



# **Aardvark Embedded Solutions**

# **PayLink Linux User Guide**

Issue	Date	Author	Comments
D1	19/04/2006	Len Meakin	Initial release for review.
D2	30/04/2006	Len Meakin	Improved installation script.
1.1	09/10/2008	Dave Bush	Addition of Firmware Loader
2.0	24/02/2011	Dave Bush	Update for Source Distribution
2.1	24/05/2016	Dave Bush	Update for USB under Intel
2.2	01/04/2022	Dave Bush	Minor general updates
2.3	04/12/2024	Dave Bush	Fedora and Suse specifics added

**Table of Contents**

---

**1 Distribution ..... 3**

1.1 The Basic Package ..... 3

1.2 USB Peripheral support ..... 3

1.3 Compilation ..... 4

1.3.1 Ubuntu / Mint..... 4

1.3.2 Fedora..... 4

1.3.3 Suse..... 4

1.4 Automated Installation..... 5

1.4.1 MEI BNR ..... 5

1.5 /usr/local/bin ..... 5

1.6 64 bit systems ..... 6

1.6.1 Paylink USB System ..... 6

1.6.2 Libusb ..... 6

1.6.3 MEI BNR ..... 6

**2 Firmware Updating..... 7**

**3 AESCDriver / Paylink ..... 8**

**4 Examples..... 9**

4.1 Coin Read ..... 9

4.2 Pay Out ..... 9

4.3 Lumina Serial Number ..... 9

4.4 AESDemo..... 10

# 1 Distribution

---

## 1.1 *The Basic Package*

The main Linux support software for Paylink is distributed as a Linux archive (PaylinkSourceDist.tar.gz) containing the source for all the PC support components. This has the advantage (over a binary distribution) that as it is totally source files, and is compiled on the user's system; it will run on any release of Linux. All the software is guaranteed compatible with the OS, and it will even run on non-Intel platforms.

## 1.2 *USB Peripheral support*

Part of the Paylink product is the support of USB connected peripherals. Here the main Paylink code base is running on the PC, and requires external verification (e.g. Dongle) to operate. This means that the main code base can not be distributed as source as the relevant source lines could just be commented out.

Instead the key files are distributed as pre-compiled Intel 32 bit object files. They can then be linked together with with the libraries installed on your system, a process that avoids any shared object mismatches.

This component of the Paylink system is distributed as a 2<sup>nd</sup> Linux archive (PaylinkCodeDist.tar.gz) which *overlays* the files and folders that are in the original source distribution.

## 1.3 Compilation

You will have to ensure that the standard development system is installed, Suitable standard commands for modern distributions follow, which should be run as root, using either su, or sudo.

### 1.3.1 Ubuntu / Mint

Firstly, to ensure that that the system is ready, it is worth running:

```
sudo apt-get update
```

Then if you don't have development tools already available, the standard way to install the tools needed is:

```
sudo apt-get install build-essential
```

If you are going to use the Paylink USB system (only available on Intel platforms) with a 64 bit distribution installed, you will also need the 32 bit libraries & headers, which will probably **not** be already available(see later.). The standard command for this is:

```
sudo apt-get install g++-multilib libc6-dev-i386
```

### 1.3.2 Fedora

Fedora distributions use a different set of commands. To get the development tools you need:

```
sudo dnf install make automake gcc gcc-c++
```

Again, if you are going to use the Paylink USB system you will need:

```
sudo dnf install glibc-devel.i686 libstdc++-devel.i686
```

### 1.3.3 Suse

Again Suse distributions use a different set of commands. To get the full development tools you need:

```
sudo zypper install -t pattern devel_basis
```

The minimum for Paylink is just the 'C++' system, which you can get with

```
sudo zypper install gcc-c++ make
```

Again, if you are going to use the Paylink USB system you will need:

```
sudo zypper install gcc-c++-32bit
```

