Loading pictures with ActiveX and VBScript

Dynamically Loading Pictures on a Screen Using an ActiveX Control and VBScript

Implementation Specifications or Requirements

Category	Item	
Software	IWS Version:	6.1 and later
	Service Pack:	N/A
	Windows NT/2000/XP:	WinXP/2000/NT
	Windows CE:	No
	Web Thin Client:	N/A
Equipment	Panel Manufacturer:	N/A
	Panel Model:	N/A
	Other Hardware:	N/A
	Comm. Driver:	All
	Controller (e.g.: PLC)	All
	Application Language:	N/A
Software Demo Application	N/A	

.

Summary

Indusoft Web Studio (IWS) provides methods to load static graphic objects onto a display screen. This note describes how to use the Microsoft PictureClip ActiveX Control (OCX) with IWS VBScripting to dynamically load pictures on a screen.

Discussion

InduSoft Web Studio (IWS) currently provides a couple methods for displaying graphic objects on a display screen. These are:

- Loading a static image at runtime as specified in the Screen Attributes Dialog Box.
- Copying graphic objects (e.g. from Microsoft Paint) and pasting them on an IWS screen. These graphic objects can then be assigned IWS controls (e.g. Position (including visibility), Size, and Command). This is done using the IWS Paste Link function (**Edit → Paste Link).** This function does not work with Windows CE.

However, neither of these methods allows a file to be dynamically loaded into a picture object at runtime. One method of accomplishing this is to use an ActiveX control (OCX) controlled by a VBScript routine.

ActiveX Controls are based on Microsoft's COM technology and provide added functionality in a modular, structured manner. IWS is an ActiveX container, allowing interoperability between the IWS development & runtime environment and an ActiveX Control's Properties, Methods and Events. Put simply, an ActiveX Control can be added to IWS to give added functionality, with the ActiveX Control placed under InduSoft Web Studio's control.

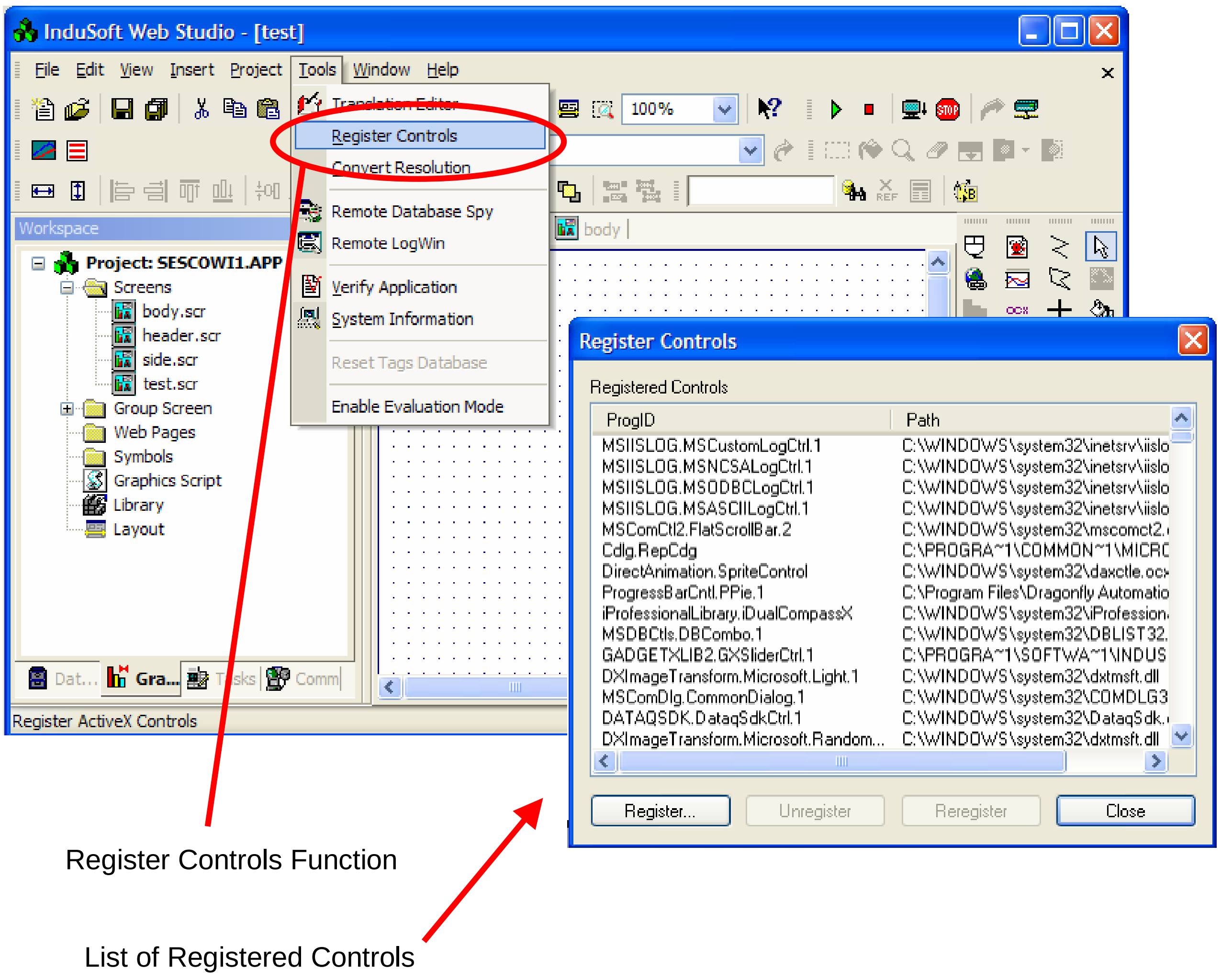
Although InduSoft does not supply an ActiveX Control for this function, there are several 3rd party ActiveX Controls that can be used. The ActiveX Control used in this application is Microsoft's PictureClip ActiveX

