Aardvark Embedded Solutions

PayLink
Dot Net User Guide

Version 2.0

Issue	Date	Author	Comments
1.0	24/02/2011	Martin Stafford	Initial Release
2.0	21/03/2020	Dave Bush	Major Update

Table of Contents

1 Introduction

1.1 User

The reader for whom this document is designed is a Windows developer familiar with Visual Studio and C# who is just about to start using the Paylink money handling system.

1.2 Contents

This document describes the class libraries, examples and other resources that are a part of the Paylink Distribution, in the folder SDK

Note: all the examples are provided on an "as is" basis - they were written as exercises by Aardvark staff and are not necessarily suitable for use as a basis for operational software.

1.3 Dot Net Versions

The main AESImhei Class Library assemblies and all the example programs are all compiled for Dot Net 4.0

1.4 Compiler Versions

The AESImhei Class Libraries and the example projects are generated / compiled with VS2017.

2 Basic Usage

This section of the document describes the C# examples supplied as a part of the SDK.

2.1 Before you start

These examples are useful to run and read in order to get started using Paylink with C#, or any other Dot Net language. Before you actually look at them it is a **really** good idea to have read the 1st 2 pages in the *Paylink* section on page 9 of the **Milan / Paylink System Manual**, specifically the sub sections "*Money representation, Acceptance"* and *"Payment".* If you're interested in using the cashless facility then also read the *Cashless Processing* section on page 40.

To understand the details of all the function calls used, you should refer to the **Milan / Paylink Application Program Interface Manual** which details all the calls that are available with Paylink.

Don't forget that these examples will only run if the Paylink USB driver program is running, and has been configured to match the real money peripherals on your system.

2.2 Example C# code

Each example is provided as the source of a VS2017 solution. To see the code in action just open the appropriate solution (.sln) file and compile / run the project.

The idea behind the 5 very simple samples is that they illustrate the AESImhei calls and data in a functioning program without the code in the application getting in the way.

Each solution should build without errors and then will run without errors, provided the USB Driver Application is running and the class library and basic AesImhei.dll interface are available (which will be the case if you ran the standard installation). If the USB Application is not running or is running but cannot detect a Paylink device, the examples will show a non-zero Open result.

2.2.1 Basic PayIn

This very simple command line application demonstrates the following:
- opening the Paylink interface,
- obtaining the starting position for money in
- obtaining details of all the Acceptors and their coins that Paylink knows about
- turning on the Paylink system
- checking for money to be inserted
- disabling and enabling all the acceptors

Source File: "SDK\Examples\C #\Basic PayIn\BasicInput.cs"

Paylink Calls: `AESImhei.OpenMHE()`
                 `AESImhei.CurrentValue()`
                 `AESImhei.ReadAcceptorDetails()`
                 `AESImhei.WriteAcceptorDetails()`
                 `AESImhei.EnableInterface()`

OS facilities   `System.Console.WriteLine()`
    and   `Thread.Sleep().`

2.2.2 Basic Payout

Again a very simple command line application demonstrating the following:
- opening the Paylink interface,
- obtaining the starting position for money out
- obtaining details of all the Dispensers that Paylink knows about
- turning on the Paylink system
- issuing a request for Paylink to pay out some money.
- checking the progress and result of the request
- checking how much money was actually paid out.

Source File: "`SDK\Examples\C #\Basic PayIn\BasicPayt.cs`"

Paylink Calls: `AESImhei.OpenMHE()`
`AESImhei.CurrentPaid()`
`AESImhei.ReadDispenserDetails()`
`AESImhei.EnableInterface()`
`AESImhei.PayOut()`
`AESImhei.LastPayStatus()`

OS facilities   `System.Console.WriteLine()`
   and   `Thread.Sleep().`

2.2.3 Simple PayIn

As an application this is a bit more complicated as it is a normal Windowing application. Apart from a bit of code to create arrays for the coin display textboxes, the code is again very simple. It shows a timer being used to:
- open the Paylink interface,
- turn on the Paylink system
- obtain the starting position for money in
- obtain details of the current acceptor and its coins
- checking for money to be inserted

and simple code for three buttons:
- NextButton   change which acceptor has its details displayed.
- InhibitAcceptor   Inhibits acceptance on the acceptor currently shown
- EnableAcceptor   Inhibits acceptance on the acceptor currently shown

