Puma.NET OCR SDK getting started

With Puma.NET recognition can be as simple as the following lines of code:

```
var pumaPage = new PumaPage("page001.jpg");

using(pumaPage)
{
    pumaPage.FileFormat = PumaFileFormat.RtfAnsi;
    pumaPage.EnableSpeller = false;
    pumaPage.Language = PumaLanguage.English;

    pumaPage.RecognizeToFile("page001.rtf");
}
```

You can either download a setup package (.msi file) that contains binary files and reference documentation in order to faster try Puma.NET or you may download the entire Visual Studio 2008 solution containing source codes of the project. The below description gives you the general information on Puma.NET and how you can gain it in your .NET applications.

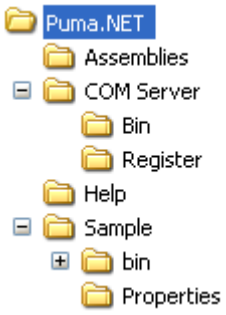
Puma.NET Runtime Components:

1. Puma.Net.dll – class library that contains the classes (actually one essential - PumaPage) that should be used in your application to gain OCR functionality;
2. CuneiForm binary files (Puma COM, DLLs, Dictionaries etc.) – the underlining recognition engine by Cognitive Technologies;
3. puma.interop.dll – interoperation assembly consumed by Puma.Net.dll in order to marshal operation calls to Puma COM Server;
4. dibapi.dll – native Win32 DLL that contains functions necessary to work with data structures that Puma COM and puma.interop.dll consume. This library should be accessible to the application referencing Puma.Net.dll.

Solution (source codes) contains merge module (msm) project holding all necessary runtime components that you may include in any MS deployment project. It will ease redistribution of OCR components with developed applications.

Installing and using Puma.NET

After downloading and installing the deployment project PumaNet.msi the target folder will be of the following contents:



- Assemblies folder – contains files Puma.Net.dll, puma.interop.dll, dibapi.dll; these three files are to be accessible to the executable referencing Puma.Net.dll;
- COM Server folder – contains binary files of CuneiForm recognition engine (as well as Puma COM Server file apuma.dll). COM server is automatically registered during installation;
- COM Server\Register folder – contains batch files that allow to manually register/unregister COM server;
- Help folder – contains PumaNET.chm with generated reference documentation on classes of Puma.Net.dll. It also contains other documentation files (like getting started you are reading);
- Sample folder – contains sample Windows Forms application (with source codes) using Puma.NET.

At first you may review the sample application that demonstrates OCR capabilities of Puma.NET.

Generally an application needs to reference Puma.Net.dll in order to access recognition classes. It's also necessary to insure that puma.interop.dll is placed in the same folder as Puma.Net.dll (that's true after installing as they are in Assemblies folder) in order when adding reference with the help of Visual Studio the environment to be capable to find it and copy both files to the output folder.

It's also important that library dibapi.dll is placed in the same folder as the running application so it could be found and loaded by it.

Steps to add Puma.NET to your project:

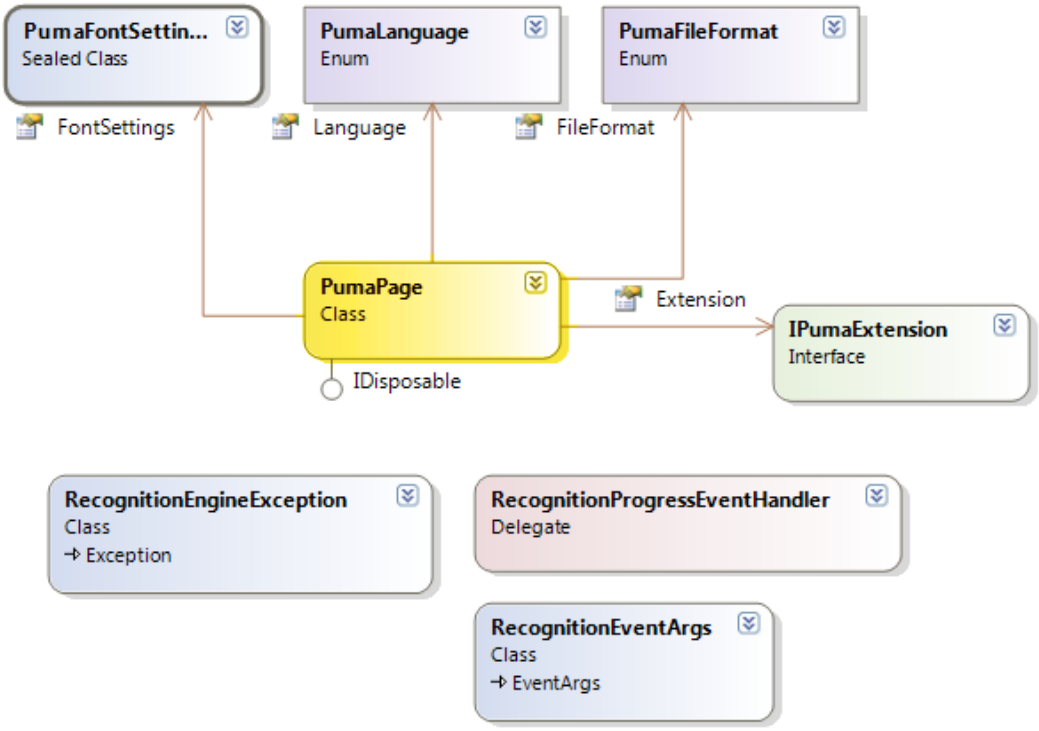
1. Add reference to Puma.Net.dll;
2. Make sure that after project building the output folder (i.e. bin\Debug or bin\Release) contains files Puma.Net.dll and puma.interop.dll. If the last is not present (IDE didn't copy it) copy it to the folder manually;
3. Copy dibapi.dll to the output folder;

After all necessary files are accessible to the application you may test OCR functionality.

Note:

- Assemblies Puma.Net.dll and puma.interop.dll are strong named and can be put to GAC;
- dibapi.dll can be placed in folder Windows\System32;
- If the above conditions are true it's not necessary for those files above to be in the same folder as consuming application.

Puma.NET classes



As you can see the class model is simple:

1. PumaPage – the main class that processes a single image (a page) and produces recognition results in a form of a file or a string value. Most recognition parameters (language, output format etc.) are set via public properties of this class.
2. PumaFontSettings – the class groups recognition parameters regarding fonts (boldness, underline, font names to produce output etc.).
3. PumaLanguage, PumaFileFormat – enumerations for corresponding properties in PumaPage.
4. IPumaExtension – classes implementing this interface can be aggregated within PumaPage in order to extend its functionality.
5. RecognitionProgressEventHandler, RecognitionEventArgs, RecognitionException – maintaining recognition events and special errors.

String literals issued by classes in this assembly are stored in resources thereby can be easily localized.