MONET +
Trust in digital

TECHNICAL DOCUMENT

MONET+ B-PROTOCOL

SUBMITTED BY: **ISSUED BY:**	**MONET+,a.s.** Za Dvorem 505, Zlín - Štípa
REVISION DATE:	**27.02.2024**
VALID TO:	*Next version*
VERSION NO:	**1.27.20**

1. CONTENTS

2. DOCUMENT HISTORY

VERSION	DATE	AUTHOR	DESCRIPTION
1.6	05.05.2006	ZSK	Purchase with cashback
1.7	10.05.2010	ZSK	Invoice number 1; Currency Code; Sequence ID; Merchant ID
1.8	01.06.2010	ZSK	Serial port; UDP/IP network clarification; example trace
1.9	09.09.2010	ZSK	Header length and CRC clarification
1.10	28.03.2010	ZSK	Printer-less terminal; correction of Currency; Seq-Number
1.11	29.09.2011	ZSK	GPE mode compatibility
1.12	25.01.2012	ZSK	TCP/IP clarification
1.12b	27.01.2012	ZSK	TCP/IP terminal detection
1.13	06.02.2012	ZSK	Added FID 'g'; correct FID 'T' length description
1.14	21.02.2012	ZSK	Added new service transaction types 80, 81 and 82
1.14b	13.03.2012	ZSK	Added virtual terminals/merchant into application info
1.15	17.04.2012	ZSK	Added explicit confirmation by Flags 0x8000
1.15b	27.04.2012	ZSK	Added 'PAN-hint' section; some formatting cleanups
1.16	18.05.2012	ZSK	Added new service 83; add FID 'm'; implement close-batch 'ticket'
1.17	16.08.2013	ZSK	Added Invoice 2 field; echo Invoice(1|2) in response
1.18	14.01.2014	DPR	Added FID T to response
1.18a	14.02.2014	DPR	Added Note to 5.4 FID sorting
1.18b	24.06.2014	DPR	Added whitelist of cards for checking availability of discount per card
1.20	04.09.2014	DPR	Add section 10. Keep-alive
1.20a	18.09.2014	DPR	Added note to 8. GPE compatibility mode
1.25	20.02.2015	DPR	Skipped version numbers of document describing B-protocol for terminal used in public transport. Added note for FID P
1.25.1	16.02.2015	DPR	Added FID J
1.25.2	19.03.2015	DPR	Added operation "cash advance" in FID T
1.25.3	26.03.2015	DPR	Added examples of currency codes
1.25.4	20.04.2015	DPR	Support for store offline transaction on cash register memory
1.25.5	23.06.2015	APO	Terminal as TCP server chapter 7.4
1.25.6	24.07.2015	DPR	Tipo revision
1.25.7	25.07.2015	DPR	Add note to FID I
1.25.8	28.08.2015	DPR	FID 9 G Gastro data, FID L - Balance
1.25.9	22.09.2015	DPR	FID F change of description
1.25.10	02.10.2015	DPR	Added graph of payment flow to the appendix 4.

1.25.11	22.10.2015	RPS	Added notice to FID 9G
1.25.12	20.10.2016	DPR	List of FID used in transactions
1.26.1	12.04.2017	DPR	The command TMS Call
1.26.2	12.05.2017	DPR	N protocol
1.26.3	19.05.2017	DPR	FID 9p - Preauth data, Balance inquiry command
1.26.4	20.09.2017	DPR	Balance request
1.26.5	06.10.2017	DPR	Some tipo correction
1.26.6	24.10.2017	DPR	Add Notes to B-protocol header
1.26.8	26.06.2018	DPR	Typo correction
1.26.9	27.08.2018	GBI	FID D possibilities of use
1.26.10	22.10.2018	ENE	Suppression of report on passivation split payment
1.27	06.03.2019	GBI	Converting document into GDocs
1.27.1	19.06.2019	GBI	New FID b - Couvert added as part of new tip functionality.
1.27.2	11.07.2019	GBI	New FID 9g added
1.27.3	28.11.2019	GBI	FID 9g field extended with additional information structure and communication example In chap. 5.1 added flags: ● Bit 9: 0 = does not show Progress status, 1 = sends Progress status ● Bit 10: 0 = no TOKEN, 1 = sends a TOKEN
1.27.4	03.02.2020	GBI	Extended information about FID 9g - Progress status (chap. 5.2.28)
1.27.5	24.04.2020	GBI	Fixed a typo on page 9 in the bin number
1.27.6	17.06.2020	GBI	Added new preauthorization operations and completion of preauthorization in chap. 5.4. and 5.2.11 FID T
1.27.7	24.07.2020	GBI	The FID 9P - Partial approval field is now mandatory for payments/sales and pre-authorization.
1.27.8	15.10.2020	GBI	● Removed chapter 8. GPE compatible mode ● Removed chapter 7.1 Serial port
1.27.9	01.12.2020	GBI	Added information to chapter 7.3 TCP / IP / Ethernet - terminal as server
1.27.10	15.01.2021	GBI	Adjust the length of the field FID D - Alternate ID
1.27.11	22.03.2021	GBI	Editing text in chap. TCP / IP / Ethernet - terminal as server
1.27.12	26.03.2021	GBI	● Field length adjustment in FID I - Totals/Batch ● In the chapter "TCP / IP / Ethernet - terminal as server" added the possibility of not filling in the IP address of the cash register ● Added APPENDIX 5: Payment transactions with explicit confirmation and APPENDIX 6: FAQ
1.27.13	08.04.2021	GBI	Field length change for FID B and FID C to max 10 bytes
1.27.14	16.04.2021	GBI	Chapter 5.3.3. and 7.2. extended by the case of TCP connection failure
1.27.15	09.03.2022	GBI	● Added chapter 6.3 Busy terminal (RC -30)