## 1.4 Automated Installation

1. This package installs into the `/usr/local/bin` and `/usr/local/lib` directories. The programs are set as owned by root with the setuid bit as access to USB peripherals is not generally available in user mode.  
Before starting, you should ensure that `/usr/local/bin` is on your standard path, by typing  
`echo $PATH`  
If it is not, see below
2. Unpack the distribution directory if necessary to get a local folder. e.g.  
`tar -zxf PaylinkSourceDist.tar.gz`
3. If you are intending to use USB peripherals, you then also need to unpack the code archives *into the same directory*. This *overlays* the main distribution directory and replaces some of the files. e.g.  
`tar -zxf PaylinkCodeDist.tar.gz`
4. Swap to the newly created directory and issue the command  
`sudo ./Install.sh`  
This script should proceed to install the `AESCDriver` Paylink driver program and a number of support programs as well as `libusb-1.0` if required (the `libusb-1.0.20` package is included in Paylink). If you do not have `libusb` installed, the script will automatically install it. If it is already present, the script will offer to rebuild it from the source package included in the Paylink distribution.
5. If you are intending to use USB peripherals, you should then issue the command  
`sudo ./BuildPaylink.sh`  
This script should proceed to compile and link `Paylink`, the driver program required for USB connected peripherals (including Paylink Lite 2).

### 1.4.1 MEI BNR

If you intend to use the MEIBNR recycler, you *first* need to ensure that you have the BNR support libraries from Crane installed (see Crane support for details).

After that, issuing the command

```
sudo ./BuildPaylinkBNR.sh
```

will build the Driver program `PaylinkBNR` which will provide full BNR support. If you have not already done so, the scripts will prompt you to rename the appropriate interface object file in folder `Paylink/DriverDLLs` from `MEIBNR.odll.a.b.c` (where a.b.c corresponds to the BNR support package version that you have installed) to be just `MEIBNR.odll`.

## 1.5 /usr/local/bin

If this is not correctly set up, then one way of fixing this is to add the lines

```
# set up local paths for packages
pathmunge /usr/local/bin after
towards the end of /etc/profile
```

If you had to do this, then there is a good chance that your system is not set up for `/usr/local/lib`. To achieve this, add the line

```
/usr/local/lib
```

to the file `/etc/ld.so.conf`,  
(or create a file containing that line as `/etc/ld.so.conf.d/libc.conf`)  
and then update the cache by running  
`ldconfig`

## 1.6 64 bit systems

The basic Paylink package will generate 64 bit (native) programs and access object from the source files.

If you wish to support 32 bit applications on a 64 bit platform, you will need to generate a 32 bit access shared object. This can be done by including the parameter `multi` on the `./Install.sh` command i.e.

```
./Install.sh multi
```

This will generate two shared objects, `libaes_access_64.so` and `libaes_access_32.so`

### 1.6.1 Paylink USB System

Provided you have installed a suitable 32 bit development package, e.g.

```
apt-get install g++-multilib libc6-dev-i386
```

the `./BuildPaylink.sh` script will build a 32 version of `libusb` if necessary (see below) and a 32 bit Paylink driver program, you issue the command:

Although the Paylink driver program this creates is a 32 program, this will communicate without problems with the 64 bit `AESImhei.so` access object already created for use by your 64 applications.

### 1.6.2 Libusb

The Paylink system uses `libusb-1.0`, and the Paylink USB system will require a version compiled in 32 bit mode. This will probably not be installed, and so the `BuildPaylink.sh` script will install it in `/usr/local/lib/i386/lib`.

In order to enable programs to find this package, the `/etc/ld.so.cache` needs updating. On modern system this is created from files in `/etc/ld.so.conf.d` - to this end the script creates a file in this directory and runs `ldconfig`. (The real story is slightly more complicated - read the script if you're interested.)

### 1.6.3 MEI BNR

Due to a complex bug in the MEIBNR support code, the MEI code cannot always find the `libusb-1.0` object. When this happens the Paylink driver detects this, and outputs a specific error message, explaining how to use the `LD_LIBRARY_PATH` variable to work around it.

A sample usage of this is included in the package as the script `RunPaylinkBNR.sh`.

## 2 Firmware Updating

6. As well as the main release archive, there is a Firmware archive available - PaylinkFirmware.tar.gz which you need to unpack: e.g.

```
tar -zxvf PaylinkFirmwareV4-1-12-13.tar.gz
```

The main distribution has already compiled **USBProgram**, which performs the download itself, the firmware can therefore be distributed as text files, and used without requiring a compilation system on the target.

This Firmware archive contains a number of self contained scripts, each of which will load a new copy of the firmware into Paylink.

