INNOVATIVE TECHNOLOGY LTD

SSP

Communications Protocol Manual

GA138

Issue version 2.0.2

INTELIGENCE IN VALIDATION

Contents

Version History

Version	Issue date	Notes
0.0.31	26 May 1998	Previous versions see issue 31 and under for details
2.0.0	12 Oct 2012	New format issue.
2.0.1	22 Oct 2012	• Correction to Enable Payout Device command. • Added Communication Pass Through command details • Added Set Fixed Encryption Key command details • Added Reset Fixed Encryption Key command details • Added Set Baud Rate command details
2.0.2	06 Nov 2012	• Various typo corrections, gramatical and format improvements. • Get All Levels command now shows correct OK response. • Added command Get Build Revision • Added NV9usb,NV11 to Get Counters and Reset Counters commands. • Added NV11 to Poll command devices supported.

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

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 are all 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 transmission with 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 take place.

It is recommended that the encryption system be used to prevent fraud through bus monitoring and tapping. This is compulsory for all payout devices.

Commands are currently provided for coin acceptors, note acceptors and coin hoppers. 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

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 a single 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.

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
Slave ID	Single byte used to identify the address of the slave the packet is intended for
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 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 its 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 for a 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.

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 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 is received and decrypted.
eDATA	Commands or data to be transferred
PACKING	Random data to make the length of the length +count + data + packing + CRCL + CRCH to be a multiple of 16 bytes
CRCL, CRCH	Low and high byte of a forward CRC-16 algorithm using the polynomial (X16 + X15 + X2 +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 and each time a received packet is successfully decrypted. 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.

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
3	Generate prime number MODULUS	
4	Use command Set Modulus to send to slave	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

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.
PARAMETER OUT OF RANGE	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.

Banknote Validator

A Banknote Validator is a device which will scan, validate and stack a banknote it detects as valid or reject it from the front if not valid. Some banknote validators can be transformed into payout devices by the addition of a pay-out unit.

All ITL™ Banknote validators support the SSP protocol described here.

Table of current ITL™ Banknote validators

Device name	Note Stacker Fitted	Can fit payout
bv20	no	no
bv50	yes	no
bv100	yes	no
nv9	yes	no
nv10	no	no
nv9usb	yes	yes
nv200	yes	yes

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.

SMART Payout

The Smart Payout is an extension of a banknote validator, all commands are sent to the validator using its address (0x00). Information on the types of note that can be handled is obtained from the standard note validator commands.

Note that payout values are in terms of the penny value of that currency. So for 5.00, the value sent and returned by the payout would be 500.

The host simply has to tell the unit the value it wishes to dispense. The unit will manage which notes are stored to be used for payout and their location to minimise the payout time, and which notes, of the type enable for storage, are sent to the stacker. This is the recommended mode of operation.

Command Index

NV9USB implementation list

Reset	001	Set Channel Inhibits	002
Display On	003	Display Off	004
Setup Request	005	Host Protocol Version	006
Poll	007	Reject Banknote	008
Disable	009	Enable	010
Get Serial Number	012	Unit Data	013
Channel Value Request	014	Channel Security Data	015
Channel Re-teach Data	016	Sync	017
Last Reject Code	023	Hold	024
Get Firmware Version	032	Get Dataset Version	033
Get Bar Code Reader Configuration	035	Set Bar Code Configuration	036
Get Bar Code Inhibit Status	037	Set Bar Code Inhibit Status	038
Get Bar Code Data	039	Set Generator	074
Set Modulus	075	Request Key Exchange	076
Poll With ACK	086	Event ACK	087
Get Counters	088	Reset Counters	089

NV10USB implementation list

Reset	001	Set Channel Inhibits	002
Display On	003	Display Off	004
Setup Request	005	Host Protocol Version	006
Poll	007	Reject Banknote	008
Disable	009	Enable	010
Get Serial Number	012	Unit Data	013
Channel Value Request	014	Channel Security Data	015
Channel Re-teach Data	016	Sync	017
Last Reject Code	023	Hold	024
Get Firmware Version	032	Get Dataset Version	033
Set Generator	074	Set Modulus	075
Request Key Exchange	076	Poll With ACK	086
Event ACK	087		

BV20 implementation list

Reset	001	Set Channel Inhibits	002
Setup Request	005	Host Protocol Version	006
Poll	007	Reject Banknote	008
Disable	009	Enable	010
Get Serial Number	012	Unit Data	013
Channel Value Request	014	Channel Security Data	015
Channel Re-teach Data	016	Sync	017
Last Reject Code	023	Hold	024
Get Firmware Version	032	Get Dataset Version	033
Set Generator	074	Set Modulus	075
Request Key Exchange	076	Poll With ACK	086
Event ACK	087		

BV50 implementation list

Reset	001	Set Channel Inhibits	002
Setup Request	005	Host Protocol Version	006
Poll	007	Reject Banknote	008
Disable	009	Enable	010
Get Serial Number	012	Unit Data	013
Channel Value Request	014	Channel Security Data	015
Channel Re-teach Data	016	Sync	017
Last Reject Code	023	Hold	024
Get Firmware Version	032	Get Dataset Version	033
Set Generator	074	Set Modulus	075
Request Key Exchange	076	Poll With ACK	086
Event ACK	087		

BV100 implementation list

Reset	001	Set Channel Inhibits	002
Setup Request	005	Host Protocol Version	006
Poll	007	Reject Banknote	008
Disable	009	Enable	010
Get Serial Number	012	Unit Data	013
Channel Value Request	014	Channel Security Data	015
Channel Re-teach Data	016	Sync	017
Last Reject Code	023	Hold	024
Get Firmware Version	032	Get Dataset Version	033
Set Generator	074	Set Modulus	075
Request Key Exchange	076	Poll With ACK	086
Event ACK	087		

NV200 implementation list

Reset	001	Set Channel Inhibits	002
Display On	003	Display Off	004
Setup Request	005	Host Protocol Version	006
Poll	007	Reject Banknote	008
Disable	009	Enable	010
Get Serial Number	012	Unit Data	013
Channel Value Request	014	Channel Security Data	015
Channel Re-teach Data	016	Sync	017
Last Reject Code	023	Hold	024
Get Firmware Version	032	Get Dataset Version	033
Get Bar Code Reader Configuration	035	Set Bar Code Configuration	036
Get Bar Code Inhibit Status	037	Set Bar Code Inhibit Status	038
Get Bar Code Data	039	Set Generator	074
Set Modulus	075	Request Key Exchange	076
Get Build Revision	079	Configure Bezel	084
Poll With ACK	086	Event ACK	087

SMART HOPPER implementation list

SMART PAYOUT implementation list

SMART SYSTEM implementation list

NV11 implementation list

Reset	001	Set Channel Inhibits	002
Display On	003	Display Off	004
Setup Request	005	Host Protocol Version	006
Poll	007	Reject Banknote	008
Disable	009	Enable	010
Get Serial Number	012	Unit Data	013
Channel Value Request	014	Channel Security Data	015
Channel Re-teach Data	016	Sync	017
Last Reject Code	023	Hold	024
Get Firmware Version	032	Get Dataset Version	033
Set Denomination Route	059	Get Denomination Route	060
Empty All	063	Get Note Positions	065
Payout Note	066	Stack Note	067
Set Value Reporting Type	069	Set Generator	074
Set Modulus	075	Request Key Exchange	076
Set Baud Rate	077	Get Build Revision	079
Smart Empty	082	Cashbox Payout Operation Data	083
Poll With ACK	086	Event ACK	087
Get Counters	088	Reset Counters	089
Disable Payout Device	091	Enable Payout Device	092
Set Fixed Encryption Key	096	Reset Fixed Encryption Key	097

SSP Command List

Complete listing of all SSP commands

back to index

Command name	Code dec	Code hex
Reset Fixed Encryption Key	**097**	**0x61**

Supported on devices:

SMART Hopper	SMART Payout	SMART System	nv11

Encryption Required
no

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.

Parameters

This command has no parameters

Command packet example:

dec

127	128	001	097	070	003

hex

7F	80	01	61	46	03

Response

Devcie responds OK for success.

back to index

Command name	Code dec	Code hex
Set Fixed Encryption Key	**096**	**0x60**

Supported on devices:

SMART Hopper	SMART Payout	SMART System	nv11

Encryption Required
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.

Eight byte for the new key data.

dec

127	128	009	096	034	087	123	233	001	432	087	023	097	143

hex

7F	80	09	60	22	57	7B	E9	01	1B0	57	17	61	8F

Response

Device responds ok for success.

dec

127	128	001	240	035	128

hex

7F	80	01	F0	23	80

back to index

Command name	Code dec	Code hex
Enable Payout Device	**092**	**0x5C**

Supported on devices:

SMART Payout	nv11

Encryption Required
no

Description

A command to enable the attached payout device for storing/paying out notes. A successful enable will return OK, If there is a problem the reply will be generic response COMMAND_CANNOT_BE_PROCESSED, followed by an error code.

For nv11 devices, this command uses an addition data byte, a bit register allows some options to be set.

Bit	function
0	GIVE_VALUE_ON_STORED. Set to 1 to enable the value of the note stored to be given with the Note Stored event
1	NO_HOLD_NOTE_ON_PAYOUT. Set to 1 to enable the function of fully rejecting the dispensed banknote rather then holding it in the bezel.
2:7	Unused - set to 0

In this example we want to enabled the nv11 note float with the options of giving the value with Stored event and not holding the note on dispense.

dec

127	128	002	092	003	063	072

hex

7F	80	02	5C	03	3F	48

For SMART Payout devices with firmware greater or equal to 4.16, this command uses an addition data byte. A bit register allows some options to be set.

Bit	function
0	REQUIRE_FULL_STARTUP. If set to 1, the Smart Payout will return busy until it has fully completed the startup procedure
1	OPTIMISE_FOR_PAYIN_SPEED. If set to 1 The Smart Payout will always move towards an empty slot when idle to try and ensure the shortest pay in speed possible.
2:7	Unused - set to 0

In this example we want to enable all options for a SMART Payout device.

dec

127	128	002	092	003	063	072

hex

7F	80	02	5C	03	3F	48

Response

The device responds with OK for successful enable.

dec

127	128	001	240	035	128

hex

7F	80	01	F0	23	80

The device responds with COMMAND CANNOT BE PROCESSED and an error byte for failure to enable. In this example an invalid currency miss-match was detected between the validator and connected payout.

dec

127	128	002	245	002	048	062

hex

7F	80	02	F5	02	30	3E

Payout Enable Error codes

Error reason	Error code
No device connected	1
Invalid currency detected	2
Device busy	3
Empty only (Note float only)	4

Device error	5

back to index

Command name	Code dec	Code hex
Disable Payout Device	**091**	**0x5B**

Supported on devices:

SMART Payout	nv11

Encryption Required
no

Description

All accepted notes will be routed to the stacker and payout commands will not be accepted.