			- Added information that a keep-alive (ENQ) default timeout of 15 seconds is set for the application on the Tetra platform.
1.27.16	14.03.2022	GBI	Fixed information that keep-alive (ENQ) default timeout of 15 seconds is set for the application on **all** platforms.
1.27.17	08.07.2022	AAD	Removed FID K
1.27.18	23.08.2022	AAD	- Added max value in FID B - Specified content of FID g while RC value is negative - Added FID 9p reversal info - Removed duplicity - section 5.2.14
1.27.19	18.12.2023	KLA	Added Card verify (06) operation
1.27.20	27.02.2024	MVC	- Clarification of 9t - Terminal ticket information in section 8.1.2 - Description of T84 in section 5.3.2

Table 1 *Version history*

3. LIST OF TERMS / SHORTCUTS

CAD	Card acceptance device = Terminal = POS
ECR	Electronic cash register
FID	Field identifier of optional data fields
POS	Point of sale - place of acceptance of payment cards; also used as a term for the terminal
POSIS	Point of Sale Information System = Authorization server

4. PURPOSE OF DOCUMENT

The present document is intended for companies which implement the B-protocol to communicate between the cash register or validator and the card acceptance terminal.

The present document defines the B-protocol interface on the part of MONET+. The document includes structure of message header, permitted message types, the fields which are used in them and contents admissible for such fields.

Because of the flexible nature of the B-protocol message, partners must collaborate on the transactions to be supported, and the optional elements to be used in those transactions, which will be specified in the separate extension document**.**

5. MESSAGE STRUCTURE

The B-Protocol is a format used as a communications protocol between cash register and terminal. The main purpose of the B-protocol is to describe the messages exchanged between the cash register and the terminal, to define all permissible transactions including all the data elements that may appear in the request and reply message pairs, and also to describe the flow of messages between the cash register and the payment device.
The message is entered by STX (0x02) and consists of a fixed length header followed by any number of optional data fields.
Each field optional in the message is marked with one FID (field identifier) character that is right before the data. Before each FID there is a special character separator field (0x1C). Some optional fields may contain sub fields that are labeled with one SFID (subfield identifier) located directly in front of the data column. There is a special character group separator (0x1D) before each SFID. Header and Data must be in ASCII format. The end of the message is marked with a special ETX (0x03).

<STX>[Header][Data]<ETX>

The following diagram summarizes the major components of the message.

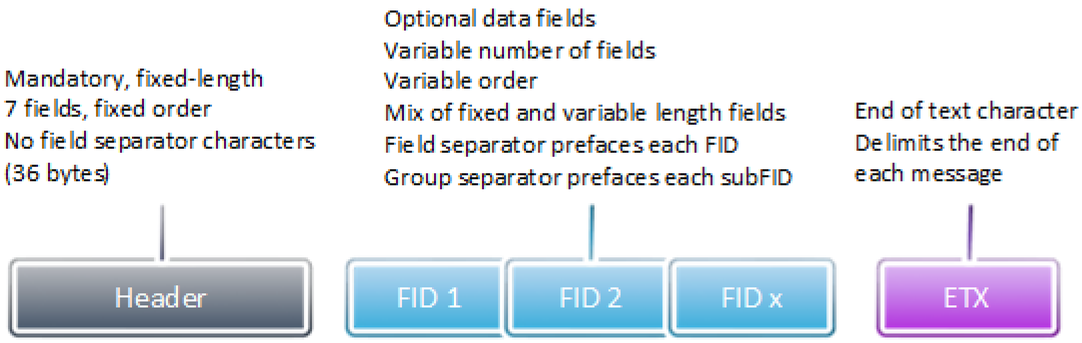


Diagram 1 *Message structure*

Each Fid may be a primitive Fid or a composite Fid. A primitive Fid contains a Fid Id and the Fid value. A Composite Fid contains a Fid ID, but then an array of Sub-Fids. Each Sub-Fid has a preceding control character with the hexadecimal value 1D. Each Sub-Fid is in turn is a Primitive fid (See below).