The naming conventions used in this folder are that:

- the first part is "Genoa", "Innov", "InnEbd" or "Mcd", the internal name of the firmware. (As the number of peripherals supported by Paylink has grown, they no longer all fit into the unit. See the "Supported Peripherals" section in the " MilanPaylinkSystemManual"
- the final part is **v***n*-*n*-*n*-*n* - the release number of the firmware.
- and there is then a suffix of .sh

This firmware updating program, when run, will automatically disable the Driver program (if running) and then download the new firmware to the Paylink. If there is any error in the programming, the program exits with a code of to exit with 1 for an error and 0 is everything is OK.

Options available for the download are as follows:

- Serial Number Specific mode (-s <Serial>): will cause the program to only update a Paylink set with the specific USB serial number (for us on a multiple Paylink installation.)
- Force mode (-f): regardless of the version already programmed, this mode will reprogram the Paylink.
- Check mode (-c): If this is set, then the programming will not happen if the Paylink already contains code with the same version number and program checksum.

### 3 AESCDriver / Paylink

---

The three interchangeable driver programs, **AESCDriver**, **Paylink** or **PaylinkBNR**, each run in command line mode, no graphical user interface has been developed. However the driver is able to run in various modes:

Verbose mode (-v): certain information is printed to stdout (console window). This is analogical to running the Windows driver with the hide traffic option selected. Without the verbose mode flag set the driver will print very minimal information (if any).

ShowTraffic mode (-t): will display all data that is being sent / received to / from the PayLink interface (all messages are time stamped to millisecond resolution). This is analogical to running the Windows driver with the show traffic option selected. This option is only usually useful when no traffic at all appears to be taking place.

High Priority mode (-p): will cause the driver program to increase the priority of the communication process.

Serial Number Specific mode (-s <Serial>): will cause the driver program to only communicate with a Paylink set with the specific USB serial number (for us on a multiple Paylink installation.)

The ShowTraffic option may be set at driver start-up (-s) or by sending a `SIGUSR1` signal to the driver. The driver will toggle the value of the ShowTraffic option upon receiving a `SIGUSR1` signal. The supplied `showtraf.sh` shell script will perform this action for you.

The shared memory segment has been named `AES` (or `AES<Serial>` if the serial number option is being used), and can be viewed as part of the file system at `/dev/shm/AES` (accessing this is not recommended and should only be used for diagnosing driver issues).

To force the driver to exit either send the driver a signal (other than `SIGUSR1`), or press `CTRL+C` on the console where the driver process was started.

Upon the driver starting it will attempt to open a link to the attached interface, if unsuccessful the driver will continue connection attempts every second until the device has been opened successfully.

## 4 Examples

---

A number of example programs are included in the distribution that can be used / compiled "straight out of the box" to show that the installation is working.

### 4.1 *Coin Read*

An example CoinRead program has been provided, to compile this program issue the following commands:

```
# cd PaylinkSourceDist/Paylink/CoinRead
# make
# ./CoinRead
```

### 4.2 *Pay Out*

An example PayOut program has been provided, to compile this program issue the following commands:

```
# cd PaylinkSourceDist/Paylink/PayOut
# make
# ./PayOut
```

### 4.3 *Lumina Serial Number*

The Lumina Serial Number program has been provided, to compile this program issue the following commands:

```
# cd PaylinkSourceDist/Paylink/LuminaSerialNo
# make
# ./LuminaSerial
```

## 4.4 AESDemo

The AES Demo application has been provided, to compile this program issue the following commands:

```
# cd PaylinkSourceDist/Paylink
# tar -zxvf AESDemo.tar.gz
# cd LinuxDemo
# ./configure
# make
# cd ./src
# ./AESDemo
```

Please ensure that X Windows is running before issuing the above commands.

This application uses the old gtk+ version 2 graphical tool kit library, if required you may have to install this with e.g.:

```
# sudo apt-get install pkg-config gtk+-2.0 libgnomeui-dev
```

It is entirely possible that the package will fail to compile on your system, I'm afraid you are on your own if so. The application has been tested with Ubuntu and Debian *Stretch* 64 bit distributions.

If the package does compile, it provides a sub set of the functionality from the Windows Demo program, the main screens are:



