June Developer Newsletter

Black Ice Software

June 2006

Volume 11, Issue 6

Inside this issue:

Phone: (603) 673-1019

Fax: (603) 672-4112

Sales@blackice.com

www.blackice.com

Low Level Interface of Thumbnail Control

Part two of the thumbnail control is about low level interface support and development for the Image SDK and Document Imaging SDK. The low level interface of the thumbnail control doesn't provide a visual interface like the high level interface, but the programming interface provides great flexibility and the developers can create unique graphical user interfaces with their own thumbnail design.

The thumbnail control has a powerful built in directory browsing capability. Developers can recursively browse a directory structure including sub directories or browse through multi page image files at the same time. The multi page image support is available for GIF, TIFF, DCX, etc. For example, when browsing through a directory with single page JPEG files and the browser encounters a ten page Tiff file, all ten image pages will be displayed as single thumbnail.

When the browser generates a thumbnail, it generates an event, and the developer can handle this event for each file. In the "event handler function" a user can display a thumbnail in a Window, in any device context DC or get information about the image. The thumbnail control has many different sets of

(Continued on page 2)

New Samples in the Cover Page SDK

New Cover Page sample sources are available in different programming languages to reduce development time for software engineers.

The Cover Page sample is available in C++, Visual Basic 6.0, C# 2005, Visual Basic 2005, J# 2005 and Delphi 5.0.

The Cover Page sample displays the template editor and the template filler using the BiCp.dll or BiCp.ocx. These samples demonstrate how to handle Black Ice cover pages in an easy way.

New Samples in the PDF SDK/ActiveX Plug-in

New samples are available in different languages to reduce development time for software engineers.

The Tiff2PDF sample is available in C# 2005, Visual Basic 2005, J# 2005 and Delphi 5.0. This sample converts TIFF files into PDF documents.

The ReadPDF sample processes the PDF document, and converts it to TIFF image. This sample is available in C# 2005, Visual Basic 2005, J# 2005 and Delphi 5.0.

(Low Level Thumbnail Continued from page 1)

properties which developers can modify. These properties are the size of the thumbnail, the position, background color, border size, and the width & color of the border line. The thumbnail generation properties are the following: disable magnification and compression with fix aspect or with distortion. The browsing properties are: recursion of directories, browse into multi-page files, file selection type and maximum file size.

The thumbnail control comes with a sample application written in several languages to help developers, such as C++, C#, J#, VB, Delphi, and VB.NET.

The following sample C++ code demonstrates how simple it is to set the displaying properties, thumbnail generation properties and browsing properties. Once the properties are set, the browsing process is started.

```
// set displaying properties
m_biThumb.SetImageSize(253, 200);
m_biThumb.SetBorderSize(5);
m_biThumb.SetBgColor(RGB(255, 255, 255));
m_biThumb.SetBorderLine(1, RGB(0, 0, 0));

// set thumbnail generation properties
m_biThumb.SetEnableMagnification(false);
m_biThumb.SetAspectFixation(true);

// set browsing properties
m_biThumb.SetRecursiveBrowsing(true);
m_biThumb.SetMultipageExpanding(true);
m_biThumb.SetSelectionOptions(SO_ALLIMAGEFILES);

// start browsing
m_biThumb.BrowseInDirectory("C:\\");
```

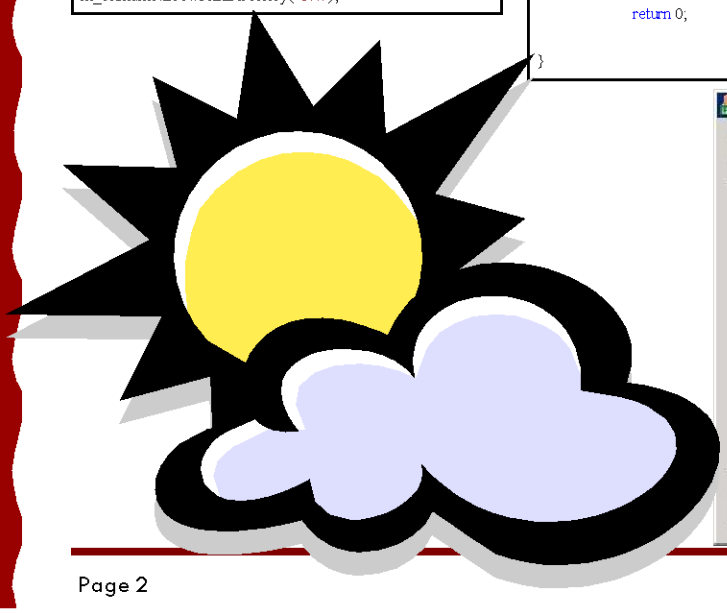
The following code contains the event handler function that displays the current thumbnail and gets the information of the image.

```
LRESULT CThumbnailLowDlg::OnThumbnailEventHandler(WPARAM wParam, LPARAM lParam)
{
	// it must be call
	BiThumbnailInfo* info = reinterpret_cast<BiThumbnailInfo*>(lParam);

	// add your code here

	// display thumbnail
	CClientDC dc(this);
	HBITMAP bmp;
	if(info->bitmap){
		bmp = info->bitmap;
	}else{
		bmp = LoadBitmap(AfxGetInstanceHandle(), MAKEINTRESOURCE(IDB_BITMAP_ILLEGAL));
	}
	m_biThumb.DisplayThumbnail(dc, 12, 45, bmp);

	// load image information into a list control
	CString strInfo;
	if(info->info.isLegal){
		strInfo.Format("%s", info->info.format);
		m_properties.SetItemText(0, 1, strInfo);
		strInfo.Format("%i", info->info.width);
		m_properties.SetItemText(1, 1, strInfo);
		strInfo.Format("%i", info->info.height);
		m_properties.SetItemText(2, 1, strInfo);
		strInfo.Format("%i", info->info.resX);
		m_properties.SetItemText(3, 1, strInfo);
		strInfo.Format("%i", info->info.resY);
		m_properties.SetItemText(4, 1, strInfo);
		strInfo.Format("%i", info->info.depth);
		m_properties.SetItemText(5, 1, strInfo);
	}
	// it must be call
	BiThumbnail::DestroyThumbnailInfo(info);
	return 0;
}
```



