DYMO Label Web Service FAQ

Contents

Working with DYMO Label Web Service

What is the DYMO Label Web Service?

In the past, developers had to provide a browser-specific plug-in for each major web browser. Nowadays, most browsers have phased out native plug-in support. Google, for example, stopped supporting Chrome their NPAPI browser extension in September 2015. In response, we released the DYMO Web Service as a new cross-browser solution allowing third-party developer applications the ability to interface with the DLS SDK in a seamless, browser-agnostic fashion. It handles all printer-related requests from the JavaScript Library that the DYMO Label Framework browser plug-ins used to perform.

How do I install the DYMO Label Web Service?

First, download the appropriate installer for your OS. You can find them at the following URL:

http://www.dymo.com/en-US/online-support/dymo-user-guides

(should install directions be more detailed than this? Screenshots?)

Windows

Double-click on the installer and follow the directions provided by the install wizard.

Mac

Double-click on the DMG file to mount it. Select the newly mounted volume and double-click on the PKG file found within it. Then follow the directions provided by the install wizard.

How can I tell if DYMO Label Web Service is installed?

The DYMO Label Web Service should be installed as long as you have installed DYMO Label Software 8.5.3 or newer using the express “Express” mode. *(add express mode screenshot?)*.

If you choose to install DYMO Label Software in “Custom” mode, be sure to select the **DYMO Label Web Service** component as follows:

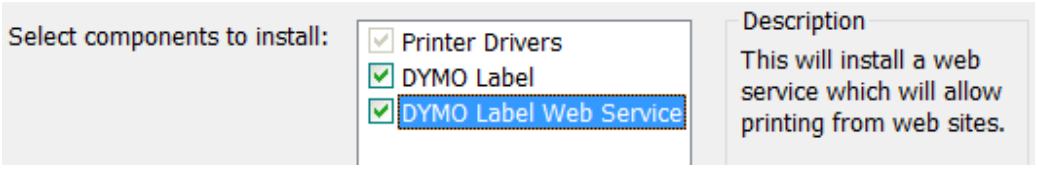

Figure 1Custom select components to install.

If installed, there will be an executable file named **DYMO.DLS.Printing.Host.exe** within the DLS working folder (normally found within the *C:\Program Files (x86)\DYMO\DYMO Label Software* folder on Windows and the */Library/Frameworks/DYMO/SDK* folder on Mac).

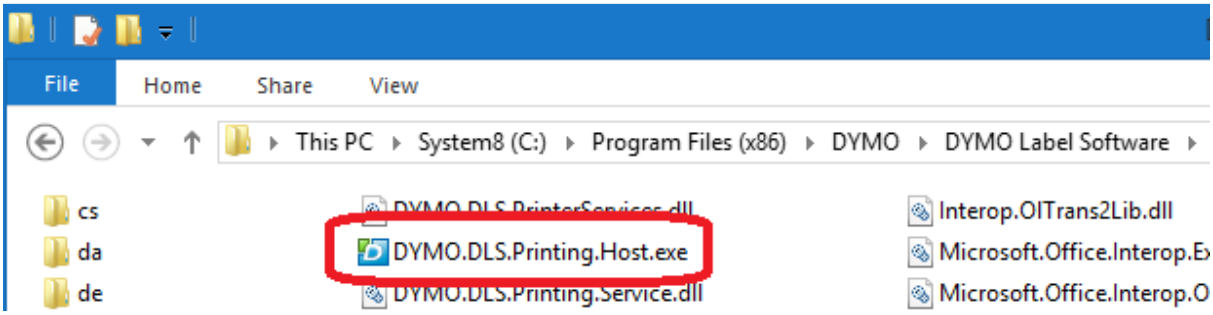

Figure 2 Executable location (Windows)

How can I tell if the DYMO Label Web Service is running?

You should see the DLS application icon within the system tray. Right-click this icon to display a context menu. The menu displays the web service's status (i.e., "**Started on port 41951**" or "**Stopped**").

The following shows what it looks like on Windows.

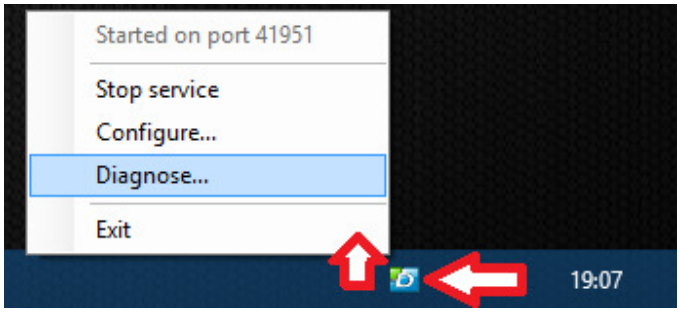

Figure 3 DLS icon and context menu (Windows)

On Mac, the DLS application icon and context menu will appear within the system tray like this:

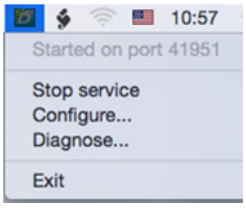

Figure 4 DLS icon and context menu (Mac)

I do not see it in the system tray. How can I start it?