Parameters

This command has no parameters

Command packet example:

dec

127	128	001	091	218	003

hex

7F	80	01	5B	DA	03

Response

Device responds with OK for success.

dec

127	128	001	240	035	128

hex

7F	80	01	F0	23	80

back to index

Command name	Code dec	Code hex
Coin Mech Options	**090**	**0x5A**

Supported on devices:

SMART Hopper	SMART System

Encryption Required
no

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.

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

dec

127	128	002	090	001	048	220

hex

7F	80	02	5A	01	30	DC

Response

Device responds with OK for success.

dec

127	128	001	240	035	128

hex

7F	80	01	F0	23	80

back to index

Command name	Code dec	Code hex
Reset Counters	**089**	**0x59**

Supported on devices:

nv9usb	SMART Payout	nv11

Encryption Required
no

Description

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

Parameters

This command has no parameters

Command packet example:

dec

127	128	001	089	213	131

hex

7F	80	01	59	D5	83

Response

The device responds with OK for successful reset.

dec

127	128	001	240	035	128

hex

7F	80	01	F0	23	80

back to index

Command name	Code dec	Code hex
Get Counters	**088**	**0x58**

Supported on devices:

nv9usb	SMART Payout	nv11

Encryption Required
no

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.

Parameters

This command has no parameters

Command packet example:

dec

127	128	001	088	208	003

hex

7F	80	01	58	D0	03

Response

The device responds generic OK if supported and then with data representing the cumulative note count for the device. In this example, we have a device with 300 notes stacked, 210 notes stored, 180 notes dispensed and 25 notes rejected.

dec

127	128	022	240	005	044	001	000	000	210	000	000	000	180	000	000	000	104	001	000	000	025
000	000	000	241	130																

hex

7F	80	16	F0	05	2C	01	00	00	D2	00	00	00	B4	00	00	00	68	01	00	00	19
00	00	00	F1	82																

Counter response

Data byte offset	Size bytes	Description
0	1	Number of counters in set
1-4	4	Notes stacked
5-8	4	Notes stored
9-12	4	Notes dispensed
14-16	4	Notes transferred from store to stacker
17-20	4	Notes rejected

back to index

Command name	Code dec	Code hex
Event ACK	**087**	**0x57**

Supported on devices:

nv9usb	nv10usb	bv20	bv50	bv100	nv200	SMART Hopper	SMART System	nv11

Encryption Required
yes

Description

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

Parameters

This command has no parameters

Command packet example:

dec

127	128	001	087	242	003

hex

7F	80	01	57	F2	03

back to index

Command name	Code dec	Code hex
Poll With ACK	**086**	**0x56**

Supported on devices:

nv9usb	nv10usb	bv20	bv50	bv100	nv200	SMART Hopper	SMART System	nv11

Encryption Required
yes

Description

A command that behaves in the same way as the Poll command but with this command, the specified events (see table below) 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.

Parameters

This command has no parameters

Command packet example:

dec

127	128	001	086	247	131

hex

7F	80	01	56	F7	83

Response

Device responds with OK and event list. See the Poll command event list table for details on Poll Ack events.

back to index

Command name	Code dec	Code hex
Configure Bezel	**084**	**0x54**

Supported on devices:

nv200

Encryption Required
no

Description

This command allows the host to configure a supported BNV bezel. If the bezel is not supported the command will return generic response COMMAND NOT KNOWN 0xF2.

Command data is configured as in the table below. In this example, we want a red bezel fixed to EEPROM.

dec

127	128	005	084	255	000	000	001	072	220

hex

7F	80	05	54	FF	00	00	01	48	DC

Bezel configuration data

Data byte index	Function
0	Red intensity (0-255)
1	Green intensity (0-255)
2	Blue intensity (0-255)
4	Config 0 for volatile,1 - for non-volatile.

Response

Device responds OK for successful set-up.

dec

127	128	001	240	035	128

hex

7F	80	01	F0	23	80

Responds with COMMAND NOT KNOWN if not supported.

dec

127	128	001	242	044	000

hex

7F	80	01	F2	2C	00

back to index

Command name	Code dec	Code hex
Cashbox Payout Operation Data	**083**	**0x53**

Supported on devices:

SMART Hopper	SMART Payout	SMART System	nv11

Encryption Required
no

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. The quantity of denominations in the response is sent as a 2 byte little endian array; the note values as 4-byte little endian array and the country code as a 3-byte ASCII array. Each denomination in the dataset will be reported, even if 0 coins of that denomination are emptied. As money is emptied from the device, the value is checked. An additional 4 bytes will be added to the response giving a count of object that could not be validated whilst performing the operation. The response is formatted as follows: byteParameter byte 0The number denominations (n) in this response (max 20) byte 1 to byte 1 + (9*n)The individual denomination level (see description below) byte 1 to byte 1 + (9*n) + 1 to byte 1 to byte 1 + (9*n) + 4 The number of un-validated objects moved. Individual level requests: byte 0 and byte 1number of coins of this denomination moved to cashbox in operation byte 2 to byte 5The denomination value byte 6 to byte 8The ascii denomination country code

Parameters

This command has no parameters

Command packet example:

dec

127	128	001	083	233	131

hex

7F	80	01	53	E9	83

Response

In this example we return data from a device EUR with 30 x 10c, 40 x 20c, 25 x 50c coins emptied with 5 unknown coins also emptied.

dec

127	128	023	003	030	000	010	000	000	000	040	000	020	000	000	000	025	000	050	000	000	000
005	000	000	000	223	135															

hex

7F	80	17	03	1E	00	0A	00	00	00	28	00	14	00	00	00	19	00	32	00	00	00
05	00	00	00	DF	87															

back to index

Command name	Code dec	Code hex
Smart Empty	**082**	**0x52**

Supported on devices:

SMART Hopper	SMART Payout	SMART System	nv11

Encryption Required
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 denomination routed to the cashbox through this operation.

Parameters

This command has no parameters

Command packet example:

dec

127	128	001	082	236	003

hex

7F	80	01	52	EC	03

Response

Device responds with OK for successful command receipt.

dec

127	128	001	240	035	128

hex

7F	80	01	F0	23	80

back to index

Command name	Code dec	Code hex
Get Hopper Options	**081**	**0x51**

Supported on devices:

SMART Hopper

Encryption Required
no

Description

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

Parameters

This command has no parameters

Command packet example:

dec

127	128	001	081	230	003

hex

7F	80	01	51	E6	03

back to index

Command name	Code dec	Code hex
Set Hopper Options	**080**	**0x50**

Supported on devices:

SMART Hopper

Encryption Required
no

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. 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

<table>
<tr><th>REG_0</th><th>Parameter</th><th></th></tr>
<tr><td>bit 0</td><td>Pay Mode</td><td>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.</td></tr>
<tr><td>bit 1</td><td>Level check</td><td>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.</td></tr>
<tr><td>bit 2</td><td>Motor speed</td><td>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.</td></tr>
<tr><td>bit 3</td><td>Cashbox pay active</td><td>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. See table below:
<table>
<tr><th>Bit 3</th><th>Bit 0</th><th>Pay mode</th></tr>
<tr><td>0</td><td>0</td><td>Split by highest value</td></tr>
</table>
</td></tr>
</table>

		0	1	Free pay
		1	0	Split by highest, use cashbox coins in split
		1	1	Free pay, use cashbox coins in split
bit 4	not used	Set to 0		
bit 5	not used	Set to 0		
bit 6	not used	Set to 0		
bit 7	not used	Set to 0		

REG_0 bits are not used set all to 0

The default values are pre-configured in the Hopper using the Hopper manager tools and can be changed off-line by the user. The defaults mentioned here are the factory supplied defaults.

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

dec

127	128	003	080	004	000	064	056

hex

7F	80	03	50	04	00	40	38

Response

The device responds OK for success.

dec

127	128	001	240	035	128

hex

7F	80	01	F0	23	80

back to index

Command name	Code dec	Code hex
Get Build Revision	**079**	**0x4F**

Supported on devices:

nv200	SMART Hopper	SMART Payout	nv11

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, the type byte is 0, for Note Float, byte is 3 and for SMART Payout the byte is 6.

Parameters

This command has no parameters

Command packet example:

dec

127	128	001	079	162	003

hex

7F	80	01	4F	A2	03

Response

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

dec

127	128	007	240	000	000	020	006	000	021	020	127

hex

7F	80	07	F0	00	00	14	06	00	15	14	7F

This is a response from a note float unit build rev 8

dec

127	128	004	240	003	000	008	151	065

hex

7F	80	04	F0	03	00	08	97	41

back to index

Command name	Code dec	Code hex
Set Baud Rate	**077**	**0x4D**

Supported on devices:

SMART Hopper	SMART Payout	SMART System	nv11

Encryption Required
no

Description

This command has two data bytes to allow communication speed to be set on a device. The first byte is the speed to change to (see table below).

Baud rate	Byte value
9600	0
38400	1
115200	2

If the second byte is 1 then the speed will be stored in the device over resets, otherwise the speed will be changed until the next reset. The device will respond with 0xF0 at the old baud rate before changing. Please allow a minimum of 100millseconds before trying to communicate at the new baud rate.

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

dec

127	128	003	077	001	000	228	039

hex

7F	80	03	4D	01	00	E4	27

Response

Device responds OK for success at current speed, then changes to requested speed for future communications.

dec

127	128	001	240	035	128

hex

7F	80	01	F0	23	80

Devcie responds with COMMAND NOT KNOWN if not supported.

dec

127	128	001	242	044	000

hex

7F	80	01	F2	2C	00

back to index

Command name	Code dec	Code hex
Request Key Exchange	**076**	**0x4C**

Supported on devices:

nv9usb	nv10usb	bv20	bv50	bv100	nv200	SMART Hopper	SMART Payout	SMART System	nv11

Encryption Required
no

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.

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

dec

127	128	009	076	121	200	156	226	222	006	000	000	157	082

hex

7F	80	09	4C	79	C8	9C	E2	DE	06	00	00	9D	52

Response

Responds OK for success.

dec

127	128	001	240	035	128

hex

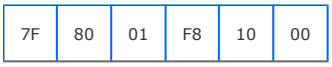
7F	80	01	F0	23	80

Responds with FAIL if the Generator and Modulus are not set.

dec

127	128	001	248	016	000

hex



back to index

Command name	Code dec	Code hex
Set Modulus	**075**	**0x4B**

Supported on devices:

nv9usb	nv10usb	bv20	bv50	bv100	nv200	SMART Hopper	SMART Payout	SMART System	nv11

Encryption Required
no

Description

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

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

dec

127	128	009	075	141	166	019	000	000	000	000	000	108	246

hex

7F	80	09	4B	8D	A6	13	00	00	00	00	00	6C	F6

Response

Responds OK if number is prime.

dec

127	128	001	240	035	128

hex

7F	80	01	F0	23	80

Responds PARAMETER OUT OF RANGE for prime test fail

dec

127	128	001	244	056	000

hex

7F	80	01	F4	38	00

back to index

Command name	Code dec	Code hex
Set Generator	**074**	**0x4A**

Supported on devices:

nv9usb	nv10usb	bv20	bv50	bv100	nv200	SMART Hopper	SMART Payout	SMART System	nv11

Encryption Required
no

Description

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

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

dec

127	128	009	074	197	005	143	058	000	000	000	000	178	115

hex

7F	80	09	4A	C5	05	8F	3A	00	00	00	00	B2	73

Response

Responds OK if number is prime.

dec

127	128	001	240	035	128

hex

7F	80	01	F0	23	80

Responds PARAMETER OUT OF RANGE for prime test fail

dec

127	128	001	244	056	000

hex

7F	80	01	F4	38	00

back to index

Command name	Code dec	Code hex
Set Coin Mech Global Inhibit	**073**	**0x49**

Supported on devices:

SMART Hopper	SMART System

Encryption Required
no

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.

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

dec

127	128	002	073	001	051	054

hex

7F	80	02	49	01	33	36

Response

Device responds OK for success.

dec

127	128	001	240	035	128

hex

7F	80	01	F0	23	80

back to index

Command name	Code dec	Code hex
Payout By Denomination	**070**	**0x46**

Supported on devices:

SMART Hopper	SMART Payout	SMART System

Encryption Required
yes

Description

A command to payout the requested quantity of individual denominations. 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.

The command is formatted as follows

Byte	function
0	The number of individual requests in this command (n) max 20
1 - 2	Number to payout
3 - 6	The denomination value
7 - 9	The ascii country code
...	Repeat the last 9 bytes for each of the n denominations requested
1 + (n*9)	The test/Payout option byte

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.

dec

127	128	039	070	004	004	000	100	000	000	000	069	085	082	005	000	010	000	000	000	069	085
082	003	000	100	000	000	000	071	066	080	002	000	050	000	000	000	071	066	080	088	148	183

hex

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

Response

The device responds with OK for a successful request.

dec

127	128	001	240	035	128

hex

7F	80	01	F0	23	80

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

Error reason	Code
Not enough value in device	0x00
Cannot pay exact amount	0x01
Device busy	0x03
Device disabled	0x04

In this case, the request failed due to an exact amount available in the device not available to pay the request.

dec

127	128	002	245	002	048	062

hex

7F	80	02	F5	02	30	3E

back to index

Command name	Code dec	Code hex
Set Value Reporting Type	**069**	**0x45**

Supported on devices:

nv11

Encryption Required
no

Description

This will set the method of reporting values of notes. There are two options, by a four-byte value of the note or by the channel number of the value from the banknote validator. If the channel number is used then the actual value must be determined using the data from the Validator command Unit Data. The default operation is by 4-byte value. Send 0x00 to set Report by value, 0x01 to set Report By Channel.

In this example we set the value reporting to report by channel.

dec

127	128	002	069	001	051	030

hex

7F	80	02	45	01	33	1E

Response

Device responds with OK for setting success.

dec

127	128	001	240	035	128

hex

7F	80	01	F0	23	80

The device responds with COMMAND CANNOT BE PROCESSED and an error byte for failure to set option. In this example an invalid currency miss-match was detected between the validator and connected payout.

dec

127	128	002	245	002	048	062

hex

7F	80	02	F5	02	30	3E

Set report error codes

Error reason	Error code
No payout connected	1
Invalid currency detected	2
Payout device error	3

back to index

Command name	Code dec	Code hex
Float By Denomination	**068**	**0x44**

Supported on devices:

SMART Hopper	SMART Payout	SMART System

Encryption Required
yes

Description

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

The command is formatted as follows

Byte	function
0	The number of individual requests in this command (n) max 20
1 - 2	Number to leave in payout
3 - 6	The denomination value
7 - 9	The ascii country code
...	Repeat the last 9 bytes for each of the n denominations requested
1 + (n*9)	The test/Payout option byte

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 float to to 5 x 1.00 EUR, 5 x 0.10 EUR, 3 x 1.00 GBP and 2 x 0.50 GBP.

dec

127	128	039	068	004	004	000	100	000	000	000	069	085	082	005	000	010	000	000	000	069	085
082	003	000	100	000	000	000	071	066	080	002	000	050	000	000	000	071	066	080	088	215	017

hex

7F	80	27	44	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	D7	11

Response

The device responds with OK for a successful request.

dec

127	128	001	240	035	128

hex

7F	80	01	F0	23	80

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

Error reason	Code
Not enough value in device	0x00
Cannot pay exact amount	0x01
Device busy	0x03
Device disabled	0x04

In this case, the request failed due to an exact amount available in the device not available to pay the request.

dec

127	128	002	245	002	048	062

hex

7F	80	02	F5	02	30	3E

back to index

Command name	Code dec	Code hex
Stack Note	**067**	**0x43**

Supported on devices:

nv11

Encryption Required
no

Description

The Note Float will stack the last note that was stored. This is the note that is in the highest position in the table returned by the Get Note Positions Command. If the stack operation is possible the Note Float will reply with generic response OK. If the stack is not possible the reply will be generic response command cannot be processed, followed by an error code as shown in the table.

Parameters

This command has no parameters

Command packet example:

dec

127	128	001	067	138	003

hex

7F	80	01	43	8A	03

Response

Device responds with OK and commences note stacking for success.

dec

127	128	001	240	035	128

hex

7F	80	01	F0	23	80

Device responds with COMMAND CANNOT BE PROCESSED and an error code for fail. In this example the note float unit was empty

dec

127	128	002	245	002	048	062

hex

7F	80	02	F5	02	30	3E

Stack Request codes

Error reason	Code
Note float unit not connected	0x01
Note float empty	0x02
Note float busy	0x03
Note float disabled	0x04

back to index

Command name	Code dec	Code hex
Payout Note	**066**	**0x42**

Supported on devices:

nv11

Encryption Required
no

Description

The Note Float will payout the last note that was stored. This is the note that is in the highest position in the table returned by the Get Note Positions Command. If the payout is possible the Note Float will reply with generic response OK. If the payout is not possible the reply will be generic response COMMAND CANNOT BE PROCESSED, followed by an error code shown in the table below

Parameters

This command has no parameters

Command packet example:

dec

127	128	001	066	143	131

hex

7F	80	01	42	8F	83

Response

Device responds with OK and commences note payout for success.

dec

127	128	001	240	035	128

hex

7F	80	01	F0	23	80

Device responds with COMMAND CANNOT BE PROCESSED and an error code for fail. In this example the note float unit was empty

dec

127	128	002	245	002	048	062

hex

7F	80	02	F5	02	30	3E

Payout request error codes

Error reason	Code
Note float unit not connected	0x01
Note float empty	0x02
Note float busy	0x03
Note float disabled	0x04

back to index

Command name	Code dec	Code hex
Get Note Positions	**065**	**0x41**

Supported on devices:

nv11

Encryption Required
no

Description

This command will return the number of notes in the Note Float and the value in each position. The way the value is reported is specified by the Set Reporting Type command. The value can be reported by its value or by the channel number of the bill validator. The first note in the table is the first note that was paid into the Note Float. The Note Float is a LIFO system, so the note that is last in the table is the only one that is available to be paid out or moved into the stacker.

Parameters

This command has no parameters

Command packet example:

dec

127	128	001	065	133	131

hex

7F	80	01	41	85	83

Response

The device will respond with data array formatted as shown the format tables.

If the currency stored in the note float does not match the currency stored in the validator the device will respond with COMMAND CANNOT BE PROCESSED with error code 2 (Invalid currency)

dec

127	128	002	245	002	048	062

hex

7F	80	02	F5	02	30	3E

Report by value

byte offset	parameter
0	Number of notes in NV11 (n)
1 - 4	Value of note in slot 1
5 - 8	Value of note in slot 2
9 - 12	Value of note in slot 3
...	...
(n*4) - (n * 4) + 4	Value of note in slot n

Report by channel

byte offset	parameter
0	Number of notes in NV11 (n)
1	Channel of note in slot 1
2	Channel of note in slot 2
3	Channel of note in slot 3
...	...
n	Channel of note in slot n

back to index

Command name	Code dec	Code hex
Set Coin Mech Inhibits	**064**	**0x40**

Supported on devices:

SMART Hopper	SMART System

Encryption Required
no

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 require 3 bytes of data. Byte 0 is the required inhibit state: 0x01 for not inhibited, 0x00 for inhibited. Bytes 1 and give the value of the coin denomination. In this example we want to inhibit 1.00 EUR coins.

dec

127	128	004	064	000	100	000	171	217

hex

7F	80	04	40	00	64	00	AB	D9

For protocol version greater than 6, we also send the 3 byte ascii code for the denomination. In this example we want to enable acceptance of EUR 0.50c coins.

dec

127	128	007	064	001	050	000	069	085	082	202	094

hex

7F	80	07	40	01	32	00	45	55	52	CA	5E

Response

The device responds with OK for success.

dec

127	128	001	240	035	128

hex

7F	80	01	F0	23	80

back to index

Command name	Code dec	Code hex
Empty All	**063**	**0x3F**

Supported on devices:

SMART Hopper	SMART Payout	SMART System	nv11

Encryption Required
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.

Parameters

This command has no parameters

Command packet example:

dec

127	128	001	063	129	130

hex

7F	80	01	3F	81	82

Response

The device responds OK for a successful command reciept

dec

127	128	001	240	035	128

hex

7F	80	01	F0	23	80

back to index

Command name	Code dec	Code hex
Get Minimum Payout	**062**	**0x3E**

Supported on devices:

SMART Hopper	SMART Payout	SMART System

Encryption Required
no

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.

dec

127	128	001	062	132	002

hex

7F	80	01	3E	84	02

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

dec

127	128	004	062	069	085	082	020	227

hex

7F	80	04	3E	45	55	52	14	E3

Response

The device responds with a 4 byte value of minimum payout. In ths case; 200

dec

127	128	005	240	200	000	000	000	167	194

hex

7F	80	05	F0	C8	00	00	00	A7	C2

back to index

Command name	Code dec	Code hex
Float Amount	**061**	**0x3D**

Supported on devices:

SMART Hopper	SMART Payout	SMART System

Encryption Required
yes

Description

A command to float the hopper 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.

On protocol versions less than 6, the command data was formatted as byte 0 and 1: the min payout to leave. Bytes 2 to 4, the value of the amount to leave. In this example we request a float value 100.00 with a minimum possible payout of 0.50c.

dec

127	128	007	061	050	000	016	039	000	000	029	028

hex

7F	80	07	3D	32	00	10	27	00	00	1D	1C

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

dec

127	128	011	061	050	000	039	016	000	000	069	085	082	088	167	218

hex

7F	80	0B	3D	32	00	27	10	00	00	45	55	52	58	A7	DA

Response

The device responds with OK for a successful request.

dec

127	128	001	240	035	128

hex

7F	80	01	F0	23	80

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

Error reason	Code
Not enough value in device	0x00
Cannot pay exact amount	0x01
Device busy	0x03
Device disabled	0x04

In this case, the request failed due to an exact amount available in the device not available to pay the request.

dec

127	128	002	245	002	048	062

hex

7F	80	02	F5	02	30	3E

back to index

Command name	Code dec	Code hex
Get Denomination Route	**060**	**0x3C**

Supported on devices:

SMART Hopper	SMART Payout	SMART System	nv11

Encryption Required
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).

Send the denomination value to get its route. This example shows a request to obtain the route of EUR 5.00 note in protocol version 6

dec

127	128	008	060	244	001	000	000	069	085	082	047	014

hex

7F	80	08	3C	F4	01	00	00	45	55	52	2F	0E

Response

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

Route	Code

Recycled and used for payouts	0x00
Detected denomination is routed to system cashbox	0x01

In this example 0 = route stored for payout.

dec

127	128	002	240	000	063	160

hex

7F	80	02	F0	00	3F	A0

With note payouts, the device responds with COMMAND CANNOT BE PROCESSED and an error byte for failure to enable. In this example an invalid currency miss-match was detected between the validator and connected payout.

dec

127	128	002	245	002	048	062

hex

7F	80	02	F5	02	30	3E

Device responds with PARAMETER out of RANGE if the denomination does not exist.

dec

127	128	001	244	056	000

hex

7F	80	01	F4	38	00

Note payout Error code

Error reason	Error code
No payout connected	1
Invalid currency detected	2
Payout device error	3

back to index

Command name	Code dec	Code hex
Set Denomination Route	**059**	**0x3B**

Supported on devices:

SMART Hopper	SMART Payout	nv11

Encryption Required
no

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 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 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).

The command data consists of route,value and country codes depending on the device and protocol version level used. See tables below. Here is an example of a request to route a 10c EUR coin to be stored for payout using protocol version 6

dec

127	128	009	059	000	010	000	000	000	069	085	082	008	067

hex

7F	80	09	3B	00	0A	00	00	00	45	55	52	08	43

Response

The device responds with OK for success.

dec

127	128	001	240	035	128

hex

7F	80	01	F0	23	80

Device responds with PARAMETER out of RANGE if the denomination does not exist.

dec

127	128	001	244	056	000

hex

7F	80	01	F4	38	00

With note payouts, the device responds with COMMAND CANNOT BE PROCESSED and an error byte for failure.
In this example an invalid currency miss-match was detected between the validator and connected payout.

dec

127	128	002	245	002	048	062

hex

7F	80	02	F5	02	30	3E

Note payout Error code

Error reason	Error code
No payout connected	1
Invalid currency detected	2
Payout device error	3

back to index

Command name	Code dec	Code hex
Halt Payout	**056**	**0x38**

Supported on devices:

SMART Hopper	SMART Payout	SMART System

Encryption Required
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.

Parameters

This command has no parameters

Command packet example:

dec

127	128	001	056	144	002

hex

7F	80	01	38	90	02

Response

Device responds with ok for a halt success.

dec

127	128	001	240	035	128

hex

7F	80	01	F0	23	80

back to index

Command name	Code dec	Code hex
Communication Pass Through	**055**	**0x37**

Supported on devices:

SMART Hopper

Description

Used with SMART Hopper only. This command sets USB pass through mode. SMART hopper then works only as USB to serial converter to allow direct communication (firmware/dataset update) with devices connected to Smart Hopper UARTS. This command was implemented in firmware versions greater or equal to 6.16.

This command has 1 data byte giving the route on the SMART Hopper connector panel where the coms is to pass through. 0 for the eSSP connection, 1 for the Coin mech connection. In this case we want to route coms to the coin mech.

Once in pass through mode, we can reset to normal communication by sending a sigature sequence of bytes:

Wait for 500ms Send 0x55 0xAA 0xAA 0x55 Wait for 500ms Send 0xAA 0x55 0x55 0xAA

Smart Hopper will then reset itself back to normal operation mode.

dec

127	128	002	055	001	054	178

hex

7F	80	02	37	01	36	B2

Response

The device responds OK for success.

dec

127	128	001	240	035	128

hex

7F	80	01	F0	23	80

back to index

Command name	Code dec	Code hex
Get Denomination Level	**053**	**0x35**

Supported on devices:

SMART Hopper	SMART Payout	SMART System

Encryption Required
no

Description

This command returns the level of a denomination stored in a payout device as a 2 byte value. In protocol versions greater or equal to 6, the host adds a 3 byte ascii country code to give mulit-currency functionality.

Send the requested denomination to find its level. In this case a request to find the amount of 0.10c coins in protocol version 5.

dec

127	128	005	053	010	000	000	000	030	073

hex

7F	80	05	35	0A	00	00	00	1E	49

In this example a request is sent to find the level of EUR 5.00 notes.

dec

127	128	008	053	244	001	000	000	069	085	082	025	158

hex

7F	80	08	35	F4	01	00	00	45	55	52	19	9E

Response

Device responds with level of 200

dec

127	128	002	200	000	060	176

hex

7F	80	02	C8	00	3C	B0

If the denomination does not exist in the device, it will respond with COMMAND CANNOT BE PROCESSED.

dec

127	128	001	245	061	128

hex

7F	80	01	F5	3D	80

back to index

Command name	Code dec	Code hex
Set Denomination Level	**052**	**0x34**

Supported on devices:

SMART Hopper	SMART System

Encryption Required
no

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. Note that protocol 6 version commands have been expanded to use a 4-byte coin value.

The command data is formatted as byte 0 and byte 1 give the number of coins to add. In protocol version 5, the denomination is then sent as a two byte value. In protocol version greater than 5, the denomination is sent as 4 byte value plus 3 bytes ascii country code. In this example we want to increase the level of .50c coin by 20 using protocol version 5.

dec

127	128	005	052	020	000	050	000	099	253

hex

7F	80	05	34	14	00	32	00	63	FD

In this example we want to increase the level of EUR 1.00 coins by 12 on a device set with protocol version 6

dec

127	128	010	052	012	000	100	000	000	000	069	085	082	199	040

hex

7F	80	0A	34	0C	00	64	00	00	00	45	55	52	C7	28

Response

The device responds with OK for level change success.

dec

127	128	001	240	035	128

hex

7F	80	01	F0	23	80

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

dec

127	128	001	245	061	128

hex

7F	80	01	F5	3D	80

back to index

Command name	Code dec	Code hex
Payout Amount	**051**	**0x33**

Supported on devices:

SMART Hopper	SMART Payout	SMART System

Encryption Required
yes

Description

A command to set the monetary value to be paid by the payout unit. Using protocol version 6, the host also sends a pre-test option byte (TEST_PAYOUT_AMOUT 0x19, PAYOUT_AMOUNT 0x58), which will determine if the command amount is tested or paid out. This is useful for multi-payout systems so that the ability to pay a split down amount can be tested before committing to actual payout.

This example is a request to payout EUR 5.00 using protocol version 4

dec

127	128	004	051	244	001	000	042	247

hex

7F	80	04	33	F4	01	00	2A	F7

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

dec

127	128	009	051	244	001	000	000	069	085	082	088	195	238

hex

7F	80	09	33	F4	01	00	00	45	55	52	58	C3	EE

Response

The device responds with OK for a successful request.

dec

127	128	001	240	035	128

hex

7F	80	01	F0	23	80

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

Error reason	Code
Not enough value in device	0x00
Cannot pay exact amount	0x01
Device busy	0x03
Device disabled	0x04

In this case, the request failed due to an exact amount available in the device not available to pay the request.

dec

127	128	002	245	002	048	062

hex

7F	80	02	F5	02	30	3E

back to index

Command name	Code dec	Code hex
Set Refill Mode	**048**	**0x30**

Supported on devices:

SMART Payout

Encryption Required
no

Description

A command sequence to set or reset the facility for the payout to reject notes that are routed to the payout store but the firmware determines that they are un-suitable for storage. In default mode, they would be re-routed to the stacker. In refill mode they will be rejected from the front of the NV200.

This example show the sequence of command bytes to set the mode.

dec

127	128	006	048	005	129	016	017	001	082	245

hex

7F	80	06	30	05	81	10	11	01	52	F5

This sequence will un-set the mode for normal operation.

dec

127	128	006	048	005	129	016	017	000	087	117

hex

7F	80	06	30	05	81	10	11	00	57	75

To read the current refill mode send this sequence: Returns 1 byte: 0x00 the option is not set, 0x01 the option is set.

dec

127	128	005	048	005	129	016	001	148	238

hex

7F	80	05	30	05	81	10	01	94	EE

Response

Responds with OK for successful setup.

dec

127	128	001	240	035	128

hex

7F	80	01	F0	23	80

Responds with data byte when the read sequence is set. In this case we read that the option is set.

dec

127	128	002	240	001	058	032

hex

7F	80	02	F0	01	3A	20

back to index

Command name	Code dec	Code hex
Get Bar Code Data	**039**	**0x27**

Supported on devices:

nv9usb	nv200

Encryption Required
no

Description

Command to obtain last valid bar code ticket data, send in response to a Bar Code Ticket Validated event. This command will return a variable length data steam, a generic response (OK) followed by a status byte, a bar code data length byte, then a stream of bytes of the ticket data in ASCII.

Parameters

This command has no parameters

Command packet example:

dec

127	128	001	039	209	130

hex

7F	80	01	27	D1	82

Response

The response data byte 0 contains status information: 0x00 â " no valid data, 0x01 ticket in escrow, 0x02 ticket stacked, 0x03 ticket rejected. The next byte is the length of the bar code data, then follows the ascii data for the barcode read itself. In this example a ticket is in escrow with data length 6 and data 123456.

dec

127	128	009	240	001	006	049	050	051	052	053	054	161	005

hex

7F	80	09	F0	01	06	31	32	33	34	35	36	A1	05

ascii

	.	.	.	.	.	1	2	3	4	5	6	.	.

back to index

Command name	Code dec	Code hex
Set Bar Code Inhibit Status	**038**	**0x26**

Supported on devices:

nv9usb	nv200

Encryption Required
no

Description

Sets up the bar code inhibit status register.

A single data byte representing a bit register is sent. Bit 0 is Currency read enable (0 = enable, 1= disable) Bit 1 is the Bar code enable (0 = enable, 1 = disable). All other bits are not used and set to 1. This example shows a request to a device to have currency enabled, bar code enabled.

dec

127	128	002	038	255	049	086

hex

7F	80	02	26	FF	31	56

Response

Device responds with OK for successful configuration

dec

127	128	001	240	035	128

hex

7F	80	01	F0	23	80

back to index

Command name	Code dec	Code hex
Get Bar Code Inhibit Status	**037**	**0x25**

Supported on devices:

nv9usb	nv200

Encryption Required
no

Description

Command to return the current bar code/currency inhibit status.

Parameters

This command has no parameters

Command packet example:

dec

127	128	001	037	222	002

hex

7F	80	01	25	DE	02

Response

Return a byte bit register. Bit 0 is Currency read enable (0 = enable, 1= disable) Bit 1 is the Bar code enable (0 = enable, 1 = disable). All other bits are not used and set to 1. This example shows a device with currency enabled, bar code disabled.

dec

127	128	002	240	254	056	034

hex

7F	80	02	F0	FE	38	22

back to index

Command name	Code dec	Code hex
Set Bar Code Configuration	**036**	**0x24**

Supported on devices:

nv9usb	nv200

Encryption Required
no

Description

This command allows the host to set-up the bar code reader(s) configuration on the device.

3 bytes of data define the configuration. In this example we enable both readers with format interleaved 1 of 5 for 18 characters.

dec

127	128	004	036	003	001	028	203	087

hex

7F	80	04	24	03	01	1C	CB	57

Bar code configuration data

Command data byte index	Function
0	0x00 Enable none, 0x01 enable top, 0x02 = enable bottom, 0x03 = enable both
1	Bar code format (0x01 = Interleaved 2 of 5)
2	Number of characters (Min 6 Max 24)

Response

Device response with OK for successful set-up.

dec

127	128	001	240	035	128

hex

7F	80	01	F0	23	80

back to index

Command name	Code dec	Code hex
Get Bar Code Reader Configuration	**035**	**0x23**

Supported on devices:

nv9usb	nv200

Encryption Required
no

Description

Returns the set-up data for the device bar code readers.

Parameters

This command has no parameters

Command packet example:

dec

127	128	001	035	202	002

hex

7F	80	01	23	CA	02

Response

Responds with 4 bytes of data formatted as in table below

Response data byte index	Function
0	Bar code hardware status (0x00 = none, 0x01 = Top reader fitted, 0x02 = Bottom reader fitted, 0x03 = both fitted)
1	Readers enabled (0x00 = none, 0x01 = top, 0x02 = bottom, 0x03 = both)

2	Bar code format (0x01 = Interleaved 2 of 5)
3	Number of characters (Min 6 max 24)

back to index

Command name	Code dec	Code hex
Get All Levels	**034**	**0x22**

Supported on devices:

SMART Hopper	SMART Payout	SMART System

Encryption Required
no

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.

Parameters

This command has no parameters

Command packet example:

dec

127	128	001	034	207	130

hex

7F	80	01	22	CF	82

Response

The first data byte oin the response is the number of counters returned. Each counter consists of 9 bytes of data made up as: 2 bytes giving the denomination level, 4 bytes giving the value and 3 bytes of ascii country code. In this example, we have a device 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. So the response will be:

dec

127	128	038	240	004	100	000	020	000	000	000	069	085	082	065	000	050	000	000	000	069	085
082	000	000	100	000	000	000	069	085	082	012	000	200	000	000	000	069	085	082	132	208	

hex

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 name	Code dec	Code hex
Get Dataset Version	**033**	**0x21**

Supported on devices:

nv9usb	nv10usb	bv20	bv50	bv100	nv200	SMART System	nv11

Encryption Required
no

Description

Returns a string of ascii codes giving the full dataset version of the device.

Parameters

This command has no parameters

Command packet example:

dec

127	128	001	033	197	130

hex

7F	80	01	21	C5	82

Response

This example shows a device with dataset version EUR01610.

dec

127	128	009	240	069	085	082	048	049	054	049	048	184	042

hex

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 name	Code dec	Code hex
Get Firmware Version	**032**	**0x20**

Supported on devices:

nv9usb	nv10usb	bv20	bv50	bv100	nv200	SMART Hopper	SMART System	nv11

Encryption Required
no

Description

Returns the full firmware version ascii data array for this device.

Parameters

This command has no parameters

Command packet example:

dec

127	128	001	032	192	002

hex

7F	80	01	20	C0	02

Response

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

dec

127	128	017	240	078	086	048	050	048	048	052	049	052	049	052	057	056	048	048	048	222	085

hex

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	.	U

In this example we can extract the firmware details:

Device version = NV200 (bytes 0 to 5 ignore leading 0).

Release version = 1498 (9-12)

Beta version = 000 (13-15)

back to index

Command name	Code dec	Code hex
Hold	**024**	**0x18**

Supported on devices:

nv9usb	nv10usb	bv20	bv50	bv100	nv200	nv11

Encryption Required
no

Description

This command may be sent to BNV when Note Read has changed from 0 to >0 (valid note seen) if the user does not wish to accept the note with the next command. This command will also reset the 10-second time-out period after which a note held would be rejected automatically, so it should be sent before this time-out if an escrow function is required. If there is no note in escrow to hold, the device will reply with COMMAND CANNOT BE PROCESSED (0xF5)

Parameters

This command has no parameters

Command packet example:

dec

127	128	001	024	083	130

hex

7F	80	01	18	53	82

Response

Responds with OK if note is held.

dec

127	128	001	240	035	128

hex

7F	80	01	F0	23	80

If there is no note to hold, response is COMMAND CANNOT BE PROCESSED 0xF5

dec

127	128	001	245	061	128

hex

7F	80	01	F5	3D	80

back to index

Command name	Code dec	Code hex
Last Reject Code	**023**	**0x17**

Supported on devices:

nv9usb	nv10usb	bv20	bv50	bv100	nv200	nv11

Encryption Required
no

Description

Returns a single byte that indicates the reason for the last banknote reject. The codes are shown in the table below. Specifics of note validation are not shown to protect integrity of manufacturers security.

Parameters

This command has no parameters

Command packet example:

dec

127	128	001	023	113	130

hex

7F	80	01	17	71	82

Response

This response shows that the last rejected note was due to slow transport detected. see table below for full code list.

dec

127	128	002	240	013	018	032

hex

7F	80	02	F0	0D	12	20

Code	Reject Reason
0x00	Note accepted
0x01	Note length incorrect
0x02	Reject reason 2
0x03	Reject reason 3
0x04	Reject reason 4
0x05	Reject reason 5
0x06	Channel inhibited
0x07	Second note inserted
0x08	Reject reason 8
0x09	Note recognised in more than one channel
0x0A	Reject reason 10
0x0B	Note too long
0x0C	Reject reason 12
0x0D	Mechanism slow/stalled
0x0E	Strimming attempt detected
0x0F	Fraud channel reject
0x10	No notes inserted
0x11	Peak detect fail
0x12	Twisted note detected
0x13	Escrow time-out
0x14	Bar code scan fail
0x15	Rear sensor 2 fail
0x16	Slot fail 1
0x17	Slot fail 2
0x18	Lens over-sample
0x19	Width detect fail
0x1A	Short note detected

0x1B	Note payout
0x1C	Unable to stack note

back to index

Command name	Code dec	Code hex
Sync	**017**	**0x11**

Supported on devices:

nv9usb	nv10usb	bv20	bv50	bv100	nv200	SMART Hopper	SMART Payout	SMART System	nv11

Encryption Required
no

Description

A command to establish communications with a slave device. 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.

Parameters

This command has no parameters

Command packet example:

dec

127	128	001	017	101	130

hex

7F	80	01	11	65	82

Response

Responds to Sync command with OK

dec

127	128	001	240	035	128

hex

7F	80	01	F0	23	80

back to index

Command name	Code dec	Code hex
Channel Re-teach Data	**016**	**0x10**

Supported on devices:

nv9usb	nv10usb	bv20	bv50	bv100	nv200	nv11

Encryption Required
no

Description

This is a vestigial command and may be deprecated in future versions. Do not use. If it is supported in a device it will return all zeros.

Parameters

This command has no parameters

Command packet example:

dec

127	128	001	016	096	002

hex

7F	80	01	10	60	02

Response

Always returns zeros if implemented in a device.

dec

127	128	004	240	000	000	000	152	193

hex

7F	80	04	F0	00	00	00	98	C1

Returns COMMAND NOT KNOWN in unsupported devices.

dec

127	128	001	242	044	000

hex

7F	80	01	F2	2C	00

back to index

Command name	Code dec	Code hex
Channel Security Data	**015**	**0xF**

Supported on devices:

nv9usb	nv10usb	bv20	bv50	bv100	nv200	nv11

Encryption Required
no

Description

Command which returns a number of channels byte (the highest channel used) and then 1 to n bytes which give the security of each channel up to the highest one, a zero indicates that the channel is not implemented. (1 = low, 2 = std, 3 = high, 4 = inhibited).

Parameters

This command has no parameters

Command packet example:

dec

127	128	001	015	033	130

hex

7F	80	01	0F	21	82

Response

In this example a validator has notes in channels 1,2,4,6,7 all at standard security.

dec

127	128	009	240	007	002	002	000	002	000	002	002	148	132

hex

7F	80	09	F0	07	02	02	00	02	00	02	02	94	84

back to index

Command name	Code dec	Code hex
Channel Value Request	**014**	**0xE**

Supported on devices:

nv9usb	nv10usb	bv20	bv50	bv100	nv200	nv11

Encryption Required
no

Description

Returns channel value data for a banknote validator. Formatted as: byte 0 - the highest channel used the a value byte representing each of the denomination values. The real value is obtained by multiplying by the value multiplier. If the validator is greater than or equal to protocol version 6 then the channel values response will be given as: Highest Channel, Value Per Channel (0 for expanded values),3 Byte ASCI country code for each channel, 4- byte Full channel Value for each channel.

Parameters

This command has no parameters

Command packet example:

dec

127	128	001	014	036	002

hex

7F	80	01	0E	24	02

Response

This example shows a response for notes in channels 1,2,4,6,7.

dec

127	128	009	240	007	005	010	000	020	000	050	100	188	218

hex

7F	80	09	F0	07	05	0A	00	14	00	32	64	BC	DA

back to index

Command name	Code dec	Code hex
Unit Data	**013**	**0xD**

Supported on devices:

nv9usb	nv10usb	bv20	bv50	bv100	nv200	nv11

Encryption Required
no

Description

Returns, Unit type (1 Byte integer), Firmware Version (4 bytes ASCII string), Country Code (3 Bytes ASCII string), Value Multiplier (3 bytes integer), Protocol Version (1 Byte, integer)

Parameters

This command has no parameters

Command packet example:

dec

127	128	001	013	046	002

hex

7F	80	01	0D	2E	02

Response

This is a response example for a banknote validator EUR 5,10,20 version 3.00 protocol version 7

dec

127	128	013	240	000	048	052	048	048	069	085	082	001	000	000	007	107	165

hex

7F	80	0D	F0	00	30	34	30	30	45	55	52	01	00	00	07	6B	A5

ascii

	.	.	.	.	0	4	0	0	E	U	R	.	.	.	.	k	.

back to index

Command name	Code dec	Code hex
Get Serial Number	**012**	**0xC**

Supported on devices:

nv9usb	nv10usb	bv20	bv50	bv100	nv200	SMART Hopper	SMART Payout	SMART System	nv11

Encryption Required
no

Description

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

Parameters

This command has no parameters

Command packet example:

dec

127	128	001	012	043	130

hex

7F	80	01	0C	2B	82

Response

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).

dec

127	128	005	240	000	028	150	044	212	151

hex

7F	80	05	F0	00	1C	96	2C	D4	97

back to index

Command name	Code dec	Code hex
Enable	**010**	**0xA**

Supported on devices:

nv9usb	nv10usb	bv20	bv50	bv100	nv200	SMART Hopper	nv11

Encryption Required
no

Description

Send this command to enable a disabled device.

Parameters

This command has no parameters

Command packet example:

dec

127	128	001	010	063	130

hex

7F	80	01	0A	3F	82

Response

Device responds OK for enable success.

dec

127	128	001	240	035	128

hex

7F	80	01	F0	23	80

For an NV11 unit the command repiles COMMAND NOT PROCESSED if the Note Float is jammed or dissconnected when the power is turned on.

dec

127	128	001	245	061	128

hex

7F	80	01	F5	3D	80

back to index

Command name	Code dec	Code hex
Disable	**009**	**0x9**

Supported on devices:

nv9usb	nv10usb	bv20	bv50	bv100	nv200	SMART Hopper	SMART System	nv11

Encryption Required
no

Description

The peripheral will switch to its disabled state, it will not execute any more commands or perform any actions until enabled, any poll commands will report disabled.

Parameters

This command has no parameters

Command packet example:

dec

127	128	001	009	053	130

hex

7F	80	01	09	35	82

Response

The device responds OK.

dec

127	128	001	240	035	128

hex

7F	80	01	F0	23	80

For an NV11 unit the command repiles COMMAND NOT PROCESSED if the Note Float is jammed or dissconnected when the power is turned on.

dec

127	128	001	245	061	128

hex

7F	80	01	F5	3D	80

back to index

Command name	Code dec	Code hex
Reject Banknote	**008**	**0x8**

Supported on devices:

nv9usb	nv10usb	bv20	bv50	bv100	nv200	nv11

Encryption Required
no

Description

A command to reject a note held in escrow in the banknote validator. For devices apart form NV11; if there is no note in escrow to be rejected, the device replies with COMMAND CANNOT BE PROCESSED (0xF5).

Parameters

This command has no parameters

Command packet example:

dec

127	128	001	008	048	002

hex

7F	80	01	08	30	02

Response

Responds with OK if a note is available for reject.

dec

127	128	001	240	035	128

hex

7F	80	01	F0	23	80

Device responds with COMMAND CANNOT BE PROCESSED (0xF5) if no banknote available to reject.

dec

127	128	001	245	061	128

hex

7F	80	01	F5	3D	80

back to index

Command name	Code dec	Code hex
Poll	**007**	**0x7**

Supported on devices:

nv9usb	nv10usb	bv20	bv50	bv100	nv200	SMART Hopper	SMART Payout	SMART System	nv11

Encryption Required
no

Description

The poll command returns the list of events that have occurred within the device since the last poll. The format of the events depends on the protocol version set within the device. Note that more than one event can occur within a poll response so ensure that the full return array is scanned.

Parameters

This command has no parameters

Command packet example:

dec

127	128	001	007	018	002

hex

7F	80	01	07	12	02

Poll event responses

Event name	**Slave reset**

Code	Description
0xF1	The device has undergone a power reset.

Poll with ACK event
No

Supported on:

All

Event data format:

This event has no data associated with it

Event name	**Read Note**

Code	Description
0xEF	A note is in the process of being scanned by the device (byte value 0) or a valid note has been scanned and is in escrow (byte value gives the channel number)

Poll with ACK event
No

Supported on:

bv20	bv50	bv100	nv9usb	nv10usb	nv200	SMART Payout	nv11

Event data format:

All protocol versions
1 byte showing the channel of the recognised note. This will be zero if the note has not yet been recognised

Event name	**Credit Note**

Code	Description
0xEE	A note has passed through the device, past the point of possible recovery and the host can safely issue its credit amount. The byte value is the channel number of the note to credit.

Poll with ACK event
Yes

Supported on:

bv20	bv50	bv100	nv9usb	nv10usb	nv200	SMART Payout	nv11

Event data format:

All protocol versions
1 byte showing the channel of the credited note.

Event name	**Note Rejecting**

Code	Description
0xED	The note is in the process of being rejected from the validator.

Poll with ACK event
No

Supported on:

bv20	bv50	bv100	nv9usb	nv10usb	nv200	SMART Payout	nv11

Event data format:

This event has no data associated with it

Event name	**Note Rejected**

Code	**Description**
0xEC	The note has been rejected from the validator and is available for the user to retrieve.

Poll with ACK event
No

Supported on:

bv20	bv50	bv100	nv9usb	nv10usb	nv200	SMART Payout	nv11

Event data format:

This event has no data associated with it

Event name	**Note Stacking**

Code	Description
0xCC	The note is being moved from the escrow position to the host exit section of the device.

Poll with ACK event
No

Supported on:

bv20	bv50	bv100	nv9usb	nv10usb	nv200	SMART Payout	nv11

Event data format:

This event has no data associated with it

Event name	**Note Stacked**

Code	Description
0xEB	The note has exited the device on the host side or has been placed within its note stacker.

Poll with ACK event
No

Supported on:

bv20	bv50	bv100	nv9usb	nv10usb	nv200	SMART Payout	nv11

Event data format:

This event has no data associated with it

Event name	**Safe Note Jam**

Code	Description
0xEA	The note is stuck in a position not retrievable from the front of the device (user side)

Poll with ACK event
No

Supported on:

bv20	bv50	bv100	nv9usb	nv10usb	nv200	SMART Payout	nv11

Event data format:

This event has no data associated with it

Event name	**Unsafe Note Jam**

Code	Description
0xE9	The note is stuck in a position where the user could possibly remove it from the front of the device.

Poll with ACK event
No

Supported on:

bv20	bv50	bv100	nv9usb	nv10usb	nv200	nv201	SMART Payout	nv11

Event data format:

This event has no data associated with it

Event name	**Disabled**

Code	Description
0xE8	The device is not active and unavailable for normal validation functions.

Poll with ACK event
No

Supported on:

All

Event data format:

This event has no data associated with it

Event name	**Fraud Attempt**

Code	Description
0xE6	The device has detected an attempt to tamper with the normal validation/stacking/payout process.

Poll with ACK event
Yes

Supported on:

bv20	bv50	bv100	nv9usb	nv10usb	nv200	nv201	SMART Payout	nv11	SMART System	SMART Hopper

Event data format:

Banknote validators
1 byte showing the channel of the note that was in process when the fraud was detected. This will be zero if the note has not yet been recognised

SMART Hopper, SMART System, SMART Payout - Protocol version less than 6
4 bytes showing the value that was paid by the payout up until the fraud attempt occurred

SMART Hopper, SMART System, SMART Payout - Protocol versions greater or equal to 6
An array of data giving the payout status at the fraud 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 (see table below)

byte	function
0-3	value dispensed
4-6	ascii country code

...	repeat bytes 0-6 for each country in the set

Event name	**Stacker Full**

Code	Description
0xE7	The banknote stacker unit attached to this device has been detected as at its full limit

Poll with ACK event
No

Supported on:

bv20	bv50	bv100	nv9usb	nv10usb	nv200	nv201	SMART Payout	nv11

Event data format:

This event has no data associated with it

Event name	**Note Cleared From Front**

Code	**Description**
0xE1	At power-up, a note was detected as being rejected out of the front of the device. The channel value, if known is given in the data byte.

Poll with ACK event
Yes

Supported on:

bv20	bv50	bv100	nv9usb	nv10usb	nv200	nv201	SMART Payout	nv11

Event data format:

All protocol versions
1 byte showing the channel of the note rejected (0 if not known)

Event name	**Note Cleared To Cashbox**

Code	Description
0xE2	At power up, a note was detected as being moved into the stacker unit or host exit of the device. The channel number of the note is given in the data byte if known.

Poll with ACK event
Yes

Supported on:

bv20	bv50	bv100	nv9usb	nv10usb	nv200	nv201	SMART Payout	nv11

Event data format:

All protocol versions
1 byte showing the channel of the note stacked (0 if not known)

Event name	**Cashbox Removed**

Code	Description
0xE3	A device with a detectable cashbox has detected that it has been removed.

Poll with ACK event
No

Supported on:

bv50	bv100	nv200	SMART Payout	nv11

Event data format:

This event has no data associated with it

Event name	**Cashbox Replaced**

Code	Description
0xE4	A device with a detectable cashbox has detected that it has been replaced.

Poll with ACK event
No

Supported on:

bv50	bv100	nv200	SMART Payout	nv11

Event data format:

This event has no data associated with it

Event name	**Bar Code Ticket Validated**

Code	Description
0xE5	A validated barcode ticket has been scanned and is available at the escrow point of the device.

Poll with ACK event
No

Supported on:

NV200	NV201

Event data format:

This event has no data associated with it

Event name	**Bar Code Ticket Acknowledge**

Code	Description
0xD1	The bar code ticket has been passed to a safe point in the device stacker.

Poll with ACK event
Yes

Supported on:

NV200	NV201

Event data format:

This event has no data associated with it

Event name	**Note Path Open**

Code	Description
0xE0	The device has detected that its note transport path has been opened.

Poll with ACK event
No

Supported on:

NV200	

Event data format:

This event has no data associated with it

Event name	**Channel Disable**

Code	Description
0xB5	The device has had all its note channels inhibited and has become disabled for note insertion.

Poll with ACK event
No

Supported on:

bv20	bv50	bv100	nv9usb	nv10usb	nv200	nv201	SMART Payout	nv11

Event data format:

This event has no data associated with it

Event name	**Initialising**

Code	Description
0xB6	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.

Poll with ACK event
No

Supported on:

bv20	bv50	bv100	nv9usb	nv10usb	nv200	nv201	SMART Payout	nv11	SMART System	SMART Hopper

Event data format:

This event has no data associated with it

Event name	**Dispensing**

Code	Description
0xDA	The device is in the process of paying out a requested value. The value paid at the poll is given in the vent data.

Poll with ACK event
No

Supported on:

SMART Payout	SMART Hopper	SMART System	nv11

Event data format:

Protocol version less than 6
4 bytes showing the value dispensed so far at the poll point

Protocol versions greater or equal to 6
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 (see table below)

byte	function
0-3	value dispensed
4-6	ascii country code
...	repeat bytes 0-6 for each country in the set

Special note for nv11 device responses
Note that for nv11 device responses, the note value will be given in the same form as setup in the Set Value Reporting Type command: either 4 bytes for value or 1 byte for note channel.

Event name	**Dispensed**

Code	Description
0xD2	The device has completed its pay-out request. The final value paid is given in the event data.

Poll with ACK event
Yes

Supported on:

SMART Payout	SMART Hopper	SMART System	nv11

Event data format:

Protocol version less than 6
4 bytes showing the value dispensed

Protocol versions greater or equal to 6
An array of data giving the value 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 (see table below)

byte	function
0-3	value dispensed
4-6	ascii country code
...	repeat bytes 0-6 for each country in the set

Special note for nv11 device responses
Note that for nv11 device responses, the note value will be given in the same form as setup in the Set Value Reporting Type command: either 4 bytes for value or 1 byte for note channel.

Event name	**Jammed**

Code	Description
0xD5	The device has detected that coins are jammed in its mechanism and cannot be removed other than by manual intervention. The value paid at the jam point is given in the event data.

Poll with ACK event
No

Supported on:

SMART Payout	SMART Hopper	SMART System	nv11

Event data format:

Protocol version less than 6
4 bytes showing the value dispensed up to the jam point

Protocol versions greater or equal to 6
An array of data giving the value dispensed up to the jam 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 (see table below)

byte	function
0-3	value dispensed
4-6	ascii country code
...	repeat bytes 0-6 for each country in the set

Special note for nv11 device responses

Note that for nv11 device responses, the note value will be given in the same form as setup in the Set Value Reporting Type command: either 4 bytes for value or 1 byte for note channel.

Event name	**Halted**

Code	Description
0xD6	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.

Poll with ACK event
No

Supported on:

SMART Payout	SMART Hopper	SMART System	nv11

Event data format:

Protocol version less than 6
4 bytes showing the value dispensed up to the halt point

Protocol versions greater or equal to 6
An array of data giving the value dispensed up to the halt 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 (see table below)

byte	function
0-3	value dispensed
4-6	ascii country code
...	repeat bytes 0-6 for each country in the set

Special note for nv11 device responses

Note that for nv11 device responses, the note value will be given in the same form as setup in the Set Value Reporting Type command: either 4 bytes for value or 1 byte for note channel.

Event name	**Floating**

Code	Description
0xD7	The device is in the process of executing a float command and the value paid to the cashbox at the poll time is given in the event data.

Poll with ACK event
No

Supported on:

SMART Payout	SMART Hopper	SMART System

Event data format:

Protocol version less than 6
4 bytes showing the value floated to the cashbox up to the poll

Protocol versions greater or equal to 6
An array of data giving the value floated to the cashbox up to the poll 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 (see table below)

byte	function
0-3	value floated
4-6	ascii country code
...	repeat bytes 0-6 for each country in the set

Event name	**Floated**

Code	Description
0xD8	The device has completed its float command and the final value floated to the cashbox is given in the event data.

Poll with ACK event
Yes

Supported on:

SMART Payout	SMART Hopper	SMART System

Event data format:

Protocol version less than 6
4 bytes showing the value floated to the cashbox

Protocol versions greater or equal to 6
An array of data giving the value floated to the cashbox 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 (see table below)

byte	function
0-3	value floated
4-6	ascii country code
...	repeat bytes 0-6 for each country in the set

Event name	**Time out**

Code	Description
0xD9	The device has been unable to complete a request. The value paid up until the time-out point is given in the event data.

Poll with ACK event
Yes

Supported on:

SMART Payout	SMART Hopper	SMART System	nv11

Event data format:

Protocol version less than 6
4 bytes showing the value dispensed up to the time-out point

Protocol versions greater or equal to 6
An array of data giving the value dispensed up to the time-out 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 (see table below)

byte	function
0-3	value dispensed
4-6	ascii country code
...	repeat bytes 0-6 for each country in the set

Event name	**Incomplete payout**

Code	Description
0xDC	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.

Poll with ACK event
Yes

Supported on:

SMART Payout	SMART Hopper	SMART System	nv11

Event data format:

Protocol version less than 6
8 bytes showing the value dispensed and the original value requested before the power down.

Protocol versions greater or equal to 6
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 the a block of data for each of the countries (see table below)

byte	function
0-3	value dispensed
4-7	value requested
8-10	ascii country code
...	repeat bytes 0-10 for each country in the set

Special note for nv11 device responses
Note that for nv11 device responses, the note value will be given in the same form as setup in the Set Value Reporting Type command: either 4 bytes for value or 1 byte for note channel.

Event name	**Incomplete float**

Code	Description
0xDD	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.

Poll with ACK event
Yes

Supported on:

SMART Payout	SMART Hopper	SMART System	nv11

Event data format:

Protocol version less than 6
8 bytes showing the value dispensed and the original value floated before the power down.

Protocol versions greater or equal to 6
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 the a block of data for each of the countries (see table below)

byte	function
0-3	value floated
4-7	value requested
8-10	ascii country code
...	repeat bytes 0-10 for each country in the set

Event name	**Cashbox paid**

Code	Description
0xDE	This is given at the end of a payout cycle. It shows the value of stored coins that were routed to the cashbox that were paid into the cashbox during the payout cycle.

Poll with ACK event
No

Supported on:

SMART Hopper	SMART System

Event data format:

Protocol version less than 6
4 bytes showing the value of the coins routed to the cashbox during the dispense cycle

Protocol versions greater or equal to 6
An array of data giving the value routed to the cashbox during the dispense cycle 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 (see table below)

byte	function
0-3	value dispensed
4-6	ascii country code
...	repeat bytes 0-6 for each country in the set

Event name	Coin credit

Code	Description
0xDF	A coin has been detected as added to the system via the attached coin mechanism. The value of the coin detected is given in the event data.

Poll with ACK event
No

Supported on:

SMART Hopper	SMART System

Event data format:

Protocol version less than 6
4 bytes showing the value of the coin detected as added by the coin mech

Protocol versions greater or equal to 6
An array of data giving the value and country code of the coin detected as added by the coin mech

byte	function
0-3	value detected
4-6	ascii country code

Event name	**Coin mech jammed**

Code	Description
0xC4	The attached coin mechanism has been detected as having a jam.

Poll with ACK event
No

Supported on:

SMART Hopper	SMART System

Event data format:

This event has no data associated with it

Event name	**Coin mech return pressed**

Code	Description
0xC5	The attached coin mechanism has been detected as having is reject or return button pressed.

Poll with ACK event
No

Supported on:

SMART Hopper	SMART System

Event data format:

This event has no data associated with it

Event name	**Emptying**

Code	Description
0xC2	The device is in the process of emptying its content to the system cashbox in response to an Empty command.

Poll with ACK event
No

Supported on:

SMART Payout	SMART Hopper	SMART System	nv11

Event data format:

This event has no data associated with it

Event name	**Emptied**

Code	Description
0xC3	The device has completed its Empty process in response to an Empty command from the host.

Poll with ACK event
No

Supported on:

SMART Payout	SMART Hopper	SMART System	nv11

Event data format:

This event has no data associated with it

Event name	**Smart emptying**

Code	Description
0xB3	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.

Poll with ACK event
No

Supported on:

SMART Payout	SMART Hopper	SMART System	nv11

Event data format:

Protocol version less than 6
4 bytes showing the value emptied to the cashbox up to the poll

Protocol versions greater or equal to 6
An array of data giving the value emptied to the cashbox up to the poll 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 (see table below)

byte	function
0-3	value floated
4-6	ascii country code
...	repeat bytes 0-6 for each country in the set

Event name	**Smart emptied**

Code	Description
0xB4	The device has completed its Smart Empty command. The total amount emptied is given in the event data.

Poll with ACK event
Yes

Supported on:

SMART Payout	SMART Hopper	SMART System	nv11

Event data format:

Protocol version less than 6
4 bytes showing the value emptied to the cashbox up to the poll

Protocol versions greater or equal to 6
An array of data giving the value emptied to the cashbox up to the poll 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 (see table below)

byte	function
0-3	value emptied
4-6	ascii country code
...	repeat bytes 0-6 for each country in the set

Event name	**Coin mech error**

Code	Description
0xB7	The attached coin mechanism has generated an error. Its code is given in the event data.

Poll with ACK event
No

Supported on:

SMART Hopper	SMART System

Event data format:

This event has no data associated with it

Event name	**Note stored in payout**

Code	Description
0xDB	The note has been passed into the note store of the payout unit.

Poll with ACK event
No

Supported on:

SMART Payout	nv11

Event data format:

Special note for nv11 device responses
Note that for nv11 device responses, the note value will be given in the same form as setup in the Set Value Reporting Type command: either 4 bytes for value or 1 byte for note channel.

Event name	**Payout out of service**

Code	Description
0xC6	This event is given if the payout goes out of service during operation. If this event is detected after a poll, the host can send the ENABLE PAYOUT DEVICE command to determine if the payout unit comes back into service.

Poll with ACK event
No

Supported on:

SMART Payout	nv11

Event data format:

This event has no data associated with it

Event name	**Jam recovery**

Code	Description
0xB0	The SMART Payout unit is in the process of recovering from a detected jam. This process will typically move five notes to the cash box; this is done to minimise the possibility the unit will go out of service

Poll with ACK event
No

Supported on:

SMART Payout

Event data format:

Protocol versions greater or equal to 7 only

Event name	**Error during payout**

Code	Description
0xB1	Returned if an error is detected whilst moving a note inside the SMART Payout unit. The cause of error (1 byte) indicates the source of the condition; 0x00 for note not being correctly detected as it is routed to cashbox or for payout, 0x01 if note is jammed in transport. In the case of the incorrect detection, the response to Cashbox Payout Operation Data request would report the note expected to be paid out.

Poll with ACK event
Yes

Supported on:

SMART Payout

Event data format:

Protocol versions greater or equal to 7
An array of data giving the value dispensed up until the error for all countries supported in the dataset plus a final byte giving an indication of the type of error which caused the problem. The first byte gives the number of countries in the set the a block of data for each of the countries (see table below)

byte	function
0-3	value dispensed
4-6	ascii country code
...	repeat bytes 0-6 for each country in the set
n	Final byte is an error code - 0x00 for note not being correctly detected as it is routed. 0x01 for note jammed in transport.

Event name	**Note transfered to stacker**

Code	Description
0xC9	Reported when a note has been successfully moved from the payout store into the stacker cashbox.

Poll with ACK event
Yes

Supported on:

SMART Payout	nv11

Event data format:

Protocol versions greater or equal to 6
An array of data giving the value and country code of the note moved from pay-out store into the stacker cashbox

byte	function
0-3	value
4-6	ascii country code

Special note for nv11 device responses
Note that for nv11 device responses, the note value will be given in the same form as setup in the Set Value Reporting Type command: either 4 bytes for value or 1 byte for note channel.

Event name	**Note held in bezel**

Code	Description
0xCE	Reported when a dispensing note is held in the bezel of the payout device.

Poll with ACK event
No

Supported on:

SMART Payout	nv11

Event data format:

Protocol versions greater or equal to 8
An array of data giving the value and country code of the note currently held in the bezel after a pay-out

byte	function
0-3	value
4-6	ascii country code

Special note for nv11 device responses
Note that for nv11 device responses, the note value will be given in the same form as setup in the Set Value Reporting Type command: either 4 bytes for value or 1 byte for note channel.

Event name	**Note paid into store at power-up**

Code	Description
0xCB	Reported when a note has been detected as paid into the payout store as part of the power-up procedure.

Poll with ACK event
Yes

Supported on:

SMART Payout	nv11

Event data format:

Protocol versions greater or equal to 8
An array of data giving the value and country code of the note detected as moved into the pay-out store at power-up

byte	function
0-3	value
4-6	ascii country code

Special note for nv11 device responses
Note that for nv11 device responses, the note value will be given in the same form as setup in the Set Value Reporting Type command: either 4 bytes for value or 1 byte for note channel.

Event name	**Note paid into stacker at power-up**

Code	Description
0xCA	Reported when a note has been detected as paid into the cashbox stacker as part of the power-up procedure.

Poll with ACK event
Yes

Supported on:

SMART Payout	nv11

Event data format:

Protocol versions greater or equal to 8
An array of data giving the value and country code of the note detected as moved into the pay-out store at power-up

byte	function
0-3	value
4-6	ascii country code

Special note for nv11 device responses
Note that for nv11 device responses, the note value will be given in the same form as setup in the Set Value Reporting Type command: either 4 bytes for value or 1 byte for note channel.

Event name	**Note Dispensed at power-up**

Code	Description
0xCD	Reported when a note has been dispensed as part of the power-up procedure.

Poll with ACK event
Yes

Supported on:

nv11

Event data format:

Protocol versions greater or equal to 6
An array of data giving the value and country code of the note detected as moved into the pay-out store at power-up

byte	function
0-3	value
4-6	ascii country code

Special note for nv11 device responses
Note that for nv11 device responses, the note value will be given in the same form as setup in the Set Value Reporting Type command: either 4 bytes for value or 1 byte for note channel.

Event name	**Note float removed**

Code	Description
0xC7	Reported when a note float unit has been detected as removed from its validator.

Poll with ACK event
No

Supported on:

nv11

Event data format:

This event has no data associated with it

Event name	**Note float attached**

Code	Description
0xC8	Reported when a note float unit has been detected as removed from its validator.

Poll with ACK event
No

Supported on:

nv11

Event data format:

This event has no data associated with it

Event name	**Device full**

Code	Description
0xC8	This event is reported when the Note Float has reached its limit of stored notes. This event will be reported until a note is paid out or stacked.

Poll with ACK event
No

Supported on:

nv11

Event data format:

This event has no data associated with it

back to index

Command name	Code dec	Code hex
Host Protocol Version	**006**	**0x6**

Supported on devices:

nv9usb	nv10usb	bv20	bv50	bv100	nv200	SMART Hopper	SMART Payout	SMART System	nv11

Encryption Required
no

Description

Dual byte command, the first byte is the command; the second byte is the version of the protocol that is implemented on the host. So for example, to enable events on BNV to protocol version 6, send 06, 06. The device will respond with OK if the device supports version 6, or FAIL (0xF8) if it does not.

Data byte is the protocl version to set the device to. In this case, 6

dec

127	128	002	006	006	036	020

hex

7F	80	02	06	06	24	14

Response

Device responds with OK if version is supported.

dec

127	128	001	240	035	128

hex

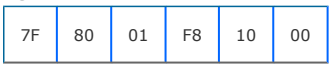
7F	80	01	F0	23	80

Device responds with FAIL if the version is not supported.

dec

127	128	001	248	016	000

hex



back to index

Command name	Code dec	Code hex
Setup Request	**005**	**0x5**

Supported on devices:

nv9usb	nv10usb	bv20	bv50	bv100	nv200	SMART Hopper	SMART System	nv11

Encryption Required
no

Description

The device responds with an array of data the format of which depends upon the device, the dataset installed and the protocol version set.

Parameters

This command has no parameters

Command packet example:

dec

127	128	001	005	029	130

hex

7F	80	01	05	1D	82

Banknote validator format response

BNV data	Response offset	Size	Notes
Unit type	0	1	0 = Banknote validator, 6 = SMART payout fitted, 7 = Note Float fitted.
Firmware version	1	4	Ascii data of device firmware (eg 0123)
Country code	5	3	The ascii code of the device dataset (eg EUR)

Value multiplier	8	3	The value to multiply the individual channels by to get the full value. If this value is 0 then it indicates that this is a protocol version 6 or greater compatible dataset where the values are given in the expanded segment of the return data.
Number of channels	11	1	The highest channel used in this device dataset [n] (1-16)
Channel value	12	n	A variable size array of byes, 1 for each channel with a value from 1 to 255 which when multiplied by the value multiplier gives the full value of the note. If the value multiplier is zero then these values are zero.
Channel security	12 + n	n	An obsolete value showing security level. This is set to 2 if the value multiplier is > 0 otherwise 0.
Real value multiplier	12 +(n * 2)	3	The value by which the channel values can be multiplied to show their full value e.g. 5.00 EUR = 500 EUR cents
Protocol version	15 + (n * 2)	1	The current protocol version set for this device
Expanded channel country code	16 + (n * 2)	n * 3	Three byte ascii code for each channel. This allows multi currency datasets to be used on SSP devices. These bytes are given only on protocol versions >= 6.
Expanded channel value	16 + (n * 5)	n * 4	4 bytes for each channel value. These bytes are given only on protocol versions >= 6.

Smart Hopper/System format response

SMART Hopper data	Response offset	Size	Notes
Unit type	0	1	3 = Smart Hopper
Firmware version	1	4	Ascii data of device firmware (eg 0123)
Country code	5	3	The ascii code of the device dataset (eg 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 for values	10 + (n *2)	n * 3	The 3 byte ascii country code for each of the coin denominations. This is given only when the device has been set to protocol versions >= 6.

Response

A response for a banknote validator protocol version 4, firmware 1.00 with EUR 5,10,20 dataset.

dec

127	128	023	240	000	048	049	048	048	069	085	082	000	000	001	003	005	010	020	002	002	002
000	000	100	004	042	037																

hex

7F	80	17	F0	00	30	31	30	30	45	55	52	00	00	01	03	05	0A	14	02	02	02
00	00	64	04	2A	25																

ascii

	.	.	.	.	0	1	0	0	E	U	R	.	.	.	.	.	.	.	.	.
.	.	d	.	.	.															

A response for a bill validator with SMART payout fitted. Firmware 6.00, protocol version 7, EUR dataset 5,10,20. Expanded values.

dec

127	128	044	240	006	048	054	048	048	069	085	082	000	000	000	003	000	000	000	000	000	000
000	000	100	007	069	085	082	069	085	082	069	085	082	005	000	000	000	010	000	000	000	020
000	000	000	190	102																	

hex

7F	80	2C	F0	06	30	36	30	30	45	55	52	00	00	00	03	00	00	00	00	00	00
00	00	64	07	45	55	52	45	55	52	45	55	52	05	00	00	00	0A	00	00	00	14
00	00	00	BE	66																	

ascii

	.	.	.	.	0	6	0	0	E	U	R	.	.	.	.	.	.	.	.	.
.	.	d	.	E	U	R	E	U	R	E	U	R	.	.	.	.	.	.	.	.	.
.	.	.	.	f																	

back to index

Command name	Code dec	Code hex
Display Off	**004**	**0x4**

Supported on devices:

nv9usb	nv10usb	nv200	nv11

Encryption Required
no

Description

This command will force the device bezel to not be illuminated even if the device is enabled.

Parameters

This command has no parameters

Command packet example:

dec

127	128	001	004	024	002

hex

7F	80	01	04	18	02

Response

Device responds with OK if the command si supported.

dec

127	128	001	240	035	128

hex

7F	80	01	F0	23	80

back to index

Command name	Code dec	Code hex
Display On	**003**	**0x3**

Supported on devices:

nv9usb	nv10usb	nv200	nv11

Encryption Required
no

Description

Use this command to re-enabled a disabled bezel illumination function (using the Display Off command). The Bezel will only be illuminated when the device is enabled even if this command is sent.

Parameters

This command has no parameters

Command packet example:

dec

127	128	001	003	009	130

hex

7F	80	01	03	09	82

Response

The device responds with OK.

dec

127	128	001	240	035	128

hex

7F	80	01	F0	23	80

back to index

Command name	Code dec	Code hex
Set Channel Inhibits	**002**	**0x2**

Supported on devices:

nv9usb	nv10usb	bv20	bv50	bv100	nv200	nv11

Encryption Required
no

Description

Variable length command, used to control which channels are enabled. The command byte is followed by 2 data bytes, these bytes are combined to create the INHIBIT_REGISTER, each bit represents the state of a channel (LSB= channel 1, 1=enabled, 0=disabled). At power up all channels are inhibited and the validator is disabled.

This example shows a command to enable channel 1,2 and 3 for note acceptance. The other channels are inhibited.

dec

127	128	004	002	240	007	000	112	055

hex

7F	80	04	02	F0	07	00	70	37

Response

The device responds with OK.

dec

127	128	001	240	035	128

hex

7F	80	01	F0	23	80

back to index

Command name	Code dec	Code hex
Reset	**001**	**0x1**

Supported on devices:

nv9usb	nv10usb	bv20	bv50	bv100	nv200	SMART Hopper	SMART Payout	SMART System	nv11

Encryption Required
no

Description

Command to instruct the slave to perform a hard reset at any point within its operational status.

Parameters

This command has no parameters

Command packet example:

dec

127	128	001	001	006	002

hex

7F	80	01	01	06	02

Response

Slave will respond with OK and then perform its reset.

dec

127	128	001	240	035	128

hex

7F	80	01	F0	23	80