Control, originally supplied with Visual Studio 6 and Microsoft Visual Basic 6. The file containing this control is **PICCLP32.OCX**.

There are a few steps you need to do in order to use an ActiveX Control. These steps are:

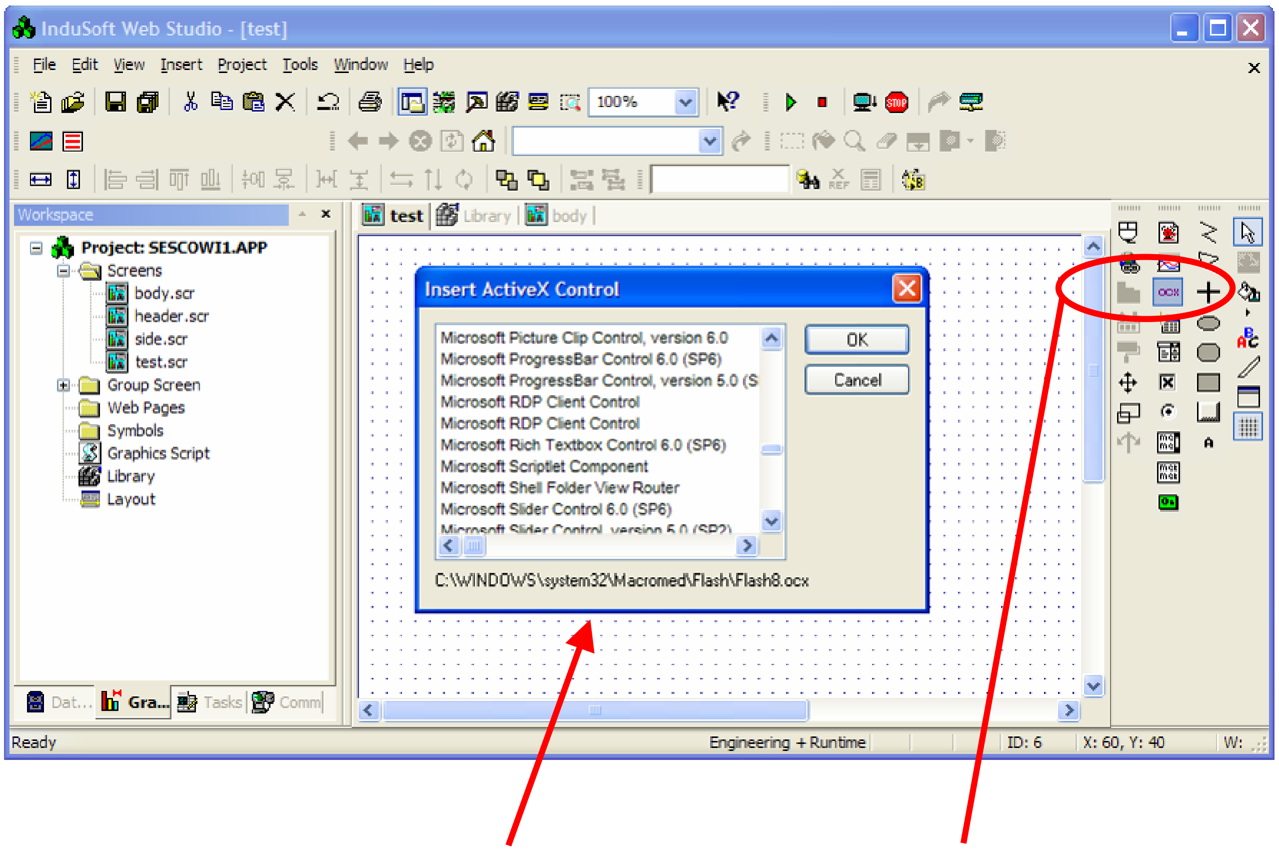
- **Be sure the ActiveX Control is registered.**
  Generally, an ActiveX Control is registered automatically when it is installed. In case it isn't, you can use the IWS **Register Controls** tool, part of the **Tools** available in the IWS menu bar. The Register Controls tool will display a list of registered controls (their ProgID, or Program ID, and the Path to the file where the ActiveX Control is located). If the Control is not registered, you can click on the Register button and browse for the Control.

