DLS SDK Release Notes

December 1, 2011

Revision History

DYMO Label v.8.2.2.994 bug fixes:

- Fix initialization problem in the DymoAddIn object. The problem is seen when the name of the printer last used by the DymoAddIn object is changed via Window's Printers and Faxes folder.
- Fix memory allocation issues in the Firefox extension where strings are freed after they were returned to the caller.

DYMO Label v.8.2.1.903 bug fixes:

- Fixed issue with the SelectPrinter() method from the IDymoAddIn interface always returning "False" regardless of whether the specified printer is selected or not.

DYMO Label v.8.2.1.909 bug fixes:

- Fixed issue with SmartPaste() related methods failing when using a label that had not been printed before.

DYMO Label v.8.2.1.912 bug fixes:

- Fixed methods returning BSTR (i.e. string memory should be allocated before returning, and the memory should be freed by the caller).

DYMO Label v.8.2.1.913 bug fixes:

- Moved unreferenced interface definitions in IDL into the library section so the definitions will appear in .NET.

Overview

The DLS SDK is a new implementation of the COM interfaces declared in the DLS 7 SDK. The implementation supports both the High Level and Low level COM interfaces described in the DLS 7 SDK manual.

The new implementation is based on the DYMO Label v.8 codebase so existing SDK applications will be able to work with the new .label file format (as well as the old .lwl format). The implementation is binary compatible with existing SDK applications so the same SDK application that worked with DLS 7 will work with DYMO Label v.8 without needing to recompile.

The library is installed and registered automatically when installing the latest DYMO Label v.8 application. This means SDK application users can have either DLS 7 or DYMO Label v.8 installed on their system and their SDK application should continue to work as is.

What's New

- Implemented using the new DYMO Label v.8 codebase.
- Binary compatible with existing SDK applications.
- Supports opening and printing of DLS 7 and DYMO Label v.8 label file formats (i.e. ".lwl" and ".label" files).
- Adds support for LabelWriter 450 and 4XL printers.
- Supports Intelligent Mail barcode (replacing the deprecated POSTNET barcode).
- Supports customized web proxy settings (see IDymoAddin6 interface).
- Supports querying printer's online and offline status (see IDymoAddin6 interface).
- Supports setting image via URL (see IDymoLabels3 interface).
- Supports print job control in the low level COM interface (see ILabelEngine4 interface).

New High Level COM Interfaces

IDymoAddin6

SetupProxySettings() method:

Definition:

```
void SetupProxySettings(string protocol, string serverName, long Port,
string proxyBypass, string userName, string password);
```

Allows customized proxy settings (different from IE's default proxy settings). All URL related function calls in the SDK will adhere to these proxy settings.

ClearProxySettings() method:

Definition:

```
void ClearProxySettings();
```

Clears all proxy settings and revert back to using IE's default proxy settings.

ProxyBypass property:

Definition:

```
bool proxyBypass;
```

Setting the property to **true** will cause all URL related SDK functions to bypass any proxy settings, including IE's default proxy settings.

Setting the property to **false** (the default value) means all URL related SDK functions will use either IE's default proxy setting or the user specified proxy settings.

IsPrinterOnline() method:

Definition:

```
bool IsPrinterOnline(string PrinterName);
```

Returns true of the specified printer is online, false if the printer is offline.

SetGraphicsAndBarcodePrintMode method:

Definition:

```
void SetGraphicsAndBarcodePrintMode([bool isModeOn);
```

When the mode is on (default value), labels containing barcode(s) will print at high quality mode but the print speed is reduced. Unsetting this mode will cause all labels to print at the fasted print speed.

IDymoLabels3

SetImageURL() method:

Definition:

```
bool SetImageURL(string ObjectName, string imageURL, string
imageTypeStr);
```

Allows specifying URL as the image source for an image object on the label. The "imageTypeStr" parameter was used to indicate the type of the image file in the URL, but it is no longer needed in the new implementation.

New Low Level COM Interfaces

ILabelEngine4

StartPrintJob() and EndPrintJob() method:

Definition:

- void StartPrintJob();
- void EndPrintJob();

Wrapping ILabelEngine.PrintLabel() and ILabelEngine2.PrintLabelEx() calls within the StartPrintJob() and EndPrintJob() calls will cause labels to be printed as pages of the same print job. The benefit is seen with reduced the print job overhead and increased label printing speed when printing to LabelWriter 400 and 450 series printers.

