# INNOVATIVE TECHNOLOGY LTD

## Protocol Manual

## SSP

SMART HOPPER, SMART SYSTEM

version GA138_2_2_223A

# Contents

<< back to index

## Introduction

This manual describes the operation of the Smiley ® Secure Protocol **SSP**.

ITL recommend that you study this manual as there are many new features permitting new uses and more secure applications.
If you do not understand any part of this manual please contact the ITL for assistance. In this way we may continue to improve our product.
Alternatively visit our web site at www.innovative-technology.co.uk

Enhancements of SSP can be requested by contacting:
support@innovative-technology.co.uk

***MAIN HEADQUARTERS***
Innovative Technology Ltd
Derker Street, Oldham, England. OL1 4EQ

Tel: +44 161 626 9999 Fax: +44 161 620 2090
E-mail: support@innovative-technology.co.uk

Web site: www.innovative-technology.co.uk



***Innovative Technology is not responsible for any loss, harm, or damage caused by the installation and use of this product.***

***This does not affect your local statutory rights.***

***If in doubt please contact innovative technology for details of any changes.***

<< back to index

## General Description

Smiley ® Secure Protocol (SSP) is a secure interface specifically designed by ITL ® to address the problems experienced by cash handling systems in gaming machines.Problems such as acceptor swapping, reprogramming acceptors and line tapping areall addressed.

The interface uses a master-slave model, the host machine is the master and the peripherals (note acceptor, coin acceptor or coin hopper) are the slaves.

Data transfer is over a multi-drop bus using clock asynchronous serial transmissionwith simple open collector drivers. The integrity of data transfers is ensured through the use of 16 bit CRC checksums on all packets.

Each SSP device of a particular type has a unique serial number; this number is used to validate each device in the direction of credit transfer before transactions can takeplace. It is recommended that the encryption system be used to prevent fraud through busmonitoring and tapping. This is compulsory for all payout devices.

Commands are currently provided for coin acceptors, note acceptors and coinhoppers. All current features of these devices are supported.

**FEATURES:**

- Serial control of Note / Coin Validators and Hoppers
- 4 wire (Tx, Rx, +V, Gnd) system
- Open collector driver, similar to RS232
- High Speed 9600 Baud Rate
- 16 bit CRC error checking
- Data Transfer Mode
- Encryption key negotiation
- 128 Bit AES Encrypted Mode

**BENEFITS:**

- Proven in the field
- Simple and low cost interfacing of transaction peripherals.
- High security control of payout peripherals.
- Defence against surrogate validator fraud.
- Straightforward integration into host machines.
- Remote programming of transaction peripherals
- Open standard for universal use.

To help in the software implementation of the SSP, ITL can provide, C/C++ Code, C#.Net Code, DLL controls available on request. Please contact: support@innovative-technology.co.uk

<< back to index

## Hardware layer

Communication is by character transmission based on standard 8-bit asynchronous data transfer.

Only four wires are required TxD, RxD, +V and ground. The transmit line of the host is open collector, the receive line of each peripheral has a 10Kohm pull-up to 5 volts.The transmit output of each slave is open collector, the receive input of the host has asingle 3k3 ohm pull-up to 5 volts.

The data format is as follows:

| | |
|---|---|
| Encoding | **NRZ** |
| Baud Rate | **9600** |
| Duplex | **Full** |
| Start bits | **1** |
| Data Bits | **8** |
| Parity | **none** |
| Stop bits | **2** |

**Caution**: **Power to peripheral devices would normally be via the serial bus. However devices that require a high current supply in excess of 1.5 Amps, e.g. hoppers, would be expected to be supplied via a separate connector.**

<< back to index

## Transport Layer

Data and commands are transported between the host and the slave(s) using a packet format as shown below:

| STX | SEQ/SLAVE ID | LENGTH | DATA | CRCL | CRCH |
|---|---|---|---|---|---|

| | |
|---|---|
| STX | Single byte indicating the start of a message - 0x7F hex |
| SEQ/ Slave ID | Bit 7 is the sequence flag of the packet, bits 6-0 represent the address of the slave the packet is intended for, the highest allowable slave ID is 0x7D |
| LENGTH | The length of the data included in the packet - this does not include STX, the CRC or the slave ID |
| DATA | Commands and data to be transferred |
| CRCL, CRCH | Low and high byte of a forward CRC-16 algorithm using the Polynomial (X16 + X15 + X2 +1) calculated on all bytes, except STX. It is initialised using the seed 0xFFFF. The CRC is<br>calculated before byte stuffing. |

**PACKET SEQUENCING**

Byte stuffing is used to encode any STX bytes that are included in the data to be transmitted. If 0x7F (STX) appears in the data to be transmitted then it should be replaced by 0x7F, 0x7F.
Byte stuffing is done after the CRC is calculated, the CRC its self can be byte stuffed. The maximum length of data is 0xFF bytes.
The sequence flag is used to allow the slave to determine whether a packet is a re-transmission due to its last reply being lost. Each time the master sends a new packet to a slave it alternates the sequence flag. If a slave receives a packet with the same sequence flag as the last one, it does not execute the command but simply repeats it's last reply. In a reply packet the address and sequence flag match the command packet.
This ensures that no other slaves interpret the reply as a command and informs the master that the correct slave replied. After the master has sent a command to one of the slaves, it will wait for 1 second fora reply. After that, it will assume the slave did not receive the command intact so it will re-transmit it with the same sequence flag. The host should also record the fact that a gap in transmission has occurred and prepare to poll the slave for its serial number identity following the current message. In this way, the replacement of the hosts validator by a fraudulent unit can be detected.
The frequency of polling should be selected to minimise the possibility of swapping a validator between polls. If the slave has not received the original transmission, it will see the re-transmission as a new command so it will execute it and reply. If the slave had seen the original command but its reply had been corrupted then the slave will ignore the command but repeat its reply. After twenty retries, the master will assume that the slave has crashed. A slave has no time-out or retry limit. If it receives a lone sync byte part way through receiving a packet it will discard the packet received so far and treat the next byte as an address byte.

<< back to index

## Encryption Layer

**PACKET FORMAT**

Encryption is mandatory for all payout devices and optional for pay in devices. Encrypted data and commands are transported between the host and the slave(s) using the transport mechanism described above, the encrypted information is stored in the data field in the format shown below:

| STX | SEQ/SLAVE ID | LENGTH | **DATA** | CRCL | CRCH |
|---|---|---|---|---|---|

DATA

| STEX | Encrypted Data |
|---|---|

Encrypted Data

| eLENGTH | eCOUNT | eDATA | ePACKING | eCRCL | eCRCH |
|---|---|---|---|---|---|

| | |
|---|---|
| STEX | Single byte indicating the start of an encrypted data block - 0x7E |
| eLENGTH | The length of the data included in the packet - this does not include STEX, COUNT, the<br>packing or the CRC |
| eCOUNT | A four byte unsigned integer. This is a sequence count of encrypted packets, it is incremented each time a packet is encrypted and sent, and each time an encrypted packet<br>is received and decrypted. |
| eDATA | Commands or data to be transferred |
| ePACKING | Random data to make the length of the length +count + data + packing + CRCL + CRCH<br>to be a multiple of 16 bytes |
| eCRCL/eCRCH | Low and high byte of a forward CRC-16 algorithm using the polynomial (X16 + X15 + X2<br>+1) calculated on all bytes except STEX. It is initialised using the seed 0xFFFF |

After power up and reset the slave will stay disabled and will respond to all commands with the generic response KEY_NOT_SET (0xFA), without executing the command, until the key has been negotiated. There are two classes of command and response, general commands and commands involved in credit transfer.

General commands may be sent with or without using the encryption layer. The slave will reply using the same method, unless the response contains credit information, in this case the reply will always be encrypted. Credit
transfer commands, a hopper payout for example, will only be accepted by the slave if received encrypted. Commands that must be encrypted on an encryption-enabled product are indicated on the command descriptions for each command. The STEX byte is used to determine the packet type. Ideally all communications will be encrypted.

After the data has been decrypted the CRC algorithm is performed on all bytes including the CRC. The result of this calculation will be zero if the data has been decrypted with the correct key. If the result of this calculation is non-zero then the peripheral should assume that the host did not encrypt the data (transmission errors are detected by the transport layer). The slave should go out of service until it is
reset.

The packets are sequenced using the sequence count; this is reset to 0 after a power cycle and each time the encryption keys are successfully negotiated. The count is incremented by the host and slave each time they successfully encrypt and transmit a
packet. After a packet is successfully decrypted the COUNT in the packet should be compared with the internal COUNT, if they do not match then the packet is discarded.

<< back to index

## Encryption Keys

The encryption key length is 128 bits. However this is divided into two parts. The lower 64 bits are fixed and specified by the machine manufacturer, this allows the manufacturer control which devices are used in their machines.
The higher 64 bits are securely negotiated by the slave and host at power up, this ensures each machine and each session are using different keys. The key is negotiated by the Diffie-Hellman key exchange method.
See: en.wikipedia.org/wiki/Diffie-Hellman

The exchange method is summarised in the table below. C code for the exchange algorithm is available from ITL.

| Step | Host | Slave |
|---|---|---|
| 1 | Generate prime number GENERATOR | |
| 2 | Use command Set Generator to send to slave Check GENERATOR is prime and store | Check GENERATOR is prime and store |
| 3 | Generate prime number MODULUS | |
| 4 | Use command Set Modulus to send to slave Check MODULUS is prime and store | Check MODULUS is prime and store |
| 5 | Generate Random Number HOST_RND | |
| 6 | Calculate HostInterKey: = GENERATOR ^ HOST_RND mod MODULUS | |
| 7 | Use command Request Key Exchange to send to slave. | Generate Random Number SLAVE_RND |
| 8 | | Calculate SlaveInterKey: = GENERATOR ^ SLAVE_RND mod MODULUS |
| 9 | | Send to host as reply to Request Key Exchange |
| 10 | Calculate Key: = SlaveInterKey ^ HOST_RND mod MODULUS | Calculate Key: = HostInterKey ^ SLAVE_RND mod MODULUS |

Note: ^ represents to the power of

<< back to index

## Generic Commands and Responses

All devices must respond to a list of so-called Generic Commands as show in the table below.

| Command | Code |
|---|---|
| Reset | 0x01 |
| Host Protocol Version | 0x06 |
| Get Serial Number | 0x0C |
| Sync | 0x11 |
| Disable | 0x09 |
| Enable | 0x0A |
| Get Firmware Version | 0x20 |
| Get Dataset Version | 0x21 |

A device will respond to all commands with the first data byte as one of the Generic responses list below..

| Generic Response | Code | Description |
|---|---|---|
| OK | 0xF0 | Returned when a command from the host is understood and has been, or is in the process of, being executed. |
| COMMAND NOT KNOWN | 0xF2 | Returned when an invalid command is received by a peripheral. |
| WRONG No PARAMETERS | 0xF3 | A command was received by a peripheral, but an incorrect number of parameters were received. |
| PARAMETERS | 0xF4 | One of the parameters sent with a command is out of range. |
| COMMAND CANNOT BE PROCESSED | 0xF5 | A command sent could not be processed at that time. E.g. sending a dispense command before the last dispense operation has completed. |
| SOFTWARE ERROR | 0xF6 | Reported for errors in the execution of software e.g. Divide by zero. This may also be reported if there is a problem resulting from a failed remote firmware upgrade, in this case the firmware upgrade should be redone. |
| FAIL | 0xF8 | Command failure |
| KEY NOT SET | 0xFA | The slave is in encrypted communication mode but the encryption keys have not been negotiated. |

<< back to index

## Protocol Versions

An SSP Poll command returns a list of events and data that have occurred in the device since the last poll.

The host machine then reads this event list taking note of the data length (if any) of each event.

On order to introduce new events, SSP uses a system of **Protocol Version** levels to identify the event types and sizes a machine can expect to see in reponse to a poll. If this were not done, new unknown events with unknown datasize to a machine not set-up for these would cause the event reading to fail.

A host system should take note of the protocol version of the device connected and ensure that it is not set for a higer version that the one it is expecting to use.

The host can also check that the device can also be set to the higher protocol level, enusring that expected events will be seen.

The listed events in this manual show the protocol version level of each event.

As part of the start-up procedure, the host should read the current protocol level of the device (using the set-up request command).

<< back to index

## SMART Hopper

SMART Hopper is a coin payout device capable of discriminating and paying out multi-denominations of stored coins from its internal storage hopper.
Coins added to the hopper can be designated to be routed to an external cashbox on detection or recycled and stored in the hopper unit to be available for a requested payout.
SMART Hopper also supports the addition of a connected cctalk™ or eSSP™ coin mechanism which will automatically add its validated coins to the SMART Hopper system levels.

Note that payout values are in terms of the of the penny value of that currency. So for 5.00, the value sent and returned by the hopper would be 500. All transactions with a SMART hopper must be encrypted to prevent dispense commands being recorded and replayed by an external device.