In our case the ActiveX Control PictureClip was already registered as part of the installation process. The system assigned ProgID is **PicClip.PictureClip.1** and the Path is **C:\Windows\system32\PicClp32.ocx**.



ProgID	Path
MSIISLOG.MSCustomLogCtrl.1	C:\WINDOWS\system32\inetsrv\iislo
MSIISLOG.MSNCSALogCtrl.1	C:\WINDOWS\system32\inetsrv\iislo
MSIISLOG.MSODBCLogCtrl.1	C:\WINDOWS\system32\inetsrv\iislo
MSIISLOG.MSASCIILogCtrl.1	C:\WINDOWS\system32\inetsrv\iislo
MSComCtl2.FlatScrollBar.2	C:\WINDOWS\system32\mscomct2.
Cdlg.RepCdg	C:\PROGRA~1\COMMON~1\MICRC
DirectAnimation.SpriteControl	C:\WINDOWS\system32\daxctle.ocx
ProgressBarCntl.PPie.1	C:\Program Files\Dragonfly Automatio
iProfessionalLibrary.iDualCompassX	C:\WINDOWS\system32\iProfession
MSDBCtls.DBCombo.1	C:\WINDOWS\system32\DBLIST32.
GADGETXLIB2.GXSliderCtrl.1	C:\PROGRA~1\SOFTWA~1\INDUS
DXImageTransform.Microsoft.Light.1	C:\WINDOWS\system32\dxtmsft.dll
MSComDlg.CommonDialog.1	C:\WINDOWS\system32\COMDLG3
DATAQSDK.DataqSdkCtrl.1	C:\WINDOWS\system32\DataqSdk.
DXImageTransform.Microsoft.Random...	C:\WINDOWS\system32\dxtmsft.dll

- **Insert the ActiveX Control on a screen**
  This is done by using the OCX tool in the IWS Toolbar, selecting the ActiveX Control and placing it on the IWS screen.



Microsoft Picture Clip Control, version 6.0
Microsoft ProgressBar Control 6.0 (SP6)
Microsoft ProgressBar Control, version 5.0 (S
Microsoft RDP Client Control
Microsoft RDP Client Control
Microsoft Rich Textbox Control 6.0 (SP6)
Microsoft Scriptlet Component
Microsoft Shell Folder View Router
Microsoft Slider Control 6.0 (SP6)
Microsoft Slider Control, version 5.0 (SP2)

