DDVue

An ActiveX OCX
to
view and retrieve
DWG/DXF information

DDVue

REFERENCE MANUAL

License Agreement

CADology Limited and Ultimate Software Solutions Limited designed and provides this computer OLE control - ActiveX(DDVue) and documentation as is and licenses its use. The user assumes responsibility for the choice of this program to achieve its intended results, and for the installation, use and results achieved.

DDVue is not, nor has ever been, public domain or free computer software.

Distribution License:
Any versions of DDVue(OCX) may only be distributed with a valid *Distribution License*; this license is obtained on a annual basis from CADology. However, the user may distribute DDVue(OCX) with their application as long as the user is a valid Sustaining or Founder member of the Open Design Alliance.

Limited Warranty:
DDVue is provided without warranty of any kind. The complete risk as to the results produced by this program is assumed by purchaser/user. If the program becomes defective, the purchaser/user assumes the complete cost of all associated damages. CADology Limited or Ultimate Software Solutions Limited shall have no responsibility or liability to any person or company.

The Governing Law
The Governing law of this Agreement shall be that of England, UK.

Acknowledgements
- AutoCAD, AutoCAD 2006, AutoCAD 2005, AutoCAD 2004, AutoCAD 2002, AutoLISP, DWG and DXF are trademarks of Autodesk
- Windows 98, Windows ME, Window NT, Windows 2000 and Windows XP are trademarks of Microsoft
- DWGDirect and DGNDirect are trademarks of Open Design Alliance

Contents

Introduction

Welcome to DDVue, DDVue is a Windows ActiveX OCX, which allows the user to view AutoCAD DWG files inside their applications. DDVue is designed with an Object Model that allows access to some data inside the drawing so it can be updated and extracted.

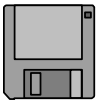
DDVue is available in two versions: -

DDVue View: Viewing of DWG and DXF drawings.

DDVue Developer Edition: This has the same options as above but allows access to data and options to update data.

DDVue has been designed to work with Windows NT 4.0, Windows 2000 and Windows XP.

Installing DDVue - Manual



Minimum Configuration

DDVue ActiveX OCX requires Windows NT 4.0, Windows 2000 or Windows XP with the latest Service Packs, although it should work with older versions of Microsoft Windows 98 and ME.

Installing DDVue Software (all versions)

DDVue has been designed to have minimal files, these are:

DDVue.ocx – CADology

DDVue is available in different compiled version using Microsoft Visual C++; this is to match the version of development software and to reduce the DLL support files.

Microsoft Visual C++ 6.0

MFC42.DLL – Microsoft
MSVCRT.DLL – Microsoft
MSVCRT40.DLL - Microsoft
MSVCP60.DLL - Microsoft

Microsoft Visual C++ 7.1 (.NET 2003)

MFC71.DLL – Microsoft; 7.10.3077.0
MSVCR71.DLL – Microsoft; 7.10.3052.4
MSVCP71.DLL – Microsoft; 7.10.3077.0

Some of these files above you may already have on your system. From the distribution media, please copy DDVue.ocx to a directory of your choice. If you not have the other DLL already present in the SYSTEM directory of your Windows operating system, just copy the DLL files to the Windows system directory.

IMPORTANT: .NET Developers

Please see the Microsoft knowledge base article 326922 titled “Redistribution of the Shared C Runtime Components”, this details that the runtime DLL files should be installed in the application directory rather than system or shared directories.

Once the DLL files are present on your computer, you will need to Register DDVue. See the next paragraph.

Registering the Custom Control

To register the DDVue control on your system, you need a program that comes with the Windows OS called REGSVR32.exe. To run this program, enter the console (or MS-DOS) prompt, change directory to the **DDVue** directory, and key in:

C:\DDVueDev\: ***REGSVR32 DDVue.ocx***

Your operating system will respond that the control has been registered.

If the DDVue.ocx cannot be registered, please contact CADology.

Updating to a newer version of DDVue

To update to a newer version of DDVue, you must first UN-REGISTER the control using the REGSVR32 program or by any other route you know.

C:\DDVueDev\: ***REGSVR32 /U DDVue.ocx***

A dialog message box will confirm if the control has been successfully unloaded.

Important:
It is also wise to remove the DDVue component from your development project and re-import DDVue otherwise some errors may appear.

Distributing DDVue with your Application

You may only distribute this software with your application only if you have purchased the “annual distribution license”. This is included in the first year but after 1 year from purchase you are obliged to purchase the license to distribute.

DDVue requires the following files and they must be included in your install routine when your application is ready, these files can be found on your Development computer.

The following files are required by DDVue and should be included in your set-up routines, but these files are mostly likely to be present anyway.

DDVue is available in 2 builds, these are:
Microsoft Visual C++ 6.0
Microsoft Visual C++ 7.1

DDVue Build: Microsoft C++ 6.0
The following files are to be placed in the Windows SYSTEM directory:
MFC42.DLL – Microsoft Foundation Classes DLL
MSVCRT.DLL – VC Runtime DLL
MSVCRT40.DLL – VC 4.0 Runtime DLL

DDVue Build: Microsoft C++ 7.1 .NET 2003
The following files are to be placed in the application directory, see important note earlier in manual:
MFC71.DLL – Microsoft Foundation Classes DLL
MSVCIRT.DLL – Microsoft C runtime
MSVCR71.DLL – Microsoft C runtime
MSVCP71.DLL – Microsoft C runtime

Control Registration

Don’t forget, DDVue will need to be registered. This can be done in the installation program of your application, your application itself or manually by the user. DDVue is a self-registering ActiveX control as it

supports self-registration by implementing the *DllRegisterServer* and *DllUnregisterServer* functions.