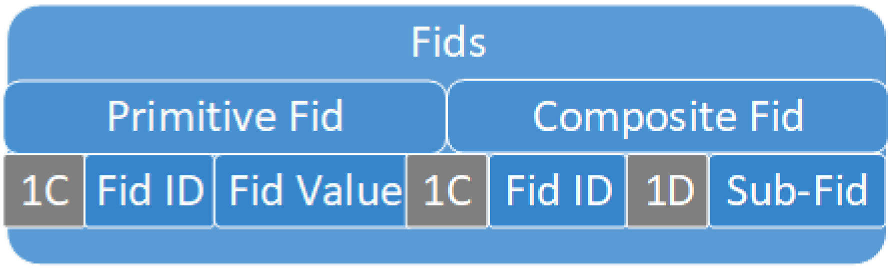


Diagram 2 *Optional fields format*

Legend of control characters:

Hex	Symbol	Type	Description
2	STX	Communication Control	Start of Text
3	ETX	Communication Control	End of Text
1C	FS	Information Separator	File Separator
1D	GS	Information Separator	Group Separator

Table 2 *Control characters*

5.1. Message header

Message header is the first part of the message. It informs both members of communication about protocol, Pos ID, date and time of transaction, len of data and can contain many additional information in field Flag.

Header structure (fix length 36B)

Field	Format	Length (B)	Field name	Value
PT	AlfaNumeric	2	Protocol Type	B0 – BF
PV	AlfaNumeric	2	Protocol Version	01 – FF
TID	AlfaNumeric	8	Terminal ID	Described later
DT	Numeric	12	Transaction DateTime	Current date+time
FLG	AlfaNumeric	4	Flag	Described later
DLN	AlfaNumeric	4	Optional Data Len	0000h – 01FFh
CRC	AlfaNumeric	4	Standard CRC16	Counted value

Table 3 *Header structure*

Example: [B1 | 01 | A123XX01 | 02 11 25 15 47 21 | 00 00 | 00 00 | A5 A5]

Protocol Type

Possible value is “B0” – “BF” (B means protocol for CADxECR).

- Left byte B means protocol ID
- Right byte 0 reserved for ActivityInfoMessage
- Right byte 1-F means message number
  - 1 - Transaction Request (cash register -> terminal)
  - 2 - Transaction Response (terminal -> cash register)
  - 3 - Ticket Request (cash register -> terminal)
  - 4 - Ticket Response (terminal -> cash register)

Protocol Version

Possible protocol version number value is “01” – “FF”.

Terminal ID

Unique 8bytes long identificator, significant only for POS messages.

Transaction Date&Time

Format YYMMDDHHMMSS = year|month|day|hour|min|sec

Flag

Used for different information, every bit can be used : "0000" – "FFFF"

- Bit 0 : 0 = NO SIGNATURE required, 1 = SIGNATURE required
- Bit 1 : 0 = NO TICKET, 1 = ECR must print TICKET
- Bit 2-8 : not used
- Bit 9: 0 = does not show Progress status, 1 = sends Progress status
- Bit 10: 0 = no TOKEN, 1 = sends a TOKEN
- Bit 11: 1 = offline trn. - cash register can read out date of the transaction
- Bit 12: 1 = amount of transaction demands verify id of customer
- Bit 13: 1 = cash register supports the Keep-alive mechanism for TCP connections
- Bit 14: 1 = check whitelist before payment
- Bit 15: 1 = explicit confirmation from cash register required

Additional information structure:

```
FLAG_SIGN                   0x0001
FLAG_TICKET                 0x0002
FLAG_EX_PROGRESS            0x0200        // if want B0 with progress status
FLAG_TOKEN                  0x0400        // if want TOKEN(family)
FLAG_OFFLINE_DATA           0x0800               // for last offline
FLAG_USER_ID                0x1000        // need user ID
FLAG_KEEP_ALIVE             0x2000        // supports keep alive
FLAG_SPLIT_SALE             0x4000        // split sale support
FLAG_CONFIRM                0x8000        // want confirmation
```

Optional Data Len

Depends on the used optional data part: "00FF" = 255B

CRC – standard CRC16

Check redundancy control - currently unused, set to constant "A5A5"

Sample of B-protocol header:

```
0x0000: 02 42 32 30 31 54 31 53 54 30 32 33 30 31 37 31 = '.B201T1ST0230171'
0x0010: 30 32 34 31 35 35 36 34 32 38 30 30 32 30 30 46 = '024155642800200F'
0x0020: 43 41 35 41 35                                  = 'CA5A5           '
```

Flag 8002 = 8002 hex = 1000000000000010 bin
Bit 15 and bit 1 are set.

