

# Milan / Paylink Firmware Version 1.1.11.5 Release Notice.

This is a **Engineering** (1) release of the Milan / Paylink Interface firmware - code version **1.11.5**. There were earlier releases under the designation 1.11.x, but that development stream was abandoned, this release is an update of 4.1.10.9 the final release of the 1.10.x development stream.

This release note describes all the significant changes since then to the firmware and associated PC support files

## **Main New Features**

The main features of this release as compared with 4.1.10.x are:

- The API and Paylink now return additional information:
  - Details associated with each coin
  - A String containing all the release / version information for each unit.
  - The serial number of each unit (where available.)
- An application level watchdog, that enables an application to:
  1. Check that the Paylink and all its interconnections are still operating.
  2. Shut down acceptance rapidly in the case of application or interconnection failure.

## **PC code versions.**

To obtain all of the new facilities described in this release document the following PC versions are required:

Aesimhei.dll	Version 1.4.0.0 or later
AESWDriver.exe	Version 1.1.3.2 or later
AESW98Driver.exe	Version 1.1.2.2 or later
Aesimhei.h	Dated 29/10/07 or later

## **Significant Side effects from version 1.10.9**

None

## **Compatibility with 1.10.9**

Programs compiled with this release of Aesimhei.h will not run with older DLLs.

The Driver program is unchanged.

The DLL will run with older firmware without problem (although the new facilities will not be available)

The DLL and firmware will operate without problem with applications compiled with the old Aesimhei.h.

## **Upgrade / Downgrades**

Any earlier version of the firmware can be upgraded to this version without any problems.

Downgrading to 1.10.4 / 6 / 7 / 9 or 1.9.x will not cause any problem

## **New Features in Detail:**

### **Token Handling (Coin Ids) (1.11.x)**

As tokens do not have a known value, they appear as coins with value zero. The only way for a game to detect tokens is to use the **CurrentUpdates()** function to detect activity, and then to check for increases in the count of the token(s) to be accepted(**Coin.Count**).

The index for the coin that holds the count for a particular token can be obtained by searching the coin array belonging to the acceptor and comparing the coin name (**Coin.CoinName**) with that of the token.

### **Dual Currency Handling (Coin Ids) (1.11.x)**

If an acceptor is being used to accept coins of more than one currency, the application can determine the currency of a specific coin by examining the first two characters of the name of the coin (**Coin.CoinName**). For supported acceptors, the firmware guarantees that a coin name will always contain a currency code as the first two characters of a coin name.

ccTalk	This contains up to eight characters as returned by the Request Coin Id (184) command.
ID-003	This contains a representation of the three bytes as return by the Get Currency Assignment (0x8A) command. The first two bytes are the hex value for country code, then a '/', then the base value as a decimal number, followed by a '^', then the count of extra zeros as a decimal number.
MDB	TBD
GPT	TBD
ARDAC	The Ardac protocol does not return any information about notes.

---

### **Read out of Acceptor Details (1.11.x)**

Different protocols / manufacturers provide different details on acceptors. The **Acceptor.Description** field is generated as follows:

ccTalk	The replies to: <ul style="list-style-type: none"><li>Request Currency Specification ID (91),</li><li>Request Currency Revision (145),</li><li>Request Software Revision (241) &amp;</li><li>Request Product Code (244) commands, separated by '~' characters.</li></ul> Each individual field is truncated to 15 characters, and is omitted if there is no response to the command, although the '~' character is still inserted.
ID-003	The entire reply to the "Get Version Request" (0x88) command
MDB	TBD
GPT	TBD
ARDAC	TBD.

The **Acceptor.SerialNumber** field is generated as follows:

ccTalk	The binary reply to the ID Serial No (242) command.
ID-003	The "standard" ID-003 protocol does not allow for a serial number. A non-standard 0x8F query is issued and any response will be stored here.
MDB	TBD
GPT	TBD
ARDAC	TBD.

---

### **Read out of Dispenser Details (1.11.x)**

Different protocols / manufacturers provide different details on acceptors. The **Description (Dispenser.Description)** field is generated as follows:

ccTalk	The replies to: <ul style="list-style-type: none"><li>Request Software Revision (241) &amp;</li><li>Request Product Code (244) commands, separated by '~' characters.</li></ul> Each individual field is truncated to 15 characters, and is omitted if there is no response to the command, although the '~' character is still inserted.
MDB	TBD

The **Dispenser.SerialNumber** field is generated as follows:

ccTalk	The binary reply to the ID Serial No (242) command.
MDB	TBD

## ***CheckOperation (1.11.x)***

### **Synopsis**

This call allows an application to check that the Paylink and its connection to the PC are operational. It also allow the application to automatically close down currency acceptance in the event of any PC malfunction.

```
long CheckOperation(long Sequence,  
                    long Timeout)
```

### **Parameters**

1. Sequence  
A unique number for this call, freely chosen by the application.
2. Timeout  
A time in milliseconds before which another **CheckOperation()** call must be made, *with a different value in **Sequence***, in order to continue the normal operation of Paylink. If zero, then this functionality is inactive from then on.

### **Return Value**

The last **Sequence** value of which the Paylink unit has been notified, or -1 if the Paylink does not support this facility.

### **Remarks**

1. In normal operation, Paylink can be expected to have updated the value to be returned by this within 100 milliseconds of the previous call. It is suggested that this call is made every 500 milliseconds or longer to allow for transient delays.
2. If the **Timeout** expires, Paylink will “silently” disable all the acceptors that are connected to it. The next call to **CheckOperation()** will “silently” re-enable them. This facility is not operation until the first call of **CheckOperation()**.