be "on" when it is "off" or to be "off" when it is "on"	105
SetGrid_c	Disable an enabled grid without a dialog; otherwise request grid configuration.	106
SetSnap_c	Disable enabled coordinate snapping without a dialog; otherwise request snap configuration.	107
Wall_c	Creates a wall	108
UserLimits_c	Sets the drawing limits	109
Circle2p_c	Creates a Circle based on two endpoints of the diameter	111
Explode_Block_c	Explodes a Block	113
File_Label_c	Edits the current drawing caption. An accepted changed caption marks the drawing modified and is included when the drawing is saved. Cancel or an unchanged answer leaves the drawing untouched. Locked drawings are not edited. A dialog superseded by another drawing or command cannot overwrite its caption.	114
Block_Label_c	Edits the caption in the file referenced by the single selected block, as in R2. The block filename is resolved through the configured block directory; BLK, AVX and AVAX drawings are supported. The drawing geometry and inserted block remain unchanged. Cancel leaves the file untouched. An answer from a dialog superseded by another selection, drawing or command is discarded. The CAD host can also supply an explicit file path.	115
Save_Icon_c	Saves a 128 by 128 pixel Windows ICO image of the drawing. This command was marked unused in the v2 manual and is implemented in v3. Choose the destination in the save dialog.	116
Parameter	**Description**	**Value**
Insert_Avx_c	Inserts an AVAX drawing using the same file, scale, angle and placement sequence as Insert_c. The file picker accepts .avax, .avx and .blk files. The placed content is retained as a block. Scale and angle transform X/Y about the source origin; source Z values are retained. SetCoordinate(x, y, z) adds the supplied XYZ placement offset, so z is added to each imported elevation without being multiplied by the XY scale. Empty/corrupt source files and unrepresentable transforms are rejected before a pending insertion is published.	117

Parameter	Description	Value
	A failed placement leaves the drawing unchanged and keeps the valid source available for a corrected point.	
Setup_c	Opens Drawing settings when DialogEnabled_p is True. Configure grid and snap spacing, selection box, magnifier, scaling and dimension formatting. OK validates all rows before applying changed values; Cancel retains settings. Length factor and Angle factor accept any finite Double, including zero and negative values. Obsolete modal answers do not replace a newer command or document.	118
Change_Font_c	Changes the selected object's font	119
Close_c	Draws a line segment from the current position to the starting point of the polyline, creating a closed polyline	120
Save_As_c	Requests a new filename and saves the current drawing. The picker offers native .avax and legacy .avx files. Successful completion updates filename; cancellation leaves the drawing and filename unchanged.	121
PLT_Out_c	Exports to a PLT file	123
PIC_Out_c	Exports to a PIC file	124
Plane_c	Switches the "Plane" to be "on" if it is "off" or "off" if it is "on"	134
Plane_On_c	Switches "Plane" on.When "Plane" is on then every new closed polyline or polygon is a plane	135
Plane_Off_c	Switches "Plane" off	136
NoWidth_On_c	Ignores the object's width	140
NoWidth_Off_c	Does not ignore the object's width	141
Triangle_c	Creates a triangle	142
APL_Out_c	Exports to a APL file	143
Resize_c	Unused	200
CalcBounds_c	Calcs the drawing bounds	201
RePaint_c	RePaints drawing display	205
Wall3p_c	Creates a wall from three points	206
SelectPoints_c	Starts the point-request input mode. Direct assignment requests one point and raises SelectedPoints with an empty SelectId and one-based Double XYZ arrays, without adding geometry. For a caller-defined identifier, point count, seed arrays and preview flags, call the SelectPoints method. Cancel discards an unfinished request. Listed as unused in the R2 manual.	207
Cancel_Keep_Select_c	Unused	208
Zoom_Pan_c	Moves the drawing display in the current viewport	209
Zoom_Dynamic_c	Zooms to display the generated portion of the drawing with a view box which you can shrink or enlarge	210
Copy_Win	Copies the selected objects to the Clipboard	211
Paste_Win	Clipboard Paste	212
Cut_Win	Erases the selected objects and copies them to the Clipboard	213
Dwg_In_c	Chooses a DWG file, requests scale and angle, and waits for its placement point. Imported entities are placed as individual objects in one undoable operation. Cancel discards the pending insertion. Scale and angle transform X/Y about the source origin; source Z values are retained. SetCoordinate(x, y, z) adds the supplied XYZ placement offset, so z is added to each imported elevation without being multiplied by the XY scale. Empty/corrupt source files and unrepresentable transforms are rejected before a pending insertion is published. A failed placement leaves the drawing unchanged and keeps the valid source available for a corrected point.	214

Parameter	Description	Value
Dwg_Out_c	Exports the selected objects to an AutoCAD 2018 DWG file relative to a picked XYZ base point. Select the objects, choose the destination, then click a base point or call SetCoordinate. Only selected objects are exported. Cancel does not write a file; a failed write keeps the request for retry. Use the full PrintToDWG overload to specify another supported DWG version. Export does not modify drawing geometry or undo history.	215
Copy_In_c	Creates a new polyline inside a selected closed polyline or a temporary polygon at a specified distance	501
Copy_Out_c	Creates a new polyline outside a selected closed polyline or a temporary polygon at a specified distance	502
Border_c	Creates a new polyline to be the border of a selected polyline or a temporary polygon	503
Centroid_c	Creates a new point in the center of a selected closed polyline or a temporary polygon	504
Radiate_c	Creates new lines from the center of a selected closed polyline or a temporary polygon to the vertices	505
CSpline1_c	Creates a CSpline1 from a selected polyline or a temporary polygon	506
CSpline2_c	Creates a CSpline2 from a selected polyline or a temporary polygon	507
OBSpline_c	Creates a OBSpline from a selected polyline or a temporary polygon	508
Bezier_c	Creates a Bezier from a selected polyline or a temporary polygon	509
CBSpline_c	Creates a CBSpline from a selected polyline or a temporary polygon	510
Hatch_c	Creates Hatch lines from a selected polyline or a temporary polygon	511
V_Area_c	Esimates the area of a closed polyline or a temporary polygon	512
V_Measure_c	Measures the lenght of a line or a polyline	513
V_Dim_c	Puts the dimension of a selected line or polyline at a specified distance	514
Roof_c	Unused	515
Taxis_c	Cleans and orders paired polyline segments by exact XY endpoint contact while preserving XYZ endpoints. Removes zero-length edges and redundant duplicate or fully contained collinear edges. A wide segment is redundant only when its complete width profile also agrees. Different widths, width planes and elevation profiles remain independent. Reversed records retain their own endpoint widths; the replacement is one undoable operation.	516
P_Break_c	Breaks all selected lines, polylines and shapes into line entities, including XY intersections between different selected objects. Each output retains its own source attributes and interpolated Z. Open shapes stay open; closed shapes include their closing edge. With no object selection, temporary SelectV vertices use the current pen. Geometry/resources/handle capacity are prepared before a single undoable replacement. Completion and failure callbacks may start a new command or replace the drawing; that input is retained. Lifecycle guards apply. The splitter now uses exact binary predicates and rational parameters, with no fixed world-unit or angular tolerance. Distinct tiny fragments, shallow crossings and positive gaps remain distinct. XY intersections are rounded consistently for both sources; Z is independently interpolated from each source without intermediate overflow. Existing zero-length/projected-point lines are retained, and projected point segments can split other lines. Incomplete XYZ records and distinct cut points that collapse to one representable Double XY location fail before any edit. This is an explicit storage-precision limit, not silent fragment deletion. Wide source segments remain individual one-segment polylines. Their start/end width vectors are interpolated at each intersection from that occurrence's own profile, including coincident and reversed source records. Undo restores the original profiles.	517
VN_Count_c	Adds the counter number for each vertex of a selected polyline or a temporary polygon	518
VT_Count_c	Adds the alphabetical counter text for each vertex of a selected polyline or a temporary polygon	519
Area_3V_c	Adds a text with the estamations of triangle area	520
V_Triangle_c	Creates a triangle	521

Parameter	Description	Value
Up_c	Moves drawing display Up	-72
Down_c	Moves drawing display Down	-80
Left_c	Moves drawing display to the Left	-75
Right_c	Moves drawing display to the Right	-77
PgUp_c	Moves drawing display one page Up	-73
PgDn_c	Moves drawing display one page Down	-81
PgLeft_c	Moves drawing display one page to the Left	-115
PgRight_c	Moves drawing display one page to the Right	-116
PgPan_c	Pans drawing display	-117
PgDynamic_c	Zooms dynamic drawing display	-118

Additional declared values

Name	Value	Description
ArcStartEndCenter_c	218	R2 point order: START, END, approximate CENTER. The third XY point is projected onto the perpendicular bisector of the endpoint chord using rational arithmetic. Analytic mode stores a center-defined Curve with center Z from the first point. Coordinates use Double. Segmented arcs preserve/interpolate endpoint Z. Collinear three-point input permits retry. Startup while stopped, disabled, paused or locked preserves existing input. CurveArcOn=False (the default) constructs a segmented Polyline_i arc, using CircleSectors or ArcSegmentsLength with MinCircleSectors/MaxCircleSectors. CompleteInput/Enter, Close or switching/cancelling finishes the accepted path. CurveArcOn=True inserts an analytic Curve_i immediately and leaves the arc command ready to repeat. A preceding pending polyline is inserted together with that curve in one undoable batch. Switching Polyline to Arc keeps the final accepted vertex as the start; switching back restores the accepted prefix and discards only unfinished arc references. Changing arc variant keeps the start/prefix and resets the unfinished references. Drafting-aid toggles retain input. Preview uses the same representation. Geometry/capacity failures preserve accepted input and Redo; success consumes points before notifying NewUserItems/Change. A host callback may replace the document, start input or throw without stale cleanup. Generic ellipses and inferred centers/radii beyond finite Double remain open.
ArcStartIntermediateEnd_c	220	R2 point order: START, THROUGH-POINT, END. In analytic mode the three picked XYZ coordinates are retained in a three-point Curve record. Coordinates use Double. Segmented arcs preserve/interpolate endpoint Z. Collinear three-point input permits retry. Startup while stopped, disabled, paused or locked preserves existing input. CurveArcOn=False (the default) constructs a segmented Polyline_i arc, using CircleSectors or ArcSegmentsLength with MinCircleSectors/MaxCircleSectors. CompleteInput/Enter, Close or switching/cancelling finishes the accepted path. CurveArcOn=True inserts an analytic Curve_i immediately and leaves the arc command ready to repeat. A preceding pending polyline is inserted together with that curve in one undoable batch. Switching Polyline to Arc keeps the final accepted vertex as the start; switching back restores the accepted prefix and discards only unfinished arc references. Changing arc variant keeps the start/prefix and resets the unfinished references. Drafting-aid toggles retain input. Preview uses the same representation. Geometry/capacity failures preserve accepted input and Redo; success consumes points before notifying NewUserItems/Change. A host callback may replace the document, start input or throw without stale cleanup. Generic ellipses and inferred centers/radii beyond finite Double remain open.
Center_c	524	Requests a one-shot center snap on a circle, ellipse or circular arc for the next drawing point. Use it while supplying points to a drawing command.
Dim3p_c	221	Creates a linear dimension from a selected line or picked references, or dimensions the sides of selected/temporary polygon vertices as appropriate to the command. Distinct small XY references are retained. Preview and committed geometry use the same builder; output is prepared completely before one undoable creation. Polygon orientation and direction-side predicates are exact on stored Double coordinates. Offset, text and marks must be representable.
DimAligned_c	526	Creates a dimension parallel to the measured segment. Pick the first measured point, the second measured point, and the dimension-line position.
DimAngular_c	527	Creates an angular dimension. Pick the vertex, a point on the first ray, a point on the second ray, and the dimension-arc position. A zero ray or a second ray in the same direction is rejected; choose another point to continue.
DimDiametric_c	529	Creates a diameter dimension. Pick the circle center, a point on its circumference, and the text position. The measured value is twice the radius.
DimLinear_c	525	Creates a horizontal or vertical linear dimension. Pick the first measured point, the second measured point, and the dimension-line position. A position outside the X span and inside the Y span selects a vertical dimension; a vertical measured pair is always vertical.
DimRadial_c	528	Creates a radius dimension. Pick the circle center, a point on its circumference, and the text position. The radius must be greater than zero.
FreehandPolyline_c	523	Free writing retains sampled XY points, stroke elevation and pen settings. The preview does not add a duplicate final segment at the last sampled position and retains the stroke Z. Mouse-release and cancellation regressions run in hidden controls, including coordinates at 1e12.

Name	Value	Description
HatchPick_c	531	V3: previews a native associative hatch inside the closed boundary surrounding the pointer. Click to create one hatch. Chooses the smallest containing loop and retains its nested island loops; connected open pieces may form a closed loop together, while unrelated open components are ignored. Hidden/locked boundaries are excluded, including after layer-state changes. An outside pick reports an error and lets you retry; Escape cancels without editing geometry. Uses current hatch spacing/angle and the boundary elevation.
Magnify_c	216	Shows a magnifying lens at the pointer without changing the stored view, drawing objects or drawing history. MagnifyScale_p controls enlargement; MagnifyLensSize_p selects small (0), medium (1) or large (2). Escape removes the lens.
NativeHatch_c	530	V3: creates one editable associative hatch from the selected visible, unlocked closed boundaries. Source boundaries remain separate drawing objects. Uses HatchDistance_p and HatchAngle_p for spacing and angle; elevation comes from the boundary plane. Source edits update the associated hatch, and creation is one Undo step. Empty selection or open/incompatible boundaries are rejected without changing source geometry.
PolylineBackStep_c	522	Removes the last accepted vertex from an active Polyline, Polygon or vertex-selection command without cancelling that command. If no vertices remain, an unfinished tangent reference is cleared. Other commands are unchanged.
UserOsnap_c	219	Requests one custom UserOsnap result without replacing the current drawing command or its accepted points. The request works independently of the automatic snap bitmask. A picked object invokes UserOsnap;finite NewX/NewY with fNoResult=False supplies the point,and the request is consumed only when a point is accepted. A finite custom reply may be outside the search bounds. A replacement command or cancellation clears the old request. The Command property and direct DoAvaxAction dispatch use the same transparent request path.

Selection add/subtract state

SelectSub_c enables subtraction. This state remains active after ordinary Select_c, Cancel, selection windows and drawing/view commands. Inherited subtraction applies to object clicks and dragged or typed selection windows; an empty click leaves the selection intact. Use SelectAdd_c to restore addition. A new drawing starts in add mode. Trim_c and Extend_c switch to add mode after accepting their boundary selection. Read the SelectAddOn output of GetAvaxStatus to inspect this state: 0 means subtract and 1 means add.

Native hatch commands

NativeHatch_c creates an editable hatch from the current closed-boundary selection. HatchPick_c requests an interior point and previews the chosen region before creation. Both are v3 additions; Hatch_c retains the separate R2 line-hatching behavior. See Native hatch methods and examples for options, association, editing and code examples.

```
' Create a native hatch from the current boundary selection.
cAvax1.Command = Avax.AvaxCommand.NativeHatch_c

' Alternatively, let the user pick inside a closed boundary.
cAvax1.Command = Avax.AvaxCommand.HatchPick_c
```

Native dimension commands

DimLinear_c, DimAligned_c, DimAngular_c, DimRadial_c and DimDiametric_c are v3 additions. Each creates one editable dimension object. Choose points in the order described in the table above; the last point places the dimension line, arc or text. Preview does not change the drawing. A completed dimension is selected and can be undone in one step. Escape cancels unfinished input.

These commands copy the current AVAX dimension properties when the dimension is created and use Pen3DElevetion for its elevation. Subsequent default changes do not alter existing dimensions. See Native dimension methods and options for styles, measurement formatting and the corresponding Add_Dim_* methods.

VB.NET example

Call this procedure with a started control. It creates one dimension of each kind using Double coordinates. Coordinate and object snapping are disabled for these explicit coordinates.

```
Private Sub AddNativeDimensions(cAvax1 As Avax.cAvax)
```

```
    cAvax1.SetAvaxProperty(Avax.AvaxProp.AutoOSnap_p, 0)
    cAvax1.SetAvaxProperty(Avax.AvaxProp.SnapOn_p, False)
    cAvax1.Command = Avax.AvaxCommand.Ortho_Off_c
    For Each command As Avax.AvaxCommand In {
        Avax.AvaxCommand.DimLinear_c,
        Avax.AvaxCommand.DimAligned_c,
        Avax.AvaxCommand.DimAngular_c,
        Avax.AvaxCommand.DimRadial_c,
        Avax.AvaxCommand.DimDiametric_c}
        cAvax1.Command = command
        cAvax1.SetCoordinate(10.0, 10.0, 0.0)
        cAvax1.SetCoordinate(40.0, 10.0, 0.0)
        If command = Avax.AvaxCommand.DimAngular_c Then
            cAvax1.SetCoordinate(10.0, 40.0, 0.0)
        End If
        cAvax1.SetCoordinate(40.0, 40.0, 0.0)
    Next
End Sub
```

Window stretch

Stretch_c moves or stretches already selected lines and polylines, as in R2. Call SetBoundsStretch to define the XY window, then supply a reference point and a displacement endpoint. Vertices on or inside the window move by that XY displacement; their Z coordinates, paired segment occurrences, metadata and width profiles are retained.

When the window contains the complete geometric bounds of an object, that object translates as a whole, including nested content and resources. Curve containment uses analytic bounds rather than display/index padding. If a rigid shape, text/document frame, image or ordinary block is only partly covered, its internal coordinates are retained. A window containing no eligible geometry adds no edit and preserves the existing Redo branch.

The v3 native extensions update included dimension definition points and supported hatch boundary vertices. Associative hatches regenerate from their edited source boundaries in the same transaction. Partially included analytic hatch edges reject the complete batch; include the complete curved boundary to move it. Live external references use their envelope corners, including references nested inside a block; a window containing neither corner leaves the reference unchanged.

BeforeMove is raised once before acceptance and can change the targets or displacement, or cancel. PauseAvaxEvents suppresses it. The operation reads current source geometry and properties before preparing the complete batch. Callback exceptions propagate; a newer command or drawing started by a callback retains ownership. Cancellation or invalid preparation does not partially replace the drawing. A zero displacement adds no edit and retains the previous Redo branch.

Preview leaves the source unchanged. If transformed preview geometry cannot be represented, its ghost is omitted; the reference-to-cursor guide can remain visible. An invalid accepted point reports an error and leaves the reference point available for retry. A successful batch is one Undo/Redo operation. See StretchItems for proportional rectangle transforms of complete objects.

VB.NET example

Call this procedure with a started control. Only the line endpoint inside the window moves.

```
Public Sub WindowStretchExample(cAvax1 As Avax.cAvax)
    ' Use a started control. Disable constraints for these explicit coordinates.
    cAvax1.SetAvaxProperty(Avax.AvaxProp.AutoOSnap_p, 0)
    cAvax1.SetAvaxProperty(Avax.AvaxProp.SnapOn_p, False)
    cAvax1.Command = Avax.AvaxCommand.Ortho_Off_c

    Dim itemHandles As Integer() = {0, cAvax1.Add_Line(0.0, 0.0, 0.0, 10.0, 0.0, 0.0)}
    cAvax1.SelectItems(itemHandles)
    cAvax1.SetBoundsStretch(5.0, -1.0, 11.0, 1.0)
    cAvax1.Command = Avax.AvaxCommand.Stretch_c
    cAvax1.SetCoordinate(0.0, 0.0, 0.0) ' Reference point.
    cAvax1.SetCoordinate(2.0, 3.0, 0.0) ' Displacement: X=2, Y=3
    ' The first endpoint stays at (0,0,0); the second becomes (12,3,0).
End Sub
```

See Grid and coordinate snap commands for the executable R2 command-name mapping, cancellation and direct actions.

See Arc commands and segmentation for CurveArcOn_p, CircleSectors_p, ArcSegmentsLength_p and the segment limits.

See Drawing modes and settings for grid/snap switching and configuration, drawing/display modes, Double-valued distances, and independent zoom image memory and view history.

See Navigation commands and view history for the 50-view Previous history, page and small-step movement, work-area limits, and resize behavior.

See Selection commands and type filters for Add/Sub and window selection, Shift/Control behavior, automatic selection and type filtering.

See Line, Point and Rectangle commands for the basic drawing commands and complete Double-coordinate examples.

See Circle and Ellipse commands for the full circular drawing commands, radii and completion behavior.

See Polyline, Polygon and Close commands for connected XYZ chains, regular polygons from a center and circumradius, and exact polyline closure.

Center snap

Center_c requests the center object snap for the next construction point, following the same command pattern as EndPoint_c and MidPoint_c. It locates a nearby circle, ellipse or arc center without replacing the drawing command in progress.

```
cAvax1.Command = Avax.AvaxCommand.Center_c
```

Triangle and angle commands

API reference

Use these AvaxCommand values with the control's Command property after StartAvax. Coordinates and lengths use Double in v3; object handles and command values use Integer. SelectItems takes a one-based Integer array whose first slot is reserved. Interactive selection and selection windows select whole drawing objects.

Triangle_c

Command value 142 constructs a triangle on one selected Line_i base. With DialogEnabled_p=True, the parameter dialog asks for the length of the side from the first endpoint and the length of the side from the second endpoint. Both initially equal the base length. With dialogs disabled, these defaults produce an equilateral triangle.

After accepting the lengths, pick a point on the desired side of the base. The pick chooses the side; it is not the new vertex. The control calculates the apex and adds two Line_i objects, leaving the original base intact. Their creation is one Undo/Redo operation. The two lengths and the base must form a nondegenerate triangle. A point exactly on the base line does not complete the command; enter another side-choice point.

The base's first endpoint Z is used for the apex. Each original endpoint retains its own Z. This elevation retention extends the older planar output. V3 also accepts a selected ordinary polyline containing exactly one straight segment. With no selected objects, v3 offers an additional three-point mode: enter three vertices to create a closed triangle polyline.

VB.NET example

The supplied control must already be started. This example disables the parameter dialog and constructs an equilateral triangle above its base.

```
Option Strict On
Imports Avax

Public Module Help_SelectedTriangle
    Public Function Draw(cad As cAvax) As Boolean
        cad.Command = AvaxCommand.Cancel_c
        Dim dialogs As Object = False
        cad.SetAvaxProperty(AvaxProp.DialogEnabled_p, dialogs)
        Dim baseHandle As Integer = cad.Add_Line(0.0R, 0.0R, 7.0R, 10.0R, 0.0R, 7.0R)
        If baseHandle = 0 Then Return False
        Dim objectHandles As Integer() = {0, baseHandle}
        If Not cad.SelectItems(objectHandles) Then Return False
        cad.Command = AvaxCommand.Triangle_c
        Dim x As Double = 5.0R, y As Double = 1.0R, z As Double = 7.0R
        Return cad.SetCoordinate(x, y, z)
    End Function
End Module
```

Ang_c

Command value 80 adds an angular dimension for two selected Line_i objects. After selecting the lines and starting the command, enter one placement point. Their extended lines define the center; the placement point chooses the sector and its distance from that center determines the arc radius. The result contains an arc, marks and a formatted angle label. Both original lines remain intact, and the complete dimension is one Undo/Redo operation.

The lines must have distinct finite XY endpoints and a representable intersection. Parallel lines cannot define an angular dimension. Invalid placement leaves the accepted references available for another point. The command captures the original line coordinates when it starts. Dimension text, scale, formatting and pen settings use the control's current dimension properties.

V3 also accepts ordinary polylines when the complete selection contains exactly two straight segments. It does not select a subset of edges inside a longer polyline; use separate Line_i objects to select individual edges. With no selected objects, the additional three-point mode accepts the center, first ray point and second ray point. It creates the minor angle with a radius of 60 percent of the shorter ray. Output retains the center Z.

VB.NET example

This example creates a right-angle dimension between two selected lines. The final coordinates are Double variables because SetCoordinate parameters are ByRef.

```
Option Strict On
Imports Avax

Public Module Help_SelectedAngle
    Public Function Draw(cad As cAvax) As Boolean
        cad.Command = AvaxCommand.Cancel_c
        Dim first As Integer = cad.Add_Line(0.0R, 0.0R, 7.0R, 10.0R, 0.0R, 7.0R)
        Dim second As Integer = cad.Add_Line(0.0R, 0.0R, 7.0R, 0.0R, 10.0R, 7.0R)
        If first = 0 OrElse second = 0 Then Return False
        Dim objectHandles As Integer() = {0, first, second}
        If Not cad.SelectItems(objectHandles) Then Return False
        cad.Command = AvaxCommand.Ang_c
        Dim x As Double = 5.0R, y As Double = 5.0R, z As Double = 7.0R
        Return cad.SetCoordinate(x, y, z)
    End Function
End Module
```

Assigning Command has no return value. SetCoordinate returns True when a point is accepted. A completed operation returns to the idle selection mode controlled by SelectStay_p. Cancelling a parameter dialog creates no geometry. Invalid input does not partially insert a result. Use the documented OperationFailed event and LastOperationError to handle reported errors.

Center object snap

API reference

Snap to the exact center of a circle, circular arc, ellipse or elliptical arc, including supported geometry inside blocks. The cursor must approach the center itself, even when the center lies outside the visible arc. The aperture is measured in screen pixels; approaching the circumference alone does not acquire the center.

Center_c

Purpose

Request Center for the next accepted point without replacing the current drawing command or its previously accepted points. The override is consumed after one accepted point. Cancel/Escape clears a pending request.

```
cad.Command = AvaxCommand.Center_c  ' 524
```

Center_o

`AvaxOSnapType.Center_o = 10` is the resolved snap mode reported by `GetOSnapStatus()` and the existing `AutoOSnapType` event. It is an enum value, not the AutoOSnap bit mask. Event cancellation follows the existing snap event contract.

Automatic Center

Setting	Meaning
AutoOSnap_p bit 512	Enable automatic Center. Preserve other modes with mask Or 512; disable only Center with mask And Not 512.
AutoOSnapApproachInPixels_p	Screen-pixel aperture around the projected center; independent of drawing units and zoom.
Command = AvaxCommand.Center_c	One-point override. Does not replace the saved automatic snap mask.

Example

Call this procedure on the owning UI thread with a control already initialized by `StartAvax()`. Handles must belong to that control. File paths are supplied by the caller.

```
Option Strict On
Imports System
Imports Avax

Public Module Help_CenterSnap
    Public Sub Run(cad As cAvax)
        Dim key As AvaxProp = AvaxProp.AutoOSnap_p
        Dim mask As Integer = CInt(cad.GetAvaxProperty(key))
        Dim enabled As Object = mask Or 512
        If Not cad.SetAvaxProperty(key, enabled) Then Throw New InvalidOperationException("Cannot enable
    Center snap.")
        cad.Command = AvaxCommand.Line_c
        ' Enter the first point, then request Center for the next point.
        cad.Command = AvaxCommand.Center_c
        ' Move the cursor near the desired center and click.
        ' To disable only automatic Center later:
        ' Dim disabled As Object = CInt(cad.GetAvaxProperty(key)) And Not 512
        ' cad.SetAvaxProperty(key, disabled)
    End Sub
End Module
```

AVAX-CAD R3

Use the Center object-snap button (command 524), or enable Center in Auto OSnap for continuous acquisition. Existing Endpoint, Midpoint and other snap selections remain available.

Related API: Command, AvaxCommand, AvaxOSnapType.

AvaxFillType

Selects the fill used by a closed shape.

Parameter	Description	Value
Solid_f	API Solid	0
Pattern_f	API Pattern	1
No_f	No fill	2
Vector_f	Fill with Avax Vectors	3
BMP_f	Fill with a Bitmap file	4
WMF_f	Fill with a WMF file	5
PatACAD_f	Fill with a AutoCad PAT file	6
ExVector_f	Fill with Avax ExVectors	7

Additional declared values

Name	Value	Description
Control_f	9	Embedded-control fill. The pattern describes a button, text field, combo, checkbox, option, picture, label or form; lFlags and lHatchIndex identify its parent and child. The Integer enum value is 9.
Document_f	11	Rich document content inside a shape. The associated text layout provides the document width, height and position. The Integer enum value is 11.
PatSVG_f	10	SVG pattern fill, described by the fill pattern and clipped to the shape boundary. The Integer enum value is 10.
Text_f	8	Text content inside a shape, using its text layout and font attributes. The Integer enum value is 8.

Native hatch methods and options

API reference

A native hatch is one `AvaxItemType.Hatch_i` object with one handle, analytical boundary loops, island rules and pattern/fill settings. Existing hatch commands and `Add_LinearHatch` retain their behavior. Call these public `cAvax` methods on its owning UI thread after StartAvax.

Creation and boundaries

When options is omitted, Add_Hatch(boundaryHandles) copies HatchAngle_p, HatchDistance_p and the current pen appearance. Boundary elevation replaces the initial pen elevation. Explicit options are copied as supplied; use Get_HatchDefaults() as a starting point. The new API accepts ordinary zero-based arrays; zero/negative entries are ignored and repeated handles are deduplicated. Legacy MoveItems/SelectItems arrays still reserve element 0.

Sources must form closed, simple, nonintersecting loops in one plane parallel to XY: rectangles, circles, ellipses, closed polylines/shapes, or connected line/arc segments. Include inner loops for holes. Source handles refer to the same drawing. Pick-inside searches existing closed contours, not arbitrary regions formed by intersecting lines.

Add_Hatch

Get_HatchDefaults

Get_Hatch

Get_HatchArea

Set_Hatch

Set_HatchBoundaries

Load_HatchPattern

AvaxHatchOptions

Property / type	Meaning and default
AngleDeg As Double	Degrees. Absolute UserDefined angle; added to PAT row angles. New options default 45; Get_HatchDefaults reads HatchAngle_p.
Associative As Boolean	Default True; sources are required for actual association.
BackgroundColor As Integer	RGB 0xRRGGBB; -1 (default) disables pattern background.
DoublePattern As Boolean	Default False; adds a perpendicular line family to UserDefined.
DrawOrder As Integer	Default 0; lower values draw behind when ShapeZorderOn is enabled.
Elevation As Double	XY-plane elevation, default 0; handle-based creation takes source elevation.
GradientAngleDeg As Double	Gradient rotation in degrees, default 0.
GradientColor1 As Integer	RGB 0xRRGGBB; default red &HFF0000.
GradientColor2 As Integer	RGB 0xRRGGBB; default cyan &H00FFFF.
GradientName As String	Default LINEAR. Also CYLINDER, INVCYLINDER, SPHERICAL, INVSPHERICAL, HEMISPHERICAL, INVHEMISPHERICAL, CURVED, INVCURVED.
GradientShift As Double	0-1; default 0.
IslandStyle As AvaxHatchIslandStyle	Normal (default): alternating filled/empty nested loops. Outer: outer regions with immediate holes. Ignore: fill through internal loops.
Kind As AvaxHatchKind	Default UserDefined. Other values: Pattern, Solid, Gradient.
OriginX As Double	Pattern origin/phase X, default 0.
OriginY As Double	Pattern origin/phase Y, default 0.
PatternLines As AvaxHatchPatternLine()	Embedded PAT definition, initially empty. Each row: angle,x,y,shift,offset,dashes. Angles are degrees, offset must be nonzero; positive dash is drawn, negative is a gap, zero a dot.
PatternName As String	Default ANSI31, which is built in. Load other patterns with Load_HatchPattern.
Scale As Double	Positive PAT scale, default 1. Does not replace UserDefined Spacing.
Spacing As Double	Positive UserDefined line spacing in drawing units; new options 0.25, defaults helper reads HatchDistance_p. Does not rescale PAT rows.
Transparency As Integer	Integer 0-90 percent, default 0.

AvaxHatchData, loops and edges

Type/member	Meaning
AvaxHatchData.Options	AvaxHatchOptions; must be present.
AvaxHatchData.Loops	Nonempty AvaxHatchLoop() of closed boundaries.
AvaxHatchData.AssociationStatus	Status text calculated by the control. Read it for feedback; do not use it to enable association.
AvaxHatchLoop.Edges	Ordered nonempty AvaxHatchEdge(); every edge endpoint joins the next, including closure.
AvaxHatchLoop.SourceHandles	Source handles in this drawing; empty for independent geometry.
AvaxHatchEdge.Kind = 1	Line: Values={x1,y1,x2,y2}.
AvaxHatchEdge.Kind = 2	Conic: Values={cx,cy,ux,uy,vx,vy,start,sweep}; C+U*cos(t)+V*sin(t). Start/sweep are radians, unlike options/PAT angles. |sweep| <= 2*pi.
AvaxHatchEdge.Kind = 4	Rational spline: Values contains x,y,positive-weight triples; Degree 1-10, ordered Knots with count=control-point-count+Degree+1, and Periodic flag.
AvaxHatchPatternLine.Values	Double() PAT row as described above.
Clone()	Data, options, loops and edges provide detached deep copies.

All five data classes have a parameterless constructor. New AvaxHatchData contains new default Options, empty Loops and an empty AssociationStatus. New loops, edges and pattern rows contain empty arrays; complete the required geometry before passing them to Add_Hatch or Set_Hatch. Clone() on a complete data, options, loop or edge object copies its nested arrays, including pattern rows, source handles, coordinates and knots. Modifying the copy does not edit the drawing; apply it with Set_Hatch. The control validates options and geometry when adding or replacing a hatch, and rejects invalid definitions without changing the existing object.

Example

Call this procedure on the owning UI thread with a control already initialized by `StartAvax()`. The handle must belong to that control.

```
Option Strict On
Imports System
Imports Avax

Public Module Help_CloneHatch
    Public Sub Run(cad As cAvax, handle As Integer)
        Dim original As AvaxHatchData = cad.Get_Hatch(handle)
        If original Is Nothing Then Throw New ArgumentException("Native hatch required.")
        Dim edited As AvaxHatchData = original.Clone()
        edited.Options.AngleDeg = 90
        If Not cad.Set_Hatch(handle, edited) Then Throw New InvalidOperationException("Hatch update
    failed.")
    End Sub
End Module
```

Example

Call this procedure on the owning UI thread with a control already initialized by `StartAvax()`. Handles must belong to that control. File paths are supplied by the caller.

```
Option Strict On
Imports System
Imports Avax

Public Module Help_ExplicitHatch
    Public Sub Run(cad As cAvax)
        Dim loopValue As New AvaxHatchLoop With {.Edges = New AvaxHatchEdge() {
            New AvaxHatchEdge With {.Kind = 1, .Values = New Double() {0, 0, 20, 0}},
            New AvaxHatchEdge With {.Kind = 1, .Values = New Double() {20, 0, 20, 10}},
            New AvaxHatchEdge With {.Kind = 1, .Values = New Double() {20, 10, 0, 10}},
            New AvaxHatchEdge With {.Kind = 1, .Values = New Double() {0, 10, 0, 0}}}}
        Dim data As New AvaxHatchData With {
            .Options = New AvaxHatchOptions With {.Kind = AvaxHatchKind.Solid, .Associative = False},
            .Loops = New AvaxHatchLoop() {loopValue}}
        If cad.Add_Hatch(data) = 0 Then Throw New InvalidOperationException("Hatch creation failed.")
    End Sub
End Module
```

AvaxHatchKind

Member	Value
Gradient	3
Pattern	1
Solid	2
UserDefined	0

AvaxHatchIslandStyle

Member	Value
Ignore	2
Normal	0
Outer	1

Association, transforms and history

Moving/editing source boundaries rebuilds the hatch in the same undo operation. Transforming the hatch and sources together preserves association without double transformation. Transforming only the hatch makes it independent. Copy/clipboard/arrays with sources remap to the new boundary handles; hatch-only copies are independent. Deleting/opening a source keeps the last valid hatch geometry and disconnects it; Undo restores both geometry and links. Deleting a hatch does not delete its sources. Hatches placed inside newly created/imported blocks are independent of internal source handles. A block may itself be a source boundary for a hatch in the drawing.

Nonuniform scaling can convert a UserDefined pattern to an embedded Pattern. For partial stretch of independent curved boundaries, edit the associated source or include the complete curve. Color, width, line style and layer use the normal object-edit API. AVAX-CAD R3 offers NativeHatch (530), HatchPick (531), selected-hatch properties, Load PAT and source-handle editing.

Storage and DXF/DWG

Export writes HATCH entities with line/arc/ellipse/spline paths, PAT or gradient, transparency, background and draw order. Association references/reactors are written when the sources are also exported; hatch-only export is independent. Import reconnects valid sources after reading the objects. Use the PrintToDXF overload with an explicit version of DXF 2004 or newer; 2018 works for both dimension and hatch objects. The one-argument overload retains its legacy R10 default and cannot store native HATCH entities. DXF/DWG preserve the island style and every original boundary loop. The current .avax writer uses format 5 (reads 1-4); use matching current EXE/DLL files. AVX/XML and native clipboard preserve additional native data.

The public version argument is `AvaxACADVersion`. For the extended 2018 version supported by this build, pass `CType(12, AvaxACADVersion)` as shown below; the internal extended-version constants are not public DLL members.

Example

Call this procedure on the owning UI thread with a control already initialized by `StartAvax()`. Handles must belong to that control. File paths are supplied by the caller.

```
Option Strict On
Imports System
Imports Avax

Public Module Help_NativeExchange
    Public Sub Run(cad As cAvax, dxfPath As String, dwgPath As String)
        Dim version As AvaxACADVersion = CType(12, AvaxACADVersion) ' AutoCAD 2018
        If Not cad.PrintToDXF(dxfPath, False, 0, 0, 0, version) Then
            Throw New InvalidOperationException("DXF export failed.")
        End If
        If Not cad.PrintToDWG(dwgPath, False, 0, 0, 0, version) Then
            Throw New InvalidOperationException("DWG export failed.")
        End If
    End Sub
End Module
```

For DXF/DWG export, OnlyLinesACADExportOn converts opaque PAT hatches to explicit clipped rows instead of HATCH. Row shifts, dashes, holes and elevation are retained; curved boundaries use their sampled display region. Solid fills and independent visible borders are retained. Transparent hatches require native HATCH export with OnlyLinesACADExportOn disabled. A conversion that cannot retain supported appearance, or that would create unsafe output density, fails without replacing an existing destination. VectorTextExportOn alone does not flatten a hatch that contains no text.

AvaxImageEffectKind Member Reference

API reference

AvaxImageEffectKind

Purpose

Identifies the image adjustment performed by an AvaxImageEffect entry. Set the entry's Kind to one of these enum values and its Amount to a value in the corresponding range. Effects are processed in array order. Use GetImageEffectDefinitions to read the ranges and suggested defaults at run time.

Declaration

```
Public Enum AvaxImageEffectKind
    Brightness = 0
    Contrast = 1
    Exposure = 2
    Gamma = 3
    Saturation = 4
    Hue = 5
    Temperature = 6
    Tint = 7
    Grayscale = 8
    Sepia = 9
    Invert = 10
    Blur = 11
    Sharpen = 12
    Pixelate = 13
    Posterize = 14
    Vignette = 15
    Crop = 16
End Enum
```

VB.NET example

For example, `New Avax.AvaxImageEffect(Avax.AvaxImageEffectKind.Exposure, 0.5)` creates an enabled effect with a half-stop exposure increase. Pass an array of these values to SetImageEffects or RenderImagePreview. Constructing an effect creates a detached value; applying or previewing it validates its amount.

Amounts must be finite Double values inside the inclusive ranges below. Pixelate and Posterize require whole numbers. The enum value is the effect identifier; it is separate from its Amount. See Effect value objects for Kind, Amount, Enabled and Clone.

Values, purposes and supported amounts

The table lists every AvaxImageEffectKind value. RGB adjustments clamp their output to the channel range. Colour adjustments preserve pixel alpha; Blur, Pixelate and Crop account for alpha when combining pixels. Amounts operate on the result of preceding enabled entries, so changing list order can change the result.

Kind (enum value)	Amount range and units	Default	Increment	Effect
Brightness (0)	-100 to 100, percentage adjustment	0	1	Adds Amount multiplied by 2.55 to each RGB channel. Positive values brighten; negative values darken.
Contrast (1)	-100 to 100, percentage adjustment	0	1	Scales channel distance from the midpoint 127.5 by 1 + Amount / 100. Zero is unchanged; -100 produces the midpoint colour.
Exposure (2)	-5 to 5, exposure stops	0	0.1	Multiplies RGB channels by 2 raised to Amount. One stop doubles channel values before clamping.
Gamma (3)	0.1 to 5, gamma exponent parameter	1	0.1	Applies output = 255 multiplied by (input / 255) raised to 1 / Amount. One is unchanged; values above one lighten midtones.

Kind (enum value)	Amount range and units	Default	Increment	Effect
Saturation (4)	-100 to 100, percentage adjustment	0	1	Scales colour distance from luminance by 1 + Amount / 100. At -100 the result is grayscale; positive values increase saturation.
Hue (5)	-180 to 180, degrees	0	1	Rotates hue using an RGB colour-rotation matrix. Zero is unchanged.
Temperature (6)	-100 to 100, relative colour adjustment	0	1	Positive values warm the image by adding red and reducing blue; negative values cool it.
Tint (7)	-100 to 100, relative colour adjustment	0	1	Positive values shift toward magenta by increasing red/blue and reducing green. Negative values shift toward green.
Grayscale (8)	0 to 100, blend percent	100	1	Blends RGB toward luminance. Zero is unchanged; 100 is fully grayscale.
Sepia (9)	0 to 100, blend percent	100	1	Blends the image with a sepia colour conversion. Zero is unchanged; 100 applies the full conversion.
Invert (10)	0 to 100, blend percent	100	1	Blends each RGB channel with 255 minus that channel. At 100 colours are fully inverted.
Blur (11)	0 to 50, pixel-based blur strength	2	0.1	Softens the image with repeated box-blur passes. Zero is unchanged. The pixel-based strength is scaled down for a reduced-resolution preview.
Sharpen (12)	0 to 300, sharpening strength percent	50	1	Adds a scaled difference between the current image and a small blurred image. Zero is unchanged; 100 adds the full difference.
Pixelate (13)	1 to 128, whole source-pixel block size	8	1	Replaces square blocks with their averaged colour. Block size scales with preview resolution.
Posterize (14)	2 to 32, whole levels per RGB channel	6	1	Quantizes each RGB channel to the requested number of equally spaced levels.
Vignette (15)	0 to 100, darkening strength percent	50	1	Darkens the image toward its corners. Zero is unchanged.
Crop (16)	0 to 40, inset percent on each side	0	1	Resamples the centred inner rectangle into the same pixel canvas. Ten removes 10% at the left, right, top and bottom, then fills the unchanged canvas with the remaining 80% width and height. Stored source bytes and drawing geometry are retained.

AvaxItemType

Identifies the kind of a stored drawing object.

Parameter	Description	Value
Any_i	Any type	0
Point_i	Point	1
Line_i	Line	2
Polyline_i	Polyline	3
Block_i	Block	4
Ellipse_i	Ellipse	5
Text_i	Text	6
Shape_i	Shape	7
ExLines_i	External Lines	8
Data_i	Data	9

Additional declared values

Name	Value	Description
Curve_i	11	An analytic curve object, for example the circular arc created by Add_CurveArc. It retains a curve definition rather than a sequence of ordinary polyline segments.
Dimension_i	14	An editable native dimension object created by Add_Dim_Linear, Add_Dim_Aligned, Add_Dim_Angular, Add_Dim_Radial or Add_Dim_Diametric. Added in v3.
Document_i	13	A formatted document object created by Add_Document, with document width, text height and rich-text content.
Hatch_i	15	An editable native hatch object created by Add_Hatch or the NativeHatch_c/HatchPick_c commands. Its boundary loops and fill options are available through the native hatch methods. Added in v3.

Pass an AvaxItemType value to GetAllHandlesArr to filter the root objects in a drawing. Any_i includes all root types, including invisible Data_i objects. ItemType and GetAvaxItemType return the stored type for a handle; Any_i is also their result when the handle is not found or the control is inactive. A block is returned as a Block_i root, without separately enumerating its children.

The values retain their R2 numbers. Dimension_i and Hatch_i identify the editable native objects added in v3. These identify an object category; their geometry and options are described by the corresponding creation methods.

VB.NET example

```
Dim itemHandles() As Integer = Nothing
Dim itemKind As Avax.AvaxItemType = Avax.AvaxItemType.Line_i
Dim sortByZOrder As Boolean = False
Dim count As Integer = cAvax1.GetAllHandlesArr(itemHandles, itemKind, sortByZOrder)
For index As Integer = 1 To count
    Dim kind As Avax.AvaxItemType = cAvax1.ItemType(itemHandles(index))
    System.Diagnostics.Debug.WriteLine(kind.ToString())
Next
```

AvaxLanguage

Selects the language used by control dialogs and messages.

Parameter	Description	Value
Greek	Greek language, country code 30. Selects MSG1.OPT in the options directory for the corresponding R2 control messages. Missing or invalid translations fall back to MSG2.OPT and then built-in English. Greek also selects the Greek legacy vector-font mapping. Direct Language assignment to Greek selects ISO-8859-7 for XML; Language_p retains the explicit encoding.	30
English	English language, country code 1. Selects MSG2.OPT in the options directory for the corresponding R2 control messages. Missing or invalid translations fall back to MSG2.OPT and then built-in English. Uses the non-Greek legacy vector-font mapping and retains the current XML encoding.	1
Italian	Italian language, country code 39. Selects MSG3.OPT in the options directory for the corresponding R2 control messages. Missing or invalid translations fall back to MSG2.OPT and then built-in English. Uses the non-Greek legacy vector-font mapping and retains the current XML encoding.	39
French	French language, country code 33. Selects MSG4.OPT in the options directory for the corresponding R2 control messages. Missing or invalid translations fall back to MSG2.OPT and then built-in English. Uses the non-Greek legacy vector-font mapping and retains the current XML encoding.	33

Additional declared values

Name	Value	Description
German	49	German language, country code 49. Selects MSG5.OPT in the options directory for the corresponding R2 control messages. Missing or invalid translations fall back to MSG2.OPT and then built-in English. Uses the non-Greek legacy vector-font mapping and retains the current XML encoding.

Message catalogs

Set the options folder with DirsSet and assign Language or Language_p. Each control reads its own files. Assigning the same language reloads edited files; changing the options directory also reloads the catalog.

Language	Country code	File
Greek	30	MSG1.OPT
English	1	MSG2.OPT
Italian	39	MSG3.OPT
French	33	MSG4.OPT
German	49	MSG5.OPT

The first line is the number of messages. Each following line contains one message enclosed in double quotes. Message numbers start at 1. Preserve spaces inside the quotes. Entry 42 contains seven equal-width fields in order: layer, color, width, line type, point type, elevation and height. The supplied R2 files contain 74 entries. Empty quoted entries are valid; a truncated or malformed file is rejected as a whole.

Original Greek files use DOS code page 737; the other supplied files use Windows-1252. UTF-8, UTF-8 with a BOM and UTF-16 with a BOM are also accepted for Unicode customization. Files are read and released immediately. Missing entries fall back to the English catalog, then built-in R2 English; an unavailable country never retains messages from the preceding language.

The catalog supplies the corresponding original control captions and messages. Labels added by a host application remain under its control. Changing language preserves drawing contents and the command in progress. Direct Language = AvaxLanguage.Greek also selects ISO-8859-7 XML encoding, as in R2. The following example uses Language_p to retain an explicitly selected XML encoding.

VB.NET example

```
Option Strict On
Imports System
Imports Avax

Public Module LanguageCatalogExample
    Public Sub Configure(cad As cAvax, optionsFolder As String, language As AvaxLanguage)
        Dim dataFolder As String = "", libraryFolder As String = "", commandFolder As String = ""
        cad.DirsSet(optionsFolder, dataFolder, libraryFolder, commandFolder)
        ' Generic assignment retains an explicitly selected XML encoding.
        If Not cad.SetAvaxProperty(AvaxProp.Language, CObj(CInt(language))) Then
            Throw New InvalidOperationException("The language could not be selected.")
        End If
    End Sub

    Public Sub Reload(cad As cAvax)
        ' Reassigning the same language reloads edited MSGn.OPT files.
        If Not cad.SetAvaxProperty(AvaxProp.Language, CObj(CInt(cad.Language))) Then
            Throw New InvalidOperationException("The messages could not be reloaded.")
        End If
    End Sub
End Module
```

AvaxOSnapType

Identifies an object-snap mode or result.