5.2. Optional data fields

After the message header, the remaining portion of the message consists of a series of optional data fields. Optional data fields can be included in requests from the CAD and responses from the ECR for each transaction type and are identified in the system by Field Identifiers (FIDs).

Message Data is done by Optional Data Fields. Every Optional Data Field shall start with Field Separator **FS** (1Ch) as you can see below.

```
<FS>[ field_ID | field_DATA ] <FS>[ field_ID| field_DATA ] ……
```

Example: `<FS>B1000 -> amount = 10.00 Kč`

There is no rule for sorting FIDs. The way of sorting FID may vary in every message and may not be the same in a future revision. Parsing of FID is based on their TAG.

The optional data fields are described below according to their FIDs. The table is organized in alphabetical order with capital letters listed before lowercase letters, and numbers listed after letters. The table lists the FID, its picture clause, the length of the field (in bytes) in the message, and its associated field name. In addition, a check mark () appears in the RQST or RESP fields if an optional data field is available for requests or responses, respectively.

FID	Picture	Length (B)	Field Name	Rqst	Resp
B	PIC 9(8)v99	1-10	Amount 1	✓	
C	PIC 9(8)v99	1-10	Amount 2	✓	
D	PIC 9(2)	1-2	Alternate ID	✓	
F	PIC X(8)	8	Approval code	✓ 1)	✓
I/E	PIC 9(3)	3	Currency code	✓ 2)	
J	PIC X(16)	1-16	Card brand		✓
L	PIC 9(16)v99	1-18	Balance		✓
P	PIC 9(19)	13-19	PAN	✓ 3)	✓
R	PIC 9(3)	3	Response Code	✓	
S	PIC 9(10)	1-10	Invoice Number 1	✓	✓
T	PIC 9(2)	2	Transaction Type	✓	✓
b	PIC 9(18)	1-18	Couvert	✓	✓
i	PIC 9(9)	9	Sequence ID		✓
g	PIC X(40)	1. 1. 1940	Server message		✓
l	PIC X(75)	75	Totals/Batch1		✓
m	PIC X(75)	75	Totals/Batch2		✓
1	PIC X(82)	1-82	Card Token		✓
9		variable		✓	✓
9P	PIC 9(1)	1	Partial approvement	✓	
9p	PIC X(90)	4-90	Preauth. data	✓	✓
9S	PIC X(20)	1-20	InvoiceNumber 2	✓	✓
9T	PIC X(43)	1-43	Ticket Line Text	✓	✓

9t	PIC X(1)	1	Ticket Type	✓	✓
9y	PIC X(32)	10-32	Transaction name		✓
9X	PIC H(4)	4	Offset of transaction data	✓	✓
9x	PIC X(400)	1-400	Data of trn. (Base64)		✓
9G	PICX(40)	5-36	Gastro data	✓	✓
9g	PIC H(4)	4	Progress status	✓	✓

Table 4 *Table of possible fids*

1) For reversal transaction
2) In case of multi currency terminals
3) Could be 1 character in special case of transaction

In addition to the FIDs, there are so called Subfids (eg: 9S). Subfid format is different from FID, using <GS>.

Example: `<FS> 9 <GS> S12345678901234567890`

When multiple sub-fids are used within FID, it is possible to subdivide sub-fid using <GS> or subfid including its Fid.

Example: `<FS> 9 <GS> t1line1 <GS> t1line2 or <FS> 9 <GS> t1line1 <FS> 9 <GS> t1line2`

Descriptions of FIDs are included on this and the following pages in the format listed below. Explanations of the headings are included in the format.

5.2.1. FID B - Amount 1

Amount 1 is the primary amount field. For transactions that involve one amount, this is the transaction amount. For transactions that involve more than one amount, this is the original or total amount. This is the amount the customer will be charged. In case of purchase with cashback transaction, this is the sum of the amount of sale and the amount of the cashback. Amount of the transaction in supported currency is in the minimum unit of the currency. For example, if the currency is CZK 10.00, amount = “1000”; EUR 1.50 = “150” (without decimal point).
The application uses the integer data type. In practice, this means that a maximum value of 2 147 483 647 can be used in FID B.

Request: Required for the transactions. Variable length of 1 to 10 bytes.

Response: Optional. Fixed length of 1 to 10 bytes.