DDVue Hints and Tips

The following is a list of notes and hints that maybe helpful as a user of DDVue.

Unlock Code

Once you have purchased DDVue, you will be given a unlock code, this code must be assigned in the programming source code before a drawing is opened. Please do not given this unlock code to persons outside of your company. Please read the unlock Product method in the DDVue class.

XML Report

The XML report is the best way to access information and data inside the drawing, the XML will report such information as Layers, Layouts and variables.

Modelspace / Paperspace / Layouts

When adding new entities to the drawing, a parameter is needed to tell DDVue where to add the entity. If the entity is to be added to Modelspace, then pass “Modelspace” for the value to the *Location* parameter. If you would like to add the entity to a Layout, just add the layout name in the *Location* parameter.

Search Paths

When loading a drawing in DDVue there will be need to find some external files that are associated with the drawing, these could be fonts files, image files, etc. To assist DDVue in finding these files, please assign a set of directories to search by calling the SetPaths() methods.

Right Mouse Menu

A right mouse click on the DDVue client area will display a small menu for the user to perform zoom and pan operations.

Enumerations

To assist in the options for some of parameters when calling Methods, the following Enumerations have been created.

DWGVersion

Values of allowed DWG version to save in GetDWGVersion, CreateDXF and SaveDrawing.

RenderMode

This allows the developer to chose between software rendering(using GDI) and OpenGL rendering.

RenderType

This defines the different render modes for OpenGL render mode. Some of the render types will only work in OpenGL; these are the non-wireframe display options like shading and hidden line.

DDVueLib Class

DDVue has been created in one base class; looking at the Object Browser that comes with your software development environment can see these classes. These classes are listed below and a full reference follows in this documentation.

In the detailed description of the parameters for the methods, a parameter that is passed to the method where the value is just used is coloured Green. And, parameter coloured Red means that this variable will be set by DDVue for your application to act upon afterwards.

DDVue Class

Overview

This is the main class and is used to load the CAD drawing files into the DDVue environment and then act upon them.

Member Properties

Caching

Defaults to False. This property will enable a Display List which will speed up the displaying of drawings. This will only work when RenderType is OpenGL. When this is set the drawing will take longer to display the first time but after that zoom and pan operations will be faster.

ErrorNumber

This is a read only property that returns the error number for DDVue, further details and error number list can be found later in this manual.

Monochrome

This property will disable any colours in the drawing hence the view will be black and white. This has to be set from your source code. This was added due to colour printers were displaying grey scale for colours.

RecoverIfNeeded

Defaults to False. This property will allow DDVue to perform a recover on the drawing automatically if the DWG/DXF can't be opened. In some cases DDVue will not be able to open a DWG file because of errors inside the DWG. By setting this to True, DDVue will perform a recover and then continue. If this is set, it is NOT recommended that the drawing is saved back out.

RenderMode

This allows the user to switch between different type of rendering that will change the way the view is presented. This is the same as the ShadeMode command in AutoCAD.

RenderType
When the RenderMode is set to OpenGL, this property is set to a value to represent the rendering display of the 3D CAD file.

RightMouseMenu
This property is a boolean and controls if the Right mouse click menu is enabled or disabled. By default this is enabled.

Event Methods

OnClick()
This event will get called when ever the user has clicked on the DDVue control area.

OnDblClick()
This is the same as the OnClick event except it happens when the user performs a double click.

OnMouseDown()
OnMouseDown(X as Long, Y as Long, nButton as Integer)

This event is called when the mouse is moved across the DDVue client area, the X and Y values are the coordinates of the control with 0,0 being the top left hand corner. The ClientToDWG() method can be used to get DWG coordinate X and Y values.

The nButton parameter indicates which mouse button has been clicked, 0 – None, 1 – Left, 2 - Right, 3 – Both.

OnMouseMove()
OnMouseMove(X as Long, Y as Long)

This event give the same X and Y values as described above in OnMouseDown().

OnMouseUp()
OnMouseUp(X as Long, Y as Long, nButton as Integer)

This is the same as OnMouseDown except the button parameter is different. In this event, the nButton parameter indicates which button has been left down when the user has released a button. This is not the same as OnMouseDown().

AuditDrawing

[form.]control.AuditDrawing <DWGfilename> <recoveredDWGfilename> <numberEntities> <numberErrors> <numberFixes>
DWGfilename: Filename of DWG to audit
recoveredDWGfilename: Filename of DWG that has been recovered
numberEntities: Number of entities
numberErrors: Number of errors found
numberFixes: Number of fixes completed
Return error number
(3)
This method will audit a drawing and recover the drawing to another drawing filename. This method will report back the number of entities in the drawing, the number of errors that have been found and the number of errors that have been fixed.

AttributeUpdateByHandle

[form.]control.AttributeUpdateByHandle <attribute value> <attribute handle>
attribute value: The new value for the attribute
attribute handle: A handle of the attribute to update
Return Boolean
(2),(3)
This allows the literal value for an Attribute entity to be updated. To identify the attribute, the method requires the entity handle of the attribute entity. Access to fonts files may be required to calculated the correct starting place for the text entity hence you must use the SetPaths() method to inform DDVue where to find the fonts files

BuildVersion

[form.]control.BuildVersion <build version> <release> <major> <minor> <build>
build version: Full version and build number in a string
release: Release component of the version
major: Major component of the version
minor: Minor component of the version
build: Build component of the version
Return Void
(1),(2),(3)
This will return the version of the DDVue control is two different ways. The first parameter is a string and will return the complete version. The remaining parameter will return the components of the version. The string version might return “1.14.4.28” and the four components in order are *Release*, *Minor*, *Major* and *Build*.