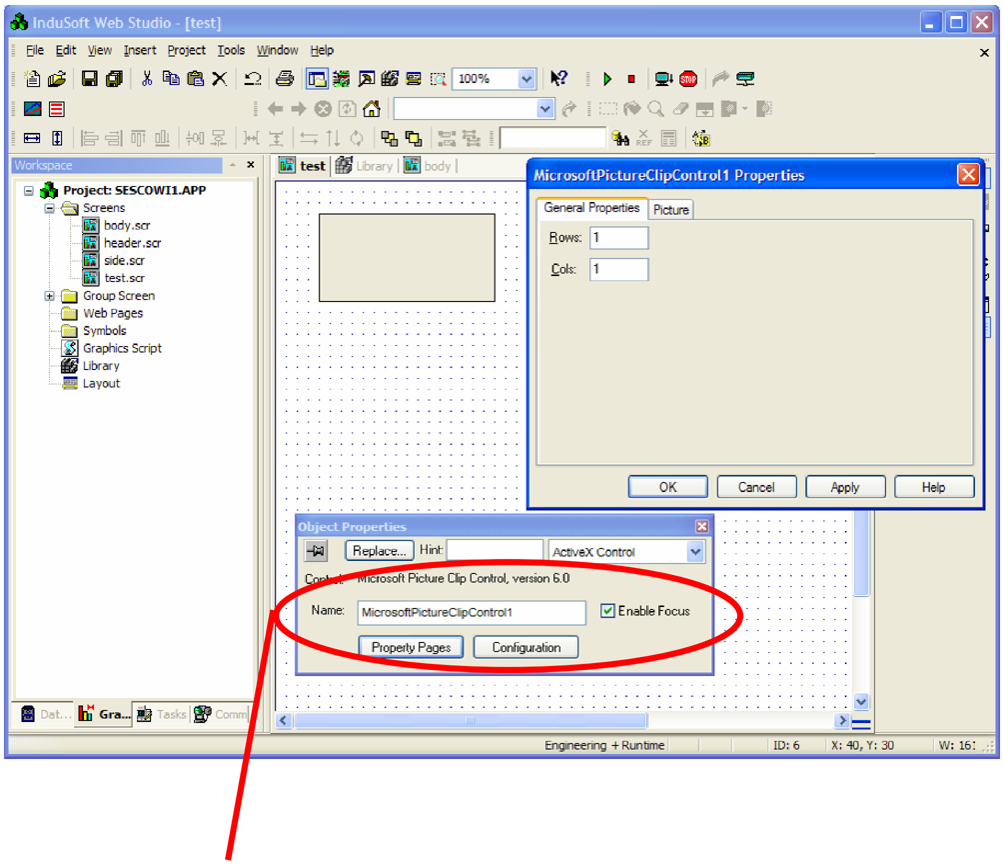
C:\WINDOWS\system32\Macromed\Flash\Flash8.ocx

List of Registered Controls that can be used by IWS   OCX Tool

In our case, the ActiveX Control is called **Microsoft PictureClip Control, version 6.0**.

- **Position the ActiveX Control frame on the Screen**
  There will usually be an object or a frame (e.g. rectangular box) that now appears on the screen display. Click and drag this to the location on the screen where you want it.

- **Rename the ActiveX Control, and Set Properties, Methods and Events**
  Double click on the ActiveX Control Frame. A dialog box for the Object Properties will appear. In the Object Properties dialog box will be the name that IWS has assigned for the ActiveX Control. Each instance of the ActiveX Control will be assigned a unique name by IWS. You can rename the Control as long as the name is unique. The dialog box also contains buttons to access Property Pages and the Configuration of the ActiveX Control (i.e. Properties, Methods, and Events).



Rows: 1
Cols: 1

ActiveX Control
Control: Microsoft Picture Clip Control, version 6.0
Name: MicrosoftPictureClipControl1
[✓] Enable Focus

IWS assigned name for ActiveX Control, Property Pages and Configuration buttons

Note that when you access the ActiveX Control Configuration, you can set the Control's Properties and trigger the Methods based on IWS tag values. ActiveX Controls almost always have one or more Properties but may or may not have Methods or Events. In our example, we are going to use VBScript to access the Control's properties instead of using the Configuration dialog.

The PictureClip Control was developed to allow multiple graphics segments to appear as one large graphic display, and have the graphic display divided into cells. Based on the area of the graphic selected, different action could occur. In our example, we will use the PictureClip Control for a more simple purpose; namely to display an object and scale it to a pre-defined image size.

As mentioned, we will use VBScript to read and set PictureClip Control's various Properties. We will also use one of VBScript's more arcane functions, the **LoadPicture** function. The VBScript **LoadPicture** function is used return a picture object that is loaded from a file. It does not display the object however, this is the function of the ActiveX control.