Parameter	Description	Value
None_o	None	0
Endpoint_o	Snap at a start or end point. Enum value 1; use automatic-snap mask bit 1 or Command = AvaxCommand.Endpoint_c for a one-shot aid. The aid preserves the active construction command and accepted first point.	1
Midpoint_o	At the middle point	2
Nearest_o	At the nearest point	3
Perpendicular_o	At a perpendicular point	4
Intersection_o	Snap at an intersection of supported drawing geometry. Enum value 5; use automatic-snap mask bit 16 or Command = AvaxCommand.Intersec_c for a one-shot aid. The legacy command name is Intersec_c. Snapping does not change the intersecting objects.	5
Tangent_o	At a tangent point	6
Quadrant_o	At a quadrant	7
Node_o	Snap at an explicit point item. Enum value 8; use automatic-snap mask bit 128 or Command = AvaxCommand.Node_c for a one-shot aid. The enum value itself is not the automatic-snap bit mask.	8
User_o	User defined	9

Additional declared values

Name	Value	Description
Center_o	10	Snap at the mathematical center of a circle, ellipse or circular arc. This v3 extension has value 10 and leaves the R2 values unchanged. Use automatic-snap mask bit 512 or Command = AvaxCommand.Center_c for a one-shot aid. GetOSnapStatus and AutoOSnapType identify it as Center_o; coordinates use Double in v3.

Center_o

Identifies a center snap to a circle, ellipse or arc. The value is 10. In the AutoOSnap bit mask, its bit is 1 shifted left by (10 - 1), or 512; the enum value itself is not the mask.

AvaxPaperSize

Selects a standard paper size for page layout and printing.

Parameter	Description	Value
epsLetter	Letter, 8 1/2 x 11 in.	1
epsLetterSmall	Letter Small, 8 1/2 x 11 in.	2
epsTabloid	Tabloid, 11 x 17 in.	3
epsLedger	Ledger, 17 x 11 in.	4
epsLegal	Legal, 8 1/2 x 14 in.	5
epsStatement	Statement, 5 1/2 x 8 1/2 in.	6
epsExecutive	Executive, 7 1/4 x 10 1/2 in. (184.15 x 266.7 mm).	7
epsA3	A3, 297 x 420 mm	8
epsA4	A4, 210 x 297 mm	9
epsA4Small	A4 Small, 210 x 297 mm	10
epsA5	A5, 148 x 210 mm	11
epsB4	B4, 250 x 354 mm	12
epsB5	B5, 182 x 257 mm	13
epsFolio	Folio, 8 1/2 x 13 in.	14
epsQuarto	Quarto, 215 x 275 mm	15
eps10x14	10 x 14 in. (254 x 355.6 mm).	16
eps11x17	11 x 17 in. (279.4 x 431.8 mm).	17
epsNote	Note, 8 1/2 x 11 in.	18
epsEnv9	Envelope #9, 3 7/8 x 8 7/8 in.	19
epsEnv10	Envelope #10, 4 1/8 x 9 1/2 in.	20
epsEnv11	Envelope #11, 4 1/2 x 10 3/8 in.	21
epsEnv12	Envelope #12, 4 3/4 x 11 in. (120.65 x 279.4 mm).	22
epsEnv14	Envelope #14, 5 x 11 1/2 in.	23
epsCSheet	C size sheet, 17 x 22 in. (431.8 x 558.8 mm).	24
epsDSheet	D size sheet, 22 x 34 in. (558.8 x 863.6 mm).	25
epsESheet	E size sheet, 34 x 44 in. (863.6 x 1117.6 mm).	26
epsEnvDL	Envelope DL, 110 x 220 mm	27
epsEnvC3	Envelope C3, 324 x 458 mm	28
epsEnvC4	Envelope C4, 229 x 324 mm	29
epsEnvC5	Envelope C5, 162 x 229 mm	30
epsEnvC6	Envelope C6, 114 x 162 mm	31
epsEnvC65	Envelope C65, 114 x 229 mm	32
epsEnvB4	Envelope B4, 250 x 353 mm	33
epsEnvB5	Envelope B5, 176 x 250 mm	34
epsEnvB6	Envelope B6, 176 x 125 mm	35

Parameter	Description	Value
epsEnvItaly	Envelope, 110 x 230 mm	36
epsenvmonarch	Envelope Monarch, 3 7/8 x 7 1/2 in.	37
epsEnvPersonal	Envelope, 3 5/8 x 6 1/2 in.	38
epsFanfoldUS	U.S. Standard Fanfold, 14 7/8 x 11 in.	39
epsFanfoldStdGerman	German Standard Fanfold, 8 1/2 x 12 in.	40
epsFanfoldLglGerman	German Legal Fanfold, 8 1/2 x 13 in.	41
epsUser	User-defined sheet dimensions; see the custom-size settings below.	256

Additional declared values

Name	Value	Description
epsA0	272	A0, 841 x 1189 mm.
epsA1	271	A1, 594 x 841 mm.
epsA2	66	A2, 420 x 594 mm.

The numeric values retain their R2 meanings. Dimensions in this table are nominal sheet sizes. PrintToPDF and PrintToPrinter swap the nominal width and height when fLandScape is True. PageOrientation explicitly selects the short or long side as the workspace page width.

epsUser uses PrintPaperX_MM_p and PrintPaperY_MM_p for the legacy print methods; workspace pages use PageCustomWidthMM and PageCustomHeightMM. The Executive, Envelope #12 and C/D/E sheet dimensions correct errors in the old size table.

AvaxPathNodeMode

API reference

AvaxPathNodeMode

Purpose

Specifies how paired cubic Bezier tangent controls behave when one control is moved at a path node. The enumeration uses Integer values.

Declaration

```
Public Enum AvaxPathNodeMode As Integer
    Corner = 0
    Smooth = 1
    Symmetric = 2
End Enum
```

Values

Name	Value	Meaning
Corner	0	The two tangent controls move independently. This is the default for a new figure or segment.
Smooth	1	Moving one tangent control keeps the opposite control collinear through the node while retaining the opposite control's length.
Symmetric	2	Moving one tangent control keeps the opposite control collinear through the node and gives it the same length as the moved control.

Usage

AvaxPathFigure.NodeMode belongs to the figure's starting node. AvaxPathSegment.NodeMode belongs to that segment's endpoint. SetPathNodeMode applies a mode to an existing object's zero-based figure/node index. Node 0 is the start; node n greater than 0 is the endpoint of Segments(n - 1).

The mode is used when paired Bezier controls are edited. Assigning a mode alone does not reposition existing controls or turn a line or arc into a Bezier segment. The mode itself has no length or angle unit; control coordinates use Double drawing units.

VB.NET example

```
Option Strict On
Public Module PathNodeModeExample
    Public Function MakeSymmetricStart() As Avax.AvaxPathFigure
        Dim figure As New Avax.AvaxPathFigure With {
            .X = 0.0, .Y = 0.0, .Closed = True,
            .NodeMode = Avax.AvaxPathNodeMode.Symmetric}
        figure.Segments.Add(Avax.AvaxPathSegment.BezierTo(
            10.0, 20.0, 20.0, 20.0, 30.0, 0.0))
        figure.Segments.Add(Avax.AvaxPathSegment.BezierTo(
            20.0, -20.0, -10.0, -20.0, 0.0, 0.0))
        Return figure
    End Function
End Module
```

AvaxPathSegmentKind

API reference

AvaxPathSegmentKind

Purpose

Identifies a compound path segment as a straight line, elliptical arc or cubic Bezier curve. It also selects the next segment type for interactive compound-path input. The enumeration uses Integer values.

Declaration

```
Public Enum AvaxPathSegmentKind As Integer
    Line = 0
    Arc = 1
    Bezier = 2
End Enum
```

Values

Name	Value	Meaning
Line	0	A straight segment to endpoint X/Y. Default for a new segment and the initial type after BeginCompoundPath.
Arc	1	An elliptical arc defined by its centre, U/V axis vectors, StartAngle and SweepAngle. The endpoint X/Y must agree with this definition. Interactive input creates a circular arc from an intermediate point and an endpoint.
Bezier	2	A cubic Bezier segment with Control1X/Y, Control2X/Y and endpoint X/Y. Interactive input takes the two controls, then the endpoint.

Usage and units

Every segment starts at the preceding segment's endpoint, or at the figure's starting X/Y for the first segment. Coordinates and axis vectors use Double drawing units. Arc StartAngle and signed SweepAngle use degrees; SweepAngle must be nonzero and no greater than 360 degrees in magnitude.

Use LineTo, ArcTo or BezierTo to create a correctly initialized segment. Assigning AvaxPathSegment.Kind alone does not recalculate its controls, arc axes or endpoint. ChangePathSegmentKind replaces a committed segment and constructs the new geometry. CompoundPathSegmentKind selects future interactive input; assigning that property discards controls belonging to an incomplete segment, even when assigning the same value.

VB.NET example

```
Option Strict On
Public Module PathSegmentKindExample
    Public Function MakeQuarterArc() As Avax.AvaxPathSegment
        Dim segment As Avax.AvaxPathSegment =
            Avax.AvaxPathSegment.ArcTo(
                0.0, 0.0, 10.0, 10.0, 0.0, 90.0)
        If segment.Kind = Avax.AvaxPathSegmentKind.Arc Then
            System.Diagnostics.Debug.WriteLine(segment.SweepAngle)
        End If
        Return segment
    End Function
End Module
```

AvaxPolygonsComp

Describes the geometric relationship returned when comparing polygons.

Parameter	Description	Value
In_p	Value 1: the entire secondary contour is inside or on the main boundary. Coincident polygons and contained boundary contact are In_p. ComparePolygons also returns this value for the same existing drawing handle.	1
Out_p	Value 2: the secondary contour is outside the main polygon without boundary contact. A secondary polygon enclosing the main polygon is still Out_p when their boundaries do not touch; the result describes the secondary contour, not symmetric filled-area overlap.	2
Cross_p	Value 3: the secondary contour has portions inside and outside the main polygon.	3
ExtContact_p	Value 4: the secondary contour is outside with boundary contact and no portion inside. Edge or corner contact counts. A contained polygon touching the boundary is In_p instead.	4

AvaxPolylineType

Selects the curve algorithm used by Add_Shape and Add_ShapePolyLine. This is not a general open/closed flag. Polyline_pt joins the supplied vertices directly; the other values sample a curve from the supplied control vertices. Names and Integer values retain their R2 meanings, with Double XYZ arrays in v3.

Parameter	Description	Value
Polyline_pt	Straight segments join the supplied XYZ vertices in their original order; no curve interpolation is performed.	0
CSpline1_pt	Cubic interpolating spline with clamped endpoint derivatives (the first R2 cubic-spline algorithm).	1
CSpline2_pt	Natural cubic interpolating spline (the second R2 cubic-spline algorithm).	2
OBSpline_pt	Open clamped B-spline based on the supplied control polygon. The first and last endpoints are retained; interior controls generally are not points on the curve.	3
Bezier_pt	One Bezier curve controlled by the supplied vertices. It passes through the first and last vertices; interior vertices guide its shape rather than being interpolated.	4
CBSpline_pt	Closed periodic cubic B-spline based on the supplied control polygon. Its sampled XYZ path closes exactly; it generally does not interpolate the supplied vertices.	5

Use this selector in Add_Shape or Add_ShapePolyLine. The corresponding standalone curve methods are Add_PolyLine, Add_CSpline1, Add_CSpline2, Add_OBSpline, Add_Bezier and Add_CBSpline. Their existing member topics contain the examples.

Coordinate arrays retain the R2 logical first index 1; element 0 is ignored. Nothing for the Z array supplies zero elevations. Caller arrays are retained, and sampled shape geometry is stored as Double XYZ values. Invalid selectors are rejected without creating a partial shape. The selector chooses interpolation; the creation method and its fill/stroke flags determine how that geometry is drawn.

AvaxProgress

API reference

Identifies the operation category supplied by the Progress event. The underlying type is Integer. The event percentage is Double.

Values

Name	Value	Purpose
None_p	0	No active category. The control does not emit this sentinel as an operation notification.
Load_p	1	Loading a drawing or document. Initial automatic fitting belongs to the same Load operation.
Save_p	2	Saving a drawing. The final notification follows a committed save.
FindMinMax_p	3	Calculating the minimum and maximum drawing coordinates.
Autolimits_p	4	Calculating automatic drawing/work bounds and fitting the viewport through AutoLimits_c or FitToAvax.
Redraw_p	5	Progress while producing a drawing frame. Direct vector rendering provides coalesced intermediate updates; prepared scenes and retained geometry report their smaller rendering batches. Frozen replay and display suspension do not fabricate completed work.
Extends_p	6	Fitting the viewport to stored drawing extents through Extends_c. CalcExtends_c first reports FindMinMax_p for recalculation, then Extends_p for the fitted view commit.
Zoom_p	7	Committing a changed viewport through zoom methods/window changes or completed interactive pan, dynamic-zoom and scrollbar gestures. An unchanged view does not generate a fictitious operation.

See Progress for the declaration, callback rules and VB.NET example.

AvaxProp

Identifies a setting passed to GetAvaxProperty or SetAvaxProperty. The Type column is the in-memory value type in v3.

Parameter	Type	Description	Value
SaveOnExit_p	Boolean	Boolean, False by default. True silently saves the current modified drawing to its accepted filename before EndAvax, restarting the control or loading another drawing. Set a durable filename before relying on automatic saving. A failed save leaves the current drawing open and retains the previously saved file; edits made by a save callback remain dirty and stop replacement of the drawing. False leaves the existing saved file unchanged. The setting is retained in properties XML.	1
SelectContinuousOn_p	Boolean	True: Selected objects are added to the selection set / False: When an object is selected the previous object is deselected. / Remarks: If the Shift or Ctrl keys are pressed during the object selection process then irregardless of the parameters value the selected object is added to the selection set	2
PrintExtendsOn_p	Boolean	Fit-to-page mode for Print_c (Boolean; default False). True fits the chosen content on one sheet; False uses PrintUnit_MM_p and produces tiled pages when needed. Explicit print-method arguments control their own fit mode.	3
DimTextVerticalToLine_p	Boolean	Rotates legacy linear dimension text perpendicular to the dimension line when True. Boolean; defaultFalse. Takes precedence over DimTextHorizontal_p.	4
ColorPenIndex_p	Integer	Predetermined pen color	5
PrintPaperRGBColor_p	Integer	Paper color for legacy print output, expressed as an OLE COLORREF Integer (default white, 16777215). This changes the printed paper/background without recoloring stored drawing entities. Workspace page output retains white paper.	6
WidthPen_p	Integer	Predetermined pen thickness	7
LineStyle_p	Integer	Predetermined line styles	8
PointStyle_p	Integer	Predetermined point styles	9
DividePointType_p	Integer	Integer point-symbol table index used by Divide_c and Centroid_c; default 1. Negative assignments reset to 1; nonnegative values above 255 wrap modulo 256, matching the R2 table while using Integer in v3. For example, SetAvaxProperty(AvaxProp.DividePointType_p, 259) selects symbol 3. It does not change PenPointStyle or existing points. Divide interpolates positions/elevations along the source geometry and leaves the source unchanged. Settings XML retains the normalized value; nonfinite input is rejected without replacing it.	10
PointRadius_p	Double	The radius length for the appearance of the points with style > 0	11
Elevation3D_p	Double	Predetermined elevation insertion point for the z coordinate	12
Height3D_p	Double	Predetermined object height for the z coordinate	13
SnapOn_p	Boolean	Snap On or Off	14
SnapStepX_p	Double	The value for Snap X Spacing in units.	15
SnapStepY_p	Double	The value for Snap Y Spacing in units.	16
GridOn_p	Boolean	Grid On	17
GridStepX_p	Double	The value for Grid X Spacing in units.	18
GridStepY_p	Double	The value for Grid Y Spacing in units.	19

Parameter	Type	Description	Value
HatchDistance_p	Double	Spacing between Hatch lines in drawing units (Double; default 0.25). A finite zero or negative assignment resets to 0.25, including settings XML. Nonfinite assignments are rejected without replacing the prior value. Explicit Add_LinearHatch Distance arguments must still be positive.	20
HatchAngle_p	Double	Angle of Hatch lines in degrees (Double; default 0). Nonfinite assignments are rejected without replacing the prior value. Get_HatchDefaults also reads this angle.	21
HatchInside_p	Boolean	True (default) draws the even-odd interior of the boundary. False draws its complement inside the bounding rectangle. This is the inverse of the Add_LinearHatch fOutside argument.	22
HatchBorder_p	Boolean	True adds the bounding-rectangle border before Hatch lines; default False. This Boolean stores the rectangle-border bit. The Hatch dialog additionally offers polygon border mode.	23
HatchMask_p	String	Repeating row mask used by Hatch. Use only 0 and 1: 1 draws a candidate row and 0 skips it. Empty (default) draws every row. An all-zero mask produces no lines or borders. Invalid command masks fail without changing the drawing.	24
MirrorStay_p	Boolean	Keeps the original selected objects and creates mirrored copies when True. Boolean; default True, as in R2. Used by the interactive Mirror command and DoAvaxAction.	25
SelectStay_p	Boolean	Retains the selected objects after the execution of a command	26
ScaleStay_p	Boolean	Keeps the original selected objects and creates scaled copies when True. Boolean; default False. Used by the interactive Scale command and DoAvaxAction.	27
ScaleEqual_p	Boolean	Uses the same factor on both axes for interactive Scale when True (the default). When False, the absolute X and Y distances from the base point supply separate factors. Explicit ScaleItems and DoAvaxAction factors are unaffected.	28
RotateStay_p	Boolean	Keeps the original selected objects and creates rotated copies when True. Boolean; default False. Used by the interactive Rotate command and DoAvaxAction.	29
SelectCursorSize_p	Integer	Selection-box size in screen pixels (Integer). Default 5; values 6 through 64 are accepted and other values reset to 5. Setting it also sets the cross-circle radius to half the fitted size. Started controls fit cursor sizes to half the smaller drawing dimension, with a minimum of 2 pixels.	30
CrossCircleSize_p	Integer	Construction crosshair circle radius in screen pixels (Integer). Default 3; values 3 through 32 are accepted and other values reset to 3. Started controls fit the radius to half the smaller drawing dimension, with a minimum of 2 pixels. It may be assigned independently after SelectCursorSize_p.	31
ScreenXFactor_p	Double	Double,default1. Multiplies horizontal display scale about the nominal view minimum X. Vertical pixel units are unchanged; UnitX equals UnitY divided by this factor. Coordinate conversion,rendering,picking,culling,selection,grid,rulers,scrollbars and embedded-control layout share the projection. Changing the factor preserves stored geometry and nominal zoom bounds. Nonpositive values reset to1 as in R2; NaN and infinities are rejected while retaining the previous value. Settings XML retains the Double value without culture-dependent precision loss. FitToAvax accounts for both displayed axes and returns False atomically if a finite fitted view cannot be represented.	32
CircleSectors_p	Integer	The number of sectors in the circumference of circles, ellipses and arcs	33
xMin_p	Double	Drawing limit xMin	34

Parameter	Type	Description	Value
yMin_p	Double	Drawing limit yMin	35
zMin_p	Double	Drawing limits zMin	36
xMax_p	Double	Drawing limits xMax	37
yMax_p	Double	Drawing limits yMax	38
zMax_p	Double	Drawing limits zMax	39
DownLeftFactor_p	Double	Down Left Factor	40
UpRightFactor_p	Double	Upper Right Factor	41
WallWidth_p	Double	Predetermined wall thickness	42
CopyInOutWidth_p	Double	Copy Offset Distance inside or outside the selected object	43
FontNumber_p	Integer	Predetermined Font number	44
TextScale_p	Double	Selected Text Height	45
LineStyleScale_p	Double	Selected Line Style Scale factor	46
DimScale_p	Double	Dimension scale factor	47
DimDist_p	Double	Dimension line offset from dimensioned object	48
DimTextAbove_p	Boolean	Places linear dimension text above the dimension line when True. Boolean; defaultFalse. Centered text may be placed above automatically when it cannot fit between the marks.	49
DimTextHorizontal_p	Boolean	Keeps legacy linear dimension text horizontal when True. Boolean; defaultFalse. Otherwise it follows the readable direction of the dimension line.	50
DimArrowSize_p	Double	Dimension Arrow size	51
DimTickSize_p	Double	Dimension Tick size	52
DimOutExtLines_p	Double	Dimension of the outside extension line	53
DimInExtLines_p	Double	Dimension of the inside extension line	54
DimTextFont_p	Integer	Selects the font-table entry used for dimension text. Integer in v3; default 1. Use a defined font number; vertex dimension jobs require 1..256.	55
DimTextScale_p	Double	Dimension Text Scale factor	56
DimDecimal_p	Integer	Sets decimal places for legacy dimension labels, normally 0..15. Integer; default 2. Fixed decimal output uses a period regardless of host culture. Fraction denominators are controlled separately by DimFractionalTick_p.	57
DimDefText_p	String	Appends a string to legacy linear dimension labels. Default empty string. Include a leading space when separating a unit suffix from the number.	58
DimFactor_p	Double	Multiplies measured lengths before formatting legacy linear and vertex dimensions. Double; default 1. Signed and zero values are accepted without changing the measured geometry. Get_DimensionDefaults also copies this setting to editable native dimensions; see native-dimensions.html#measurement-formatting.	59
DimArrowType_p	Integer	Selects dimension endpoint marks: 0 oblique ticks, 1 open arrows, 2 circles. Integer; default 0. Values outside 0..2 reset to 0 after Integer rounding.	60
AngleDefText_p	String	Appends a string to legacy angular dimension labels after the unit symbol. Default empty string. Unicode text is supported.	61

Parameter	Type	Description	Value
AngleFactor_p	Double	Multiplies measured angles before formatting legacy angular dimensions. Double; default 1. Signed and zero values are accepted without changing the angular geometry. Get_DimensionDefaults also copies this setting to editable native dimensions; see native-dimensions.html#measurement-formatting.	62
PrintUnit_MM_p	Double	Default millimeters per drawing unit for print commands (Double; default 10). Finite zero or negative values, NaN and infinity restore 10. An explicit UnitMM argument to a print method takes precedence; nonpositive explicit arguments request fitting to the page.	63
PrintLandScapeOn_p	Boolean	Landscape orientation for Print_c (Boolean; default False). True swaps the nominal sheet width and height for output; False retains their nominal order. An explicit fLandScape argument to a print method controls that call independently.	64
PrintFactorX_p	Double	Horizontal print calibration multiplier (Double; default 1). Nonpositive values restore 1. Nonfinite values are rejected without changing the prior setting.	65
PrintFactorY_p	Double	Vertical print calibration multiplier (Double; default 1). Nonpositive values restore 1. Nonfinite values are rejected without changing the prior setting.	66
PrintAllOn_p	Boolean	Print-command area mode (Boolean; default False). False requests two window corners before printing. True starts Print_c immediately, using the selected items when a selection exists or the whole drawing otherwise. Legacy plot-file commands also use this flag to choose a window or the whole drawing.	67
PrintPaperX_MM_p	Double	Custom print-sheet width in millimeters for epsUser (Double; default 210). Nonpositive values restore 210, including when loading settings XML. Nonfinite values are rejected. Workspace pages use PageCustomWidthMM instead.	68
PrintPaperY_MM_p	Double	Custom print-sheet height in millimeters for epsUser (Double; default 297). Nonpositive values restore 297, including when loading settings XML. Nonfinite values are rejected. Workspace pages use PageCustomHeightMM instead.	69
PrintPaperMarginXY_MM_p	Double	Equal print margins in millimeters (Double; default 5). Zero permits edge-to-edge layout; negative values restore 5, including when loading settings XML. Nonfinite values are rejected. A margin that consumes the sheet is rejected by the output operation before publication.	70
SelectEnabled_p	Boolean	True: The object selection process is performed by the program False: The object selection process is not performed by the program upon a mouse click on the object but by the programmer using the available events (BeforeClick, Click) and the command SelectItemsTrue	71
DialogEnabled_p	Boolean	Boolean, True by default. True permits the built-in command dialogs; False uses the current/default drawing parameters and suppresses built-in file pickers and confirmation windows. File-based commands that need a filename can be driven through the file methods with an explicit path. Changing this setting preserves the active drawing command and its accepted points. The setting is stored in properties XML.	72
LayerPen_p	Integer	Selected Layer	73
Language_p	Integer	Integer language country code shared with Language: English=1 (default), Greek=30, Italian=39, French=33 and German=49. Generic assignment and settings XML reload the corresponding per-control MSGn.OPT message catalog while preserving the explicit XMLencoding_p. The direct Language=Greek assignment also selects ISO-8859-7. The options directory is selected with DirsSet. See AvaxLanguage for the file mapping, catalog format, fallback and configuration example.	74

Parameter	Type	Description	Value
FontName_p	String	Font name	75
FontBold_p	Boolean	Font Bold	76
FontItalic_p	Boolean	Font Italic	77
FontStrikethrough_p	Boolean	Font Strikethrough	78
FontUnderline_p	Boolean	Font Underline	79
FontCharset_p	Integer	Font Character Set	80
FontSize_p	Integer	Font Size	81
WinFontsEnabled_p	Boolean	Enables Window fonts	82
FillType_p	AvaxFill_t	Current fill descriptor (AvaxFill_t). Its iType field uses AvaxFillType; the other fields contain pattern, colors, flags and z-order. Changing this setting does not edit existing shapes. Pass the descriptor fields to a full Add_Shape overload or ChangeShapeFill when required.	83
ShapeZorderOn_p	Boolean	Enables shape z-order when True (default). Larger signed Integer zOrder values paint later; equal values use handle order. False uses handle order while retaining each stored zOrder.	84
SelectRegionOn_p	Boolean	Shows or hides the selection frame and its grips without changing which objects are selected. Boolean; default True.	85
SelectInsideShapeOn_p	Boolean	Enables selection inside shapes	86
RegionMarkWidth_p	Integer	Sets the selection-grip width in logical pixels. Integer; default 5. V3 keeps a minimum visible width of 5 for usable grips, including old zero-width settings. Larger explicit widths are honored. DPI scaling applies.	87
RegionRGBColor_p	Integer	Sets the selection frame and grip color as an Integer COLORREF (red + 256*green + 65536*blue). V3 defaults to yellow (&H00FFFF). A color matching the background is displayed with contrast.	88
RegionBoundingBoxOn_p	Boolean	Shows the frame connecting selection grips. Boolean; default True. When region dragging is disabled, a bounding box remains visible as in R2.	89
OriginTopLeft_p	Boolean	Boolean,default False. Selects the display Y origin. False places minimum Y at the bottom; True maps pixel zero to the nominal view minimum Y,with increasing world Y downward. Coordinate conversion,pointer input,vector rendering,picking,selection,culling,grid,rulers,scrollbars,embedded-control layout and view fitting share this setting. Text glyphs remain readable. Changing it preserves stored drawing geometry,the nominal zoom rectangle,the current drafting command and accepted points. Read or change it with GetAvaxProperty/SetAvaxProperty; settings XML retains its value.	90
AnimationOn_p	Boolean	Boolean; the default is True. Enables frame selection and repainting for animated pictures and polygon fills. False stops the animation repaint timer and displays the first frame. Re-enabling uses the current AnimationInterVal_p. Printing displays the first frame. In an AAF sequence containing movies this setting selects the sequence slot; each movie retains its own playback timing. The setting belongs to the control instance and is included in settings XML.	91
AnimationInterVal_p	Integer	Integer time in milliseconds between frames of an animated picture or fill; the default is 10. Updating an active sequence changes its repaint timer immediately. While animation is disabled the value is stored for the next enable. Zero and negative settings are retained but playback uses one millisecond to prevent division by zero and invalid timer intervals. Invalid or unrepresentable values leave the previous setting unchanged. The interval is included in settings XML and does not change an AVI movie's own frame timing.	92

Parameter	Type	Description	Value
TilePictureFile_p	String	String filename of an image repeated at its native device dimensions across the viewport; the default is empty. While nonempty, this takes priority over BackgroundPictureFile_p, including when the tile file is unavailable. Empty restores any configured aligned background. The supported formats, relative-path resolution, file-change detection, explicit Regen and safe unavailable-file behavior follow BackgroundPictureFile_p. Metafile repetition retains its original device extent. This is a display setting and does not create drawing entities.	93
PatternsOn_p	Boolean	Enables shape fills when True (default). False hides the fill without changing the stored fill, outline geometry or drawing undo history.	94
CheckBackColorOn_p	Boolean	When AVAX draws an object changes it's color when it is like the background color	95
FixedSizeText_p	Integer	Integer screen text height in pixels. Positive values override zoom-dependent text height; zero or negative values disable fixed sizing. Applies to ordinary text, documents and text shapes without resizing their stored geometry. Printing and exported images retain drawing dimensions. Fixed sizing takes precedence over MinTextSize_p. The selection area follows the displayed text and refreshes after zoom, including framed RTF and document shapes and either viewport origin. Ordinary documents retain the full affine frame while applying the fixed display height once. The font remains anchored at its document insertion point under reflection, shear and either viewport origin; ScreenXFactor_p continues to affect horizontal projection. Selection dither keeps its screen spacing. Printing uses the stored document size.	96
GridLinesOn_p	Boolean	Shows grid lines when grid is on	97
CrossHairStyle_p	Integer	Crosshair stroke style (Integer): 0 solid, 1 dash, 2 dot, 3 dash-dot, 4 dash-dot-dot, 5 invisible, 6 inside-frame solid. Default 0.	98
SelectCursorHideOn_p	Boolean	Boolean; default False. True hides the CAD selection pickbox, including modification picks. It does not hide an assigned custom Windows pointer or the construction crosshair. Use CrossHairOff_p for construction graphics or mAvax.HideCursor_m for temporary cursor/preview clearing. The current command and its accepted points are retained.	99
SelectCursorMousePointer_p	Integer	Integer legacy pointer code: -1 hides the Windows pointer while retaining separately controlled CAD cursor graphics; 0..15 select standard pointers (4 uses the default-pointer fallback; 5/15 are SizeAll, 12 is NoDrop and 13 is arrow/hourglass); 99 selects the .cur file named by SelectCursorFile_p. Default 0 uses the native default pointer. GetAvaxProperty returns the Integer code. Standard pointer 0/1 values may share the Windows default-arrow handle. Invalid codes are rejected without replacing the accepted setting.	100
SelectCursorFile_p	String	String path to the custom .cur file used when SelectCursorMousePointer_p is 99; default empty. Set the code and filename in either order. The file is loaded when custom mode becomes active or the active custom filename changes. A failed load leaves the accepted cursor and active filename unchanged. Empty custom filename falls back to the crosshair. XML stores the filename and pointer code, not a native cursor handle.	101
UndoRedoOn_p	Boolean	Enables undo and redo	102
RegionDragDisable_p	Boolean	Prevents dragging selection grips. Boolean; default False. When True, the grips are hidden and the bounding box remains visible.	103
Parameter	**Type**	**Description**	**Value**
FixedSizePoint_p	Integer	Integer, default 0. A positive value fixes the device-size parameter of nonzero point styles on both axes, independently of zoom and ScreenXFactor. Fixed size takes precedence over MinPointSize. Zero or a negative value disables the fixed override. The original signed Integer setting is retained; the painted device-size parameter is	104

Parameter	Type	Description	Value
		bounded to 0.5 through 1,000,000 pixels for stable rendering. Each axis half-extent is the size parameter divided by sqrt(2), before the pen outline. Style 16 is a filled ellipse. Style-zero dots use pen width instead of this size. Point objects, point shapes and nested point symbols use the same geometry for painting,picking,selection bounds,culling and fitting. Raster/SVG export includes a symbol when its ink crosses the output boundary even if its insertion point lies outside it. Raster and memory-HDC sizes use output device pixels; SVG uses CSS pixels at 96 pixels per inch and preserves the physical viewport dimensions. No stored coordinate,entity attribute or active construction point is changed. Set the value with SetAvaxProperty(AvaxProp.FixedSizePoint_p, CObj(12)); read it with CInt(GetAvaxProperty(AvaxProp.FixedSizePoint_p)). Settings XML preserves the preference.	
MinTextSize_p	Integer	Integer minimum screen text height in pixels. Positive values keep text readable when zooming out; zero or negative values disable the minimum. Text shapes also enforce this minimum, including at zoom levels where their stored frame is smaller than one pixel. Printing and exported images retain drawing dimensions. FixedSizeText_p takes precedence when positive. The selection area follows the displayed text and refreshes after zoom, including framed RTF and document shapes and either viewport origin. Ordinary documents apply the minimum display height once to their complete affine frame, including reflection and shear. Text already larger than the minimum keeps its natural size. Both viewport origins and horizontal projection retain the same document anchor; printing and stored geometry are unchanged.	105
MinPointSize_p	Integer	Integer, default 0. Sets the minimum device-size parameter of each axis of a nonzero point symbol, unless FixedSizePoint is positive. Natural size still follows PointRadius and the current projection when it is larger than the minimum. Zero or negative values disable this minimum. The signed Integer preference is retained, with the same effective device-size bounds and sqrt(2) half-extent convention described for FixedSizePoint_p. A style-zero dot continues to use its pen width. Painting,picking,selection,edge culling,raster/SVG export and memory-HDC output consume the same size policy; changing this display preference does not alter stored world coordinates or interrupt the current command. Set the value with SetAvaxProperty(AvaxProp.MinPointSize_p, CObj(12)); read it with CInt(GetAvaxProperty(AvaxProp.MinPointSize_p)). The preference is per control and is preserved by settings XML.	106
CrossHairRGBColor_p	Integer	Crosshair RGB color (Integer) in Windows COLORREF byte order. Default 0; a contrasting display color is used when it matches the drawing background.	107
GridRGBColor_p	Integer	Integer COLORREF (0x00BBGGRR), default &HC0C0C0. Sets the displayed grid color in dot and line modes; it does not change object colors. Zero means black. The grid is opaque and always uses background contrast: a color equal to the drawing background is inverted, independently of CheckBackColorOn_p. GetAvaxProperty retains the requested value; GetPalColor(257) resolves its screen contrast. For example, SetAvaxProperty(AvaxProp.GridRGBColor_p, &H332211) requests RGB(17,34,51). GridOn_p controls visibility. Settings XML retains this value.	108
ACADVersion_p	Integer	Integer target code for the DXF export command. Default 3 selects the R2 ACAD_10 format. Codes 0 through 7 are the named AvaxACADVersion members; numeric v3 codes 8 through 12 select 2004, 2007, 2010, 2013 and 2018. The setting survives SaveAvaxPropertiesToXML/LoadAvaxPropertiesFromXML. The full PrintToDXF Version argument overrides this command setting; the short overload keeps its ACAD_10 default. Invalid targets fail at export without replacing an existing destination.	109

Parameter	Type	Description	Value
TransparentPicOn_p	Boolean	Boolean; default False. True enables the R2 raster transparency rule for embedded pictures and bitmap/metafile-filled shapes: each opaque RGB byte is combined with the destination using bitwise OR when the active drawing/paper background is black, or bitwise AND otherwise. The rule is chosen from that configured background, not each underlying object color. Black pixels therefore preserve underlying drawing on a black background; white pixels do so on other backgrounds. PNG alpha remains coverage, including fully transparent pixels. False restores normal source-over image drawing. For example, SetAvaxProperty(AvaxProp.TransparentPicOn_p, True) changes the appearance of existing pictures without changing their stored bytes or coordinates. The flag is saved in settings XML. CPU, GPU and recorded drawing use the same color operation.	110
PixelWidth_MM_p	Double	Sets the display and printing pixel width in mm	111
RoundEndCapOn_p	Boolean	Boolean, default True. Selects round rather than flat (butt) ends for open stroked lines and paths. Round ends extend by half the pen width beyond each endpoint. Closed paths retain their joins; changing the cap preference does not round the corners of a filled polygon or alter geometric ribbon widths. R2 keeps a thick style-zero dot circular even when ordinary lines have butt caps; a one-pixel dot remains a visible pixel. Painting,picking and crossing selection use the same dot silhouette and leave its empty bounding-square corners unselected. Raster and memory-HDC output follow the same cap preference. Dashed hDC strokes retain their centreline dash lengths and phase before caps are added, with vector output. SVG writes explicit line-cap and round-join attributes, preserves connected/disconnected/closed polyline contours and exports the actual point-symbol ink. Set the value with SetAvaxProperty(AvaxProp.RoundEndCapOn_p, CObj(False)); read it with CBool(GetAvaxProperty(AvaxProp.RoundEndCapOn_p)). The preference is per control,round-trips through settings XML,and preserves the current drawing command and geometry.	112

Additional declared values

Name	Value	Description
ACADPaperSpaceOn_p	128	Boolean; default False imports model-space drawing entities. True additionally imports drawing entities from paper-space layouts. Applies to Add_DXF, Add_DWG and their insertion commands. The sheet-boundary viewport is layout metadata and is skipped; model-view viewports are not expanded and unsupported viewport objects still report an import error. This setting is independent of attribute-text filtering.
AntiAliasingOn_p	164	Boolean switch for antialiased vector drawing. True smooths supported line, curve and text edges; False uses the ordinary pixel rendering path. It changes presentation rather than stored geometry.
ArcSegmentsLength_p	166	Non-negative Double. Positive values replace Add_Arc full-circle resolution with floor(2*pi*radius/value), clamped by MinCircleSectors and MaxCircleSectors. Zero uses CircleSectors. The old inaccurate circumference constant is corrected. This controls stored Add_Arc chords, not analytic Add_CurveArc geometry.
AutoOSnap_p	136	Bit mask of enabled automatic object snaps: bit (OSnapType-1) corresponds to each type 1..9. Zero disables automatic snapping; an explicit one-shot snap command overrides the mask until a valid point is accepted or the command is cancelled/replaced. A custom User_o result requires a UserOsnap handler. Discrete candidates retain priority over Nearest inside the pixel aperture. These focused tests cover projected line/curve nearest, perpendicular, intersection filtering and tangent-first priority; all entity/mode combinations, persistence and legacy client interactions are not yet fully certified.
AutoOSnapApproachInPixels_p	138	Integer approach radius used to acquire object-snap candidates,measured in screen pixels. The R2 default is20. Integer conversion uses midpoint-to-even rounding;the full widened Integer value is retained by GetAvaxProperty and settings XML. Zero and negative values disable candidate acquisition;they do not select a fallback radius. Positive values use the same aperture for automatic and explicit object snaps. Horizontal aspect scaling and either Y origin do not change the pixel radius. Candidate ranking is measured in pixel space,including cases where world-coordinate subtraction or conversion would overflow or underflow at an intermediate step. UserOsnap receives the corresponding world-coordinate search bounds;unrepresentable exterior bounds are clipped to the finite Double range. The appearance radius of a snap marker is separately controlled by AutoOSnapCircleRadiusInPixels_p.
AutoOSnapCircleRadiusInPixels_p	137	Integer radius of the displayed object-snap marker in device pixels. The default is 6. Rendering bounds the radius to 2 through 128 while GetAvaxProperty and settings XML retain the full requested Integer value. Marker size is independent of zoom, ScreenXFactor and OriginTopLeft. Point acquisition continues to use AutoOSnapApproachInPixels_p. Invalid or unrepresentable values leave the last accepted radius unchanged.
AvaxControlShrinkInvisibleOn_p	162	Boolean; the default is True. Each active hidden embedded control reclaims its original world height by shifting controls whose minimum world Y is strictly lower, independently of horizontal overlap. Erased hidden controls do not contribute; restoring a hidden control restores its contribution. Toggling the property immediately repositions native editors. Disabling it clears the layout displacement. Original drawing geometry and the active command remain unchanged. Native editors and cAvControls geometry queries use the same active hidden rows. The value belongs to the control instance and is saved in settings XML.
BackBitmapDrawingOn_p	155	Boolean switch enabling drawing through the bitmap background/cache helpers. False disables those helper bitmap drawing operations. It does not change the stored drawing objects.
BackBitmapSize_p	161	Integer requested bitmap-cache edge in pixels; the default is 6000. Legacy bitmap rendering uses the platform default (3000 for tablets, 6000 otherwise) for zero; negative requests decline allocation without replacing an existing cache. Adaptive overview rendering bounds allocations to 256 through 8192 pixels and uses 4096 for a nonpositive request. The full requested Integer is retained in settings XML. Changing it does not rescale drawing geometry or the view. Invalid or unrepresentable values preserve the last accepted setting.
BackgroundPictureAlignment_p	149	Integer alignment of an unscaled background image in the drawing viewport: 0, 1 and 2 are top-left, top-centre and top-right; 3, 4 and 5 are middle-left, centre and middle-right; 6, 7 and 8 are bottom-left, bottom-centre and bottom-right. Out-of-range values use top-left. A tiled picture takes priority over this alignment.

Name	Value	Description
BackgroundPictureFile_p	148	String filename of one background image in the drawing viewport; the default is empty. Empty removes the image. BackgroundPictureAlignment_p selects one of nine native-device-size positions. A nonempty TilePictureFile_p takes priority. BMP,PNG,JPEG,GIF,TIFF,ICO,WMF and EMF are supported; background GIFs use frame zero. The exact supplied filename is retained, including relative paths, Unicode and spaces. Relative paths resolve against the process current directory. Ordinary file changes/creation/deletion are noticed on paint; Regen forces reload even when timestamp and length are unchanged. Missing,locked or undecodable files leave a clear canvas and usable control. File changes do not create drawing edits.
CancelArray_p	134	Boolean array-stop request, initially False. True before execution stops an array without allocating copies. Set it from BeforeCopy to stop before the next array position: the current cell still completes, and earlier cells remain in the drawing. BeforeCopy sees previous copies already committed, in column, row and source order. BeforeCopy.fCancel skips only its current cell. A consumed stop request resets to False and all completed cells form one Undo group; an existing host UndoRecorderOn scope is preserved. NewUserItems and Change announce the completed array once. Setting CancelArray_p from these final notifications requests cancellation of the next array. The setting is retained in property XML.
ComplexLineTypesSupport_p	150	Boolean switch for text and shape symbols in complex line-type definitions. True enables their presentation; False uses the ordinary line pattern without those complex symbols. Stored definitions and geometry are retained.
CrossHairOff_p	120	Hides the construction crosshair and transient selection/zoom rectangles when True; False restores them. Default False. A host-selected pointer and MouseDrawing_p graphics remain available, and the active drawing command is preserved.
CurveArcOn_p	163	Boolean, default False. CurveArcOn=False (the default) constructs a segmented Polyline_i arc, using CircleSectors or ArcSegmentsLength with MinCircleSectors/MaxCircleSectors. CompleteInput/Enter, Close or switching/cancelling finishes the accepted path. CurveArcOn=True inserts an analytic Curve_i immediately and leaves the arc command ready to repeat. A preceding pending polyline is inserted together with that curve in one undoable batch. Switching Polyline to Arc keeps the final accepted vertex as the start; switching back restores the accepted prefix and discards only unfinished arc references. Changing arc variant keeps the start/prefix and resets the unfinished references. Drafting-aid toggles retain input. Preview uses the same representation. Geometry/capacity failures preserve accepted input and Redo; success consumes points before notifying NewUserItems/Change. A host callback may replace the document, start input or throw without stale cleanup. The explicit Add_Arc and Add_CurveArc methods remain independent of this property. CurveArcType was an internal R2 field, not a missing public control property.
DimAngularUnitFormat_p	141	Selects legacy angular dimension units: 0 decimal degrees, 1 degrees/minutes/seconds, 2 decimal gradians, 3 gradians/minutes/seconds, 4 radians. Integer; default 0. Get_DimensionDefaults also copies this setting to editable native dimensions; see native-dimensions.html#measurement-formatting.
DimFractionalTick_p	140	Sets the fraction denominator for architectural and fractional dimensions. Integer; default 32. Negative inputs reset to 32. Zero is retained in the property and uses an effective denominator 2 when formatting. Fractions are reduced and carry into the whole value.
DimUnitFormat_p	139	Selects linear dimension units: 0 decimal, 1 engineering (feet and decimal inches), 2 architectural (feet and fractional inches), 3 scientific, 4 fractional inches. Integer; default 0.
DXFAttdefAsTextOn_p	147	Boolean; default False. True includes visible ATTDEF definition text during DXF/DWG import through the control methods and commands. An assigned block attribute replaces its definition and is not duplicated. DXFAttribAsTextOn_p separately controls the assigned value; hidden attributes stay hidden. fNoText and fNoBlock retain their separate filtering effects.
DXFAttribAsTextOn_p	146	Boolean; default True. Includes visible assigned ATTRIB values in DXF/DWG imports, including inserted blocks. False excludes those values while retaining block geometry and independent text. Hidden attributes are always omitted. DXFAttdefAsTextOn_p separately controls definition text; fNoText suppresses all text and fNoBlock skips the whole insert.

Name	Value	Description
DXFCurveArcOn_p	143	Boolean preference for newly imported DXF/DWG arcs, False by default. Full CIRCLE entities remain native circles for either value. False converts partial circular arcs into connected sampled polylines using the circle/arc segmentation settings; True retains supported planar zero-width arcs as analytic Curve_i objects. A complete analytic arc uses the same full-circle exception as Add_CurveArc. Changing the preference does not convert existing objects or cancel the current drawing command. Tilted curves and polylines with constant or tapered geometric width use sampled XYZ geometry where an analytic planar representation cannot retain their plane or width. Their width profiles remain intact for either value. Values are local to each control and are included in property XML.
FitOnResize_p	165	Boolean resize policy. True fits a nonempty drawing to the resized control. False retains the current view rectangle and refreshes its presentation.
FixedDocumentTextSizeOn_p	152	Boolean, default True. When scaling an ordinary document or document shape, preserve the font height while changing the page width and reflowing the text. The document shape frame is recomputed for the resulting lines, so its old glyphs are not stretched to fill a scaled rectangle. False scales font height along the transformed vertical direction. Nonuniform scaling, shear and reflection retain their directed frame axes; frame corner Z coordinates are preserved. The chosen result is retained by Undo/Redo and drawing-file round trips.
JPGCompressQuality_p	144	Integer JPEG quality used by PrintToJPG and other JPEG exporters. R2 default is 0. Values convert to Integer with midpoint-to-even rounding; a converted value outside 0..100 resets to 0, matching cAvaxProp.JPGCompressQuality. Nonfinite values are rejected by the common property validator. PrintToJPG captures the quality before PrintingPage callbacks. Actual JPEG quantization tables at 0,50,100 are tested; source conversion and 100.4/100.6 range boundaries are covered.
MagnifyLensSize_p	132	Integer lens-size selector: 0 = small (67 x 67 screen pixels), 1 = medium (113 x 113; default), 2 = large (160 x 160). Other Integer values reset to 1 as in R2. The lens fits inside the viewport and is limited by a smaller viewport. For example, SetAvaxProperty(AvaxProp.MagnifyLensSize_p, 2) selects the large lens. The active Magnify_c lens updates immediately; the drawing and current zoom remain unchanged. Settings XML retains the normalized selector. MagnifyScale_p controls the independent magnification factor.
MagnifyScale_p	131	Double magnification for the Magnify_c lens; default 1.5, matching R2. Positive finite values are retained at Double precision, including factors below one. Zero or a negative value resets to 1.5; nonfinite input is rejected without replacing the accepted setting. For example, SetAvaxProperty(AvaxProp.MagnifyScale_p, 3.0) enlarges lens detail three times. Changes apply to an open lens without changing the drawing, view or zoom history. Unrepresentable extreme spans skip rendering safely. Settings XML uses the same normalization and precision.
MaxCircleSectors_p	168	Integer upper bound for automatic full-circle arc resolution when ArcSegmentsLength is positive; default 120. Add_Arc validates 3 <= minimum <= maximum <= 1000000 before editing. It does not force an analytic Add_CurveArc into line segments.
MinCircleSectors_p	167	Integer lower bound for automatic full-circle arc resolution when ArcSegmentsLength is positive; default 6. Add_Arc validates 3 <= minimum <= maximum <= 1000000 before editing. The stored arc uses the corresponding fraction of the circle.
MouseDrawing_p	118	Extra cursor geometry in a two-dimensional Double array: rows 1..n, columns 1..7 are x1, y1, x2, y2, palette color index, width and line-style index. Row/column zero are reserved. Coordinates are drawing units and stay fixed in the drawing; they are not offsets from the pointer. A normal zero-based Double array is shared as in R2: pass CObj(lines) to SetAvaxProperty and edit lines from OnMouseMove to update the next frame, including idle selection mode. GetAvaxProperty returns the same array. Original Single and other numeric arrays are accepted by conversion into an independent Double array; use Double for shared live edits in v3. Nothing or numeric/string zero clears the overlay. Initial nonfinite fields, malformed bounds and metadata outside Integer range are rejected without replacing the previous array. A row invalidated by a later host edit is skipped while other rows draw; repair it to restore drawing. SaveAvaxPropertiesToXML returns Nothing for invalid live data. Valid XML loads into an independent array. PauseAvax and temporary cursor clearing suppress the overlay; TouchScreenOn_p does not suppress host-supplied lines.