Open

[form.]control.Open <DWG/DXF filename> [DWG password] [Zoom Extents]
DWG/DXF filename: A filename and path to a drawing file
DWG password: A password for the drawing file
ZoomExtents: Allow the drawing to be opened at the extents
Return Boolean
(1),(2),(3)
This is used to open (load) the CAD filename into the DDVue component so that information can then be accessed.

It is very important that each opened drawing is closed and calling the Close method performs this.

From AutoCAD 2004 drawing files in DWG format a password can be assigned to the file, the password must be passed as a parameter so that DDVue can open the drawing.

ClientToDWG

[form.]control.ClientToDWG <Client X> <Client Y> <DWG X> <DWG Y>
Client X: The X coordinate of the client area
Client Y: The Y coordinate of the client area
DWG X: The DWG X coordinate
DWG Y: The DWG Y coordinate
Return void
(1),(2),(3)
This allows the user to convert the coordinate in the client area of the DDVue control to DWG world position coordinates. The client coordinates are passed as doubles and double values for the DWG coordinates are passed back. See also DWGToClient().

Close

[form.]control.Close
Return Boolean
(1),(2),(3)
This must be called when an opened drawing is no longer needed, this will release the memory that has been allocated by DDVue for the drawing.

ContextMenuCommand

[form.]control.CcontextMenuCommand <Command number>
Command number: A enum for the command to be called
Return Boolean
(1),(2),(3)
This allows your application to call the command defined in the Right Click menu in DDVue. The commands in the menu are numbered and are listed in a enumeration. 1 – Pan, 2 – Zoom Extents, 3 – Zoom Previous, 4 - Zoom Window.

This is useful in the PocketPC environment, where Right Mouse menu is not an option. Or perhaps, you want your own menu and need to call the in-built functions in DDVue.

CopyToClipboard

[form.]control.CopyToClipboard
Return Boolean
(1),(2),(3)
This will enable the drawing currently displayed to be copied to the Windows clipboard and used in another application that can access the clipboard.

CreateArc

[form.]control.CreateArc <Handle> <Location> <XCentre> <YCentre> <ZCentre> <Radius> <Start angle> <End angle>
Handle: The value of the entity handle if created succesfully
Location: The layout name or modelspace name fwhere the entity will be created
X,Y,ZCentre: Coordinates of the centre of the arc
Radius: Radius of the arc
Start angle: Start angle for the arc
End angle: End angle of the arc
Return Boolean
(2),(3)
This will create a arc entity. The arc can be created in a layout by passing the layout name in the *Location* parameter or if required the line is to be placed in modelspace, then pass “Modelspace” in the *Location* parameter. The start and end angles are in degrees and 0 degrees in the X-Axis.

CreateCircle

[form.]control.CreateCircle <Handle> <Location> <X1> <Y1> <Z1> <Radius>
Handle: The value of the entity handle if created succesfully
Location: The layout name or modelspace name fwhere the entity will be created
X1,Y1,Z1: Coordinates of the centre of the circle
Radius: Radius of the circle
Return Boolean
(2),(3)

This will create a circle entity using the coordinates passed. The circle can be created in a layout by passing the layout name in the *Location* parameter or if required the line is to be placed in modelspace, then pass “Modelspace” in the *Location* parameter.

The handle of the created circle entity will be return, this is useful if the entity header values need changing.

The entity will be created using the default values, i.e Layer 0 etc.

CreateDXF

[form.]control.CreateDXF <DXF filename> <DXF version> <DXF precision>
DXF filename: A filename and path where the DXF will be created
DXF version: Version of the DXF file required
DXF precision: Precision of the DXF file required
Return Boolean
(3)

This will create a DXF file from the drawing opened in DDVue, there are parameters to control the DXF version and also the precision required. If the precision is passed as 0, then a binary DXF file will be created instead of a ASCII one.

CreateHatchFromPolyline

[form.]control.CreateHatchFromPolyline <Handle> <Polyline handle> <Pattern name> <Scale Factor>
Handle: The value of the entity handle if created succesfully
Polyline handle: The handle of the polyline that will be used to create the hatch boudary
Patter names: This is the name of the pattern required
Scale Factor: This is the scale factor applied to the hatching
Return Boolean
(2),(3)

This will create a hatch entity based on the boundary of the polyline passed by the handle. The hatch pattern will retain the same Layer, Colour and Layout as the Polyline. The pattern name must be one of the predefined patterns found in the DWG structure.

CreateLine

[form.]control.CreateLine <Handle> <Location> <X1> <Y1> <Z1> <X2> <Y2> <Z2>
Handle: The value of the entity handle if created succesfully
Location: The layout name or modelspace name fwhere the entity will be created
X1,Y1,Z1,X2,Y2,Z2: Coordinates of the Line entity
Return Boolean
(2),(3)

This will create a line entity using the coordinates passed. The line can be created in a layout by passing the layout name in the *Location* parameter or if required the line is to be placed in modelspace, then pass "Modelspace" in the *Location* parameter.

The handle of the created line entity will be return, this is useful if the entity header values need changing.

The entity will be created using the default values, i.e Layer 0 etc.

CreateLWPolyline

[form.]control.CreateLWPolyline <Handle> <Location> <vPoly Points>
Handle: The value of the entity handle if created succesfully
Location: The layout name or modelspace name fwhere the entity will be created
vPoly Points: Coordinates of the polyline entity
Return Boolean
(2),(3)

This will create a lightweight polyline entity using the coordinates passed. The polyline can be created in a layout by passing the layout name in the *Location* parameter or if required the line is to be placed in modelspace, then pass "Modelspace" in the *Location* parameter.