The LoadPicture syntax is:

```
LoadPicture (filename)
```

Example:

```
oPicture = LoadPicture (“c:\path\MyPictureFile.jpg”)
```

File formats supported by the LoadPicture function include:

- Bitmap (*.bmp)
- Icon (*.ico)
- JPEG (*.jpg)
- GIF (*.gif)
- Metafile (*.wmf)
- Enhanced Metafile (*.emf)
- Run-length Encoded (*.rle)

Now that we have defined the ActiveX Control to be used and the VBScript function we are going to use, we need to know about the ActiveX Control's Properties (and Methods) required to achieve our desired function. Since this Control is a Microsoft control, information can be found on Microsoft's MSDN website.

The PictureClip Control Properties we are going to use are:

- **.Picture**
  The Picture object (loaded by the VBScript **LoadPicture** function)
- **.CellHeight**
  This is a read-only value indicating the image height (Y axis)
- **.CellWidth**
  This is a read-only value indicating the image width (X axis)
- **.ClipX**
  This is the starting X-axis coordinate for the clip we want to define (from the upper left corner of the image)
- **.ClipY**

This is the starting Y-axis coordinate for the clip we want to define (from the upper left corner of the image)

- **.ClipHeight**
  This is the height of the image clip we want to use
- **.ClipWidth**
  This is the width of the image clip we want to use
- **.StretchX**
  This is the final x-axis size of the clip (i.e. we want to stretch/shrink it to the specified # of pixels)
- **.StretchY**
  This is the final x-axis size of the clip (i.e. we want to stretch/shrink it to the specified # of pixels)

In our demo example, we will load the picture, determine it's image size, define the image as a Clip and scale it to a pre-defined window size.

VBScript Demo Code

The demo has 3 display screens that combine to form a screen group. The function of each screen is as follows:

Header.SCR	- Contains a Combo Box that selects the city. The City code is placed in the IWS tag LocID - InduSoft Logo (static image) - Exit Graphic to call the system Shutdown function
Side.SCR	- Displays current logon UserName and computer IP address
Body.SCR	- Contains frame for PictureClip ActiveX Control - Text I/O Display for City name and other data
Body.SCR Screen Script	- VBScript code to read records from an Excel database file. Each record in the file corresponds to data about a city (Not necessarily accurate data). - Acquiring the correct image to display based on the city selected, and scaling the image to fit into a pre-defined window size.
Startup Script	- Acquire the System IP address and store in IWS string tag

Important Note: In VBScript, to access an ActiveX Control that has been instanced in an IWS screen display (as is done in our example), when referring to the ActiveX Control from VBScript, the name of the ActiveX Control **is case sensitive.**

The code below is what is contained in the Body.SCR Screen Script.