Example:

```
// this printing loop creates a 10 page print job
StartPrintJob();
for (i = 0; i < 10; i++)
{
      // update some fields on the label
      ...

      LabelEngine.PrintLabel(…); // print one label
}
```

```
    EndPrintJob();
```

this code above will print labels much faster than the code below:

```
    // this printing loop creates 10 different one page print jobs
    for (i = 0; i < 10; i++)
    {
          // update some fields on the label
          ...

          LabelEngine.PrintLabel(…); // print one label
    }
```

Functionality Differences

Because the DLS SDK is based on the DYMO Label v.8 code, some features available in the DLS 7 SDK are deprecated or changed. Understanding these differences can help you determine if the DLS SDK works with your existing SDK application.

Functional differences in the High Level COM SDK implementation

The following functions are deprecated in the **IDymoAddIn** interface:

- **IDymoAddIn.Hide() method**
- **IDymoAddIn.Show() method**
- **IDymoAddIn.Systray() method**
- **IDymoAddIn.Quit() method**

These functions are changed to do nothing (no-op) when called from an SDK application.

IDymoAddIn.Open() function:

The function will try to open the specified label file name with the .label extension first, even if the parameter specifies a different file extension.

For example, if your application calls:

```
IDymoAddIn.Open(“mylabel.lwl”);
```

The function will try to look for the file in the following order:

- mylabel.label
- mylabel.lwl
- mylabel.lwt

* The reason for this behavior has to do with how the implementation handles both ".lwl" and ".label" file formats. The implementation converts ".lwl" format into ".label" format internally before performing any actions on a label file. What this means is that when a label that was opened as ".lwl" will be saved as a ".label" file if the Save() or SaveAs() method is called.

So if an SDK application opens a ".lwl" label file, modifies it, then saves the file. The file would be saved as a ".label" file. When the application returns to open the same ".lwl" file expecting to see the modifications, the Open() method would need to open the ".label" file for the modification to be seen.

IDymoAddIn.Save() function:

The function will save the currently opened label in the .label file extension, even if the parameter specifies a different file extension.

For example, if your application calls:

```
IDymoAddIn.Save();
```

The function will save the file with the same name as the currently opened file but with a .label file extension. The label file is in the new DYMO Label v.8 .label file format.

IDymoAddIn.SaveAs()function:

The function will save the specified label file name in the .label file extension, even if the parameter specifies a different file extension.

For example, if your application calls:

```
IDymoAddIn.SaveAs(“newlabel.lwl”);
```

The function will save the file as “newlabel.label”. The label file is in the new DYMO Label v.8 .label file format.

IDymoAddIn.Print() and IDymoAddIn2.Print2() function:

The parameter that controls whether a print dialog is shown during printing is ignored.

IDymoAddIn2.Open2() function:

Similar to the change in IDymoAddIn.Open() method. The function is changed to open the specified label file name with the .label extension first.

IDymoAddIn2.GetMRULabelFiles()function:

The function used to return the same list of files in the "Label Files" dropdown of the DLS 7 application. It’s changed to:

The first time any SDK application is run, the MRU list is initialized with the "Recent Layouts" list from the DYMO Label v.8 application. Once the MRU list is initialized, it is maintained separately from the DYMO Label v.8 application. All SDK applications share the same MRU list within the same user account.

IDymoAddIn5.OpenURL() function:

The function will use IE’s proxy settings by default. However, you can override the default settings using methods in the new IDymoAddIn6 interface.

Functional differences in the Low Level COM SDK implementation

ILabelList interface:

The implementation returns the new label type names in DYMO Label v.8. However, calling the function with old DLS 7 label names will continue to work.

For example, when the DLS 7 “Address (30252)” label name is used, the implementation will map it to the equivalent “Address Label” in DYMO Label v.8.

ITextAttributes interface:

The implementation changed to that Font1 and Font2 fields are **not** kept separately from the text field: if the text contains formatting information, the Font1 and Font2 values represent the font format of line1 and line2 + subsequent lines. Setting the Font1 and Font2 property will change the text so line1 of text is in Font1 format and line2 + subsequent lines are in the Font2 format.

If you set RTF text in the Text field, Font1 and Font2 fields are automatically changed to reflect the font formats in line1 and line2 + subsequent lines.

If you set plain text in the Text field and the Text field was empty, the default font format (i.e. Arial, 16pt) is used for the plain text. If the Text field holds some value, then the font format for the plain text will derive from the previous text field data.

The following font styles are not supported:

- Shadow font
- Outline font

ITextAttributes.Text property

The new implementation will return plain text. The previous implementation will return RTF formatted text if the object was initialized with RTF text.