The <vPolyPoints> parameter is a array of 2D points that create the vertices of the LWPolyline. The coordinates are 2D because the entity is planar, sample VB code below.

```
Dim sHandle As String
Dim dblArray(3, 1) As Double

Dim vPtr As Variant

dblArray(0, 0) = 0 ' Point 1
dblArray(0, 1) = 0
dblArray(1, 0) = 10 ' Point 2
dblArray(1, 1) = 0
dblArray(2, 0) = 10 ' Point 3
```

```
dblArray(2, 1) = 10
dblArray(3, 0) = 0 ' Point 4
dblArray(3, 1) = 0

vPtr = dblArray ' Assign Array to Variant

Dim bRes As Boolean
bRes = DDVue1.CreateLWPolyline(sHandle, "MODELSPACE", vPtr)
```

CreateText

[form.]control.CreateText <Handle> <Location> <Text value> <Text style> <X> <Y> <Z> <Height> <Rotation angle> <Justification>

Handle: The value of the entity handle if created succesfully
Location: The layout name or modelspace name fwhere the entity will be created
Text value: The value for new text entity
Text style: The text style required for the new text entity
X,Y,Z: Coordinates of the Line entity
Height: Height of the text
Rotation angle: Rotation angle for the text entity
Justification: Justification for the text based on the X,Y,Z coordinate

Return Boolean
(2),(3)

This will create a text entity using the information passed.

The <X,Y,Z> parameters are a series of double variables for each X, Y and Z value. The angle is expected in Degrees. The Style name must be present in the drawing, if not it will default to using the “Standard” text style. The just parameter controls the justification of the text placement and can be one of the following:
L, C, R, TL, TC, TR, ML, MC, MR, BL, BC, BR
Where L is Left, R is Right, T is Top, M is Middle and B is Bottom.

DeleteEntity

[form.]control.DeleteEntity <Entity handle>

Entity hanlde: Handle of the entity to be deleted from the drawing

Return Boolean
(3)

This method will delete entities from the drawing identified by the handle. Any entities can be delete so you must make sure you have a valid entity handle. True or False will be returned depending if the delete sequence has been successful.

Display

[form.]control.Display
Return Boolean
(1),(2),(3)
This method is used when the display render mode has changed and needs to be updated. Example code:

```
DDVue1.Render = OpenGL
DDVue1.RenderMode = kGouraudShaded
DDVue1.Display
```

DWGToClient

[form.]control.DWGToClient <DWG X> <DWG Y> <Client X> <Client Y>
DWG X: The X coordinate in the DWG
DWG Y: The Y coordinate in the DWG
Client X: The client area X coordinate
Client Y: The client area Y coordinate
Return void
(1),(2),(3)
This allows the user to convert a coordinate in the drawing to it's position on the client area of the DDVue control. See ClientToDWG() also.

Explode

[form.]control.Explode <Entity handle>
Entity handle: Handle of entity to explode
Return Boolean
(2),(3)
This method is used to explode an INSERT entity(block reference) to lower order graphics that defined the block. This designed for MODELSPACE objects only.

GetDWGVersion

[form.]control.GetDWGVersion <DWG filename>
DWG filename: Filename and path of DWG to get version from
Return Long
(1),(2),(3)
This method is used to get the AutoCAD DWG version of a drawing file. The return value will indicate the version of the DWG file. The enumuration DWGVERSION can be used to identify the return value to the version.

Example code:

```
Dim IDWGVersion As Long
IDWGVersion = DDVue1.GetDWGVersion(OpenDwg.FileName)
Select Case IDWGVersion
   Case ddvuelib.DWGVERSION.AutoCAD2000
      MsgBox "AutoCAD 2000 to AutoCAD 2002 drawing"
   Case ddvuelib.DWGVERSION.AutoCAD2004
      MsgBox "AutoCAD 2004 to AutoCAD 2005 drawing"
End Select
```

GetEntityHeaderByHandle

[form.]control.GetEntityHeaderByHandle <Entity handle> <Colour> <Layer name> <Linetype> <Line weight> <Extrusion X> <Extrusion Y> <Extrusion Z>

Entity handle: Handle of entity to explode
Colour: Colour of the entity
Layer name: Layer name of entity
Linetype: Linetype of the entity
Lineweight: Lineweight of entity
Extrusion X: X value of extrusion vector
Extrusion Y: Y value of extrusion vector
Extrusion Z: Z value of extrusion vector

Return Boolean

(2),(3)

This method will return the entity header data of a entity located by it's entity handle. The extrusion will only be returned by planar entities.

GetEntityRangeByHandle

[form.]control.GetEntityRangeByHandle <Entity handle> <X min> <Y min> <Z min> <X max> <Y max> <Z max>

Entity handle: Handle of entity to explode
X min, Y min, Z min: Minimum coordinate point of entity range
X max, Y max, Z maz: Maximum coordinate point of entity range

Return Boolean

(2),(3)

This method will return the range of the entity selected by the handle. The range of the entity is the bounding box that will encompass the graphical extents..

GetPoint

[form.]control.GetPoint(<X,Y,Z array> <mode> <X,Y, Z base array>)

X,Y,Z Array: A variant array of 3 that will hold the point

mode: The type of rubber banding required

X,Y,Z base array: A base point used for rubber banding initial point

Return Boolean

(3)

This method allow the user to select a point from the control window. The point selected by the user is returned in the parameters X,Y and Z of double data type in an variant array.

The mode variable allows for different types of rubber banding, the options are:
0 – No rubber banding
1 – Line rubber banding using the base X, Y and Y as the start point
2 – Rectangular rubber banding using the base X, Y and Z as the start point. There is a enum in the DDVue control for rubber banding.

