POWAPOS
payment freedom

PowaPOS SDK User Guide - Windows

Document Version 1.3

Document Date: 18 June 2015

POWAPOS
payment freedom

Document Control

Version and Date	Revisions	Document Owner
Version 1.0 Oct, 2014	First release.	Jossuan Martell Abel Duarte Miguel Silva
Version 1.1 Feb, 2015	Add ***Hardware and Software Requirements*** section. Add ***Troubleshooting*** section. Add instructions and screen shots for installing signed device driver. Clean up sample code and add section with instructions to create a sample application	Mark Williams
Version 1.2 June 11, 2015	Add SDK target version to Section 2.1. ***API and Reference Documentation.*** Add new supported call back events for SDK vs.1.3 to Section 3.2.3 ***Setting Up Your Code to Receive Events.*** Modify code to comply with SDK vs.1.3 changes in Section 3.2.5 ***Write Code to Display Connection Status.***	Mark Williams
Version 1.3 June 18, 2015	Change target version to Section 2.1. ***API and Reference Documentation.*** Modify code for supported call back events for SDK vs.1.4 to Section 3.2.3 ***Setting Up Your Code to Receive Events.***	Mark Williams

Table of Contents

1 Introduction

The *PowaPOS SDK* offers a complete set of functionality to implement Point of Sale operations effectively reducing the complexity of *PowaPOS T25* based applications.

This document provides a quick walkthrough on the *PowaPOS SDK* artefacts and illustrates its usage within a *PowaPOS T25* Payment Application with sample code.

This User Guide is intended for developers integrating *PowaPOS SDK* into their own Payment Applications.

1.1 Document Structure

The document presents an initial overview of the PowaPOS architecture and components followed by the SDK package artifact description and a step-by-step illustration of how to integrate the SDK in a Payment Application.

1.2 Terms and Concepts

The following table describes the terms and acronyms found across the document.

Term	Acronym	Description
Application Programming Interface	API	A set of calling conventions that defines how a service is invoked through software. An API enables programs written by users or third parties to communicate with certain vendor-supplied software.
Mobile Communications Device	MCD	Portable device with access to a data network and enhanced user interface capabilities. Can be a smartphone, a tablet or a similar dedicated device.
PIN Entry Device	PED	A Hardware device dedicated to securely capture and cipher PIN codes according to PCI-PTS requirements.
Point of Sale	POS	The terminal where the customer and card acceptor are located at the time a card is used for purchase or cash.
Software Development Kit	SDK	A set of software development tools that allows for the creation of applications for a certain software package.
Microcontroller Unit	MCU	Generally referred to in this document as the controller for the T-Series hardware.

1.3 Support Resources

Support for PowaPOS APIs and SDKs can be found online at the *PowaPOS SDK Partner Portal* - https://powaposdeveloper.powa.com.

1.4 Hardware and Software Requirements

Any PC, Laptop or Tablet running Windows 7 or 8.1.

Microsoft Visual Studio 2013 or later - Express or Pro.

Connection to the PowaPOS handheld scanner or other Bluetooth device requires Bluetooth 2.0 support on the Windows PC, Laptop or Tablet.

2 PowaPOS SDK Artefact Overview

The following sections describe the contents of the SDK package.

2.1 API and Reference Documentation

This document targets the PowaPOS SDK for Windows version 1.4 and later.

The following documents are provided with the SDK and are referenced across this guide:

[1] PowaPOS SDK API Specification

This document provides an initial context of the PowaPOS architecture and describes the Application Programming Interface required to use the SDK. This document focuses on the User Interface and high-level interactions between the Payment Application and the SDK.

[2] The SDK reference documentation

This hyperlinked documentation is generated automatically based on the SDK software. It provides a reference for developers describing the exact syntax of methods, event handlers and interface objects.

2.2 SDK Libraries

The PowaPOS SDK libraries are provided in the form of DLLs (**D**ynamic **L**ink **L**ibrary) to be used in Windows projects. These libraries must be included and referenced in your application and provide the API and other functionality described in the companion documentation.