Addressing

**The SMART Hopper has a default SSP Address of 16 dec 0x10 hex.**

The setup request reponse table for coin hopper types:

**Protocol version less than 6:**

| Data | byte offset | size (bytes) | notes |
|---|---|---|---|
| Unit type | 0 | 1 | 3 = SMART Hopper |
| Firmware version | 1 | 4 | ASCII data of device firmware version (e.g. '0110' = 1.10) |
| Country code | 5 | 3 | ASCII code of the device dataset (e.g. 'EUR') |
| Protocol Version | 8 | 1 | The current protocol version set for this device |
| Number of coin values | 9 | 1 | The number of coin denominations in this device dataset. [n] |
| Coin values | 10 | n * 2 | 2 byte each value for the coin denominations (e.g. 0.05 coin = 0x05,0x00) |

**Protocol version greater or equal to 6:**

| Data | byte offset | size (bytes) | notes |
|---|---|---|---|
| Unit type | 0 | 1 | 3 = SMART Hopper |
| Firmware version | 1 | 4 | ASCII data of device firmware version (e.g. '0110' = 1.10) |
| Country code | 5 | 3 | ASCII code of the device dataset (e.g. 'EUR') |
| Protocol Version | 8 | 1 | The current protocol version set for this device |
| Number of coin values | 9 | 1 | The number of coin denominations in this device dataset. [n] |
| Coin values | 10 | n * 2 | 2 byte each value for the coin denominations (e.g. 0.05 coin = 0x05,0x00) |
| Country codes | 10 + (n * 2) | | An obsolete value showing security level. This is set to 2 if the value multiplier is > 0 otherwise 0. |

<< back to index

## Smart System

The Smart System device is a multi-coin pay-in, pay-out system with detachable fast coin pay-in feeder.

Coins fed into the pay-in head will be validated and counted and recognised coins are routed to the attached hopper while rejected coins are fed out of the front of the system.

Coin hopper levels are adjusted internally.

The system can function as a stand-alone hopper payout system if the pay-in feeder head is removed.

**The SMART Systemhas a default SSP Address of 16 dec 0x10 hex**

The setup request reponse table for coin hopper types:

**Protocol version less than 6:**

| Data | byte offset | size (bytes) | notes |
|---|---|---|---|
| Unit type | 0 | 1 | 3 = SMART Hopper |
| Firmware version | 1 | 4 | ASCII data of device firmware version (e.g. '0110' = 1.10) |
| Country code | 5 | 3 | ASCII code of the device dataset (e.g. 'EUR') |
| Protocol Version | 8 | 1 | The current protocol version set for this device |
| Number of coin values | 9 | 1 | The number of coin denominations in this device dataset. [n] |
| Coin values | 10 | n * 2 | 2 byte each value for the coin denominations (e.g. 0.05 coin = 0x05,0x00) |

**Protocol version greater or equal to 6:**

| Data | byte offset | size (bytes) | notes |
|---|---|---|---|
| Unit type | 0 | 1 | 3 = SMART Hopper |
| Firmware version | 1 | 4 | ASCII data of device firmware version (e.g. '0110' = 1.10) |
| Country code | 5 | 3 | ASCII code of the device dataset (e.g. 'EUR') |
| Protocol Version | 8 | 1 | The current protocol version set for this device |
| Number of coin values | 9 | 1 | The number of coin denominations in this device dataset. [n] |
| Coin values | 10 | n * 2 | 2 byte each value for the coin denominations (e.g. 0.05 coin = 0x05,0x00) |
| Country codes | 10 + (n * 2) | | An obsolete value showing security level. This is set to 2 if the value multiplier is > 0 otherwise 0. |

<< back to index

## SMART HOPPER Command Table

| | Header code (hex) | dec |
|---|---|---|
| Sync | 0x11 | 17 |
| Reset | 0x01 | 1 |
| Host Protocol Version | 0x06 | 6 |
| Poll | 0x07 | 7 |
| Get Serial Number | 0x0C | 12 |
| Disable | 0x09 | 9 |
| Enable | 0x0A | 10 |
| Get Firmware Version | 0x20 | 32 |
| Get Dataset Version | 0x21 | 33 |
| Setup Request | 0x05 | 5 |
| Poll With Ack | 0x56 | 86 |
| Event Ack | 0x57 | 87 |
| Set Denomination Route | 0x3B | 59 |
| Get Denomination Route | 0x3C | 60 |
| Payout Amount | 0x33 | 51 |
| Get Denomination Level | 0x35 | 53 |
| Set Denomination Level | 0x34 | 52 |
| Halt Payout | 0x38 | 56 |
| Float Amount | 0x3D | 61 |
| Get Min Payout | 0x3E | 62 |
| Set Coin Mech Inhibits | 0x40 | 64 |
| Payout By Denomination | 0x46 | 70 |
| Float By Denomination | 0x44 | 68 |
| Empty All | 0x3F | 63 |
| Set Options | 0x50 | 80 |
| Get Options | 0x51 | 81 |
| Coin Mech Global Inhibit | 0x49 | 73 |
| Smart Empty | 0x52 | 82 |
| Cashbox Payout Operation Data | 0x53 | 83 |
| Get All Levels | 0x22 | 34 |
| Set Generator | 0x4A | 74 |
| Set Modulus | 0x4B | 75 |
| Request Key Exchange | 0x4C | 76 |
| Coin Mech Options | 0x5A | 90 |
| Get Build Revision | 0x4F | 79 |
| Comms Pass Through | 0x37 | 55 |
| Set Baud Rate | 0x4D | 77 |
| Ssp Set Encryption Key | 0x60 | 96 |
| Ssp Encryption Reset To Default | 0x61 | 97 |
| Set Cashbox Payout Limit | 0x4E | 78 |

## SMART HOPPER Event Table

| | Header code (hex) | dec |
|---|---|---|
| Slave Reset | 0xF1 | 241 |
| Disabled | 0xE8 | 232 |
| Fraud Attempt | 0xE6 | 230 |
| Initialising | 0xB6 | 182 |
| Dispensing | 0xDA | 218 |
| Coins Low | 0xD3 | 211 |
| Hopper Jammed | 0xD5 | 213 |
| Halted | 0xD6 | 214 |
| Floating | 0xD7 | 215 |
| Floated | 0xD8 | 216 |
| Timeout | 0xD9 | 217 |
| Incomplete Payout | 0xDC | 220 |
| Incomplete Float | 0xDD | 221 |
| Cashbox Paid | 0xDE | 222 |
| Coin Credit | 0xDF | 223 |
| Coin Mech Jammed | 0xC4 | 196 |
| Coin Mech Return Active | 0xC5 | 197 |
| Emptying | 0xC2 | 194 |
| Emptied | 0xC3 | 195 |
| Smart Emptying | 0xB3 | 179 |
| Smart Emptied | 0xB4 | 180 |
| Calibration Failed | 0x83 | 131 |
| Coin Mech Error | 0xB7 | 183 |
| Attached Coin Mech Disabled | 0xBD | 189 |
| Attached Coin Mech Enabled | 0xBE | 190 |

<< back to index

## SMART SYSTEM Command Table

| | Header code (hex) | dec |
|---|---|---|
| Sync | 0x11 | 17 |
| Reset | 0x01 | 1 |
| Host Protocol Version | 0x06 | 6 |
| Poll | 0x07 | 7 |
| Get Serial Number | 0x0C | 12 |
| Disable | 0x09 | 9 |
| Enable | 0x0A | 10 |
| Get Firmware Version | 0x20 | 32 |
| Get Dataset Version | 0x21 | 33 |
| Set Inhibits | 0x02 | 2 |
| Setup Request | 0x05 | 5 |
| Poll With Ack | 0x56 | 86 |
| Event Ack | 0x57 | 87 |
| Set Denomination Route | 0x3B | 59 |
| Get Denomination Route | 0x3C | 60 |
| Payout Amount | 0x33 | 51 |
| Get Denomination Level | 0x35 | 53 |
| Set Denomination Level | 0x34 | 52 |
| Halt Payout | 0x38 | 56 |
| Float Amount | 0x3D | 61 |
| Get Min Payout | 0x3E | 62 |
| Set Coin Mech Inhibits | 0x40 | 64 |
| Payout By Denomination | 0x46 | 70 |
| Float By Denomination | 0x44 | 68 |
| Empty All | 0x3F | 63 |
| Set Options | 0x50 | 80 |
| Get Options | 0x51 | 81 |
| Coin Mech Global Inhibit | 0x49 | 73 |
| Smart Empty | 0x52 | 82 |
| Cashbox Payout Operation Data | 0x53 | 83 |
| Get All Levels | 0x22 | 34 |
| Get Counters | 0x58 | 88 |
| Reset Counters | 0x59 | 89 |
| Set Generator | 0x4A | 74 |
| Set Modulus | 0x4B | 75 |
| Request Key Exchange | 0x4C | 76 |
| Coin Mech Options | 0x5A | 90 |
| Get Build Revision | 0x4F | 79 |
| Comms Pass Through | 0x37 | 55 |
| Set Baud Rate | 0x4D | 77 |
| Ssp Set Encryption Key | 0x60 | 96 |
| Ssp Encryption Reset To Default | 0x61 | 97 |
| Get Real Time Clock Configuration | 0x62 | 98 |
| Set Real Time Clock | 0x64 | 100 |
| Get Real Time Clock | 0x63 | 99 |
| Set Cashbox Payout Limit | 0x4E | 78 |
| Coin Stir | 0x5D | 93 |
| Payout Amount By Denomination | 0x39 | 57 |

## SMART SYSTEM Event Table

| | Header code (hex) | dec |
|---|---|---|
| Slave Reset | 0xF1 | 241 |
| Disabled | 0xE8 | 232 |
| Fraud Attempt | 0xE6 | 230 |
| Initialising | 0xB6 | 182 |
| Dispensing | 0xDA | 218 |
| Dispensed | 0xD2 | 210 |
| Hopper Jammed | 0xD5 | 213 |
| Halted | 0xD6 | 214 |
| Floating | 0xD7 | 215 |
| Floated | 0xD8 | 216 |
| Timeout | 0xD9 | 217 |
| Incomplete Payout | 0xDC | 220 |
| Incomplete Float | 0xDD | 221 |
| Cashbox Paid | 0xDE | 222 |
| Coin Mech Jammed | 0xC4 | 196 |
| Coin Mech Return Active | 0xC5 | 197 |
| Emptying | 0xC2 | 194 |
| Emptied | 0xC3 | 195 |
| Smart Emptying | 0xB3 | 179 |
| Smart Emptied | 0xB4 | 180 |
| Calibration Failed | 0x83 | 131 |
| Device Full | 0xCF | 207 |
| Coin Mech Error | 0xB7 | 183 |
| Attached Coin Mech Disabled | 0xBD | 189 |
| Attached Coin Mech Enabled | 0xBE | 190 |
| Value Added | 0xBF | 191 |
| Pay-in Active | 0xC1 | 193 |

<< back to index

| Command | Code hex | Code decimal |
|---|---|---|
| **Sync** | 0x11 | 17 |

| Implemented on | Encryption Required |
|---|---|
| SMART HOPPER, SMART SYSTEM | **optional** |

## Description

SSP uses a system of sequence bits to ensure that packets have been received by the slave and the reply received by the host. If the slave receives the same sequence bit as the previous command packet then this is signal to re-transmit the last reply.

A mechanism is required to initially set the host and slave to the same sequence bits and this is done by the use of the SYNC command.

A Sync command resets the seq bit of the packet so that the slave device expects the next seq bit to be 0. The host then sets its next seq bit to 0 and the seq sequence is synchronised.

The SYNC command should be the first command sent to the slave during a session.

## Packet examples

Set seq bit to 1

Host transmit: **7F 80 01 11 65 82**
Slave Reply: **7F 80 01 F0 23 80**

<< back to index

| Command | Code hex | Code decimal |
|---|---|---|
| **Reset** | 0x01 | 1 |

| Implemented on | Encryption Required |
|---|---|
| SMART HOPPER, SMART SYSTEM | **optional** |

| Description |
|---|

Performs a software and hardware reset of the device.

After this command has been acknowledged with **OK (0xF0)**, any encryption, baud rate changes, etc will be reset to default settings.

| Packet examples |
|---|

No data parameters, sequence bit set and address 0

Host transmit: **7F 80 01 01 06 02**
Slave Reply: **7F 80 01 F0 23 80**

<< back to index

| Command | Code hex | Code decimal |
|---|---|---|
| **Host Protocol Version** | 0x06 | 6 |

| Implemented on | Encryption Required |
|---|---|
| SMART HOPPER, SMART SYSTEM | **optional** |

## Description

ITL SSP devices use a system of protocol levels to control the event responses to polls to ensure that changes would not affect systems with finite state machines unable to test for new events with non-defined data lengths.

Use this command to allow the host to set which protocol version to operate the slave device.

If the device supports the requested protocol **OK (0xF0)** will be returned. If not then **FAIL (0xF8)** will be returned

## Packet examples

The slave supports the protocol version 8

Host transmit: **7F 80 02 06 08 03 94**
Slave Reply: **7F 80 01 F0 23 80**

Host protocol version 9 not supported

Host transmit: **7F 80 02 06 09 06 14**
Slave Reply: **7F 80 01 F8 10 00**

<< back to index

| Command | Code hex | Code decimal |
|---|---|---|
| **Poll** | 0x07 | 7 |

| Implemented on | Encryption Required |
|---|---|
| SMART HOPPER, SMART SYSTEM | **optional** |

## Description

This command returns a list of events occured in the device since the last poll was sent.

The SSP devices share some common events and have some unique events of their own. See event tables for details for a specific device.

## Packet examples

Poll command returning device reset and disabled response

Host transmit: **7F 80 01 07 12 02**
Slave Reply: **7F 80 03 F0 F1 F8 DC 0C**

Event response note credit channel 1 and note stacked

Host transmit: **7F 80 01 07 12 02**
Slave Reply: **7F 80 04 F0 EE 01 EB B9 48**

<< back to index

| Command | Code hex | Code decimal |
|---|---|---|
| **Get Serial Number** | 0x0C | 12 |

| Implemented on | Encryption Required |
|---|---|
| SMART HOPPER, SMART SYSTEM | **optional** |

| Description |
|---|

This command returns a 4-byte big endian array representing the unique factory programmed serial number of the device.

| Packet examples |
|---|

The device responds with 4 bytes of serial number data. In this case, the serial number is 01873452 = 0x1c962c. The return array is formatted as big endian (MSB first).

Host transmit: **7F 80 01 0C 2B 82**
Slave Reply: **7F 80 05 F0 00 1C 96 2C D4 97**

<< back to index

| Command | Code hex | Code decimal |
|---|---|---|
| **Disable** | 0x09 | 9 |

| Implemented on | Encryption Required |
|---|---|
| SMART HOPPER, SMART SYSTEM | **optional** |

| Description |
|---|

Disabled the slave device from operation.

For example, this command would block a banknote validator from allowing any more banknotes to be entered.

For most SSP devices, the default state is to be disabled after reset.

| Packet examples |
|---|

Single byte command with no parameters

Host transmit: **7F 80 01 09 35 82**
Slave Reply: **7F 80 01 F0 23 80**

NV11 when note float is jammed/disconnected responds COMMAND_CANNOT_BE_PROCESSED

Host transmit: **7F 80 01 09 35 82**
Slave Reply: **7F 80 01 F5 3D 80**

<< back to index

| Command | Code hex | Code decimal |
|---|---|---|
| **Enable** | 0x0A | 10 |

| Implemented on | Encryption Required |
|---|---|
| SMART HOPPER, SMART SYSTEM | **optional** |

| Description |
|---|

This command will enable the SSP device for normal operation. For example, it will allow a banknote validator to commence validating banknotes entered into it's bezel.

| Packet examples |
|---|

Single byte command with no parameters

Host transmit: **7F 80 01 0A 3F 82**
Slave Reply: **7F 80 01 F0 23 80**

NV11 when note float is jammed/disconnected responds COMMAND_CANNOT_BE_PROCESSED

Host transmit: **7F 80 01 0A 3F 82**
Slave Reply: **7F 80 01 F5 3D 80**

<< back to index

| Command | Code hex | Code decimal |
|---|---|---|
| **Get Firmware Version** | 0x20 | 32 |

| Implemented on | Encryption Required |
|---|---|
| SMART HOPPER, SMART SYSTEM | **optional** |

| Description |
|---|

Returns a variable length ASCII array containg the full firmware version of the attached device.

| Packet examples |
|---|

In this example, the firmware version of the device is: NV02004141498000

| | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Host transmit: | **7F** | **80** | **01** | **20** | **C0** | **02** | | | | | | | | | | | | | | |
| Slave Reply: | **7F** | **80** | **11** | **F0** | **4E** | **56** | **30** | **32** | **30** | **30** | **34** | **31** | **34** | **31** | **34** | **39** | **38** | **30** | **30** | **30** | **DE** | **55** |
| ascii: | | | | | **.** | **N** | **V** | **0** | **2** | **0** | **0** | **4** | **1** | **4** | **1** | **4** | **9** | **8** | **0** | **0** | **0** | |

<< back to index

| Command | Code hex | Code decimal |
|---|---|---|
| **Get Dataset Version** | 0x21 | 33 |

| Implemented on | Encryption Required |
|---|---|
| SMART HOPPER, SMART SYSTEM | **optional** |

| Description |
|---|

Returns a varibale length ASCII array giving the installed dataset version of the device.

| Packet examples |
|---|

This example shows a device with dataset version EUR01610.

Host transmit: **7F 80 01 21 C5 82**
Slave Reply: **7F 80 09 F0 45 55 52 30 31 36 31 30 B8 2A**
ascii: **. E U R 0 1 6 1 0**

<< back to index

| Command | Code hex | Code decimal |
|---|---|---|
| **Set Inhibits** | 0x02 | 2 |

| Implemented on | Encryption Required |
|---|---|
| SMART SYSTEM | **optional** |

| Description |
|---|

Sets the channel inhibit level for the device. each byte sent represents 8 bits (channels of inhibit).

Nv200 has the option to send 2,3,or 4 bytes to represent 16,24, or 64 channels, the other BNV devices have the option of sending 1 or 2 bytes for 8 or 16 channel operation.

Set the bit low to inhibit all note acceptance on that channel, high to allow note acceptance.

| Packet examples |
|---|

Set channels 1-3 enabled, 4-16 inhibited

Host transmit: **7F 80 03 02 07 00 2B B6**
Slave Reply: **7F 80 01 F0 23 80**

All channels enabled

Host transmit: **7F 80 03 02 FF FF 25 A4**
Slave Reply: **7F 80 01 F0 23 80**

<< back to index

| Command | Code hex | Code decimal |
|---|---|---|
| **Setup Request** | 0x05 | 5 |

| Implemented on | Encryption Required |
|---|---|
| SMART HOPPER, SMART SYSTEM | **optional** |

| Description |
|---|

Request the set-up configuration of the device. Gives details about versions, channel assignments, country codes and values.

Each device type has a different return data format. Please refer to the table information for each individual device.

SMART Ticket/Coupon Printer Response

| Smart Ticket Data | Response Offset | Size | Notes |
|---|---|---|---|
| Unit Type | 0 | 1 | 0x08 = SMART Ticket, 0x0B = Coupon Printer |
| Firmware Version | 1 | 4 | Ascii data of device firmware (eg 0123) |
| Cutter Enabled | 5 | 1 | (0 for disabled) |
| Tab enabled status | 6 | 1 | (0 for disabled) |
| Reverse validation enabled status | 7 | 1 | (0 for disabled) |
| Font pack code (ASCII) | 8 | 3 | e.g. FP1 |
| Printer type | 11 | 1 | Printer Type: 0x0 for Fan Fold, 0x1 Paper Roll (Cutter fitted) |
| SD card fitted status | 12 | 1 | (1 for detected) |
| Printer darkness/quality setting | 13 | 1 | value between 0 - 3 |
| SSP Protocol Version | 14 | 1 | |

| Packet examples |
|---|

This example shows the data returned for a BNV with GBP dataset, firmware version 1.00, 3 channels GBP 5, GBP 10, GBP 20

| | |
|---|---|
| Host transmit: | **7F 80 01 05 1D 82** |
| Slave Reply: | **7F 80 17 F0 00 30 31 30 30 47 42 50 00 00 01 03 05 0A 14 02 02 02 40 00 00 05 61 81** |
| ascii: | **. . 0 1 0 0 G B P . . . . . @ . . .** |

<< back to index

| Command | Code hex | Code decimal |
|---|---|---|
| **Poll With Ack** | 0x56 | 86 |

| Implemented on | Encryption Required |
|---|---|
| SMART HOPPER, SMART SYSTEM | **yes** |

| Description |
|---|

A command that behaves in the same way as the Poll command but with this command, the specified events will need to be acknowledged by the host using the EVENT ACK command (0x56).

The events will repeat until the EVENT ACK command is sent and the BNV will not allow any further note actions until the event has been cleared by the EVENT ACK command. If this command is not supported by the slave device, then generic response 0xF2 will be returned and standard poll command (0x07) will have to be used.

| Packet examples |
|---|

<< back to index

| Command | Code hex | Code decimal |
|---|---|---|
| **Event Ack** | 0x57 | 87 |

| Implemented on | Encryption Required |
|---|---|
| SMART HOPPER, SMART SYSTEM | **yes** |

| Description |
|---|

This command will clear a repeating Poll ACK response and allow further note operations.

| Packet examples |
|---|

Host transmit: **7F 80 01 57 F2 03**
Slave Reply: **7F 80 01 F0 23 80**

<< back to index

| Command | Code hex | Code decimal |
|---|---|---|
| **Set Denomination Route** | 0x3B | 59 |

| Implemented on | Encryption Required |
|---|---|
| SMART HOPPER, SMART SYSTEM | **yes** |

| Description |
|---|

This command will configure the denomination to be either routed to the cashbox on detection or stored to be made available for later possible payout.

**Note on protocol versions: For protocol versions less than 6 a value only data array is sent. For protocol version greater or equal to 6, a 3 byte country code is also sent to allow mulit-currency functionality to the payout.**

**Please note that there exists a difference in the data format between SMART Payout and SMART Hopper for protocol versions less than 6. In these protocol versions the value was determined by a 2 byte array rather than 4 byte array for SMART Hopper.**

For NV11 devices the host must send the required note value in the same form that the device is set to report by (see Set Value Reporting Type command).

Protocol version less than 6 command format:

| byte | function | size |
|---|---|---|
| 0 | requested route (0 = payout, 1= cashbox) | 1 |
| 1 | value (2 bytes for hopper, 4 bytes for others) | 2 or 4 |

Protocol version greater of equal to 6 format:

| byte | function | size |
|---|---|---|
| 0 | requested route (0 = payout, 1= cashbox) | 1 |
| 1 | value of requested denomination to route (4 byte integer) | 4 |
| 5 | ASCII country code of requested denomination | 3 |

With note payouts, the device responds with COMMAND CANNOT BE PROCESSED and an error byte for request failure:

| Error | code |
|---|---|
| No payout connected | 1 |
| Invalid currency detected | 2 |
| Payout device failure | 3 |

## Packet examples

An example of a request to route a 10c EUR coin to be stored for payout using protocol version 6

Host transmit: **7F 80 09 3B 00 0A 00 00 00 45 55 52 08 43**
Slave Reply: **7F 80 01 F0 23 80**

Example command with error response Invalid currency detected

Host transmit: **7F 80 09 3B 00 0A 00 00 00 45 55 52 08 43**
Slave Reply: **7F 80 02 F5 02 30 3E**

<< back to index

| Command | Code hex | Code decimal |
|---|---|---|
| **Get Denomination Route** | 0x3C | 60 |

| Implemented on | Encryption Required |
|---|---|
| SMART HOPPER, SMART SYSTEM | **yes** |

| Description |
|---|

This command allows the host to determine the route of a denomination.
**Note protocol versions:**
For protocol versions less than 6 a value only data array is sent. For protocol version greater or equal to 6, a 3 byte country code is also sent to allow multi-currency functionality to the payout.

**Please note that there exists a difference in the data format between SMART Payout and SMART**
**Hopper for protocol versions less than 6. In these protocol versions the value was determined by a 2 byte array rather than 4 byte array**
For NV11 devices the host must send the required note value in the same form that the device is set to report by (see Set Value Reporting Type command).

Protocol version less than 6 command format:

| byte | function | size |
|---|---|---|
| 0 | value (2 bytes for hopper, 4 bytes for others) | 2 or 4 |

Protocol version greater of equal to 6 format:

| byte | function | size |
|---|---|---|
| 0 | value of requested denomination to route (4 byte integer) | 4 |
| 4 | ASCII country code of requested denomination | 3 |

The device responds with a data byte representing the current route of the denomination.

| byte | function | size |
|---|---|---|
| 0 | Generic OK | 1 |
| 1 | Route (0 = recycle for payout,1 = system cashbox) | 1 |

With note payouts, the device responds with COMMAND CANNOT BE PROCESSED and an error byte for request failure:

| Error | code |
|---|---|
| No payout connected | 1 |
| Invalid currency detected | 2 |
| Payout device failure | 3 |

## Packet examples

This example shows a request to obtain the route of EUR 5.00 note in protocol version 6. Returns 0 for payout.

Host transmit: **7F 80 08 3C F4 01 00 00 45 55 52 2F 0E**
Slave Reply: **7F 80 02 F0 00 3F A0**

<< back to index

| Command | Code hex | Code decimal |
|---|---|---|
| **Payout Amount** | 0x33 | 51 |

| Implemented on | Encryption Required |
|---|---|
| SMART HOPPER, SMART SYSTEM | **yes** |

| Description |
|---|

A command to set the monetary value to be paid by the payout unit.

**This command was expanded after and including protocol version 6 to include country codes and payout test option.**

Command format protocol version less than 6:

| byte | function | size |
|---|---|---|
| 0 | payout value (4 byte integer of the full penny amount) | 4 |

Command format protocol greater than or equal to 6:

| byte | function | size |
|---|---|---|
| 0 | payout value (4 byte integer of the full penny amount) | 4 |
| 4 | ASCII country code of currency to pay | 3 |
| 8 | Option byte (TEST_PAYOUT_AMOUT 0x19, PAYOUT_AMOUNT 0x58), | 1 |

For request failure, the device responds with COMMAND CANNOT BE PROCESSED and a data byte showing the
error code.

| Error | Code |
|---|---|
| Not enough value in device | 1 |
| Cannot pay exact amount | 2 |
| Device busy | 3 |
| Device disabled | 4 |

## Packet examples

Shows a request to payout EUR 5.00 using protocol version 4

Host transmit: **7F 80 05 33 F4 01 00 00 32 50**
Slave Reply: **7F 80 01 F0 23 80**

Shows an example is a request to payout EUR 5.00 in protocol version 6 with commit option.