Name	Value	Description
MouseIsBlocked_p	142	Boolean, default False. True blocks the control mouse actions, navigation drags and wheel zoom while continuing to publish OnMouseDown, OnMouseMove and OnMouseUp to the host. BeforeClick is not accepted while blocked. The active construction command and its accepted points remain; explicit SetCoordinate calls still work. Enabling the setting ends a current pan/selection/region/freehand gesture and releases capture. For example, SetAvaxProperty(AvaxProp.MouseIsBlocked_p, True) lets the host handle mouse input itself. Assign False to resume normal mouse input. Settings XML retains the Boolean.
OLEDropMode_p	129	Integer, 0 by default. Zero disables OLE drag-and-drop; 1 or 2 enables the control and its drawing surface to receive drag data and file drops. Integer values outside 0..2 normalize to zero. OnDragOver and OnDragDrop allow the host to change the data/effect or cancel the transfer. Switching the mode to zero from a drop callback stops that transfer. Accepted drops preserve the source geometry and elevations and form one Undo operation. The normalized setting is stored in properties XML.
OnLoadAutoLimitsOn_p	114	Boolean load policy, True by default. Enables automatic drawing-limit fitting when a loaded document is assigned through the normal load path. False retains the applicable saved/current view instead of requesting that fit.
OnlyLinesACADExportOn_p	113	Boolean; default False retains supported editable CAD objects. True exports supported curves, paths and text as linework, including the presentation of native dimensions and PAT hatch rows. Native dimension/HATCH semantics are replaced by their graphics; explicit SOLID fills and independent visible borders are retained. PAT lines preserve spacing, row shift, dashes, holes and elevation. Curved hatch boundaries are sampled for clipping. Unsupported transparent hatch linework, tilted PAT planes, extended vector fills or unsafe output density report an error and preserve an existing destination. Disable the setting to retain native HATCH transparency.
OpenGL_p	169	Boolean; the default is False, selecting CPU rendering as in R2. True requests the GPU canvas. A failure of the current GPU surface falls back to CPU rendering and the getter then returns False. Late failures from replaced or disposed surfaces are ignored. Backend changes preserve drawing geometry, coordinates and a Line command in progress. The preference is included in settings XML.
PltTextQuality_p	122	Integer sampling quality for vectorized text,including complex line-style text and SVG,PLT,DXF/DWG vector-text output. Default1 is normal detail;0 is coarse and2 is finer. The legacy value -1 is accepted and retains the format-specific old-vector profile. Legacy vector fonts retain their stored strokes. Values convert to Integer with midpoint-to-even rounding. A converted value outside-1..2 resets to1,so getter,settings XML and exporters agree. Changing quality invalidates cached text/line rendering. See LoadACADLineStyles for complex line-style text. In PrintToPLT and PLT_Out_c, -1 explicitly selects the R2 DOS stroke-font profile for ordinary Text_i even if native HPGL labels were requested. The selected S0..S8 font supplies its fixed glyph design; an index above 8 uses S1. Underlines/strikeouts and stored affine geometry are preserved. Windows text shapes and documents keep their own renderer. The coordinate geometry remains Double until the HPGL Integer wire boundary. SVG Windows vector text at qualities 0/1/2 uses filled contours, with finer canonical curve sampling at higher quality. In SVG, -1 with vector export enabled uses the R2 ordinary-text stroke-font profile, including underlines/strikeout, nominal-height row advance, stored affine geometry and Double coordinates. With legacy resources absent, Windows outlines use normal sampling. This profile does not replace the Windows fonts of rich documents. Ordinary editable text produced by decomposing an RTF file follows the ordinary-text profile.
PltUnit_p	125	Double scale of one HPGL plotter coordinate; default 0.025, matching R2. Add_PLT multiplies file coordinates by this value before insertion. PrintToPLT divides the output coordinates (after PrintUnit_MM_p and the print factors) by it. For example, SetAvaxProperty(AvaxProp.PltUnit_p, 0.5) makes an imported HPGL distance of 8 become 4 drawing units. A finite zero or negative value resets to 0.025; nonfinite values are rejected without replacing the last accepted value. Settings XML retains exact Double precision. Configure this setting before import/export; it does not rescale existing geometry.

Name	Value	Description
PltVectorTextOn_p	135	Boolean PLT text-output selector; default True. True exports text as vector strokes/outlines using PltTextQuality_p. False writes native HPGL LB labels where the text fits the legacy Windows-1253 encoding and the full label is inside the crop. Unrepresentable Unicode or a clipped label falls back to vector outlines rather than losing characters. For example, SetAvaxProperty(AvaxProp.PltVectorTextOn_p, False) requests

Name	Value	Description
		native labels for PrintToPLT. The setting does not edit stored text or affect unrelated export formats. It is retained in settings XML.
PrintAllBlackOn_p	126	Print stored colored drawing ink in black when True; default False. The configured print-paper color is retained and stored entity colors are unchanged.
ScaleSensitivity_p	151	Sets the Double divisor for uniform interactive scaling: scale factor = distance from base point / sensitivity. Default 1. Zero resets to 1. Finite negative values are supported and scale through the base point. Nonfinite input fails without changing the setting. Ignored when ScaleEqual_p is False.
SelectTextDitherOn_p	133	Enables the hatch overlay on selected text and formatted RTF documents. Boolean; default True. Unselected text is unaffected.
ShowRealPixelWidth_p	130	With positive PixelWidth_MM, scales the physical pen by the drawing-to-zoom ratio. The denominator uses viewport pixel size, independently of horizontal coordinate stretch. False leaves the ordinary screen pixel width. This batch verifies software rendering only.
SilenceOn_p	160	Retained Boolean silent-mode preference, False by default. The drawing core stores it per control and includes it in settings XML. Use it when coordinating silent behavior in the host application. Application-owned sounds and dialog policy are configured by the host; DialogEnabled_p remains independent. Invalid values preserve the last accepted preference, and changing it preserves an in-progress drawing command.
SVGImportConvertToShapeOn_p	172	Boolean SVG outline-import selector; default False. False retains SVG lines as native Line items and circles/ellipses as native analytic Ellipse items with their affine frame; general paths/polylines use line segments. True creates sampled Shape items and retains the outline kind: Line_s, Circle_s, Ellipse_s or Polyline_s. A nonuniformly transformed circle becomes Ellipse_s. Filled regions remain filled shapes in either mode; text import follows its own text rules. For example, SetAvaxProperty(AvaxProp.SVGImportConvertToShapeOn_p, True) requests editable outline shapes for subsequent Add_SVG calls. It preserves placement, colors, properties and one-based return handles; changing the flag does not convert previously imported objects. The setting survives XML.
TextAlignWidth_p	123	Double width of the alignment field for Add_Text with Center_a or Right_a as iStyle. Default 21; a negative value restores 21 and zero disables the field. Only a single line narrower than the field is moved, along its insertion angle. Multiline text is not moved. PltTextQuality_p = -1 disables field alignment.
TextExcessBackgroundColor_p	171	Integer OLE color used by TextExcessSpaceFactor_p. Default 0 is black. Standard RGB/OLE colors and Windows system-color values are supported. This setting changes the temporary screen background, not the text color or saved drawing content.
TextExcessSpaceFactor_p	170	Double multiplier for the temporary text background width. Default 0 disables it. Positive values extend right from the text origin; negative values extend left. The background uses TextExcessBackgroundColor_p. It is displayed on screen only and is omitted from printing and exported images, including text shapes and RTF. Nonfinite values are rejected.
TextInitHeight_p	117	Initial text height also affects complex line text and invalidates cached rendering when changed.
TextInitWidth_p	116	Double initial text-width measure in drawing units, default 0.2. Used with TextInitHeight_p by vector-text advance and dimension-label construction.
TextWidthToHeightFactor_p	115	Double character-width factor for Windows text presentation. Default 0 uses the natural or fitted glyph width. Positive values set the character width relative to its height; negative values restore 0. In text shapes, a fixed or enforced minimum pixel height takes precedence over this width factor. Nonfinite values are rejected. Printing retains native text-shape glyph ink beyond the original logical frame when the requested output crop contains that ink; an enlarged width factor does not clip those glyphs to the old frame.

Name	Value	Description
TouchScreenOn_p	159	Boolean controlling touch presentation, independently of input blocking. The initial value follows R2: True for the Windows tablet metric or a primary screen no taller than 16 cm using the R2 logical-DPI calculation, otherwise False. This default is sampled once per control; SetAvaxProperty can override it. True hides the hovering construction crosshair and CAD pickbox, and disables the effective transparent-selection fill while keeping its outline and stored preference. It retains host MouseDrawing lines and normal

Name	Value	Description
		construction/selection input. For example, SetAvaxProperty(AvaxProp.TouchScreenOn_p, True) enables this presentation explicitly. XML stores the explicit touch value and underlying fill preference, so disabling touch after loading restores the configured fill.
TransparentSelectBoxOn_p	156	Enables the translucent fill inside a selection window. Boolean; default True. TouchScreenOn_p temporarily makes this property read False and suppresses the fill; leaving touch mode restores the stored preference.
TransparentSelectBoxOpacityPercentage_p	158	Sets selection-window fill opacity as a Double percentage from 0 to 100. V3 defaults to 11. Zero hides the fill; 100 is opaque. Values outside the range reset to the default. Nonfinite input fails without changing the current value.
TransparentSelectBoxRGB_p	157	Sets the selection-window fill color as an Integer COLORREF. V3 defaults to &HFFA000. Both window and crossing selection use this fill color; their outlines still distinguish their selection modes.
VectorTextExportOn_p	124	Boolean; default False retains supported editable TEXT/MTEXT. True converts supported text to glyph linework in DXF/DWG. Native dimensions are exported as static graphics so their labels are outlined too. Ordinary circles and editable paths, and native hatches without text, retain their normal representation when OnlyLinesACADExportOn_p is False. Either export switch outlines text. Unsupported document/image/fill formats retain the restrictions described by the export methods. In SVG, True uses filled sampled Windows glyph paths at quality 0/1/2, preserving the text frame and decorations without requiring the reader to install the font. False keeps ordinary Windows text as native SVG text. The text remains editable in the AVAX drawing. Ordinary DOS-font text and rich documents follow their documented font rules; quality -1 selects its legacy stroke-font profile, retaining decorations and the stored frame. Missing legacy resources fall back to the drawing Windows font; rich documents keep their own run fonts.
VectorTextPresentationOn_p	153	Boolean text-presentation switch. True draws font glyph outlines as vector paths. False uses ordinary font text drawing where Windows-font presentation is enabled. Stored text and font attributes are retained.
XMLencoding_p	119	String XML encoding name, UTF-8 by default. Supported aliases normalize to Encoding.WebName. An empty value leaves the in-memory XML declaration encoding unspecified and uses UTF-8 when saved. ISO-8859-7, Windows-1253, UTF-8 and UTF-16 retain Unicode text through XML character references when needed. Drawing XML and settings XML use the selected encoding. SetAvaxProperty returns False for invalid names, preserving the accepted setting. Direct Language=Greek selects ISO-8859-7; generic Language_p assignments and settings XML retain the explicit encoding.
ZoomFactor_p	121	R2 zoom divisor, default 10, not a percentage or a direct scale. ZoomIn reduces each span by 2/factor; ZoomOut increases it by 2/factor. Higher divisors give smaller steps. V3 buttons and wheel use this same convention. Finite values <=2 are reset to 10 to avoid the original zero/inverted windows; nonfinite values are rejected. This supersedes earlier inferred percentage behavior.
ZoomMemoryOn_p	145	Boolean switch allowing zoom/scene bitmap caches, True by default. It works with BackBitmapDrawingOn_p and the display-size options. This image-memory setting is independent of the Previous-view command history.

See Grid and coordinate snap commands for the executable R2 command-name mapping, cancellation and direct actions.

See Arc commands and segmentation for CurveArcOn_p, CircleSectors_p, ArcSegmentsLength_p and the segment limits.

See Drawing modes and settings for grid/snap switching and configuration, drawing/display modes, Double-valued distances, and independent zoom image memory and view history.

See Navigation commands and view history for the 50-view Previous history, page and small-step movement, work-area limits, and resize behavior.

See Selection commands and type filters for Add/Sub and window selection, Shift/Control behavior, automatic selection and type filtering.

Background and tile images

BackgroundPictureFile_p places one image at its native device size using BackgroundPictureAlignment_p. TilePictureFile_p repeats an image across the viewport and takes priority while its filename is nonempty. These settings do not create drawing entities. Call the example with a control initialized by StartAvax.

Supported images include BMP, PNG, JPEG, GIF, TIFF, ICO, WMF and EMF. Background GIFs use their first frame. Relative filenames resolve against the process current directory. Normal file changes, creation and deletion are noticed on the next paint; call Regen when replacing content while preserving the file timestamp and byte length. Missing, locked or undecodable files leave the canvas clear. Metafile alignment and repetition retain the original device dimensions.

VB.NET example

```
Option Strict On
Public Module BackgroundImageExample
    Public Sub ShowBackground(cad As Avax.cAvax, fileName As String)
        cad.SetAvaxProperty(Avax.AvaxProp.TilePictureFile_p, "")
        cad.SetAvaxProperty(Avax.AvaxProp.BackgroundPictureFile_p, CObj(fileName))
        cad.SetAvaxProperty(Avax.AvaxProp.BackgroundPictureAlignment_p,
                            Avax.BackgroundPictureAlignment.MiddleCenter)
    End Sub
    Public Sub ShowTiles(cad As Avax.cAvax, fileName As String)
        cad.SetAvaxProperty(Avax.AvaxProp.TilePictureFile_p, CObj(fileName))
    End Sub
    Public Sub ReloadImages(cad As Avax.cAvax)
        cad.Regen()
    End Sub
    Public Sub ClearImages(cad As Avax.cAvax)
        cad.SetAvaxProperty(Avax.AvaxProp.TilePictureFile_p, "")
        cad.SetAvaxProperty(Avax.AvaxProp.BackgroundPictureFile_p, "")
    End Sub
End Module
```

Shape fill and display settings

FillType_p returns an AvaxFill_t value. Read it, change its fields, and pass it back with SetAvaxProperty. The short Add_Shape overloads retain their R2 default-fill behavior; use the full overload to supply a chosen descriptor explicitly. The color fields are palette indices.

```
Dim fill As Avax.AvaxFill_t = DirectCast(
    cAvax1.GetAvaxProperty(Avax.AvaxProp.FillType_p), Avax.AvaxFill_t)
fill.iType = CByte(Avax.AvaxFillType.Solid_f)
fill.iInForeColor = 1
fill.zOrder = 10
cAvax1.SetAvaxProperty(Avax.AvaxProp.FillType_p, CObj(fill))
cAvax1.SetAvaxProperty(Avax.AvaxProp.PatternsOn_p, True)
cAvax1.SetAvaxProperty(Avax.AvaxProp.ShapeZorderOn_p, True)

Dim metadata As Object = Nothing
Dim shapeHandle As Integer = cAvax1.Add_ShapeRectangle(
    0.0, 0.0, 40.0, 20.0, metadata,
    fill.iType, fill.lHatchIndex, fill.iInForeColor, fill.iInBackColor,
    fill.sPattern, fill.zOrder, fill.lFlags, 1, 1, 0, 0)
```

See Dimension formatting properties for units, precision, factors, labels, endpoint marks and a complete VB.NET example.

See Transformation properties for keep-original modes, uniform scaling, sensitivity and a complete VB.NET example.

See Selection appearance properties for the selection frame, grips, text dither, transparent window fill and a complete VB.NET example.

See Selection and crosshair properties for cursor sizes, stroke styles, RGB colors, visibility and a complete VB.NET example.

See Drawing coordinate limits for the six Double coordinate properties, sequential assignment, transactional XML validation, and native-file preservation on invalid bounds.

See Idle selection and spatial queries for SelectStay_p and SelectInsideShapeOn_p, their R2 defaults and their effect on the selection pointer.

Snap marker and hidden control rows

AutoOSnapCircleRadiusInPixels_p accepts an Integer pixel radius. Its default is 6; rendering bounds the radius to 2 through 128 while retaining the requested setting. It does not change AutoOSnapApproachInPixels_p, which controls point acquisition.

AvaxControlShrinkInvisibleOn_p defaults to True. Hidden active controls reclaim their world height from controls with a strictly lower minimum world Y, independently of X overlap. Toggling the setting updates native editor positions immediately; it leaves stored drawing coordinates and the current drawing command intact. False restores the original layout.

Both settings belong to the control instance and are included in SaveAvaxPropertiesToXML/LoadAvaxPropertiesFromXML. Invalid values are rejected without replacing the last accepted setting.

VB.NET example

```
Option Strict On
Imports Avax

Public Module VisualFeedbackSettings
    ' Call on the owning UI thread after StartAvax.
    Public Function ConfigureVisualFeedback(cad As cAvax) As Boolean
        Dim markerRadius As Object = 12
        Dim shrinkHiddenRows As Object = True
        Return cad.SetAvaxProperty(AvaxProp.AutoOSnapCircleRadiusInPixels_p, markerRadius) AndAlso
               cad.SetAvaxProperty(AvaxProp.AvaxControlShrinkInvisibleOn_p, shrinkHiddenRows)
    End Function
End Module
```

Animation settings

AnimationOn_p is Boolean and defaults to True. False displays the first frame of an animated picture or fill. AnimationInterVal_p is an Integer interval in milliseconds and defaults to 10. Changes apply to the current drawing; an interval set while animation is disabled takes effect when it is enabled again.

Zero and negative intervals are retained as settings but playback safely uses one millisecond. Invalid or out-of-range values are rejected without changing the previous setting. Printing uses the first frame. These properties select frames in animated image sequences; they do not replace an AVI movie's own playback timing.

VB.NET example

```
Option Strict On
Imports Avax

Public Module AnimationSettings
    ' Call on the owning UI thread after StartAvax.
    Public Function ConfigureAnimation(cad As cAvax, enabled As Boolean) As Boolean
        Dim intervalMilliseconds As Object = 40
        Dim animate As Object = enabled
        Return cad.SetAvaxProperty(AvaxProp.AnimationInterVal_p, intervalMilliseconds) AndAlso
               cad.SetAvaxProperty(AvaxProp.AnimationOn_p, animate)
    End Function
End Module
```

Rendering preferences

OpenGL_p is Boolean and defaults to False, selecting CPU rendering as in R2. True requests GPU rendering. A failure of the current GPU surface falls back to CPU rendering; GetAvaxProperty then returns False. Backend changes preserve drawing geometry, coordinates and the current Line command.

BackBitmapSize_p is an Integer requested cache edge in pixels, with default 6000. Legacy bitmap rendering uses the tablet/desktop default of 3000/6000 when the value is zero; a negative value declines cache creation while retaining an existing cache. Adaptive overview rendering bounds its edge to 256 through 8192 pixels and uses 4096 for a nonpositive request. These settings affect image allocation, not drawing scale.

SilenceOn_p is a retained Boolean preference, False by default. Store and read it when coordinating silent behavior in the host application. Host-owned sounds and dialog policy are configured by the host; DialogEnabled_p remains a separate setting. All three values belong to the control instance and are included in settings XML.

VB.NET example

```
Option Strict On
Imports Avax

Public Module RenderingSettings
```

```
    ' Call on the owning UI thread after StartAvax.
    Public Function ConfigureRendering(cad As cAvax, useGpu As Boolean) As Boolean
        Dim cacheSizeInPixels As Object = 2048
        Dim gpuRendering As Object = useGpu
        Dim silentPreference As Object = True
        Return cad.SetAvaxProperty(AvaxProp.BackBitmapSize, cacheSizeInPixels) AndAlso
               cad.SetAvaxProperty(AvaxProp.OpenGL, gpuRendering) AndAlso
               cad.SetAvaxProperty(AvaxProp.SilenceOn, silentPreference)
    End Function
End Module
```

AvaxScrollBarsStatus

Selects which scroll bars the control displays.

Parameter	Description	Value
Both	Both ScrollBars	3
Vertical	Vertical ScrollBars	2
horizontal	Horizontal ScrollBars	1
NoBars	No ScrollBars	0

AvaxShapeType

Identifies the geometric kind of a shape.

Parameter	Description	Value
Any_s	Any Avax Shape Type	0
Rectangle_s	Rectangle	1
Circle_s	Circle	2
Ellipse_s	Ellipse	3
RegularPolygon_s	Regular Polygon	4
Arc_s	Arc	5
Line_s	Line	6
Polyline_s	Polyline	7
Text_s	Text	8
Point_s	Point	9

Additional declared values

Name	Value	Description
Document_s	10	Document text shape. Its text-layout data defines a bounded rich-document area, which can contain formatted text and embedded content.

AvaxUndo

Identifies an operation in the undo or redo history.

Parameter	Description	Value
None_u	None	0
Erase_u	Erase	1
Unerase_u	Unerase	2
Create_u	Create	3
Rotate_u	Rotate	4
Mirror_u	Mirror	5
Scale_u	Scale	6
Move_u	Move	7
Transform_u	Transform	8
Zoom_u	Recorded viewport change. Enable adding these steps to normal Undo/Redo with ZoomUndo_On_c; the default is off. CurrentUndo/CurrentRedo and Undo/Redo availability events report Zoom_u. Undo restores the before view and Redo restores the after view without modifying drawing geometry or dirty state. View steps participate in mixed edit and recording/continuation groups. A completed drag is one step. Previous_c retains its independent 50-view history; ZoomMemoryOn_p controls caching separately.	9
Stretch_u	Stretch	10
Freeze_u	Freeze	11
Thaw_u	Thaw	12
Change_u	Change	13

See Undo and Redo for availability notifications, committed history queries and the R2 Redo event/query label distinction.

Zoom_u

Identifies a recorded view change. View recording in the normal Undo/Redo history is off by default. Enable it with ZoomUndo_On_c and stop adding records with ZoomUndo_Off_c. Existing records remain available. CurrentUndo and CurrentRedo return Zoom_u for these records, and Undo/Redo availability events use the same value.

Undo and Redo restore the Double viewport bounds without changing drawing objects or marking the drawing modified. View changes participate in chronological mixed edits, UndoRecorderOn and ContinueUndo groups. A completed pan or dynamic-zoom drag contributes one record. The separate Previous_c list retains up to 50 views independently of ZoomUndo and ZoomMemoryOn_p. Assigning a new drawing clears both histories; its initial automatic fit adds no undo step.

VB.NET example

```
Option Strict On
Imports Avax

Public Module ZoomUndoExample
    ' Call on the owning UI thread after StartAvax.
    Public Sub ZoomAndRestore(cad As cAvax)
        cad.Command = AvaxCommand.ZoomUndo_On_c
        cad.ZoomIn()
        Dim kind As AvaxUndo
        If cad.CurrentUndo(kind) > 0 AndAlso kind = AvaxUndo.Zoom_u Then
            cad.Command = AvaxCommand.Undo_c
            cad.Command = AvaxCommand.Redo_c
        End If
    End Sub
End Module
```

Additional topics and reference guides

Use the linked topic list in Contents or the PDF bookmarks to locate a member or guide.

Add_Document

API reference

Add_Document

Add_ShapeDocument

API reference

Add_ShapeDocument

Control API reference

Home

The reference follows the methods, properties, events and enumerations in the original AVAX developer help. Each member topic includes its purpose, current v3 declaration, parameter or value descriptions and an associated VB.NET example.

- Methods
- Properties
- Events
- Enumerations

Property geometry layouts

Arc commands and segmentation

API reference

Picking an arc

Arc_c accepts start, end and a point through the arc. ArcStartIntermediateEnd_c accepts start, a point through the arc and end. ArcStartEndCenter_c accepts start, end and a center pick; the center pick is projected onto the endpoints' perpendicular bisector to obtain a circle through both endpoints. Coincident endpoints and invalid three-point geometry are rejected while earlier accepted points remain available for correction.

Standalone completion

A standalone arc creates one object and ends after its final accepted point. CurveArcOn_p=False stores a segmented polyline; True stores an analytic curve. Completion restores the SelectStay_p idle mode and consumes pending input before Change and NewUserItems, so a callback may start a newer operation without being overwritten. Starting an arc clears an earlier object selection.

Joining a polyline

When changing from Polyline_c to an arc command, the last accepted polyline vertex becomes the arc start. With CurveArcOn_p=False the completed arc joins the pending polyline and returns to line continuation. Close_c closes and commits the complete chain. Switching to another command or cancelling commits accepted polyline geometry while discarding unaccepted arc picks. With CurveArcOn_p=True the prefix and analytic arc are separate objects committed as one batch and one Undo step; the arc tool then ends.

Arc resolution properties

CircleSectors_p is the Integer full-circle resolution. Zero selects the default 72 sectors. Valid explicit resolutions are 3 through 1000000. ArcSegmentsLength_p is a nonnegative Double: zero uses CircleSectors_p; a positive length calculates floor(2*pi*radius/length), clamped by MinCircleSectors_p and MaxCircleSectors_p. Their defaults are 6 and 120, and the active limits must satisfy 3 <= minimum <= maximum <= 1000000. Sequential property assignment may temporarily leave inactive settings inconsistent; invalid active settings reject geometry preparation without altering the drawing.

For a sweep larger than one full-circle sector, the number of chords is the ceiling of sweep/(2*pi/resolution), including a shorter final interval. Very short arcs at or below one sector use resolution/4 intervals. The same settings apply to Add_Arc, which always creates segmented geometry independently of CurveArcOn_p. Add_CurveArc requests analytic geometry. Coincident positive endpoint rays retain the complete-circle special case.

Double XYZ and atomic edits

Coordinates and segment length use Double; resolution, limits and handles use Integer in v3. Segment endpoints retain their supplied Z values, with safe interpolation between them. This corrects the loss of endpoint elevation in the older segmented representation. Current pen and separate elevation/height metadata are preserved. Nonfinite coordinates, overflowing extents or invalid active resolution settings leave accepted input, prior drawing objects and Undo/Redo history intact.

Complete VB.NET examples

Draw one standalone arc

```
Option Strict On
Imports Avax

Public Module Help_DrawStandaloneArc
    Public Function Draw(cad As cAvax, analytic As Boolean) As Boolean
        cad.SetAvaxProperty(AvaxProp.CurveArcOn_p, (analytic))
        cad.Command = AvaxCommand.Arc_c
        Dim x As Double = 0, y As Double = 0, z As Double = 7.5
        If Not cad.SetCoordinate(x, y, z) Then Return False
        x = 10 : y = 0 : z = -3.25
        If Not cad.SetCoordinate(x, y, z) Then Return False
        x = 5 : y = 5 : z = 11.75
        Return cad.SetCoordinate(x, y, z)
    End Function
End Module
```

Join a line, an arc and a closing edge

```
Option Strict On
Imports Avax

Public Module Help_DrawArcChain
    Public Function Draw(cad As cAvax) As Boolean
        cad.SetAvaxProperty(AvaxProp.CurveArcOn, False)
        cad.Command = AvaxCommand.Polyline
        If Not cad.SetCoordinate(-5, 0, 0) Then Return False
        If Not cad.SetCoordinate(0, 0, 0) Then Return False
        cad.Command = AvaxCommand.Arc
        If Not cad.SetCoordinate(10, 0, 0) Then Return False
        If Not cad.SetCoordinate(5, 5, 0) Then Return False
        If Not cad.SetCoordinate(15, 0, 0) Then Return False
        cad.Command = AvaxCommand.Close
        Return cad.Command <> AvaxCommand.Polyline
    End Function
End Module
```

Configure a target chord length and full-circle limits

```
Option Strict On
Imports Avax

Public Module Help_SetArcResolution
    Public Sub Configure(cad As cAvax)
        cad.SetAvaxProperty(AvaxProp.CircleSectors, 24)
        cad.SetAvaxProperty(AvaxProp.ArcSegmentsLength, 2.5R)
        cad.SetAvaxProperty(AvaxProp.MinCircleSectors, 6)
        cad.SetAvaxProperty(AvaxProp.MaxCircleSectors, 120)
    End Sub
End Module
```

Line, Point and Rectangle commands

API reference

Start the Windows Forms control with StartAvax before issuing commands. Coordinates accepted by SetCoordinate are Double in v3. Every created root handle and the one-based NewUserItems.NewHandles array use Integer.

Line_c

Creates connected straight line segments. The first coordinate establishes the XYZ start without changing the drawing. Each subsequent accepted coordinate creates one segment and becomes the start of the next segment. GetLineStartPoint returns the current XYZ continuation point. Both endpoints retain their independent Z coordinates. Cancel_c or Cancel_Keep_Select_c abandons the next unfinished segment and keeps completed geometry.

Point_c

Creates a point for each accepted coordinate and stays active for further points. The original R2 drawing tool is planar: its stored vertex Z is zero. This differs from the Add_Point method, which accepts and stores an explicit Z coordinate. Pen3DElevetion and Pen3DHeight remain separate object metadata. Point_c uses PenPointStyle, whereas Line_c and Rectangle_c use PenLineStyle. A point tool has no unfinished segment for GetLineStartPoint.

Rectangle_c

Accepts two opposite XY corners and creates a closed polyline with four edges, beginning with the vertical edge from the first corner. Either width or height may be negative. The rectangle vertices have Z=0, as in R2; coordinate Z does not change the separate elevation metadata. Zero width or height is rejected while retaining the first corner for correction. After successful creation the tool ends and returns to the SelectStay_p idle mode before Change and NewUserItems callbacks. Start Rectangle_c again to draw another rectangle.

Selection, metadata and Undo

Starting these drawing tools clears the previous selection; modification tools retain their selected sources. Current pen color, width, layer and 3D metadata apply to newly created objects. Each accepted point, line segment or rectangle is one creation operation, one creation batch and one Undo step. Undo and Redo commands can be issued after cancelling incomplete input.

Invalid input and host events

SetCoordinate returns False for nonfinite coordinates, unavailable construction or invalid geometry. It preserves prior accepted input on a rejected final coordinate. If generated geometry overflows, the control reports the error without partially changing the drawing. A host event may start a new command, supply its first point, or end the control; an older drawing operation does not overwrite that newer work. Host event-handler exceptions propagate after geometry and consumed input have committed.

Complete VB.NET examples

Create two connected XYZ line segments

```
Option Strict On
Imports Avax

Public Module Help_DrawConnectedLines
    Public Function Draw(cad As cAvax, origin As Double) As Integer
        Dim before As Integer = cad.MaxItems()
        cad.Command = AvaxCommand.Line_c
        Dim x As Double = origin + 10.125, y As Double = 20.25, z As Double = 7.5
        If Not cad.SetCoordinate(x, y, z) Then Return 0
        x = origin + 30.5 : y = 40.75 : z = -3.25
        If Not cad.SetCoordinate(x, y, z) Then Return 0
        x = origin + 50.875 : y = 60.5 : z = 11.75
        If Not cad.SetCoordinate(x, y, z) Then Return 1
        cad.Command = AvaxCommand.Cancel_Keep_Select_c
        Return cad.MaxItems() - before
    End Function
End Module
```

Create a planar point with the current point style

```
Option Strict On
Imports Avax

Public Module Help_DrawPlanarPoint
    Public Function Draw(cad As cAvax, x As Double, y As Double) As Boolean
        cad.Command = AvaxCommand.Point_c
        Dim z As Double = 0
        Return cad.SetCoordinate(x, y, z)
    End Function
End Module
```

Create one rectangular closed polyline

```
Option Strict On
Imports Avax

Public Module Help_DrawRectangle
    Public Function Draw(cad As cAvax, x0 As Double, y0 As Double,
                         x1 As Double, y1 As Double) As Boolean
        cad.Command = AvaxCommand.Rectangle_c
        Dim z As Double = 0
        If Not cad.SetCoordinate(x0, y0, z) Then Return False
        Return cad.SetCoordinate(x1, y1, z)
    End Function
End Module
```

ByLayer color and line style

API reference

AVAX-CAD R3 and Melobytes Draw expose ByLayer in their object color and line-style choices, including new-object pen defaults and selected-object properties. Their Layers dialogs edit each layer's concrete color and line style. Color and line-style inheritance are independent; choosing a fixed value for one keeps the other setting. In Melobytes Draw, the object color choice controls the outline, including styled text outlines; fill and gradient colors remain separate appearance settings.

Color and line style can independently follow an object's layer. Changing the object's layer, the layer color or the layer line style updates the effective appearance. Explicit object attributes remain unchanged. Layer 0 contents of imported blocks follow the containing block's layer. Rendering uses cached effective values; it does not search the layer table on every mouse move.

PenColorByLayer

PenLineStyleByLayer

SetItemByLayer

SetItemsByLayer

GetItemByLayer

SetLayerColor

GetLayerColor

SetLayerLineStyle

GetLayerLineStyle

Example

Run after StartAvax. This example uses existing layer 1 and built-in line style 1.

```
cAvax1.SetLayerColor(1, System.Drawing.ColorTranslator.ToOle(System.Drawing.Color.Red))
cAvax1.SetLayerLineStyle(1, 1)
cAvax1.PenLayer = 1
cAvax1.PenColorByLayer = True
cAvax1.PenLineStyleByLayer = True
Dim handle As Integer = cAvax1.Add_Line(0, 0, 0, 10, 5, 0)

' The existing line now becomes blue and keeps following layer 1.
cAvax1.SetLayerColor(1, System.Drawing.ColorTranslator.ToOle(System.Drawing.Color.Blue))

' Freeze only the color; the line style continues to follow the layer.
cAvax1.SetItemByLayer(handle, False, True)
```

Storage and CAD exchange

Native .avax drawings retain the inheritance flags, layer appearance and per-entity line-type scale. Files containing inherited objects use native format version 9; older files remain readable. Legacy .avx and .blk retain the effective appearance for older readers and supplementary inheritance data for v3. Older readers do not implement the new dynamic behavior.

DXF/DWG import retains ByLayer independently for color and line style. Export writes the actual ByLayer setting and the layer color/line-type resources, rather than replacing the setting with its resolved color or dash pattern. Existing CAD format and entity limitations still apply.

GetPaperSize

API reference

GetPaperSize

Add_TextC

API reference

Add_TextC

Circle and Ellipse commands

API reference

These commands construct a complete analytic ellipse object, even when both radii are equal. Start the control with StartAvax before issuing a command. SetCoordinate accepts Double coordinates in v3; root handles and NewUserItems.NewHandles use Integer.

Circle_c

Accept the center, then a point on the circumference. The XY distance determines the radius.

Circle2p_c

Accept two opposite endpoints of a diameter. The center is their midpoint and the radius is half their XY distance.

Circle3p_c

Accept three distinct, noncollinear XY points on the circumference. The control calculates their circumcircle. A rejected final point leaves the earlier points available for correction.

Ellipse_c

Accept the center, then the opposite corner of its positive-quadrant radius box. The absolute X and Y differences are the two radii; either pick direction is supported. The original R2 axis-aligned fallback is preserved: if one difference is zero, the nonzero difference supplies both radii.

Completion, selection and representation

Each command creates one object and then ends, restoring automatic idle selection according to SelectStay_p. Start the command again for another object. Starting a new circle or ellipse clears prior selection. Completion consumes pending points before Change and NewUserItems, so a callback can query the completed drawing, start another command or end the control. Host exceptions propagate after the object has committed.

CurveArcOn_p selects the representation of interactive arcs; these complete circle and ellipse commands remain analytic ellipse objects in either setting, as in R2. Current color, width, line style, layer, Pen3DElevetion and Pen3DHeight apply independently to the new object. Each object has one Undo step and can be restored with Redo.

XYZ and Double precision

All four v3 commands retain the first point's Z as the center elevation. This preserves the first-point behavior of center/radius, diameter and ellipse construction and corrects the R2 three-point circle's loss of that elevation. Radius calculations use XY coordinates. Coordinate Z remains separate from Pen3DElevetion and Pen3DHeight metadata. Tiny and huge representable geometry is retained; nonfinite input, coincident points, collinear three-point input and unrepresentable extents are rejected without partial geometry.

Complete VB.NET examples

Circle from a center and radius

```
Option Strict On
Imports Avax

Public Module Help_DrawCircle
    Public Function Draw(cad As cAvax, cx As Double, cy As Double, radius As Double) As Boolean
        cad.Command = AvaxCommand.Circle_c
        Dim z As Double = 0, x As Double = cx + radius
        If Not cad.SetCoordinate(cx, cy, z) Then Return False
        Return cad.SetCoordinate(x, cy, z)
    End Function
End Module
```

Circle through three points

```
Option Strict On
Imports Avax
```

```
Public Module Help_DrawCircle3Points
    Public Function Draw(cad As cAvax) As Boolean
        cad.Command = AvaxCommand.Circle3p_c
        Dim x As Double = 15, y As Double = 10, z As Double = 0
        If Not cad.SetCoordinate(x, y, z) Then Return False
        x = 10 : y = 15
        If Not cad.SetCoordinate(x, y, z) Then Return False
        x = 5 : y = 10
        Return cad.SetCoordinate(x, y, z)
    End Function
End Module
```

Ellipse from its center and radii

```
Option Strict On
Imports Avax

Public Module Help_DrawEllipse
    Public Function Draw(cad As cAvax, cx As Double, cy As Double,
                         radiusX As Double, radiusY As Double) As Boolean
        cad.Command = AvaxCommand.Ellipse_c
        Dim z As Double = 0, x As Double = cx + radiusX, y As Double = cy + radiusY
        If Not cad.SetCoordinate(cx, cy, z) Then Return False
        Return cad.SetCoordinate(x, y, z)
    End Function
End Module
```

CommandFilePicker

API reference

CommandFilePicker

cAvax Member Reference

API reference

AddCompoundPath

GetCompoundPath

ChangeCompoundPath

ConvertShapeToCurves

SetPathNodeMode

BeginCompoundPath

FinishCompoundPath

IsCreatingCompoundPath

CompoundPathSegmentKind

CompoundPathClosed

BeginCurveNodeEditing

GetPathSegmentParameters

ChangePathSegmentParameter

SplitPathSegment

ChangePathSegmentKind

ReverseCompoundPath

CombineCompoundPaths

cAvControls Member Reference

API reference

`Avax.cAvControls` exposes embedded drawing controls by parent handle and child index. A registry slot is a separate one-based number. Use the helper on the drawing's UI thread after `StartAvax`. A new instance attaches to the active drawing. Queries see live owner edits; setters update both native editors and persistent drawing data.

Attachment and explicit recording

`InitAvaxControls` starts explicit recording. Pass one-based Double world polygons to `AddAvaxControl`. The first valid addition replaces the previous pass. Invalid input preserves that pass. Recording does not add drawing entities. Set `avIndex = 0` to return to automatic attachment. Removing a drawing entity leaves an inactive slot; replacing the drawing clears attached slots.

Drawing and resource ownership

Use `RefreshControls(Integer)` or the `m_hDC` field for a borrowed Windows drawing DC. Its origin must be the viewport's top-left pixel, excluding ruler and scrollbar margins. The caller owns the DC and keeps it alive while rendering, clipping or open lists use it. The helper owns only its temporary surfaces and regions. Call `KillControls` when finished, while the owner and borrowed DC remain valid.

Refresh restores valid previous control backgrounds and paints visible siblings in order. This removes hidden/moved recorded controls without erasing underlying geometry. Forms stay behind other fields. Popup lists are painted last. Saved pixels are discarded after view or drawing revision changes. If an application replaces pixels in an external DC without changing the drawing/view, reinitialize the recording pass before using that destination again.

Pointer input and events

Forward viewport mouse coordinates to `MouseInControl`, then call `ParentHandleControlClick` for the click identity. This helper does not install a Windows message filter. `PointInControl` is a geometric query. Interactive picking excludes hidden, disabled, locked and erased fields. A drawing command or active native editor retains input ownership.

Set `bMyList = True` for helper-rendered combo lists. Keep the same registry across movement and clicks so its open-list state persists. A list click returns an absolute one-based item index; the application decides how to update its text/value. Helper click identification does not emit a second owner click event.

Refresh requests applicable behavior/caption/text/list/value/picture/tooltip data from the owner; PauseAvaxEvents suppresses requests. Explicit-record behavior is requested first, and hiding the record suppresses later requests. Results affect transient display. A callback that clears the helper, replaces its owner document or edits the current record cancels stale work. Erasing a root during initial widget creation retires its native controls and reports the erased parent once after the request unwinds. Exceptions propagate; recursive painting returns False.

Types and errors

Former Single coordinates/scales are Double and former Short values are Integer. Geometry uses world units; pointer positions and GetControlSize use viewport pixels. Colors are OLE RGB. Arrays of vertices and list items are one-based. Missing identities return zero, False or Nothing according to the member. Invalid numbers throw InvalidDataException; malformed/nonfinite geometry throws ArgumentException; invalid border width throws ArgumentOutOfRangeException. Cross-thread owner access throws InvalidOperationException.

Members

Declaration	Behavior
Sub New()	Creates a registry associated lazily with the active drawing. Use it on that drawing's UI thread and call KillControls when finished. It never owns a caller-supplied drawing DC.
Sub AddAvaxControl(ByRef lpPointX As Double(), ByRef lpPointY As Double(), ByRef nCount As Integer, ByRef mFillType As AvaxFill_t, ByRef ShapeType As AvaxShapeType, ByRef shBrdClr As Integer, ByRef lBrdWidth As Integer, ByRef shBrdStyle As Integer, ByRef fToDel As Boolean, ByRef TextX1 As Double, ByRef TextY1 As Double, ByRef TextX2 As Double, ByRef TextY2 As Double)	Records a polygon and decoded pattern. Arrays are one-based Double coordinates; nCount is the number of polygon vertices. Fill type 9, a nonzero lFlags parent, and lHatchIndex child index identify the control. TextX1/Y1 is the text anchor and TextX2/Y2 is the other end of its height axis, in world coordinates. The axis length determines text height; its direction determines text rotation. For ordinary bottom-left coordinates, a vertical axis from top to bottom produces horizontal text. Label MultiLines divides this cell height, and FontScale scales it. A zero-length axis uses the ordinary automatic text layout. Layout offsets move the text axis together with the polygon. Width and style are Integer. A duplicate identity replaces its existing slot. fToDel or calls outside recording mode are ignored. Geometry is copied.
Function AvaxControlGetAllControls(ByVal ParentHandle As Integer(), ByVal Index As Integer(), ByVal sName As String(), ByVal sTag As String(), ByVal sData As String()) As Integer	Retains the old ByVal-array signature. Fills available slots of caller-preallocated arrays and returns the full registry count. Nothing arrays can be used for count-only queries. Unlike the old implementation, it does not silently replace only local array references.
Function AvaxControlGetData(ByRef ParentHandle As Integer, ByRef Index As Integer) As String	Returns the encoded current pattern for a parent/index, including erased records. Missing identities return Nothing. Lists and unknown extension fields are retained.
Function AvaxControlGetShape(ByRef ParentHandle As Integer, ByRef Index As Integer, ByRef ShapeX As Double(), ByRef ShapeY As Double()) As Integer	Returns a vertex count and newly allocated one-based Double arrays. Missing identities return zero and leave both caller arrays unchanged, including Nothing. The public cAvax query follows the same R2 rule. Caller edits cannot alter the stored polygon.
Function AvaxControlRefresh(ByRef ParentHandle As Integer, ByRef Index As Integer) As Boolean	Refreshes callbacks for the given parent/index and repaints the visible controls in overlap order. Returns False if the identity is absent/erased or rendering cannot proceed. Callback results are transient; explicit data setters persist attached drawing changes.
Function AvaxControlRefreshAll() As Boolean	Refreshes all existing controls using m_hDC. Forms are drawn behind other controls; zOrder and registration order determine the remaining overlap order. Invalidated backgrounds are discarded after drawing/view changes.
Function AvaxControlSetData(ByRef ParentHandle As Integer, ByRef Index As Integer, ByRef sData As String) As Boolean	Validates and replaces the pattern for an existing parent/index. False means the identity is absent or its live editor no longer exists. Invalid data throws before replacing registry data.
Function BeginControlSelectClipRegion(ByRef lRegCnt As Integer) As Boolean	Temporarily excludes the one-based control polygon from the destination clip, restricted to the current viewport. Returns False without a live record, owner or DC. A repeated Begin restores the previous saved state before installing a new exclusion. The original caller clip is intersected, never enlarged.
Function CanFocus(ByRef ParentHandle As Integer, ByRef Index As Integer) As Boolean	Returns True only for an existing, visible, enabled, unlocked TextBox or ComboBox. This query does not move focus. Disabled and locked records correctly return False.
Function CheckActionHandles(ByRef h As Integer()) As Boolean	Returns True if any active record belongs to a parent in the supplied one-based Integer array. Nothing and erased parents do not match.
Function CloseAllListControls() As Boolean	Closes every open custom list and restores valid backgrounds. Returns True if at least one list was open, otherwise False.
Function CloseOtherlListControls(ByRef lControlCnt As Integer) As Boolean	Closes every open list except the one belonging to lControlCnt (a one-based slot). Returns whether any list was closed. The historical spelling is retained.
Function ControlCntByHandle(ByRef ParentHandle As Integer, ByRef Index As Integer) As Integer	Returns the one-based slot for ParentHandle and Index, or zero if absent. Erased records retain their slots.
Function ControlDecimals(ByVal lRegCnt As Integer) As Integer	Returns the Integer Decimals field (default -1); invalid slots return zero.
Function ControlDropDownHeight(ByVal lRegCnt As Integer) As Integer	Returns Integer DropDownHeight (default zero), including values above the former Short limit.