Sample Visual Basic:

```
Dim basePt(2) As Double
Dim userPt(2) As Double
Dim vBasePt As Variant
Dim vUserPt As Variant

VBasePt = basePt
VUserPt = userPt
DDVue1.GetPoint vBasePt, rbMode, vBasePt
```

GetPolylineAreaPerimeterByHandle

[form.]control.GetPolylineAreaPerimeterByHandle(<Entity handle> <Area> <Perimeter>)

Entity handle: Handle of the closed polyline entity

Area: Area of the polyline

Perimeter: Perimter of the polyline

Return Boolean

(2), (3)

This method will calculated the area and perimeter length of a 2D closed polyline entity. The polyline is found in the drawing by the handle.

GetRGB

[form.]control.GetRGB(<colour index> <red> <green> <blue>)
Return Boolean
(3)
This method will access the colours defined in the RGB table. Thistable is used to display the colours in the AutoCAD drawing. The colour index parameter indicates the entry in the table and the Red, Green and Blue parameters will be set between 0 to 255. See also SetRGB method.

GetTimeDWGEdited

[form.]control.GetTimeDWGEdited <Edit duration>
Return Boolean
Edit duration: Contains edit duration of drawing
(1),(2),(3)
This will retrieve the time that the DWG file has been edited inside an AutoCAD drawing editor session. This method uses the TDINDWG system variable. The format returned is:

HighlightEntity

[form.]control.Highlight <entity handle>
Return Boolean
Entity handle: Handle of the entity to highlight
(1),(2),(3)
This will highlight the entity identified by the handle parameter. Only *main* entities can be highlighted, hence Attributes can't be highlighted as they are sub-entities of the INSERT entity.

To un-Highlight an entity, you can call the ZoomIn method with a value of 1.0, this will redraw the screen.

InsertBlock

[form.]control.InsertBlock <Handle> <Location> <Block name> <X> <Y> <Z> <Xs> <Ys> <Zs> <Rotation angle>

Handle: The value of the entity handle if created succesfully
Location: The layout name or modelspace name fwhere the entity will be created
Block name: Block name to insert in the drawing
X,Y,Z: Coordinates of the Insert entity
Xs,Ys,Zs: Scale factors for each axisy
Rotational Angle: Rotation angle
Return Boolean
(2),(3)

This method is used to add a block already defined in the block table to the drawing. If the block name is not in the drawing, then use the InsertDWG() method to add the external DWG.

The sample below will load an external drawing and then place in the current drawing

```
If (DDVue1.InsertDWG("c:\parent.dwg") = True) Then
   Dim handle As String
   DDVue1.InsertBlock handle, "MODELSPACE", "parent", 0, 0, 0, 1, 1, 1, 0
End If
```

ImportDWF

[form.]control.ImportDWF <DWF filename> <Background colour>
DWF filename: Filename and path of external DXF to import
Background Colour: This is the background colour of the DWF
Return Boolean
(3)

This method will import an external DWF file and add it to the opened drawing in DDVue. The Background colour parameter allows the user to adjust the background colour in DDVue the same as the DWF, if needed.

The DWF file will be import to fit a A4 papersize in portrait.

InsertDWG

[form.]control.InsertDWG <DWG filename>
DWG filename: Filename and path of external DWG to import
Return Boolean
(2),(3)

This method will import an external DWG file and add it to the opened drawing in DDVue, this would be used with the InsertBlock() method to add blocks to the drawing.

PanByDirection

[form.]control.PanByDirection <direction> <distancePecentage>
direction: U,D,L.R,N,E,S,W
distance: distance to pan by percentage
Return Boolean
(1),(2),(3)
This method will pan the drawing in an orthogonal direction, the direction is selected by the first character of Up, Down, Left, Right, North, East, South, West.

The distance is defined by a percentage, where 100% is the current screen width or height, so if you wanted to pan by a quarter then 25 would be the valued passed.

PrintDraft

[form.]control.PrintDraft <showPrintDialog> <monochrome> <current view>
showPrintDialog: True or False show the printing dialog
monochrome: True or False to allow for monochrome printing
current view: True or False to allow for Modelspace current view
Return Boolean
(1),(2),(3)
This method allows DDVue to print the current opened drawing. The user will be presented with the print dialog box where print settings can be changed.

PrintDraftLayouts

[form.]control.PrintDraftLayouts [layout names]
layout names: List of specific layout names
Return Boolean
(1),(2),(3)
This method will print all the layouts in the drawing to the default printer, there is an optional parameter which allows the parent application to list the Layout name to be printed instead of all of them. If this is used then commas must separate the layout list.

Regen

[form.]control.Regen
Return void
(1),(2),(3)
This method will instruct DDVue to redisplay the drawing, this method is not always needed but may be useful if you are overwrite your own graphics on the DDVue client area.

RenameTableEntry

[form.]control.RenameTableEntry <table> <oldName> <newName>
tabke: Table enumeration
oldName: Old name to be changed
newName: New value for table
Return Boolean
(1),(2),(3)
This method will allow a table entry to be rename, the oldName paramter is the name currently in the name and this used to locate the entry.

The following table are allowed: Layers. Contact CADology is you require another table be accessed by this method.

RestoreView

[form.]control.RestoreView <View name>
View name: Name of view to restore and display
Return Boolean
(1),(2),(3)
This method will set the view of the drawing to a saved view already defined in the drawing file. Remember view can be saved in Modelspace and Paperspace, hence you are not able to restore a Modelspace view if you are in Paperspace.

SaveAsBMP

[form.]control.SaveAsBMP <BMP filename>
BMP filename: Filename of the BMP to be created
Return Boolean
(1),(2),(3)
This allows the current opened view of the drawing(DWG/DXF) to be saved to a Windows BMP file. The BMP will be based on the client area that the DDVue control is using.