Host transmit: **7F 80 09 33 F4 01 00 00 45 55 52 58 C3 EE**
Slave Reply: **7F 80 01 F0 23 80**

Shows an example is a request to payout EUR 5.00 in protocol version 6 failed due to cannot pay exact amount

Host transmit: **7F 80 09 33 F4 01 00 00 45 55 52 58 C3 EE**
Slave Reply: **7F 80 02 F5 02 30 3E**

<< back to index

| Command | Code hex | Code decimal |
|---|---|---|
| **Get Denomination Level** | 0x35 | 53 |

| Implemented on | Encryption Required |
|---|---|
| SMART HOPPER, SMART SYSTEM | **optional** |

## Description

This command returns the level of a denomination stored in a payout device as a 2 byte value.

This command was expanded in protocol version 6 to include country codes for multi-currency functionality.

Protocol version 5 command format:

| byte | function | size |
|---|---|---|
| 0 | 4 byte value of denomination requested | 4 |

Protocol version 6 and greater command format:

| byte | function | size |
|---|---|---|
| 0 | 4 byte value of denomination requested | 4 |
| 4 | ASCII country code of denomination required | 3 |

## Packet examples

Example shows a request to find the amount of 0.10c coins in protocol version 5. Returns a level of 100

Host transmit: **7F 80 05 35 0A 00 00 00 1E 49**
Slave Reply: **7F 80 03 F0 64 00 C5 F0**

Shows a request to find the level of EUR 5.00 notes using protocol version 6. Returns 12.

Host transmit: **7F 80 08 35 F4 01 00 00 45 55 52 19 9E**
Slave Reply: **7F 80 03 F0 0C 00 C3 80**

If the denomination is not in the device, it will respond with COMMAND CANNOT BE PROCESSED

Host transmit: **7F 80 08 35 F4 01 00 00 45 55 52 19 9E**
Slave Reply: **7F 80 01 F5 3D 80**

<< back to index

| Command | Code hex | Code decimal |
|---|---|---|
| **Set Denomination Level** | 0x34 | 52 |

| Implemented on | Encryption Required |
|---|---|
| SMART HOPPER, SMART SYSTEM | **yes** |

Description

A command to increment the level of coins of a denomination stored in the hopper. The command is formatted with the command byte first, amount of coins to add as a 2-byte little endian, the value of coin as 2-byte little endian and (if using protocol version 6) the country code of the coin as 3 byte ASCII. The level of coins for a denomination can be set to zero by sending a zero level for that value.

**This command was updated when using version 6 and greater to allow for larger 4 byte coin values and country codes.**

Protocol version less than 6:

| byte | function | size |
|---|---|---|
| 0 | number of coins to add to level (0 will clear the level) | 2 |
| 2 | value fo denimonation to set | 2 |

Protocol version great or equal to 6:

| byte | function | size |
|---|---|---|
| 0 | number of coins to add to level (0 will clear the level) | 2 |
| 2 | value of denomination to set | 4 |
| 6 | ASCII country code of denomination | 3 |

Packet examples

Example to increase the level of .50c coin by 20 using protocol version 5

Host transmit: **7F 80 05 34 14 00 32 00 63 FD**
Slave Reply: **7F 80 01 F0 23 80**

Example to increase the level of EUR 1.00 coins by 12 on a device set with protocol version 6

Host transmit: **7F 80 0A 34 0C 00 64 00 00 00 45 55 52 C7 28**

Slave Reply: **7F 80 01 F0 23 80**

<< back to index

| Command | Code hex | Code decimal |
|---|---|---|
| **Halt Payout** | 0x38 | 56 |

| Implemented on | Encryption Required |
|---|---|
| SMART HOPPER, SMART SYSTEM | **yes** |

| Description |
|---|

A command to stop the execution of an existing payout. The device will stop payout at the earliest convenient place and generate a Halted event giving the value paid up to that point.

| Packet examples |
|---|

Ok response for halt command accepted.

Host transmit: **7F 80 01 38 90 02**
Slave Reply: **7F 80 01 F0 23 80**

<< back to index

| Command | Code hex | Code decimal |
|---|---|---|
| **Float Amount** | 0x3D | 61 |

| Implemented on | Encryption Required |
|---|---|
| SMART HOPPER, SMART SYSTEM | **yes** |

| Description |
|---|

A command to float the payout unit to leave a requested value of money, with a requested minimum possible payout level. All monies not required to meet float value are routed to cashbox. Using protocol version 6, the host also sends a pre-test option byte (TEST_FLOAT_AMOUT 0x19, FLOAT_AMOUNT 0x58), which will determine if the command amount is tested or floated. This is useful for multi-payout systems so that the ability to pay a split down amount can be tested before committing to actual float.

**This command was expanded after and including protocol version 6 to include country codes and payout test option.**

Command format protocol version less than 6:

| byte | function | size |
|---|---|---|
| 0 | value of minimum payout to remain | 2 |
| 2 | float value (4 byte integer of the full penny amount) | 4 |

Command format protocol greater than or equal to 6:

| byte | function | size |
|---|---|---|
| 0 | value of minimum payout to remain | 2 |
| 2 | payout value (4 byte integer of the full penny amount) | 4 |
| 6 | ASCII country code of currency to pay | 3 |
| 9 | Option byte (TEST_FLOAT_AMOUT 0x19, FLOAT_AMOUNT 0x58), | 1 |

For request failure, the device responds with COMMAND CANNOT BE PROCESSED and a data byte showing the
error code.

| Error | Code |
|---|---|
| Not enough value in device | 1 |
| Cannot pay exact amount | 2 |
| Device busy | 3 |
| Device disabled | 4 |

## Packet examples

Example to request to float to a value of 100.00 leaving a min possible payout of 0.50c for protocol version 5

Host transmit: **7F 80 07 3D 32 00 10 27 00 00 1D 1C**
Slave Reply: **7F 80 01 F0 23 80**

In protocol version greater than 6, we add a 3 byte ascii country code and a test or commit data byte. In this example a request to float to a value of EUR 100.00 leaving a min possible payout of 0.50c

Host transmit: **7F 80 0B 3D 32 00 27 10 00 00 45 55 52 58 A7 DA**
Slave Reply: **7F 80 01 F0 23 80**

<< back to index

| Command | Code hex | Code decimal |
|---|---|---|
| **Get Min Payout** | 0x3E | 62 |

| Implemented on | Encryption Required |
|---|---|
| SMART HOPPER, SMART SYSTEM | **optional** |

| Description |
|---|

A command to request the minimum possible payout amount that this device can provide.

For protocol versions less than 6, no parameters are sent.

For protocol version 6 or greater, we add the 3 byte country code of the country we are requesting.

| Packet examples |
|---|

Example for protocol version 5 returning min payout of 200

Host transmit: **7F 80 01 3E 84 02**
Slave Reply: **7F 80 05 F0 C8 00 00 00 A7 C2**

Protocol version 6 example returning a min payout value of 5.00 EUR

Host transmit: **7F 80 04 3E 45 55 52 14 E3**
ascii: **. . > E U R . .**
Slave Reply: **7F 80 05 F0 F4 01 00 00 BA 72**
ascii: **. . . . .**

<< back to index

| Command | Code hex | Code decimal |
|---|---|---|
| **Set Coin Mech Inhibits** | 0x40 | 64 |

| Implemented on | Encryption Required |
|---|---|
| SMART HOPPER, SMART SYSTEM | **yes** |

## Description

This command is used to enable or disable acceptance of individual coin values from a coin acceptor connected to the hopper.

Protocol versions less than 6:

| byte | function | size |
|---|---|---|
| 0 | Requested inhibit state (0 =inhibit,1=enable) | 1 |
| 1 | coin value (2 byte integer) | 2 |

Protocol versions greater or equal to 6:.

| byte | function | size |
|---|---|---|
| 0 | Requested inhibit state (0 =inhibit,1=enable) | 1 |
| 1 | coin value (2 byte integer) | 2 |
| 3 | ASCII country code of value | 3 |

## Packet examples

Example we want to enable acceptance of EUR 0.50c coins in protocol version 6.

Host transmit: **7F 80 07 40 01 32 00 45 55 52 CA 5E**
ascii: **. . @ . 2 . E U R . ^**
Slave Reply: **7F 80 01 F0 23 80**

<< back to index

| Command | Code hex | Code decimal |
|---|---|---|
| **Payout By Denomination** | 0x46 | 70 |

| Implemented on | Encryption Required |
|---|---|
| SMART HOPPER, SMART SYSTEM | **yes** |

| Description |
|---|

A command to payout the requested quantity of individual denominations.

***Requires Protocol Version 6 or above.***
***Attempting to use the command with an earlier protocol version will generate a response 0xF4 (parameter out of range).***

The quantities of denominations to pay are sent as a 2 byte little endian array; the money values as 4-byte little endian array and the country code as a 3-byte ASCII array.

The host also adds an option byte to the end of the command array (TEST_PAYOUT_AMOUT 0x19 or PAYOUT_AMOUNT 0x58). This will allow a pre-test of the ability to payout the requested levels before actual payout executes.

Command format:

| byte | function | size |
|---|---|---|
| 0 | the number of individual requests in this command (max 20) | 1 |
| 1 | the number to pay | 2 |
| 3 | the denomination value | 4 |
| 7 | the denomination ASCII country code | 3 |
| 10 | repeat block for each required denomination | |
| | The option byte (TEST_FLOAT_AMOUT 0x19 or FLOAT_AMOUNT 0x58). | 1 |

For request failure, the device responds with COMMAND CANNOT BE PROCESSED and a data byte showing the
error code.

| Error | Code |
|---|---|
| Not enough value in device | 1 |
| Cannot pay exact amount | 2 |
| Device busy | 3 |
| Device disabled | 4 |

## Packet examples

Example - A hopper unit has stored 100 x 0.10 EUR, 50 x 0.20 EUR, 30 x 1.00 EUR, 10 x 1.00 GBP, 50 x 0.50 GBP and the host wishes to payout to 5 x 1.00 EUR, 5 x 0.10 EUR, 3 x 1.00 GBP and 2 x 0.50 GBP.

```
Host transmit:  7F 80 27 46 04 04 00 64 00 00 00 45 55 52 05 00 0A 00 00 00 45 55 52 03
                00 64 00 00 00 47 42 50 02 00 32 00 00 00 47 42 50 58 94 B7
ascii:             .  '  F  .  .  .  d  .  .  .  E  U  R  .  .  .  .  .  .  E  U  R  .
                .  d  .  .  .  G  B  P  .  .  2  .  .  .  G  B  P  X  .  .
Slave Reply:    7F 80 01 F0 23 80
```

<< back to index

| Command | Code hex | Code decimal |
|---|---|---|
| **Float By Denomination** | 0x44 | 68 |

| Implemented on | Encryption Required |
|---|---|
| SMART HOPPER, SMART SYSTEM | **yes** |

| Description |
|---|

A command to float (leave in device) the requested quantity of individual denominations.

***Requires Protocol Version 6 or above.***
***Attempting to use the command with an earlier protocol version will generate a response 0xF4 (parameter out of range).***

The quantities of denominations to leave are sent as a 2 byte little endian array; the money values as 4-byte little endian array and the country code as a 3-byte ASCII array. The host also adds an option byte to the end of the command array (TEST_PAYOUT_AMOUT 0x19 or PAYOUT_AMOUNT 0x58). This will allow a pre-test of the ability to float to the requested levels before actual float executes.

Command format:

| byte | function | size |
|---|---|---|
| 0 | the number of individual requests in this command (max 20) | 1 |
| 1 | the number required to leave in device (little endian array) | 2 |
| 3 | the denomination value (little endian array) | 4 |
| 7 | the denomination ASCII country code | 3 |
| 10... | repeat block for each required denomination | |
| last | The option byte (TEST_FLOAT_AMOUT 0x19 or FLOAT_AMOUNT 0x58). | 1 |

For request failure, the device responds with COMMAND CANNOT BE PROCESSED and a data byte showing the
error code.

| Error | Code |
| --- | --- |
| Not enough value in device | 1 |
| Cannot pay exact amount | 2 |
| Device busy | 3 |
| Device disabled | 4 |

Events used to indicate progress:

While floating is being carried out, the Floating and Floated events are used to keep the host informed.

| Packet examples |
| --- |

<< back to index

| Command | Code hex | Code decimal |
|---|---|---|
| **Empty All** | 0x3F | 63 |

| Implemented on | Encryption Required |
|---|---|
| SMART HOPPER, SMART SYSTEM | **yes** |

| Description |
|---|

This command will direct all stored monies to the cash box without reporting any value and reset all the stored counters to zero. See Smart Empty command to record the value emptied.

A poll command during this process will respond with Emptying and Empty events

| Packet examples |
|---|

Command format (no parameters) for acknowledged request.

Host transmit: **7F 80 01 3F 81 82**
Slave Reply: **7F 80 01 F0 23 80**

<< back to index

| Command | Code hex | Code decimal |
|---|---|---|
| **Set Options** | 0x50 | 80 |

| Implemented on | Encryption Required |
|---|---|
| SMART HOPPER, SMART SYSTEM | **yes** |

| Description |
|---|

The host can set the following options for the Smart Hopper. These options do not persist in memory and after a reset they will This command is valid only when using protocol version 6 or greater.

Table below shows the available options for the SMART Hopper. The command data is formatted as a 2 byte register REG_0 and REG_1

**Reg_0 bits and their meaning**

| Bit | parameter | |
|---|---|---|
| 0 | pay mode | Split by highest value (0x00) The device will attempt to payout a requested value by starting from the highest to the lowest coins available. This mode will payout the minimum number of coins possible. Free pay (0x01) (Default state after reset). The device will payout a coin as it passes its discriminator system if it fits into the current payout value and will leave enough of other coins to payout the rest of the value. This may give a faster payout but could result in a large number of coins of small denominations paid out. |
| 1 | level check | Disabled (0x00). The device will not refer to the level counters when calculating if a payout value can be made. Enabled (0x01) (Default state after reset). The device will check the level counters and accept or refuse a payout request based on levels and/or split of available levels. |
| 2 | motor speed | Low speed (0x00). Payouts run at a lower motor speed. High Speed (Default state after reset) (0x01). The motors run at max speed for payouts. |
| 3 | cashbox pay active | This bit is used in conjunction with Bit 0. If bit 3 is zero, then the Pay modes will be as described in bit 0. If Bit 3 is set then coins routed to the cashbox will be used in coins paid out of the front if they can fit into the current payout request. |
| 4 | Route 0 level coins to cashbox | Set to 1 means that any coins detected with a level setting of 0 will be paid to the cashbox, even if it is routed to the payout |
| 5 | High efficiency split | Set to 1 to enable a more efficient, smarter coin payout algorithm which will tend to use coins which have higher level counts - thus speeding up the payout process |
| 6 | Unknown to payout | Set to 1 means any unknown coins will be paid out during Smart Empty (otherwise they will be routed to cashbox) |
| 7 | Value added | set to 0 for coin added event  set to 1 for value added event |

REG_1: required but not used so bits set to 0.

Response

When responding to this command, the Smart Hopper returns a byte which indicates the current operational mode as follows:

**Set Options: Response Codes**

| Code | Meaning |
|---|---|
| 0xFC | Highest split, use coins routed to cashbox in the split |
| 0xFD | Free pay, use coins routed to cashbox in the split |
| 0xFE | Highest split |
| 0xFF | Free pay |

Packet examples

The example shows a request to turn off level check, run at high speed and split by highest value.

Host transmit: **7F 80 03 50 04 00 40 38**
Slave Reply: **7F 80 02 F0 FE 38 22**

<< back to index

| Command | Code hex | Code decimal |
|---|---|---|
| **Get Options** | 0x51 | 81 |

| Implemented on | Encryption Required |
|---|---|
| SMART HOPPER, SMART SYSTEM | **yes** |

| Description |
|---|

This command returns 2 option register bytes described in Set Options command.

| Packet examples |
|---|

<< back to index

| Command | Code hex | Code decimal |
|---|---|---|
| **Coin Mech Global Inhibit** | 0x49 | 73 |

| Implemented on | Encryption Required |
|---|---|
| SMART HOPPER, SMART SYSTEM | **yes** |

| Description |
|---|

This command allows the host to enable/disable the attached coin mech in one command rather than by each individual value with previous firmware versions. Send this command and one Mode data byte: Data byte =0x00 - mech disabled. Date byte = 0x01 - mech enabled.

| Packet examples |
|---|

In this example we are sending a command to enable the coin mech.

Host transmit: **7F 80 02 49 01 33 36**
Slave Reply: **7F 80 01 F0 23 80**

<< back to index

| Command | Code hex | Code decimal |
|---|---|---|
| **Smart Empty** | 0x52 | 82 |

| Implemented on | Encryption Required |
|---|---|
| SMART HOPPER, SMART SYSTEM | **yes** |

## Description

Empties payout device of contents, maintaining a count of value emptied. The current total value emptied is given is response to a poll command. All coin counters will be set to 0 after running this command. Use Cashbox Payout Operation Data command to retrieve a breakdown of the denominations routed to the cashbox through this operation.

## Packet examples

Host transmit: **7F 80 01 52 EC 03**
Slave Reply: **7F 80 01 F0 23 80**

<< back to index

| Command | Code hex | Code decimal |
|---|---|---|
| **Cashbox Payout Operation Data** | 0x53 | 83 |

| Implemented on | Encryption Required |
|---|---|
| SMART HOPPER, SMART SYSTEM | **yes** |

## Description

Can be sent at the end of a SMART Empty, float or dispense operation. Returns the amount emptied to cashbox from the payout in the last dispense, float or empty command.

Response format:

| byte | function | size |
|---|---|---|
| 0 | generic OK | 1 |
| 1 | number of denominations in report | 2 |
| 3 | qty of denomination | 2 |
| 6 | denomination value | 4 |
| 10 | denomination country (ASCII) | 3 |
| ... | **repeated above block for each denomination** | **...** |
| ... | qauntity of unknown | 4 |

## Packet examples

<< back to index

| Command | Code hex | Code decimal |
|---|---|---|
| **Get All Levels** | 0x22 | 34 |

| Implemented on | Encryption Required |
|---|---|
| SMART HOPPER, SMART SYSTEM | **optional** |

## Description

Use this command to return all the stored levels of denominations in the device (including those at zero level).
This gives a faster response than sending each individual denomination level request.

Response data consists of blocks of nine bytes data for each denimonation in the device:

| byte | function | size |
|---|---|---|
| 0 | Generic OK | 1 |
| 1 | number of denominations in the device | 1 |
| 2 | level of denomination stored | 2 |
| 4 | denomination value (4 byte little endian integer) | 4 |
| 7 | denomination code (3 Byte ASCII) | 3 |
| 10.. | Repeat for each denomination | 9 |
| | | |

## Packet examples

In this example, we have a device coin dataset of EURO s with 20c,50c,1 EUR and 2 EUR. It currently has 100 x 20c, 65 x 50x, 0 x 1 EUR and 12 x 2 EUR.

Host transmit: **7F 80 01 22 CF 82**
Slave Reply: **7F 80 26 F0 04 64 00 14 00 00 00 45 55 52 41 00 32 00 00 00 45 55 52 00 00 64 00 00 00 45 55 52 0C 00 C8 00 00 00 45 55 52 84 D0**

<< back to index

| Command | Code hex | Code decimal |
|---|---|---|
| **Get Counters** | 0x58 | 88 |

| Implemented on | Encryption Required |
|---|---|
| SMART SYSTEM | **optional** |

| Description |
|---|

A command to return a global note activity counter set for the slave device. The response is formatted as in the table below and the counter values are persistent in memory after a power down- power up cycle.

These counters are note set independent and will wrap to zero and begin again if their maximum value is reached. Each counter is made up of 4 bytes of data giving a max value of 4294967295.

Response format:

| byte | function | size |
|---|---|---|
| 0 | Generic OK | 1 |
| 1 | Number of counters in set | 1 |
| 2 | Stacked | 4 |
| 6 | Stored | 4 |
| 10 | Dispensed | 4 |
| 14 | Transferred to stack | 4 |
| 18 | Rejected | 4 |

| Packet examples |
|---|

<< back to index

| Command | Code hex | Code decimal |
|---|---|---|
| **Reset Counters** | 0x59 | 89 |

| Implemented on | Encryption Required |
|---|---|
| SMART SYSTEM | **optional** |

| Description |
|---|

Resets the note activity counters described in Get Counters command to all zero values.

| Packet examples |
|---|

Command format (no parameters) for acknowledged request.

Host transmit: **7F 80 01 59 D5 83**
Slave Reply: **7F 80 01 F0 23 80**

<< back to index

| Command | Code hex | Code decimal |
|---|---|---|
| **Set Generator** | 0x4A | 74 |

| Implemented on | Encryption Required |
|---|---|
| SMART HOPPER, SMART SYSTEM | **optional** |

## Description

Part of the eSSP encryption negotiation sequence.

Eight data bytes are sent. This is a 64 bit number representing the Generator and must be a prime number. The slave will reply with OK or PARAMETER_OUT_OF_RANGE if the number is not prime.

## Packet examples

In this example we are sending the prime number 982451653. This = 3A8F05C5 hex

Host transmit: **7F 80 09 4A C5 05 8F 3A 00 00 00 00 B2 73**
Slave Reply: **7F 80 01 F0 23 80**

<< back to index

| Command | Code hex | Code decimal |
|---|---|---|
| **Set Modulus** | 0x4B | 75 |

| Implemented on | Encryption Required |
|---|---|
| SMART HOPPER, SMART SYSTEM | **optional** |

| Description |
|---|

Part of the eSSP encryption negotiation sequence.

Eight data bytes are sent. This is a 64 bit number representing the Moduls and must be a prime number. The slave will reply with OK or PARAMETER_OUT_OF_RANGE if the number is not prime.

| Packet examples |
|---|

In this example we are sending the prime number 1287821. This = 13A68D hex

Host transmit: **7F 80 09 4B 8D A6 13 00 00 00 00 00 6C F6**
Slave Reply: **7F 80 01 F0 23 80**

<< back to index

| Command | Code hex | Code decimal |
|---|---|---|
| **Request Key Exchange** | 0x4C | 76 |

| Implemented on | Encryption Required |
|---|---|
| SMART HOPPER, SMART SYSTEM | **optional** |

| Description |
|---|

The eight data bytes are a 64 bit number representing the Host intermediate key. If the Generator and Modulus have been set the slave will calculate the reply with the generic response and eight data bytes representing the slave intermediate key. The host and slave will then calculate the key.

If Generator and Modulus are not set then the slave will reply FAIL.

| Packet examples |
|---|

An example of Host intermediate key of 7554354432121 = 6DEE29CC879 hex

Host transmit: **7F 80 09 4C 79 C8 9C E2 DE 06 00 00 9D 52**
Slave Reply: **7F 80 01 F0 23 80**

<< back to index

| Command | Code hex | Code decimal |
|---|---|---|
| **Coin Mech Options** | 0x5A | 90 |

| Implemented on | Encryption Required |
|---|---|
| SMART HOPPER, SMART SYSTEM | **yes** |

## Description

The host can set the following options for the Smart Hopper. These options do not persist in memory and after a reset they will go to their default values.

Bit function
0 Coin Mech error events 1 = ccTalk format, 0 = Coin mech jam and Coin return mech open only
1:7 Unused set to 0

If coin mech error events are set to ccTalk format, then event Coin Mech Error 0xB7 is given with 1 byte ccTalk
coin mech error reason directly from coin mech ccTalk event queue. Otherwise only error events Coin Mech
Jam 0xC4 and Coin Mech Return 0xC5 are given.

## Packet examples

In this example we send register byte configured to return cctalk style events.

Host transmit: **7F 80 02 5A 01 30 DC**
Slave Reply: **7F 80 01 F0 23 80**

<< back to index

| Command | Code hex | Code decimal |
|---|---|---|
| **Get Build Revision** | 0x4F | 79 |

| Implemented on | Encryption Required |
|---|---|
| SMART HOPPER, SMART SYSTEM | **optional** |

| Description |
|---|

A command to return the build revision information of a device. The command returns 3 bytes of information representing the build of the product.

Byte 0 is the product type, next two bytes make up the revision number(0-65536).
For NV200 and Nv9usb, the type byte is 0, for Note Float, byte is 3 and for SMART Payout the byte is 6.

| Packet examples |
|---|

This example is from an NV200 (issue 20) with payout attached (issue 21).

Host transmit: **7F 80 01 4F A2 03**
Slave Reply: **7F 80 07 F0 00 14 00 06 15 00 0F 97**

<< back to index

| Command | Code hex | Code decimal |
|---|---|---|
| **Comms Pass Through** | 0x37 | 55 |

| Implemented on | Encryption Required |
|---|---|
| SMART HOPPER, SMART SYSTEM | **optional** |

## Description

The SMART Hopper includes two serial connections and this command enables the user to convert either of these into a USB to serial convertor so that the host can communicate directly with periferla connected to these ports.

This may be usful for updating or special configurations outside of the scope of the usual SMART Hopper to periferal protocols.

Command data format:

| byte | function | size |
|---|---|---|
| 0 | UART select (0 - SSP Uart, 1 - cctalk UART) | 1 |

Once this command is sent the device will respond with OK (0xF0) and from then all serial data via the USB will be routed to the periferal port directly.

To exit this mode, the host waits for at least 500ms since the last communication then sends byte array 0x55,0xAA,0xAA,0x55 waits for 500ms and then sends the array again. The device will then reset and communications will restore to normal.

## Packet examples

Command format (no parameters) for acknowledged request.

Host transmit: **7F 80 01 37 B2 02**
Slave Reply: **7F 80 01 F0 23 80**

<< back to index

| Command | Code hex | Code decimal |
|---|---|---|
| **Set Baud Rate** | 0x4D | 77 |

| Implemented on | Encryption Required |
|---|---|
| SMART HOPPER, SMART SYSTEM | **optional** |

## Description

This command has two data bytes to allow communication speed to be set on a device.

| byte | function | size |
|---|---|---|
| 0 | Required rate (0= 9600, 1=38400, 2= 15200) | 1 |
| 1 | Change persist (1=change will remain over reset, 0=rate sets to default after reset) | 1 |

The device will respond with 0xF0 at the old baud rate before changing. Please allow a minimum of 100 millseconds before attempting to communicate at the new baud rate.

## Packet examples

In this example, we want to set the speed to 38400 bd with but to reset to default (9600) on reset.

