

Extend Existing Applications with the **LEADTOOLS** Virtual Printer SDK

LEADTOOLS[®]
THE WORLD LEADER IN IMAGING SDKs



Introduction

A common problem that faces many developers is extending existing applications when changing the underlying source code is not a possibility. As needs and systems change, there has to be a mechanism to add new functionality and maintain efficient workflows for end-users. The LEADTOOLS Virtual Printer SDK is one of several solutions LEADTOOLS provides to overcome this problem.

With the LEADTOOLS Virtual Printer SDK, developers can leverage the print feature of any application to extend and enhance that application. Developers benefit without changing source code that may not be available because quality assurance or security measures prohibit changes. End-users benefit because a known and loved application interface does not change and the new process is the same from all applications that print.

LEADTOOLS Document and Medical Imaging lines of SDK products include the LEADTOOLS Virtual Printer and Network Virtual Printer drivers. The main difference between the two drivers is the Network Virtual Printer driver includes network-printing mechanisms, such as the Internet Printing Protocol (IPP), to create custom network or web-based printing solutions, which result in lightweight client deployments, and centralized maintenance and management.

Possibilities and Benefits of Using the LEADTOOLS Virtual Printer SDK

The LEADTOOLS Virtual Printer Driver SDK is one powerful resource that offers several benefits, including:

Store Print Jobs Electronically

The LEADTOOLS Virtual Printer driver provides each page as a memory stream that can be stored to disk, database, cloud, PACS, or any accessible repository.

Export Formats not Supported by the Printing Application

With a couple lines of code, LEADTOOLS converts the EMF memory stream to more than 150 image and document formats. For example, save an Excel spreadsheet to a non-Excel format such as TIFF.

Broadcast Printing

With one print, the end-user sends the job to multiple destinations in multiple formats. For example, the end-user prints one time to save as PNG, add to SharePoint, send to an email group as a PDF attachment, print to paper, and send to a fax driver in parallel.

Preprocess the Print Job before Storage or Transmission

Preprocessing includes reorganization, removal, and insertion of pages, redaction of sensitive information, addition of a Bates stamp, or any other modification required by the user or business rules.

Improve Efficiency with Process Automation

Developers may implement any of the above and more to occur automatically. With a single print, the task printer completes complex tasks behind the scenes, which saves the end-user time and reduces the possibility of human error.

Existing Applications Do Not Change, and Developers Maintain Separation of Concerns Design Principals

Existing complex applications do not change and do not go back through any QA or UAT processes. Developers can extend any application that prints with these features and benefits without requiring any changes to the existing application's source code.

Implement a Task Printer with LEADTOOLS Virtual Printer

A task printer utilizes either LEADTOOLS Virtual Printer or Network Virtual Printer driver. A developer's first step when creating a task printer is to define the task-printer's properties. Once the developer defines the properties, one line of code adds the task printer into the list of Windows printers.

Define Printer Properties

Defining the properties is straight forward and accomplished by simply setting the values of a `Leadtools.Printer.PrinterInfo` instance.

```

internal static PrinterInfo CreatePrinterInfo( string printerName,
string printerPassword )
{
    const string documentPrinterRegPath =
        @"SOFTWARE\LEAD Technologies, Inc.\Printer\";
    const string printerDriverName = "LEADTOOLS VIRTUAL PRINTER";

    string taskPrinterPath =
        System.Reflection.Assembly.GetExecutingAssembly().Location;
    string taskPrinterRootDirectory = Path.GetDirectoryName( taskPrinterPath );

    return new PrinterInfo() {
        DriverName = printerDriverName,
        PortName = printerName,
        MonitorName = printerName,
        ProductName = printerName,
        PrinterName = printerName,
        Password = printerPassword,
        RegistryKey = documentPrinterRegPath + printerName,
        RootDir = Properties.Settings.Default.DriverRootDir,
        Url = "https://www.leadtools.com/support",
        PrinterExe = taskPrinterPath,
        AboutString = String.Format( "{0} - {1}", printerDriverName, printerName ),
    };
}

```

The key properties of the `PrinterInfo` object are `PrinterName`, `RootDir`, and `PrinterExe`.

- The `PrinterExe` property is the name of the task-printer executable to link the printer listed in the Windows printers. When an end-user prints to the task printer, the LEADTOOLS Virtual Printer executes the application defined in `PrinterExe`.
- The `PrinterName` property is the name of the printer that displayed in the list of Windows printers. Additionally, the `PrinterName` links the task-printer application (`PrinterExe`) to the printer listed in the list of Windows printers and subscribes to its events.
- The `RootDir` is the location of `LeadtoolsPrinter.exe`, which is included with the LEADTOOLS SDK and used by all task printers.