2.3 Supported Peripherals

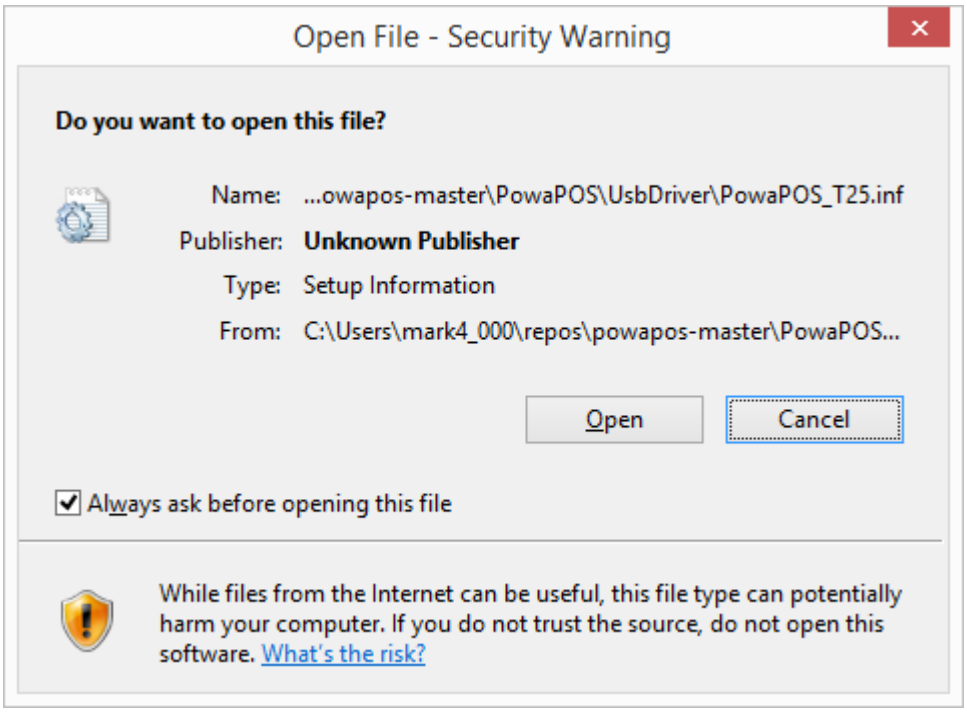
As of today the SDK supports the T-Series peripherals namely printers, bar code scanner, cash drawers, USB ports and rotation sensor.