Code Source File: "`SDK\Examples\C #\Simple PayIn\Form1.cs`"

Paylink Calls: `AESImhei.OpenMHE()`
`AESImhei.CurrentValue()`
`AESImhei.ReadAcceptorDetails()`
`AESImhei.WriteAcceptorDetails()`
`AESImhei.EnableInterface()`

2.2.4 Simple Payout

As an application this is a bit more complicated as it is a normal Windowing application but the code is again very simple. It shows a timer being used to:

- open the Paylink interface
- turn on the Paylink system
- obtain the starting position for money out
- monitor the ongoing situation with money out
- monitor the ongoing status of the payout system

and the code for a single button **PayButton** which expects an amount to have been typed into a test field and then calls `AESImhei.PayOut()` with that value. The ongoing progress of the resultant pay-out is monitored by the timer above.

Code Source File: "`SDK\Examples\C #\Simple Payout\Form1.cs`"

Paylink Calls: `AESImhei.OpenMHE()`
                         `AESImhei.CurrentPaid()`
                         `AESImhei.EnableInterface()`
                         `AESImhei.LastPayStatus()`
                         `AESImhei.PayOut()`

2.2.5 Cashless Demo

This application is designed to help people use the Paylink support for cashless peripherals. As you can read in the **Milan / Paylink System Manual**, the Paylink model for using such peripherals is slightly complicated, and so this example allows developers to *use* the application to see the Paylink cashless system operate, and to then see the interface calls that support this.

The application itself consists of a timer loop that continually displays the current state of the cashless system, and a button for each of the possible cashless system function calls. In addition there is a small amount of logic that only displays those buttons that are a sensible function call given the current state.

Code Source File: "`SDK\Examples\C #\Cashless Demo\Cashless Dotnet.sln`"

Paylink Calls: `AESImhei.OpenMHE()`
                         `AESImhei.EnableInterface()`
                         `AESImhei.CashlessReadData()`
                         `AESImhei.CashlessEnable()`
                         `AESImhei.CashlessDisable()`
                         `AESImhei.CashlessRequestCredit()`
                         `AESImhei.CashlessTakeCredit()`
                         `AESImhei.CashlessRefuseCredit()`
                         `AESImhei.CashlessReset()`
                         `AESImhei.CashlessCancelCredit()`
                         `AESImhei.CashlessDisable()`

3   Class Library details

3.1 The Class Library

Two functionally identical Class libraries are located at "SDK\Dot Net", called AesImhei.Net.dll for 32 bit environments and AesImhei64.Net.dll for 64 bit environments. One of these should be added to the reference section of any Paylink related C# or VB.Net project you create, chosen as described in the next section.

In order to then use the resultant dot net program with a Paylink you need to have the following:
•   A running instance of Paylink.exe
•   AesImhei.Net.dll   - in the same folder as your assembly (.exe)
•   AesImhei.dll   - C:\Windows\System (Installed by Paylink installer)
                                         – or in the same folder as your assembly (.exe)

Or if you are looking at a 64 bit environment:
•   A running instance of Paylink.exe
•   AesImhei64.Net.dll   - in the same folder as your assembly (.exe)
•   AesImhei64.dll   - C:\Windows\System (Installed by Paylink installer)
                                         – or in the same folder as your assembly (.exe)

For reference, the source of the AesImhei.Net class library, and an associated VS2017 project, can be found at "SDK\Dot Net\AesImhei.Net". Both AesImhei.Net.dll and AesImhei64.Net.dll use the same source – the only difference is the .lib file they use to link to the native DLL.

If it is necessary, for any reason, you should have no problem in opening the solution file at "SDK\Dot Net\AesImhei.Net\AesImhei.Net.sln" and rebuilding everything.

3.2 Which AESImhei should you use?

If you are running 32-bit Windows on both your development and target systems, then you just use AESImhei.Net.dll in your project(s).

If you are running 64-bit Windows on both your development and target systems, then both systems will probably be fully 64 bit, so you should use AESImhei64.Net.dll in your project(s).

Execution problems on 64 bit Windows systems.
The native mode DLL used by Paylink can cause execution errors when executing Dot Net programs under a 64 version of Windows.

The problem arises because a program executing on a 64 version of Window can either be 32 bit or 64 bit, but cannot combine the two.
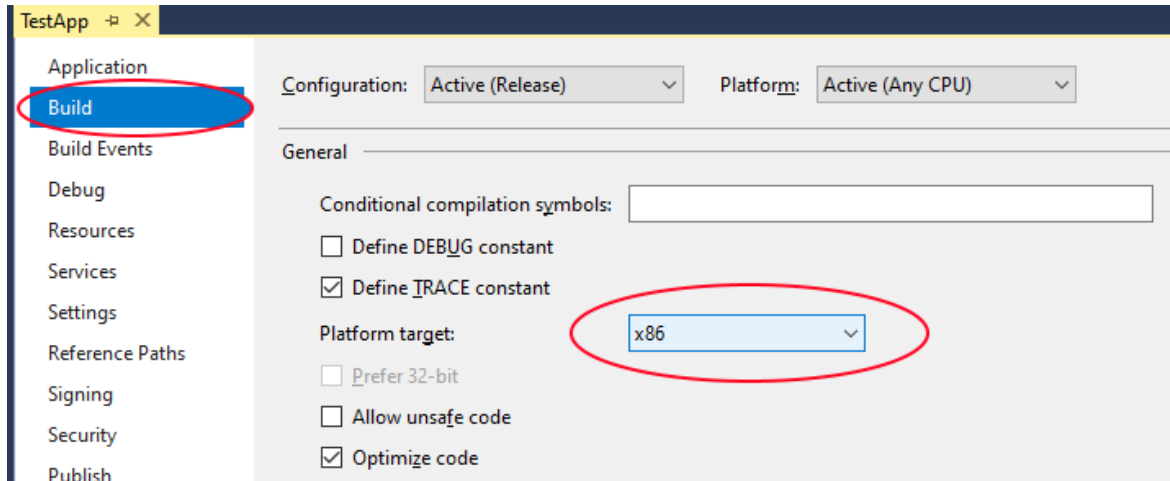
Dot Net (C#) programs / assemblies (and DLLs) are interpreted and a 64 bit version of Windows has **two** separate interpreters available; a 32 bit one and a 64 bit one. The program / assembly created by compiling the C# / VB source is one single format. Most Dot Net projects do not care which interpreter is used and are marked as Neutral. With 64 bit Windows systems, these Neutral programs cause the 64-bit interpreter to be run, as it is the default on those systems.

The execution time error can occur because you have to use a Paylink native DLL. When control is transferred to a Native DLL, the DLL in question has to match the calling program - in this case *the interpreter*. So, if the system is running the 64 bit interpreter, you have to be *explicitly* calling AESImhei64.dll - which you do by using AESImhei64.Net.dll. If it is running the 32 bit interpreter, you have to be *explicitly* calling AESImhei.dll - which you do by using AESImhei.Net.dll

If you are developing on a 64 bit system, but deploying on a 32 bit system, and you build and test your project using AESImhei.Net.dll, then by default it will not run on the 64 bit system. To do this you have to mark the program as *only* to be run by the 32-bit (Win32 / x86) interpreter.

The marking must be done on the project for the actual application. (Program launch is when the interpreter is chosen.) Processor markings on the .Net DLLs *are not relevant* and ignored.

The SDK as released includes projects for TestApp and TestApp64 - which show the platform target already selected e.g.

3.3 Dot Net version before 4.0

The class libraries projects described above are not backwards compatible and cannot be used with projects using Visual Studio earlier than VS2017 or Dot Net 3.5 or Dot Net 2.0.

If you are using one of these, then a 2nd folder "AesImhei.Net.2.0" contains the class libraries (and basic test program) with a solution file for Visual Studio 2005 and assemblies for Dot Net 2.0. If your environment is later than this, then you should be able to open the VS2005 solution file with any Visual Studio later than 2005, which will import and convert the solution file and then build the assembly to fit your desired application.

4 Full Demonstration Programs

To illustrate a more comprehensive set of the available Paylink facilities the standard Demo program has been partially rewritten in C# and Visual Basic.

Both of these demonstration applications have been written using Visual Studio 2017 and built to run under Microsoft's .NET Framework version 4 or later. The complete project and all source files are included as a part of the SDK.

To form a functioning system you will require the standard Paylink items, these applications are compatible with versions 1.12.4 or higher of the Paylink firmware.

