
Bill Recycler Operation (1.12.3)

Introduction

The original Paylink model was based around the concept of independent acceptors and dispensers, with the main specification for a dispenser being the MCL / cctalk hopper.

The advent of bill / note recyclers has meant that the Paylink model has had to be enhanced to include these. The approach adopted has been based around the idea that bill / note recycler (and the coin changer MDB device) is a combination in a single unit of an acceptor and number of dispensers.

This section describes the evolved handling of recyclers (and bill dispensers) that is in place in versions 1.12.3 and later.

Security

The connections used by Paylink fall into three categories, cctalk, MDB and RS232. These communications systems tend to match up with three different markets:

- RS232 is used with a number of protocols to connect expensive bill acceptors / recyclers for applications that are typically in expensive, secure enclosures. Any interference with this connection will generally tend to be visible to the Paylink application.
- MDB is used in very cheap systems, typically vending machines. The connection is vulnerable to interference, but the amounts of money involved tend to be low.
- cctalk is a very versatile system, which can be used in high value applications. The connection is relatively vulnerable to interference and so encryption is used to provide security. The latest security option for cctalk is **DES** encryption, which involves the exchange of local, random keys.

Paylink has, as a philosophy, the idea that it will connect to anything. Using **DES** encryption for a bill and a coin acceptor do not therefore make sense and people wishing to create a fully secure system using these are referred to the DES Paylink system (which is closed down and secure)

A **DES** bill recycler however makes sense, as the locked down aspect here is in the peripheral - Paylink can connect to any recycler, but the recycler will only communicate with Paylink. Given the vulnerability to high value fraud of a cctalk recycler, Paylink therefore *insists* that all new cctalk recyclers supported must use **DES** encryption.

As a special case the Merkur recycler is supported - this is not a general purpose protocol and Merkur recyclers are not expected to be used in vulnerable systems.

Component Identity

The general approach to identifying these devices is that the acceptor part usually contains the overall description of the unit, and the dispensers are (relatively arbitrarily) identified by a sequence number (from 1 upwards) and the value that they dispense.

Where necessary a dispenser can be tied to acceptor by type and by having the same serial number. The `DispenserBlock.m_UnitAddress` field(s) will contain a "sequence" number that identifies the dispenser. If the unit only has one dispenser, the field will contain 1.

On multi-drop system, such as cctalk and MDB, the address of the parent acceptor will be OR'ed with this sequence number so that multiple units can be distinguished.

Routing

Dispenser Destination

Paylink, during its initialisation of the unit, determines the value of the coin / bill in the dispenser(s) and which coins / bills are routed into which dispensers. The “sequence” number(s) are stored into the `AcceptorCoin.Path` field(s). All the other `AcceptorCoin.Path` fields will be zero.

When a coin / bill is accepted and routed into a dispenser this fact is always identified by Paylink and the `AcceptorCoin.PathCount` is accurately incremented to show this.

Paylink then updates the `DispenserBlock.CoinCount` field by actually querying the unit. Depending upon the actual unit this will be either accurate or an approximation. With a bill recycler the result is usually an accurate figure, with an MDB changer the result is often approximate.

The value returned will however *always* be that reported by the device, any systematic corrections will have to be handled by the application.

Routing Control.

For some bill recycler units, such as the Merkur MD100 and MDB Changers, the routing is fixed and it is not possible for Paylink, and hence the application, to change this.

For other units, the routing can be changed by Paylink. The application notifies Paylink of the desired routing by changing the `Path` fields of the incoming `Coin (Bill)` array item to contain the associated dispenser serial number.

Options available are:

- If the `Path` field for a currently recycled bill is set to zero, the unit will stop diverting bills into the recycler. If there are no bills stored, then the `Dispenser` value will go to 999999999, (this is irrelevant to payouts, as the dispenser will return “empty” if it is attempted to be used), if bills are currently stored they may remain available to be paid out (depending upon the device capabilities)
- The application should not set two or more separate bills to contain the same value in the `Path` field, if this is done, then this situation is undefined
- For all coin / bills with a non-zero path number, if the path field for a bill corresponds to the sequence number for a dispenser, then Paylink will update the recycler to direct that bill into the corresponding dispenser. As a part of this process, many units will cause any bills already in the dispenser will be stored into the cashbox.

Note that as the specification is “where to send the bill” it is not possible to have two dispensers regarded as having the same value bill.

The options for automatic re-routing using `DefaultPath` and `PathSwitchLevel` are only available with coin acceptors. For note recyclers, these fields are not used.

