

## Milan / Paylink Firmware Version 3.1.10.1 Release Notice.

This is a **Beta test** (3) release of the Milan / Paylink Interface firmware - code version **1.10.1**.  
*This has not been fully tested, but is believed to be stable enough to be used in a release to customers if the facilities it implements are essential.*

This release implements the MDB protocol and new hopper facilities.

This release is an upgrade to 1.9.x. All changes are referenced to the FULL (4) distribution of the 1.9.x firmware and associated PC support files.

The main features of this release as compared with 1.9.x are:

- Support for MDB coin-changer and bill acceptor devices
- Many dispenser improvements, including
  - Support for Hopper / Tube level monitoring
  - Support for Money Control Combi Hopper
  - Accurate monitoring and recovery of payout counts, including over resets.
  - Support of “unencrypted” cc-talk dispensers, using the SCH1 protocol
  - Auditing events where hopper level changes are unrelated to Payouts.

In order to provide the improved hopper facilities, the Dispenser detail block has been extended. This requires use of the updated `aesimhei.h` (`INTERFACE_VERSION == DISPENSER_UPDATE`) that was released as part of the Full 1.9.x release together with a DLL version 1.3.2.2 or later.

### **Significant Side effects from version 1.9.x**

The dispenser detail field **Count** was zero at unit reset in version 1.9.x. From version 1.10.1 onwards, this field is recovered from cctalk hoppers. *User applications may accidentally depend upon this reset.* For hoppers such as the SCH2 which guarantee the correctness of their dispensed coin counts, this field is equally guaranteed to be accurate, even if the system is reset during a payout. The **CurrentPaid()** function is similarly guaranteed to be accurate.

### **Upgrade / Downgrades**

Any earlier version of the firmware can be upgraded to this version without any problems.  
Due to residual data in the E<sup>2</sup>Prom, downgrading from this version to other versions may or may not work; the only guaranteed downgrade path is to version x.1.9.6.

### **MDB changer support.**

The MDB protocol is now implemented on the MDB slave connection which is a **three** pin connector on the Paylink (next but one to the meter connector) or the **three** pin connector on the MDB daughterboard for the Milan card. (The *four* pin connector connects to an MDB master for peripheral applications.)

If a changer is found it will appear as an acceptor in very much the same way as any other acceptor. The coins that are routed to tubes can be distinguished as having a non zero Routed Path.

With the payout, the situation is slightly more complicated. The MDB changer protocol supports two different payout mechanisms, a basic one that is always present and an extended one, which is supported on some level 3 changers. The basic provides control over the individual payout tube, but has no feedback as to whether the payout works. The extended one provides feedback as to the success of the payout, but does not allow any control over which tubes the payout is from.

The solution adopted is to always provide one dispenser for each tube, which is run using the basic mechanism and if the extended mechanism is present to provide an additional dispenser which is run using the extended mechanism. Where an extended mechanism dispenser is available, the individual tubes are pre-set to inhibited.

To perform a “normal” payout, you just issue a **PayOut()** request and call **PayStatus()** and **CurrentPaid()** to monitor the results. If you have a level 2 changer, **CurrentPaid()** will update almost instantaneously rather than at the end and will always show that all coins have been paid. If you have a

level 3 changer, **CurrentPaid()** will update during the process, and you may get a PAY\_EMPTY status, with **CurrentPaid()** reflecting the actual payout achieved.

Should you wish to perform an operation on a specific tube (e.g. emptying it), you should inhibit the extended mechanism dispenser and enable the specific tube you wish to control.

As the manufacturer is shown in the acceptor detail block for the changer, the extended mechanism dispenser has the constant type DP\_MDB\_TYPE\_3\_PAYOUT while the individual tubes have the type DP\_MDB\_LEVEL\_2\_TUBE .

The current levels of MDB tubes, *as reported by the coin-changer*, are returned in the new field **CoinCount**. In addition, the field **CoinCountStatus** will contain the value DISPENSER\_ACCURATE for a normal tube, and DISPENSER\_ACCURATE\_FULL if the changer is reporting the tube as full. Note that the levels reported by the changer do not necessarily update in a “sensible” fashion after a payout.

### ***ccTalk Hopper Support.***

The new CoinCountStatus will be filled in if the hopper supports the appropriate cctalk command and the sensors are fitted. Possible returned statuses are:

DISPENSER_COIN_NONE	No sensor fitted / level command not supported / old firmware
DISPENSER_COIN_LOW	Less than the low sensor level
DISPENSER_COIN_MID	Above low sensor but below high
DISPENSER_COIN_HIGH	High sensor level reported

### ***Combi Hopper Support.***

This single unit dispenses two different coin values. It is therefore handled in a similar way to the MDB system. There is a primary dispenser, which is set up as a normal unit with a **Unit** field of DP\_MCL\_SCH3, and a **Value** field with the lower coin value in it. The **Count** in this dispenser is the count of the lower value coins dispensed. In addition, another dispenser is set up, with a matching **Address** field, a **Unit** field of DP\_CC\_GHOST\_HOPPER, the **Value** of the higher coin and the **Count** of the higher value coins dispensed.

Note that, due to limitations of the unit, during a payout operation the **Count** of the main dispenser only is updated as though all coins dispensed were of this value. At the end of the sequence, while **LastPayStatus()** is still returning PAY\_ONGOING, the accurate count of both coins is retrieved and the two **Count** fields updates. This means that, as the operation finishes, the lower value **Count** decrements.