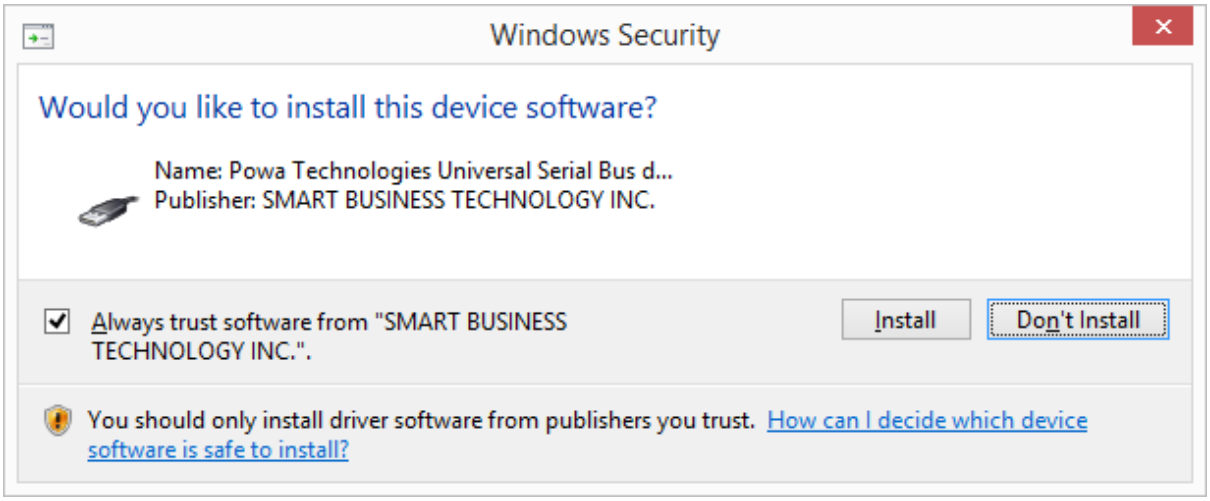
2.4 Device Driver

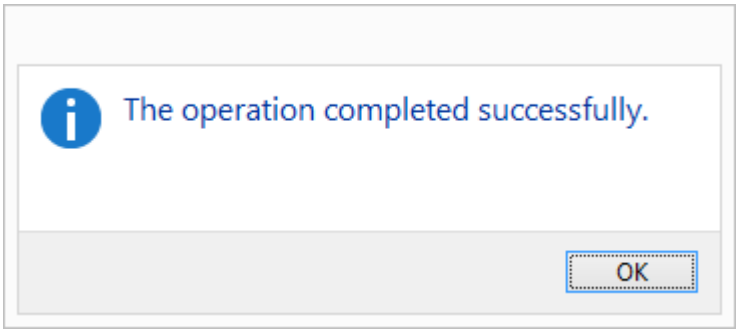
Windows device drivers for supported peripherals are supplied with this SDK and must be installed on your Windows PC, laptop or tablet. Our PowaPOS device driver is a signed component as required for Windows 8.1 and may require that your Windows device contain the latest Microsoft updates in order to be verified. If you are using a Windows 8.1 device please make sure that it is installed with the latest updates prior to installation of the driver.

To install the PowaPOS device driver do the following:

1) Locate the 'PowPOS_T25.inf' and 'PowPOS_T25.cat' files included with the SDK package and copy them to a directory on the hard drive of your Windows device. Be sure to remember where you copied the files.
2) Using Windows Explorer navigate to the location of the driver files.
3) Right click the 'PowPOS_T25.inf' file and select "Install" from the pop up menu.
4) You may get an "Open File – Security Warning" dialog (see below). If you get this dialog click 'Open'.
5) You will then get a pop up dialog asking "Do you want to allow the following program to make changes to this computer?" Click 'Yes'.
6) You will then get another pop up dialog asking if you would like to install device driver software from "Smart Business Technology, Inc.". Select 'Install'.
7) You should then get a final pop up dialog saying that the "The operation completed successfully". Click 'OK' on this dialog.
8) Plug your PowaPOS T25 device into any USB port on your Windows PC and power on the T25.
9) Run the 'Device Manager' Windows utility. In Device Manager under 'Universal Serial Bus Devices' you should see an entry for 'PowaPOS T25'.
10) You are now ready to write application software that can access your PowaPOS T25 device.







3 Getting Started

The following sections describe the step-by-step procedures to create a simple Windows application and illustrate how to use the *PowaPOS SDK* to drive the *PowaPOS T25*.

3.1 Sample Projects

The SDK package includes a sample project (*'PowaPOS.Samples.sln'*) that illustrates SDK integration in a simple application. You can refer to this application for guidance as you develop an application of your own to drive to PowaPOS T25. Parts of the code in this sample project are used in this guide to illustrate the step-by-step procedures to integrate to the SDK.

3.2 Creating a Simple Application

The following sections show a step-by-step implementation of a PowaPOS SDK project. Follow these steps to set up and perform initial configuration for your own T25 project.

3.2.1 Create and Configure Your Solution

Open Visual Studio and create a new C# Windows Desktop WPF application.

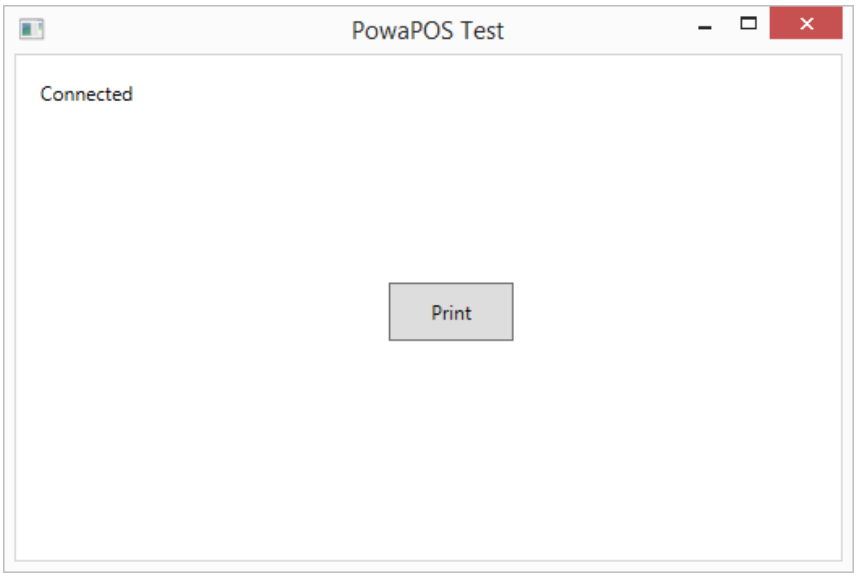
Select .NET Framework 4.5.1.

Name your project 'PowaPosTest' and place it in a directory of your choice. (This complete sample project is included with your PowPOS SDK package).

Place a single button on your MainWindow dialog and call it 'btnPrint'.

Add a label control to the upper left side of your dialog and call it 'ConnectionStatus'.

Double click the print button to create a click event handler for your button in your MainWindow C# code behind file.



3.2.2 Add References to the SDK Libraries

You must now add references to the SDK DLLs in your project. Copy the following files from the SDK package to the root directory of your solution.

- Powa.Win.Sdk.Commons.dll

-Powa.Win.Sdk.PowaPOS.dll

-WinUsbNet.dll

In Solution Explorer, right click on the 'References' menu item under your solution and select 'Add Reference'.

On the left side of the 'Reference Manager' dialog, select 'Browse' and then click the <Browse> button.

Navigate to your solution root directory and select the three SDK DLLs that you copied there and click <OK>.

In your MainWindow code behind file (MainWindow.xaml.cs) add references to the following namespaces.

```
using Powa.Win.Sdk.PowaPOS.Core;
using Powa.Win.Sdk.PowaPOS.Core.DataObjects;
using Powa.Win.Sdk.PowaPOS.Core.Callback;
using Powa.Win.Sdk.PowaPOS.Drivers.S10;
using Powa.Win.Sdk.PowaPOS.Drivers.TSeries;
```