There are 2 optional parameters to define the height and width of your BMP, so you could use the format:

SaveAsBMP “Filename.bmp”, 800, 600

SaveAsDWF

[form.]control.SaveAsDWF <DWF filename>
DWF filename: Filename of the DWF to be created
Return Boolean
(1),(2),(3)
This allows the current opened view of the drawing to be saved to a DWF file.

More details to follow when fully implement.

SaveAsSVG

[form.]control.SaveAsSVG <SVG filename>
SVG filename: Filename of the SVG to be created
Return Boolean
(1),(2),(3)
This allows the current opened view of the drawing to be saved to a SVG file.

More details to follow when fully implement.

SaveDrawing

[form.]control.SaveDrawing <DWG filename> <DWG version>
DWG filename: Filename of the DWG to be created
DWG version: Version of the DWG to be created
Return Boolean
(1),(2),(3)
This method is used to create a DWG file from the drawing currently opened in DDVue. The allowed DWG versions are:

-1	Same version as loaded DWG file
2004	DWG 2004
2000	DWG 2000
14	DWG Release 14

Select

[form.]control.Select <Entity handles>
Entity handles: Handles of any entities selected by user
Return Boolean
(3)
This options allows the user to select a point on the screen and the entities in the small region of the point clicked will be returned. The data returned is passed back in groups of two, delimited by a comma. The first of the data is the handle and the next is type of entity. So you might get back:

14,AcDbLine,16,AcDbText

From this we know that handle 14 is a Line and 16 is text entity.

Note: When working in Paperspace(Layouts) you will also get back AcDbLayout and AcDbViewport; these can be ignored and must never be deleted.frequently

SetEntityColorByHandle

[form.]control.SetEntityColorByHandle <Entity handle> <Colour>
Entity handle: Handle of the entity to set the colour
Color: Colour value to set the entity
Return Boolean
(1),(2),(3)
This will change the colour of the entity located by it's handle.

SetEntityLayerByHandle

[form.]control.SetEntityLayerByHandle <Entity handle> <Layer name>
Entity handle: Handle of the entity to set the layer
Layer: Layer value to set the entity
Return Boolean
(1),(2),(3)
This will change the layer of the entity located by it's handle. A layer will be created using the default values if the layer name does not exist already in the drawing.

SetEntityVisibilityByHandle

[form.]control.SetEntityVisibiltyByHandle <visibility> <Entity handle>
Visibility: True or False to set visibility
Entity handle: Handle of the entity to set the layer
Return Boolean
(1),(2),(3)
This method will update the value of an entity visibility selected by the object handle. A True or False is returned based whether the entity is found in the Model / Paper space or a Layout.

SetLayerColor

[form.]control.SetLayerColor <Layer name> <Colour>
Layer name: Layer name to apply change
Colour: Colour value to set the layer
Return Boolean
(1),(2),(3)
This allows a colour value to be be assigned to a Layer in the Layer table.

SetLayerState

[form.]control.SetLayerState <Layer name> <State>

Layer name: Layer name to apply change

State: State to change the layer to

Return Boolean

(1),(2),(3)

This allows the layers to be turned on and off in the displaying of the drawing. The *State* parameter can be set to ON or OFF.

SetLayout

[form.]control.SetLayout <Layout>

Layout: Layout to display

Return Boolean

(1),(2),(3)

Once a drawing is being displayed in DDVue, this method will allow the application to display a different Layout or modelspace, Modelspace is displayed by passing “Model” instead of a Layout name.

SetPaths

[form.]control.SetPaths <Search paths>

Search paths: List of paths to find associated files

Return Boolean

(1),(2),(3)

This method will instruct DDVue to search for any require support files it needs when loading a drawing, this is no so important for US based DWG but European alphabets will need their own SHX files to display the extra characters correctly. The string variable passed can be like the PATH statement in DOS / Windows NT and have multiple directories separated by the ; symbol.

SetRGB

[form.]control.SetRGB(<colour index> <red> <green> <blue>)

Return Boolean

(3)

This method will update a colour defined in the RGB table. The colour index parameter indicates the entry in the table and the Red, Green and Blue parameters can be set to 0 to 255. See also GetRGB method.

ShowThumbnail

[form.]control.ShowThumbnail(<Bitmap type>)
Bitmap type: Not used, just pass 0
Returns: TRUE or FALSE
(1),(2),(3)
This method will display the bitmap image of the opened DWG file in the OCX control window on your form.

ShowThumbnailInDWG

[form.]control.ShowThumbnailInDWG(<DWG filename> <Bitmap type>)
Bitmap type: Not used, just pass 0
Returns: TRUE or FALSE
(1),(2),(3)
This method will display the bitmap image of a DWG file.

TextUpdateByHandle

[form.]control.TextUpdateByHandle <Entity handle> <text value>
Entity handle: A handle of the text entity to update
Text value: The new value for the text entity
Return Boolean
(3)
This allows the literal value for an Text entity to be updated. To identify the text, the method requires the entity handle of the text entity. Access to fonts files may be required to calculated the correct starting place for the text entity hence you must use the SetPaths() method to inform DDVue where to find the fonts files.

ViewRotateXAxis
ViewRotateYAxis
ViewRotateZAxis

[form.]control.ViewRotateXAxis <Rotation angle>
Rotation angle: Rotation angle in degrees
Return void
(1),(2),(3)
This allows the view to be rotated when viewing 3D models, there is a method to rotate by a angle in degrees around each of the 3 axis; X, Y and Z.

