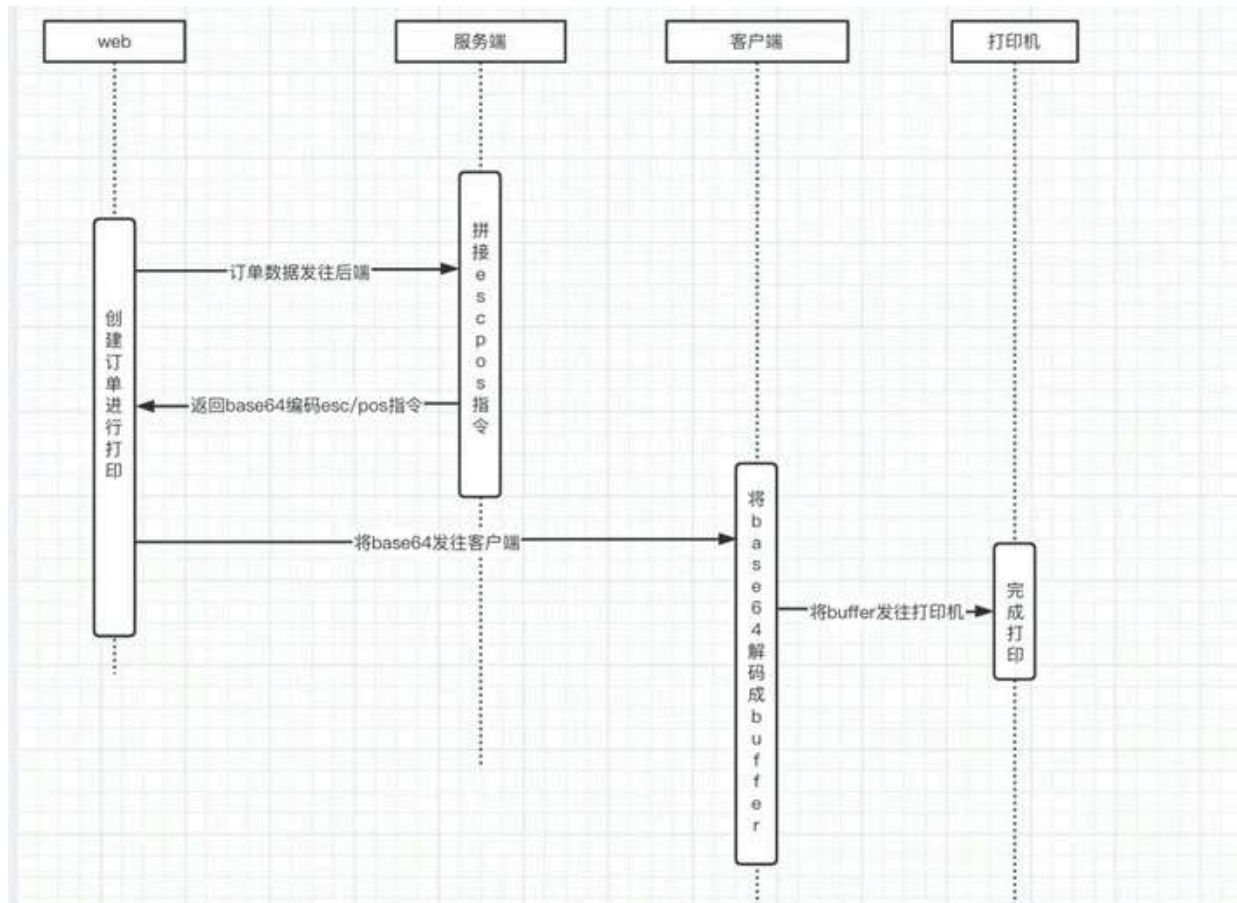


# 【转载】使用c++调用windows打印api进行打印的示例代码

原文: <https://www.jb51.net/article/189553.htm>

## 前言

在近期开发的收银台项目中，需要使用打印机进行小票打印，打印流程的时序图如下所示：



在客户的使用过程中，遇到一个问题，如果机器安装了打印机驱动，那么调用厂商提供的 sdk 进行打印的话，会导致出现小票只打印一半的情况，对此，需要绕过厂商 sdk 使用系统的打印才能够解决这一问题。