Install the Task-printer

Once a developer defines a `PrinterInfo` object, only one call to a static method is required to install the task printer into the list of Windows printers.

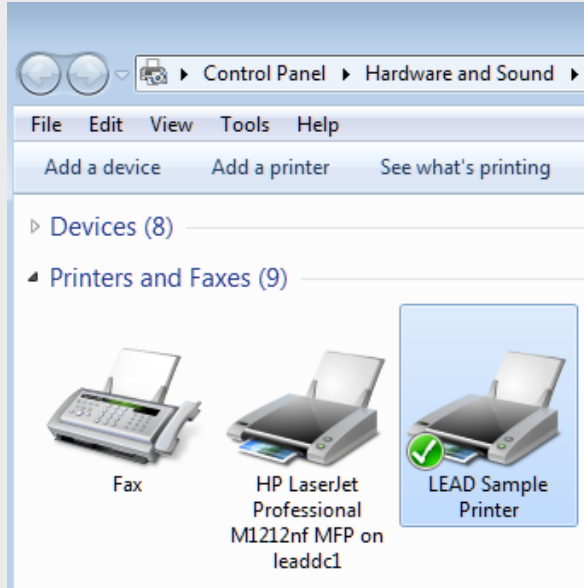
```

Printer.Install( CreatePrinterInfo( printerName, printerPassword ) );

```

The call to `Printer.Install()` results in the following:

- Windows displays the task printer defined by the `PrinterInfo` class in the list of Windows printers, just like any other printer.
- Printing to the newly installed task printer triggers the task-printer executable defined in the `PrinterInfo.PrinterExe` property to run.



The application that installs a LEADTOOLS task printer does not have to be the same application that implements the tasks performed by the printer. An installation application defines and installs task printers on the fly and provides a means to have multiple custom task printers specifically tailored to a specific task or set of tasks. For instance, one installation application may create a task printer that stores as PDF in SharePoint and a different task printer to search and redact social security numbers before emailing the print job as a TIFF.

Implement the Task of the Printer

The next step of development is to have something useful occur when the task printer receives a print job. To facilitate this, the `Leadtools.Printer.Printer` class provides two events:

- **Job Event** – Fires when a print job starts and ends.
- **Emf Event** – Fires for each page of the print job and provides the EMF produced by the printer spooler

The task-printer implementation can be a simple console or a complex GUI application. The decision between a console and GUI task printer primarily depends on the need of user interaction such as reorganizing and merging print jobs or manually redacting user-designated areas. In both cases, installation and event handling are the same.

The LEADTOOLS SDK includes GUI examples implemented in .NET and C++. Code snippets below show a basic, required implementation with a simple C# console application. The full C# solution is available at <https://www.leadtools.com/blog/document-imaging/white-paper-extend-existing-applications-leadtools-virtual-printer-sdk/>

Create an Instance of the Virtual Printer

To create an instance of the `Printer` object, the task printer uses the `PrinterInfo.PrinterName` value set by the installation application to hook to the correct printer.

```
Printer printer = new Printer( PrinterName );  
printer.EmfEvent += new EventHandler<EmfEventArgs>( VirtualPrinter_EmfEvent );  
printer.JobEvent += new EventHandler<JobEventArgs>( VirtualPrinter_JobEvent );
```

The task printer raises the `Job` Event at the beginning and end of each print job, and provides information such as the id of the print job and name of the printer.

In this example, the application ends when the print job ends because the printer has no additional tasks to complete. In cases where the user interaction is required, the application remains open. The LEADTOOLS SDK includes C++ and .NET projects demonstrating “stay open” task-printer implementations.

```

static void VirtualPrinter_JobEvent( object sender, JobEventArgs e )
{
    string printerName = e.PrinterName;
    int jobID = e.JobID;

    switch ( e.JobEventState )
    {
        case EventState.JobEnd:
            // Exit with success
            Environment.Exit( 0 );
            break;
        case EventState.JobStart:
            Console.WriteLine(
                String.Format(
                    "{0}: Job {1} has started", printerName, jobID ) );
            break;
        default:
            break;
    }
}

```

The bulk of processing usually occurs in the `EmfEvent`. Alternatively, the printer may store the EMF for later processing. The event provides each page as an EMF stream, which may be saved to disk or kept in memory for additional processing.