Printer Driver Tips: Auto- Print module of the Resource Tool Kit

There are several pitfalls to Automatically Printing documents from MS Office by using the RTK BiAutoPrint.dll of the printer driver.

If you use the Black Ice BiAutoPrint dynamic link library or ActiveX control, you can print documents programmatically. The BiAutoPrint can handle several file formats, for example doc, xls, pps, pdf, html, txt, rtf, etc. You can add the BiAutoPrint modul to your project easily and you can print documents without user interaction, so you can implement batch printing applications.

The BiAutoPrint uses third party applications for printing documents, for example if you want to print a file with the doc extension, the BiAutoPrint will print the document with Microsoft Word. If you want to print PDF files, the BiAutoPrint starts Adobe Acrobat Reader for printing.

Because the BiAutoPrint starts another application, the application should be closed after printing. The BiAutoPrint contains a BIAPEndPrinting method for closing the printing application. You should call this method, when the printing is finished. When the Black Ice printer sends the BLACKICE_MESSAGE_ENDDOC message, the BIAPEndPrinting method call won't close the printing application, because the printing didn't finish yet completely.

The BiAutoPrint sample contains a method for checking the spooler of the printer. If the specified job is removed from the spooler, or the spooler is getting empty, you can call the BIAPEndPrinting method to close the printing application. In this case the printing application (for example Microsoft Word) can be closed correctly.

The following code snippets show you how to get the count of the jobs in the spooler:

[C++]

```
int CAutoPrintDlg::JobsCount()
{
	HANDLE hPrinter;
	DWORD dwNeeded=0;
	BOOL bFlag;
	int iRet;
	PRINTER_INFO_2 *ppi2=NULL;


	bFlag=OpenPrinter(m_szActualPrnName, &hPrinter, NULL);
	if (bFlag && hPrinter)
	{
		SetLastError(0);
		bFlag=GetPrinter(hPrinter, 2, 0, 0, &dwNeeded);

		if (!bFlag)
		{
		 if ((GetLastError() != ERROR_INSUFFICIENT_BUFFER) ||
(dwNeeded==0))
		 {
			ClosePrinter(hPrinter);
			return TRUE;
		 }
		}

		// Allocate enough space for PRINTER_INFO_2.
		ppi2=(PRINTER_INFO_2 *)GlobalAlloc(GPTR, dwNeeded);
		if (!ppi2)
		{
		 ClosePrinter(hPrinter);
		 return TRUE;
		}
		// The second GetPrinter() will fill in all the current information
		// so that all you have to do is modify what you are interested in.
		bFlag=GetPrinter(hPrinter, 2, (LPBYTE)ppi2, dwNeeded,
&dwNeeded);
		if (!bFlag)
		{
		 ClosePrinter(hPrinter);
		 GlobalFree(ppi2);
		 return TRUE;
		}

		iRet=ppi2->cJobs;

		GlobalFree(ppi2);
		ClosePrinter(hPrinter);
	}

	return iRet;
}
```

[VB]

```
Public Function JobsCount(PrinterName As String) As Long
   Dim hPrinter As Long
   Dim ret As Long
   Dim lJobsCount As Long
   Dim pinfo2 As PRINTER_INFO_2
   Dim bufferSize As Long
   Dim prn() As Long

   lJobsCount = 0

   ret = OpenPrinter(PrinterName, hPrinter, ByVal CLng(0))

   If (ret > 0) And (hPrinter > 0) Then
      'The first GetPrinter() tells you how big our buffer must
      'be to hold ALL of PRINTER_INFO_2.
      ret = GetPrinter(hPrinter, 2, ByVal 0&, 0, bufferSize)
      'ret = 0 or
      If bufferSize = 0 Then
         ClosePrinter (hPrinter)
         JobsCount = 0
         Exit Function
      End If

      ReDim prn((bufferSize \ 4)) As Long

      ret = GetPrinter(hPrinter, 2, prn(0), bufferSize, bufferSize)
      If ret = 0 Then
         ClosePrinter (hPrinter)
         JobsCount = 0
         Exit Function
      End If

      ClosePrinter (hPrinter)

      lJobsCount = prn(19)
   End If

   JobsCount = lJobsCount
End Function
```