3.2.3 Setting Up Your Code to Receive Events

Your application requires a callback client that is derived from the 'PowaPOSCallback' class. This class will contain the event handlers to handle responses from the PowaPOS T25. Add the following code as a subclass to your MainWindow class.

```
private class PowaPOSCallbackClient : PowaPOSCallback
{
    private MainWindow parent;
    public PowaPOSCallbackClient(MainWindow parent)
    {
        this.parent = parent;
    }

    public override void OnMCUInitialized(PowaPOSEnums.InitializedResult result)
    { }
    public override void OnMCUAvailableFirmwareResult(PowaPOSEnums.PowaFirmwareAvailableResult
       result, PowaFirmwareInfo info)
       { }
    public override void OnMCUFirmwareDownloaded(PowaPOSEnums.PowaFirmwareDownloadResult result,
       PowaFirmwareInfo info, byte[] bytes)
       { }
    public override void OnCashDrawerAttached()
    { }
    public override void OnCashDrawerDetached()
    { }
    public override void OnCashDrawerStatus(PowaPOSEnums.CashDrawerStatus status)
    { }

    public override void OnUSBDeviceAttached(PowaPOSEnums.PowaUSBCOMPort port)
    { }
    public override void OnUSBDeviceDetached(PowaPOSEnums.PowaUSBCOMPort port)
    { }
    public override void OnUSBReceivedData(PowaPOSEnums.PowaUSBCOMPort port, byte[]
        data)
    { }
    public override void OnMCUFirmwareUpdateFinished()
    { }
    public override void OnMCUBootloaderUpdateStarted()
    { }
    public override void OnMCUBootloaderUpdateProgress(int progress)
    { }
    public override void OnMCUBootloaderUpdateFinished()
    { }
    public override void OnMCUBootloaderUpdateFailed(PowaPOSEnums.
        BootloaderUpdateError error)
    { }
    public override void OnMCUFirmwareUpdateProgress(int progress)
    { }
    public override void OnMCUFirmwareUpdateStarted()
    { }
    public override void OnMCUSystemConfiguration(Dictionary<string, string>
        configuration)
    { }
```

```
    public override void OnPrintJobResult(PowaPOSEnums.PrintJobResult result)
    { }
    public override void OnRotationSensorStatus(PowaPOSEnums.RotationSensorStatus
        status)
    { }
    public override void OnScannerInitialized(PowaPOSEnums.InitializedResult result)
    { }
    public override void OnScannerRead(string data)
    { }
    public override void OnPrinterOutOfPaper()
    { }
}
```

Then create the callback client object in the constructor of 'MainWindow':

```
PowaPOSCallbackClient powaPOSCallbacks;
Powa.Win.Sdk.PowaPOS.PowaPOS powaPOS;

public MainWindow()
{
    InitializeComponent();

    powaPOSCallbacks = new PowaPOSCallbackClient(this);
    powaPOS = new Powa.Win.Sdk.PowaPOS.PowaPOS(powaPOSCallbacks);
```

3.2.4 Initialize the T-Series

Now copy the configuration file from the SDK package to the root directory of your solution:

```
config_tseries.xml
```

Next we will read the device driver identifier from the PowaPOS configuration file and create our reference to the T25. Still in the constructor of 'MainWindow' add the following code:

```
    var fileName = System.IO.Path.Combine(AppDomain.CurrentDomain.BaseDirectory,
        @"config_tseries.xml");
    var xdoc = XDocument.Load(fileName);
    var guid = Convert.ToString(xdoc.Descendants("guid").First().Value);
    var tseries = new PowaTSeries(new Guid(guid));
    powaPOS.AddPeripheral(tseries);

}
```

The 'guid' is the T25 Windows driver identifier. This value is contained in the TSeries configuration file (config_tseries.xml) included with the PowaPOS SDK and should not be changed unless you are instructed to do so by PowaPOS support.

When your application ends you MUST call the 'Dispose()' method of the 'powaPos' object to release resources. Add the following code to the 'Closing' event of 'MainWindow'.

```
private void Window_Closing(object sender, System.ComponentModel.CancelEventArgs e)
{
    this.powaPOS.Dispose();
}
```

3.2.5 Write Code to Display Connection Status

Add the following code to the 'OnMCUInitialized()' event handler of the *'PowaPOSCallbackClient'* class and then run the application. The word "Connected" should display in the upper left corner of your application dialog.

```
public override void OnMCUInitialized(PowaPOSEnums.InitializedResult result)
{
    Application.Current.Dispatcher.Invoke(new Action(() =>
    {
        parent.ConnectionState.Content = (parent.powaPOS.MCU.GetConnectionState() ==
        Powa.Win.Sdk.Commons.Base.PowaEnums.ConnectionState.CONNECTED)
             ? "Connected"
             : "Disconnected";
    }));
}
```