Save EMF Directly to Disk

The following snippet shows how to save the EMF stream directly to disk. Once the EMF is saved to disk, it may be processed later without sacrificing memory, especially if the print job has many pages.

```

// Write the EMF as file to disk.
static void VirtualPrinter_EmfEvent( object sender, EmfEventArgs e )
{
    string taskPrinterPath = Assembly.GetExecutingAssembly().Location;
    string emfPath = Path.Combine(
        @"c:\Output\EMF\",
        Path.GetFileNameWithoutExtension(Path.GetRandomFileName() )
    ) + ".emf";
    Directory.CreateDirectory(Path.GetDirectoryName( emfPath ) );
    File.WriteAllBytes( emfPath, e.Stream.ToArray() );
}

```

Process the EMF with LEADTOOLS

If additional or advanced processing is required immediately, the `RasterImageConverter` class easily converts the EMF stream to a `RasterImage`. The following snippet shows an example that converts the EMF to a `RasterImage`, fills a predefined area with black, and saves the result as a PNG. While this example fills a predefined area, the end-user may select a specific area or any number of regions to redact in a GUI application. There are certain cases when processing the EMF before committing to disk is desirable such as redacting personal information for security or privacy reasons.

```
// Get the EMF, convert to an image, fill a region and save as PNG.
static void VirtualPrinter_EmfEvent( object sender, EmfEventArgs e )
{
    string taskPrinterPath = Assembly.GetExecutingAssembly().Location;
    string pngPath = Path.Combine(
        @"c:\Output\PNG\",
        Path.GetFileNameWithoutExtension( Path.GetRandomFileName() )
    ) + ".png";
    Directory.CreateDirectory( Path.GetDirectoryName( pngPath ) );
    // Get the EMF.
    Metafile metaFile = new Metafile( e.Stream );
    IntPtr hEmf = metaFile.GetHenhmetafile();
    // Convert to an image.
    RasterImage image =
        RasterImageConverter.FromEmf( hEmf, 0, 0, RasterColor.White );
    // Define the region to fill.
    RasterRegion region =
        new RasterRegion(
            new LeadRect( 20, 30, 100, 200 ) );
    // Set the region in the image.
    image.SetRegion( null, region, RasterRegionCombineMode.Set );
    // Fill image region with black.
    new FillCommand( RasterColor.FromKnownColor( RasterKnownColor.Black ) )
        .Run( image );
    // Save as PNG.
    RasterCodecs codecs = new RasterCodecs();
    codecs.Save( image, pngPath, RasterImageFormat.Png, 32 );
}
```

Save the EMF as PDF

LEADTOOLS includes the `DocumentWriter` class that can save the EMF as several document formats including DOCX, PDF, HTML, SVG, and XPS. The following code snippet shows how easy it is to save the EMF as PDF.


```
// Write the EMF as file to disk as PDF.
static void VirtualPrinter_EmfiEvent( object sender, EmfiEventArgs e )
{
    string pdfPath = Path.Combine(
        @"c:\Output\PDF\",
        Path.GetFileNameWithoutExtension( Path.GetRandomFileName() )
    ) + ".pdf";
    Directory.CreateDirectory( Path.GetDirectoryName( pdfPath ) );

    // Create an instance of the LEADTOOLS DocumentWriter
    DocumentWriter docWriter = new DocumentWriter();
    docWriter.BeginDocument( pdfPath, DocumentFormat.Pdf );

    DocumentEmfiPage page = new DocumentEmfiPage() {
        EmfiHandle = new Metafile( e.Stream )
        .GetHenhmetafile()
    };
    docWriter.AddPage( page );
    docWriter.EndDocument();
}
```

Much more advanced processing is possible with the LEADTOOLS SDK. Additionally, any storage option such as SharePoint, database, or cloud, such as Salesforce, Amazon and Google, is easily obtainable now that the basic task-printer implementation is in place.

Conclusion

Considering the ubiquitous nature of the print command combined with imagination and a powerful SDK such as LEADTOOLS, the LEADTOOLS Virtual Printer driver opens up an unlimited number of possibilities. From staying “green,” to streamlining complex workflow scenarios, to capturing information from multiple sources into one common repository, File > Print is one way LEADTOOLS gets it done. For more information on how LEAD Technologies can image-enable your application and boost your ROI, visit www.leadtools.com to download a free evaluation, or give us a call at +1-704-332-5532.

Sales: +1(704) 332-5532
sales@leadtools.com

Support: +1(704) 372-9681
support@leadtools.com



LEAD Technologies, Inc.
1927 South Tryon Street
Suite 200
Charlotte, NC 28203

About LEAD Technologies

With a rich history of over 28 years, LEAD has established itself as the world's leading provider of software development toolkits for document, medical, multimedia, raster, and vector imaging. LEAD's flagship product, LEADTOOLS, holds the top position in every major country throughout the world and boasts a healthy, diverse customer base and strong list of corporate partners including some of the largest and most influential organizations from around the globe.

LEADTOOLS[®]
THE WORLD LEADER IN IMAGING SDKs