5.2.2. FID C - Amount 2

Amount 2 is the secondary amount field. For transactions with two amounts involved, Amount 2 is the revised amount. For transactions with cash back, Amount 2 is the cash back amount. Currently used only in the purchase with cashback transaction.

Request: Required for purchase-with-cashback transaction. Variable length of 1 to 10 bytes.
Response: Optional. Fixed length of 1 to 10 bytes.

5.2.3. FID D - Alternate ID

Alternate ID - Index of alternate Merchant/Terminal ID used during the transaction. Indexing can take place based on various parameters. Alternate ID can be used to:

- Terminal differentiation (under one dealer of multiple terminals at a shop)
- Definition of the purpose of the payment (eg fines)
- Payment of pledges (pseudo-light virtualization - TID virtualization) - Daughter index value - 1

Request: Optional. Length 1-2 bytes (possible value 0-10; limit of virtual IDs is 10). Required for the financial transaction on multi-merchant enabled terminals, ignored otherwise.

Response: Optional, fixed length of 10 bytes. Format #:MERCHANTID

5.2.4. FID F - Approval Code

The **Approval Code** represents a unique alphanumeric string generated by the authorizer for the transaction.

Request: Optional. Fixed length of 8 bytes. In a request, it is present for the transaction authorized by a terminal (purchase offline, refund). It is also used as the key for follow-up transactions (Completion before authorization).

Response: Optional. Fixed length of 8 bytes. If this field is included in the request message, then it can be echoed in the response. If this field is not included in the request message, it can be generated by the authorizer.

5.2.5. FID I/E - Currency code

The **currency code** used during the transaction. For historical reasons, the FID 'I' is supported, although FID 'E' is preferred.

Note: Supported currencies depend on the parameters of the terminal and a number representing the currency is based on ISO 4217.

Currency	Alphabetic Code	Numeric Code
Czech Koruna	CZK	203
Euro	EUR	978
US Dollar	USD	840
Pound Sterling	GBP	826
Zloty	PLN	985
Forint	HUF	348

Table 5 *List of currencies*

For more currency codes see www.currency-iso.org (ISO4217).

Request: Optional. Fixed length of 3 bytes. Required for the financial transaction on multi-currency enabled terminals, ignored otherwise.

Response: Not used.

5.2.6. FID J - Card brand

This field contains the name **brand cards**. The names of brands, is dependent on the settings of the file of the parameters terminal.

Request: Not used.

Response: Optional. Variable length of 1 to 16 bytes.

5.2.7. FID L - Balance

This field contains the balance of the card. Must be supported by the authorization server.

Request: Not used.

Response: Optional. Variable length of 1 to 18 bytes.

5.2.8. FID P - Primary Account Number

Primary Account Number (PAN).

Request: Not used, except for REFUND. If present in a REFUND transaction, it provides the 'hint' of the original card for the terminal. Should be the same format (masked), as the original transaction FID P received from the terminal.

Extension: If it contains only "-" terminal prompts you to manually enter the card number and expiration date.

Response: Variable len up to 19 bytes. Will be masked off, by the payment industry standards!

5.2.9. FID R - Response Code

Response Code informs the ECR about the result of the transaction.
Values of possible codes are in Appendix “Response Codes”.

- If the first char is '-', the next 2 digits mean the response code from the terminal.
- 3 digit responses are from server

Request: Not used.

Response: Fixed len 3 bytes.

5.2.10. FID S - Invoice Number

The **Invoice Number** enables a terminal to submit a unique stamp to further identify a transaction. Also known as '**variable symbol**'.

Request: Optional for the financial transaction. Variable length of 1 to 10 bytes. Used as a unique stamp of the transaction, generated by a merchant. Will be propagated into the settlement.

Response: Echoed from request.

Remark: Invoice 2 takes precedence over Invoice 1. If both are included in the request, only Invoice 2 is used and returned back.

5.2.11. FID T - Transaction Type

These codes represent individual types of transactions in FID T.

Transaction type	Value
NORMAL PURCHASE (Sale)	00
PREAUTHORIZATION	01
COMPLETION OF PREAUTHORIZATION	02
MERCHANDISE RETURN (Refund)	04
CASH ADVANCE	05
CARD VERIFY	06
BALANCE INQUIRY	07
PURCHASE WITH CASHBACK	08
REVERSAL	10
CLOSE TOTALS	60
SUBTOTALS	65
HANDSHAKE	95
GET APP INFO	80
PASSIVATE	81
GET LAST TRANS	82
GET LAST SUMS	83
GET APP CONFIG	84
TMS CALL	90

Table 6 *Transaction types*

Request: Fixed len of 2 bytes.

Response: Echoed.

5.2.12. FID b - Couvert

This FID is for tipping, but the cashier can also send a special value of '-', indicating that it supports tipping on the terminal and thus an 'increased' amount (FID 'B') and a specific amount indicating tips (FID 'b').