3.2.6 Write code to print

Now let's print something! Add the following code to the event handler for your 'Print' button. When you click the 'Print' button the PowaPOS T25 should print a receipt.

```
private void btnPrint_Click(object sender, RoutedEventArgs e)
{
    decimal total = 0;
    String text = "\n\n";
    text += "     POWA Technologies \n";
    text += "        123 Hialeah \n";
    text += "       Miami, Florida \n";
    text += " \n";
    text += String.Format(" Date: {0:MM/dd/yyyy} Time: {0,8:t} \n", DateTime.Now);
    text += " ------------------------------ \n";
    text += " SALE                           \n";
    for (int i = 0; i < 5; i++)
    {
        text += String.Format(" 30067855{0}  Ticket Item {1}  {2:#0.00}   \n", i, i+1,
            10+i);
        total += (10 + i);
    }
    text += " \n";
    text += String.Format(" Subtotal                  {0:###0.00}   \n", total);
    text += " Tax                         0.00   \n";
    text += " ------------------------------ \n";
    text += String.Format(" TOTAL                     {0:###0.00}   \n", total);
    text += "------------------------------ \n";
    text += " \n\n";
    powaPOS.PrintText(text);
}
```

3.2.7 Add Additional Functionality

You can add additional functionality to you test application as desired by using the additional capabilities of the API. Additional sections of this document show you how to get the rotation sensor status and to initialize the scanner and retrieve scanner data.

3.2.8 Write code to use the rotation sensor status

The following code illustrates how to trigger the Rotation sensor status request and the handling of the resulting event:

```
//request the status reading
powaPOS.RequestMCURotationSensorStatus();
//receives and handles the status
class PowaPOSCallbackClient : PowaPOSCallback
{
      public void OnRotationSensorStatus(PowaPOSEnums.RotationSensorStatus status)
      {
          //do something with status
      }
}
```

3.2.9 Initialize the Scanner

The PowaPOS T25 barcode scanner is interfaced to your Windows device via Bluetooth and must be paired prior to use. For pairing purposes the Bluetooth device name of the scanner is "PowaPOS xxxxxxx" where 'xxxxxxx' is the last 7 digits of the serial number found on the back of the scanner. After the scanner is paired, you can add code to initialize it and handle the scanner read event as illustrated below.

```
PowaS10Scanner scanner = new PowaS10Scanner();
powaPOS.AddPeripheral(scanner);
```

3.2.10 Write code to handle scanned barcodes

The '*OnScannerRead()*' of the *'PowaPOSCallbackClient'* event returns scanner data after a barcode scan:

```
class PowaPOSCallbackClient : PowaPOSCallback
{
       public override void OnScannerRead(string data)
       {
           //do something with the data
       }
}
```

4 Troubleshooting

The following sections some common troubleshooting issues along with solutions.

Support for PowaPOS APIs and SDKs can be found online at the *PowaPOS SDK Partner Portal* - https://powaposdeveloper.powa.com.

4.1 USB Driver Exception

"Open device did not match object passed in 'OnDeviceChange' event. This error can be caused by not calling the 'Dispose()' method of the 'Powa.Win.Sdk.PowaPOS.PowaPOS' object at termination of your application."

The issue occurs when part of your application process remains running in the background and causes the USB driver to hold open the connection to the T25. We currently know of three possible causes for this issue:

1) The application does not call '… PowaPOS.Dispose()' before exiting the program. Be sure to dispose the PowaPOS object prior to exiting your application.
2) The application left a background thread running that uses the PowaPOS object. Be sure to close any background threads especially if they are using PowaPOS resources.
3) The application crashed and hence did not exit normally.

The techniques for items 1 and 2 are demonstrated in our sample applications delivered with the SDK.

Once this error occurs you must find the application background process in device manager and close it. The background process should be easy to find because it will have the same name as your application. You can also reboot the computer to free the connection to the T25 or add code to the beginning of your app to automatically close any previous running instances.

4.2 PowaPOS Configuration Exception

“'config_tseries.xml' is missing ..."

This error can occur when the PowaPOS demo applications cannot find 'config_tseries.xml'. Make sure that 'config_tseries.xml' is present in the same directory as the executable.