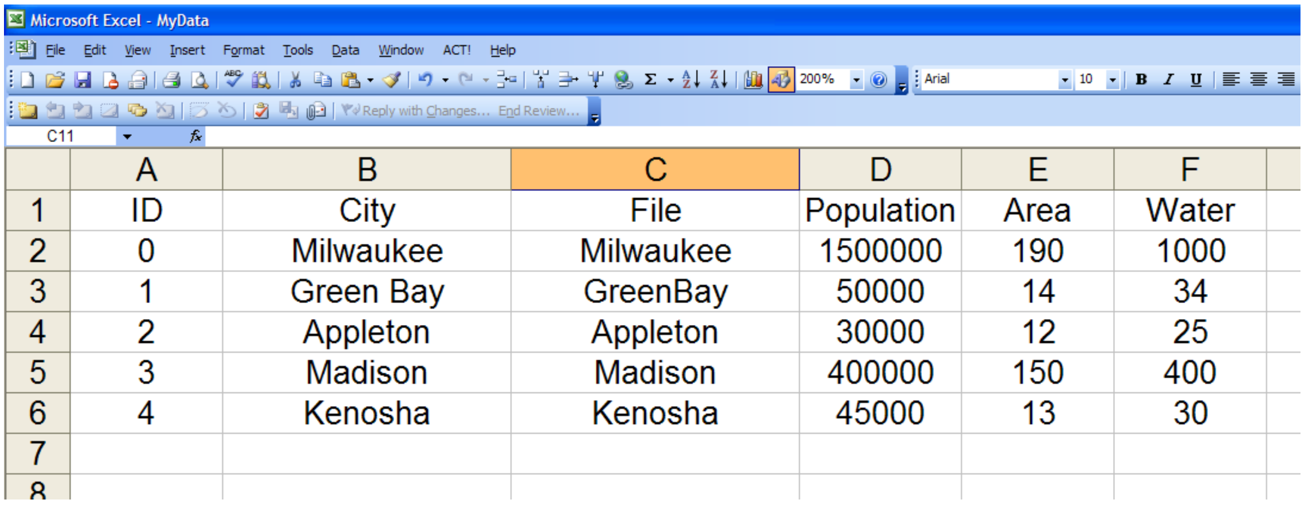
```
Variables available on this screen can be declared and initialized here.
```

```
Dim db
Dim rs
Dim strConn
Dim myFile
Dim OldLocID
Dim SizeX
Dim SizeY
Const DisplaySizeX = 250
Const DisplaySizeY = 250

'Procedures available on this screen can be implemented here

'This procedure is executed just once when this screen is open.
Sub Screen_OnOpen()

' Create and open ADO connections to Database and RecordSet
myFile =  $InfoAppDir() & "\MyData.xls"
strConn = "Provider=Microsoft.Jet.OLEDB.4.0; Data Source=" & myfile & ";"
strConn = strConn & "Extended Properties = Excel 8.0"
Set db = CreateObject("ADODB.Connection")
db.open = strConn
Set rs = CreateObject("ADODB.Recordset")

' Execute SQL to get all cities data
Set rs = db.Execute("Select * FROM [Sheet1$]")

' Load Recordset Data from Excel Database
While Not rs.EOF
  $CityInfo[rs("ID")].CityName = CStr(rs("City"))
  $CityInfo[rs("ID")].PicFile = $InfoAppDir() & "\" & CStr(rs("File")) & ".JPG"
  $CityInfo[rs("ID")].Population = rs("Population")
  $CityInfo[rs("ID")].Area = rs("Area")
  $CityInfo[rs("ID")].WaterCapacity = rs("Water")
  rs.MoveNext
Wend
OldLocID = -1
End Sub

'This procedure is executed continuously while this screen is open.
Sub Screen_WhileOpen()
' Display picture
If OldLocId <> $LocId Then
   OldLocId = $LocId
   PictureClip.Picture=LoadPicture($CityInfo[$LocId].PicFile)
   SizeY = PictureClip.CellHeight
   SizeX =  PictureClip.CellWidth
   PictureClip.ClipX = 0
   PictureClip.ClipY = 0
   PictureClip.ClipHeight = SizeY
   PictureClip.ClipWidth = SizeX
' Scale to Fit into a 200 x 200 Frame
   If SizeX > SizeY Then
      ScaleFactor = SizeX/DisplaySizeX
```

```
    SizeY = SizeY/ScaleFactor
    SizeX = DisplaySizeX
  Else
    ScaleFactor = SizeY/DisplaySizeY
    SizeX = SizeX/ScaleFactor
    SizeY=DisplaySizeY
  End If
  PictureClip.StretchY = SizeY
  PictureClip.StretchX = SizeX
  PictureClip.Picture = PictureClip.Clip
End If
End Sub

'This procedure is executed just once when this screen is closed.
Sub Screen_OnClose()
' Close connection and database. Release memory
rs.close
db.close
Set rs = Nothing
Set db = Nothing
End Sub
```

More about the Microsoft PictureClip Control

The contents of the Excel file (MyData.xls) located in the application folder is as follows: For file, a .jpg format is assumed.


Microsoft Excel - MyData
File Edit View Insert Format Tools Data Window ACT! Help
200%
Arial
10
Reply with Changes... End Review...
C11

	A	B	C	D	E	F
1	ID	City	File	Population	Area	Water
2	0	Milwaukee	Milwaukee	1500000	190	1000
3	1	Green Bay	GreenBay	50000	14	34
4	2	Appleton	Appleton	30000	12	25
5	3	Madison	Madison	400000	150	400
6	4	Kenosha	Kenosha	45000	13	30
7						
8						

More about the Microsoft PictureClip Control

Although our use of the PictureClip Control has been very basic, this Control actually has additional capabilities that let you store and view multiple picture resources into one object and then allow you to select the region of the source picture you are interested in. An example use would be graphical navigation to different screens.

For more information on PictureClip Control, visit the Microsoft MSDN website at www.msdn.microsoft.com.

Important Note: PicClp32.ocx must be installed in the Windows System32 (or System) subdirectory. The control must be registered before it is used by IWS.

.