Type **N18** or "**-**"**.** The cashier can send a special value of '-' to indicate that a pinpad tip can be entered and returned to the cash register. Attention, if you enter a tip, change the FID 'B' total amount - the tip is always counted (as well as the cashback amount).

5.2.13. FID i - Sequence Number

The **sequence ID** is a 'unique' serial number of the transaction, generated by the terminal. Can be used to identify transactions. Doesn't propagate into the settlement!

Composition of 3 fields:

- Shift number 9(3) - incremented by each 1000th batch.
- Batch number 9(3) - incremented by each close batch.
- Sequence number 9(3) - incremented with each transaction, cleared by close batch.

Request: Not used.

Response: Fixed length of 9 bytes. Generated for the financial transactions.

5.2.14. FID g - Server message

The **Server message** is used in case of the denial of the transaction by the server, this field can carry an additional text message clarifying the server's reason to deny. FID g is optional in the case of negative RC.

Request: Not used.

Response: Variable length 1 to 48 bytes, should contain only unaccented characters.

5.2.15. FID l - Totals/Batch

The **Totals/Batch (TOTALS1)** field consists of a group of fields representing batch totals as accumulated by the terminal. This field includes a shift and batch ID, along with counts and amounts of debits, credits, and adjustments. The authorization center saves these totals and the totals accumulated by the authorization center in the POS (ECR) Transaction Log. These totals contain a sign character (+ or -) in the first byte of the amount fields. The format is as follows:

- Shift number 9(3) – Shift number
- Batch number 9(3) – Batch number
- Number of debits in batch 9(4)

- Amount of debits in batch S9(15)v99 (18 characters)
- Number of credits in batch 9(4)
- Amount of credits in batch S9(15)v99 (18 characters)

Example: "001001000 2+00000000000020000 0001+00000000000001000"

2 transactions debit (Sale) 200.00
1 transaction credit (Refund) 10.00

Request: Not used.
Response: Optional. Fixed length of 50 bytes.

5.2.16. FID m - TOTALS2

If the totals between the terminal and the bank doesn't match, this field **TOTALS2** carries the terminal totals, and the **FID l** carries the bank totals.
In case the totals match, this field is not used.

Request: Not used.
Response: Optional. Fixed length of 50 bytes. See **FID l**.

5.2.17. FID 1 - Card token

Card token - also known as '**variable symbol**'.

Request: Optional for the financial transaction. Variable length of 1 to 20. Used as a unique stamp of the transaction, generated by a merchant. Will be propagated into the settlement.

Response: Echoed from request.

Remark: Invoice 2 takes precedence over Invoice 1. If both are included in the request, only Invoice 2 is used and returned back.

5.2.18. FID 9 - Customer SubFIDs

The **Customer SubFIDs** field is reserved for future use by customers. If used, it will be specified in the **B-Protocol_customization_appendix**.

Request: Optional. Variable length of 64 bytes.
Response: Not applicable.

5.2.19. FID 9P - Partial approvement

The **Partial approval** is used to indicate the checkout supports partial authorization. Ie., that the payment may be paying a smaller amount than originally requested by FID B.
In the case of partial authorization, the return code FID is R = 10 and in FID B the answer is the value of the amount paid by this transaction.

Request: Mandatory for financial transactions for which the cashier allows partial payment. Length 1B value '1'.

Response: Not used

5.2.20. FID 9p - Preauth. data

The **Preauth. data** contains transaction authorization data. It is returned in the terminal's response to payment, preauthorization, completion of reclassification, cashback and return (i.e. all financial transactions).

Note: Data lengths of 4-90 bytes are encoded in BASE64, their content is not significant to the cash system, but they serve the terminal to identify the previous transaction.

Example: <FS> 9 <RS> pIQABAAUA4QfOAtwFFAAwMDEwMDUwMDMAAAAQFH57 / LTJxGGvYiBu71q7mujXwijq1wcoAA ==

Request: Optionally on reversal, it allows to reverse the transaction based on their Preauth. data. These transactions can be reversed until the Close Batch has been made.
Mandatory when the pre-authorization is complete.

Response: This field is returned in the refund after the financial transaction.

5.2.21. FID 9S - INVOICE NUMBER 2

INVOICE NUMBER 2, also known as 'variabilní symbol'.

Request: Optional for the financial transaction. Variable length of 1 to 20. Used as a unique stamp of the transaction, generated by a merchant. Will be propagated into the settlement.

Response: Echoed pro request.

Remark: Invoice 2 takes precedence over Invoice 1. If both are included in the request, only Invoice 2 is used and returned back.

5.2.22. FID 9G - Gastro data

This FID is related to payment with food stamps.