XMLReport

[form.]control.XMLReport <XML filename> <Attributes> <Attribute definitions> <References> <Layers> <Layouts> <System variables> <Images> <text literals>

XML filename: Filename of XML file you would like DDVue to create.
Attributes: Set to 1 to export Attribute data to XML.
Attribute definitions: Set to 1 to export Attribute definitions data to XML.
References: Set to 1 to export Reference data to XML.
Layers: Set to 1 to export Layer data to XML.
Layouts: Set to 1 to export Layouts data to XML.
Sysem variables: Set to 1 to export a selection of system variables to XML.
Images: Set to 1 to export image data to XML.
Text literals: Set to 1 to export text literals to XML.
(1),(2),(3)

This method is used to export data inside the drawing to a XML file passed by the calling application. The XML filename will be deleted and a new data from the drawing will be added.

The parameters control which data is exported to the XML file, simply pass a 1 to output the data or 0 if it's not needed in the XML file.

Text Extraction

For the text option, both Text and MText entities will have the literals extracted, the MText will have the formatting show. The text values can be updated using the UpdateTextByHandle method.

If more system variables are required, please email CADology with required variables to be added.

Attribute Extraction

The XML report is very flexible for getting access to attribute values but there are some key points you will need to know about getting the correct attribute data. In the XML for the INSERT element, there is information that tells us the number of attributes and the number of constant attributes. Constant Attribute values are NOT stored following the INSERT entity and hence will not be found in the XML in the <Attribute> element. If there is a constant attribute you will need to then visit the <AttributeDefinition> part of the XML file to get access to the constant value. The normal attributes will follow in the XML.

The user may have used the MINSERT command to place a block with attributes in the drawing, for this case there is the <rows> and <columns> element to inform you of the details.

ZoomEntityExtents

[form.]control.ZoomEntityExtents <entity handle> <zoom factor>
entity handle: Handle of entity to zoom to
Return Void
(1),(2),(3)
This will zoom the drawing so that the entity handle passed will be displayed in DDVue.

Note: Currently the Zoom Factor option is not working.

ZoomExtents

[form.]control.ZoomExtents
Return Void
(1),(2),(3)
This will zoom the drawing so that all of the drawing is displayed within the client area of the DDVue control.

ZoomIn

[form.]control.ZoomIn <factor>
factor: A factor to zoom by
Return Void
(1),(2),(3)
This will zoom the drawing by the factor passed.

ZoomOut

[form.]control.ZoomOut <factor>
factor: A factor to zoom by
Return Void
(1),(2),(3)
This will zoom the drawing by the factor passed.

ZoomPrevious

[form.]control.ZoomPrevious
Return Boolean
(1),(2),(3)
This will instruct DDVue to redisplay the last zoom position, if there is one.

Error Numbers and Descriptions

The following is a list of error numbers and possible reason:

30	Can't open the file
131	Drawing is encrypted or wrong password has been used

XML Definition

The XML file has a main section called DToolsX and then further sections, these are:

<DrawingInformation>
<Attributes>
<AttributeDefinition>
<ExternalReference>
<Layers>
<Layouts>
<Variable>
<Images>
<Text>

Sample XML file; This sample may not show the latest version.

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
- <DToolsX>
  - <DrawingInformation>

      <DatabaseFilename>c:\parent.dwg</DatabaseFile
      name>
    <ApproxNumObjects>137</ApproxNumObjects>
    <NumberOfSaves>0</NumberOfSaves>
    <OriginalFileVersion>AC1015</OriginalFileVersion>
    <LastSavedVersion>AC1018</LastSavedVersion>

      <OriginalFileSavedByVersion>AC1015</OriginalFile
      SavedByVersion>
    <Hyperlinkbase />
    <Title />
    <Subject />
    <Author />
    <Comments />
    <Keywords />
    <LastSavedBy />
    <Custom />
  </DrawingInformation>
  - <Attributes>
    - <Block name="*Model_Space">
        <LayoutName name="Model" />
      - <Insert name="ref">
          <LayerName>0</LayerName>
        </Insert>
      - <Insert name="ref">
```

```
      <LayerName>0</LayerName>
    </Insert>
  - <Insert name="ref">
      <LayerName>0</LayerName>
    </Insert>
  - <Insert name="db">
      <LayerName>0</LayerName>
    - <Attribute>
        <Handle>E1</Handle>
        <ElementId>225</ElementId>

          <AttributeTag>NAME2</Attribute
          Tag>

          <AttributeValue>XXXWWXX</Att
          ributeValue>
      </Attribute>
    - <Attribute>
        <Handle>E2</Handle>
        <ElementId>226</ElementId>
        <AttributeTag>NAME</AttributeTag>

          <AttributeValue>XXXWWXX</Att
          ributeValue>
      </Attribute>
    </Insert>
  - <Insert name="desk">
      <LayerName>0</LayerName>
    - <Attribute>
        <Handle>137</Handle>
        <ElementId>311</ElementId>
        <AttributeTag>ROOM</AttributeTag>

          <AttributeValue>Bedroom</Attrib
          uteValue>
      </Attribute>
    </Insert>
  </Block>
- <Block name="*PAPER_SPACE">
    <LayoutName name="Dawson" />
  </Block>
- <Block name="*PAPER_SPACE0">
    <LayoutName name="Bunn" />
  - <Insert name="db">
      <LayerName>0</LayerName>
    - <Attribute>
        <Handle>F7</Handle>
        <ElementId>247</ElementId>
```

```
                    <AttributeTag>NAME2</Attribute
                    Tag>

                    <AttributeValue>LAYOUT2</Attri
                    buteValue>
                </Attribute>
              - <Attribute>
                    <Handle>F8</Handle>
                    <ElementId>248</ElementId>
                    <AttributeTag>NAME</AttributeTag>