在 web 端打印中，需要调用浏览器打印 api 进行网页打印。这意味着，之前后端编写的esc/pos无法复用到，同时，前端还得花费精力来编写 html 以及css 来完成打印内容的排版，这无疑增加了复杂度以及工作量。正打算开始时，得到高人指点。

可以使用 windows api 进行打印

具体参见[这篇文档](#)

于是开始这方面的研究，功夫不负有心人，使用 windows api 完成了系统的打印，于是编写这篇文章记录踩过的坑。首先看看如何进行打印：

```
1  BOOL RawDataToPrinter(LPCTSTR szPrinterName, LPBYTE lpData, DWORD dwCount)
2  {
3      HANDLE    hPrinter;
4      DOC_INFO_1 DocInfo;
5      DWORD     dwJob;
6      DWORD     dwBytesWritten;
7
8      // Need a handle to the printer.
9      if (!OpenPrinter(szPrinterName, &hPrinter, NULL)) {
```

```

10     int y = GetLastError();
11     cout << "openFail" << y << endl;
12     return FALSE;
13 }
14
15 // Fill in the structure with info about this "document."
16
17 DocInfo.pDocName = LPSTR("My Document\0");
18 DocInfo.pOutputFile = NULL;
19 DocInfo.pDatatype = NULL; // LPWSTR("RAW\0");
20 // Inform the spooler the document is beginning.
21 if ((dwJob = StartDocPrinter(hPrinter, 1, (LPBYTE)&DocInfo)) == 0)
22 {
23     int x = GetLastError();
24     cout << "StartDocPrinter Fail" << x << endl;
25     ClosePrinter(hPrinter);
26     return FALSE;
27 }
28 // Start a page.
29 if (!StartPagePrinter(hPrinter))
30 {
31     EndDocPrinter(hPrinter);
32     ClosePrinter(hPrinter);
33     return FALSE;
34 }
35 // Send the data to the printer.
36 if (!WritePrinter(hPrinter, lpData, dwCount, &dwBytesWritten))
37 {
38     EndPagePrinter(hPrinter);
39     EndDocPrinter(hPrinter);
40     ClosePrinter(hPrinter);
41     return FALSE;
42 }
43 // End the page.
44 if (!EndPagePrinter(hPrinter))
45 {
46     EndDocPrinter(hPrinter);
47     ClosePrinter(hPrinter);
48     return FALSE;
49 }
50 // Inform the spooler that the document is ending.
51 if (!EndDocPrinter(hPrinter))
52 {
53     ClosePrinter(hPrinter);
54     return FALSE;
55 }
56 // Tidy up the printer handle.
57 ClosePrinter(hPrinter);
58 // Check to see if correct number of bytes were written.
59 if (dwBytesWritten != dwCount)
60     return FALSE;
61 return TRUE;
62 }

```

在文档中提到，打开打印机时"OpenPrinter"可以传入 null 以使用本地打印服务，因为不知道打印机名称，于是就传入了 null，结果在 StartDocPrinter 时一直提示失败，后来了解到使用 GetLastError 可以查看 error code，得到错误码后一对照，发现是 handle 是无效的，也就意味这 OpenPrinter 这一步骤没有打开需要的打印机。于是尝试使用 设备与打印机中的打印机名称，还真就连上了，成功调用打印服务。