Declaration	Behavior
Function ControlMousePointer(ByRef lRegCnt As Integer) As Integer	Returns the stored Integer MousePointer code; invalid slots return zero. This getter does not itself change a cursor.
Function ControlToolTip(ByRef lRegCnt As Integer) As String	Returns TooltipText with [P] and [p] expanded to CRLF. Invalid slots return Nothing.
Sub CreateOpenListRegion(ByRef lRegCnt As Integer)	Updates the hovered row of the open list belonging to a one-based control slot. Hit regions are derived from world coordinates at query time instead of retaining stale regions across zoom changes. Missing slots and closed lists are ignored.
Function EndControlSelectClipRegion() As Boolean	Restores the exact DC saved by Begin, even if the legacy m_hDC field changed. Returns the native restore result; False if no exclusion is active. Reinitialization and KillControls also end the exclusion.
Sub EraseAvaxControl(ByRef mFillType As AvaxFill_t)	Erases only the parent/index represented by a type-9 fill; siblings remain intact.
Sub EraseChildAvaxControl(ByRef AvaxHandle As Integer)	Marks every recorded child of a nonzero parent erased, preserving values and slots. Attached editors are hidden without deleting drawing entities. Raises the owner erasure event once for an actual transition.
Function ExistingControls() As Integer	Returns the number of present, unerased records. Invisible, disabled and locked records still exist.
Function GetComboData(ByRef lRegCnt As Integer, ByRef bSorted As Boolean, ByRef iStyle As Integer) As Boolean	Returns Sorted and Integer style for an existing ComboBox; returns False for other kinds. Fixes the old enum-shadowing error.
Function GetControl(ByRef lRegCnt As Integer, ByRef eType As AvaxControlType, ByRef ParentHandle As Integer, ByRef lIndex As Integer, ByRef sName As String, ByRef sTag As String, ByRef lStatus As AvaxControlStatus) As Boolean	Returns kind, parent, child index, name, tag and status through ByRef outputs. Returns False for an invalid slot without overwriting outputs. Erased records remain queryable.
Function GetControlBehavior(ByRef lRegCnt As Integer, ByRef bVisible As Boolean, ByRef bLocked As Boolean, ByRef bEnable As Boolean) As Boolean	Returns visible, locked and enabled flags for an existing record. Missing or erased records return False.
Function GetControlColors(ByRef lRegCnt As Integer, ByRef lStartBackColor As Integer, ByRef lEndBackColor As Integer, ByRef lHighlightColor As Integer, ByRef lTextFillColor As Integer, ByRef lTextPathColor As Integer) As Boolean	Returns the five OLE RGB colors: background start/end, highlight, text fill and text outline. Missing or erased records return False.
Function GetControlFont(ByRef lRegCnt As Integer, ByRef sFontName As String, ByRef nFontScale As Double, ByRef bFontBold As Boolean, ByRef bFontItalic As Boolean, ByRef bFontUnderline As Boolean, ByRef bFontStrikeOut As Boolean, ByRef lFontCharset As Integer) As Boolean	Returns family, Double scale, four style flags and Integer charset. Scale is adjusted by 96/device DPI for an attached drawing. An explicit zero scale uses 2/3 before DPI adjustment. Missing or erased records return False.
Function GetControlSize(ByRef lRegCnt As Integer, ByRef lLeft As Integer, ByRef lTop As Integer, ByRef lWidth As Integer, ByRef lHeigth As Integer, ByRef iMutliLines As Integer) As Boolean	Returns projected viewport pixels (left, top, width, height) and the multiline count, which is at least one. Requires a started owner. Includes invisible-control shrink displacement and current origin/scale. Each edge is rounded separately from Double coordinates, matching the integer hit region and native editor bounds. Negative offscreen positions remain signed. Missing slots, an unavailable owner, or bounds that cannot fit Integer pixel outputs return False without changing the outputs.
Function GetControlText(ByRef lRegCnt As Integer) As String	Returns current Text for a recorded slot, including erased records; Nothing if the slot is invalid.
Function GetControlValue(ByRef lRegCnt As Integer) As Integer	Returns the stored Integer Value, or zero for an invalid slot. Direct storage retains the full promoted Integer range.
Function GetDxDy(ByRef mFillType As AvaxFill_t, ByRef dx As Double, ByRef dy As Double) As Boolean	Returns Double world displacement for a type-9 fill identity. False leaves outputs unchanged if the fill is not a control or its identity is absent.
Function GetPattern(ByVal mFillType As AvaxFill_t) As String	For a type-9 fill, returns current registry data when the identity exists, otherwise the supplied sPattern. Other fill types return Nothing.

Declaration	Behavior
Sub InitAvaxControls(ByVal hDC As Integer, ByRef lBackColor As Integer)	Starts an explicit recording pass and restores outstanding helper clips/backgrounds. The first valid AddAvaxControl replaces prior records. hDC is borrowed and its origin is the drawing viewport. Pass the 32 significant Windows handle bits through the original Integer signature. Zero permits data and hit queries without painting. lBackColor is an OLE RGB color.
Sub KillAllOpenList()	Closes all custom popups and releases their background images in reverse order. Does not remove the registered controls.
Sub KillControls()	Ends helper clipping, closes popups, restores valid recorded-control backgrounds and clears slots/current fields. Remains in explicit recording mode; it does not delete drawing entities. Reassign avIndex to zero to return to automatic attachment.
Sub KillOpenList(ByRef ParentHandle As Integer, ByRef Index As Integer)	Closes the popup for ParentHandle/Index, restores its saved background when still valid and frees owned resources. Higher overlapping popups are preserved and their backgrounds are recaptured over the new underlay, so later closure cannot resurrect the removed popup. Missing identities are ignored. Does not remove the control.
Function MaxControls() As Integer	Returns the number of registry slots, including erased records. Slots are one-based and remain stable within a drawing or recording pass.
Function MouseInControl(ByRef X As Integer, ByRef Y As Integer) As Integer	Updates ClientX/ClientY, MouseOnControlCnt and hover status from viewport pixels. Returns a one-based interactive slot or -1. Hidden, erased, disabled and locked records cannot intercept input. Returns -1 while a drawing operation or native text editor owns input. Hosts using the helper directly must forward mouse movement to this method.
Function ParentHandleControlClick(ByRef eType As AvaxControlType, ByRef lIndex As Integer, ByRef ListIndex As Integer, ByRef sName As String, ByRef sTag As String) As Integer	Uses the latest helper pointer position (or the owner pointer before the first helper query). Returns the parent handle, or zero on no interactive target. ByRef outputs receive control type, child index, name, tag and the absolute one-based selected list row; ListIndex is zero for a normal control click. It closes other custom lists and toggles a custom combo popup when bMyList is True. Publishes CurrentAvaxControl fields. Does not raise a second click event or change host-owned text/value. Recursive clicks and clicks requested during helper painting return zero and reset ListIndex without replacing the other caller outputs. The click guard is released even when a host data callback throws.
Function PointInControl(ByRef X As Integer, ByRef Y As Integer) As Integer	Updates ClientX/ClientY from Integer viewport pixels and returns a one-based slot or -1. Open-list polygons have priority. Where popups overlap, the last painted (last registered) eligible popup is tested first, independently of the base controls' z-order. Hidden/erased controls are excluded; disabled and locked controls remain available to this geometric query. Regions follow current Double world coordinates and the current view. Does not emit a click.
Function ReCalcControls() As Boolean	Recalculates the displacement caused by invisible controls when the owner enables AvaxControlShrinkInvisibleOn. Each active hidden control adds its original world height to controls with a strictly lower minimum world Y, independently of horizontal overlap. Equal-height rows do not shift each other. Erased hidden controls do not reclaim space. Disabling shrink clears the offsets. Returns whether any active record exists; a destination DC is not required. Original geometry and document revision remain unchanged.
Function RedrawBackofftheOpenList() As Boolean	Restores and releases saved popup backgrounds, leaving their logical open state intact. If the saved view or drawing has changed, regenerates the current drawing, grid, workspace guides and origin on the borrowed destination, then paints base controls without reopening popup pixels. The caller clip and DC state are retained. Normal control data-request events are refreshed as in R2 when AutoRedraw is enabled; recursive background redraw requests return False. Host paint callbacks are not raised by regenerating the drawing. A data callback that ends or replaces the drawing cancels completion. Host data exceptions propagate, and the recursion guard is released. A following refresh redraws the open lists. Returns False if recovery has no usable owner/destination or native drawing fails. With no open list or saved background the call is a successful no-op.
Function RefreshBehavior(ByRef eType As AvaxControlType, ByRef ParentHandle As Integer, ByRef lIndex As Integer, ByRef bVisible As Boolean, ByRef bEnable As Boolean, ByRef bLocked As Boolean) As Boolean	Updates visible/enabled/locked together for the matching identity and recalculates shrink offsets. An incorrect kind or missing/erased record returns False.
Function RefreshCaption(ByRef eType As AvaxControlType, ByRef ParentHandle As Integer, ByRef lIndex As Integer, ByRef sCaption As String) As Boolean	Updates Caption for the matching Button, Label, CheckBox or CheckOption identity. An incorrect kind or missing/erased record returns False.

Declaration	Behavior
Function RefreshControls(Optional ByRef hDC As Integer = 0) As Boolean	Refreshes data callbacks and renders the controls. Optional Integer hDC temporarily selects the destination; zero uses m_hDC. Returns False without a started owner, during recursive refresh, with AutoRedraw off, or when an explicit registry has no destination. An attached registry with zero DC refreshes its native editors. Caller clip and GDI state are preserved.
Function RefreshsList(ByRef eType As AvaxControlType, ByRef ParentHandle As Integer, ByRef lIndex As Integer, ByRef sList As String()) As Boolean	Replaces the matching ComboBox list from a one-based String array; Nothing clears it. Existing ItemData and other pattern fields are retained. Literal pipe characters and a single empty row are retained as separate list entries through native/custom editors and storage; caller arrays remain independent.
Function RefreshText(ByRef eType As AvaxControlType, ByRef ParentHandle As Integer, ByRef lIndex As Integer, ByRef sText As String) As Boolean	Updates Text for the matching TextBox or ComboBox identity. An incorrect kind or missing/erased record returns False.
Function RefreshValue(ByRef eType As AvaxControlType, ByRef ParentHandle As Integer, ByRef lIndex As Integer, ByRef dValue As Double) As Boolean	Normalizes a finite Double to zero or one for the matching CheckBox or CheckOption. Other kinds return False.
Sub RepaintControl(ByRef lRegCnt As Integer)	Refreshes and repaints the one-based registry slot. Invalid or erased slots are ignored. Other visible siblings are redrawn to retain their overlap order, without requesting their data callbacks again.
Sub SetControlTag(ByRef lRegCnt As Integer, ByRef sTag As String)	Stores the application tag. An attached record updates persistent drawing data. Invalid slots are ignored.
Sub SetControlText(ByRef lRegCnt As Integer, ByRef sText As String)	Stores Unicode Text for a recorded slot. An attached record updates its real editor and persistent drawing data. Invalid slots are ignored.
Sub SetControlToolTip(ByRef lRegCnt As Integer, ByVal sTip As String)	Normalizes line endings and stores [P] paragraph markers. Unicode and literal text are retained.
Sub SetControlValue(ByRef lRegCnt As Integer, ByRef iValue As Integer)	Stores the full Integer Value for a recorded slot. Use RefreshValue for normalized checkbox/option values.
Function SetOpenList(ByRef ListRectX As Double(), ByRef ListRectY As Double(), ByRef ParentHandle As Integer, ByRef Index As Integer) As Boolean	Stores a copied four-vertex, one-based Double polygon for an existing ComboBox parent/index. Vertices are world coordinates in top-left, top-right, bottom-right, bottom-left order. Returns False for a missing/erased/non-combo identity. Null, wrong-length or nonfinite arrays throw ArgumentException before replacing existing geometry. Does not change the host-owned selected text/value.
Sub UnEraseAvaxControl(ByRef mFillType As AvaxFill_t)	Restores only the parent/index represented by a type-9 fill.
Sub UnEraseChildAvaxControl(ByRef AvaxHandle As Integer)	Restores recorded children of a parent and raises the paired owner event once for an actual transition.
Property avIndex As Integer	Integer drawing-context token, separate from an entity handle. Zero binds lazily to the active drawing. Changing the token clears cached records and returns to automatic attachment mode.
Property MouseOnControlCnt As Integer	One-based slot of the most recent interactive hover target, or zero when none. Updated by MouseInControl; does not independently track Windows mouse messages.
bMyList As Boolean	When True, ParentHandleControlClick opens/closes helper-rendered ComboBox lists. False leaves normal native-editor list behavior to the owner. Explicit SetOpenList geometry is still queryable and paintable.
ClientX As Integer	Last viewport-pixel X supplied to PointInControl or MouseInControl; also usable as explicit input to ParentHandleControlClick.
ClientY As Integer	Last viewport-pixel Y supplied to PointInControl or MouseInControl; also usable as explicit input to ParentHandleControlClick.
CurrentAvaxControlIndex As Integer	The Index of the last completed ParentHandleControlClick. This state is published after callbacks finish and is cleared when the registry/current target is cleared. It is not a substitute for forwarding pointer input.
CurrentAvaxControlName As String	The Name of the last completed ParentHandleControlClick. This state is published after callbacks finish and is cleared when the registry/current target is cleared. It is not a substitute for forwarding pointer input.

Declaration	Behavior
CurrentAvaxControlParentHandle As Integer	The ParentHandle of the last completed ParentHandleControlClick. This state is published after callbacks finish and is cleared when the registry/current target is cleared. It is not a substitute for forwarding pointer input.
CurrentAvaxControlStatus As Integer	The Status of the last completed ParentHandleControlClick. This state is published after callbacks finish and is cleared when the registry/current target is cleared. It is not a substitute for forwarding pointer input.
CurrentAvaxControlTag As String	The Tag of the last completed ParentHandleControlClick. This state is published after callbacks finish and is cleared when the registry/current target is cleared. It is not a substitute for forwarding pointer input.
CurrentAvaxControlType As AvaxControlType	The Type of the last completed ParentHandleControlClick. This state is published after callbacks finish and is cleared when the registry/current target is cleared. It is not a substitute for forwarding pointer input.
CurrentAvaxMultilines As Integer	The last clicked control's MultiLines count, at least one. Cleared with current-control state.
lControlIndex As Integer	Integer layout counter retained for R2 callers. The R2 document renderer sets it to -1 and increments before each field, producing child indices 0, 1, and so on. Registry synchronization, InitAvaxControls, KillControls, hover and click operations preserve this caller-owned field. Use MouseOnControlCnt for a one-based hover slot and CurrentAvaxControlIndex for the last clicked child identity.
m_hDC As Integer	Legacy Integer borrowed DC field. Used only when m_hDC is zero. Prefer m_hDC or the IntPtr methods on x64. Direct field assignment does not perform the lifecycle work of the m_hDC setter.
m_RefreshControls As Boolean	Retained Boolean storage field. The original helper declares it but does not read it; it is not a rendering switch. Use AutoRedraw and explicit refresh methods.

VB.NET examples

```
Imports Avax

Public Module ControlRegistryExample
    ' Run on the drawing control's UI thread after StartAvax.
    Public Function ChangeText(cad As cAvax, parentHandle As Integer, childIndex As Integer, text As
    String) As Boolean
        ' A normal drawing API call activates that drawing for legacy helpers.
        If String.IsNullOrEmpty(cad.AvaxControlGetData(parentHandle, childIndex)) Then Return False
        Dim registry As New cAvControls()
        Try
            Dim slot = registry.ControlCntByHandle(parentHandle, childIndex)
            If slot = 0 OrElse Not registry.CanFocus(parentHandle, childIndex) Then Return False
            registry.SetControlText(slot, text)
            Return True
        Finally
            registry.KillControls()
        End Try
    End Function

    Public Function DescribeControls(cad As cAvax) As String()
        cad.AvaxControlGetData(0, 0)
        Dim registry As New cAvControls()
        Try
            Dim capacity = registry.MaxControls()
            Dim parents(capacity), indices(capacity) As Integer
            Dim names(capacity), tags(capacity), patterns(capacity) As String
            Dim count = registry.AvaxControlGetAllControls(parents, indices, names, tags, patterns)
            Dim lines(count) As String
            For i = 1 To count
                lines(i) = parents(i).ToString() & ":" & indices(i).ToString() & " " & names(i)
            Next
            Return lines
        Finally
            registry.KillControls()
        End Try
```

```
    End Function

    ' x/y are viewport pixels, excluding ruler and scrollbar margins.
    ' This query neither emits a click event nor changes the selected value.
    Public Function HitControl(cad As cAvax, x As Integer, y As Integer, ByRef parentHandle As Integer,
    ByRef childIndex As Integer) As Boolean
        cad.AvaxControlGetData(0, 0)
        Dim registry As New cAvControls()
        Try
            Dim slot = registry.PointInControl(x, y)
            If slot < 1 Then Return False
            Dim kind As AvaxControlType, status As AvaxControlStatus
            Dim name As String = Nothing, tag As String = Nothing
            Return registry.GetControl(slot, kind, parentHandle, childIndex, name, tag, status)
        Finally
            registry.KillControls()
        End Try
    End Function

    ' The caller owns this DC and must keep it valid throughout the call.
    ' Its origin is the viewport's top-left pixel. Pass the 32 significant handle bits.
    Public Function PaintControls(cad As cAvax, viewportDc As Integer) As Boolean
        cad.AvaxControlGetData(0, 0)
        Dim registry As New cAvControls()
        Try
            Return registry.RefreshControls(viewportDc)
        Finally
            registry.KillControls()
        End Try
    End Function
End Module
```

Current integration limits

- Standalone TextHandle native-editor registration is available. Exact legacy ambient owner-counter sequences and advanced mixed rendering remain under review.
- Advanced R2 widget visual styles, icon/layout/gradient/opacity combinations and exact font metrics are not fully reproduced by the current widget renderer.
- Custom popup row layout assumes the supplied top/bottom edges describe normal vertical rows; arbitrary rotated or sheared popup row picking still needs verification.
- External R2 clients and every combined widget/document/printing mode have not all been exercised.

Form backgrounds

A Form is a noninteractive background painted in the drawing, behind other embedded controls. It does not place an opaque panel over the result. StartBackColor is the centre colour and EndBackColor is the colour at the top and bottom edges; a mirrored gradient is scaled to the form's height. Equal colours produce a solid form background. Disabled forms use the legacy diagonal-cross hatch.

For compatibility, the Form pattern's Opacity value means the percentage of the existing drawing/background blended into the form colour. For example, 50 mixes equal contributions, and 100 preserves the existing drawing. Zero, or an omitted value, uses the R2 default contribution of 10 percent. Values above 100 are limited to 100. Values use Double in v3; the pattern stores them with a decimal point.

VB.NET pattern example

Use the returned string as the sPattern of a control fill of type 9. This example requests a red centre, blue edges and a 25.5 percent background contribution.

```
Option Strict On
Imports System
Imports System.Globalization

Public Module RegistryFormExample
    Public Function FormPattern(backgroundPercent As Double) As String
        If Double.IsNaN(backgroundPercent) OrElse Double.IsInfinity(backgroundPercent) OrElse
```

```
    backgroundPercent < 0 Then
            Throw New ArgumentOutOfRangeException(NameOf(backgroundPercent))
        End If
        Return "<form><StartBackColor255><EndBackColor16711680><Opacity" &
               backgroundPercent.ToString("R", CultureInfo.InvariantCulture) & ">"
    End Function
End Module
```

Button pictures and icons

Button and PictureBox patterns may supply picture, HighlightedPicture, DownPicture and DisablePicture filenames. The current status chooses the matching file; disabled behavior takes priority over hover/pressed status. An empty state filename falls back to picture. AvaxControlPictureFile receives the selected filename for both Button and PictureBox, and the returned sPictureFile is used for that state without replacing the independent normal picture.

A valid button picture fills the control's polygon and takes priority over its skin and icon; the caption is retained above the picture. If no valid picture is available, the button uses its skin and optional icon. In the helper's drawing-DC output the icon keeps its aspect ratio, has half the control height and starts at a quarter-height inset. Native buttons own image copies, select their state pictures, and resize their icons when their bounds change. Explicit refresh reloads the current image file, and reading an image does not retain a file lock.

VB.NET callback example

Pass a started control and a readable replacement image path. Register once during host setup. Keep the returned Action and invoke it when the host no longer needs this handler. The callback changes only the pressed image of buttons.

```
Option Strict On
Imports Avax

Public Module RegistryButtonImageExample
    Public Function SubstitutePressedPicture(cad As cAvax, replacementFile As String) As Action
        Dim handler As cAvax.AvaxControlPictureFileEventHandler =
            Sub(ByRef kind As AvaxControlType, ByRef parent As Integer,
                ByRef index As Integer, ByRef name As String, ByRef tag As String,
                ByRef status As AvaxControlStatus, ByRef filename As String)
                If kind = AvaxControlType.Button AndAlso status = AvaxControlStatus.Pressed Then
                    filename = replacementFile
                End If
            End Sub
        AddHandler cad.AvaxControlPictureFile, handler
        Return Sub() RemoveHandler cad.AvaxControlPictureFile, handler
    End Function
End Module
```

Button styles and caption decoration

The helper's drawing-DC output uses LayoutStyle to select the corresponding AvaxButtonStyle: Normal, Soft, Flat, Java, OfficeXP, WindowsXP, Plastik, Galaxy, Keramik or MacOSX. Status chooses the normal, highlighted, pressed or disabled appearance. UseCustomColors enables StartBackColor and HighlightColor; otherwise the style uses its normal system colours. Disabled rendering uses the disabled state. Rounded skin corners retain the drawing underneath, and the registered polygon remains the outer clip.

FontUnderline and FontStrikeOut add their respective decorations to captions. They may be combined, work with the automatic text layout or the explicit text-height axis, and follow the caption's font size, colour and rotation. A refresh with no drawing DC continues to update attached native editors according to the normal refresh rules.

VB.NET pattern example

Assign the returned pattern to the sPattern field of the AvaxFill_t passed to AddAvaxControl. This complete module builds a button pattern without changing the caller's enum value.

```
Option Strict On
Imports Avax
Imports System.Globalization

Public Module RegistryWidgetStyleExample
    Public Function ButtonPattern(style As AvaxButtonStyle) As String
        Return "<button><caption|Run><LayoutStyle" &
```

```
                CInt(style).ToString(CultureInfo.InvariantCulture) &
                "><UseCustomColors1><StartBackColor12632256>" &
                "<HighlightColor255><FontUnderline1><FontStrikeOut1>"
    End Function
End Module
```

Recording examples

Use a registry associated with a started drawing on its owning UI thread. viewportDc is a borrowed viewport DC; zero allows recording without immediate output. The caller retains ownership of that DC. Begin a pass, then record the label with a nonzero parent ID and an application child index. This example uses bottom-left world coordinates. RefreshControls paints to the supplied destination under the normal redraw rules.

InitAvaxControls example

```
Option Strict On
Imports Avax

Public Module RegistryRecordingBeginExample
    Public Sub BeginPass(registry As cAvControls, viewportDc As Integer, backColor As Integer)
        registry.InitAvaxControls(viewportDc, backColor)
    End Sub
End Module
```

AddAvaxControl example

```
Option Strict On
Imports Avax

Public Module RegistryRecordingAddExample
    Public Sub RecordLabel(registry As cAvControls, parentId As Integer, childIndex As Integer)
        Dim x As Double() = {0, 10, 110, 110, 10}
        Dim y As Double() = {0, 20, 20, 60, 60}
        Dim count As Integer = 4
        Dim fill As New AvaxFill_t With {
            .iType = 9, .lFlags = parentId, .lHatchIndex = childIndex,
            .sPattern = "<label><caption|Hello><border0>"}
        Dim shape As AvaxShapeType = AvaxShapeType.Any_s
        Dim color As Integer = 7, width As Integer = 1, style As Integer = 0
        Dim deleted As Boolean = False
        Dim textX1 As Double = 14, textY1 As Double = 56
        Dim textX2 As Double = 14, textY2 As Double = 42
        registry.AddAvaxControl(x, y, count, fill, shape, color, width, style,
                                deleted, textX1, textY1, textX2, textY2)
    End Sub
End Module
```

ConvertToShape

API reference

ConvertToShape

Drawing coordinate limits

API reference

xMin_p, yMin_p and zMin_p

Get or set the minimum X, Y or Z coordinate of the total drawing work area through GetAvaxProperty and SetAvaxProperty. Each coordinate is a Double in v3. Changing a coordinate affects only the named field; it does not move the current view, change geometry or cancel an active construction command.

xMax_p, yMax_p and zMax_p

Get or set the corresponding maximum coordinate. As in R2, individual setters may be called in either order while configuring a pair. For example, xMin_p can be assigned 100 before xMax_p is assigned 200, even if the old maximum was 20. No automatic exchange of the minimum and maximum occurs. Finish the coordinate pairs before displaying the whole work area or saving a drawing.

Validation and file behavior

Scalar coordinates must be finite. Invalid scalar input returns False without changing a coordinate. Completed work areas require strictly increasing X/Y limits with finite positive spans. Equal Z limits are valid; a minimum Z greater than its maximum is invalid. All_c leaves an unfinished or unrepresentable XY configuration and its current view unchanged. Native saving rejects an invalid work area before replacing the destination file. Use SetBoundsAvax to assign all six limits atomically; that method normalizes opposite corners and adjusts the XY rectangle to the canvas aspect ratio.

Settings XML

SaveAvaxPropertiesToXML writes all six coordinates at Double precision. LoadAvaxPropertiesFromXML validates the combined proposed limits before changing any setting, so one invalid coordinate pair rejects the entire batch. Small positive representable XY spans at a large origin are valid. Unrelated partial XML updates do not require replacing the coordinate limits.

Complete VB.NET examples

Use a started, enabled control on its owning UI thread. Coordinates that were Single in v2 are Double in v3. AvaxProp values use Integer.

Assign six independent coordinates

```
Option Strict On
Imports System
Imports Avax

Public Module Help_LimitProperties
    Public Sub SetIndividually(cad As cAvax)
        Dim keys As AvaxProp() = {AvaxProp.xMin_p, AvaxProp.yMin_p,
            AvaxProp.zMin_p, AvaxProp.xMax_p, AvaxProp.yMax_p, AvaxProp.zMax_p}
        Dim coordinates As Double() = {100, 200, 300, 400, 500, 600}
        For i As Integer = 0 To keys.Length - 1
            Dim key As AvaxProp = keys(i)
            Dim value As Object = coordinates(i)
            If Not cad.SetAvaxProperty(key, value) Then
                Throw New ArgumentException("Invalid work-area coordinate.")
            End If
        Next
    End Sub
End Module
```

Set a complete work area

```
Option Strict On
Imports System
Imports Avax

Public Module Help_LimitWindow
    Public Sub Configure(cad As cAvax)
        If Not cad.SetBoundsAvax(-1000, -1000, -100, 1000, 1000, 100) Then
```

```
            Throw New ArgumentException("Invalid work-area bounds.")
        End If
    End Sub
End Module
```

Export limit settings

```
Option Strict On
Imports System.Xml
Imports Avax

Public Module Help_LimitSettings
    Public Function Save(cad As cAvax) As XmlDocument
        Dim settings As XmlDocument = cad.SaveAvaxPropertiesToXML()
        Return settings
    End Function
End Module
```

CurrentZoomPercentage

API reference

CurrentZoomPercentage

AvaxProp Member Reference

API reference

Use SetAvaxProperty to set these values and GetAvaxProperty to read them. Changing the cursor preserves the active drawing command and its accepted points. Sizes are measured in screen pixels and use Integer values in v3.

SelectCursorSize_p

Sets the selection-box size. The default is 5. Values from 6 through 64 are accepted; other values reset to 5. Setting this property also sets CrossCircleSize_p to half the fitted box size, rounded to the nearest Integer, and applies the circle's limits. Set the circle size afterwards to choose it independently.

CrossCircleSize_p

Sets the construction crosshair's circle radius. The default is 3. Values from 3 through 32 are accepted; other values reset to 3. This property does not change the selection-box size.

While the control is started, both sizes fit within half the smaller drawing-area dimension, with a minimum of 2 pixels. The fitted size becomes the stored value. Assign the property again to enlarge it after enlarging the control. A stopped control stores the requested normalized sizes without fitting them to its window.

CrossHairStyle_p

Sets the Integer stroke style of the construction crosshair. The default is 0.

Value	Style
0	Solid
1	Dashed
2	Dotted
3	Dash-dot
4	Dash-dot-dot
5	Invisible stroke
6	Inside-frame solid stroke

CrossHairRGBColor_p

Sets the Integer RGB color of the crosshair in Windows COLORREF order: red occupies the low byte, green the next byte and blue the third byte. For example, &H0000FF is red and &H00FF00 is green. The default is 0. If the chosen color matches the drawing background, a contrasting display color keeps the cursor visible.

CrossHairOff_p

Set True to hide the construction crosshair and transient selection/zoom rectangles, and False to restore them. A pointer explicitly chosen by the host remains available. The default is False. The setting does not remove host graphics supplied through MouseDrawing_p. Touchscreen mode and PauseAvax also suppress the hovering construction crosshair.

SaveAvaxPropertiesToXML and LoadAvaxPropertiesFromXML preserve all five values. In particular, a separately assigned circle size is restored independently of the selection-box size.

VB.NET example

The form has already called StartAvax on its control.

```
Option Strict On
Imports Avax

Public Module Help_CursorProperties
    Public Function Configure(cad As cAvax) As Boolean
```

```
        Return cad.SetAvaxProperty(AvaxProp.SelectCursorSize_p, 20) AndAlso
            cad.SetAvaxProperty(AvaxProp.CrossCircleSize_p, 12) AndAlso
            cad.SetAvaxProperty(AvaxProp.CrossHairStyle_p, 1) AndAlso
            cad.SetAvaxProperty(AvaxProp.CrossHairRGBColor_p, &H00FF00) AndAlso
            cad.SetAvaxProperty(AvaxProp.CrossHairOff_p, False)
    End Function
End Module
```

Selection pointer and custom cursor

As in R2, CAD pickbox/crosshair graphics are independent of the Windows pointer code. Hiding the CAD pickbox or turning off the construction crosshair reveals the configured Windows pointer. Code 0 is the default; a named settings XML roundtrip retains this behavior. Assigning the control's Cursor directly is a host override, outside the legacy AvaxProp XML settings.

`SelectCursorMousePointer_p` uses Integer codes -1, 0 through 15, or 99. -1 suppresses the Windows pointer; CAD crosshair and pickbox visibility remain separately controlled. Code 99 loads `SelectCursorFile_p` as a .cur file, retaining its hotspot. `SelectCursorHideOn_p=True` hides the CAD pickbox while leaving that custom pointer visible. It does not turn off the construction crosshair.

The SetAvaxProperty example demonstrates the three custom-pointer settings together. An invalid active cursor filename does not replace the accepted cursor or its filename. Settings XML retains the code, Boolean and filename. See legacy cursor and refresh commands to clear transient graphics without cancelling a command.

DefaultExtension

API reference

DefaultExtension

AvaxProp Member Reference

API reference

Use SetAvaxProperty and GetAvaxProperty with these AvaxProp values. They supply the settings for the R2-compatible dimension methods and commands.

DimUnitFormat_p

Selects linear dimension units: 0 decimal, 1 engineering (feet and decimal inches), 2 architectural (feet and fractional inches), 3 scientific, 4 fractional inches. Integer; default 0.

DimAngularUnitFormat_p

Selects legacy angular dimension units: 0 decimal degrees, 1 degrees/minutes/seconds, 2 decimal gradians, 3 gradians/minutes/seconds, 4 radians. Integer; default 0.

DimFractionalTick_p

Sets the fraction denominator for architectural and fractional dimensions. Integer; default 32. Negative inputs reset to 32. Zero is retained in the property and uses an effective denominator 2 when formatting. Fractions are reduced and carry into the whole value.

DimArrowType_p

Selects dimension endpoint marks: 0 oblique ticks, 1 open arrows, 2 circles. Integer; default 0. Values outside 0..2 reset to 0 after Integer rounding.

DimDecimal_p

Sets decimal places for legacy dimension labels, normally 0..15. Integer; default 2. Fixed decimal output uses a period regardless of host culture. Fraction denominators are controlled separately by DimFractionalTick_p.

DimDefText_p

Appends a string to legacy linear dimension labels. Default empty string. Include a leading space when separating a unit suffix from the number.

AngleDefText_p

Appends a string to legacy angular dimension labels after the unit symbol. Default empty string. Unicode text is supported.

DimFactor_p

Multiplies measured lengths before formatting legacy linear and vertex dimensions. Double; default 1. Signed and zero values are accepted without changing the measured geometry.

AngleFactor_p

Multiplies measured angles before formatting legacy angular dimensions. Double; default 1. Signed and zero values are accepted without changing the angular geometry.

DimTextFont_p

Selects the font-table entry used for dimension text. Integer in v3; default 1. Use a defined font number; vertex dimension jobs require 1..256.

DimTextAbove_p

Places linear dimension text above the dimension line when True. Boolean; defaultFalse. Centered text may be placed above automatically when it cannot fit between the marks.

DimTextHorizontal_p

Keeps legacy linear dimension text horizontal when True. Boolean; defaultFalse. Otherwise it follows the readable direction of the dimension line.

DimTextVerticalToLine_p

Rotates legacy linear dimension text perpendicular to the dimension line when True. Boolean; defaultFalse. Takes precedence over DimTextHorizontal_p.

Precision and persistence

Measurement factors use Double instead of Single; font, decimal and arrow values use Integer instead of Short. Dimension properties retain their types and text in SaveAvaxPropertiesToXML/LoadAvaxPropertiesFromXML. Nonfinite numeric property input fails without changing the previous value.

Engineering and angular conversions avoid intermediate overflow for large finite Double values. An actual result outside the finite Double range is rejected. Seconds are rounded with carry into minutes and whole degrees or gradians. Vertex dimension labels trim trailing spaces from their suffix, matching the R2 vertex job.

Native dimension entities use their own options. Get_DimensionDefaults copies DimFactor_p, AngleFactor_p and all five angular unit modes, including gradians/minutes/seconds. Both measurement factors accept signed and zero finite values. Native precision is limited to 0..8 and fractional precision to a power-of-two denominator. See native measurement formatting for editable styles, storage and CAD exchange.

VB.NET example

Call Configure with an initialized control, then call Add_Dim. A length of 14.03125 is shown as 1 foot, 2 and 1/32 inches followed by the suffix.

```
Option Strict On
Imports Avax

Public Module Help_DimensionProperties
    Public Function Configure(cad As cAvax) As Boolean
        Return cad.SetAvaxProperty(AvaxProp.DimUnitFormat_p, 2) AndAlso
            cad.SetAvaxProperty(AvaxProp.DimFractionalTick_p, 32) AndAlso
            cad.SetAvaxProperty(AvaxProp.DimDecimal_p, 2) AndAlso
            cad.SetAvaxProperty(AvaxProp.DimFactor_p, 1.0) AndAlso
            cad.SetAvaxProperty(AvaxProp.DimDefText_p, " length") AndAlso
            cad.SetAvaxProperty(AvaxProp.DimArrowType_p, 1) AndAlso
            cad.SetAvaxProperty(AvaxProp.DimTextAbove_p, True)
    End Function
End Module
```

Display properties

API reference

These properties configure the appearance and input state of an AVAX control. Use ScrollBars to display drawing scroll bars. Each example accepts a control that your form has already initialized with StartAvax.

RulerX

RulerY

OriginVisible

BackColor

ForeColor

BorderStyle

Enabled

GetDocumentClearText

API reference

GetDocumentClearText

Document font height and wrapping width

API reference

These methods change the layout of documents created by Add_Document and Add_ShapeDocument. Dimensions are in drawing units. A document font height is a drawing height, not a Windows font size in points. Width is the logical page width used for wrapping; occupied text and image extents can be different.

ChangeDocumentFontSize

ChangeDocumentWidth

ChangeShapeDocumentFontSize

ChangeShapeDocumentWidth

GetDocumentFontSize

GetDocumentWidth

GetShapeDocumentText

Dimension sizes and wall thickness

API reference

Read and write these AvaxProp settings with GetAvaxProperty and SetAvaxProperty. Their values are Double in v3, replacing the Single values in v2. Use the control on its owning UI thread after StartAvax. Property assignment returns Boolean success; GetAvaxProperty returns an Object containing the Double value.

DimScale_p

Overall dimension component scale. Default 1. It multiplies dimension text height, endpoint marks and extension lengths once. It does not multiply the measured geometry, the displayed measurement factor or the offset from the source object. A dimension requires a positive scale and representable resulting geometry.

DimDist_p

Initial offset from the dimensioned object, in drawing units. Default 1. Vertex dimension commands use it when no explicit offset is supplied. An Add_Dim OffsetD argument or a placement pick supplies its own geometric distance. This distance is independent of DimScale_p.

DimArrowSize_p

Angular endpoint-mark size and linear centered-text gap allowance. Default 0.2. Angular oblique ticks have total length DimArrowSize_p * DimScale_p; each open-arrow leg has half that length. Circle marks have diameter DimArrowSize_p * DimScale_p. The configured size is retained even when larger than the angular arc radius. V3 corrects the old circle-mark omission of DimScale_p. A nonpositive setting disables angular marks and the additional linear gap allowance.

DimTickSize_p

Size of linear dimension endpoint marks. Default 0.1. DimArrowType_p chooses oblique ticks, open arrows or circles. The effective size is DimTickSize_p * DimScale_p: an oblique tick extends that distance to each side of its center, an open-arrow leg has that length, and a circle has that diameter. A nonpositive setting disables these marks.

DimOutExtLines_p

Extension-line length beyond the dimension line, away from the source object. Default 0.1. The effective length is DimOutExtLines_p * DimScale_p. It is measured from the dimension line, independently of the geometric offset. A nonpositive setting disables this portion.

DimInExtLines_p

Extension-line length from the dimension line towards the source object. Default 0.2. The effective length is DimInExtLines_p * DimScale_p. This is a length from the dimension line, rather than a gap from the source endpoint. A nonpositive setting disables this portion.

DimTextScale_p

Dimension text height before overall scaling. Default 1. The resulting height is DimTextScale_p * DimScale_p in drawing units. The height and its stored text footprint must remain positive and representable. Dimension text uses DimTextFont_p and the current dimension formatting settings.

WallWidth_p

Predetermined wall thickness for Wall_c. Default 0.2 drawing units. Assigning zero, including a zero read from settings XML, restores 0.2. Other finite values are retained with full Double precision. Use a positive value for a wall thickness. A negative stored value makes the Wall options start at 1; with dialogs disabled, that positive initial value is used.

Wall_c asks for a width when DialogEnabled_p=True, then accepts two baseline points and a third point choosing the side. With dialogs disabled it uses the current positive width. Wall3p_c derives its width from the perpendicular distance to the third point and does not use WallWidth_p. Both commands create one closed polyline and one Undo/Redo operation.

Validation and settings XML

Finite signed dimension settings retain their supplied values. Construction validates their effective use: zero or negative text/overall scale is rejected, while nonpositive mark and extension sizes disable their components. NaN and infinities are rejected during assignment without replacing the previous value. SaveAvaxPropertiesToXML and LoadAvaxPropertiesFromXML preserve the complete Double values and apply the same zero-to-default wall rule as direct assignment.

Dimension preparation rejects overflowing, collapsed or unrepresentable output before inserting any part of the result. See dimension formatting for measurement factors, units and text options, and Triangle and angle commands for the selected-line input workflow.

VB.NET example: dimension sizes

The supplied control must already be started. This example configures the component sizes and creates a linear dimension using the configured unscaled offset.

```
Option Strict On
Imports Avax

Public Module Help_DimensionSizes
    Public Function Draw(cad As cAvax) As Boolean
        If Not cad.SetAvaxProperty(AvaxProp.DimScale, 2.0R) Then Return False
        If Not cad.SetAvaxProperty(AvaxProp.DimDist, 3.25R) Then Return False
        If Not cad.SetAvaxProperty(AvaxProp.DimTextScale, 0.75R) Then Return False
        If Not cad.SetAvaxProperty(AvaxProp.DimArrowSize, 0.5R) Then Return False
        If Not cad.SetAvaxProperty(AvaxProp.DimTickSize, 0.25R) Then Return False
        If Not cad.SetAvaxProperty(AvaxProp.DimInExtLines, 0.75R) Then Return False
        If Not cad.SetAvaxProperty(AvaxProp.DimOutExtLines, 0.25R) Then Return False
        cad.SetAvaxProperty(AvaxProp.DimTextAbove, True)
        Dim x1 As Double = 0, y1 As Double = 0
        Dim x2 As Double = 100, y2 As Double = 0
        Dim offset As Double = CDbl(cad.GetAvaxProperty(AvaxProp.DimDist))
        Dim sideX As Double = 50, sideY As Double = 10
        Return cad.Add_Dim(x1, y1, x2, y2, offset, sideX, sideY)
    End Function
End Module
```

VB.NET example: restore the default wall thickness

The supplied control must already be started. Setting the width to zero restores 0.2; it does not create a zero-width wall.

```
Option Strict On
Imports Avax

Public Module Help_DefaultWallThickness
    Public Function Draw(cad As cAvax) As Boolean
        cad.Command = AvaxCommand.Cancel
        cad.SetAvaxProperty(AvaxProp.DialogEnabled, False)
        If Not cad.SetAvaxProperty(AvaxProp.WallWidth, 0.0R) Then Return False
        cad.Command = AvaxCommand.Wall
        Dim x As Double = 0, y As Double = 0, z As Double = 0
        If Not cad.SetCoordinate(x, y, z) Then Return False
        x = 10 : y = 0
        If Not cad.SetCoordinate(x, y, z) Then Return False
        x = 5 : y = 1
        Return cad.SetCoordinate(x, y, z)
    End Function
End Module
```

cAvax Member Reference

API reference

LastOperationError

ExportNativeDrawing

PrintWorkspaceToImage

PrintToAPL

Drawing modes and settings

API reference

Use Command with an AvaxCommand value on a started, enabled control. The following modes retain the active drawing command and its accepted points. Unstarted, disabled, ended and disposed controls ignore these commands. Changing these modes does not create drawing objects or geometry undo records.

AvaxCommand values

Command	Purpose and behavior
`Grid_c`	Configure independent X/Y grid spacing and enable the grid when accepted.
`Grid_On_c`	Enable valid stored grid spacing without a dialog; unusable spacing requests configuration.
`Grid_Off_c`	Disable grid display; repeated calls leave it disabled.
`SetGrid_c`	Disable an enabled grid without a dialog, or request grid configuration when disabled.
`Snap_c`	Configure independent X/Y snap intervals and enable snapping when accepted.
`SetSnap_c`	Disable enabled snapping without a dialog, or request snap configuration when disabled.
`Ortho_c`	Toggle horizontal/vertical construction about the previously accepted point.
`Ortho_On_c`	Enable orthogonal construction.
`Ortho_Off_c`	Disable orthogonal construction.
`Plane_c`	Toggle the legacy plane flag for subsequently created drawing objects.
`Plane_On_c`	Enable the legacy plane flag for new objects.
`Plane_Off_c`	Disable the legacy plane flag for new objects.
`Fasttext_c`	Toggle outline presentation of screen text.
`FastText_On_c`	Display text outlines instead of screen glyphs.
`FastText_Off_c`	Restore normal screen text presentation.
`NoText_On_c`	Hide text on screen without removing text objects.
`NoText_Off_c`	Show text on screen again.
`NoWidth_On_c`	Use thin screen lines while retaining stored widths and printed widths.
`NoWidth_Off_c`	Restore the normal screen line widths.
`ZoomUndo_On_c`	Record subsequent view changes in normal Undo/Redo as Zoom_u.
`ZoomUndo_Off_c`	Stop adding view records to Undo/Redo. Existing records and Previous_c history remain available.

Grid_c and Snap_c configure distances. SetGrid_c and SetSnap_c disable an enabled mode or request configuration when disabled, matching the executable R2 control. DialogEnabled_p=False cancels configuration without changing settings. See Grid and coordinate snap commands for exact names, cancellation, direct actions and complete examples. Rejected, cancelled or obsolete input is not applied to a changed drawing or construction command. Grid display and snap intervals remain independent, including at dense zoom levels.

AvaxProp values

Parameter	v3 value type	Purpose
GridOn_p	Boolean	Show or hide the grid independently of coordinate snapping.
GridStepX_p	Double	Horizontal grid spacing in drawing units.
GridStepY_p	Double	Vertical grid spacing in drawing units.
GridLinesOn_p	Boolean	True draws grid lines; False draws grid dots.
SnapOn_p	Boolean	Enable snapping to SnapStepX_p and SnapStepY_p.
SnapStepX_p	Double	Horizontal coordinate snap interval in drawing units.
SnapStepY_p	Double	Vertical coordinate snap interval in drawing units.
Elevation3D_p	Double	Finite elevation used by new plane objects; negative elevations are permitted.
Height3D_p	Double	Finite height of new plane objects.
ZoomMemoryOn_p	Boolean	Allow cached raster images for interactive zoom rendering; independent of view-history recording.
BackBitmapDrawingOn_p	Boolean	Allow the bitmap-backed drawing optimization. Zoom image caching also requires ZoomMemoryOn_p.

Distances, coordinates, elevation and height use Double in v3. Command/parameter enum values and status bit masks use Integer. Use positive finite grid and snap distances for an active lattice. Smaller display grids may be thinned to readable screen spacing; this does not change the numeric snap interval.

ZoomMemoryOn_p controls adaptive raster caching for eligible large drawings. Disabling it uses vector rendering and does not erase Previous_c history or change ZoomUndo_On_c/ZoomUndo_Off_c. Cache images are retained in memory; this setting does not create the Zoom*.bmp temporary files used by R2.

Complete VB.NET examples

Pass a control already initialized with StartAvax, and run these procedures on its owning UI thread.

Programmatic grid and snap settings

```
Option Strict On
Imports System
Imports Avax

Public Module Help_GridSnapSettings
    Public Sub Configure(cad As cAvax, gridX As Double, gridY As Double,
                         snapX As Double, snapY As Double)
        For Each value As Double In New Double() {gridX, gridY, snapX, snapY}
            If Double.IsNaN(value) OrElse Double.IsInfinity(value) OrElse value <= 0 Then
                Throw New ArgumentOutOfRangeException("spacing", "Spacing must be positive and finite.")
            End If
        Next
        cad.SetAvaxProperty(AvaxProp.GridStepX_p, (gridX))
        cad.SetAvaxProperty(AvaxProp.GridStepY_p, (gridY))
        cad.Command = AvaxCommand.Grid_On_c
        If Not cad.SnapToStepSet(snapX, snapY) Then
            Throw New InvalidOperationException("Could not enable coordinate snapping.")
        End If
    End Sub
End Module
```

Change a mode during line construction

```
Option Strict On
Imports Avax

Public Module Help_OrthoLine
    Public Function Draw(cad As cAvax) As Boolean
```

```
        cad.Command = AvaxCommand.Line.c
        If Not cad.SetCoordinate(1.25, 2.5, 0) Then Return False
        cad.Command = AvaxCommand.Ortho_On.c
        Return cad.SetCoordinate(21.25, 6.5, 0)
    End Function
End Module
```

Independent zoom image memory and history

```
Option Strict On
Imports Avax

Public Module Help_ZoomImageMemory
    Public Sub Configure(cad As cAvax, cacheImages As Boolean, recordViews As Boolean)
        cad.SetAvaxProperty(AvaxProp.ZoomMemoryOn.n, (cacheImages))
        cad.Command = If(recordViews, AvaxCommand.ZoomUndo_On.c, AvaxCommand.ZoomUndo_Off.c)
    End Sub
End Module
```

Editable shape methods

API reference

These v3 methods use Double coordinates and dimensions, and Integer handles and counts. The native outline remains authoritative: arbitrary node edits can turn a parameterized object into a freely edited path.

AddEditableShape

GetShapeParameters

ChangeShapeParameter

GetShapeClosed

ChangeShapeClosed

SetEditablePointSelectionPreview

Using event handlers

API reference

Control events let a host react to drawing changes, selection, mouse and keyboard input, view changes and requests for application-supplied geometry. Each event reference describes its trigger and arguments and links to a complete form example.

Choose the exact handler signature

Not every event uses the standard Sender/EventArgs pair. For example, Change has no arguments. Use the declaration in the event topic instead of assuming a Windows Forms event signature.

```
Private Sub cAvax1_Change() Handles cAvax1.Change
    Me.Text = "Drawing changed"
End Sub
```

A WithEvents field supports a Handles clause. If you create the control dynamically, attach and detach a matching procedure with AddHandler and RemoveHandler.