                    <AttributeValue>LAYOUT2</Attri
                    buteValue>
                </Attribute>
            </Insert>
        </Block>
        <Block name="ref" />
        <Block name="xdata" />
        <Block name="db" />
        <Block name="telephone" />
      - <Block name="desk">
          - <Insert name="telephone">
                <LayerName>0</LayerName>
              - <Attribute>
                    <Handle>130</Handle>
                    <ElementId>304</ElementId>

                    <AttributeTag>TELEPHONE</Attri
                    buteTag>

                    <AttributeValue>0737</Attribute
                    Value>
                </Attribute>
            </Insert>
        </Block>
    </Attributes>
  - <AttributeDefinitions>
      - <Block name="*Model_Space">
          - <AttributeDefinition>
                <Handle>D4</Handle>
                <ElementId>212</ElementId>
                <AttributeTag>NAME2</AttributeTag>

                <AttributePrompt>name2</AttributePro
                mpt>
            </AttributeDefinition>
          - <AttributeDefinition>
                <Handle>D7</Handle>
                <ElementId>215</ElementId>
```

```
            <AttributeTag>NAME2</AttributeTag>
                <AttributePrompt>name2</AttributePro
                mpt>
          </AttributeDefinition>
        </Block>
        <Block name="*PAPER_SPACE" />
        <Block name="*PAPER_SPACE0" />
        <Block name="ref" />
        <Block name="xdata" />
      - <Block name="db">
        - <AttributeDefinition>
            <Handle>DC</Handle>
            <ElementId>220</ElementId>
            <AttributeTag>NAME2</AttributeTag>
                <AttributePrompt>name2</AttributePro
                mpt>
          </AttributeDefinition>
        - <AttributeDefinition>
            <Handle>DD</Handle>
            <ElementId>221</ElementId>
            <AttributeTag>NAME</AttributeTag>
            <AttributePrompt>name</AttributePrompt>
          </AttributeDefinition>
        </Block>
      - <Block name="telephone">
        - <AttributeDefinition>
            <Handle>118</Handle>
            <ElementId>280</ElementId>
            <AttributeTag>TELEPHONE</AttributeTag>
            <AttributePrompt />
          </AttributeDefinition>
        </Block>
      - <Block name="desk">
        - <AttributeDefinition>
            <Handle>12C</Handle>
            <ElementId>300</ElementId>
            <AttributeTag>ROOM</AttributeTag>
            <AttributePrompt />
          </AttributeDefinition>
        </Block>
      </AttributeDefinitions>
    - <ExternalReferences>
      - <Reference name="ref">
          <PathName>C:\ref.dwg</PathName>
          <Unloaded>TRUE</Unloaded>
        </Reference>
      - <Reference name="xdata">
```

```
        <PathName>C:\xdata.dwg</PathName>
        <Unloaded>FALSE</Unloaded>
      </Reference>
    </ExternalReferences>
  - <Layers>
    - <LayerName name="0">
        <Display>On</Display>
      </LayerName>
    - <LayerName name="DEFPOINTS">
        <Display>Off</Display>
      </LayerName>
    - <LayerName name="ref|Dawson">
        <Display>On</Display>
      </LayerName>
    - <LayerName name="Level 51">
        <Display>On</Display>
      </LayerName>
    - <LayerName name="Level 52">
        <Display>On</Display>
      </LayerName>
    - <LayerName name="Level 53">
        <Display>On</Display>
      </LayerName>
    - <LayerName name="Level 54">
        <Display>On</Display>
      </LayerName>
    - <LayerName name="Level 55">
        <Display>On</Display>
      </LayerName>
    - <LayerName name="Level 57">
        <Display>On</Display>
      </LayerName>
    - <LayerName name="Blue">
        <Display>On</Display>
      </LayerName>
    </Layers>
  - <Layouts>
    - <LayoutName name="Bunn">
        <TabOrder>2</TabOrder>
      </LayoutName>
    - <LayoutName name="Dawson">
        <TabOrder>1</TabOrder>
      </LayoutName>
    - <LayoutName name="Model">
        <TabOrder>0</TabOrder>
      </LayoutName>
    </Layouts>
  - <Variables>
    - <TILEMODE>
```

```
        <Value>1</Value>
      </TILEMODE>
    - <EXTMIN>
        <X>-61.131838</X>
        <Y>-0.009063</Y>
        <Z>-0.001526</Z>
      </EXTMIN>
    - <EXTMAX>
        <X>360.293491</X>
        <Y>268.639085</Y>
        <Z>0.000000</Z>
      </EXTMAX>
    - <PEXTMIN>
        <X>24.361804</X>
        <Y>20.422867</Y>
        <Z>20.422867</Z>
      </PEXTMIN>
    - <PEXTMAX>
        <X>219.450667</X>
        <Y>183.889628</Y>
        <Z>183.889628</Z>
      </PEXTMAX>
  </Variables>
</DToolsX
```

Wanted : New Methods and Properties

Suggestions:
If you require or have an idea for a new method or property, please contact CADology by email with details. If your suggestion is included, then the latest version will be shipped free of charge.

Please contact CADology with your requirements.

Glossary

Entity Handle

Each entity is a AutoCAD drawing has a unique identifier, this is a hexadecimal number, which is stored in a character string in DDVue. MicroStation DGN drawings also have these handles but are called Element ID instead.

Technical Support

The latest version of DDVue(Software, revision history and this manual) can be found and downloaded from the CADology web site on Internet; www.CADology.com

For technical support, please contact CADology either by :

Electronic Mail :
techsupp@CADology.com

Voice :
(International) 44 (0)20 8786 7774

Fax :
(International) 44 (0)20 8786 7775

Post :
CADology Limited
Meach House,
71 Nonsuch Walk,
Cheam, Surrey. SM2 7LF
United Kingdom