Host transmit: **7F 80 03 4D 01 00 E4 27**
Slave Reply: **7F 80 01 F0 23 80**

<< back to index

| Command | Code hex | Code decimal |
|---|---|---|
| **Ssp Set Encryption Key** | 0x60 | 96 |

| Implemented on | Encryption Required |
|---|---|
| SMART HOPPER, SMART SYSTEM | **yes** |

| Description |
|---|

A command to allow the host to change the fixed part of the eSSP key. The eight data bytes are a 64 bit number representing the fixed part of the key. This command must be encrypted.

| byte | function | size |
|---|---|---|
| 0 | new fixed key 64 bit, 8 byte | 8 |

| Packet examples |
|---|

Example to set new fixed key to 0x0123456701234567

Host transmit: **7F 80 09 60 67 45 23 01 67 45 23 01 BF 6F**
Slave Reply: **7F 80 01 F0 23 80**

<< back to index

| Command | Code hex | Code decimal |
|---|---|---|
| **Ssp Encryption Reset To Default** | 0x61 | 97 |

| Implemented on | Encryption Required |
|---|---|
| SMART HOPPER, SMART SYSTEM | **optional** |

| Description |
|---|

Resets the fixed encryption key to the device default. The device may have extra security requirements before it will accept this command (e.g. The Hopper must be empty) if these requirements are not met, the device will reply with Command Cannot be Processed. If successful, the device will reply OK, then reset. When it starts up the fixed key will be the default.

| Packet examples |
|---|

Command format (no parameters) for acknowledged request.

Host transmit: **7F 80 01 61 46 03**
Slave Reply: **7F 80 01 F0 23 80**

<< back to index

| Command | Code hex | Code decimal |
|---|---|---|
| **Get Real Time Clock Configuration** | 0x62 | 98 |

| Implemented on | Encryption Required |
|---|---|
| SMART SYSTEM | **optional** |

| Description |
|---|

Returns the configuration of the device Real Time Clock.

Response
The device responds with 1 data byte giving the configuration of the RTC. Data = 0, the RTC resets on power up and the date/time will need to be setup. Data = 1, the date/time is persistant after a power cycle.

| Packet examples |
|---|

In this example the device responds that the RTC does not hold it\'s settings after a power cycle.

Host transmit: **7F 80 01 62 4C 03**
Slave Reply: **7F 80 02 F0 00 3F A0**

<< back to index

| Command | Code hex | Code decimal |
|---|---|---|
| **Set Real Time Clock** | 0x64 | 100 |

| Implemented on | Encryption Required |
|---|---|
| SMART SYSTEM | **optional** |

## Description

Send six bytes of parameter data to set the system time and date.

Command data format:

| byte | function | size |
|---|---|---|
| 0 | Generic OK | 1 |
| 1 | Day of month (1-31) | 1 |
| 2 | Month of year (1-12) | 1 |
| 3 | Year (0-99) | 1 |
| 4 | Hour of day (0-23) | 1 |
| 5 | Minute of hour (0-59) | 1 |
| 6 | Second of minute (0-59) | 1 |

## Packet examples

Packet example for setting system time to 21st December 2012 10:22:30

Host transmit: **7F 80 07 64 15 0C 0C 0A 16 1E AF EC**
Slave Reply: **7F 80 01 F0 23 80**

<< back to index

| Command | Code hex | Code decimal |
|---|---|---|
| **Get Real Time Clock** | 0x63 | 99 |

| Implemented on | Encryption Required |
|---|---|
| SMART SYSTEM | **optional** |

| Description |
|---|

Gets the current system RTC date and time. Responds with 6 bytes of data.

Response format:

| byte | function | size |
|---|---|---|
| 0 | Generic OK | 1 |
| 1 | Day of month (1-31) | 1 |
| 2 | Month of year (1-12) | 1 |
| 3 | Year (0-99) | 1 |
| 4 | Hour of day (0-23) | 1 |
| 5 | Minute of hour (0-59) | 1 |
| 6 | Second of minute (0-59) | 1 |

| Packet examples |
|---|

In this example the system time is 21st December 2012 10:22:30

Host transmit: **7F 80 01 63 49 83**
Slave Reply: **7F 80 07 F0 15 0C 0C 0A 16 1E EC F1**

<< back to index

| Command | Code hex | Code decimal |
|---|---|---|
| **Set Cashbox Payout Limit** | 0x4E | 78 |

| Implemented on | Encryption Required |
|---|---|
| SMART HOPPER, SMART SYSTEM | **yes** |

| Description |
|---|

Allow the host to specify a maximum level of coins, by denomination, to be left in the hopper.

During any payout operation, if there are coins in the hopper in excess of the set levels, when they are encountered on the conveyor belt they will be sent to the cashbox (beneath the hopper).

This means that over time (and multiple payout operations) any excess coins will be sent to the cashbox and the desired level will be achieved.

It effectively allows the hopper to do the 'floating' for the host machine i.e. it is an auto float mechanism.

NB: If a coin route is changed from cashbox to payout and then back to cashbox then the level for this coin will be reset to 0 (any of the coins will then be sent to cashbox).

Command format.

| byte | function | size |
|---|---|---|
| 0 | The number of individual requests | 1 |
| 1 | The level limit to set | 2 |
| 3 | The denomination value | 4 |
| 7 | The denomination country code (3 byte ASCII) | 3 |
| **...** | **Repeat above block for each denomination required** | **...** |
| | | |

| Packet examples |
|---|

<< back to index

| Command | Code hex | Code decimal |
|---|---|---|
| **Coin Stir** | 0x5D | 93 |

| Implemented on | Encryption Required |
|---|---|
| SMART SYSTEM | **yes** |

| Description |
|---|

Mixes the coins by performs a rotation of the Coin Hopper Motor for a specifed time.

Command has 1 parameter, a byte value (1-255) giving the time in seconds for which to stir the coins.

| Packet examples |
|---|

Stir the coins for 5 seconds

Host transmit: **7F 80 02 5D 05 28 CE**
Slave Reply: **7F 80 01 F0 23 80**

<< back to index

| Command | Code hex | Code decimal |
|---|---|---|
| **Payout Amount By Denomination** | 0x39 | 57 |

| Implemented on | Encryption Required |
|---|---|
| SMART SYSTEM | **yes** |

| Description |
|---|

This command is similar to 'Payout Amount' but has two values in the payout which you can select the denominations for each.

| Packet examples |
|---|

<< back to index

| Event | Code hex | Code decimal |
|---|---|---|
| **Slave Reset** | 0xF1 | 241 |

| Implemented on |
|---|
| SMART HOPPER, SMART SYSTEM |

| Description |
|---|

An event gven when the device has been powered up or power cycled and has run through its reset process.

Protocol minimum version 4

| Type | Data size (bytes) | Repeat | Poll with Ack |
|---|---|---|---|
| **Status** | **0** | **no** | **no** |

| Packet examples |
|---|

Poll returns slave reset event

Host transmit: **7F 80 01 07 12 02**
Slave Reply: **7F 80 01 F1 26 00**

<< back to index

| Event | Code hex | Code decimal |
|---|---|---|
| **Disabled** | 0xE8 | 232 |

| Implemented on |
|---|
| SMART HOPPER, SMART SYSTEM |

## Description

A disabled event is given in response to a poll command when a device has been disabled by the host or by some other internal function of the device.

Protocol minimum version 4

| Type | Data size (bytes) | Repeat | Poll with Ack |
|---|---|---|---|
| **Status** | **0** | **no** | **no** |

## Packet examples

Response to poll showing disabled event

Host transmit: **7F 80 01 07 12 02**
Slave Reply: **7F 80 02 F0 E8 4F A2**

<< back to index

| Event | Code hex | Code decimal |
|---|---|---|
| **Fraud Attempt** | 0xE6 | 230 |

| Implemented on |
|---|
| SMART HOPPER, SMART SYSTEM |

## Description

The validator system has detected an attempt to mauipulate the coin/banknote in order to fool the system to register credits with no monies added.

Protocol minimum version 4

| Type | Data size (bytes) | Repeat | Poll with Ack |
|---|---|---|---|
| **Fraud** | **1** | **no** | **yes** |

Additional infomation

The data byte indicates the dataset channel of the banknote that is being tampeted with. A zero indicates that the channle is unknown.

## Packet examples

Poll response showing fraud attempt seen on channel 2

Host transmit: **7F 80 01 07 12 02**
Slave Reply: **7F 80 03 F0 E6 02 C0 7C**

<< back to index

| Event | Code hex | Code decimal |
|---|---|---|
| **Initialising** | 0xB6 | 182 |

| Implemented on |
|---|
| SMART HOPPER, SMART SYSTEM |

## Description

This event is given only when using the Poll with ACK command. It is given when the BNV is powered up and setting its sensors and mechanisms to be ready for Note acceptance. When the event response does not contain this event, the BNV is ready to be enabled and used.

Protocol minimum version 7

| Type | Data size (bytes) | Repeat | Poll with Ack |
|---|---|---|---|
| **Status** | **0** | **yes** | **yes** |

Additional infomation

**This event is only given when using the Poll With Ack command.**

## Packet examples

Host transmit: **7F 80 01 07 12 02**
Slave Reply: **7F 80 02 F0 B6 88 23**

<< back to index

| Event | Code hex | Code decimal |
|---|---|---|
| **Dispensing** | 0xDA | 218 |

| Implemented on |
|---|
| SMART HOPPER, SMART SYSTEM |

## Description

The device is in the process of paying out a requested value. The value paid at the poll is given in the event data.

### Protocol minimum version 4

| Type | Data size (bytes) | Repeat | Poll with Ack |
|---|---|---|---|
| **Status** | **4** | **yes** | **no** |

Additional infomation

$ byte data giving the amount dispensed up to the poll.

### Protocol minimum version 6

| Type | Data size (bytes) | Repeat | Poll with Ack |
|---|---|---|---|
| **Status** | **variable** | **yes** | **no** |

Additional infomation

An array of data giving the dispensed at the poll point for each of the countries supported in the dataset. The first byte gives the number of countries in the set the a block of data for each of the countries.

| byte | function | size |
|---|---|---|
| 0 | number of countries in set | 1 |
| 1 | value dispensed up to this point | 4 |
| 5 | country code | 3 |
| ... | repeat above block for each country in set | .. |

## Packet examples

Protocol version 5 poll response showing 12.50 dispensed at this point

Host transmit: **7F 80 01 07 12 02**
Slave Reply: **7F 80 05 F0 E2 04 00 00 F8 4A**

Protocol version 6 poll response showing 23.00 EUR and 12.00 GBP dispensed to this point

| | | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Host transmit: | **7F** | **80** | **01** | **07** | **12** | **02** | | | | | | | | | | | | | | | |
| Slave Reply: | **7F** | **80** | **10** | **F0** | **02** | **FC** | **08** | **00** | **00** | **45** | **55** | **52** | **B0** | **04** | **00** | **00** | **47** | **42** | **50** | **04** | **B3** |
| ascii: | | | | **.** | **.** | **.** | **.** | **.** | **.** | **E** | **U** | **R** | **.** | **.** | **.** | **.** | **G** | **B** | **P** | | |

<< back to index

| Event | Code hex | Code decimal |
|---|---|---|
| **Dispensed** | 0xD2 | 210 |

| Implemented on |
|---|
| SMART SYSTEM |

| Description |
|---|

Show the total value the device has dispensed in repsonse to a Dispense command.

Protocol minimum version 4

| Type | Data size (bytes) | Repeat | Poll with Ack |
|---|---|---|---|
| **Status** | **4** | **no** | **yes** |

| Additional infomation |
|---|

4 byte value showing total value dispensed.

Protocol minimum version 6

| Type | Data size (bytes) | Repeat | Poll with Ack |
|---|---|---|---|
| **Status** | **variable** | **no** | **yes** |

| Additional infomation |
|---|

An array of data giving the total dispensed for each of the countries supported in the dataset. The first byte gives the number of countries in the set the a block of data for each of the countries.

| byte | function | size |
|---|---|---|
| 0 | number of countries in set | 1 |
| 1 | value dispensed | 4 |
| 5 | country code | 3 |
| ... | repeat above block for each country in set | .. |

| Packet examples |
|---|

<< back to index

| Event | Code hex | Code decimal |
|---|---|---|
| **Coins Low** | 0xD3 | 211 |

| Implemented on |
|---|
| SMART HOPPER |

| Description |
|---|

| Packet examples |
|---|

<< back to index

| Event | Code hex | Code decimal |
|---|---|---|
| **Hopper Jammed** | 0xD5 | 213 |

| Implemented on |
|---|
| SMART HOPPER, SMART SYSTEM |

## Description

An event showing the hopper unit has jammed and giving the value paid/floated upto that jam.

On the smart payout this event is used when a jam occurs during a payout / float / empty operation.

### Protocol minimum version 5

| Type | Data size (bytes) | Repeat | Poll with Ack |
|---|---|---|---|
| **Error** | **4** | **yes** | **no** |

Additional infomation

4 bytes showing the value dispensed up to the jam point

### Protocol minimum version 6

| Type | Data size (bytes) | Repeat | Poll with Ack |
|---|---|---|---|
| **Error** | **variable** | **yes** | **no** |

Additional infomation

An array of data giving the dispensed/floated at the jammed point for each of the countries supported in the dataset. The first byte gives the number of countries in the set the a block of data for each of the countries.

| byte | function | size |
|---|---|---|
| 0 | number of countries in set | 1 |
| 1 | value dispensed/floated up to this point | 4 |
| 5 | country code | 3 |
| ... | repeat above block for each country in set | .. |

## Packet examples

Protocol version 5 poll response showing 2.30 paid up to the jam point

Host transmit: **7F 80 01 07 12 02**
Slave Reply: **7F 80 06 F0 D5 E6 00 00 00 49 DB**

<< back to index

| Event | Code hex | Code decimal |
|---|---|---|
| **Halted** | 0xD6 | 214 |

| Implemented on |
|---|
| SMART HOPPER, SMART SYSTEM |

| Description |
|---|

Triggered when payout is interrupted for some reason.