Dispenser Emptying

The high value represented by the bills in recyclers means that the dispensing of bills requires the interaction of the recipient. The high value of the bills stored in the recycler also means that users are liable to want to empty them at the end of the day.

These two factors mean that bill recycler manufacturers implement a “dump to cash box” facility so that the bills can easily be retrieved.

Full Dump

A full dump is where the recycler takes every bill from a dispenser into the cash box until the dispenser registers as empty.

Triggering this is implemented on Paylink by the user setting a `DispenserBlock.Status` value of `DISPENSER_CASHBOX_DUMP`.

On recyclers that maintain guaranteed accurate counts of bill, the application can monitor the dump process by observing the `DispenserBlock.CoinCount` going to zero.

On both these and other recyclers, the application can check for the `DISPENSER_CASHBOX_DUMP` value being replaced by another status. Where the dump process completes normally, the status will take value of `DISPENSER_DUMP_FINISHED`.

Partial Dump

As well as the above facility to cycle every bill into the cashbox, many recyclers provide the ability to perform a partial cashbox dump processes, which can be used to leave a “float” of bills in the recycler.

Triggering this is implemented on Paylink by the user setting the count of bills that are to be dumped in the new `DispenserBlock.NotesToDump` field and a value of `DISPENSER_PARTIAL_DUMP` in the `DispenserBlock.Status` field.

The application can usually monitor the dump process by observing the `DispenserBlock.CoinCount` field reducing by the requested amount.

The application can check for the `DISPENSER_PARTIAL_DUMP` value being replaced by another status. Where the dump process completes normally, the status will take value of `DISPENSER_DUMP_FINISHED`.

Payout Progress

Cancelling Payout

Bill recyclers / dispensers in general hold the bills awaiting collection by the user, and Paylink does not regard the Payout as complete until the bill has actually been taken. If the application program decided that the bill has been forgotten, it can abandon the payout by setting the inhibit flag on the dispenser. This will cause Paylink to request that the recycler abandons the payout and returns the bill to the cash box.

Note that following the abandonment of the bill payout Paylink will automatically proceed to attempt payout in coins, so in the usual case all the coin dispensers should be disabled at the same time as the bill recycler.

Notification of progress

While a bill recycler / dispenser is holding a bill awaiting collection by the user, Paylink does not regard the Payout as complete. The fact that the bill is available to be taken is however possibly of significance to the application, and therefore Paylink will update the `DispenserBlock.Count` and `DispenserBlock.CoinCount` fields for the relevant dispenser as soon as the bill is accessible. They will not then change when it is taken.

Power Fail

Temporary power interruption

Should the power / communications to the recycler fail during a payout while Paylink continues to run, Paylink will initially just wait for the communications to restart, and will then continue as though there has been no interruptions.

Where an interruption lasts “a long time” then Paylink will abandon the payout attempt. This will result in a dispenser / payout status of `PAY_US`. If at the time the payout is abandoned, Paylink is aware of a bill awaiting collection by the user, it will be regarded as having been paid out. It will not therefore be substituted by coins and can result in a normal payout completion status.

After the timeout, if / when normal communication with the recycler is resumed, Paylink will check the current status of the unit.

- A bill that was paid and collected during the interruption will cause the `DispenserBlock.Count` field to be incremented by the appropriate amount and a `IMHEI_NOTE_DISPENSER_UPDATE` event entered in the `NextEvent()` queue.

- A bill that has been sent from a dispenser to the cash box (as a part of the start up recovery process) will merely result in the `DispenserBlock.CoinCount` being updated.
- A bill that was awaiting collection, and has still not been taken, will cause the dispenser status to change to `PAYOUT_ONGOING` until it is eventually collected. This will have no effect on the `DispenserBlock.Count` field or Payout system.
- A bill that was awaiting collection but is ***now*** known to have been automatically recycled to the cash box, will cause the `DispenserBlock.Count` field to be decremented (to “undo” the payout) and a `IMHEI_NOTE_DISPENSER_UPDATE` event entered in the `NextEvent()` queue.

Full power Failure

Should the power to PC and recycler fail during a payout the application may wish to reconstruct the partial results of the last payout attempt. To facilitate this, Paylink will attempt to handle the interrupted payout according to the above rules

To do this requires that the `DispenserBlock.Count` field be maintained over power cycles - thus applications that so desire can record the `Count` fields before a payout is started, and then react accordingly if on startup they discover that a payout was in progress.

With coin hoppers, the speed of the payout means that the only place where an accurate record of interrupted payouts can possibly be obtained is in the hopper itself. Note dispensers typically do not provide such facilities, and Paylink therefore maintains the record itself.