4.1 C# Demonstration

The "SDK\Examples\C #\Full C#Demo" folder on the distribution contains the entire source and project file for C#Demo.exe, a demonstration program written in C# to illustrate the usage of most of the interface. This is the C# equivalent to the standard Demo.exe program that is a part of the standard release. A built executable is included as a part of the installation, and should be ready to run at "SDK\Examples\C #\Full C#Demo\ C#Demo\bin\Debug\C#Demo.exe". If the USB Application is not running or is running but cannot detect a Paylink device, the error message 'MHE Open Error nn' occurs at Startup.

If you need to re-compile this with your development environment for any reason, the solution file should just load and rebuild.

The C#Demo program uses the standard Aesimhei.Net.dll class library to communicate via the native Aesimhei.dll with the USB Driver Application, which in turn communicates with the Paylink device via the USB line.

4.2 VB.Net Demonstration

The "SDK\Examples\VB Demo" folder on the distribution contains the entire source and project for the VB.Net PaylinkDemonstration.exe program, a similar demonstration program written in VB to illustrate the usage of most of the interface.

Again the executable is included as a part of the installation, and should be ready to run at "SDK\Examples\VB Demo\PaylinkDemonstration\bin\Debug\PaylinkDemonstration.exe". If the USB Application is not running or is running but cannot detect a Paylink device, the error message 'MHE Open Error nn' occurs at Startup.

If you need to re-compile this with your development environment for any reason, the solution file should just load and rebuild.
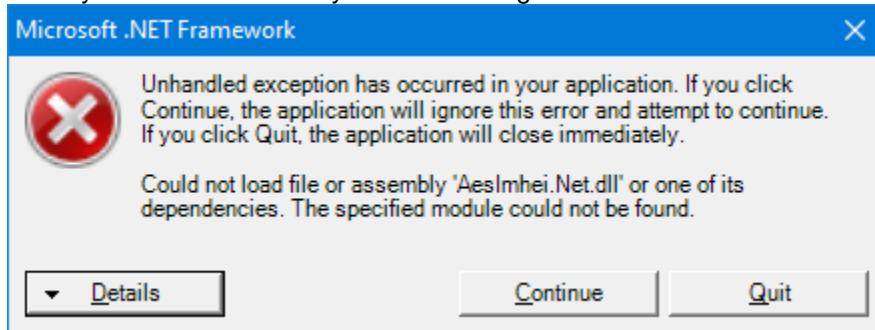
5   vcruntime140.dll (Dot Net load errors)

vcruntime140.dll is a part of the Microsoft Visual C redistributable package, and is required in order to use the Paylink Dot Net Class library.

Normally this dll will be installed on your development machine as a part of the Visual Studio installation, but it is also required on your target system. (Note that the compiled versions of the Dot Net demonstration programs in the release have a copy of vcruntime140 alongside the applications.)

IF vcruntime140.dll cannot be found on a system, then any attempt to run a Dot Net Paylink application will generate an exception of:
System.IO.FileNotFoundException: Could not load file or assembly 'AesImhei.Net.dll' or one of its dependencies.
File name: 'AesImhei.Net.dll'

Usually this will be shown by the error dialog:

To fix this, you can download the official Microsoft redistributable releases from The Microsoft Website (ignore the reference to 2015), or you can use the releases located in the folder "SDK\Dot Net".

Alternatively, if you are using Windows 10 on your development and target machines you can manually copy just **the correct** vcruntime140.dll onto the machine in question.

If you are using the 64 bit AESImhei64.Net.dll, then you can just copy the 64 bit vcruntime140.dll from C:\Windows\System32 on your 64 bit development system onto the 64 bit target machine – either into the application folder or into C:\Windows\System32

If you are using the 32 bit AESImhei.Net.dll *and* you have a 32 bit development system, then you can just copy the 32 bit vcruntime140.dll from C:\Windows\System32 on your development system onto the target machine – either into the application folder or into C:\Windows\System32

If you are using the 32 bit AESImhei.Net.dll and you have a 64 bit development system, then you need to copy the 32 bit vcruntime140.dll from C:\Windows\SysWOW64 on your development system onto the 32 bit target machine – either into the application folder or into C:\Windows\System32

If you are using Windows 7 or earlier, then vcruntime140.dll requires a number of subsidiary interface DLLs, so using the official Microsoft redistributable releases is recommended.