Input and output event data

Events with argument objects expose their data as fields. In v3, drawing coordinates use Double and former Short fields generally use Integer. Some fields report input; others allow the handler to cancel an action or supply a result. Only change fields documented as writable outputs.

```
Private Sub cAvax1_BeforeClick(sender As Object,
                               e As Avax.cAvax.BeforeClickEventArgs) Handles cAvax1.BeforeClick
    If preventDrawingClick Then
        e.fCancel = True
    End If
End Sub
```

In this fragment, `preventDrawingClick` is a Boolean field owned by the host application. It illustrates a cancellation decision; declare it in the form before using the fragment.

Application-supplied external lines

Add_ExternalLines stores a named reference. The corresponding SetExternalLines event asks the host for coordinate arrays. Supply Double arrays using the indexing required by that event. See its example for reading the original map data without changing the stored Single file layout.

Thread and lifetime

Use the control and update its form on the owning UI thread. Keep event handlers brief, and avoid making a drawing change from Change unless your code guards against re-entering the same handler. PauseAvaxEvents controls the delivery of control events; restore it after a temporary suppression period.

Grid and coordinate snap commands

API reference

Grid display and coordinate snapping are independent. Grid spacing controls the visible lattice. Snap spacing constrains accepted construction coordinates. The commands below preserve the current drawing command, accepted XYZ coordinates and object selection.

Command	Value	Purpose
Grid_c	87	Configure independent X/Y grid spacing. Accepting positive finite distances enables grid display. Calling it again opens configuration again.
Grid_On_c	88	Enable the existing valid grid without a dialog. If its stored spacing is unusable, request configuration instead. Repeated calls retain the enabled state.
Grid_Off_c	89	Disable grid display without changing its distances or coordinate snapping.
Snap_c	90	Configure independent X/Y snap spacing. Accepting positive finite distances enables coordinate snapping. Repeated calls configure rather than disable it.
SetGrid_c	106	If the grid is enabled, disable it without a dialog. Otherwise request grid configuration, even when stored distances are valid.
SetSnap_c	107	If snapping is enabled, disable it without a dialog. Otherwise request snap configuration.

These names and values follow the executable v2 control and its toolbar. The original v2 command table described Grid_c and SetGrid_c/SetSnap_c differently; use the table above when porting calls.

Configuration and cancellation

Configuration requires DialogEnabled_p=True. When dialogs are disabled, configuration requests do nothing; they do not silently accept or repair current distances. Cancelling or entering invalid values preserves both distances and enabled state. Validation checks each Double distance independently, so very small or very large finite positive distances remain valid. It does not multiply the two distances to decide validity.

A dialog answer is discarded when the application changes the drawing, command, accepted points, selection, pause/edit-lock state or grid/snap settings before the answer is applied. A newer configuration, disable command, property assignment or SnapToStepSet call takes precedence. Exceptions raised by application callbacks propagate to the caller.

Programmatic settings

To configure without a dialog, assign GridStepX_p and GridStepY_p and issue Grid_On_c. Call SnapToStepSet with Double variables to configure snapping. SnapToStepSet accepts zero to disable snapping while storing the requested intervals; it rejects negative or non-finite values. You may also set SnapOn_p and the two SnapStep properties explicitly.

Direct actions

DoAvaxAction accepts all six commands above. Supply a one-based Integer handle array containing at least one existing object, even for a global setting. The method rejects invalid handles before changing settings or opening a dialog. vIn is unused. Successful dispatch returns the unique supplied handles in OutHandles, fNewHandles=False, vOut=Nothing and errCode=0. Cancellation and suppressed dialogs are successful dispatches with unchanged settings; query GridOn_p or SnapToStepStatus when you need the resulting state. A stopped, disabled or disposed control returns False with errCode=1; invalid handles return errCode=2. These modes may be used while input is paused or editing is locked and create no geometry Undo entry.

Direct actions follow the current grid/snap settings and the same dialog rules as Command. They preserve the active drawing command and accepted construction points.

Complete VB.NET examples

Pass a started control and call these procedures on its owning UI thread. Coordinates and distances are Double; handles, command values and error codes are Integer.

Configure independent intervals without a dialog

```
Option Strict On
Imports Avax
```

```
Public Module Help_GridSnapIntervals
    Public Sub Configure(cad As cAvax, gridX As Double, gridY As Double,
                         snapX As Double, snapY As Double)
        For Each distance As Double In New Double() {gridX, gridY, snapX, snapY}
            If Double.IsNaN(distance) OrElse Double.IsInfinity(distance) OrElse distance <= 0 Then
                Throw New System.ArgumentOutOfRangeException("distance")
            End If
        Next
        cad.SetAvaxProperty(AvaxProp.GridStepX, n, (gridX))
        cad.SetAvaxProperty(AvaxProp.GridStepY, n, (gridY))
        cad.Command = AvaxCommand.Grid_On_c
        If Not cad.SnapToStepSet(snapX, snapY) Then
            Throw New System.InvalidOperationException("Could not set snap intervals.")
        End If
    End Sub
End Module
```

Toggle the modes using the R2 executable command names

```
Option Strict On
Imports Avax

Public Module Help_GridSnapToggle
    Public Sub Grid(cad As cAvax)
        ' Enabling requests configuration; DialogEnabled_n must be True.
        cad.Command = AvaxCommand.SetGrid_c
    End Sub

    Public Sub Snap(cad As cAvax)
        cad.Command = AvaxCommand.SetSnap_c
    End Sub
End Module
```

Dispatch a mode against a valid one-based handle array

```
Option Strict On
Imports Avax

Public Module Help_GridSnapDirect
    Public Function Apply(cad As cAvax, entityHandles As Integer(), action As AvaxCommand) As Integer()
        Select Case action
            Case AvaxCommand.Grid_c, AvaxCommand.Grid_On_c, AvaxCommand.Grid_Off_c,
                 AvaxCommand.SetGrid_c, AvaxCommand.Snap_c, AvaxCommand.SetSnap_c
            Case Else
                Throw New System.ArgumentOutOfRangeException("action")
        End Select
        Dim input As Object = Nothing, extra As Object = Nothing
        Dim created As Boolean, output As Integer() = Nothing, errorCode As Integer
        If Not cad.DoAvaxAction(entityHandles, input, action, created, output, extra, errorCode) Then
            Throw New System.InvalidOperationException("Mode could not be dispatched. Error: " &
    errorCode.ToString())
        End If
        Return output
    End Function
End Module
```

Hatch command

API reference

Hatch_c

Command value 511 creates parallel hatch lines inside a boundary, or outside it within its bounding rectangle. It adds independent Line_i objects at Z=0 and leaves the source objects intact. All generated lines form one Undo/Redo operation. Coordinates, spacing and angles use Double; handles, command values and error codes use Integer. Call the control on its owning UI thread after StartAvax.

Boundary input

Ordinary Command use first reads any nonempty temporary vertex sequence entered with SelectV_c or restored with LastSv_c. Otherwise it extracts the selected paths as one connected XYZ boundary. Shared endpoints must match exactly; disconnected or branching selections are rejected. Lines, polylines, shapes, curves and sampled ellipses use the selected vertex boundary rules. A boundary needs three finite vertices before an optional exact closing marker. Its final edge is implicit. Temporary input takes priority even when it has too few vertices; selected objects are not a fallback.

Z is retained in the input and LastSv backup but does not affect hatch geometry. Crossing boundaries use even-odd interiors. Coincident boundary intervals are included without duplicating fill segments. A collinear guide has no filled interior. A zero-width or zero-height bounding rectangle produces no lines. These rules avoid the old tolerance-based loss of small edges and inconsistent tangent intersections.

Settings

Property	Type and purpose
HatchDistance_p	Double. Perpendicular distance between rows, default 0.25. A finite zero or negative property assignment resets to 0.25, also when loaded from settings XML. Nonfinite assignments are rejected without replacing the prior value. Explicit Add_LinearHatch Distance arguments and accepted dialog distances must be positive.
HatchAngle_p	Double. Finite angle in degrees; equivalent angles repeat every 180 degrees.
HatchInside_p	Boolean. True fills the boundary; False creates its complement within the original bounding rectangle.
HatchBorder_p	Boolean. Adds the bounding rectangle before the hatch rows.
HatchMask_p	String. Empty selects every row. Otherwise only 0 and 1 are accepted; the pattern repeats over candidate rows. 100 selects every third row beginning with the first.
DialogEnabled_p	Boolean. Ordinary Hatch displays its options when True. When False, v3 executes with current settings.

The dialog also offers Boundary and Both borders for that invocation. HatchBorder_p remains Boolean and stores only the bounding-rectangle part of the choice. Accepted settings are stored after complete geometry and capacity preparation. Cancelling or rejecting settings preserves the existing drawing, accepted input and Redo history. Newer settings supplied by a host event take precedence.

Line order and precision

Rows are anchored to the boundary's axis-aligned bounding rectangle. A row exactly on the final boundary is included. Horizontal rows run left to right; vertical rows are ordered from left to right and each runs bottom to top. The mask counts candidate rows from the same origin even if a row produces no interior segment.

When requested, the bounding border is emitted first: left upwards, top rightwards, right downwards, bottom leftwards. A dialog boundary border then follows the input path. Fill segments follow in row order. Zero-length segments are omitted. An all-zero mask suppresses both borders and fill. Valid empty output leaves ordinary input intact; direct use returns False with error code 2.

Current pen color, width, line style, layer, elevation and height apply to each new line. Geometry coordinates have Z=0 independently of that metadata. Exact scan predicates preserve small finite features; a result that cannot be represented by distinct finite Double coordinates, or exceeds output capacity, is rejected as a whole.

Input and events

After successful preparation, ordinary Hatch clears the temporary vertices and all selected objects, including an unrelated selection when temporary vertices supplied the boundary. LastSv_c restores the exact captured XYZ sequence. Results are not selected by ordinary Command use. DoAvaxAction selects its newly created lines.

A pending Polyline is completed only after the hatch can first be prepared. Its creation remains a separate Undo operation. The boundary is revalidated after its events. NewUserItems and Change observe a complete hatch batch. If a source-clear callback changes the command, selection, document or edit state, the older hatch request is abandoned. Host exceptions retain their original identity.

DoAvaxAction

Pass Hatch_c as myAction, a one-based InHandles array and vIn=Nothing. The explicit handles supply the boundary, regardless of temporary input or the current selection. Repeated handles are used once. This route reads the five Hatch properties and does not open a dialog. No Object-array or numeric-array vIn format is supported.

Success returns True, fNewHandles=True, ordered one-based OutHandles, vOut=Nothing and errCode=0. Failure returns False with fNewHandles=False, OutHandles={0} and vOut=Nothing. Error 2 means invalid input, settings, empty output or preparation failure; error 1 means lifecycle or input-ownership rejection. Direct creation is allowed while paused when no pending Polyline needs completion. Edit lock, disabled or ended controls reject it.

VB.NET example: explicit boundary

The control must already be started. Supply the handles of one connected boundary.

```
Option Strict On
Imports Avax
Imports System

Public Module Help_HatchDirect
    Public Function Create(cad As cAvax, boundaryHandles As Integer(), distance As Double) As Integer()
        Dim value As Object = distance
        cad.SetAvaxProperty(AvaxProp.HatchDistance_n, value)
        value = 0.0R : cad.SetAvaxProperty(AvaxProp.HatchAngle_n, value)
        value = True : cad.SetAvaxProperty(AvaxProp.HatchInside_n, value)
        value = False : cad.SetAvaxProperty(AvaxProp.HatchBorder_n, value)
        value = "" : cad.SetAvaxProperty(AvaxProp.HatchMask_n, value)
        Dim action As AvaxCommand = AvaxCommand.Hatch_c
        Dim input As Object = Nothing, extra As Object = Nothing
        Dim created As Integer() = Nothing, newHandles As Boolean, errorCode As Integer
        If Not cad.DoAvaxAction(boundaryHandles, input, action, newHandles, created, extra, errorCode)
    Then
            Throw New InvalidOperationException("Hatch failed. Error code: " & errorCode.ToString())
        End If
        Return created
    End Function
End Module
```

VB.NET example: ordinary command

This example explicitly cancels the previous input before selecting its boundary. Existing hatch settings are used without a dialog.

```
Option Strict On
Imports Avax

Public Module Help_HatchCommand
    Public Sub Execute(cad As cAvax, boundaryHandles As Integer())
        cad.Command = AvaxCommand.Cancel_c
        cad.SelectItems(boundaryHandles)
        Dim dialogs As Object = False
        cad.SetAvaxProperty(AvaxProp.DialogEnabled_n, dialogs)
        cad.Command = AvaxCommand.Hatch_c
    End Sub
End Module
```

Related operations

Add_LinearHatch returns one compound/polyline object at the first input Z. It uses the same inclusive boundary rows, bottom-to-top vertical segments, bounding-rectangle border prefix and cyclic mask phase described above. A mask containing only zeros suppresses the border as well as the fill. A single generated segment keeps both of its distinct endpoints. Native hatch methods create editable hatch objects. Those storage and input contracts are separate from Hatch_c. The low-level legacy vertex-job file protocol also has its own settings format.

Interactive Command-based creation publishes NewUserItems as described here. A direct DoAvaxAction call is an automatic operation: use OutHandles and fNewHandles for its result; it publishes Change without NewUserItems. Previously accepted user construction completed before that action retains its own notification.

cAvax Member Reference

API reference

Image effects adjust the displayed appearance of a still bitmap while retaining its stored source image. A list contains up to 32 effects, applied in array order. Effects, effect descriptions and returned image bytes are ordinary detached .NET values. The examples accept an existing drawing control and image handle; editing examples require a started, enabled, editable control.

GetImageEffectDefinitions

IsImageShape

GetImageSource

GetImageEffects

SetImageEffects

ResetImageEffects

RenderImagePreview

Effect value objects

Construct an effect with `New Avax.AvaxImageEffect(kind, amount)` or its parameterless constructor and writable properties. Construction only creates a value; validation occurs when applying or previewing the list. The parameterless defaults are Kind = Brightness, Amount = 0 and Enabled = True.

AvaxImageEffect value	Type	Meaning
Kind	AvaxImageEffectKind	Effect to apply. The enum values and supported amounts are listed below.
Amount	Double	Finite amount in the selected effect's units. Fractional values are accepted except for Pixelate and Posterize.
Enabled	Boolean	True applies the entry; False retains it in the list without applying it. Default True.

`Clone()` returns an independent AvaxImageEffect with the same Kind, Amount and Enabled values. `ToString()` returns a display label containing the kind and amount in the current culture, prefixed by `[Off]` when disabled. Neither method edits an image.

AvaxImageEffectInfo value	Type	Meaning
Kind	AvaxImageEffectKind	Identifier accepted by AvaxImageEffect.Kind.
Minimum, Maximum	Double	Inclusive accepted Amount range.
DefaultValue	Double	Suggested initial amount when adding this kind to an editor.
Increment	Double	Suggested editor step. It does not round an Amount automatically.
IsInteger	Boolean	True for Pixelate and Posterize, whose amounts must be whole numbers.

These descriptions have a parameterless constructor and writable scalar properties. Use GetImageEffectDefinitions for populated catalog values. Creating a description does not register a new effect.

Effect descriptions and amount ranges

The table lists every AvaxImageEffectKind value. RGB adjustments clamp their output to the channel range. Colour adjustments preserve pixel alpha; Blur, Pixelate and Crop account for alpha when combining pixels. Amounts operate on the result of preceding enabled entries, so changing list order can change the result.

Kind (enum value)	Amount range and units	Default	Increment	Effect
Brightness (0)	-100 to 100, percentage adjustment	0	1	Adds Amount multiplied by 2.55 to each RGB channel. Positive values brighten; negative values darken.

Kind (enum value)	Amount range and units	Default	Increment	Effect
Contrast (1)	-100 to 100, percentage adjustment	0	1	Scales channel distance from the midpoint 127.5 by 1 + Amount / 100. Zero is unchanged; -100 produces the midpoint colour.
Exposure (2)	-5 to 5, exposure stops	0	0.1	Multiplies RGB channels by 2 raised to Amount. One stop doubles channel values before clamping.
Gamma (3)	0.1 to 5, gamma exponent parameter	1	0.1	Applies output = 255 multiplied by (input / 255) raised to 1 / Amount. One is unchanged; values above one lighten midtones.
Saturation (4)	-100 to 100, percentage adjustment	0	1	Scales colour distance from luminance by 1 + Amount / 100. At -100 the result is grayscale; positive values increase saturation.
Hue (5)	-180 to 180, degrees	0	1	Rotates hue using an RGB colour-rotation matrix. Zero is unchanged.
Temperature (6)	-100 to 100, relative colour adjustment	0	1	Positive values warm the image by adding red and reducing blue; negative values cool it.
Tint (7)	-100 to 100, relative colour adjustment	0	1	Positive values shift toward magenta by increasing red/blue and reducing green. Negative values shift toward green.
Grayscale (8)	0 to 100, blend percent	100	1	Blends RGB toward luminance. Zero is unchanged; 100 is fully grayscale.
Sepia (9)	0 to 100, blend percent	100	1	Blends the image with a sepia colour conversion. Zero is unchanged; 100 applies the full conversion.
Invert (10)	0 to 100, blend percent	100	1	Blends each RGB channel with 255 minus that channel. At 100 colours are fully inverted.
Blur (11)	0 to 50, pixel-based blur strength	2	0.1	Softens the image with repeated box-blur passes. Zero is unchanged. The pixel-based strength is scaled down for a reduced-resolution preview.
Sharpen (12)	0 to 300, sharpening strength percent	50	1	Adds a scaled difference between the current image and a small blurred image. Zero is unchanged; 100 adds the full difference.
Pixelate (13)	1 to 128, whole source-pixel block size	8	1	Replaces square blocks with their averaged colour. Block size scales with preview resolution.
Posterize (14)	2 to 32, whole levels per RGB channel	6	1	Quantizes each RGB channel to the requested number of equally spaced levels.
Vignette (15)	0 to 100, darkening strength percent	50	1	Darkens the image toward its corners. Zero is unchanged.
Kind (enum value)	**Amount range and units**	**Default**	**Increment**	**Effect**
Crop (16)	0 to 40, inset percent on each side	0	1	Resamples the centred inner rectangle into the same pixel canvas. Ten removes 10% at the left, right, top and bottom, then fills the unchanged canvas with the remaining 80% width

Kind (enum value)	Amount range and units	Default	Increment	Effect
				and height. Stored source bytes and drawing geometry are retained.

RenderImagePreviewAsync

API reference

RenderImagePreviewAsync

Item drawing order

API reference

These v3 methods use Integer handles and order values. The ShapeZorderOn_p display setting controls whether stored order or ordinary handle order is presented. Changing stored order does not silently enable that setting.

GetItemZOrder

ChangeItemOrder

ItemType and GetAvaxItemType

API reference

Use these methods to identify the kind of an object before calling a type-specific geometry or editing method. Both return an AvaxItemType value from the current control's drawing.

ItemType

GetAvaxItemType

Return value and behavior

The return value is the stored object type, such as Line_i, Polyline_i, Text_i, Document_i or Curve_i. Any_i (zero) means that no object is available for the requested handle. Zero, negative, missing and deleted handles return Any_i. A control that has not started, has ended, is disabled or is disposed also returns Any_i.

These methods do not select or change the object. They do not create an Undo entry or change event. Explicit handle queries can read objects on hidden or locked layers; LockEdit and PauseAvax do not prevent reading. A handle from another control is not a cross-control reference: the number is always resolved in the receiving control.

Both handle parameters remain Integer in v3. Do not infer a type from an object's handle or from a method name: for example, Add_BlockLines can create a Polyline_i. Use LastHandle to obtain the latest allocated handle, then check its type before using it.

VB.NET example

Pass an already started control and an object handle to this function. It demonstrates both calling forms and returns a description without modifying the drawing.

```
Option Strict On
Imports System

Public Module TypeQueryExample
    Public Function DescribeItem(cad As Avax.cAvax, handle As Integer) As String
        If cad Is Nothing Then Throw New ArgumentNullException(NameOf(cad))
        Dim byValue As Avax.AvaxItemType = cad.ItemType(handle)
        Dim inputHandle As Integer = handle
        Dim byReference As Avax.AvaxItemType = cad.GetAvaxItemType(inputHandle)
        If inputHandle <> handle OrElse byValue <> byReference Then
            Throw New InvalidOperationException("The drawing changed during the query.")
        End If
        If byValue = Avax.AvaxItemType.Any_i Then Return "Object not available."
        Return byValue.ToString()
    End Function
End Module
```

For example, after `Dim h As Integer = cad.Add_Line(0, 0, 0, 10, 10, 0)`, `TypeQueryExample.DescribeItem(cad, h)` returns `Line_i`.

LastImportWarnings

API reference

LastImportWarnings

DoAvaxAction and public value structures

API reference

Direct action parameters

Executes a supplied action against a one-based Integer handle array. Index zero is reserved. vIn carries action-specific parameters; fNewHandles reports whether replacement or copied objects were created, OutHandles identifies the resulting targets, vOut carries an additional result, and errCode is zero on success. A rejected request returns False with a nonzero error code. Host event exceptions propagate to the caller.

Move_c and Copy_c take AvaxMove_t; Rotate_c takes AvaxRotate_t; Mirror_c takes AvaxMirror_t; Scale_c takes AvaxScale_t. The Transform_m value uses a Double(3,3) matrix with logical indices 1 through 3. Block_c takes a one-based String array containing a name, optionally followed by base X and Y. The immediate action reference describes direct drafting/display modes, filtered selection and the combined Border_c action. Other implemented actions include selection with None_c, Erase_c, Close_c, Explode_c, Explode_Block_c, Pedit_c, Taxis_c, Measure_c, Area_c and P_Break_c. An unsupported action is rejected; supplying an arbitrary drawing-command value does not make it a supported direct action.

Transform parameters

All fields below are Double in v3; R2 used Single. XY coordinates use drawing units, scale factors are unitless, and theta uses degrees. RotateStay_p, MirrorStay_p and ScaleStay_p choose whether DoAvaxAction creates transformed copies or updates the supplied objects. Move_c adds the supplied XY distances to the existing objects; Copy_c applies those distances to new copies and leaves the originals in place. Zero distance is allowed. The action returns one-based result handles and supports Undo/Redo. Distances must be finite; invalid values reject the complete request without changing geometry or history.

Value type	Field As Double	Purpose
AvaxMove_t	xDist	Signed displacement along X, in drawing units. Positive values move toward increasing X; zero preserves X.
AvaxMove_t	yDist	Signed displacement along Y, in drawing units. Positive values move toward increasing Y; zero preserves Y.
AvaxRotate_t	RotX	X coordinate of the rotation center.
AvaxRotate_t	RotY	Y coordinate of the rotation center.
AvaxRotate_t	theta	Rotation angle in degrees; positive angles rotate counterclockwise.
AvaxMirror_t	mirX1	X coordinate of the first point on the mirror axis.
AvaxMirror_t	mirY1	Y coordinate of the first point on the mirror axis.
AvaxMirror_t	mirX2	X coordinate of the second point on the mirror axis.
AvaxMirror_t	mirY2	Y coordinate of the second point on the mirror axis.
AvaxScale_t	sclx1	X coordinate of the scale base point.
AvaxScale_t	scly1	Y coordinate of the scale base point.
AvaxScale_t	sclx2	X scale factor; a negative factor reflects the X direction.
AvaxScale_t	scly2	Y scale factor; a negative factor reflects the Y direction.

Transform_m matrix

Pass `CType(Avax.mAvax.Transform_m, Avax.AvaxCommand)` as the action and a boxed `Double(3, 3)` array as vIn. The enum retains its R2 value, 203. The logical 3 by 3 matrix occupies indices 1 through 3; row and column zero are reserved. All nine logical entries must be finite. For an affine matrix, set `m(1,3)=0`, `m(2,3)=0` and `m(3,3)=1`. Coordinates use the row-vector convention: `xNew = x*m(1,1) + y*m(2,1) + m(3,1)` and `yNew = x*m(1,2) + y*m(2,2) + m(3,2)`. Translation therefore belongs in row 3. Z coordinates are retained. The direct action updates the supplied objects in place; use the TransformItems overload with fTransformStay=True for transformed copies. See MoveItems and CopyItems for displacement examples.

Close existing polylines

For Close_c, vIn is unused. Every target must be a polyline containing at least two connected segments. Consecutive endpoints must have exactly matching XYZ coordinates. A disconnected input rejects the complete batch before any object is changed. If the last XYZ endpoint differs from the first, a closing segment is appended without replacing either endpoint. A tiny XY difference or a different Z remains significant. An exactly closed polyline receives no extra segment. Existing handles and attributes are retained; fNewHandles is False. The batch is one undoable edit, and successful completion refreshes the result selection.

Preserving segments in replacement edits

Pedit_c appends usable source segments into one paired-segment polyline without sorting, reversing or removing duplicates; attributes follow the first usable source. Explode_c replaces a multi-segment object with individual lines. Taxis_c orders each source polyline by exact XY contacts and removes exact duplicates and fully contained collinear XYZ edges. Partial overlaps, different elevations, finite short segments and Z-only edges remain significant. Original XYZ endpoints are preserved and positive gaps are not closed. If Taxis has no nonzero XYZ segment, it rejects the batch and preserves the source objects and history. See Pedit and Taxis for the command contracts and a complete example.

Public value structures

These are independent value types. Constructing one does not read the current pen or drawing. New values contain numeric zero, False and Nothing strings, including nested structures. Assigning or boxing a value copies it; editing the copy, including its nested fill or text, leaves the original unchanged. Strings distinguish Nothing, empty text and Unicode content.

In v3, fields formerly declared Single use Double, and fields formerly declared Short use Integer. Byte, Boolean, String and enum fields retain their declared kinds. The structures themselves do not validate whether a stored value is suitable for a particular drawing operation. AvaxAttributes_t, AvaxStretch_t and AvaxStatus_t remain available as public data containers; no automatic connection to control state is implied.

AvaxAttributes_t

Field	Type	Purpose
Type_a	AvaxItemType	Classification of the drawing item; its default is Any_i.
Color_a	Integer	Drawing color value.
Width_a	Integer	Drawing line width value.
Style_a	Integer	Line or point style value appropriate to the item.
Layer_a	Integer	Layer index.
Fill_a	AvaxFill_t	Nested fill/content attributes.
ShapeType_a	AvaxShapeType	Shape classification; its default is Any_s.
Text_a	AvaxText_t	Nested text and font attributes.
Elevation3D_a	Double	Separate object elevation attribute.
Height3D_a	Double	Separate object height attribute.

AvaxFill_t

Field	Type	Purpose
iType	Byte	Fill/content kind. Interpret it using AvaxFillType for the operation.
lHatchIndex	Integer	Type-dependent hatch, resource or content index.
iInForeColor	Integer	Foreground color.
iInBackColor	Integer	Background color.
sPattern	String	Pattern name or content specification for the chosen kind.
zOrder	Integer	Drawing order value.
lFlags	Integer	Flags interpreted according to the fill/content kind.

AvaxText_t

Field	Type	Purpose
Text_t	String	Text content.
FontNumber_t	Integer	Font-resource index.
fItalic_t	Boolean	Italic style.
fUnderline_t	Boolean	Underline style.
fStrikeOut_t	Boolean	Strikeout style.
fBold_t	Boolean	Bold style. Each style flag is independent.
lCharSet_t	Integer	Font character-set value.
lTextLines	Integer	Stored line count.
lTextChars	Integer	Stored character count. Assigning Text_t does not recalculate either count.

AvaxStretch_t

Fields	Type	Purpose
xOldMin, yOldMin	Double	Minimum corner of the source rectangle.
xOldMax, yOldMax	Double	Maximum corner of the source rectangle.
xNewMin, yNewMin	Double	Minimum corner of the destination rectangle.
xNewMax, yNewMax	Double	Maximum corner of the destination rectangle.

The original stretch parameters describe source and destination extents from which scaling and translation can be derived. Constructing this value does not execute a stretch or enable a direct action that is otherwise unsupported.

AvaxStatus_t

Field	Type	Purpose
zoomon	Integer	Zoom interaction state.
selecton	Integer	Selection interaction mode.
rotateon	Integer	Rotation state.
mirroron	Integer	Mirror state.
scaleon	Integer	Scaling state.
StretchOn	Integer	Stretch interaction state.
moveon	Integer	Move state.
lineon	Integer	Line or other line-based construction state.
OsnapOn	Integer	Object-snap mode.
intersecon	Integer	Intersection operation state.
SelectVertices	Integer	Vertex-selection state.
SnapOn	Integer	Grid-snap state.
OrthoOn	Integer	Orthogonal drafting state.
GridOn	Integer	Grid display state.
SelectAdd	Integer	Add/subtract selection mode.
pointOn	Integer	Point construction state.

These fields contain Integer states, not Boolean fields. Several historical modes have values above one. GetAvaxStatus supplies individual ByRef Integer outputs; the example below assembles them into a value.

Complete VB.NET examples

Build and copy attributes and stretch parameters

```
Option Strict On
Imports Avax

Public Module Help_CreateAttributeValues
    Public Function Create() As AvaxAttributes_t
        Dim value As New AvaxAttributes_t With {
            .Type_a = AvaxItemType.Text_i, .Color_a = 6, .Width_a = 1,
            .Style_a = 0, .Layer_a = 0, .Elevation3D_a = 2.125R,
            .Height3D_a = 4.25R,
            .Text_a = New AvaxText_t With {.Text_t = "Sample", .FontNumber_t = 1,
                .fBold_t = True, .lTextLines = 1, .lTextChars = 6}}
        Dim copy = value
        copy.Text_a.Text_t = "Independent copy"
        Return value
    End Function

    Public Function StretchBounds() As AvaxStretch_t
        Return New AvaxStretch_t With {
            .xOldMin = 1000000000000.125R, .yOldMin = 0,
            .xOldMax = 1000000000010.125R, .yOldMax = 10,
            .xNewMin = 1000000000020.375R, .yNewMin = 5,
            .xNewMax = 1000000000040.375R, .yNewMax = 25}
    End Function
End Module
```

Close supplied existing polylines

```
Option Strict On
Imports Avax

Public Module Help_CloseExistingPolylines
    Public Function Close(cad As cAvax, entityHandles As Integer()) As Integer()
        Dim input As Object = Nothing, extra As Object = Nothing
        Dim command = AvaxCommand.Close_c, created As Boolean
        Dim output As Integer() = Nothing, errorCode As Integer
        If Not cad.DoAvaxAction(entityHandles, input, command, created,
                                output, extra, errorCode) Then
            Throw New System.InvalidOperationException(
                "The polylines could not be closed. Error: " & errorCode.ToString())
        End If
        Return output
    End Function
End Module
```

Read an interaction-status value

```
Option Strict On
Imports Avax

Public Module Help_ReadInteractionStatus
    Public Function Read(cad As cAvax) As AvaxStatus_t
        Dim value As New AvaxStatus_t
        If Not cad.GetAvaxStatus(value.zoomon, value.selecton, value.rotateon,
            value.mirroron, value.scaleon, value.StretchOn, value.moveon,
            value.lineon, value.OsnapOn, value.intersecon, value.SelectVertices,
            value.SnapOn, value.OrthoOn, value.GridOn, value.SelectAdd,
            value.pointOn) Then
            Throw New System.InvalidOperationException("Status is unavailable.")
        End If
        Return value
    End Function
End Module
```

Immediate actions and selection borders

API reference

DoAvaxAction can execute the immediate actions below against a started, enabled control. Pass a one-based Integer array containing at least one existing handle; duplicate handles are removed in their first-occurrence order. Invalid requests return False with errCode=2 before changing input, selection or drawing history. A stopped, disabled or disposed control returns errCode=1. vIn is unused for these actions, and vOut is Nothing. Coordinates and dimensions use Double; handles and state outputs use Integer.

Drafting and display modes

Actions	Effect
Ortho_On_c, Ortho_Off_c, Ortho_c	Enable, disable or toggle orthogonal drafting.
Plane_On_c, Plane_Off_c, Plane_c	Enable, disable or toggle the plane setting for subsequent objects.
FastText_On_c, FastText_Off_c, Fasttext_c	Enable, disable or toggle simplified text display.
NoText_On_c, NoText_Off_c	Hide or display text.
NoWidth_On_c, NoWidth_Off_c	Suppress or display drawing line widths.
ZoomUndo_On_c, ZoomUndo_Off_c	Enable or disable adding subsequent view changes to normal Undo/Redo as Zoom_u. The separate Previous_c list remains available.
Freeze_c, Thaw_c	Set or clear the frozen bit of the current PenLayer. Other layer status bits remain intact.

These actions preserve the current drawing command, accepted XYZ coordinates and object selection. OutHandles acknowledges the unique supplied handles and fNewHandles is False. Changing a drafting or display mode does not create a geometry Undo entry. Freeze/Thaw affect the current pen layer, independently of the layers of InHandles. A real layer-status change is one undoable edit; repeating the same state adds no edit. LockEdit rejects Freeze/Thaw with errCode=1. The other listed modes may change while drawing input is paused or editing is locked.

Grid_c, Grid_On_c, Grid_Off_c, SetGrid_c, Snap_c and SetSnap_c are also supported direct modes. See Grid and coordinate snap commands for their configuration/toggle names and suppressed-dialog behavior.

Selection actions

Select_All_c selects visible drawing roots that match the current SetSelectFilter mask. Its result is the applied one-based handle array, in ascending order. Unselect_c clears the selection and returns {0}. Both set fNewHandles=False and publish SelectChange only when the selection changes. A callback sees committed selection; if it starts another command or drawing, the outer action does not reapply its former selection. OutHandles continues to describe the result of the requested action.

These selection commands end transient construction input through the normal command path. An accepted Polyline chain is completed once before selection is applied. If its completion fails or a host callback takes ownership of another operation, the requested selection returns False with errCode=1. Invalid InHandles never triggers this completion.

Border around supplied objects

Border_c adds one closed rectangular polyline around the overall XY bounds of all supplied objects. It keeps the original objects and uses the current pen attributes. Vertex Z comes from the first supplied object's geometry, matching the selected Border command. All four edges and the exact repeated first endpoint are retained, including very small frames and zero-width or zero-height bounds.

OutHandles contains the new border handle and fNewHandles=True. The border is one creation batch and one Undo entry. LockEdit rejects the action. Geometry and handle space are prepared before input is consumed. Previously accepted Polyline segments are completed first, as a separate creation; they are never discarded to make the border. The method returns the requested border's handle, rather than the earlier Polyline's handle. A callback taking ownership during that earlier completion prevents the outer border from being created. Creating the border then consumes old transient input and restores the SelectStay_p idle mode before its creation callbacks.

Complete VB.NET examples

Pass a started control and a valid one-based target array to these procedures. The target array is required even for a global mode change.

Change Ortho while retaining a live line

```
Option Strict On
Imports Avax

Public Module Help_DirectOrtho
    Public Function Apply(cad As cAvax, entityHandles As Integer(), enabled As Boolean) As Boolean
        Dim action = If(enabled, AvaxCommand.Ortho_On_c, AvaxCommand.Ortho_Off_c)
        Dim input As Object = Nothing, extra As Object = Nothing
        Dim created As Boolean, output As Integer() = Nothing, errorCode As Integer
        Return cad.DoAvaxAction(entityHandles, input, action, created, output, extra, errorCode)
    End Function
End Module
```

Create one border around several objects

```
Option Strict On
Imports Avax

Public Module Help_DirectSelectionBorder
    Public Function Draw(cad As cAvax, entityHandles As Integer()) As Integer()
        Dim action = AvaxCommand.Border_c
        Dim input As Object = Nothing, extra As Object = Nothing
        Dim created As Boolean, output As Integer() = Nothing, errorCode As Integer
        If Not cad.DoAvaxAction(entityHandles, input, action, created, output, extra, errorCode) Then
            Throw New System.InvalidOperationException("Border could not be created. Error: " &
    errorCode.ToString())
        End If
        Return output
    End Function
End Module
```

Select visible lines

```
Option Strict On
Imports Avax

Public Module Help_DirectVisibleLines
    Public Function SelectLines(cad As cAvax, entityHandles As Integer()) As Integer()
        ' The filter remains active for subsequent interactive selections.
        cad.SetSelectFilter(2)
        Dim action = AvaxCommand.Select_All_c
        Dim input As Object = Nothing, extra As Object = Nothing
        Dim created As Boolean, output As Integer() = Nothing, errorCode As Integer
        If Not cad.DoAvaxAction(entityHandles, input, action, created, output, extra, errorCode) Then
            Throw New System.InvalidOperationException("Selection could not be completed. Error: " &
    errorCode.ToString())
        End If
        Return output
    End Function
End Module
```

Legacy cursor and refresh commands

These Integer values belong to `Avax.mAvax`. Assign a converted value to Command, for example `cAvax1.Command = CType(Avax.mAvax.Refresh_m, Avax.AvaxCommand)`. The getter still reports the active drawing command. A started, enabled control preserves its accepted points, selection, document and undo history. BeginUpdate batching remains effective.

Value	Code	Purpose
Refresh_m	202	Present the existing drawing and clear transient cursor/preview graphics, including when AutoRedraw is False. This is a presentation refresh; use Regen when external sources must be regenerated.

Value	Code	Purpose
HideCursor_m	204	Temporarily clear the CAD crosshair, pickbox, snap marker, rubber-band preview and MouseDrawing graphics. Persistent object-selection adornments remain. The next pointer

Value	Code	Purpose
		or coordinate update restores the appropriate transient graphics. A host/custom Windows pointer that is already visible remains available.

DoAvaxAction also accepts these values using its normal valid one-based handle-array requirement. vIn is unused, vOut is Nothing, fNewHandles is False, and OutHandles reports the supplied unique handles. Use Command for a display-only request on an empty drawing. The operations do not move the system mouse or change its global visibility counter.

cAvax palette reference

API reference

A color property normally identifies a zero-based palette slot. Its value is an OLE color Integer: red is the least significant byte, green the next byte, and blue the third byte. Keep the slot number and its color value distinct when importing or changing palettes.

Public API

SetPalette supplies at least 16 entries, with no legacy 256-entry ceiling in v3. Native resource indices are nonnegative Integer values; practical capacity depends on memory. GetPalette returns the current table through its ByRef array and returns the number of entries. GetPalColor obtains a slot's displayed color and may apply background-contrast rules; use GetPalette when you need the stored RGB values. The v3 slot arguments use Integer instead of the old Short parameters. The color value itself remains Integer.

Saving and reopening

The native .avax format preserves the complete resource table. A .blk with expanded resource tables is written in the native format and requires v3. Legacy .avx block levels use byte indices and therefore support at most 256 distinct local color slots. When a drawing contains larger unions of independent block resources, the exporter assigns local tables at each block level. It fails without replacing the destination when one level cannot be represented.

Files written by v3 retain the logical palette count and its exact values. Repeated save/reopen operations do not add reserved slots to the application palette and do not change the existing color indices. The live drawing's palette is not modified by export. Untagged legacy tables remain readable without guessing that their final entries are padding.

The AVX header includes a framed AVAXPALETTE field, revision 1, with the logical count. The COLORS field has a reserved zero tail for the original R2 reader. These are header fields within the existing AVX v2 container, not a change to the .avax document version. Invalid counts, unsupported field revisions and nonzero reserved bytes are rejected instead of silently discarding color data.

Using files with the original R2 DLL

Operation	Behavior
Open AVX with the v3 DLL	All logical entries, including slot 255, retain their exact stored values.
Open v3-written AVX with original R2, 16-255 colors	All specified colors are retained. R2 may report one additional unused black slot.
Open v3-written AVX with original R2, 256 colors	Slots 0-254 are retained. The original R2 reader substitutes black for slot 255 because of its copy-length defect. Use v3 when exact preservation of that slot is required.
Insert a block using v3 Add_AVX	Source colors and fonts are merged into the receiving drawing. Text attributes and explicit font mode survive Explode.
Insert a block using original R2 Add_AVX	The original host retains its own palette and font tables. An index can therefore refer to a different color or face from the block's source table.

Replacing the DLL in an older application and opening files in an unchanged R2 application are different integration cases. The new DLL corrects the old reader defect; it cannot alter the code already running inside an unchanged R2 DLL.

VB.NET: read the stored last color

```
Dim colors() As Integer = Nothing
Dim count As Integer = cad.GetPalette(colors)
If count > 0 Then
    Dim lastIndex As Integer = count - 1
    Dim storedColor As Integer = colors(lastIndex)
    Dim red As Integer = storedColor And &HFF
    Dim green As Integer = (storedColor >> 8) And &HFF
    Dim blue As Integer = (storedColor >> 16) And &HFF
End If
```

The example assumes an initialized cAvax control named cad. It reads the stored table, so display contrast does not change the returned channels.

Real palette entries 256 through 259 take precedence over historical display-color aliases when those entries exist. NearestPaletteIndexColor reuses an exact match or appends the requested RGB color; it no longer approximates colors at 256 entries. See native resource capacity for all four resource tables.

Resource table capacity

API reference

V3 no longer limits layers, line styles, palette colors or fonts to the old 255/256 slots. Resource tables and entity references use Integer indices. Capacity is governed by available memory and the native file's integrity checks; it is not an unlimited-memory guarantee.

Resource	V3 indexing	Legacy .avx boundary per block level
Layers	Nonnegative indices; the default table still starts with 256 entries.	256 local slots, 0-255.
Line styles	Zero is continuous; positive indices reference the full style table.	255 named styles plus continuous slot zero.
Palette colors	Zero-based table with at least 16 entries. Missing exact RGB colors can be appended.	256 local slots, 0-255. See R2 palette interoperability.
Fonts	Positive indices reference the full table; zero retains its inherited/explicit-name semantics. Charset values remain -1 or 0-255.	256 font slots, 1-256, plus reserved zero.

Creation, attribute changes, Undo, rendering, native .avax storage, plain XML and the PAL/FNT/LTP resource files retain expanded indices. Existing files remain readable. Real palette indices 256-259 and line-style indices above 255 take precedence over historical display aliases when the corresponding table entries exist.

Saving .avx compacts the resources used at each block level into compatible local tables without changing the live drawing. Unused slots and duplicate resources can be reused. If the required resources cannot fit, saving fails with a message to use .avax; an existing destination is preserved. Native .avax files preserve the entire table, including unused resources.

A .blk with expanded resource tables is saved using the native file signature and requires v3. V3 reads both native and legacy BLK files. A BLK whose tables fit the legacy range retains the existing legacy encoding.

The CAD layer editor permits new rows, the line-style editor retains an append row, and palette editors scroll through all colors while painting only visible cells. Font selectors keep the actual table positions, including inherited entries.

Navigation commands and view history

API reference

Previous_c

Restore the most recently recorded view. The R2 history limit is 50 previous views. Reapplying an identical view does not consume a slot. SetBoundsZoom, completed pan/dynamic zoom gestures and completed scrollbar movements share this history. A completed gesture records one entry. Calling Previous_c with no remaining entry leaves the current view unchanged.

Page navigation

PgUp_c, PgDn_c, PgLeft_c and PgRight_c move the viewport by one visible canvas height or width. The canvas excludes enabled rulers and scrollbars. Horizontal distances respect ScreenXFactor_p; vertical direction respects OriginTopLeft_p. Up_c/Down_c use two UI font heights, and Left_c/Right_c use two UI character widths. These commands change the view without creating drawing objects.

UserLimits_c

Configure the work-area XY limits using the minimum X/Y and maximum X/Y values. Accepted limits must be finite, have positive finite width and height and remain representable as Double. Small valid windows at large coordinate origins are accepted. Cancellation, invalid input or a host change to the drawing, lifecycle or current construction command abandons the pending input.

Zoom recording and image memory

ZoomUndo_On_c adds subsequent changed views to normal Undo/Redo as Zoom_u; ZoomUndo_Off_c stops adding these records and retains existing entries. View Undo recording is off by default. CurrentUndo/CurrentRedo and the Undo/Redo events report Zoom_u. Previous_c keeps its own 50-view list independently of these commands. ZoomMemoryOn_p independently controls cached zoom images.

Other navigation commands

Zoom_c accepts opposite corners of a zoom window. Zoom_Pan_c pans with a drag and Zoom_Dynamic_c zooms about the starting drag position. PgPan_c and PgDynamic_c select the corresponding navigation modes. All_c shows the work area; Extends_c fits stored drawing bounds; CalcExtends_c recalculates those bounds before fitting. CalcBounds_c recalculates bounds without creating geometry. AutoLimits_c includes DownLeftFactor_p and UpRightFactor_p margins. Resize_c refreshes the viewport and fits the drawing when FitOnResize_p is enabled. On an actual control resize with fitting disabled, the horizontal view bounds and vertical center remain fixed while the vertical span follows the new canvas aspect ratio.

Complete VB.NET examples

Use a started, enabled control on its owning UI thread. SetBoundsZoom fits the requested rectangle to the canvas aspect ratio and restricts it to the configured work area; configure that area with SetBoundsAvax when navigating beyond its limits. Numeric drawing coordinates use Double in v3; command and parameter enum values use Integer.

Specify a view with Double coordinates

```
Option Strict On
Imports System
Imports Avax

Public Module Help_ViewWindow
    Public Sub Show(cad As cAvax, left As Double, bottom As Double,
                    right As Double, top As Double)
        If Not cad.SetBoundsZoom(left, bottom, right, top) Then
            Throw New ArgumentException("A finite, nonzero zoom window is required.")
        End If
    End Sub
End Module
```

Restore the previous view

```
Option Strict On
Imports Avax
```

```
Public Module Help_ViewHistory
    Public Sub Restore(cad As cAvax)
        cad.Command = AvaxCommand.Previous_c
    End Sub
End Module
```

Enable or stop view recording

```
Option Strict On
Imports Avax

Public Module Help_ViewRecording
    Public Sub Configure(cad As cAvax, enabled As Boolean)
        cad.Command = If(enabled, AvaxCommand.ZoomUndo_On_c, AvaxCommand.ZoomUndo_Off_c)
    End Sub
End Module
```

cAvax, AvaxCommand, AvaxProp Member Reference

API reference

CopyInside creates an offset inside a source boundary; CopyOutside creates one outside it. The originals remain in the drawing. Both methods accept a one-based Integer handle array and a Double distance in drawing units. Negative method distances reverse the named direction. The control must be started on its owning UI thread.

Geometry

Polygon copies intersect the parallel sides to retain full mitered corners, including acute corners. The resulting contour is cleaned so an inward offset may split into several components or disappear when the region is completely eroded. Copies are editable polylines. Circles and arcs remain analytic, and ellipse offsets change their world principal semiaxes.

Constant Z is retained exactly. A planar boundary remains on its plane. For a nonplanar boundary, heights are interpolated from the nearest original XY edge, averaging distance ties. Source attributes and properties are retained. A distance or corner that cannot be resolved at the polygon's coordinate precision fails before any copy is added.

Concave polygons also retain narrow features when their length and width differ greatly. Both outward and inward offsets preserve full mitered corners. An inward offset can split the result into separate closed contours or remove it completely; an outward offset can close a narrow bay. Results must still be representable as distinct Double coordinates.

Convex polygons retain very narrow sides and acute corners across the Double coordinate range. Their shifted supporting lines are intersected before rounding the output coordinates. An inward offset may remove a redundant side or erode the polygon completely. If the result exceeds Double, collapses when stored, or the distance is too small to move the stored boundary, the operation fails atomically.

Return value and handles

The methods return True if copies were created. The overload with vRetHandles assigns an Object containing an Integer array with an unused zero element followed by this operation's new handles. Source handle arrays and distance arguments remain unchanged. Repeated source handles are processed once.

Expected preparation errors return False and leave the output argument, drawing and history unchanged. Completely eroded sources produce no new entity; surviving results are committed together. Originals remain in place. The complete copy is one Undo action, and notifications observe the complete result. Exceptions thrown by host event handlers propagate normally.