**Protocol Version 6 and earlier**

This event is given when:

- the host has requested a halt to the device.
- the payout is automatically cancelled (due to a jam/reverse validation fail/cashbox error etc.)

The value paid at the point of halting is given in the event data.

---

**Protocol Version 7 and later**

This event is given when:

- the host has requested a halt to the device.

The value paid at the point of halting is given in the event data.

Note: a different event 'Error During Payout' is generated when errors occur

Protocol minimum version 4

| Type | Data size (bytes) | Repeat | Poll with Ack |
|---|---|---|---|
| **Status** | **4** | **no** | **no** |

Additional infomation

4 byte showing the value paid up to the halt point

Protocol minimum version 6

| Type | Data size (bytes) | Repeat | Poll with Ack |
|---|---|---|---|
| **Status** | **variable** | **no** | **no** |

Additional infomation

An array of data giving the dispensed/floated at the poll point for each of the countries supported in the dataset. The first byte gives the number of countries in the set the a block of data for each of the countries.

| byte | function | size |
|---|---|---|
| 0 | number of countries in set | 1 |
| 1 | value dispensed/floated up to this point | 4 |
| 5 | country code | 3 |
| ... | repeat above block for each country in set | .. |

## Packet examples

Protocol version 6 poll response showing 15.30 GBP to the halt point

Host transmit: **7F 80 01 07 12 02**
Slave Reply: **7F 80 0A F0 D6 01 FA 05 00 00 45 55 52 4D 49**
ascii: **. . . . . . . E U R**

<< back to index

| Event | Code hex | Code decimal |
|---|---|---|
| **Floating** | 0xD7 | 215 |

| Implemented on |
|---|
| SMART HOPPER, SMART SYSTEM |

## Description

Event showing the amount of cash floated up to the poll point

### Protocol minimum version 4

| Type | Data size (bytes) | Repeat | Poll with Ack |
|---|---|---|---|
| **Status** | **4** | **yes** | **no** |

Additional infomation

4 bytes showing the value floated to the cashbox up to the poll

### Protocol minimum version 6

| Type | Data size (bytes) | Repeat | Poll with Ack |
|---|---|---|---|
| **Status** | **variable** | **yes** | **no** |

Additional infomation

An array of data giving the floated value at the poll point for each of the countries supported in the dataset. The first byte gives the number of countries in the set the a block of data for each of the countries.

| byte | function | size |
|---|---|---|
| 0 | number of countries in set | 1 |
| 1 | value floated to this point | 4 |
| 5 | country code | 3 |
| ... | repeat above block for each country in set | .. |

## Packet examples

Protocol version 5 poll response showing 45.00 floated

Host transmit: **7F 80 01 07 12 02**

Slave Reply: **7F 80 05 F0 94 11 00 00 E8 F3**

<< back to index

| Event | Code hex | Code decimal |
|---|---|---|
| **Floated** | 0xD8 | 216 |

| Implemented on |
|---|
| SMART HOPPER, SMART SYSTEM |

## Description

Event given at the end of the floating process which will display the amount actually floated.

### Protocol minimum version 4

| Type | Data size (bytes) | Repeat | Poll with Ack |
|---|---|---|---|
| **Status** | **4** | **no** | **yes** |

Additional infomation

4 Bytes showing the amount floated

### Protocol minimum version 6

| Type | Data size (bytes) | Repeat | Poll with Ack |
|---|---|---|---|
| **Status** | **variable** | **no** | **yes** |

Additional infomation

An array of data giving the floated value at the end of the process for each of the countries supported in the dataset. The first byte gives the number of countries in the set the a block of data for each of the countries.

| byte | function | size |
|---|---|---|
| 0 | number of countries in set | 1 |
| 1 | value floated | 4 |
| 5 | country code | 3 |
| ... | repeat above block for each country in set | .. |

## Packet examples

Protocol version 6 poll response showing a floated value of 20.50 EUR

Host transmit: **7F 80 01 07 12 02**
Slave Reply: **7F 80 0A F0 D8 01 02 08 00 00 45 55 52 81 C0**
ascii: **. . . . . . . E U R**

<< back to index

| Event | Code hex | Code decimal |
|---|---|---|
| **Timeout** | 0xD9 | 217 |

| Implemented on |
|---|
| SMART HOPPER, SMART SYSTEM |

| Description |
|---|

The device has been unable to complete a request. The value paid up until the time-out point is given in the event data.

Protocol minimum version 4

| Type | Data size (bytes) | Repeat | Poll with Ack |
|---|---|---|---|
| **Status** | **4** | **no** | **yes** |

Additional infomation

4 bytes showing the value dispensed or floated to that point.

Protocol minimum version 6

| Type | Data size (bytes) | Repeat | Poll with Ack |
|---|---|---|---|
| **Status** | **variable** | **no** | **yes** |

Additional infomation

An array of data giving the dispensed/floated at the poll point for each of the countries supported in the dataset. The first byte gives the number of countries in the set the a block of data for each of the countries.

| byte | function | size |
|---|---|---|
| 0 | number of countries in set | 1 |
| 1 | value dispensed/floated up to this point | 4 |
| 5 | country code | 3 |
| ... | repeat above block for each country in set | .. |

| Packet examples |
|---|

<< back to index

| Event | Code hex | Code decimal |
|---|---|---|
| **Incomplete Payout** | 0xDC | 220 |

| Implemented on |
|---|
| SMART HOPPER, SMART SYSTEM |

## Description

The device has detected a discrepancy on power-up that the last payout request was interrupted (possibly due to a power failure). The amounts of the value paid and requested are given in the event data.

### Protocol minimum version 4

| Type | Data size (bytes) | Repeat | Poll with Ack |
|---|---|---|---|
| **Pay-out** | **8** | **no** | **yes** |

Additional infomation

Eight data bytes showing the value dispensed and the value requested.

### Protocol minimum version 6

| Type | Data size (bytes) | Repeat | Poll with Ack |
|---|---|---|---|
| **Pay-out** | **variable** | **no** | **yes** |

Additional infomation

An array of data giving the value dispensed and the original value requested before the power down for each of the countries supported in the dataset. The first byte gives the number of countries in the set then a block of data for each of the countries (see table below).

| byte | function | size |
|---|---|---|
| 0 | number of countries in set | 1 |
| 1 | value dispensed | 4 |
| 5 | value requested | 4 |
| 9 | country code (ASCII) | 3 |
| ... | repeat above block for each country in set | .. |

## Packet examples

Protocol version 5 poll response showing 25.20 paid out of request for 50.00

Host transmit: **7F 80 01 07 12 02**
Slave Reply: **7F 80 09 F0 D8 09 00 00 58 0D 00 00 3B C9**

Protocol version 6 poll response showing 23.00 EUR paid out of a request to payout 50.00 EUR

| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Host transmit: | **7F** | **80** | **01** | **07** | **12** | **02** | | | | | | | | | | |
| Slave Reply: | **7F** | **80** | **0D** | **F0** | **01** | **FC** | **08** | **00** | **00** | **88** | **13** | **00** | **00** | **45** | **55** | **52** | **C3** | **E5** |
| ascii: | | | | | | | **.** | **.** | **.** | **.** | **.** | **.** | **.** | **.** | **.** | **E** | **U** | **R** |

<< back to index

| Event | Code hex | Code decimal |
|---|---|---|
| **Incomplete Float** | 0xDD | 221 |

| Implemented on |
|---|
| SMART HOPPER, SMART SYSTEM |

| Description |
|---|

The device has detected a discrepancy on power-up that the last float request was interrupted (possibly due to a power failure). The amounts of the value paid and requested are given in the event data.

Protocol minimum version 5

| Type | Data size (bytes) | Repeat | Poll with Ack |
|---|---|---|---|
| **Pay-out** | **8** | **no** | **yes** |

Additional infomation

8 data bytes giving the value of floated and the float value requested before the power was interrupted

Protocol minimum version 6

| Type | Data size (bytes) | Repeat | Poll with Ack |
|---|---|---|---|
| **Pay-out** | **variable** | **no** | **yes** |

Additional infomation

An array of data giving the value floated and the original value requested before the power down for each of the countries supported in the dataset. The first byte gives the number of countries in the set then a block of data for each of the countries (see table below).

| byte | function | size |
|---|---|---|
| 0 | number of countries in set | 1 |
| 1 | value floated | 4 |
| 5 | value requested | 4 |
| 9 | country code (ASCII) | 3 |
| ... | repeat above block for each country in set | .. |

| Packet examples |
|---|

<< back to index

| Event | Code hex | Code decimal |
|---|---|---|
| **Cashbox Paid** | 0xDE | 222 |

| Implemented on |
|---|
| SMART HOPPER, SMART SYSTEM |

## Description

Coin values have been detected and paid to the cashbox since the last poll.

### Protocol minimum version 5

| Type | Data size (bytes) | Repeat | Poll with Ack |
|---|---|---|---|
| **Status** | **4** | **no** | **no** |

Additional infomation

Data bytes show the coin value paid

### Protocol minimum version 6

| Type | Data size (bytes) | Repeat | Poll with Ack |
|---|---|---|---|
| **Status** | **variable** | **no** | **no** |

Additional infomation

Data bytes give country codes and values for each of the currencies in the dataset:

| byte | function | size |
|---|---|---|
| 0 | number of countries in set | 1 |
| 1 | value dispensed | 4 |
| 5 | country code | 3 |
| ... | repeat above block for each country in set | .. |

## Packet examples

Protocol version 5 poll response showing 2.00 (200 c) coin paid to cashbox

Host transmit: **7F 90 01 07 51 83**
Slave Reply: **7F 90 06 F0 DE C8 00 00 00 68 00**

Protocol version 6 poll response showing 5.30 GBP adn 0.20 EUR paid to cashbox

Host transmit: **7F 90 01 07 51 83**
Slave Reply: **7F 90 11 F0 DE 02 12 02 00 00 47 42 50 14 00 00 00 45 55 52 3A 50**
ascii: . . . . . . . **G B P** . . . . **E U R**

<< back to index

| Event | Code hex | Code decimal |
|---|---|---|
| **Coin Credit** | 0xDF | 223 |

| Implemented on |
|---|
| SMART HOPPER |

| Description |
|---|

A coin has been detected as added to the system. This would be usually via the seperate coin mech attached to the system port.

Protocol minimum version 5

| Type | Data size (bytes) | Repeat | Poll with Ack |
|---|---|---|---|
| **Status** | **4** | **no** | **no** |

| Additional infomation |
|---|

Data gives 4 byte value of the coin added

Protocol minimum version 6

| Type | Data size (bytes) | Repeat | Poll with Ack |
|---|---|---|---|
| **Status** | **7** | **no** | **no** |

| Additional infomation |
|---|

Data bytes give 4 byte coin value and 3 byte ASCII country code of the coin added

| Packet examples |
|---|

Protocol version 5 poll response showing 1.00 (100 c) coin added

Host transmit: **7F 90 01 07 51 83**
Slave Reply: **7F 90 05 F0 64 00 00 00 97 A3**

Protocol version 6 poll response showing 5.00 GBP coin added

Host transmit: **7F 90 01 07 51 83**
Slave Reply: **7F 90 09 F0 DF F4 01 00 00 47 42 50 89 0F**
ascii: **. . . . . . G B P**

<< back to index

| Event | Code hex | Code decimal |
|---|---|---|
| **Coin Mech Jammed** | 0xC4 | 196 |

| Implemented on |
|---|
| SMART HOPPER, SMART SYSTEM |

| Description |
|---|

The attached coin mechanism has been detected as having a jam.

Protocol minimum version 5

| Type | Data size (bytes) | Repeat | Poll with Ack |
|---|---|---|---|
| **Status** | **0** | **no** | **no** |

| Packet examples |
|---|

Poll response showing coin mech jam

Host transmit: **7F 90 01 07 51 83**
Slave Reply: **7F 90 02 F0 C4 A2 62**

<< back to index

| Event | Code hex | Code decimal |
|---|---|---|
| **Coin Mech Return Active** | 0xC5 | 197 |

| Implemented on |
|---|
| SMART HOPPER, SMART SYSTEM |

| Description |
|---|

The attached coin mechanism has been detected as having it's reject or return button pressed.

Protocol minimum version 5

| Type | Data size (bytes) | Repeat | Poll with Ack |
|---|---|---|---|
| **Status** | **0** | **no** | **no** |

| Packet examples |
|---|

<< back to index

| Event | Code hex | Code decimal |
|---|---|---|
| **Emptying** | 0xC2 | 194 |

| Implemented on |
|---|
| SMART HOPPER, SMART SYSTEM |

| Description |
|---|

The device is currently performing is empty operation following an Empty command request.

Protocol minimum version 5

| Type | Data size (bytes) | Repeat | Poll with Ack |
|---|---|---|---|
| **Status** | **0** | **yes** | **no** |

| Packet examples |
|---|

Poll response showing device emptying

Host transmit: **7F 80 01 07 12 02**
Slave Reply: **7F 80 02 F0 C2 B0 22**

<< back to index

| Event | Code hex | Code decimal |
|---|---|---|
| **Emptied** | 0xC3 | 195 |

| Implemented on |
|---|
| SMART HOPPER, SMART SYSTEM |

| Description |
|---|

The device has completed it's empty operation in response to the Empty command.