但客户电脑上的打印机名称是不固定的，不能使用固定打印机名称，所以得拿到已经连接了的打印机列表，于是搜索到了 EnumPrinters 这一api，具体用法如下：

```
1 void getPrinterList() {
2     PRINTER_INFO_2* printerList;
3     unsigned char size;
4     unsigned long pcbNeeded;
5     unsigned long pcReturned;
6
7     EnumPrinters(PRINTER_ENUM_LOCAL, NULL, 2, NULL, 0, &pcbNeeded, &pcReturned);
8
9     if ((printerList = (PRINTER_INFO_2*)malloc(pcbNeeded)) == 0) {
10         return;
11     }
12
13     if (!EnumPrinters(PRINTER_ENUM_LOCAL, NULL, 2, (LPBYTE)printerList, pcbNeeded, &pcbNeeded, &pcReturned)) {
14         free(printerList);
15         return;
16     }
17
18     for (int i = 0; i < (int)pcReturned; i++) {
19
20         string printName(printerList[i].pPrinterName);
21         if (printerList[i].Attributes & PRINTER_ATTRIBUTE_NETWORK) {
22             cout << "网络打印机" << printName << endl;
23         }
24         else {
25             cout << "本地打印机" << printName << endl;
26         }
27     }
28
29     cout << "number " << pcReturned << endl;
30
31 }
```

通过这一方式，的确获取到了系统中可用的打印机，可是拿到可用的打印机后还是有一个问题：“如何知道哪一个是小票打印机”？

为此又进行了搜索，又找到了一个 api GetDefaultPrinter，用法如下：

```
1 string getDefaultPrinterName() {
2     DWORD size = 0;
3     GetDefaultPrinter(NULL, &size);
4
5     if (size) {
6         TCHAR* buffer = new TCHAR[size];
7         GetDefaultPrinter(buffer, &size);
8         string printerName(buffer);
9         return printerName;
10    }
11    else {
12        return "";
13    }
```

```
13     }
14 }
```

通过此方法获取到系统默认打印机，客户只需要设置默认的打印机为小票打印机就完美解决问题了。

以下是完整代码：

```
1  #include <iostream>
2  #include <windows.h>
3  #include "node.h"
4  #include "base64.h"
5
6  using namespace std;
7  using v8::FunctionCallbackInfo;
8  using v8::Isolate;
9  using v8::Local;
10 using v8::NewStringType;
11 using v8::Object;
12 using v8::String;
13 using v8::Value;
14 using v8::Integer;
15 using v8::Int8Array;
16
17 BOOL RawDataToPrinter(LPSTR szPrinterName, LPBYTE lpData, DWORD dwCount);
18 string getDefaultPrinterName();
19
20 void localPrintRawData(const FunctionCallbackInfo<Value>& args) {
21     Isolate* isolate = args.GetIsolate();
22     Local<v8::Context> context = isolate->GetCurrentContext();
23     v8::String::Utf8Value portString(isolate, args[0]);
24     std::string base64Str(*portString);
25
26     vector<BYTE> bytes = base64_decode(base64Str);
27     char* buffer = new char[bytes.size()];
28     copy(bytes.begin(), bytes.end(), buffer);
29     string printerName = getDefaultPrinterName();
30     if (printerName.size() > 0) {
31         printerName += "\\0";
32         wstring ws(printerName.begin(), printerName.end());
33         RawDataToPrinter(const_cast<char*>(printerName.c_str()), &bytes[0], bytes.size());
34     }
35     else {
36         cout << "no printer" << endl;
37     }
38 }
39
40 BOOL RawDataToPrinter(LPSTR szPrinterName, LPBYTE lpData, DWORD dwCount)
41 {
42     HANDLE hPrinter;
43     DOC_INFO_1 DocInfo;
44     DWORD dwJob;
45     DWORD dwBytesWritten;
46
47     // Need a handle to the printer.
48     if (!OpenPrinter(szPrinterName, &hPrinter, NULL)) {
49         int y = GetLastError();
```

```

50     cout << "openFial" << y << endl;
51     return FALSE;
52 }
53
54 // Fill in the structure with info about this "document."
55
56 DocInfo.pDocName = LPSTR("My Document\0");
57 DocInfo.pOutputFile = NULL;
58 DocInfo.pDatatype = NULL; // LPWSTR("RAW\0");
59 // Inform the spooler the document is beginning.
60 if ((dwJob = StartDocPrinter(hPrinter, 1, (LPBYTE)&DocInfo)) == 0)
61 {
62     int x = GetLastError();
63     cout << "StartDocPrinter Fial" << x << endl;
64     ClosePrinter(hPrinter);
65     return FALSE;
66 }
67 // Start a page.
68 if (!StartPagePrinter(hPrinter))
69 {
70     EndDocPrinter(hPrinter);
71     ClosePrinter(hPrinter);
72     return FALSE;
73 }
74 // Send the data to the printer.
75 if (!WritePrinter(hPrinter, lpData, dwCount, &dwBytesWritten))
76 {
77     EndPagePrinter(hPrinter);
78     EndDocPrinter(hPrinter);
79     ClosePrinter(hPrinter);
80     return FALSE;
81 }
82 // End the page.
83 if (!EndPagePrinter(hPrinter))
84 {
85     EndDocPrinter(hPrinter);
86     ClosePrinter(hPrinter);
87     return FALSE;
88 }
89 // Inform the spooler that the document is ending.
90 if (!EndDocPrinter(hPrinter))
91 {
92     ClosePrinter(hPrinter);
93     return FALSE;
94 }
95 // Tidy up the printer handle.
96 ClosePrinter(hPrinter);
97 // Check to see if correct number of bytes were written.
98 if (dwBytesWritten != dwCount)
99     return FALSE;
100 return TRUE;
101 }
102
103 void getPrinterList() {
104     PRINTER_INFO_2* printerList;