Copy_In_c, Copy_Out_c and CopyInOutWidth_p

These vertex commands create one offset polyline from a common boundary in the XY plane, with Z=0 and the current pen attributes. Existing temporary SelectV/MakeSv vertices take priority, even when insufficient. Otherwise, selected lines, polylines or curves must form one unambiguous connected boundary; a single closed object is also accepted. Separate closed boundaries are rejected. Selected ellipses use sixty chords. Temporary XYZ vertices remain reusable; object selection is cleared only when it supplied the boundary. The ordinary result remains unselected. Surviving contours after inward erosion are stored as segment pairs in the same result polyline. CopyInOutWidth_p supplies the initial distance; its default is 0.2. Setting the property to zero restores that default. The commands use the magnitude of the stored value and remember the accepted positive distance. With dialogs disabled, that distance is used directly. An invalid or unsupported source rejects the batch without adding partial copies.

Offset_c also uses full polygon miters. It retains its own accepted distance and uses the selected side point to choose inward or outward movement.

DoAvaxAction: Copy_In_c and Copy_Out_c

These immediate actions extract one common boundary from the explicitly supplied one-based InHandles array. Duplicate handles are removed. The explicit boundary takes priority over current selection and temporary vertices. The output is one polyline at Z=0 using the current pen attributes, including when an inward offset splits into several contours. CopyInOutWidth_p supplies the distance magnitude; vIn is unused and may be Nothing. The property remains unchanged. Direct calls do not open a distance dialog.

On success, fNewHandles is True, OutHandles contains the new handles with an unused zero element, vOut is Nothing and errCode is zero. The new polyline is selected and forms one Undo action. Originals remain in the drawing. Invalid or unsupported input, insufficient handle space, or a completely eroded result returns False with errCode=2 before changing accepted construction input. Failure outputs are fNewHandles=False, OutHandles={0} and vOut=Nothing.

The control must be started and enabled, and editing must be unlocked. PauseAvax does not block this programmatic operation. An accepted pending Polyline is completed separately before the requested copies. If its completion handler starts a different command, changes selection, disables the control or locks editing, the direct copy stops with errCode=1. Source geometry and the distance property are read again after that completion. Exceptions from host handlers propagate; edits already committed remain undoable.

VB.NET example

This complete example creates a triangle at Z=7 and copies it outward by one drawing unit. Its original vertices are (0,0), (12,0) and (0,5). The copied vertices are (-1,-1), (17,-1) and (-1,6.5), all at Z=7. The corner at (17,-1) is the full intersection of the shifted sides. Pass an already started control.

```
Option Strict On
Imports Avax
Imports System

Public Module Help_OffsetMiter
    Public Function CopyTriangle(cad As cAvax) As Integer
        If cad Is Nothing Then Throw New ArgumentNullException(NameOf(cad))
        Dim x = New Double() {0, 0, 12, 0, 0}
        Dim y = New Double() {0, 0, 0, 5, 0}
        Dim z = New Double() {0, 7, 7, 7, 7}
        Dim source = cad.Add_PolyLine(x, y, z)
        If source = 0 Then Throw New InvalidOperationException("The triangle could not be added.")
        Dim sourceHandles = New Integer() {0, source}
        Dim distance As Double = 1
        Dim output As Object = Nothing
        If Not cad.CopyOutside(sourceHandles, distance, output) Then
            Throw New InvalidOperationException("The triangle could not be offset.")
        End If
        Dim created = DirectCast(output, Integer())
        Return created(1)
    End Function
End Module
```

Copy explicit objects through DoAvaxAction

Pass a started control, one-based handles forming one connected boundary and a positive distance in drawing units. Use Copy_In_c in place of Copy_Out_c for an inward copy.

```
Option Strict On
Imports Avax
Imports System

Public Module Help_OffsetDispatch
    Public Function CopyOut(cad As cAvax, entityHandles As Integer(), distance As Double) As Integer()
        If cad Is Nothing Then Throw New ArgumentNullException(NameOf(cad))
        If Double.IsNaN(distance) OrElse Double.IsInfinity(distance) OrElse distance <= 0 Then
            Throw New ArgumentOutOfRangeException(NameOf(distance))
        End If
        cad.SetAvaxProperty(AvaxProp.CopyInOutWidth_n, CObj(distance))
        Dim action = AvaxCommand.Copy_Out_c
        Dim input As Object = Nothing, extra As Object = Nothing
        Dim created As Boolean, output As Integer() = Nothing, errorCode As Integer
        If Not cad.DoAvaxAction(entityHandles, input, action, created, output, extra, errorCode) Then
            Throw New InvalidOperationException("The objects could not be offset. Error: " &
    errorCode.ToString())
        End If
        Return output
    End Function
End Module
```

Offset temporary vertices

Pass a started control. This example creates a planar offset from three temporary XYZ vertices. It adds one polyline at Z=0 and leaves the temporary vertices available for another vertex command.

```
Option Strict On
```

```
Imports Avax
Imports System

Public Module Help_OffsetVertices
    Public Sub Create(cad As cAvax)
        If cad Is Nothing Then Throw New ArgumentNullException(NameOf(cad))
        cad.SetAvaxProperty(AvaxProp.DialogEnabled, False)
        cad.SetAvaxProperty(AvaxProp.CopyInOutWidth, 1.0)
        cad.Command = AvaxCommand.SelectV
        cad.SetCoordinate(0, 0, 5)
        cad.SetCoordinate(12, 0, 6)
        cad.SetCoordinate(0, 5, 7)
        cad.Command = AvaxCommand.Copy_Out
    End Sub
End Module
```

Copy a very narrow triangle

The triangle is one trillion units long and one unit high. Copying it outward by one unit preserves the full acute corner. Coordinates and distances are Double.

```
Option Strict On
Imports Avax
Imports System

Public Module Help_ConvexOffset
    Public Function CopyNarrow(cad As cAvax) As Integer
        If cad Is Nothing Then Throw New ArgumentNullException(NameOf(cad))
        Dim x = New Double() {0, 0, 1.0E12, 0, 0}
        Dim y = New Double() {0, 0, 0, 1, 0}
        Dim z = New Double() {0, 7, 7, 7, 7}
        Dim source = cad.Add_PolyLine(x, y, z)
        Dim sourceHandles = New Integer() {0, source}
        Dim distance As Double = 1
        Dim output As Object = Nothing
        If source = 0 OrElse Not cad.CopyOutside(sourceHandles, distance, output) Then
            Throw New InvalidOperationException("The triangle could not be copied.")
        End If
        Return DirectCast(output, Integer())(1)
    End Function
End Module
```

Copy an ellipse while retaining its center

CopyOutside adds the distance to both world principal semiaxes; CopyInside subtracts it. The ellipse remains analytic, retaining its center, elevation and attributes. Rotation, reflection and shear are included in the world axes. Very different axis lengths do not require a polygon approximation. An inward distance that reaches the minor semiaxis erodes the ellipse completely. A result that cannot retain the center and endpoints at Double precision is rejected before anything is added.

```
Option Strict On
Imports Avax
Imports System

Public Module Help_EllipseOffset
    Public Function CopyEllipse(cad As cAvax) As Integer
        If cad Is Nothing Then Throw New ArgumentNullException(NameOf(cad))
        ' Pass an already started control. The source radii are 6 and 3.
        Dim source = cad.Add_Ellipse(10, 20, 7, 6, 3)
        If source = 0 Then Throw New InvalidOperationException("The ellipse could not be added.")
        Dim sourceHandles = New Integer() {0, source}
        Dim distance As Double = 1
        Dim result As Object = Nothing
        If Not cad.CopyOutside(sourceHandles, distance, result) Then
            Throw New InvalidOperationException("The ellipse could not be offset.")
        End If
        ' New radii: 7 and 4. Center: (10,20), elevation: 7.
        Return DirectCast(result, Integer())(1)
```

```
    End Function
End Module
```

OperationFailed

API reference

OperationFailed

cAvax Member Reference

API reference

Use the catalog to build a shape gallery, AddParametricShape to create a preset and the control-point methods to edit it. Pass an already started control on its owning UI thread to examples that create or edit drawing objects. The two Shared catalog/icon methods do not need a control instance.

Built-in shape catalog

The catalog contains the following 18 presets. Kind and parameter names are case-sensitive. Values in the range column are inclusive; every supplied value must be finite. Only Points requires an integer. Other parameters use Double values and have a suggested editor increment of 0.01. A missing parameter uses its default.

Kind	Parameter	Supported range	Default
Triangle	None	-	-
Right triangle	None	-	-
Square	None	-	-
Parallelogram	Skew	-0.9 to 0.9	0.25
Rhombus	Peak	0.1 to 0.9	0.5
Trapezoid	TopRatio TopOffset	0.01 to 1 -0.5 to 0.5	0.6 0
Pentagon	None	-	-
Hexagon	None	-	-
Octagon	None	-	-
Star	Points InnerRatio	Integer 3 to 100 0.05 to 0.95	5 0.45
Arrow	HeadLength ShaftWidth	0.05 to 0.9 0.05 to 0.9	0.35 0.35
Chevron	Thickness	0.05 to 0.8	0.35
Cross	ArmWidth	0.05 to 0.9	0.3
Heart	NotchDepth	0.05 to 0.65	0.25
Ring	InnerRatio	0.05 to 0.95	0.6
Speech bubble	TailPosition	0.2 to 0.8	0.4
Cylinder	Depth	0.05 to 0.4	0.2
Cube	Depth	0.05 to 0.4	0.2

Arrow, Chevron and Speech bubble have Category = `Symbols`; the other presets have Category = `Basic shapes`. DisplayName is the current Kind text. Read the returned descriptions to populate captions and controls instead of hard-coding the table. Width and height define the preset's local frame independently, including for the Square preset. Cylinder and Cube are 2D drawings; they do not create a 3D solid.

To read or change an existing preset's numeric values, use GetShapeParameters and ChangeShapeParameter. Their dictionary contains the preset's parameters plus ShapeWidth and ShapeHeight, the positive lengths of its current local frame axes. A direct parameter change must satisfy the catalog range; frame lengths must be positive. These frame keys are for editing an existing object and are not accepted in the initial AddParametricShape parameter dictionary. Use that method's width and height arguments for creation.

ParametricShapeDefinitions

RenderParametricShapeIcon

GetParametricShapes

AddParametricShape

GetParametricShapeKind

GetShapeControlPoints

ChangeShapeControlPoint

Add_PictureFile

API reference

Add_PictureFile

SetPointSelectionAspectRatio

API reference

SetPointSelectionAspectRatio

SetPointSelectionPreview

API reference

SetPointSelectionPreview

Polyline, Polygon and Close commands

API reference

Command starts an interactive drawing operation. Coordinates passed to SetCoordinate use Double values. Mouse input and typed coordinates follow the same construction rules.

Polyline_c

Accepts successive XYZ vertices and creates one polyline containing the connected segments. The accepted chain remains pending until the user presses Enter, requests Close_c, changes to an ordinary command or cancels. Enter requires at least two vertices and preserves an open chain. Changing command or cancelling commits an accepted chain of two or more vertices; a lone starting vertex creates no object. Requesting Polyline_c again preserves its current chain. Arc commands can continue the chain as described in Arc commands and segmentation.

Close_c

Closes the current polyline to its exact first XYZ vertex. At least three vertices must have been accepted. With fewer vertices, Close leaves the current input intact. It also leaves an unrelated construction command intact. If the final XYZ vertex already equals the first, no additional closing segment is added. A distinct nearby point, or a point with the same XY and a different Z, is preserved before the exact closing edge.

Polygon_c

Creates a regular closed polyline. Each invocation requests a whole number of vertices from 3 through 60. Pick the center, then a point defining the circumradius. The direction of the second pick does not rotate the polygon: its first vertex is on the positive X axis from the center, and subsequent vertices run counterclockwise in drawing coordinates. All generated vertex Z values are zero, as in R2. Pen elevation and height remain separate object attributes.

The second valid point completes the command. A zero radius or unrepresentable result is rejected while retaining the center for another pick. Cancelling the vertex-count request creates nothing; the previous drawing command has already ended. With DialogEnabled_p=False and no ParameterDialogInput callback, the command is cancelled because no vertex count was supplied. A callback can supply the accepted count without a built-in dialog. Use Add_ShapeRegularPolygon when the application needs a shape rather than an interactive polyline.

ParameterDialogInput

Completion and events

A completed polyline or polygon is one undoable creation and produces one NewUserItems batch with one-based Integer handles. Completion consumes the construction input and restores idle selection according to SelectStay_p before creation callbacks. A callback may start another operation without the outer completion overwriting it. Validation or handle-capacity failure creates no partial object or history entry. Polyline completion remains retryable; a rejected Polygon endpoint retains its center. Pen color, width, line style, layer, elevation and height are copied from the current pen.

VB.NET examples

Pass a started AVAX control to these procedures. All coordinates use Double. The examples require no external files.

Commit an open XYZ polyline

```
Option Strict On
Imports Avax

Public Module Help_DrawOpenPolyline
    Public Function Draw(cad As cAvax) As Boolean
        cad.Command = AvaxCommand.Polyline_c
        If Not cad.SetCoordinate(10.125R, 20.25R, 7.5R) Then Return False
        If Not cad.SetCoordinate(30.5R, 40.75R, -3.25R) Then Return False
        If Not cad.SetCoordinate(50.875R, 25.5R, 11.75R) Then Return False
        ' Ends the tool and commits its already accepted segments.
        cad.Command = AvaxCommand.Cancel_Keep_Select_c
        Return True
    End Function
End Module
```

Close a three-dimensional polyline

```
Option Strict On
Imports Avax

Public Module Help_DrawClosedPolyline
    Public Function Draw(cad As cAvax) As Boolean
        cad.Command = AvaxCommand.Polyline_c
        If Not cad.SetCoordinate(0, 0, 2.5R) Then Return False
        If Not cad.SetCoordinate(20, 0, 4.25R) Then Return False
        If Not cad.SetCoordinate(10, 15, 8.75R) Then Return False
        cad.Command = AvaxCommand.Close_c
        Return True
    End Function
End Module
```

Start interactive regular-polygon drawing

```
Option Strict On
Imports Avax

Public Module Help_BeginRegularPolygon
    Public Function BeginPolygon(cad As cAvax) As Boolean
        cad.SetAvaxProperty(AvaxProp.DialogEnabled_n, True)
        ' The control asks for 3 through 60 vertices.
        ' After OK, the user picks the center and a radius point.
        cad.Command = AvaxCommand.Polygon_c
        Return cad.Command = AvaxCommand.Polygon_c
    End Function
End Module
```

Property geometry layouts

GetProperties returns one-based Object() arrays. Each geometric item is a Double() with the layout below. Lengths include slot zero: Dim data(6) creates seven entries. Slot zero is reserved except where the curve discriminator is specified.

Object type	Array length	Indices and meaning
Point_i	4	Index 0 reserved; 1..3 = X, Y, Z.
Line_i	7	Index 0 reserved; 1..3 = start XYZ; 4..6 = end XYZ.
Polyline_i	6n + 1	Index 0 reserved; each group of six values is a complete start/end XYZ pair. Shared vertices are repeated between consecutive segments. Independent segments are also preserved.
Shape_i	3n + 1	Index 0 reserved; triples of XYZ vertices. Shape points/lines may have one/two vertices. A closed contour may repeat its first vertex. Text/document shapes use four box corners, optionally followed by a closing corner. Preserve the returned corner order when transforming a text frame: rotation, shear and reflection also affect its glyphs. In imported R2 ordinary text frames, the first-to-second edge defines height and the first-to-fourth edge defines width. Do not assume all text frames start at the same geometric corner.
Ellipse_i	28	0 reserved; 1..6 derived world XYZ bounds; 7..9 local center XYZ; 10..12 radii (X/Y positive, third value retained); 13..15 local arc start XYZ; 16..18 local arc end XYZ; 19..27 row-vector affine matrix. Edit center/radii/endpoints/matrix; bounds are recomputed.
Text_i	14	Exactly 14 public entries. 0 reserved; 1..6 derived world XYZ bounds; 7/8 anchor X/Y; 9 non-negative width; 10 positive height; 11 reserved (normalizes to one); 12 baseline angle in degrees; 13 anchor Z. Edit anchor, width, height and angle; bounds are recomputed. An existing sheared or reflected text frame is retained: angle turns both axes, width/height scale their respective axes, and anchor translates the frame. No additional frame values are exposed in this array. Text content/font/style use the corresponding text APIs.
Document_i	16	0 reserved; 1..3 local anchor XYZ; 4 positive page width; 5 positive font height; 6..14 row-vector affine matrix; 15 Integer-valued font number (stored in the Double array). Nonpositive font numbers normalize to 1; positive numbers are not capped at the old byte range.
Curve_i (R2 arc)	19	Index 0 is meaningful: 1 = start/end/third-point arc, 2 = start/end/center arc. Indices 1..3 start XYZ, 4..6 end XYZ, 7..9 third-point or center XYZ, 10..18 row-vector affine matrix. Points must define a nondegenerate arc.
Curve_i (v3 cubic)	9n + 4	Index 0 = -4; then 3n+1 XYZ Bezier control points for n cubic spans, with the final point of one span reused by the next. This is a v3 representation, not the R2 arc layout.
Curve_i (v3 sampled)	3n + 1	Index 0 = 0; then n XYZ vertices, n >= 2. Some curves of arbitrary degree are represented as sampled paths. Broader R2 spline conversion/editing parity remains under review.
Block_i	7	0 reserved; 1..3 minimum XYZ; 4..6 maximum XYZ, derived from resolved contents. Changing the bounds scales/translates nested geometry. A zero-extent axis cannot be expanded through bounds alone. Empty/unresolved blocks allow unchanged geometry plus metadata edits. Blocks containing live external references allow Z translation but not Z scaling.
ExLines_i	7	0 reserved; minX/minY/minZ/maxX/maxY/maxZ. This public layout excludes the internal reference matrix. XY bounds and Z translation are editable while preserving the live source. Z scaling and expansion of a zero-extent axis are rejected.
Data_i	7 on read	No drawable geometry: GetProperties returns zero bounds. SetProperties/ChangeProperties ignore geometry, including Nothing, and update string metadata only.

ResolvedObjectSnap

API reference

ResolvedObjectSnap

Roof command

API reference

Roof_c

Command value 515 constructs the plan-view hips and ridges of an equal-pitch roof from a polygon boundary. The result consists of separate Line_i objects at Z=0, using the current pen attributes. The original boundary objects are retained. All generated roof lines belong to one Undo/Redo operation. Coordinates use Double and handles use Integer.

Boundary input

Set Command to Roof_c on a started, enabled control. A nonempty temporary sequence entered with SelectV_c or restored with LastSv_c takes priority over selected objects. Roof retains that temporary XYZ sequence and any unrelated object selection. Incomplete temporary input reports an error; it does not fall back to the selection.

When there are no temporary vertices, the selected lines, polylines, shapes, curves or sampled ellipses must form one connected boundary under the selected vertex boundary rules. Shared XYZ endpoints must match exactly. The final closing edge is implicit; an explicit repeated endpoint is accepted. At least three vertices forming a simple polygon with nonzero area are required. Self-intersecting, disconnected or branching boundaries are rejected. On success, ordinary Command clears the source selection without deleting its objects.

Roof geometry is calculated from XY; output line coordinates have Z=0. The command calculates an inward-moving edge construction, not roof elevations or a three-dimensional roof mesh. Outlines that the geometry calculation cannot resolve are rejected before roof objects are inserted.

Internal calculations retain small features independently of drawing units and the world origin. Final coordinates are stored as Double. If rounding would collapse a nonzero roof segment to one point, the complete result is rejected instead of silently omitting that segment.

DoAvaxAction

Pass Roof_c as myAction and a one-based InHandles array containing the boundary handles. Duplicate handles are used once, preserving their first occurrence. Explicit handles supply the boundary independently of temporary input and the current selection. vIn is unused; pass Nothing. No options dialog is opened.

Success returns True, fNewHandles=True, one-based OutHandles containing all new lines, vOut=Nothing and errCode=0. The new lines are selected. Failure returns False, fNewHandles=False, OutHandles={0} and vOut=Nothing. Error 2 indicates an invalid boundary or preparation failure; error 1 indicates an unavailable edit state or a newer request from a host event. Direct creation is allowed while paused if no pending Polyline needs completion. Locked, disabled and stopped controls reject creation.

Input and events

The complete result and its handle capacity are checked before accepted input is finished. A pending Polyline is completed as a separate Undo operation, and the boundary is then checked again. Invalid initial requests preserve accepted input, selection and Redo history. NewUserItems observes the complete roof batch. A newer command, selection or document supplied by a source-selection event takes precedence; an exception thrown by a host event is propagated unchanged.

DoVertex file job 515

DoVertex also accepts the legacy five-line roof job: command 515, record size (8, 12 or 24), input byte count, packed input filename and output filename. Filenames may be quoted. Binary records retain IEEE Single values for file compatibility; the first XY pair of each record supplies a vertex. Other fields are retained in the unchanged input file. Every numeric field in a record must be finite.

A processed request replaces the job with four reply lines: semantic status (1 for a roof, 0 for an invalid or empty boundary), kind 5, original input byte count and output filename. The method Boolean reports whether the request was processed; read the first reply line to determine whether roof geometry was produced.

On success the output contains one line with value 1 followed by six invariant-culture coordinate lines for each independent Line: X1, Y1, Z1, X2, Y2, Z2. Both Z coordinates are zero. A semantic failure leaves an existing output file intact. Malformed requests and publication failures return False and preserve existing files. When geometry is produced, the output filename must differ from the input and job filenames; aliases are rejected before publication. The request has no roof angle or settings file.

VB.NET example

Call this function with a started control and the handles of one boundary. The returned handles identify the generated roof lines.

```
Option Strict On
Imports Avax
Imports System

Public Module Help_Roof
    Public Function Create(cad As cAvax, boundaryHandles As Integer()) As Integer()
        Dim action As AvaxCommand = AvaxCommand.Roof_c
        Dim input As Object = Nothing, extra As Object = Nothing
        Dim created As Integer() = Nothing, newHandles As Boolean, errorCode As Integer
        If Not cad.DoAvaxAction(boundaryHandles, input, action, newHandles, created, extra, errorCode)
    Then
            Throw New InvalidOperationException("Roof failed. Error code: " & errorCode.ToString())
        End If
        Return created
    End Function
End Module
```

Interactive Command-based creation publishes NewUserItems as described here. A direct DoAvaxAction call is an automatic operation: use OutHandles and fNewHandles for its result; it publishes Change without NewUserItems. Previously accepted user construction completed before that action retains its own notification.

Runtime and OLE methods

API reference

InitProgramID

Unlock

OleDrag

cAvax, AvaxCommand Member Reference

API reference

Pedit_c creates a paired-segment polyline from usable source geometry. Taxis_c cleans and orders the records within each source polyline. Use Command after selecting the sources, or pass a one-based Integer handle array to DoAvaxAction. The control must have been started on its owning UI thread.

Pedit_c

Pedit appends the usable source segments in processing order. It retains the direction and XYZ endpoints of each record, including disconnected segments, duplicates, contained segments, short segments and Z-only edges. It does not sort or clean them. No artificial connecting segments are added. The result is one Polyline_i using the first usable source's attributes and properties; the converted sources are replaced.

Lines and polylines contribute their line geometry; supported curves are sampled by the existing curve conversion. Sources without usable segments and block containers are not converted. Zero-length XYZ records contribute no line segment. The separate temporary-vertex Pedit path constructs a polyline along the retained vertices and does not use Taxis cleanup.

Taxis_c

Taxis processes each selected Polyline_i independently and replaces it with a cleaned polyline. Other entity types are preserved. It removes zero-length XYZ edges and redundant duplicate or contained collinear XYZ edges. For wide polylines, a duplicate or contained edge is redundant only when its complete width profile agrees with the retained edge. A partial overlap or different elevation profile remains significant and is retained.

Ordering uses exact XY endpoint contacts. It may reverse a complete XYZ record to extend a chain, but does not move endpoints or close a positive gap. Segments that meet in XY may have different Z values at the contact; both heights are preserved. Disconnected components and branches remain separate records. Do not depend on a particular starting vertex or component order. These rules also apply to the DoVertex 516 file job.

If a source polyline has no nonzero XYZ segment, Taxis rejects the batch and preserves every source, including any earlier valid source in the same request. Attributes and properties belong to each source polyline; cleanup is not performed across separate source objects.

Start/end width vectors stay with their source segment when records are reordered or reversed. Different taper or width planes remain independent even when their centrelines coincide. Cleanup compares the complete width profile without a drawing-unit tolerance, so a small but representable width difference is preserved.

DoAvaxAction

InHandles is an Integer array with an unused zero element. Duplicate source handles are processed once, in first-occurrence order. Pass Pedit_c or Taxis_c as myAction; vIn is unused. On success fNewHandles is True, OutHandles contains an unused zero followed by all newly created handles, and errCode is zero. The input array and action value are unchanged.

Preparation completes before sources or history change. Each successful command is one undoable replacement; Undo restores the original records, including removed duplicates, and Redo restores the complete result. Notifications see the complete replacement. A newer drawing command started by an event handler remains active.

VB.NET example

This complete example creates four lines, combines them with Pedit, and then uses Taxis to remove the contained line and order the three remaining segments from (0,0) to (30,0). The two editing actions each have their own Undo entry. Pass an already started control.

```
Option Strict On
Imports Avax
Imports System

Public Module Help_SegmentEdits
    Public Function CreateOrderedPolyline(cad As cAvax) As Integer
        If cad Is Nothing Then Throw New ArgumentNullException(NameOf(cad))
        Dim sources = New Integer() {0,
            cad.Add_Line(10, 0, 0, 20, 0, 0),
            cad.Add_Line(30, 0, 0, 20, 0, 0),
```

```
            cad.AddLine(0, 0, 0, 10, 0, 0),
            cad.AddLine(2, 0, 0, 8, 0, 0)}
        Dim combined = Apply(cad, sources, AvaxCommand.Pedit, c)
        Return Apply(cad, combined, AvaxCommand.Taxis, c)(1)
    End Function

    Private Function Apply(cad As cAvax, sources As Integer(), action As AvaxCommand) As Integer()
        Dim input As Object = Nothing, extra As Object = Nothing
        Dim created As Boolean, result As Integer() = Nothing, errorCode As Integer
        If Not cad.DoAvaxAction(sources, input, action, created, result, extra, errorCode) Then
            Throw New InvalidOperationException("The polyline action failed: " & errorCode.ToString())
        End If
        If Not created OrElse result Is Nothing OrElse result.Length < 2 Then
            Throw New InvalidOperationException("The action did not return new polyline handles.")
        End If
        Return result
    End Function
End Module
```

Selected dimension command

API reference

Dim_c

Command value 79 adds linear dimensions for selected drawing geometry or an active temporary vertex sequence. The source objects remain in the drawing. The result consists of ordinary dimension line, mark, extension, point and text components, according to the current dimension settings. It is one Undo/Redo operation. Coordinates and distances use Double in v3; handles and command values use Integer.

Source and vertex rules

An active nonempty temporary vertex sequence takes precedence over selected objects and remains available for reuse. SelectItems ends active vertex entry; use LastSv_c to restore the last temporary sequence when combining it with an unrelated object selection. Raw temporary records are not deduplicated and do not lose a terminal closure. Temporary A,B,C produces three edge dimensions; temporary A,B,C,A fails because the implicit last-to-first edge has zero XY length. Even a one-record temporary request takes precedence and fails rather than falling back to a selected line.

Without temporary vertices, suitable selected paths supply one connected XYZ sequence. One exact repeated terminal XYZ closure is removed from extracted selected input. Disconnected, ambiguous or zero-length input rejects the complete operation. A selected point is not a dimension guide. Selected connected paths beyond the original single-line/polyline use case are supported by v3. See the selected vertex boundary rules for source types and extraction.

For selected and temporary sources, output geometry is planar at Z=0, as in the R2 vertex dimension route. Source Z values and the source objects themselves are retained. The measured value is the XY edge length, formatted with the current dimension factor, units, precision and suffix. See dimension formatting, component sizes and the vertex dimension settings.

Two source vertices: offset and side

The command first requests a positive finite offset distance. With DialogEnabled_p=False and no ParameterDialogInput provider, it uses DimDist_p; a nonpositive initial value becomes 1 for this two-point operation. With dialogs enabled, the parameter editor requests the distance. ParameterDialogInput can supply an accepted value or cancel without displaying a window.

After the distance is accepted, supply one placement point with SetCoordinate or normal drawing input. This point chooses the side of the source line. Its distance from that line does not change the accepted offset. An on-line point rejects and retains the guide for another side pick. The accepted offset is captured for this request and does not change if DimDist_p is subsequently edited. A two-point operation does not write its offset back to DimDist_p.

This fixed-offset, side-pick behavior follows R2 Dim_c. V_Dim_c with two vertices instead derives its local offset from the placement point. Use Dim3p_c when the measured endpoints and perpendicular placement should all be picked without an existing guide.

Three or more source vertices

The command requests a positive finite offset and dimensions every successive edge plus the last-to-first edge, even for an open source polyline. It uses the same local-neighbor placement, ordered planar components, text settings and complete preparation as V_Dim_c. An accepted offset updates DimDist_p only with a fully prepared, owned successful operation.

Completion, failure and ownership

Call on the control's owning UI thread after StartAvax. Interactive creation requires an enabled control with PauseAvax and LockEdit off. Geometry, dimension settings, resources and capacity are checked before insertion. Failure creates no partial dimension or history entry. Used source selection clears only after complete preparation; temporary input keeps unrelated selection. Ordinary result components are not selected automatically.

Cancellation and invalid parameter answers preserve sources, selection and DimDist_p. A drawing, selection or input change invalidates an older captured guide. A parameter or selection callback that begins a newer command keeps ownership of its input. Host callback exceptions retain their original identity. Command assignment has no Boolean return value; SetCoordinate reports whether its point was accepted.

For compatibility with existing v3 applications, Dim_c with no selected objects and no temporary vertices retains its source-free three-pick extension. That mode follows Dim3p_c placement and retains endpoint Z values. Applications can use Dim3p_c explicitly to request this mode regardless of existing selection.

VB.NET example

Supply an already started control. This example creates a ten-unit source at Y=5, accepts an offset of 2 and picks the upper side at Y=14. The dimension line is at Y=7. The distance from the placement point to the source is deliberately different from the accepted offset.

```
Option Strict On
Imports Avax

Public Module Help_SelectedDimension
    Public Function Draw(cad As cAvax) As Boolean
        cad.Command = AvaxCommand.Cancel_c
        cad.Command = AvaxCommand.Ortho_Off_c
        cad.SetAvaxProperty(AvaxProp.SnapOn_p, False)
        cad.SetAvaxProperty(AvaxProp.AutoOSnap_p, 0)
        cad.SetAvaxProperty(AvaxProp.DialogEnabled_p, False)
        cad.ParameterDialogInput = Nothing
        cad.SetAvaxProperty(AvaxProp.DimTextAbove_p, True)
        cad.SetAvaxProperty(AvaxProp.DimDist_p, 2.0R)
        Dim source As Integer = cad.Add_Line(0, 5, 0, 10, 5, 0)
        If source <= 0 Then Return False
        Dim objectIds As Integer() = {0, source}
        If Not cad.SelectItems(objectIds) Then Return False
        cad.Command = AvaxCommand.Dim_c
        Dim x As Double = 5, y As Double = 14, z As Double = 0
        Return cad.SetCoordinate(x, y, z)
    End Function
End Module
```

Selected vertex curves, centers and labels

API reference

These commands create new objects from an ordered boundary while retaining the source objects. Coordinates, elevations and text scales use Double. Handles, counts and command values use Integer. Call the control on its owning UI thread after StartAvax.

Command

Select suitable objects with SelectItems, then assign one of the commands below. You can also collect a temporary boundary with SelectV_c and SetCoordinate, or obtain it from one selected object with MakeSv_c. With no selected or temporary boundary, the five curve commands, Centroid_c, Radiate_c and VN_Count_c accept new coordinates; press Enter to finish. VT_Count_c operates on the selected or temporary vertices. Interactive editing is unavailable while input is paused, editing is locked, or the control is stopped, disabled or disposed.

Temporary vertices take precedence

For all nine commands on this page, a nonempty temporary vertex sequence takes precedence over selected objects. The command uses that sequence as one boundary or curve guide. It does not combine the temporary vertices with the selected objects. Even a single temporary vertex is authoritative: if the sequence is too short or otherwise unsuitable, the command creates no result or reports its validation failure. It does not fall back to a suitable selected object. Command has no Boolean result; use the drawing, error notification or NewUserItems handles to observe the outcome.

When temporary vertices are used, an unrelated selected object remains selected and its geometry is unchanged. Ordinary Command does not select the new result automatically, and creation alone does not require a SelectChange event. The temporary sequence remains available after any of these nine commands succeeds; it is not replaced by the generated samples or labels. It can be reused until another operation replaces or clears it. LastSv_c restores the last remembered temporary sequence after it has been cleared. MakeSv_c and LastSv_c create no drawing object themselves.

The selection-clearing behavior below applies only when the command actually uses selected objects because no temporary vertices are available. By contrast, DoAvaxAction always uses its explicit InHandles for the requested result; an existing temporary boundary does not redirect that direct action.

For an ordinary Command using selected objects with no temporary vertices, the complete result is validated and prepared before the source selection is cleared. The new result is not automatically selected. An initially invalid request retains its source selection. If the selection-clearing callback starts another command, changes the selection, edits or replaces the drawing, or ends the control, that newer state takes ownership and the prepared result is not added. A callback exception also prevents that result from being added and propagates to the caller.

Straight polylines imported by Add_DXF or Add_DWG are editable polyline objects. They retain their ordered XYZ segment endpoints and can be selected for MakeSv or boundary commands. With DXFCurveArcOn_p set to False, Add_DXF joins sampled bulged arcs into that same polyline, in source vertex order and with exact shared endpoints. CircleSectors_p and the arc segmentation parameters control the sampling. When DXFCurveArcOn_p is True, horizontal bulged arcs retain a composite analytic representation; Add_DWG also preserves these analytic arcs. Arcs in tilted planes are sampled in world XYZ and joined into a polyline. An analytic composite or an actual inserted block is not directly accepted as a single MakeSv boundary.

Boundary and topology rules

Lines, polylines, shapes, curves and ellipses can supply vertices. A suitable boundary must contain at least two finite XYZ vertices. Other object types supply no vertices; a request containing no suitable boundary fails. Composite objects and shape objects carrying hatch or dimension data are not expanded into an arbitrary set of boundaries.

Several selected open paths may form one connected chain. Endpoints must match exactly in all three coordinates; a nearby point or the same XY with a different Z does not join. Paths may be reversed or prepended during assembly. Disconnected segment pairs, branching joins, endpoints lying inside another path's segment, and duplicate, reversed or positively overlapping collinear edges across paths are rejected. A closed path cannot be combined with another selected boundary. Separate closed rings and holes are not combined. Use one ordered polyline when the starting vertex and numbering direction must be explicit.

One exact repeated closing XYZ vertex is removed from the control sequence. Distinct small edges are retained. Bezier and both B-spline commands preserve repeated controls; the two interpolating cubic-spline commands coalesce consecutive exact duplicate XYZ controls. Consecutive exact duplicate vertices are also omitted for centroid and both label counters. Centroid and Radiate additionally require at least three effective XY vertices forming a simple polygon with nonzero area; an open chain is implicitly closed for that calculation. Curves and counters do not require a simple polygon.

Bezier_C

Creates one polyline approximating a Bezier curve whose degree is one less than the number of control vertices. The first and last controls are endpoints; the remaining controls guide the curve rather than being interpolation points. Repeated controls influence the curve and are retained. The result has ten samples per control vertex, including the two exact endpoints. This command uses the complete control sequence, rather than independent groups of four.

CSpline1_c

Creates one interpolating cubic-spline polyline through the control vertices, with zero first derivative at both ends. Each span is represented by fifteen chord intervals. The endpoint condition distinguishes it from CSpline2_c.

CSpline2_c

Creates one interpolating cubic-spline polyline with natural end conditions: the second derivative is zero at each end. It uses fifteen chord intervals per span. Both interpolating commands preserve their endpoint coordinates and interpolate the supplied Z values.

OBSpline_c

Creates one open, clamped quadratic B-spline polyline. It passes through the first and last controls and generally does not pass through the interior controls. Twenty sampled points describe the result, including its exact endpoints.

CBSpline_c

Creates one periodic cubic B-spline polyline with exact XYZ closure. At least three effective controls are required. For three through twelve controls, midpoint guides refine the control sequence; larger sequences use the original guides. Each cyclic guide span has ten chord intervals. The closing sample is exactly the first sample. This construction produces a smooth closed boundary without treating the first control as an interpolation point.

All five curve results are stored as Polyline_i objects containing connected XYZ segment pairs. They preserve three-dimensional control geometry in v3. The supported maximum is 3,200 curve controls. Curve output that cannot be represented as finite geometry is rejected before any object is added.

Centroid_c

Creates one Point_i at the polygon's area centroid in XY. This is the center weighted by enclosed area, not the arithmetic mean of the vertices. Clockwise and counterclockwise boundaries give the same XY center. The point uses DividePointType_p and the current pen color, width, layer, elevation and height. Its Z is the first vertex's Z in the assembled chain.

Radiate_c

Creates one Polyline_i containing a center-to-vertex segment pair for every effective boundary vertex. The object contains separate spoke pairs; it does not add edges between their outer endpoints. The common center is the same XY area centroid used by Centroid_c, with the first collected vertex's Z. Each spoke destination retains its own source vertex Z. The current line pen attributes apply to the result.

Topology and centroid calculations use the exact stored Double coordinates, retaining representable tiny and large-coordinate detail. Source XYZ arrays remain unchanged. Preserving XYZ in these v3 results is intentional; applications should not assume that the generated geometry is flattened to Z=0. Pen elevation and height are separate metadata.

VN_Count_c

Creates one Text_i object per effective vertex, numbered 1, 2, 3 and onward in chain order. Number strings use invariant decimal notation. Numeric numbering does not read alphabet files.

VT_Count_c

Creates one Text_i object per effective vertex using the selected alphabet. Numbering uses complete Unicode text elements, so a supplementary character or a base character with combining accents remains one symbol. After the single-symbol labels come two-symbol labels and then longer labels. A Latin alphabet gives A through Z, AA through ZZ, then AAA; numbering does not repeat A after ZZ.

Label placement

Both counters use an absolute offset of 0.2 drawing units, independently of text scale. At each vertex, the insertion candidates are tried in this order: (+0.2,+0.2), (-0.2,+0.2), (+0.2,-0.2), (-0.2,-0.2). The first point outside the implicitly closed XY boundary is used; a point on its boundary counts as inside. If the first three candidates are inside, the fourth is used. Each label retains that vertex's Z. An offset or text footprint that cannot be represented at the supplied coordinates causes preparation to fail.

DimTextFont_p

Selects the font-table entry used for vertex labels. It is independent of PenFontNumber. The current pen color, width, line style, layer, elevation, height and font style flags are copied to the new label objects.

DimTextScale_p

Supplies the positive finite label text scale. Vertex labels do not multiply it by DimScale_p or PenFontScale. SetAvaxProperty takes its property value ByRef as Object: use an Object variable for a Double or Integer variable, as shown in the example.

Custom alphabet settings

The options directory supplied through the first argument of DirsSet is authoritative when it is nonempty. In that directory, country.opt contains an Integer country code. The matching alpha<code>.opt supplies at least three lines: the alphabet, an area-caption string, and a finite numeric precision value. The caption and precision are retained settings-file fields; they do not change vertex counter text. Later records and trailing blank lines are permitted. Trailing ASCII spaces are removed from the alphabet; leading symbols remain significant.

For example, country.opt containing 37 selects alpha37.opt. Its first three lines may be:

```
AB
Area
2
```

That alphabet produces A, B, AA, AB, BA, BB, AAA and onward. UTF-8, UTF-16 and UTF-32 files with byte-order marks are read as Unicode; files without a byte-order mark use the Windows default encoding. Alphabets must be nonempty, contain distinct text elements without control characters, and contain at most 1,024 symbols. Each settings file is limited to 1 MiB. The control reads these files without overwriting them or changing the process working directory.

If no options directory is configured, the control looks for country.opt in optavx under the current working directory, then under the application directory. If neither file exists, Language selects the built-in Greek or Latin alphabet. Missing, malformed or unreadable files in an explicitly configured directory fail the operation; they do not silently select a different alphabet.

DoAvaxAction

All nine commands can be executed directly. InHandles is a one-based Integer array with an unused zero element and at least one existing handle. All active handles must be valid before processing starts. Duplicate handles are removed in their first-occurrence order, then suitable boundary objects are assembled into one chain. vIn is unused, and the caller's input array, action and input data are unchanged.

On success, the method returns True, fNewHandles=True, errCode=0 and vOut=Nothing. OutHandles is a new one-based array containing the created roots: one curve, centroid point or Radiate polyline, or one handle for each label. The new roots become selected. Existing boundary objects remain in the drawing.

Invalid handles, unsuitable topology, invalid settings and result-preparation failures return False with errCode=2. A stopped, disabled, disposed or edit-locked control returns errCode=1. Failure returns fNewHandles=False, OutHandles={0} and vOut=Nothing. An accepted Polyline chain is completed once before the requested result is committed, as a separate creation. The requested geometry and required capacity are checked before this completion. A completion failure or a callback that takes ownership of another operation prevents the outer action and returns errCode=1. OutHandles describes the requested result, not that earlier Polyline.

Completion, failure and callbacks

The full result, font resources and required handle range are prepared before editing the drawing. Initial validation or preparation failure creates no partial result and preserves source geometry, accepted input and the previous Redo branch. A successful result is one creation batch and one Undo step. With events enabled, NewUserItems supplies the one-based created handles; completion ends the old point-entry construction and establishes the SelectStay_p idle state before creation callbacks. A retained temporary boundary remains available as described above.

Creation and Change callbacks see the complete committed batch. The earlier source-unselection callback, when selected objects are used, runs before that result is committed. An application callback may start a new command, select another object, replace the drawing or end the control; the outer operation does not overwrite that newer state. Exceptions raised by application callbacks propagate with their original identity after the completed result has committed. When a callback throws, inspect the committed drawing or event handles; the method has not returned its normal ByRef output values. A callback's own drawing edit has its own history entry.

Complete VB.NET examples

Pass a started, enabled, edit-unlocked control and a one-based array of suitable source handles. These modules use only public APIs, create new results and keep their source objects. A rectangle can supply the four controls for the curve example and the closed boundary for the label and center examples.

Create a Bezier result from selected controls

```
Option Strict On
Imports System
Imports Avax

Public Module Help_SelectedCurve
    Public Function Create(cad As cAvax, entityHandles As Integer()) As Integer()
        Dim action As AvaxCommand = AvaxCommand.Bezier_C
        Dim input As Object = Nothing, extra As Object = Nothing
        Dim created As Boolean, output As Integer() = Nothing, errorCode As Integer
        If Not cad.DoAvaxAction(entityHandles, input, action, created, output, extra, errorCode) Then
            Throw New InvalidOperationException("Curve could not be created. Error: " &
    errorCode.ToString())
        End If
        If Not created OrElse output Is Nothing OrElse output.Length <> 2 Then
            Throw New InvalidOperationException("Expected one new curve polyline.")
        End If
        Return output
    End Function
End Module
```

Create numeric or alphabetic vertex labels

```
Option Strict On
Imports System
Imports Avax

Public Module Help_VertexLabels
    Public Sub Configure(cad As cAvax, fontNumber As Integer, textScale As Double)
        If Double.IsNaN(textScale) OrElse Double.IsInfinity(textScale) OrElse textScale <= 0 Then
            Throw New ArgumentOutOfRangeException("textScale")
        End If
        Dim propertyName As AvaxProp = AvaxProp.DimTextFont_n
        Dim value As Object = fontNumber
        If Not cad.SetAvaxProperty(propertyName, value) Then
            Throw New InvalidOperationException("Could not set the label font.")
        End If
        propertyName = AvaxProp.DimTextScale_n
        value = textScale
        If Not cad.SetAvaxProperty(propertyName, value) Then
            Throw New InvalidOperationException("Could not set the label scale.")
        End If
    End Sub

    Public Function Create(cad As cAvax, entityHandles As Integer(), numericLabels As Boolean) As
    Integer()
        ' Configure(cad, 1, 2.5R) can be called first to choose font and scale
        Dim action As AvaxCommand = If(numericLabels, AvaxCommand.VN_Count_c, AvaxCommand.VT_Count_c)
        Dim input As Object = Nothing, extra As Object = Nothing
        Dim created As Boolean, output As Integer() = Nothing, errorCode As Integer
        If Not cad.DoAvaxAction(entityHandles, input, action, created, output, extra, errorCode) Then
            Throw New InvalidOperationException("Labels could not be created. Error: " &
    errorCode.ToString())
```

```
        End If
        If Not created OrElse output Is Nothing OrElse output.Length < 2 Then
            Throw New InvalidOperationException("Expected new vertex label handles.")
        End If
        Return output
    End Function
End Module
```

Create an area centroid or one spoke polyline

```
Option Strict On
Imports System
Imports Avax

Public Module Help_VertexCenter
    Public Function Create(cad As cAvax, entityHandles As Integer(), radiate As Boolean) As Integer()
        Dim action As AvaxCommand = If(radiate, AvaxCommand.Radiate_c, AvaxCommand.Centroid_c)
        Dim input As Object = Nothing, extra As Object = Nothing
        Dim created As Boolean, output As Integer() = Nothing, errorCode As Integer
        If Not cad.DoAvaxAction(entityHandles, input, action, created, output, extra, errorCode) Then
            Throw New InvalidOperationException("Center result could not be created. Error: " &
    errorCode.ToString())
        End If
        If Not created OrElse output Is Nothing OrElse output.Length <> 2 Then
            Throw New InvalidOperationException("Expected one centroid point or one spoke polyline.")
        End If
        Return output
    End Function
End Module
```

Use temporary boundary A while object B remains selected

This example takes a started, enabled, edit-unlocked control and two existing nondegenerate axis-aligned rectangles with different, exactly representable XY centers. It creates no source drawing. For example, rectangle A from (0,0) to (4,2) gives a centroid at (2,1), even when rectangle B from (100,100) to (110,120) is selected. MakeSv_c remembers A, SelectItems selects B, and LastSv_c restores the remembered temporary sequence before the ordinary Centroid_c command. The assertions verify the new point belongs to A, that B stays selected, and that both source geometries remain unchanged. Run this example without callbacks that intentionally take ownership of another operation.

```
Option Strict On
Imports System
Imports System.Linq
Imports Avax

Public Module Help_VertexPrecedence
    Public Function Create(cad As cAvax, sourceA As Integer, selectedB As Integer) As Integer()
        If cad Is Nothing Then Throw New ArgumentNullException("cad")
        If sourceA <= 0 OrElse selectedB <= 0 OrElse sourceA = selectedB Then
            Throw New ArgumentException("Supply two different existing rectangle handles.")
        End If
        Dim geometryA = ReadGeometry(cad, sourceA)
        Dim geometryB = ReadGeometry(cad, selectedB)
        Dim centerA = RectangleCenter(geometryA), centerB = RectangleCenter(geometryB)
        If centerA.SequenceEqual(centerB) Then
            Throw New ArgumentException("The rectangles must have different XY centers.")
        End If

        ' Remember rectangle A as the temporary vertex boundary.
        Dim requested = New Integer() {0, sourceA}
        If Not cad.SelectItems(requested) Then
            Throw New InvalidOperationException("Could not select source A.")
        End If
        cad.Command = AvaxCommand.MakeSv_c

        ' Selecting B does not make it the input while A's temporary vertices exist.
        requested = New Integer() {0, selectedB}
        If Not cad.SelectItems(requested) Then
```

```
            Throw New InvalidOperationException("Could not select object B.")
        End If
        cad.Command = AvaxCommand.LastSv c
        Dim before = AllIds(cad)
        cad.Command = AvaxCommand.Centroid c
        Dim created = New Integer() {0}.Concat(AllIds(cad).Skip(1).Except(before.Skip(1))).ToArray()
        If created.Length <> 2 Then
            Throw New InvalidOperationException("Expected one centroid from temporary boundary A.")
        End If

        Dim kinds As AvaxItemType() = Nothing
        Dim properties As Object() = Nothing, values As Object() = Nothing
        Dim output = DirectCast(created.Clone(), Integer())
        If Not cad.GetProperties(output, kinds, properties, values) Then
            Throw New InvalidOperationException("Could not read the centroid.")
        End If
        Dim point = TryCast(values(1), Double())
        If kinds(1) <> AvaxItemType.Point i OrElse point Is Nothing OrElse point.Length <> 4 Then
            Throw New InvalidOperationException("The result must be one centroid point.")
        End If
        If point(1) <> centerA(0) OrElse point(2) <> centerA(1) OrElse
            (point(1) = centerB(0) AndAlso point(2) = centerB(1)) Then
            Throw New InvalidOperationException("The centroid must belong to A, not selected object B.")
        End If
        Dim selected As Integer() = Nothing
        cad.GetSelectedHandlesArr(selected)
        If selected Is Nothing OrElse Not selected.SequenceEqual(New Integer() {0, selectedB}) OrElse
            Not ReadGeometry(cad, sourceA).SequenceEqual(geometryA) OrElse
            Not ReadGeometry(cad, selectedB).SequenceEqual(geometryB) Then
            Throw New InvalidOperationException("Both sources and the unused selection B must remain
    unchanged.")
        End If
        Return created
    End Function

    Private Function AllIds(cad As cAvax) As Integer()
        Dim ids As Integer() = Nothing
        cad.GetAllHandlesArr(ids)
        Return If(ids, New Integer() {0})
    End Function

    Private Function ReadGeometry(cad As cAvax, id As Integer) As Double()
        Dim ids = New Integer() {0, id}, kinds As AvaxItemType() = Nothing
        Dim properties As Object() = Nothing, values As Object() = Nothing
        If Not cad.GetProperties(ids, kinds, properties, values) OrElse
            (kinds(1) <> AvaxItemType.Polyline i AndAlso kinds(1) <> AvaxItemType.Shape i) Then
            Throw New ArgumentException("Supply existing rectangular polyline or shape handles.")
        End If
        Return DirectCast(values(1), Double())
    End Function

    Private Function RectangleCenter(data As Double()) As Double()
        Dim count = (data.Length - 1) \ 3
        If count < 4 Then Throw New ArgumentException("A rectangle needs four corners.")
        Dim xs = Enumerable.Range(0, count).Select(Function(i) data(1 + 3 * i)).ToArray()
        Dim ys = Enumerable.Range(0, count).Select(Function(i) data(2 + 3 * i)).ToArray()
        If xs.Min() >= xs.Max() OrElse ys.Min() >= ys.Max() Then
            Throw New ArgumentException("Supply nondegenerate axis-aligned rectangles.")
        End If
        Return New Double() {xs.Min() / 2 + xs.Max() / 2, ys.Min() / 2 + ys.Max() / 2}
    End Function
End Module
```

MakeSv_c: create temporary vertices from one object

Select exactly one line, polyline, shape, curve or ellipse and assign MakeSv_c to Command. Imported analytic polylines can combine lines and arcs; extraction follows their original traversal, including negative bulges and a curved closing segment. Their topology survives native AVAX save/open and affine transformations. The control copies its ordered boundary into the temporary vertex sequence, clears the source selection once and enters SelectV_c so more vertices can be added. Ellipses use sixty chords; analytic arcs and cubic curves use their sampled boundaries. The original object remains unchanged. One exact closing XYZ marker is removed; distinct short edges, differing Z values and repeated interior controls remain. LastSv_c can restore the sequence later.