Protocol minimum version 5

| Type | Data size (bytes) | Repeat | Poll with Ack |
|---|---|---|---|
| **Status** | **0** | **no** | **no** |

| Packet examples |
|---|

Poll response showing device emptied

Host transmit: **7F 80 01 07 12 02**
Slave Reply: **7F 80 02 F0 C3 B5 A2**

<< back to index

| Event | Code hex | Code decimal |
|---|---|---|
| **Smart Emptying** | 0xB3 | 179 |

| Implemented on |
|---|
| SMART HOPPER, SMART SYSTEM |

| Description |
|---|

The device is in the process of carrying out its Smart Empty command from the host. The value emptied at the poll point is given in the event data

Protocol minimum version 5

| Type | Data size (bytes) | Repeat | Poll with Ack |
|---|---|---|---|
| **Status** | **4** | **yes** | **no** |

Additional infomation

4 byte integer showing the value emptied so far.

Protocol minimum version 6

| Type | Data size (bytes) | Repeat | Poll with Ack |
|---|---|---|---|
| **Status** | **variable** | **yes** | **no** |

Additional infomation

Data bytes give country codes and values for each of the currencies in the dataset:

| byte | function | size |
|---|---|---|
| 0 | number of countries in set | 1 |
| 1 | value dispensed | 4 |
| 5 | country code | 3 |
| ... | repeat above block for each country in set | .. |

| Packet examples |
|---|

A device has emptied 22.60 EUR up to this poll with protocol version 5

Host transmit: **7F 80 01 07 12 02**
Slave Reply: **7F 80 07 F0 B3 01 D4 08 00 00 53 F7**

A device has emptied 22.60 EUR up to this poll with protocol version 6

| | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Host transmit: | **7F** | **80** | **01** | **07** | **12** | **02** | | | | | | | | |
| Slave Reply: | **7F** | **80** | **0A** | **F0** | **B3** | **01** | **D4** | **08** | **00** | **00** | **45** | **55** | **52** | **44** | **F6** |
| ascii: | | | | **.** | **.** | **.** | **.** | **.** | **.** | **.** | **E** | **U** | **R** | | |

<< back to index

| Event | Code hex | Code decimal |
|---|---|---|
| **Smart Emptied** | 0xB4 | 180 |

| Implemented on |
|---|
| SMART HOPPER, SMART SYSTEM |

| Description |
|---|

The device has completed its Smart Empty command. The total amount emptied is given in the event data.

Protocol minimum version 5

| Type | Data size (bytes) | Repeat | Poll with Ack |
|---|---|---|---|
| **Status** | **4** | **no** | **yes** |

| Additional infomation |
|---|

4 byte interger showing the total value emptied in this session.

Protocol minimum version 6

| Type | Data size (bytes) | Repeat | Poll with Ack |
|---|---|---|---|
| **Status** | **variable** | **no** | **yes** |

| Additional infomation |
|---|

Data bytes give country codes and values for each of the currencies in the dataset of the total amount emptied.

| byte | function | size |
|---|---|---|
| 0 | number of countries in set | 1 |
| 1 | value dispensed | 4 |
| 5 | country code | 3 |
| ... | repeat above block for each country in set | .. |

| Packet examples |
|---|

<< back to index

| Event | Code hex | Code decimal |
|---|---|---|
| **Calibration Failed** | 0x83 | 131 |

| Implemented on |
|---|
| SMART HOPPER, SMART SYSTEM |

| Description |
|---|

During the devices normal re-calibration process, an error has been detected which indicates a sensor failure or out-of-range issue. This usually indicate a hardware failure and the device should be taken out of service until the cause is found.

Protocol minimum version 7

| Type | Data size (bytes) | Repeat | Poll with Ack |
|---|---|---|---|
| **Error** | **1** | **no** | **no** |

Additional infomation

A data byte error reason is given detailed in the table below.

| Error | Code |
|---|---|
| | 0 |
| Payout flap sensor | 1 |
| Exit sensor | 2 |
| Coil 1 sensor | 3 |
| Coil 2 sensor | 4 |
| Unit not initialised | 5 |
| Checksum error | 6 |
| Recalibration by command required (obsolete) | 7 |
| Motor opto slot error | 8,9 |
| Exit sensor error 2 | 10 |

| Packet examples |
|---|

The example below shows a calibration fail due to an issue with coil 1.

Host transmit: **7F 80 01 07 12 02**
Slave Reply: **7F 80 03 F0 83 03 C0 22**

<< back to index

| Event | Code hex | Code decimal |
|---|---|---|
| **Device Full** | 0xCF | 207 |

| Implemented on |
|---|
| SMART SYSTEM |

| Description |
|---|

The device has detected that it is full of coins/banknotes and no more can be added.

Protocol minimum version 5

| Type | Data size (bytes) | Repeat | Poll with Ack |
|---|---|---|---|
| **Status** | **0** | **yes** | **no** |

| Packet examples |
|---|

<< back to index

| Event | Code hex | Code decimal |
|---|---|---|
| **Coin Mech Error** | 0xB7 | 183 |

| Implemented on |
|---|
| SMART HOPPER, SMART SYSTEM |

| Description |
|---|

**This event will only be gererated if the Coin Mech Options command has been sent to the device with data bit set to enable error events.**

The data byte given with this event indicates the error type.

Code
Error
Description
1
Reject coin
A coin was inserted which did not match any of the programmed types. The coin is returned to the customer and no credit is given.
2
Inhibited coin
A coin was inserted which did match a programmed window type but was prevented from accepting by the inhibit register. The inhibit register can be controlled serially but may also be linked to external DIL switches.
3
Multiple window
A coin was inserted which matched more than one enabled window type. This coin was rejected as the credit code was indeterminate.
4
Wake-up timeout
A coin acceptor fitted with a wake-up sensor picked up a coin entering the acceptor but it was not seen subsequently in the validation area. Possible coin jam.
5
Validation timeout
A coin was detected entering the validation area but failed to leave it. Possible coin jam.
6
Credit sensor timeout
A coin was validated as true but never made it to the post-gate credit sensor. Possible coin jam.
7
Sorter opto timeout
A coin was sent into the sorter / diverter but was not seen coming out. Possible coin jam.
8
2nd close coin error
A coin was inserted too close to the one in front. One or both coins will have rejected.
9
Accept gate not ready
A coin was inserted while the accept gate for the coin in front was still operating. Coins have been inserted too quickly.
10
Credit sensor not ready
A coin was still over the credit sensor when another coin was ready to accept. Coins have been inserted too quickly.
11

Sorter not ready
A coin was inserted while the sorter flaps for the coin in front were still operating. Coins have been inserted too quickly.
12
Reject coin not cleared
A coin was inserted before a previously rejected coin had time to clear the coin acceptor. Coins have been inserted too quickly.
13
Validation sensor not ready
The validator inductive sensors were not ready for coin validation. Possible fault developing.
14
Credit sensor blocked
There is a permanent blockage at the credit sensor. The coin acceptor will not accept any more coins.
15
Sorter opto blocked
There is a permanent blockage at the sorter exit sensor. The coin acceptor will not accept any more coins.
16
Credit sequence error
A coin or object was detected going backwards through a directional credit sensor. Possible fraud attempt.
17
Coin going backwards
A coin was detected going backwards through the coin acceptor. Possible fraud attempt.
18
Coin too fast ( over credit sensor )
A coin was timed going through the credit sensor and was too fast. Possible fraud attempt.
19
Coin too slow ( over credit sensor )
20
C.O.S. mechanism activated
( coin-on-string )
A specific sensor for detecting a ‘coin on string’ was activated. Possible fraud attempt.
21
DCE opto timeout
A coin acceptor fitted with a Dual Coin Entry chute saw a coin or token which was not seen subsequently in the validation area. Possible coin jam.
22
DCE opto not seen
A coin acceptor fitted with a Dual Coin Entry chute saw a coin which was not seen previously by the chute sensor. Possible fraud attempt.
23
Credit sensor reached too early
A coin was timed from the end of the validation area to the post-gate credit sensor. It arrived too early. Possible fraud attempt.
24
Reject coin ( repeated sequential trip )
A coin was rejected N times in succession with no intervening true coins. Statistically unlikely if N greater than or equal to 5. Possible fraud attempt.
25
Reject slug
A coin was rejected but was identified as a known slug type - this may be a pre-programmed fraud coin or a known fraud material.
26
Reject sensor blocked
There is a permanent blockage at the reject sensor. The coin acceptor will not accept any more coins. Not all coin acceptors have a reject sensor.
27
Games overload
Totaliser mode : A game value was set too low - possibly zero. This is a product configuration error.
28
Max. coin meter pulses exceeded
Totaliser mode : A meter value was set too low - possibly zero. This is a product

configuration error.
29
Accept gate open not closed
The accept gate was forced open when it should have been closed.
30
Accept gate closed not open
The accept gate did not open when the solenoid was driven.
31
Manifold opto timeout
A coin was sent into the manifold module ( coin diverter ) but was not seen coming out. Possible coin jam.
32
Manifold opto blocked
There is a permanent blockage at the manifold module sensor ( coin diverter ). The coin acceptor will not accept any more coins.
128
Inhibited coin ( Type 1 )
A true coin ( type 1, coin in position 1 ) was inserted but was prevented from accepting by the inhibit register.
…
Inhibited coin ( Type n )
A true coin ( type n, coin in position n ) was inserted but was prevented from accepting by the inhibit register.
159
Inhibited coin ( Type 32 )
A true coin ( type 32, coin in position 32 ) was inserted but was prevented from accepting by the inhibit register.
253
Data block request ( note α )
A 'not yet used' mechanism for a coin acceptor to request attention from the host machine. Perhaps it needs some data from the host machine or another peripheral.
254
Coin return mechanism activated
( Flight deck open )
An attempt to clear a coin jam by opening the flight deck was detected. The coin acceptor cannot operate until the flight deck is closed.
255
Unspecified alarm code
Any alarm code which does not fit into the above categories.

Protocol minimum version 7

| Type | Data size (bytes) | Repeat | Poll with Ack |
|---|---|---|---|
| **Error** | **1** | **no** | **no** |

## Packet examples

A coin error: too slow detected

Host transmit: **7F 80 01 07 12 02**
Slave Reply: **7F 80 03 F0 B7 14 B1 1A**

<< back to index

| Event | Code hex | Code decimal |
|---|---|---|
| **Attached Coin Mech Disabled** | 0xBD | 189 |

| Implemented on |
|---|
| SMART HOPPER, SMART SYSTEM |

| Description |
|---|

The device seperate coin mechanism attached to this device has been disabled.

Protocol minimum version 6

| Type | Data size (bytes) | Repeat | Poll with Ack |
|---|---|---|---|
| **Status** | **0** | **no** | **no** |

| Packet examples |
|---|

Poll response showing coin mech disabled

Host transmit: **7F 90 01 07 51 83**
Slave Reply: **7F 90 02 F0 BD B7 E3**

<< back to index

| Event | Code hex | Code decimal |
|---|---|---|
| **Attached Coin Mech Enabled** | 0xBE | 190 |

| Implemented on |
|---|
| SMART HOPPER, SMART SYSTEM |

| Description |
|---|

The seperate coin mechanism attached to this device has been enabled.

Protocol minimum version 6

| Type | Data size (bytes) | Repeat | Poll with Ack |
|---|---|---|---|
| **Status** | **0** | **no** | **no** |

| Packet examples |
|---|

Poll response showing coin mech enabled

Host transmit: **7F 90 01 07 51 83**
Slave Reply: **7F 90 02 F0 BE BD E3**

<< back to index

| Event | Code hex | Code decimal |
|---|---|---|
| **Value Added** | 0xBF | 191 |

| Implemented on |
|---|
| SMART SYSTEM |

| Description |
|---|

An event giving the cumulative value of currency detected as added to the system since the last poll.

Protocol minimum version 7

| Type | Data size (bytes) | Repeat | Poll with Ack |
|---|---|---|---|
| **Pay-in** | **variable** | **no** | **yes** |

Additional infomation

Data bytes give country codes and values for each of the currencies where value has been added

| byte | function | size |
|---|---|---|
| 0 | Generic OK | 1 |
| 1 | Event code | 1 |
| 2 | number of countries in data | 1 |
| 3 | value added (4 byte integer) | 4 |
| 7 | country code (3 Byte ASCII) | 3 |
| ... | repeat above block for each country data | .. |

| Packet examples |
|---|

5.50 EUR has been added since the last poll

Host transmit: **7F 80 01 07 12 02**
Slave Reply: **7F 80 0A F0 BF 01 26 02 00 00 45 55 52 ED 91**

2.20 EUR and 3.60 GBP have been added since the last poll

Host transmit: **7F 80 01 07 12 02**
Slave Reply: **7F 80 11 F0 BF 02 DC 00 00 00 45 55 52 68 01 00 00 47 42 50 D1 05**

<< back to index

| Event | Code hex | Code decimal |
|---|---|---|
| **Pay-in Active** | 0xC1 | 193 |

| Implemented on |
|---|
| SMART SYSTEM |

| Description |
|---|

The pay-in function of the system is active.

Protocol minimum version 7

| Type | Data size (bytes) | Repeat | Poll with Ack |
|---|---|---|---|
| **Status** | **0** | **yes** | **no** |

| Packet examples |
|---|

Poll response showing pay-in function is active

Host transmit: **7F 90 01 07 51 83**
Slave Reply: **7F 90 02 F0 C1 BC 62**