```

```

105 unsigned char size;
106 unsigned long pcbNeeded;
107 unsigned long pcReturned;
108
109 EnumPrinters(PRINTER_ENUM_LOCAL, NULL, 2, NULL, 0, &pcbNeeded, &pcReturned);
110
111 if ((printerList = (PRINTER_INFO_2*)malloc(pcbNeeded)) == 0) {
112     return;
113 }
114
115 if (!EnumPrinters(PRINTER_ENUM_LOCAL, NULL, 2, (LPBYTE)printerList, pcbNeeded, &pcbNeeded, &pcReturned)) {
116     free(printerList);
117     return;
118 }
119
120 for (int i = 0; i < (int)pcReturned; i++) {
121
122     string printName(printerList[i].pPrinterName);
123     if (printerList[i].Attributes & PRINTER_ATTRIBUTE_NETWORK) {
124         cout << "网络打印机" << printName << endl;
125     }
126     else {
127         cout << "本地打印机" << printName << endl;
128     }
129 }
130
131 cout << "number " << pcReturned << endl;
132
133 }
134
135 string getDefaultPrinterName() {
136     DWORD size = 0;
137     GetDefaultPrinter(NULL, &size);
138
139     if (size) {
140         TCHAR* buffer = new TCHAR[size];
141         GetDefaultPrinter(buffer, &size);
142         string printerName(buffer);
143         return printerName;
144     }
145     else {
146         return "";
147     }
148 }
149
150 void Initialize(Local<Object> exports) {
151     NODE_SET_METHOD(exports, "localPrintRawData", localPrintRawData);
152 }
153
154 NODE_MODULE(zq_device, Initialize)

```

## 参考:

<https://support.microsoft.com/zh-cn/help/138594/howto-send-raw-data-to-a-printer-by-using-the-win32-api>

<https://docs.microsoft.com/en-us/windows/win32/printdocs/openprinter>

<https://stackoverflow.com/questions/6682286/understanding-a-c-sample-printers-handles-strings>

<https://social.msdn.microsoft.com/Forums/windowsdesktop/en-US/a27c6615-9452-44b1-90fc-9b91b15f0e50/openprinter-returning-errorinvalidprintername1801-when-called-with?forum=windowsgeneraldevelopmentissues>

<https://social.msdn.microsoft.com/Forums/vstudio/en-US/de7c55a1-ae63-49c9-a87a-fe3bf32822e4/how-to-use-the-enumprinters-function-to-be-able-to-classify-installed-printers-into-quot-network?forum=vclanguage>

<https://docs.microsoft.com/en-us/windows/win32/debug/system-error-codes--0-499->

<https://docs.microsoft.com/zh-cn/windows/win32/debug/system-error-codes--1700-3999-?redirectedfrom=MSDN>