The source is checked before accepted input is consumed. Empty or multiple selections, unsupported objects, ordinary blocks and disconnected polyline pairs report an error without changing the current selection or accepted input. A successful extraction adds no drawing object or Undo entry and preserves Redo. The control must be started, enabled and not paused. LockEdit permits this temporary selection operation. If the source-unselection event handler changes the drawing, selection, command or lifecycle, its new state takes ownership and the prepared vertices are not published. Exceptions from that handler propagate.

```
Option Strict On
Imports Avax
Imports System

Public Module Help_MakeVertices
    Public Function Extract(cad As cAvax, sourceHandle As Integer) As Double()()
        If cad Is Nothing Then Throw New ArgumentNullException(NameOf(cad))
        Dim sourceHandles As Integer() = {0, sourceHandle}
        cad.SelectItems(sourceHandles)
        cad.Command = AvaxCommand.MakeSv_c
        Dim x As Double() = Nothing, y As Double() = Nothing, z As Double() = Nothing
        Dim count = cad.GetSelectedPoints(x, y, z)
        If cad.Command <> AvaxCommand.SelectV_c OrElse count < 2 Then
            Throw New InvalidOperationException("The selected object cannot supply temporary vertices.")
        End If
        Dim result(count - 1)() As Double
        For i As Integer = 1 To count
            result(i - 1) = New Double() {x(i), y(i), z(i)}
        Next
        Return result
    End Function
End Module
```

Interactive Command-based creation publishes NewUserItems as described here. A direct DoAvaxAction call is an automatic operation: use OutHandles and fNewHandles for its result; it publishes Change without NewUserItems. Previously accepted user construction completed before that action retains its own notification.

PreserveSelectionAppearance

API reference

PreserveSelectionAppearance

Selection commands and type filters

API reference

SelectItems

Select drawing roots using a one-based Integer array; index zero is reserved. Nothing or a one-slot array expands to all live root handles and returns that array through ByRef. A truly empty array clears selection. Duplicate and missing handles are ignored. The expanded array is ordered by handle even after objects have been deleted. The typed overload selects only the specified AvaxItemType, independently of the interactive filter. The method returns True when at least one matching object is selected and False otherwise.

Selection state

isSelected reports whether there are selected roots. IsSelectedItem checks one handle. MaxSelectedItems returns the selected root count. Queries on an unstarted, disabled, ended or disposed control return neutral values. Explicit SelectItems and SetSelectFilter return False on inactive controls and preserve their ByRef inputs.

Select_c, SelectAdd_c and SelectSub_c

Select_c performs ordinary point selection. With SelectContinuousOn_p=False, a hit replaces the previous selection; with True, hits accumulate. SelectAdd_c adds a hit to the existing selection, and hitting an already selected object leaves it selected. SelectSub_c removes a hit and never adds an unselected object; selection grips cannot intercept this removal. An unchanged Add/Sub click does not publish a false selection change.

None_c, Cancel_c and Cancel_Keep_Select_c

None_c leaves no active drawing command. Cancel_c discards current input and clears selection. Cancel_Keep_Select_c discards current input while retaining selection. During an active grip edit the first cancellation abandons that edit and preserves its selection.

WinSel_c and InWinSel_c

Specify two opposite corners using clicks, a drag, or SetCoordinate with Double coordinates. WinSel_c includes objects whose visible geometry crosses the rectangle; InWinSel_c includes only fully contained objects. Add/Sub selection mode is retained across the window operation. Shift or Control, or SelectContinuousOn_p=True, makes an ordinary selection window additive. Alt alone does not. Rulers, screen factors and large coordinate origins do not change the geometric meaning of the rectangle.

Select_All_c and Unselect_c

Select_All_c selects every eligible root permitted by the interactive type mask and selectable layer state. Unselect_c clears selection. These operations do not change drawing geometry.

SelectEnabled_p and SelectContinuousOn_p

SelectEnabled_p controls automatic mouse selection, including grip interaction. Disabling it abandons an unfinished mouse selection; explicit SelectItems remains available for host-controlled selection. Pausing the control also abandons unfinished mouse selection. SelectContinuousOn_p preserves existing selection on ordinary object hits, explicit selection windows and empty clicks. Shift or Control makes ordinary object and window selection additive regardless of that setting; Alt alone does not. Repeated additive hits retain the selected object. In the R3 application, an ordinary empty click clears the selection unless continuous selection is enabled; this also applies to Add mode.

SetSelectFilter

Set an Integer bit mask before interactive selection, including from BeforeClick. Zero accepts every type. Combine bits with Or. The legacy curve bit is 256 even though Curve_i has enum value 11; do not derive that bit by shifting the enum value. Explicit SelectItems uses its own optional type parameter and ignores this interactive mask.

Root type	Mask bit
Point	1
Line	2

Root type	Mask bit
Polyline	4
Block	8
Ellipse / circle / arc	16
Text	32
Shape	64
External lines	128
Curve	256
Document	4096
Native dimension	8192
Native hatch	16384

SelectChange

The event runs after the selection has been committed, with the control as Sender. API selection reports its last valid selected handle and reports zero when it clears the selection. A point hit reports the selected hit or a remaining selected handle; an empty selection reports zero. Repeating Select_All_c or Unselect_c without a change sends no extra event. The host may read selection state or make a newer selection in the handler; the previous operation does not replay old handles afterward. PauseAvaxEvents suppresses notifications while preserving state changes. Mutating the event argument does not mutate the selection.

Complete VB.NET examples

Select explicit roots

```
Option Strict On
Imports Avax

Public Module Help_SelectExplicit
    Public Function SelectRoots(cad As cAvax, entityHandles As Integer()) As Boolean
        Return cad.SelectItems(entityHandles)
    End Function
End Module
```

Read the selected count

```
Option Strict On
Imports Avax

Public Module Help_SelectionState
    Public Function Count(cad As cAvax) As Integer
        Return cad.MaxSelectedItems()
    End Function
End Module
```

Select lines and circles with the interactive mask

```
Option Strict On
Imports Avax

Public Module Help_SelectFilter
    Public Sub Configure(cad As cAvax)
        Dim mask As Integer = 2 Or 16
        cad.SetSelectFilter(mask)
        cad.Command = AvaxCommand.Select_c
    End Sub
End Module
```

AvaxProp Member Reference

API reference

Use SetAvaxProperty and GetAvaxProperty with the following AvaxProp values.

SelectRegionOn_p

Shows or hides the selection frame and its grips without changing which objects are selected. Boolean; default True.

RegionBoundingBoxOn_p

Shows the frame connecting selection grips. Boolean; default True. When region dragging is disabled, a bounding box remains visible as in R2.

RegionDragDisable_p

Prevents dragging selection grips. Boolean; default False. When True, the grips are hidden and the bounding box remains visible.

RegionMarkWidth_p

Sets the selection-grip width in logical pixels. Integer; default 5. V3 keeps a minimum visible width of 5 for usable grips, including old zero-width settings. Larger explicit widths are honored. DPI scaling applies.

RegionRGBColor_p

Sets the selection frame and grip color as an Integer COLORREF (red + 256*green + 65536*blue). V3 defaults to yellow (&H00FFFF). A color matching the background is displayed with contrast.

SelectTextDitherOn_p

Enables the hatch overlay on selected text and formatted RTF documents. Boolean; default True. Unselected text is unaffected.

TransparentSelectBoxOn_p

Enables the translucent fill inside a selection window. Boolean; default True. TouchScreenOn_p temporarily makes this property read False and suppresses the fill; leaving touch mode restores the stored preference.

TransparentSelectBoxRGB_p

Sets the selection-window fill color as an Integer COLORREF. V3 defaults to &HFFA000. Both window and crossing selection use this fill color; their outlines still distinguish their selection modes.

TransparentSelectBoxOpacityPercentage_p

Sets selection-window fill opacity as a Double percentage from 0 to 100. V3 defaults to 11. Zero hides the fill; 100 is opaque. Values outside the range reset to the default. Nonfinite input fails without changing the current value.

Behavior

These settings control screen decorations; they do not modify drawing geometry or cancel the current drawing command. Turning off the selection region leaves the selection itself intact. The frame includes a margin, corner grips for a line, a move grip and a rotation grip.

In touch mode, region marks are enlarged by 12 on systems reporting touch contacts, or by 4 otherwise, as in R2. V3 also applies display DPI scaling. The stored RegionMarkWidth_p value is unchanged.

The V3 defaults use a yellow selection frame and a light blue window fill at 11 percent opacity. R2 used a blue frame and cyan fill at 25 percent; applications can explicitly choose their own colors and opacity.

SaveAvaxPropertiesToXML and LoadAvaxPropertiesFromXML retain these settings. As in R2, saving in touch mode writes the effective False value of TransparentSelectBoxOn_p. The opacity is Double rather than Single; fractional values retain Double precision.

VB.NET example

Call Configure with an initialized AVAX control. The example sets all nine properties and returns False if a setting is rejected.

```
Option Strict On
Imports Avax

Public Module Help_SelectionProperties
    Public Function Configure(cad As cAvax) As Boolean
        Return cad.SetAvaxProperty(AvaxProp.SelectRegionOn, True) AndAlso
            cad.SetAvaxProperty(AvaxProp.RegionBoundingBoxOn, True) AndAlso
            cad.SetAvaxProperty(AvaxProp.RegionDragDisable, False) AndAlso
            cad.SetAvaxProperty(AvaxProp.RegionMarkWidth, 8) AndAlso
            cad.SetAvaxProperty(AvaxProp.RegionRGBColor, &H00FFFF) AndAlso
            cad.SetAvaxProperty(AvaxProp.SelectTextDitherOn, True) AndAlso
            cad.SetAvaxProperty(AvaxProp.TransparentSelectBoxOn, True) AndAlso
            cad.SetAvaxProperty(AvaxProp.TransparentSelectBoxRGB, &HFFA000) AndAlso
            cad.SetAvaxProperty(AvaxProp.TransparentSelectBoxOpacityPercentage, 25.5)
    End Function
End Module
```

Idle selection and spatial queries

API reference

SelectStay_p

Controls automatic object selection when a command finishes or the control is idle. True enables the selection pointer and grips; False leaves normal drawing-pointer behavior until the host explicitly starts a selection command. Assigning the property does not discard an accepted construction point or clear the selected handles. Explicit Select_c, SelectAdd_c and SelectSub_c remain available with SelectStay_p=False. The corrected GetAvaxStatus reports selecton=2 for active automatic selection and zero for idle selection disabled; the original R2 wrapper incorrectly returned its scale status in that output.

Starting a new drawing command

Starting Line_c, Point_c, Polyline_c, Circle_c or Rectangle_c clears the previous selection, as in R2, before accepting the first construction point. One SelectChange event with LastSelectedHandle=0 reports an actual cleared selection; an already empty selection produces no event. SelectStay_p controls automatic idle selection, not retention when starting a new drawing tool. Cancel_Keep_Select_c preserves the selection that currently exists; it does not restore an earlier selection removed when drawing began. Modification tools such as Move_c, Copy_c, Rotate_c, Scale_c and Mirror_c keep their selected source objects. A disabled, paused or edit-locked drawing tool does not clear selection.

SelectInsideShapeOn_p

Controls whether a point hit inside a filled shape can select it. The R2 default is False. True enables interior hits while the visible outline remains selectable in either setting. This point-pick policy does not remove filled geometry from explicit area-query results.

GetHandlesUnderArea

Find every visible root whose displayed geometry intersects a Double XY rectangle, including when that rectangle is outside the current view. Opposite corners may be reversed; caller coordinates are not rewritten. Bounding-box overlap alone is insufficient for outlines or curves. Every successful call returns a new one-based Integer array, with index zero reserved. A count of zero for a valid empty area returns an array containing only zero. A busy, degenerate, nonfinite or overflowing request returns -1 and preserves the caller's array. An inactive control returns zero without replacing the caller's array.

The overload with fzSort=True orders roots by ascending z-order, then handle. Otherwise roots are ordered by handle. Counts and handles use Integer; coordinates use Double in v3. Tiny representable areas are not enlarged using a tolerance proportional to their distance from the origin. Queries preserve drawing data, selection, settings and Undo history.

GetHandlesUnderSelectCursor

Find all roots under the current visible selection pointer, using its actual pixel box. It is distinct from choosing one nearest object. Both overloads follow the same count, fresh-array and sorting conventions as GetHandlesUnderArea. An unavailable pointer returns -1 without replacing output: examples include a pointer outside the canvas, an idle control with SelectStay_p=False, a hidden selection pointer, touch mode, pause, or an active construction/drag. An inactive control returns zero and preserves output. Explicitly enabling a selection command makes the pointer available again.

Layers, blocks and filters

Hidden geometry is excluded. Locked or nonprinting geometry can still be read by spatial queries. A block root is returned once even if several visible children intersect; hidden children do not create hits. SetSelectFilter applies its interactive root-type mask to these queries. Display-sized text and point symbols are tested at their current displayed size. Filled interiors are actual geometry; a hollow outline does not fill its interior merely because its bounds overlap.

Queries during ZoomChange

A completed zoom-window operation permits the explicit area query from its ZoomChange event, as in the original manual example. Starting a different command in that callback revokes this completed-view exception. A query does not end the repeatable Zoom command.

Complete VB.NET examples

Configure automatic idle selection

```
Option Strict On
Imports Avax

Public Module Help_IdleSelection
    Public Sub Configure(cad As cAvax, enabled As Boolean)
        cad.SetAvaxProperty(AvaxProp.SelectStay_n, (enabled))
    End Sub
End Module
```

Read a Double-coordinate area

```
Option Strict On
Imports System
Imports Avax

Public Module Help_QueryArea
    Public Function Read(cad As cAvax, xMin As Double, yMin As Double,
                         xMax As Double, yMax As Double) As Integer()
        Dim entityHandles As Integer() = Nothing
        Dim count As Integer = cad.GetHandlesUnderArea(entityHandles,
            xMin, yMin, xMax, yMax, True)
        If count < 0 Then Throw New InvalidOperationException("The area query is unavailable.")
        Return entityHandles
    End Function
End Module
```

Keep cursor-query availability separate from an empty result

```
Option Strict On
Imports Avax

Public Module Help_QueryCursor
    Public Function Read(cad As cAvax, ByRef entityHandles As Integer()) As Integer
        Return cad.GetHandlesUnderSelectCursor(entityHandles, True)
    End Function
End Module
```

Precise stroke width

API reference

These v3 methods read and change a Double pen-width attribute. PenWidth, WidthPen_p and ChangeWidth retain their Integer contracts for existing applications. A stroke width is a pen-unit multiplier, not a model-space polyline ribbon width. PixelWidth_MM_p and ShowRealPixelWidth_p control conversion to displayed or printed width. Positive screen strokes have a minimum display width of one pixel.

GetStrokeWidth

ChangeStrokeWidth

CreateSVGHatchPattern

API reference

CreateSVGHatchPattern

Three-point dimension command

API reference

Dim3p_c

Command value 221 creates a linear dimension from three accepted points: the first measured endpoint, the second measured endpoint, and a placement point. The third point selects the side of the measured line and the perpendicular distance to the dimension line. It is not a third measured vertex. The command uses these picks rather than an existing object selection.

Coordinates and distances use Double in v3. The measured value is the XY endpoint distance, formatted with the current dimension factor, units, precision and suffix. Component sizes, font and pen attributes use the current dimension settings. See dimension formatting and dimension sizes.

The two endpoints must have different XY positions. A placement on their line cannot produce a positive offset. Invalid final input leaves the earlier accepted reference points available for another placement. Geometry, resources and capacity are checked before the complete set of line, mark, extension and text components is inserted. Successful completion returns to the idle selection mode controlled by SelectStay_p. The complete result is one Undo/Redo operation.

V3 retains each endpoint's Z in the dimension-line geometry and uses the first endpoint Z for the text. This extends the older planar dimension output. The placement point's Z does not change the measured XY distance or perpendicular offset. Pen elevation and height remain separate object attributes.

Call the control on its owning UI thread after StartAvax. Normal interactive edit/lifecycle guards apply. Ortho and snap settings affect accepted coordinates. Command has no Boolean return value; SetCoordinate reports whether its point was accepted. For a direct Boolean result and returned component handles, use Add_Dim with an explicit positive offset and direction point. For an ordered vertex sequence, see V_Dim_c.

VB.NET example

The supplied control must already be started. This example dimensions the segment from (0,5) to (10,5), placing the dimension line at Y=1. It explicitly turns off construction constraints so the supplied placement is accepted as written.

```
Option Strict On
Imports Avax

Public Module Help_ThreePointDimension
    Public Function Draw(cad As cAvax) As Boolean
        cad.Command = AvaxCommand.Cancel_c
        cad.Command = AvaxCommand.Ortho_Off_c
        cad.SetAvaxProperty(AvaxProp.SnapOn_p, False)
        cad.SetAvaxProperty(AvaxProp.AutoOSnap_p, 0)
        cad.SetAvaxProperty(AvaxProp.DimTextAbove_p, True)
        cad.Command = AvaxCommand.Dim3p_c
        Dim x As Double = 0, y As Double = 5, z As Double = 0
        If Not cad.SetCoordinate(x, y, z) Then Return False
        x = 10 : y = 5
        If Not cad.SetCoordinate(x, y, z) Then Return False
        x = 5 : y = 1
        Return cad.SetCoordinate(x, y, z)
    End Function
End Module
```

AvaxProp Member Reference

API reference

Use SetAvaxProperty and GetAvaxProperty with these AvaxProp values.

RotateStay_p

Keeps the original selected objects and creates rotated copies when True. Boolean; default False. Used by the interactive Rotate command and DoAvaxAction.

ScaleStay_p

Keeps the original selected objects and creates scaled copies when True. Boolean; default False. Used by the interactive Scale command and DoAvaxAction.

MirrorStay_p

Keeps the original selected objects and creates mirrored copies when True. Boolean; default True, as in R2. Used by the interactive Mirror command and DoAvaxAction.

ScaleEqual_p

Uses the same factor on both axes for interactive Scale when True (the default). When False, the absolute X and Y distances from the base point supply separate factors. Explicit ScaleItems and DoAvaxAction factors are unaffected.

ScaleSensitivity_p

Sets the Double divisor for uniform interactive scaling: scale factor = distance from base point / sensitivity. Default 1. Zero resets to 1. Finite negative values are supported and scale through the base point. Nonfinite input fails without changing the setting. Ignored when ScaleEqual_p is False.

Commands and methods

For Rotate, Scale and Mirror, the first point defines the pivot or first mirror-axis point. The second point specifies the rotation direction, scale distance or second mirror-axis point. Previewing does not alter the original objects or raise BeforeRotate, BeforeScale or BeforeMirror. The corresponding event runs at commit and can cancel the operation. Each completed transform or set of copies is one Undo operation.

The stay properties apply to interactive commands and DoAvaxAction. The explicit fRotateStay, fScaleStay and fMirrorStay arguments of RotateItems, ScaleItems and MirrorItems control those method calls independently. Those calls do not change the stored properties.

For uniform Scale, moving 5 drawing units from the base with sensitivity 2.5 gives a factor of 2 on both axes. With sensitivity -2.5, the factor is -2. With ScaleEqual_p=False, X and Y distances of 3 and 4 give separate factors of 3 and 4; sensitivity is not used. Ortho supplies the nonzero factor to both axes when one component is zero. A degenerate or unrepresentable transform is rejected instead of altering the drawing.

These properties can be changed while an interactive command is waiting for its next point. SaveAvaxPropertiesToXML and LoadAvaxPropertiesFromXML preserve all five settings. ScaleSensitivity_p uses Double precision in v3 instead of Single.

VB.NET example

Call Configure with an initialized control to keep original objects for all three commands and use uniform scaling with a sensitivity of 2.5.

```
Option Strict On
Imports Avax

Public Module Help_TransformProperties
    Public Function Configure(cad As cAvax) As Boolean
        Return cad.SetAvaxProperty(AvaxProp.RotateStay_p, True) AndAlso
            cad.SetAvaxProperty(AvaxProp.ScaleStay_p, True) AndAlso
            cad.SetAvaxProperty(AvaxProp.MirrorStay_p, True) AndAlso
            cad.SetAvaxProperty(AvaxProp.ScaleEqual_p, True) AndAlso
```

```
            cad.SetAvaxProperty(AvaxProp.ScaleSensitivity, 2.5)
    End Function
End Module
```

Triangle area annotations

API reference

Area_3V_c

Command value 520 adds six Text_i objects describing the three sides, semiperimeter and area of one triangle. It preserves the source objects. It does not construct a new triangle boundary; V_Triangle_c is a separate command. Coordinates and text scales use Double; handles, counts and command values use Integer. Call the control on its owning UI thread after StartAvax.

Six annotation objects

The first three alphabet symbols identify successive edges from vertex 0 to 1, 1 to 2, and 2 to 0. They are side labels, not names placed at the vertices or labels of opposite sides. With symbols A, B and C, the created objects have this order:

Order	Text	Insertion
1	A	XY midpoint of edge 0 to 1
2	B	XY midpoint of edge 1 to 2
3	C	XY midpoint of edge 2 to 0
4	A=<side 0>, B=<side 1>, C=<side 2>	(MaxX+0.2, MaxY)
5	T=(A+B+C)/2=<semiperimeter>	(MaxX+0.2, MaxY-0.36)
6	<caption>=sqrt(T*(T-A)*(T-B)*(T-C))=<area>	(MaxX+0.2, MaxY-0.72)

MaxX and MaxY are the triangle's maximum input coordinates. Every insertion and text object has Z=0 and no rotation. The 0.2 horizontal offset and 0.36 vertical spacing are absolute drawing distances; text scale does not alter them. Source XYZ coordinates remain unchanged. Pen elevation and height are separate metadata.

For the ordered vertices (0,0), (3,0), (3,4), the side lengths are 3, 4 and 5, the semiperimeter is 6 and the area is 6. With alphabet ABC, caption Area and precision 2, the six text values are:

```
A
B
C
A=3.00  B=4.00  C=5.00
T=(A+B+C)/2=6.00
Area=sqrt(T*(T-A)*(T-B)*(T-C))=6.00
```

The three side insertions are (1.5,0,0), (3,2,0) and (1.5,2,0). The summary insertions are (3.2,4,0), (3.2,3.64,0) and (3.2,3.28,0). Changing the input Z values does not change these annotation positions or the XY measurements.

Triangle input and Command

Ordinary Command uses a nonempty retained temporary vertex stream before selected source objects. That stream must contain exactly three raw records. A stream of one, two or four records fails without falling back to an otherwise valid selected triangle. In particular, temporary A,B,C,A is four records and is rejected. A valid temporary stream remains available after completion; unrelated selected objects remain selected.

With no temporary stream, suitable selected source paths are assembled into one connected XYZ boundary. One exact repeated terminal closing vertex is removed from extracted source input, then exactly three vertices are required. Thus a selected closed triangle with an exact A,B,C,A marker is accepted. Shared endpoints must agree exactly in XYZ. Disconnected paths, ambiguous branching and duplicated edges across source paths are rejected. The selected vertex boundary rules describe extraction; a sampled curved boundary is not automatically a three-vertex triangle.

Selected-source execution clears source selection only after the complete result is prepared. The six ordinary-command results are not selected. The source objects and their geometry remain in the drawing. Assigning Command has no public success acknowledgement or return value; use DoAvaxAction when the application needs explicit success and the new handles.

Exactly three raw temporary records may have collinear or repeated XY positions. These cases still produce six annotations with numeric zero area. This command does not impose the nonzero-area requirement of V_Area_c. Annotations use XY side lengths and area even when the three source vertices have different Z coordinates.

DimTextFont_p and DimTextScale_p

The six annotations use the dimension font-table entry selected by DimTextFont_p, whose index must be from 1 through 256, and the positive finite text scale from DimTextScale_p. They do not use PenFontNumber or PenFontScale and do not multiply the scale by DimScale_p. Current font style flags, the selected font entry's name and charset, and current pen color, width, line style, layer, elevation and height apply to all six new text objects.

DirsSet and alphabet settings

The first DirsSet argument selects the control's options directory. A nonempty configured directory is authoritative. Its country.opt contains a signed Integer country code on the first line, with optional blank trailing lines. The code chooses alpha<code>.opt in the same directory. Missing, malformed or unreadable configured files fail before any annotation is committed; they do not silently select another alphabet.

The alphabet file supplies at least three records: the alphabet, the area caption, and the numeric precision. Only the first three complete Unicode text elements are used as side symbols. Trailing ASCII spaces are removed from the alphabet and caption; leading content remains significant. The caption may be empty but must not contain control characters. Extra alphabet symbols and later file records are ignored. The alphabet must contain at least three distinct usable symbols. Precision must be a finite integer-valued number from 0 through 15 for this command. VN_Count_c and VT_Count_c retain their separate policy for the unused precision field, including finite fractional values.

For example, country.opt containing 37 selects alpha37.opt. These first three records choose symbols X, Y and Z, the caption Plot and two decimal places:

```
XYZ
Plot
2
```

UTF-8, UTF-16 and UTF-32 files with byte-order marks are decoded as Unicode. Files without a byte-order mark use the runtime's default encoding. The shared alphabet reader rejects control characters, duplicate symbols and alphabets larger than 1,024 symbols, and limits each settings file to 1 MiB. It does not overwrite the files or change the process working directory.

Without a configured options directory, the control looks for country.opt in optavx under the current working directory, then under the application directory. When neither exists, Language chooses the built-in Greek or Latin alphabet, the caption is Area for either language, and precision is 2.

Numbers and formatting

Side lengths, semiperimeter and area are calculated from the stored Double XY coordinates without narrowing to Single. The area comes from the absolute triangle cross product; the displayed Heron expression explains the result rather than controlling the numeric calculation. A zero-area triangle is permitted, and a positive exact area may round to zero at the Double limit.

Public control coordinates remain Double. Existing R2 vertex-job clients keep their packed Single wire records of 8, 12 or 24 bytes; the file format and its coordinate precision are unchanged. Job 520 promotes the decoded coordinates into the same Double geometry and calculation. It requires exactly three raw records and does not remove a fourth closing record. Three records with repeated or collinear XY positions are allowed and produce numeric zero area.

Numbers use invariant fixed decimal places and MidpointRounding.AwayFromZero. There are no thousands separators, and zero is printed numerically, such as 0.00 at precision 2. These rules deliberately repair old Single narrowing, culture-dependent text and blank zero values. The readable sqrt(...) expression replaces the old vector-font control-glyph encoding.

All final measurements, midpoints, offsets and text bounds must be finite and representable. A true numeric overflow or an unrepresentable fixed offset rejects the complete operation. The command does not enlarge an offset or create a partial annotation batch to work around that failure.

DoAvaxAction

Direct execution uses only the supplied one-based Integer handle array, with an unused zero element and at least one existing source handle. Duplicate handles are considered once in first-occurrence order. Unrelated temporary vertices and current selection do not replace the explicit request. vIn is unused; the caller's handle array, input data and action value are unchanged.

Successful direct execution returns True, fNewHandles=True, errCode=0 and vOut=Nothing. OutHandles is a one-based array containing exactly the six new text handles in annotation order. The six new objects become selected. Source objects are preserved. Direct execution requires no parameter or measurement dialog.

Direct rejection returns False, fNewHandles=False, OutHandles={0} and vOut=Nothing. Invalid handles, wrong input count, unsuitable geometry, settings, capacity and result-preparation failures use errCode=2. A stopped, disabled, disposed or edit-locked control uses

errCode=1. A direct creation may run while PauseAvax is enabled if no accepted Polyline chain needs completion. A paused pending chain cannot be completed and returns errCode=1. Ordinary Command creation is unavailable while paused or edit-locked.

Preparation, history and callbacks

The command prepares all six strings, geometry, font resources and required handles before clearing source selection or committing the result. Initial failure preserves source geometry, selection, temporary input and the existing Redo branch. A completed six-text result is one creation batch, one NewUserItems notification, one Change notification and one Undo step when events and recording are enabled. Redo restores the complete batch.

When a direct request must first complete accepted Polyline input, the requested annotations and required capacity are checked before that separate Polyline creation. The explicit request is rebuilt and revalidated after the completion callbacks. An application callback that takes ownership of another command, selection, drawing or control lifecycle prevents stale outer completion and uses errCode=1 on normal direct return. The already completed Polyline and callback edits retain their own history entries.

Source-unselection callbacks run before ordinary selected-source annotation creation; a takeover or exception there prevents stale text creation. NewUserItems and Change callbacks for the result see the complete six-text batch. Callback exceptions propagate with their original identity. If a callback throws after creation, the batch remains committed; the public method has not returned its normal ByRef output values. A callback's newer command, selection or document remains owned by that callback.

For file job 520, choose a command output path distinct from country.opt and the selected alpha<country>.opt. An alias to either settings file returns False and preserves all existing files, including when the path differs only by letter case.

Complete VB.NET examples

Pass a started, enabled, edit-unlocked control and suitable existing source handles in a one-based array. The first example returns the six created handles. The second invokes ordinary Command; a retained temporary stream may take precedence over its selected handles, and the procedure does not report success.

Create six annotations through the direct method

```
Option Strict On
Imports System
Imports Avax

Public Module Help_TriangleArea
    Public Function Create(cad As cAvax, entityHandles As Integer()) As Integer()
        If cad Is Nothing Then Throw New ArgumentNullException("cad")
        Dim action As AvaxCommand = AvaxCommand.Area_3V_c
        Dim input As Object = Nothing, extra As Object = Nothing
        Dim created As Boolean, output As Integer() = Nothing, errorCode As Integer
        If Not cad.DoAvaxAction(entityHandles, input, action, created, output, extra, errorCode) Then
            Throw New InvalidOperationException("Triangle annotations failed. Error: " &
    errorCode.ToString())
        End If
        If Not created OrElse errorCode <> 0 OrElse extra IsNot Nothing OrElse
           output Is Nothing OrElse output.Length <> 7 OrElse output(0) <> 0 Then
            Throw New InvalidOperationException("Expected six new triangle text handles.")
        End If
        Return output
    End Function
End Module
```

Invoke ordinary selected or retained-temporary annotations

```
Option Strict On
Imports System
Imports Avax

Public Module Help_TriangleAreaCommand
    Public Sub Execute(cad As cAvax, entityHandles As Integer())
        If cad Is Nothing Then Throw New ArgumentNullException("cad")
        If entityHandles Is Nothing OrElse entityHandles.Length < 2 Then
            Throw New ArgumentException("Supply a one-based array of existing source handles.",
    "entityHandles")
        End If
        If Not cad.SelectItems(entityHandles) Then
```

```
            Throw New InvalidOperationException("No source object could be selected.")
        End If
        ' Retained temporary vertices may take precedence over this selection.
        ' Use the direct method when explicit handles must own the request.
        cad.Command = AvaxCommand.Area_3V_c
    End Sub
End Module
```

Interactive Command-based creation publishes NewUserItems as described here. A direct DoAvaxAction call is an automatic operation: use OutHandles and fNewHandles for its result; it publishes Change without NewUserItems. Previously accepted user construction completed before that action retains its own notification.

Drawing updates and refresh

API reference

BeginUpdate

EndUpdate

Refresh

Regen

RegenControlsZoomBitmap

avVertex.cVertex Member Reference

API reference

Jobs 506 through 510 turn control vertices into connected polyline chords. Use this interface with applications that already exchange R2 vertex job files. The direct Add methods and control commands retain XYZ coordinates; these legacy file jobs construct planar curves at Z=0.

Request

Write five text lines: job number, record size (8, 12 or 24 bytes), total input byte count, packed input filename, and auxiliary information. The binary records contain IEEE Singles. Only the first XY pair of each record is a control vertex; stored Z and the additional endpoint of a 24-byte record do not influence the curve. All numeric fields must be finite.

Job	Curve	Sampling and endpoint conditions
506	Interpolating cubic spline 1	Chord-length parameters, zero endpoint derivatives, 15 subdivisions per nonzero guide interval.
507	Interpolating cubic spline 2	Chord-length parameters, natural endpoint conditions, 15 subdivisions per nonzero guide interval.
508	Open B-spline	Clamped quadratic spline at 20 parameter samples, producing 19 chords. Two guides produce a straight line with the same sample count.
509	Bezier curve	Degree is guide count minus one. N guides produce 10N samples and 10N-1 chords, including both endpoints.
510	Closed B-spline	Periodic cubic with 10 chords per guide span. Up to 12 guides are refined with intervening midpoints, producing 20N chords; more than 12 produce 10N chords. The last endpoint exactly matches the first.

Supply 2 to 3200 raw records. For job 510, at least three guides must remain after removing an optional repeated terminal XY closure marker. The 3200-record limit includes that marker. Consecutive duplicate XY guides do not add intervals to interpolating cubic splines. Repeated guides in Bezier and B-spline curves retain their weighting effect.

Reply and geometry

DoVertex returns True when it processes the request. A successful four-line reply replaces the job file: 1, 2, output byte count, unchanged auxiliary information. The input binary file is replaced by complete 24-byte chord records: X1, Y1, 0, X2, Y2, 0, each a Single. Consecutive records share endpoints. Retain a copy of the source file if the application still needs the original guides.

Too few or more than 3200 guides produce the reply 0, 0, original input byte count, unchanged auxiliary information. The Boolean return is True, but the binary input is preserved. Check the first reply field as well as the Boolean result.

Calculations use Double precision and the same reviewed curve construction as the control. Final binary coordinates retain Single precision for file compatibility. Small sampled chords can round to zero length in that format. A cubic overshoot outside the finite Single range, malformed input, file locks or publication errors return False and preserve both files. In v3, Bezier endpoints are not omitted, periodic curves close exactly, and the open B-spline is not restricted by the old fixed 20-slot working array.

VB.NET example

This complete example creates a quadratic Bezier from (0,0), (29,58), (58,0), returning its 29 chords as Double values read from packed Single output. It removes its temporary files in Finally. No control instance or StartAvax call is required.

```
Option Strict On
Imports Avax
Imports System
Imports System.IO
Imports System.Text

Public Module Help_VertexCurves
    Public Function CreateBezier(workingDirectory As String) As Double()
        Dim folder = Path.Combine(workingDirectory, "avax-curve-" & Guid.NewGuid().ToString("N"))
        Directory.CreateDirectory(folder)
        Try
            Dim data = Path.Combine(folder, "vertices.bin")
```

```
            Dim job = Path.Combine(folder, "job.txt")
            Using output = File.Create(data), writer As New BinaryWriter(output)
                For Each coordinate As Single In New Single() {0, 0, 29, 58, 58, 0}
                    writer.Write(coordinate)
                Next
            End Using
            File.WriteAllLines(job, {"509", "8", "24", data, ""}, Encoding.Default)
            Dim processor As New avVertex.cVertex()
            If Not processor.DoVertex(job) Then Throw New IOException("The curve job could not be
    processed.")
            Dim reply = File.ReadAllLines(job, Encoding.Default)
            If reply(0) <> "1" Then Throw New InvalidOperationException("The guides did not produce a
    curve.")
            Dim bytes = File.ReadAllBytes(data)
            Dim values(bytes.Length \ 4 - 1) As Double
            For i = 0 To values.Length - 1
                values(i) = BitConverter.ToSingle(bytes, i * 4)
            Next
            Return values
        Finally
            Directory.Delete(folder, True)
        End Try
    End Function
End Module
```

Vertex dimension commands

API reference

V_Dim_c

Command value 514 adds linear dimensions for one ordered vertex sequence. Two vertices produce one dimension. Three or more vertices produce a dimension for every successive edge and the final-to-first edge, even when the source polyline is open. Each dimension consists of ordinary Line_i, Point_i, Ellipse_i and Text_i roots, according to the selected options. The source objects remain in the drawing. Coordinates and dimension sizes use Double; handles, command values and error codes use Integer. Call the control on its owning UI thread after StartAvax.

This reference describes V_Dim_c. Dim_c, Add_Dim and the native dimension methods have their own input and storage contracts; V_Dim_c does not create one native dimension object for each edge.

Source vertices and boundary rules

All source XYZ coordinates must be finite. A sequence needs at least two vertices, and every dimensioned edge must have nonzero XY length. Z does not contribute to a dimension. An edge with identical XY endpoints is invalid even when its endpoint Z values differ. One such edge rejects the complete operation; a partial set of dimensions is never added.

Three or more vertices are a closed walk for this command. Collinear and crossing walks are allowed; V_Dim_c does not impose the simple, nonzero-area polygon requirement of V_Area_c or Centroid_c. Distinct tiny edges are retained rather than removed by a drawing tolerance.

Suitable selected paths are assembled into one connected XYZ sequence. Shared endpoints must agree exactly in XYZ. Disconnected paths, ambiguous branches and overlapping duplicate paths are rejected. One exact repeated terminal closing XYZ vertex is removed from extracted selected-source input. Thus a selected closed triangle with A,B,C,A can supply three effective vertices. See the selected vertex boundary rules for the supported source types and extraction.

A nonempty temporary vertex sequence takes precedence over selected objects in ordinary Command use. Its raw records are retained, with no terminal-closure removal. Temporary A,B,C produces three dimensions, but temporary A,B,C,A fails because the implicit final-to-first edge is zero length. A one-record temporary sequence also fails; it does not fall back to a valid selected object. Successful execution retains the temporary sequence for reuse and leaves unrelated selected objects selected.

Command

Select suitable source objects with SelectItems, or supply temporary vertices through the vertex commands, then assign Command=AvaxCommand.V_Dim_c. Ordinary command assignment has no Boolean success acknowledgement and returns no new handles.

With three or more input vertices, the command requests a positive offset. If DialogEnabled_p is False, it uses the current DimDist_p without opening a dialog. With dialogs enabled, cancellation or invalid input creates nothing. A successful, fully prepared operation updates DimDist_p to the accepted offset. Cancellation, invalid geometry, bad settings, insufficient capacity and preparation failure preserve its previous value.

With exactly two input vertices, the command waits for one placement point. Supply that point with SetCoordinate or normal drawing input. Its perpendicular distance from the source line supplies the offset, and its side of the line supplies the dimension side. The point must not be on the source line. This placement distance is the actual perpendicular distance, rather than the current DimDist_p. It is local to this dimension: both successful and failed two-vertex placements preserve DimDist_p.

When selected objects supply the input, their selection is cleared only after the complete result has passed validation and preparation. Ordinary results are not automatically selected. Initial rejection preserves source selection, accepted input and the existing Redo branch. A selected-source selection-clearing callback can take ownership before the dimensions are added.

Ordinary creation is unavailable while PauseAvax or LockEdit is enabled, or while the control is stopped, disabled or disposed. A command with no usable input is rejected; it does not start an unrelated free-form dimension construction.

DimDist_p

Supplies the positive finite ordinary-command offset when a three-or-more-vertex request runs without a parameter dialog. The offset is an absolute perpendicular drawing distance. DimScale_p does not multiply it. A two-vertex ordinary placement derives a local distance from its placement point and preserves this property. Direct execution uses the explicit schema below and never changes DimDist_p.

Dimension settings

The command takes one settings snapshot for an invocation. Required numbers must be finite. Invalid settings reject the whole operation before source selection, history or drawing geometry changes.

Property	Purpose and valid values
DimScale_p	Positive scale applied once to text scale, text gap, marks and extension lengths. It does not scale the source, offset or measured value.
DimTextScale_p	Positive text scale. The effective text scale is DimTextScale_p multiplied by DimScale_p.
DimTextFont_p	Font-table entry from 1 through 256, used for the created text instead of PenFontNumber.
TextInitWidth_p and TextInitHeight_p	Positive synthetic text width and height metrics used by the R2 placement and stored bounds described below.
DimDecimal_p	Integer precision from 0 through 15, used by the selected number format.
DimFactor_p	Positive multiplier for the displayed XY length. It changes the label, not the dimension geometry.
DimUnitFormat_p	0: fixed decimal; 1: feet and decimal inches; 2: feet and fractional inches; 3: scientific notation; 4: fractional inches.
DimFractionalTick_p	Positive Integer fraction denominator for fractional formats. The control's default is 2.
DimDefText_p	Suffix appended to the formatted value. An empty string is allowed; Nothing and control characters are invalid. Trailing ASCII spaces are removed; leading content remains significant.
DimArrowSize_p	Nonnegative inline text-gap allowance on each side of the occupied text span. This is distinct from the endpoint mark size.
DimTickSize_p	Nonnegative endpoint mark size. Zero omits the marks; the circle radius is half this effective size.
DimArrowType_p	0: diagonal ticks; 1: two-arm arrows; 2: circles.
DimOutExtLines_p	Nonnegative extension length from the dimension baseline away from the source line. Zero omits these two roots.
DimInExtLines_p	Nonnegative extension length from the dimension baseline towards the source line. A positive value also adds the two endpoint point markers.
DimTextAbove_p	True creates a continuous baseline with text above it. False requests an inline text gap; insufficient room changes that edge to above text.
DimTextHorizontal_p	Requests horizontal text and the corresponding R2 placement. Its occupied inline span is limited by the text height projected onto the dimension direction.
DimTextVerticalToLine_p	Applies the R2 perpendicular text orientation and placement after the other text options. It takes precedence over horizontal orientation.

Geometry and text placement

For a two-vertex request, the supplied side point chooses which parallel offset line is used. For three or more vertices, each edge makes the R2 local side decision independently: compare the two infinite offset lines against the preceding and following finite source edges, counting endpoint intersections. Choose the candidate with fewer intersections; a tie uses the second candidate. This is a local neighbor rule, not a global clockwise/counterclockwise area test.

Dimension endpoints are put in the R2 upright order before marks and text are placed. Text follows that baseline unless horizontal or perpendicular text is requested. The legacy perpendicular rule is preserved, including its orientation for a descending baseline: a baseline angle of 315 degrees gives perpendicular text at 225 degrees. The horizontal and perpendicular options retain their separate R2 insertion adjustments.

Text layout and stored bounds use the R2 synthetic metrics, independently of Windows glyph measurement: width is 0.5 multiplied by the UTF-16 caption length, TextInitWidth_p, DimTextScale_p and DimScale_p; height is TextInitHeight_p multiplied by DimTextScale_p and DimScale_p. The original numeric caption width is used for gap and insertion calculations. Any prefix spaces required by perpendicular above-text placement are included in the final stored text bounds. Changing the actual font does not substitute a measured glyph width for this layout rule.

Inline text reserves its occupied span plus two effective DimArrowSize_p allowances. For an individual edge, an allowance at least half the edge length is limited to one third of that length. When the text cannot fit, that edge uses above text. The clamp is calculated

independently for each edge. A mathematically zero inline end-piece can be represented as a Point_i root, as with the R2 importer; a nonzero requested component that collapses at coordinate storage precision is rejected.

Ticks cross the baseline diagonally. Arrow arms retain the R2 finite baseline/chord intersection rule, including its one-time direction reversal for oversized marks. Outside extensions always point away from the source and inside extensions always point towards it, even when their requested lengths exceed the offset. The point markers lie towards the source at the effective inside length plus mark size.

All created geometry has Z=0. Source XYZ geometry is preserved, and pen elevation and height remain separate metadata. Current pen color, width, line style, layer, elevation and height apply to the new roots; point markers use the current point style. Dimension text uses its selected font-table entry and current font style flags.

Ordered output roots

Each edge contributes its roots in the following order, then the next edge starts its own group:

1. One continuous main baseline for above text, or two main end-pieces for inline text.
2. All marks at the first upright endpoint, then all marks at the second endpoint.
3. Outside extension at the first endpoint, then at the second endpoint, if enabled.
4. Inside extension at the first endpoint, then at the second endpoint, if enabled.
5. Point marker at the first endpoint, then at the second endpoint, when inside extensions are enabled.
6. One text root.

The main pieces, marks and extensions are separate roots; extensions are not joined into one line through the baseline. If M is 1 for above text or 2 for inline text, K is 0 without marks, 2 for ticks or circles, or 4 for arrows, O is 1 when outside extensions are enabled, and I is 1 when inside extensions are enabled, the edge contributes M+K+2*O+4*I+1 roots. For example, above text with ticks and both extension options produces 10 roots per edge.

DoAvaxAction

Pass V_Dim_c and a one-based Integer handle array with an unused zero element and at least one existing source handle. All active handles must be valid. Duplicate handles are considered once in first-occurrence order, and suitable sources are assembled into one boundary. Explicit handles own a direct request; current selection and unrelated temporary vertices do not replace them. The input array, action value and input data are unchanged.

Collected vertices	Accepted vIn	Meaning
Exactly two	A Double() of exactly three elements: {offset, sideX, sideY}	Positive explicit offset and a finite side point not on the source line. This schema is required.
Three or more	Nothing	Use the current positive DimDist_p.
Three or more	A Double() of exactly one element: {offset}	Use this positive explicit offset.

No other data form is accepted. A scalar Double, Object() array, wrong-length array or three-element array for a three-or-more-vertex walk is invalid. This strict direct-input schema is a new v3 contract; it is not claimed as an R2 generic-dispatch input format. Direct execution requires no parameter or placement dialog.

Success returns True, fNewHandles=True, errCode=0 and vOut=Nothing. OutHandles is a one-based array containing every new root in the edge and primitive order above; element zero is unused. The new roots become selected, and all source objects remain unchanged. The returned handle count depends on the geometry and active settings, so applications must not assume one handle per edge.

Rejection returns False, fNewHandles=False, OutHandles={0} and vOut=Nothing. Invalid handles, schemas, geometry, settings, capacity or preparation use errCode=2. A stopped, disabled, disposed or edit-locked control uses errCode=1. A direct creation may run while PauseAvax is enabled when no accepted Polyline chain needs completion. A paused pending chain cannot be completed and returns errCode=1. Check the Boolean result and error code before using the output array.

Atomic preparation, history and callbacks

The complete collection of roots, text bounds, font resources and required handles is prepared before drawing edits or source selection clearing. Finite Double calculations preserve small representable differences near large origins and avoid intermediate overflow or underflow in the geometry calculations. Every final length, scaled component, coordinate, offset and text bound must be representable. Invalid initial requests preserve accepted input, selection, source geometry, Undo and the previous Redo branch.

A successful dimension result is one creation batch and one Undo step. With events and recording enabled, it publishes one NewUserItems batch and one Change notification. Direct result selection is also published as one batch. Redo restores the complete ordered result. No source object is deleted.

If direct execution must finish an accepted Polyline chain, requested dimensions and capacity are validated before that separate creation. The dimension request is prepared again after the Polyline completion callbacks. A callback that changes the command, selection, drawing or lifecycle takes ownership and prevents stale outer completion; normal direct rejection from that takeover uses errCode=1. The completed Polyline and any callback edits keep their own history entries.

Ordinary selected-source selection clearing happens before dimension creation. A takeover or exception in that callback prevents stale result creation. Creation callbacks see the complete committed result. Exceptions raised by application callbacks propagate with their original identity. If a callback throws after creation, the result remains committed and the public method has not returned its normal ByRef outputs. The outer operation does not overwrite a callback's newer command, selection or drawing.

R2 compatibility notes

The configured R2 dimension wrappers supplied the settings required by the old vertex job, but the generic V_Dim_c dispatch did not consistently provide those settings. The v3 direct contract above supplies a defined public route instead of treating that old wiring failure as intended behavior.

V3 applies the dimension scale once per invocation and derives each edge's gap from that unchanged snapshot. R2 could multiply shared text and component settings again on each edge and retain a reduced gap for later edges; those accumulation bugs are deliberately corrected. V3 also gives inside and outside extensions their stated geometric directions when they are overlong, correcting the old finite-segment side heuristic. These corrections do not alter the public command value or add a new public settings class.

Complete VB.NET examples

Use a started, enabled, edit-unlocked control. Supply existing source handles in a one-based array. The direct example requires a source sequence containing exactly two effective vertices, such as a Line_i. Configure dimension properties before calling it. For the ordinary example, a three-or-more-vertex source runs immediately with the current DimDist_p when DialogEnabled_p is False; a two-vertex source waits for the next placement point.

Create one two-vertex dimension through the direct method

```
Option Strict On
Imports System
Imports Avax

Public Module Help_VertexDimensions
    Public Function Create(cad As cAvax, entityHandles As Integer(),
                           offset As Double, sideX As Double,
                           sideY As Double) As Integer()
        If cad Is Nothing Then Throw New ArgumentNullException("cad")
        If entityHandles Is Nothing OrElse entityHandles.Length < 2 OrElse entityHandles(0) <> 0 Then
            Throw New ArgumentException("Supply a one-based array of existing source handles.",
    "entityHandles")
        End If
        If Double.IsNaN(offset) OrElse Double.IsInfinity(offset) OrElse offset <= 0 Then
            Throw New ArgumentOutOfRangeException("offset")
        End If
        If Double.IsNaN(sideX) OrElse Double.IsInfinity(sideX) OrElse
           Double.IsNaN(sideY) OrElse Double.IsInfinity(sideY) Then
            Throw New ArgumentException("The dimension side point must be finite.")
        End If
        Dim action As AvaxCommand = AvaxCommand.V_Dim_c
        Dim input As Object = New Double() {offset, sideX, sideY}
        Dim extra As Object = Nothing, output As Integer() = Nothing
        Dim created As Boolean, errorCode As Integer
        If Not cad.DoAvaxAction(entityHandles, input, action, created, output, extra, errorCode) Then
```

```
            Throw New InvalidOperationException("Vertex dimensions failed. Error: " &
    errorCode.ToString())
        End If
        If Not created OrElse errorCode <> 0 OrElse extra IsNot Nothing OrElse
           output Is Nothing OrElse output.Length < 2 OrElse output(0) <> 0 Then
            Throw New InvalidOperationException("Expected a one-based array of new dimension roots.")
        End If
        For index As Integer = 1 To output.Length - 1
            If output(index) <= 0 Then Throw New InvalidOperationException("A new dimension handle is
    invalid.")
        Next
        Return output
    End Function
End Module
```

Invoke the ordinary selected or temporary vertex command

```
Option Strict On
Imports System
Imports Avax

Public Module Help_VertexDimensionsCommand
    Public Sub Execute(cad As cAvax, entityHandles As Integer())
        If cad Is Nothing Then Throw New ArgumentNullException("cad")
        If entityHandles Is Nothing OrElse entityHandles.Length < 2 OrElse entityHandles(0) <> 0 Then
            Throw New ArgumentException("Supply a one-based array of existing source handles.",
    "entityHandles")
        End If
        If Not cad.SelectItems(entityHandles) Then
            Throw New InvalidOperationException("No source object could be selected.")
        End If
        ' The caller chooses DialogEnabled_n and DimDist_n before this call.
        ' A nonempty retained temporary sequence takes precedence over selection.
        ' For two vertices, the next SetCoordinate supplies the placement point.
        ' Command assignment itself does not acknowledge successful creation.
        cad.Command = AvaxCommand.V_Dim_c
    End Sub
End Module
```

Interactive Command-based creation publishes NewUserItems as described here. A direct DoAvaxAction call is an automatic operation: use OutHandles and fNewHandles for its result; it publishes Change without NewUserItems. Previously accepted user construction completed before that action retains its own notification.

DoVertex dimension job

API reference

Job 514 constructs linear dimensions along a vertex sequence and writes legacy drawing commands. Use it when integrating an application that already exchanges vertex job files. For dimensions inserted directly into a control, see V_Dim_c and DoAvaxAction.

Request and result

Supply a text job containing five lines: 514, record size (8, 12 or 24), total input byte count, packed input filename and settings/output filename. Filenames may be quoted. The input contains packed IEEE Single records for compatibility; each record's first XY pair supplies a vertex. Other fields remain unchanged. Every numeric record field must be finite. Settings and emitted text coordinates support Double precision.

Two vertices produce one dimension. Three or more form a closed walk; a single terminal XY closure marker is removed before construction. Collinear and self-crossing walks can be dimensioned edge by edge. A zero-length XY edge is invalid. Two-point placement uses the side point; other walks use the neighboring-edge rule. All generated geometry has Z=0.

DoVertex returns True when it processes the request. Read the four reply lines written to the job: semantic status (1 for dimensions, 0 when the guide cannot produce a complete result), kind 5, original input byte count and output filename. On semantic failure, the settings/output file is preserved. Malformed settings, file errors and publication failures return False without replacing the job or settings/output. A successful job replaces the settings file with drawing commands; retain or recreate the settings before another invocation.

Settings lines

Supply exactly 19 lines, or 22 lines including the optional unit settings. Numeric text uses invariant culture. Flags must be 0 or 1. Scales, text metrics and the length factor must be positive; component sizes must be nonnegative.

Line	Setting
1	Overall dimension scale
2	Text above the dimension line
3	Keep text horizontal
4	Gap beside inline text
5	Tick/arrow size; circle marks use half this value as radius
6	Outside extension length, away from the source
7	Inside extension length, toward the source
8	Integer font identifier copied to the text command, including legacy sentinel values
9	Text scale
10	Decimal precision, 0 through 15
11	Initial character width
12	Initial text height
13	Single-line suffix; leading spaces are preserved and trailing ASCII spaces removed
14	Length factor
15	Mark type: 0 tick, 1 arrow, 2 circle
16-17	Side-point X and Y
18	Offset distance. Zero is accepted by this file interface and places the baseline on the source; control commands require a positive offset.
19	Text perpendicular to the dimension line; takes precedence over horizontal text
20	Optional length format: 0 decimal, 1 engineering, 2 architectural, 3 scientific, 4 fractional
21	Optional fractional denominator; nonpositive values select 2

Line	Setting
22	Optional angular format, 0 through 4; retained by the format but not used for linear dimensions

The 19-line form uses decimal length formatting and denominator 2. Overall scale is applied once per component, consistently across all edges. Text scale also scales character width and height. The offset and length factor are independent of overall scale. A short-edge gap adjustment applies only to that edge.

Output records

For each edge, records are ordered as baseline segment(s), first and second endpoint marks, outside extensions, inside extensions, point markers and text. A point marker is a zero-length line record.

- Kind 1: six coordinate lines, X1, Y1, Z1, X2, Y2, Z2.
- Kind 2: twelve numeric lines: minimum XYZ, maximum XYZ, center XYZ, X radius, Y radius and final Z. Circle marks use equal radii and zero Z values.
- Kind 3: label line, font-identifier line, insertion XYZ, X/Y text scales, transform flag, and nine row-major values of the legacy row-vector affine matrix. Rotation is about the insertion point. Perpendicular text retains the transform flag even when the matrix is identity.

Coordinates are emitted as invariant-culture round-trip Double text. The command stream uses the Windows default text encoding required by legacy readers. An unencodable suffix is rejected before publication. Unrepresentable geometry is rejected as a complete result rather than exposing a partial dimension.

VB.NET example

This example creates one dimension above a twenty-unit line. It creates a unique subfolder under an existing working directory and returns the output filename. Import the command stream and remove that subfolder when finished. No control instance or StartAvax call is required for the file processor.

```
Option Strict On
Imports Avax
Imports System
Imports System.IO
Imports System.Text

Public Module Help_VertexDimensions
    Public Function Create(workingDirectory As String) As String
        Dim folder = Path.Combine(workingDirectory, "vertex-dimension-" & Guid.NewGuid().ToString("N"))
        Directory.CreateDirectory(folder)
        Try
            Dim data = Path.Combine(folder, "vertices.bin")
            Dim output = Path.Combine(folder, "dimensions.txt")
            Dim job = Path.Combine(folder, "job.txt")
            Using writer As New BinaryWriter(File.Create(data))
                writer.Write(0.0F) : writer.Write(0.0F)
                writer.Write(20.0F) : writer.Write(0.0F)
            End Using
            Dim settings As String() = {
                "1", "1", "0", "0.5", "0", "0", "0", "2", "1", "2",
                "1", "1", "", "1", "0", "0", "10", "2", "0"}
            File.WriteAllLines(output, settings, Encoding.Default)
            File.WriteAllLines(job, {"514", "8", "16", data, output}, Encoding.Default)
            Dim processor As New avVertex.cVertex()
            If Not processor.DoVertex(job) Then Throw New IOException("The dimension job could not be
    processed.")
            Dim reply = File.ReadAllLines(job)
            If reply(0) <> "1" Then Throw New InvalidOperationException("The guide did not produce
    dimensions.")
            Return output
        Catch
            Directory.Delete(folder, True)
            Throw
        End Try
    End Function
```

```
End Module
```

avVertex.cVertex Member Reference

API reference

Job 518 numbers vertices starting at 1. Job 519 labels vertices using the configured alphabet. These file jobs produce text commands without creating or requiring a control instance.

Input

Write five text lines: 518 or 519, binary record size, total binary input byte count, input filename, and command output filename. Record sizes 8, 12 and 24 contain two, three or six IEEE Singles respectively. Only the first X and Y of each record are used. Every raw record receives a label, including repeated vertices and an explicit closing vertex. A single vertex is valid.

Label placement

For each vertex, candidate positions are tried in this order: (X+0.2,Y+0.2), (X-0.2,Y+0.2), (X+0.2,Y-0.2), (X-0.2,Y-0.2). The first position outside the closed XY boundary is used; a position on its boundary counts as inside. If all four positions are inside, the fourth is used. Z is always zero. The offset is in drawing units and is not multiplied by a current text or dimension scale. If a coordinate is too large for the offset to produce a distinct finite Double, processing fails without replacing the files.

Alphabet settings for job 519

The process working directory must contain optavx/country.opt. Its first line is an Integer country identifier; trailing blank lines are permitted. The matching optavx/alpha<country>.opt must contain at least three lines: alphabet, area caption, and a finite numeric precision. Extra lines are ignored. The caption and precision are read for the shared R2 format but are not used by these label jobs. Job 518 does not read either settings file.

Only trailing ordinary spaces are removed from the alphabet. Symbols must be distinct printable text elements; a letter with combining marks is one symbol. Up to 1024 symbols are accepted. An alphabet AB produces A, B, AA, AB, BA, BB, AAA, and so on; numbering continues beyond the old two-character limit. Numeric numbering also continues beyond 32767.

Settings files with a byte-order mark support UTF-8, UTF-16 and UTF-32. Files without a mark use the Windows ANSI code page. The command output retains the R2 ANSI encoding (Encoding.Default); if a label cannot be encoded without loss, DoVertex returns False and preserves existing files. Choose an alphabet supported by the process code page.

Reply and output

A successful job replaces the request with four lines: 1, 5, original binary input byte count, command output filename. The binary input and alphabet settings remain unchanged. Each label uses eighteen lines:

1. Command number: 3.
2. Label text.
3. Font identifier: -1 (the legacy font sentinel).
4. X, Y and Z, each on its own line.
5. X scale and Y scale, each 1 on its own line.
6. Transform flag: 0.
7. Nine row-major identity matrix entries, each on its own line: 1,0,0,0,1,0,0,0,1.

Numeric text uses invariant Double precision. An earlier dimension job cannot change the font sentinel, unit scale or identity transform of these records. The output filename must differ from the request, binary input and, for job 519, both settings files. A malformed request or configuration, locked output, or publication error returns False and preserves existing files. Output and reply are published together.

VB.NET example

This complete example writes a triangle as packed XY Singles, numbers its three vertices, and returns the command lines. The first label is placed at (-0.2,0.2). The example removes its own temporary files in Finally and needs no StartAvax call.

```
Option Strict On
Imports Avax
Imports System
```

```
Imports System.IO
Imports System.Text

Public Module Help_VertexLabels
    Public Function NumberTriangle(workingDirectory As String) As String()
        Dim folder = Path.Combine(workingDirectory, "avax-labels-" & Guid.NewGuid().ToString("N"))
        Directory.CreateDirectory(folder)
        Try
            Dim data = Path.Combine(folder, "vertices.bin")
            Dim job = Path.Combine(folder, "job.txt")
            Dim output = Path.Combine(folder, "commands.txt")
            Using stream = File.Create(data), writer As New BinaryWriter(stream)
                For Each coordinate As Single In New Single() {0, 0, 10, 0, 0, 10}
                    writer.Write(coordinate)
                Next
            End Using
            File.WriteAllLines(job, {"518", "8", "24", data, output}, Encoding.Default)
            Dim processor As New avVertex.cVertex()
            If Not processor.DoVertex(job) Then Throw New IOException("The vertex label job could not be
    processed.")
            Dim reply = File.ReadAllLines(job, Encoding.Default)
            If reply(0) <> "1" Then Throw New InvalidOperationException("The vertices did not produce
    labels.")
            Return File.ReadAllLines(output, Encoding.Default)
        Finally
            Directory.Delete(folder, True)
        End Try
    End Function
End Module
```

Vertex area and perimeter commands

API reference

V_Area_c and V_Measure_c calculate one measurement from an ordered vertex boundary. They retain the source objects and their XYZ coordinates. Coordinates and returned measurements use Double; handles, command values and error codes use Integer. Call the control on its owning UI thread after StartAvax.

V_Area_c

Command value 512 calculates the absolute enclosed area in XY drawing units squared. The boundary needs at least three effective XY vertices forming a simple polygon with nonzero mathematical area. Clockwise and counterclockwise traversal give the same answer. An open boundary is implicitly closed from its last vertex to its first. A crossing or exactly degenerate polygon is rejected. A valid nonzero area smaller than the representable Double range may round to zero successfully.

V_Measure_c

Command value 513 calculates length in XY drawing units. Two vertices contribute one segment; identical XY positions give a successful zero-length result even when their Z values differ. A sequence of three or more vertices contributes every successive edge and an implicit final-to-first edge, even when its source polyline is open. This decision uses the collected sequence before removing repeated vertices: A,B,A and A,A,B each measure twice the XY distance from A to B. A repeated closing vertex contributes a zero-length implicit closing edge. Z does not contribute to the distance.

For example, (0,0), (3,0), (3,4) gives area 6 and vertex length 12. The two points (0,0), (3,4) give length 5. The answer is unchanged if those vertices have different Z values.

Boundary and numeric rules

The request uses one ordered boundary, rather than a sum of unrelated objects. Suitable connected source paths can be assembled into one chain. Shared endpoints must agree exactly in XYZ: the same XY with a different Z is not a connected endpoint. Disconnected paths, ambiguous branches and duplicated edges across source paths are rejected. Separate closed rings are not combined into a union or treated as holes. See the selected vertex boundary rules for source extraction.

Curved sources are measured through their collected vertex approximation. These commands do not promise an analytic curve length or area; the entity-measurement methods follow their own curve rules.

Measurements use the stored Double coordinates without removing distinct short edges by a drawing tolerance. The calculation is in XY even when its source geometry is three-dimensional. Source Z, elevation and height remain unchanged. The final answer must be a finite representable Double; an overflowing result fails instead of returning Infinity. Pen width, line style, zoom and display tessellation do not scale the answer.

Command

For ordinary command use, assign V_Area_c or V_Measure_c after selecting suitable objects with SelectItems, or after collecting a temporary boundary with SelectV_c and SetCoordinate. A retained nonempty temporary boundary takes precedence over selected objects. Without temporary vertices, selected suitable paths form one aggregate boundary. The measurement uses that aggregate once.

A successful ordinary measurement retains temporary vertices for reuse and leaves unrelated selected objects selected. When selected objects supplied the boundary, their selection is cleared after successful validation and before the answer is published. An initially invalid boundary does not clear the source selection. The measurement creates no objects or Undo record.

Ordinary commands can display the answer when DialogEnabled_p is enabled. With dialogs disabled, Command has no public numeric return value or public measurement-failure event. Use DoAvaxAction when the application needs a numeric result and an explicit success or error code. Assigning Command is not an acknowledgement that a measurement succeeded.

Ordinary command dispatch is unavailable while input is paused, editing is locked, or the control is stopped, disabled or disposed. These guards leave the accepted input unchanged.

DoAvaxAction

Both commands support direct execution. Supply a one-based Integer array with an unused zero element and at least one existing source handle. Duplicate handles are considered once in first-occurrence order. The supplied handles own the direct request; unrelated retained temporary vertices and current selection do not replace them. vIn is unused. The caller's handle array, input object and action value are unchanged.

Successful direct execution returns True, errCode=0, fNewHandles=False and a boxed Double in vOut. OutHandles is {0}: the measurement creates no new root handles. Direct execution returns the number without displaying a measurement dialog, independently of DialogEnabled_p.

The numeric vOut contract is an intentional v3 correction. The old generic vertex-command wrapper dispatched these commands without reliably assigning their public numeric result. Applications should use the v3 result fields rather than infer an answer from the old wrapper's Boolean return.

Direct rejection returns False, fNewHandles=False, OutHandles={0} and vOut=Nothing. Invalid handles, unsuitable topology, invalid polygons and unrepresentable results use errCode=2. A control that is not started, is disabled or is disposed uses errCode=1. Use the returned status before reading vOut.

A direct query may run while PauseAvax or LockEdit is enabled when there is no accepted Polyline chain waiting for completion. If such a chain exists, paused or edit-locked input cannot be completed: the query returns False with errCode=1 without editing the drawing. After successful validation and any required Polyline completion, direct execution consumes the old construction and temporary vertex input and establishes the SelectStay_p idle state. It does not change the current selection or publish a selection-change notification merely for the numeric query.

If an accepted Polyline chain needs completion, the requested measurement is validated before that chain is committed as a separate creation. After its callbacks, the direct method reads the supplied source handles again and recalculates the measurement. A callback that takes ownership of another command, selection, document or control lifecycle prevents the outer query from overwriting that state; the outer query returns errCode=1. The earlier Polyline creation remains its own completed operation. OutHandles remains {0} for the numeric query.

Source data and history

The measurement itself creates, removes and replaces no drawing objects and adds no Undo entry. Initial validation and calculation failure preserve source geometry, metadata, accepted construction input and the previous Redo branch. A measurement-only call produces no NewUserItems or Change notification.

Callbacks from an earlier Polyline completion may start another command, change selection, replace the drawing or end the control. Their newer state remains owned by that callback. Exceptions raised by application callbacks propagate with their original identity. When an exception escapes, the method has not returned its normal ByRef output values.

Measure_c and Area_c

V_Measure_c and V_Area_c consume one vertex stream. Measure_c and Area_c measure selected entities under their own entity rules and can total several independent objects. In particular, GetPerimeter on an open polyline follows its stored edges, whereas V_Measure_c implicitly closes a vertex stream with three or more vertices. Do not substitute one family for the other when the closing edge or aggregation matters.

Complete VB.NET examples

Pass a started, enabled control and suitable existing source handles in a one-based array. The first example returns the programmatic result. The second invokes the ordinary command and deliberately makes no claim about a numeric result or success acknowledgement.

Return a numeric vertex measurement

```
Option Strict On
Imports System
Imports Avax

Public Module Help_VertexMeasure
    Public Function Query(cad As cAvax, entityHandles As Integer(), area As Boolean) As Double
        If cad Is Nothing Then Throw New ArgumentNullException("cad")
        Dim action As AvaxCommand = If(area, AvaxCommand.V_Area_c, AvaxCommand.V_Measure_c)
        Dim input As Object = Nothing, value As Object = Nothing
        Dim created As Boolean = True, output As Integer() = Nothing, errorCode As Integer = -1
        If Not cad.DoAvaxAction(entityHandles, input, action, created, output, value, errorCode) Then
```

```
            Throw New InvalidOperationException("Vertex measurement failed. Error: " &
    errorCode.ToString())
        End If
        If created OrElse errorCode <> 0 OrElse output Is Nothing OrElse
           output.Length <> 1 OrElse output(0) <> 0 OrElse Not TypeOf value Is Double Then
            Throw New InvalidOperationException("Unexpected vertex measurement result.")
        End If
        Dim answer As Double = DirectCast(value, Double)
        If Double.IsNaN(answer) OrElse Double.IsInfinity(answer) Then
            Throw New InvalidOperationException("The measurement is not finite.")
        End If
        Return answer
    End Function
End Module
```

Invoke the ordinary selected-boundary command

```
Option Strict On
Imports System
Imports Avax

Public Module Help_VertexMeasureCommand
    Public Sub Execute(cad As cAvax, entityHandles As Integer(), area As Boolean)
        If cad Is Nothing Then Throw New ArgumentNullException("cad")
        If entityHandles Is Nothing OrElse entityHandles.Length < 2 Then
            Throw New ArgumentException("Supply a one-based array of existing source handles.",
    "entityHandles")
        End If
        If Not cad.SelectItems(entityHandles) Then
            Throw New InvalidOperationException("No source object could be selected.")
        End If
        ' A retained temporary boundary takes precedence over this selection.
        ' Use Query when the caller needs the returned number or error status.
        cad.Command = If(area, AvaxCommand.V_Area_c, AvaxCommand.V_Measure_c)
    End Sub
End Module
```

avVertex.cVertex Member Reference

API reference

Use these file jobs to calculate a polygon's centroid or area, measure a vertex walk, construct its bounding rectangle, or connect its centroid to its vertices. The interface accepts the packed vertex files used by R2 applications.

Input

Write five text lines: job number, record size (8, 12 or 24 bytes), total input byte count, binary input filename, and auxiliary information. Each packed IEEE Single record contributes only its first XY pair. Z and the additional endpoint of a 24-byte record do not affect these planar operations. Every record field must be finite. The input must contain at least one complete record.

DoVertex returns True when processing and publication succeed. Read the first reply field to determine whether the geometry itself produced a result. Malformed input, file locks or publication errors return False and preserve the job and binary input.

Jobs and reply fields

Job	Purpose	Successful reply
503	Axis-aligned bounding rectangle of all first-XY positions.	1, 2, 96, original auxiliary information.
504	Area-weighted centroid of a simple, nonzero polygon.	1, 0, 0, XYZ, centroid X, centroid Y, 0.
505	One radius from the polygon centroid to each raw source vertex.	1, 2, output byte count, original auxiliary information.
512	Positive polygon area, independent of winding.	1, 4, area.
513	XY length of the vertex walk.	1, 4, length.

Each listed field is on its own line. Text coordinates, areas and lengths use invariant Double values. Area and centroid calculations avoid cancellation at large world coordinates. Lengths use the rounded Double sum of XY edge lengths.

Geometry and special cases

Jobs 503 and 505 replace the binary input with complete 24-byte edge records: X1, Y1, 0, X2, Y2, 0, each a Single. Border 503 always writes four edges in this order: bottom-left to top-left, top-left to top-right, top-right to bottom-right, bottom-right to bottom-left. A one-point or line-shaped boundary retains its zero-length sides. Bounds are not restricted by the old fixed min/max sentinel.

Radiate 505 emits one edge for every source record, in input order. Repeated vertices and a terminal closure marker therefore produce repeated radii. The 12- and 24-byte input cases produce the same complete planar edge layout as 8-byte inputs. Retain a copy of the source binary file before either geometry job if it is still needed.

Centroid 504, area 512 and length 513 preserve the binary input. A polygon is closed implicitly; consecutive duplicate XY vertices and a terminal closure marker do not change its area or centroid. Fewer than three distinct noncollinear vertices, self-crossings and non-simple boundaries have no valid area or centroid. For 504, failure replies are 0, 0, 0, an empty information line, with no XYZ fields. For 512 the reply is 0, 4, 0. For 505 the reply is 0, 0, original input byte count, original auxiliary information, and the binary input is preserved.

Length 513 treats exactly two raw records as one segment. Three or more raw records form a closed walk, including the edge back to the first record. Thus A,B measures AB once, while A,B,A measures it twice. Crossing, collinear and repeated vertices remain measurable. One raw record returns 0, 4, 0; two identical points successfully return 1, 4, 0.

VB.NET example

This complete example measures a 10 by 6 rectangle. It returns its area, centroid X, centroid Y and perimeter: 60, 5, 3, 32. It rewrites the request before each operation because DoVertex replaces the request with a reply, and removes its own temporary directory in Finally. No control instance or StartAvax call is required.

```
Option Strict On
Imports Avax
Imports System
Imports System.Globalization
```

```
Imports System.IO
Imports System.Text

Public Module Help_VertexMeasures
    Public Function MeasureRectangle(workingDirectory As String) As Double()
        Dim folder = Path.Combine(workingDirectory, "avax-measure-" & Guid.NewGuid().ToString("N"))
        Directory.CreateDirectory(folder)
        Try
            Dim data = Path.Combine(folder, "vertices.bin")
            Dim job = Path.Combine(folder, "job.txt")
            Using output = File.Create(data), writer As New BinaryWriter(output)
                For Each coordinate As Single In New Single() {0, 0, 10, 0, 10, 6, 0, 6}
                    writer.Write(coordinate)
                Next
            End Using
            Dim areaReply = RunMeasurement(job, data, 512)
            Dim centerReply = RunMeasurement(job, data, 504)
            Dim lengthReply = RunMeasurement(job, data, 513)
            Return {Double.Parse(areaReply(2), CultureInfo.InvariantCulture),
                    Double.Parse(centerReply(4), CultureInfo.InvariantCulture),
                    Double.Parse(centerReply(5), CultureInfo.InvariantCulture),
                    Double.Parse(lengthReply(2), CultureInfo.InvariantCulture)}
        Finally
            Directory.Delete(folder, True)
        End Try
    End Function
    Private Function RunMeasurement(job As String, data As String, number As Integer) As String()
        File.WriteAllLines(job, {number.ToString(CultureInfo.InvariantCulture), "8", "32", data, ""},
    Encoding.Default)
        Dim processor As New avVertex.cVertex()
        If Not processor.DoVertex(job) Then Throw New IOException("The measurement job could not be
    processed.")
        Dim reply = File.ReadAllLines(job, Encoding.Default)
        If reply(0) <> "1" Then Throw New InvalidOperationException("The vertices did not produce a
    measurement.")
        Return reply
    End Function
End Module
```

avVertex.cVertex Member Reference

API reference

Jobs 501 and 502 create an inward or outward parallel boundary from a closed polygon. They provide the file interface used by applications that exchange R2 vertex jobs. Both job numbers use the same settings; the side character determines inward or outward operation.

Request

Write five text lines: job number (501 or 502), record size (8, 12 or 24 bytes), total input byte count, packed input filename, and offset settings. The final line begins with the actual character ChrW(1) for inside or ChrW(2) for outside, immediately followed by the invariant numeric distance. It does not begin with the printable digit "1" or "2". The distance supports Double precision; its absolute value is used. Zero copies the planar boundary.

The binary input remains a sequence of packed IEEE Single records. Each record's first XY pair supplies a vertex; Z and the additional endpoint of a 24-byte record do not define the boundary. All fields must be finite. Consecutive duplicate XY points and a terminal closure point are accepted. The remaining points must form a simple, nondegenerate polygon. The closing edge is implicit.

Result and file lifetime

On success, the input binary file is replaced by closed contour edges. Each edge contains six packed Singles: X1, Y1, 0, X2, Y2, 0. Acute corners retain full miter intersections. Very elongated concave polygons retain their narrow features. Concave polygons can split into multiple contours during an inward offset; consume every edge, without inventing connections between contours. Edge order and the starting vertex are not guaranteed.

DoVertex returns True when it processes a valid request. The four reply lines replacing the job file are semantic status 1, kind 2, output byte count and the unchanged settings string. The byte count divided by 24 is the number of returned edges. Preserve the original input yourself if the application also needs the source boundary.

If the source is degenerate or self-crossing, or inward offset removes the whole polygon, the reply is 0, 0, original input byte count and the unchanged settings string. The binary input is preserved. This is a processed request, so the Boolean return is still True.

Malformed settings, file locks, unsupported numeric precision and output outside the packed Single range return False and preserve both files. In particular, distinct output vertices must remain distinct after conversion to Single. Output geometry and its reply are published together; a failed write does not leave a partial replacement. Use separate job and data paths, and do not modify them while processing.

VB.NET example

This complete example offsets a 20 by 10 rectangle outward by 2 and returns the packed result as Double values. Its temporary job directory is removed in Finally. No control instance or StartAvax call is needed.

```
Option Strict On
Imports Avax
Imports System
Imports System.IO
Imports System.Text

Public Module Help_VertexOffset
    Public Function OffsetRectangle(workingDirectory As String) As Double()
        Dim folder = Path.Combine(workingDirectory, "avax-offset-" & Guid.NewGuid().ToString("N"))
        Directory.CreateDirectory(folder)
        Try
            Dim data = Path.Combine(folder, "vertices.bin")
            Dim job = Path.Combine(folder, "job.txt")
            Using output = File.Create(data), writer As New BinaryWriter(output)
                For Each coordinate As Single In New Single() {0, 0, 20, 0, 20, 10, 0, 10}
                    writer.Write(coordinate)
                Next
            End Using
            File.WriteAllLines(job, {"502", "8", "32", data, ChrW(2) & "2"}, Encoding.Default)
            Dim processor As New avVertex.cVertex()
            If Not processor.DoVertex(job) Then Throw New IOException("The offset job could not be
    processed.")
```

```
            Dim reply = File.ReadAllLines(job, Encoding.Default)
            If reply(0) <> "1" Then Throw New InvalidOperationException("The polygon did not produce an
    offset.")
            Dim bytes = File.ReadAllBytes(data)
            Dim values(bytes.Length \ 4 - 1) As Double
            For i = 0 To values.Length - 1
                values(i) = BitConverter.ToSingle(bytes, 4 * i)
            Next
            Return values
        Finally
            Directory.Delete(folder, True)
        End Try
    End Function
End Module
```

avVertex.cVertex Member Reference

API reference

Job 516 cleans and orders segment records into connected chains. Job 517 splits segments at their XY intersections and writes drawing commands. Both operate on complete XYZ segments; they preserve endpoint elevations, unlike the planar vertex construction jobs.

Input

Write five text lines: 516 or 517, record size 24, total binary input byte count, input filename, and auxiliary information (516) or command output filename (517). Each binary record contains six finite IEEE Singles: X1, Y1, Z1, X2, Y2, Z2. Other record sizes are invalid for these jobs.

516: cleanup and ordering

Cleanup removes zero-length XYZ edges, exact duplicate edges in either direction, and edges fully contained in another collinear XYZ edge. Partial overlaps are retained. Different elevation profiles are retained even when they have the same XY projection. A Z-only segment is not a zero-length XYZ edge.

Ordering matches XY endpoints exactly. It can reverse a segment's two complete XYZ endpoints, but does not move them or snap away a positive gap. Segments meeting in XY may have different heights at that contact. Disconnected components and branches remain separate records; no bridging edges are invented. Do not depend on a particular starting vertex or component order.

A successful four-line reply replaces the job file: 1, 2, output byte count, unchanged auxiliary information. The binary input is replaced by the ordered 24-byte records. If cleanup leaves no usable edges, the reply is 0, 0, original input byte count, unchanged auxiliary information, and the binary input is preserved. DoVertex returns True in both cases; inspect the first reply field.

517: intersections and output

Breaking uses XY intersections and interpolates Z independently along each source segment. T junctions split the segment whose interior is touched. Collinear overlap boundaries partition the sources without merging their elevations or removing duplicate owners. Z-only segments and point records are retained. All intersections are processed; the old small output-size cutoff does not limit the result.

The binary input is preserved. The four-line reply is 1, 5, original input byte count, command output filename. The output file is replaced by line commands: one line containing 1, followed by six lines containing X1, Y1, Z1, X2, Y2, Z2. Numeric text uses invariant Double precision, including intersections that cannot be represented exactly as Singles.

Use a command output path distinct from both the job and binary input. A malformed request, locked file, invalid output path or unrepresentable split returns False and preserves existing files. Geometry or commands and their reply are published together. Retain a copy of the source binary file if it is needed after successful job 516.

VB.NET example

This complete example splits two crossing segments with different height profiles. It returns four XYZ segments. At the shared XY intersection (5,0), the first source has Z=5 and the second has Z=150. The example removes its temporary directory in Finally and needs no control instance or StartAvax call.

```
Option Strict On
Imports Avax
Imports System
Imports System.Collections.Generic
Imports System.Globalization
Imports System.IO
Imports System.Linq
Imports System.Text

Public Module Help_VertexSegments
    Public Function SplitCrossing(workingDirectory As String) As Double()()
        Dim folder = Path.Combine(workingDirectory, "avax-segments-" & Guid.NewGuid().ToString("N"))
        Directory.CreateDirectory(folder)
        Try
            Dim data = Path.Combine(folder, "segments.bin")
            Dim job = Path.Combine(folder, "job.txt")
```

```
            Dim output = Path.Combine(folder, "commands.txt")
            Using stream = File.Create(data), writer As New BinaryWriter(stream)
                For Each coordinate As Single In New Single() {0, 0, 0, 10, 0, 10, 5, -5, 100, 5, 5, 200}
                    writer.Write(coordinate)
                Next
            End Using
            File.WriteAllLines(job, {"517", "24", "48", data, output}, Encoding.Default)
            Dim processor As New avVertex.cVertex()
            If Not processor.DoVertex(job) Then Throw New IOException("The segment job could not be
    processed.")
            Dim reply = File.ReadAllLines(job, Encoding.Default)
            If reply(0) <> "1" Then Throw New InvalidOperationException("The segments did not produce a
    result.")
            Dim rows = File.ReadAllLines(output, Encoding.Default)
            If rows.Length Mod 7 <> 0 Then Throw New InvalidDataException("Incomplete line command.")
            Dim result As New List(Of Double())
            For i = 0 To rows.Length - 7 Step 7
                If rows(i) <> "1" Then Throw New InvalidDataException("Expected a line command.")
                result.Add(rows.Skip(i + 1).Take(6).Select(Function(value) Double.Parse(value,
    CultureInfo.InvariantCulture)).ToArray())
            Next
            Return result.ToArray()
        Finally
            Directory.Delete(folder, True)
        End Try
    End Function
End Module
```

Vertex triangle command

API reference

V_Triangle_c

Command value 521 constructs two sides of a triangle from a guide, a baseline and two side lengths. The baseline itself is not created. The guide chooses which side of the baseline contains the apex; it does not set the apex height. Call the control on its owning UI thread after StartAvax.

Four input records

The control accepts four ordered XYZ records entered through SelectV_c, restored through LastSv_c, or extracted from selected source paths. X and Y have the following meanings:

Record	X	Y
1	Guide X	Guide Y
2	First baseline endpoint X	First baseline endpoint Y
3	Second baseline endpoint X	Second baseline endpoint Y
4	Length from the first endpoint to the apex	Length from the second endpoint to the apex

All control coordinates and lengths use Double in v3. Z is retained in temporary input but does not affect this planar construction; output geometry has Z=0. The fourth record is a pair of lengths, rather than a fourth geometric vertex. Do not reorder the records or remove a fourth record merely because it equals the guide.

A nonempty temporary stream takes priority over the object selection, even if its record count or geometry is invalid. It must contain exactly four raw records. The stream remains reusable after successful construction, and an unrelated object selection is retained. With no temporary stream, selected paths must form one unambiguous connected XYZ path. Its ordered vertices supply the request. One exact fifth terminal closing record may be removed; four records are then required. Source objects remain intact, and ordinary selected-source execution clears their selection after successful preparation.

Geometry and completion

Both baseline endpoints must be distinct in XY, the side lengths must be positive, and the three lengths must form a nondegenerate triangle. The guide must lie strictly on one side of the baseline. The first length always belongs to the first endpoint, including when the baseline is reversed. Finite small geometry and subnormal guide offsets are evaluated without a fixed world-unit tolerance.

The result consists of two Line_i objects in order: first endpoint to apex, then apex to second endpoint. For guide (3,4), endpoints (0,0) and (6,0), and lengths 5 and 5, the apex is (3,4). Reversing the guide's Y sign chooses (3,-4). The two lines use the current pen attributes and form one Undo/Redo operation. All final geometry, resources and handle capacity are prepared before committing; failure does not create a partial triangle.

The complete request executes immediately and opens no length dialog. Ordinary Command has no Boolean success return. It reports preparation errors through OperationFailed and returns to the idle selection mode controlled by SelectStay_p. The six area/side annotations of Area_3V_c are a separate operation.

DoAvaxAction

Pass V_Triangle_c, a one-based Integer array of source handles and vIn=Nothing. The explicit source paths supply the four records regardless of retained temporary input or current selection. Repeated handles are used once. Success returns True, fNewHandles=True, errCode=0, vOut=Nothing and an OutHandles array containing zero followed by the two new line handles. The source objects remain intact and the results become selected.

Invalid input or preparation returns False, fNewHandles=False, OutHandles={0}, vOut=Nothing and errCode=2. Lifecycle or input-ownership rejection uses error 1. Automatic direct creation publishes Change without NewUserItems. Ordinary interactive creation publishes its usual complete-batch notifications. A callback may start a newer command without the older completion clearing its input.

Other input forms

When there is no retained legacy request, v3 preserves its additional selected-Line base and three-picked-vertex modes by entering Triangle_c. Command then reports Triangle_c while that interactive mode is active. Nonempty legacy input always owns the request and is never replaced by this fallback.

The separate DoVertex file job 521 keeps its original eight-Single binary layout for existing file-job clients. That wire format does not narrow the public control's Double coordinate arrays.

VB.NET example: retained records

Pass an already started control. This example enters the four records and invokes ordinary construction.

```
Option Strict On
Imports Avax
Imports System

Public Module Help_VertexTriangleCommand
    Public Sub Construct(cad As cAvax)
        cad.Command = AvaxCommand.Cancel_c
        cad.Command = AvaxCommand.SelectV_c
        Dim records As Double()() = {
            New Double() {3, 4, 0},
            New Double() {0, 0, 0},
            New Double() {6, 0, 0},
            New Double() {5, 5, 0}}
        For Each record As Double() In records
            Dim x As Double = record(0), y As Double = record(1), z As Double = record(2)
            If Not cad.SetCoordinate(x, y, z) Then
                Throw New InvalidOperationException("The construction record could not be entered.")
            End If
        Next
        cad.Command = AvaxCommand.V_Triangle_c
    End Sub
End Module
```

VB.NET example: explicit source

requestSource is an existing polyline containing the four ordered records. The function returns the two new handles in a one-based array.

```
Option Strict On
Imports Avax
Imports System

Public Module Help_VertexTriangleDirect
    Public Function Construct(cad As cAvax, requestSource As Integer) As Integer()
        Dim objectHandles As Integer() = {0, requestSource}
        Dim action As AvaxCommand = AvaxCommand.V_Triangle_c
        Dim input As Object = Nothing, extra As Object = Nothing
        Dim created As Boolean, output As Integer() = Nothing, errorCode As Integer
        If Not cad.DoAvaxAction(objectHandles, input, action, created, output, extra, errorCode) Then
            Throw New InvalidOperationException("Triangle construction failed. Error: " &
    errorCode.ToString())
        End If
        If Not created OrElse output Is Nothing OrElse output.Length <> 3 Then
            Throw New InvalidOperationException("Expected two new line handles.")
        End If
        Return output
    End Function
End Module
```

avVertex.cVertex Member Reference

API reference

For control-based Double input, see the separate V_Triangle_c command.

Job 521 constructs two sides of a triangle from a baseline, two side lengths and a guide point. The guide selects the half-plane containing the new apex. The baseline itself is not emitted. To annotate the lengths and area of an existing triangle, use job 520 instead.

Input

Write five request lines: 521, record size 0 or 8, byte count 32, binary input filename, and command output filename. Zero is accepted because the original R2 caller left the record-size field uninitialized. Both values describe the same fixed sequence of eight finite IEEE Singles:

1. Guide X, guide Y.
2. First baseline endpoint X, Y.
3. Second baseline endpoint X, Y.
4. Distance from the first endpoint to the apex, distance from the second endpoint to the apex.

The binary layout remains Single for existing R2 clients. Calculations and numeric output use Double. There are no Z coordinates or alphabet settings in this job. Paths containing spaces may be enclosed in double quotes.

Construction

The baseline endpoints must be distinct, both side lengths must be positive, and the three lengths must form a nondegenerate triangle. The guide must be strictly on one side of the baseline. Its distance from the baseline does not set the triangle height. The first side length always belongs to the first endpoint, including when the baseline direction is reversed.

The result follows the requested half-plane and preserves the baseline endpoints. Very small nonzero geometry is not discarded by a fixed tolerance. Calculated coordinates may exceed the Single range because the output is Double text. If the apex cannot be stored as a distinct finite Double point on the requested side, the job fails without replacing files.

Reply and output

Successful construction returns True and replaces the request with four lines: 1, 5, 32, command output filename. The binary input is preserved. The command file contains two seven-line records: command number 1 followed by X1,Y1,Z1,X2,Y2,Z2, each on its own line. The first record runs from the first baseline endpoint to the apex; the second runs from the apex to the second endpoint. All Z values are zero. Numbers use invariant Double precision.

Invalid triangle lengths, coincident endpoints or a guide on the baseline are processed geometry failures: DoVertex returns True, the reply is 0,5,32,output filename, and any existing command output remains unchanged. Check the first reply field in addition to the Boolean return value.

A malformed request, wrong input length, nonfinite value, unusable output path, locked file or unrepresentable apex returns False and preserves the request, binary input and previous output. The output filename must differ from the request and input. Output and reply are published together.

VB.NET example

This complete example uses a baseline from (0,0) to (6,0), side lengths 5 and 5, and a guide above the baseline. It returns the two line commands with apex (3,4). It needs no control instance or StartAvax call and removes its own temporary files in Finally.

```
Option Strict On
Imports Avax
Imports System
Imports System.IO
Imports System.Text

Public Module Help_VertexTriangle
    Public Function ConstructTriangle(workingDirectory As String) As String()
        Dim folder = Path.Combine(workingDirectory, "avax-triangle-" & Guid.NewGuid().ToString("N"))
        Directory.CreateDirectory(folder)
```

```
        Try
            Dim data = Path.Combine(folder, "triangle.bin")
            Dim job = Path.Combine(folder, "job.txt")
            Dim output = Path.Combine(folder, "commands.txt")
            Using stream = File.Create(data), writer As New BinaryWriter(stream)
                For Each value As Single In New Single() {3, 4, 0, 0, 6, 0, 5, 5}
                    writer.Write(value)
                Next
            End Using
            File.WriteAllLines(job, {"521", "0", "32", data, output}, Encoding.Default)
            Dim processor As New avVertex.cVertex()
            If Not processor.DoVertex(job) Then Throw New IOException("The triangle job could not be
    processed.")
            Dim reply = File.ReadAllLines(job, Encoding.Default)
            If reply(0) <> "1" Then Throw New InvalidOperationException("The lengths and guide did not
    define a triangle.")
            Return File.ReadAllLines(output, Encoding.Default)
        Finally
            Directory.Delete(folder, True)
        End Try
    End Function
End Module
```

cAvax Member Reference

API reference

These methods address an embedded control by its owning drawing-object handle and child index. Document children start at zero. Shape controls retain their saved indices. Use AvaxControlGetAllControls to obtain existing identities.

AvaxControlGetData

AvaxControlGetShape

AvaxControlSetData

Embedded-control refresh and enumeration

API reference

AvaxControlRefresh

AvaxControlRefreshAll

AvaxControlGetAllControls

VB.NET example

Call this procedure outside an embedded-control event, after starting the drawing. It enumerates the current controls and refreshes each one. The returned value counts successful refreshes.

```
Option Strict On
Imports System

Public Module WidgetRefreshExample
    Public Function RefreshControls(cad As Avax.cAvax) As Integer
        If cad Is Nothing Then Throw New ArgumentNullException(NameOf(cad))
        Dim count As Integer = cad.AvaxControlGetAllControls(Nothing, Nothing, Nothing, Nothing, Nothing)
        Dim parentIds(count), controlIndices(count) As Integer
        Dim names(count), tags(count), patterns(count) As String
        cad.AvaxControlGetAllControls(parentIds, controlIndices, names, tags, patterns)
        Dim refreshed As Integer = 0
        For i As Integer = 1 To count
            If cad.AvaxControlRefresh(parentIds(i), controlIndices(i)) Then refreshed += 1
        Next
        Return refreshed
    End Function
End Module
```

FitDrawingToWorkspacePage

API reference

FitDrawingToWorkspacePage