The net result is that lifetime totals are maintained and reported in the `Count` fields, which are retrieved and updated by Paylink depending on the data read from the device during startup.

If this startup processing causes Paylink to suspected an uncompleted payout actually completed, an `IMHEI_COIN_DISPENSER_UPDATE` event is entered in the `NextEvent()` queue.

Unpaid Bills

Some devices (e.g. the B2B bill recycler and F56 bill dispenser) have a delivery stage where bills are accumulated for eventual payout.

Following a power failure, it can happen that bills are in this output stage and are inaccessible to the user. The only thing that Paylink can do at this point is to complete the delivery of these bills, but as there is a potentially long time since the application requested the dispense, it is inappropriate to just deliver them at power up.

(Some F56 / F53 models also have a problem that following a power failure during a dispense there can be an unknown number of bills awaiting delivery.)

In these cases, for each dispenser device that is believed to have notes ready for delivery, Paylink marks the device inhibited, and generates an `IMHEI_NOTE_DISPENSER_PENDING` event with the number of bills as the `RawEvent` field. If the number of bills is unknown then 99 is used.

To complete the delivery process, the application should clear the inhibit on ***all*** the relevant dispensers.

Device Specific Functionality

The above description is the ideal that Paylink strives to achieve. The actual functionality provided by specific devices can however interfere with this, so all supported models of note / bill recycler are itemised here:

Cashcode B2B

The Cashcode B2B accumulates bills to be dispensed in separate unit before presenting them to the user. When bills are “found” in the dispenser during startup, the only thing the unit *can* do is to dispense them.

When Paylink discovers this situation, during startup, or following a “long” power fail, it will undertake **Unpaid Bill** processing as above.

Merkur 100

This recycler automatically restarts a payout on power up, unless a software reset is issued before the acceptor reaches the point at which the delivery is under way.

During the startup process, Paylink issues such a reset, so if the two units power up approximately at the same time, no spurious bill is paid. If this succeeds, then any bills “in progress” will be returned to the stacker.

JCM UBA Recycler

This unit can have bills “manually” loaded into the storage stackers. When this occurs, the UBA unit doesn’t know about the event and so does not update its internal counts, these are only updated when bills that have been accepted are stacked.

Similarly, if bills that have been stacked automatically are manually taken, the counts are not reduced.

The counts returned by Paylink are those from the UBA unit, and so under these circumstances will be incorrect - it is up to the application to compensate for those bills it knows have been manually inserted.

When the counter of the number of bills in a stacker reaches zero Paylink will still attempt to pay bills - if this succeeds the UBA counter will remain at zero - it will not go negative.

Similarly, if the stacker runs out of bills when the UBA counter is non-zero, the UBA will zeroize the counter.

JCM Vega & JCM UBA Recycler

These recyclers can retrieve a bill waiting for collection, and send it to the cash box..

If the dispenser is inhibited during a payout then, as well as preventing further payouts, Paylink will actually retrieve the bill waiting for collection. As this occurs after the bill has been accounted for, both the `DispenserBlock.Count` field and the `CurrentPaid()` return value will decrement.

This is command also used in those recovery situations where the application has not been informed that the bill has been dispensed.

This can help avoid the situation where payouts that were uncompleted actually occur.

F56 / F53 Bill Dispenser

The F56 device comes with a number of different options. Although the F53 is a different device number, Paylink just regards it as another option of an F56. All descriptions are therefore of an F56.

These F56 have a number of unique characteristics:

- The F56 only reports cassettes that are present, the existence of a location for a cassette is not discoverable. Paylink therefore only reports the status of cassette locations in which it has seen a cassette.
- A cassette can have a pattern of magnets set into it to indicate the type of bills with which it is loaded. The F56 configuration can include bill descriptions corresponding to these magnet patterns, which can specify value, bill length and bill thickness. If a newly discovered cassette matches such a pattern specification, then the bill value and sizes are set from the specification.
- If no magnet specification is given, or if there is no match, then the sizes default to a generic accept all size and the value is set as 999999999. This can be overridden to its correct value using the standard Paylink facilities.
- A pool area / note delivery option is possible, with delivery to the front or to the rear. Part of the configuration specification of an F56 has to include whether or not a delivery option is fitted.
- The F56 records in non-volatile memory the number bills delivered from a payout position. This value is reported to the application in the `DispenserBlock.Count` field.

- Some F56 models allow for the recovery of a failed dispense operation - on others this information is not available. Where this information is not available and the unit has a final dispense stage, then the notes are left in the pool area and **Unpaid Bill** processing performed.