Windows

You can start the web service again by navigating to the DLS working folder and running the executable named **DYMO.DLS.Printing.Host.exe**.

Mac

Open a Finder window, navigate to the */Library/Frameworks/DYMO/SDK/* folder, and click on the **DYMO.DLS.Printing.Host.app** icon.

Open a terminal prompt and enter the following command:

```
launchctl start com.dymo.dls.webservice
```

How can I start or stop the DYMO Label Web Service?

You can start or stop the web service at any time by clicking on the **Start service** and **Stop service** menu items, respectively. Although the service icon will remain in the system tray after stopping it, no API functions will be executed. *(add screenshots?)*

How can I configure the DYMO Label Web Service?

Clicking the **Configure** menu item will cause a configuration window to appear. This allows you to change the language and listening port. The web service will normally try to use the first available port within the 41951-41960 range. You can override this behavior by checking the **Use single port** checkbox, which makes the service only try using the specified port only. You cannot specify a port number that does not fall within the specified range. The service will not try using any other port if an error occurs while using this option.

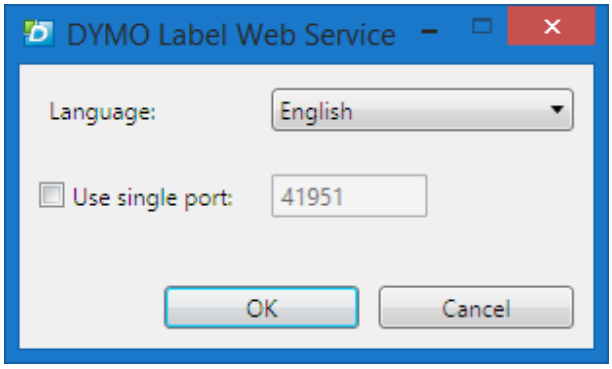

Figure 5 Web Service configuration dialog (Windows)

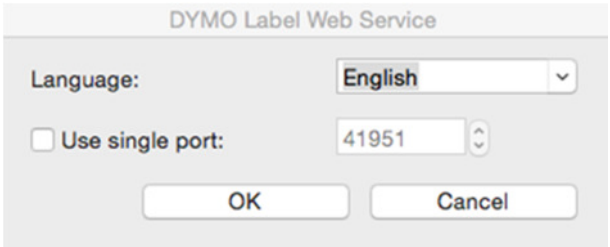

Figure 6 Web Service configuration dialog (Mac)

How can I tell if the DYMO Label Web Service is functioning properly?

Click the **Diagnose** menu item within the context menu while the service is running. If the self-test succeeds, a dialog box will appear asking you to open a test page in your browser to see if SSL certificate is working.

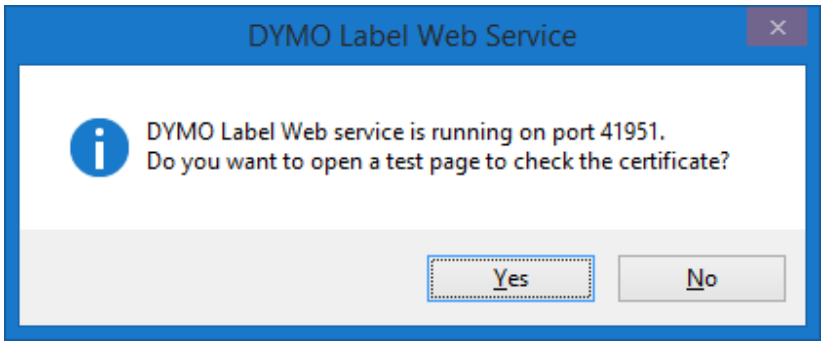

Figure 7 Diagnose successful (Windows)

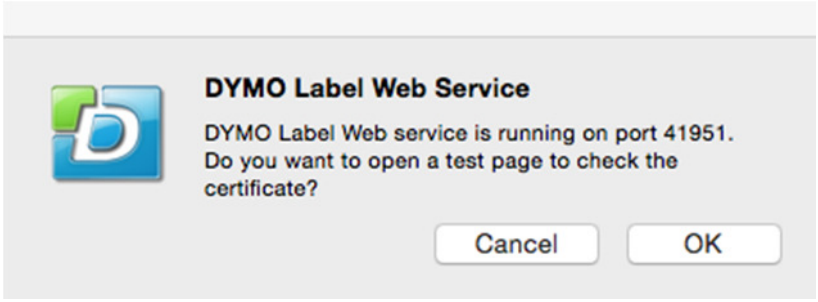

Figure 8 Diagnose successful (Mac)

Click the **Yes** button to open your default web browser. The browser should display a page indicating the web service is running correctly. The page address should be something to the following effect:

https://localhost:41951/DYMO/DLS/Printing/Check

The port number may vary from machine to machine.

Figure 9 Web Service is up and running confirmation.

How do I use the DYMO Label JavaScript Library?

Getting Started

You need to link to the DYMO Label Framework's JavaScript library in order to use the DYMO Label Web Service via web pages. You accomplish this by using the following code snippet:

```
<script src="dymo.label.framework.js" type="text/javascript" charset="UTF-8"></script>
```

Basic Service API Functions

The following is a list of basic API functions provided by the DYMO Label JavaScript Library.

dymo.label.framework.getPrinters()

Purpose: Returns a list of installed DYMO printers
Parameters: None.

Example:

```
printers = dymo.label.framework.getPrinters();
for(var i = 0; i < printers.length; i++)
{
var printer = printers[i];
 console.log(printer);
}
```

dymo.label.framework.openLabelFile(labelUri)

Purpose: Returns a document object model (DOM) for label file.
Parameters:

labelUri - label file Uri

Example:

```
var labelUri = "file:///Volumes/DATA/DieCut.label";
var label = dymo.label.framework.openLabelFile(labelUri);
console.log(label.toString());
```

Comments:

It is necessary to use the **label.getLabelXml()** or **label.toString()** functions to obtain XML representations of a label.

dymo.label.framework.renderLabel(labelXml, paramXml, printerName)

Purpose: Returns a graphic representation of the label (PNG) encoded in base64 format.
Parameters:

labelXml - DOM or XML representation of the label
paramXml - render parameters
printerName - printer name

Example:

```
var image = document.createElement('img');
var labelXml = dymo.label.framework.openLabelFile(labelUri).getLabelXml();
var pngData = dymo.label.framework.renderLabel(labelXml, "", printerName);
image.src = "" + pngData;
```

dymo.label.framework.printLabel(printerName, paramXml, labelXml, labelSetXml)

Purpose: Prints a label on the specified printer.
Parameters:

printerName - printer name
paramXml - printing parameters
labelXml - DOM or XML representation of the label
labelSetXml - data set for label objects

Example:

```
var paramsXml = dymo.label.framework. createLabelWriterPrintParamsXml ({ copies: 2 });
var labelSetXml = new dymo.label.framework.LabelSetBuilder();
var record = labelSet.addRecord();
record.setText("Address", "Test Address String");
dymo.label.framework.printLabel(printerName, paramsXml, labelXml, labelSetXml);
```

Do I need to change my code to work with the new DYMO Label JavaScript Library?

Although it *may* work with old unmodified code in some cases, it is highly recommended to make a few changes to avoid future problems.

The biggest change to the JavaScript Library is the move from a synchronous architecture to an asynchronous one. This move has several advantages, improved UI responsiveness and shorter discovery time while scanning the available port range. Synchronous AJAX calls are already marked deprecated in most major web browsers, so we recommend switching to asynchronous JavaScript Library initialization as soon as possible.

What will happen if I leave my old code unchanged?

If you do not update your code to make use of asynchronous calls, the JavaScript Library will fall back to the synchronous behavior upon accessing the library for the first time when a page is loaded. It will synchronously try to scan the first port. This is either the default port or the last known working one (i.e., the cached port number). The library will use web service-based functionality if it successfully connects to the port. Otherwise, it will fall back to using native plug-ins. All subsequent calls to the library will continue to reuse whatever method succeeded until page is reloaded.

What do I need to change in my code to make it properly work with new the JavaScript library?

The library now makes use of a new initialization method to perform asynchronous initialization through use of a callback method. Since the library performs initialization in the background, calling an SDK function prior to initialization results in an error. Your code should be updated to call the new **dymo.label.framework.init()** method while providing it a callback to be invoked by the library when initialization completes. Please note this callback is invoked whether or not initialization completes successfully.

Any code initialization involving calls to the JavaScript Library is typically located inside an event handler (i.e., **window.onload**). Below is typical JavaScript code demonstrating how to do this with the old synchronous architecture:

```
function startupCode() {
   /* access DLS SDK */
}

window.onload = startupCode;
```

This code will not work correctly under the new asynchronous architecture. It should be changed to call an intermediate "helper" method that firsts calls new **init()** method accepting your callback as a parameter. The library invokes the callback containing your original startup code after library initialization completes. Below is the updated JavaScript code that uses asynchronous initialization:

```
function startupCode() {
   /* access DLS SDK */
}

function frameworkInitHelper() {
        // init, then invoke a callback
        dymo.label.framework.init(startupCode);
}

window.onload = frameworkInitHelper;
```

If initialization is asynchronous, are other methods asynchronous as well?

The short answer is, "Yes!" As stated previously, synchronous AJAX requests are already deprecated, so our new JavaScript library contains new asynchronous equivalents for every method that calls the web service. These methods have the same names as their counterparts in the former architecture along with the **Async** suffix appended to them. They take the same number of parameters with corresponding types as well. The only major difference is that the new methods return a **Promise** object instead of the actual return value. You make use of this object by providing its **then()** method your callback function and one argument which receives the actual return value when the asynchronous method completes.

```
dymo.label.framework.getPrintersAsync().then(function(printers) {
   // printers list (same list the getPrinters() returns)
});
```

How do I make use of Promise objects in asynchronous programming?

The asynchronous JavaScript programming is not covered here. Please refer Google's documentation on Promise object usage and errors handling:

https://developers.google.com/api-client-library/javascript/features/promises

How can I tell if the web service is currently used by the JavaScript library?

The **dymo.label.framework.checkEnvironment()** method returns a **CheckEnvironmentResult** object along with multiple parameters (please refer existing documentation on **checkEnvironment()** method). The new JavaScript library contains an additional property within this object called **isWebServicePresent**. You can use this property to determine if the web service is actually used.

As with all other SDK-related methods, do not call **checkEnvironment()** until **init()** finishes initializing. Synchronous library initialization and web service discovery will occur if it is called before **init()** completes.

TROUBLESHOOTING

What is Trace functionality?

Our JavaScript Library now includes trace functionality that may help you debug your code. You can enable it by adding the following code line *before* calling the **init()** method:

```
dymo.label.framework.trace = 1;
dymo.label.framework.init(callback);
```

When using this code, you will see which steps the library takes every time it attempts to initialize (i.e., synchronous vs asynchronous initialization, port number if web service is discovered, fallback implementation selection, etc.).

Below is sample trace output when **init()** is called and the web service is present:

```
checkEnvironment > cachedWebPort : 41951
checkEnvironment > trying async service discovery
_createFramework > return _framework : undefined (async)
onEnvironmentChecked > checkResult isBrowserSupported : true, isFrameworkInstalled: true, isWebServicePresent: true
chooseEnvironment > WebServicePresent
```

The following is an example for case when **init()** is not called but the web service is running (fallback mode for legacy user code):

```
checkEnvironment > cachedWebPort : 41951
checkEnvironment > trying sync service discovery
Synchronous XMLHttpRequest on the main thread is deprecated because of its detrimental effects to the end user's experience.
checkEnvironment > web service found at port :41951
onEnvironmentChecked > checkResult isBrowserSupported : true, isFrameworkInstalled: true, isWebServicePresent: true
chooseEnvironment > WebServicePresent
```

This sample output demonstrates the case when the web service is not running (fallback mode for legacy plug-ins). Here the **init()** method was called; not calling the **init()** method will eventually lead to same fallback behavior but will additionally result in a hung UI during the initialization phase):

```
checkEnvironment > cachedWebPort : undefined
checkEnvironment > trying async service discovery
_createFramework > return _framework : undefined (async)
checkLegacyPlugins > WIN platform
checkLegacyPlugins > non-IE
checkLegacyPlugins > 'application/x-dymolabel'
onEnvironmentChecked > checkResult isBrowserSupported : true, isFrameworkInstalled: true, isWebServicePresent: false
chooseEnvironment > WIN
```

How do I perform error logging?

Improper DYMO Label SDK usage or multiple unexpected external factors may result in errors. You may need to retrieve logging data in these cases to help resolve problems.

Network and web service errors

To see communication errors and service fault messages, you will need to open Developer Tools (invoked by F12 key in most browsers) and open "Network" tab. If you see erroneous response in the list, you can click it and look for details in adjacent window (look and position is browser-specific). Response body would normally contain error description.

Web service log

The log file is located at **%LocalAppData%\DYMO\DLS8\DLSWebService.log**

If you are performing some specific tests, we recommend you delete the existing log before making a test run to eliminate any unnecessary log messages.