Request:
"x, food, soc" where:

The first character "x", says mode control products. 0 denotes full authorization, 1 denotes partial authorization.
The "food", says the price of food (Food) throughout the buying in hundredths of a currency.
The "soc" says food prices, plus cost of social services. This value is a reserve for the implementation of similar cards solutions for social benefits and if it is some of the cash systems still will not use, apply here the same value "food".

Logically Amount> = Soc> = Food. For today's purchases totaling $ 500 CZK at the cash register without the support of social goods, where the food will be worth 300 CZK will therefore Amount = 50000, Soc = 30000 = 30000 Food and auxiliary field will 0,30000,30000

Response:
"n, food, soc" where:

The first number "n" (int 0-99) says Index network cards.The index is = 0 for the bank card . In the case of the food stamp card index will be taken from the prefix table defined by the bank.
The "food" says the amount deducted from a meal voucher cards (or 0 for bank cards)
The "soc" says the amount deducted from the social card (or 0 for bank cards). This feature is defined as RFU so there will always be zero.

Note:
Mode 0 - full authorization
In this mode, the authorization requirement "food stamp card" checks whether the entire purchase amount authorized contains only foods - that Amount = Food and authorization against the headquarters of cards held to Amount. In the case of 'social' card Analogy value Soc.

If the conditions are not fulfilled, or item not listed Food, the transaction will be rejected with an error Prohibited goods before making an authorization to the card center.

Mode 1 - partial authorization
In this mode, the card authorizes only the amount needed for food or social goods. Amount of the entire value of the purchase is not checked and carry the authorization from headquarters reader directly to Food. In the case of 'social' card Analogy value Soc.

If the transaction is confirmed by the Centre in response, the terminal will include the new response code "partially confirmed = 01" instead of "confirmed = 00" and the auxiliary field will contain relevant information, see below. Amount field in the response will contain the actual authorized amount. It may therefore, in case of partial authorization, differ from Amount requirement and therefore can not be used for matching a query-response.

In the case that the cashregister is in the mode GASTRO is required to perform FID 9G even if the amount of food is zero (in the purchase there are no food items).

5.2.23. FID 9g - Progress status

This is a flag of what is being executed (if you want B0 with progress status). In order for the terminal to send progress status in B0, it is necessary to request it. That is, the request in the bit mask sets that the cash register wants extended progress (see below).

Additional information structure:

```
FLAG_SIGN                    0x0001
FLAG_TICKET                  0x0002
FLAG_EX_PROGRESS             0x0200       // if want B0 with progress status
FLAG_TOKEN                   0x0400       // if want TOKEN(family)
FLAG_OFFLINE_DATA            0x0800       // for last offline
FLAG_USER_ID                 0x1000       // need user ID
FLAG_KEEP_ALIVE              0x2000              // supports keep alive
FLAG_SPLIT_SALE              0x4000              // split sale support
FLAG_CONFIRM                 0x8000       // want confirmation
```

Response: This field is returned in a **4H format** (four ascii-hex numbers).

Hi-byte - indicates what action it is.
Lo-byte - the progress in which the action is.

```
#define  IFACE_FLAG_EX_PROGRESS   0x0200  // if want B0 with progress status
Type of operation
         case COMM_AUTH_TXT:      State=0x0100; break;//Authorisation
         case COMM_DCC_TXT:       State=0x0100; break;//DCC
         case COMM_CANCEL_TXT:    State=0x0100; break;//Cancel
         case COMM_HANDSHAKE_TXT: State=0x0200; break;//Handshake
         case COMM_CLOSE_TXT:     State=0x0300; break;//Close batch
         case COMM_INTERSUM_TXT:  State=0x0300; break;//Intersum
         case COMM_PCINIT_TXT:    State=0x0400; break;//Parameters
         case COMM_RIP_TXT:       State=0x0400; break;//Parameters
         case COMM_XCHIP_TXT:     State=0x0400; break;//Parameters
-
State
          case COMM_DONE_TXT:     State|=0x0007;//End of action
          case COMM_INIT_TXT:     State|=0x0000; break;//Initialization
          case MAN_CMD_PHASE_PINNING: State|=0x0001; break;//Entering PIN
          case COMM_DIAL_TXT:     State|=0x0002; break;//Dialing
          case COMM_CONN_TXT:     State|=0x0003; break;//Connection
          case COMM_SEND_TXT:     State|=0x0004; break;//Sending data
          case COMM_REC_TXT:      State|=0x0005; break;//Receiving  response
          case COMM_END_TXT:      State|=0x0006; break;//Communication ended
```

Example of communication:

Packet 'Send': len=47

```
 0x0000: 02 42 31 30 31 20 20 20 20 20 20 20 20 31 39 31 = '.B101        191'
  0x0010: 31 32 38 30 37 34 35 33 39 30 32 30 30 30 30 30 = '1280745390200000'
  0x0020: 39 41 35 41 35 1c 54 30 30 1c 42 32 32 32 03    = '9A5A5.T00.B222. '
```

NetworkSend() -> 47

Packet 'Recv': len=46

```
0x0000: 02 42 30 30 31 54 31 53 54 30 32 34 36 31 39 31 = '.B001T1ST0246191'
  0x0010: 31 32 38 30 38 34 35 35 31 30 32 30 30 30 30 30 = '1280845510200000'
  0x0020: 38 41 35 41 35 1c 39 1d 67 30 31 30 30 03       = '8A5A5.9.g0100.  '
```

Got packet: B0 (len=46)
    Activity/Confirmation
Extended progress: Flags=0x0200
    Progress='0100'
2019-11-28 08:45:52.353
2019-11-28 08:45:52.353
Packet 'Recv': len=46

```
0x0000: 02 42 30 30 31 54 31 53 54 30 32 34 36 31 39 31 = '.B001T1ST0246191'
  0x0010: 31 32 38 30 38 34 35 35 31 30 32 30 30 30 30 30 = '1280845510200000'
  0x0020: 38 41 35 41 35 1c 39 1d 67 30 31 30 33 03       = '8A5A5.9.g0103.  '
```

5.3. Service messages

5.3.1. Get Application Info (80)

Get the application identification (header) and application version (**FID g**). If virtual terminals/merchants are configured, it returns the corresponding values (multiple **FID D**).

Can be used by cashregister to:

- identify the CSOB terminal application
- find out the version of the application

Request:

```
0x0000: 02 42 31 30 31 20 20 20 20 20 20 20 20 31 32 30 = '.B101    120'
0x0010: 32 32 31 31 35 30 31 33 38 30 30 30 30 30 30 30 = '2211501380000000'
0x0020: 34 41 35 41 35 1c 54 38 30 03                  = '4A5A5.T80.      '
```

Response:

```
 0x0000: 02 42 32 30 31 4c 49 4e 55 58 36 36 36 31 32 30 = '.B201LINUX666120'
  0x0010: 33 31 35 30 39 33 33 30 33 30 30 30 30 30 30 34 = '3150933030000004'
  0x0020: 41 41 35 41 35 1c 52 30 30 30 1c 67 56 3a 34 2e = 'AA5A5.R000.gV:4.'
  0x0030: 31 2e 38 1c 44 31 3a 4c 49 4e 55 58 31 31 31 1c = '1.8.D1:LINUX111.'
  0x0040: 44 32 3a 4c 49 4e 55 58 32 32 32 1c 44 33 3a 4c = 'D2:LINUX222.D3:L'
  0x0050: 49 4e 55 58 33 33 33 1c 44 34 3a 4c 49 4e 55 58 = 'INUX333.D4:LINUX'
  0x0060: 34 34 34 1c 44 35 3a 4c 49 4e 55 58 35 35 35 03 = '444.D5:LINUX555.'
```

5.3.2. Get Application Config (84)

Used for mPos solution (terminal connected to Android phone) Allows to get information from the terminal about the list of applications and their possible rearrangement in the menu or about the type/format of the variable symbol.

5.3.3. Passivate (81)

When some transaction is started and the terminal waits for the card insertion, this command can be used to try to passivate the terminal. The response will be **B2** with **R-01** (interrupted), **R-22** (no transaction in progress) or the real (either accepted or denied) result from the real transaction.

5.3.4. Get Last Transaction (82)

This command allows the cash register to retrieve the transaction data from **the last transaction**. The result packet is the same as the result from the real transaction or the error -**22** if the last transaction wasn't successful.

It's the duty of the cashregister to compare the **Amount(s)**, **Authorisation Code**, **Sequence ID**, **Invoice** to its own last transaction and to optionaly reverse this transaction. If the GetLastTransaction command is executed successfully, the transaction ticket can be retrieved by standard means of B3/B4 messages.

If the cash register crashes, in case of re-establishing a TCP connection by the cash register or terminals (depending on the type of client/server connection) after reconnection, it is necessary to determine the last transaction status using the T82 Get Last Transaction command.

Request:

```
0x0000: 02 42 31 30 31 20 20 20 20 20 20 20 20 31 32 30 = '.B101        120'
0x0010: 32 32 31 31 35 35 34 31 35 30 30 30 30 30 30 30 = '2211554150000000'
0x0020: 34 41 35 41 35 1c 54 38 32 03                   = '4A5A5.T82.      '
```

Result: **if last transaction was successfully authorized**

```
0x0000: 02 42 32 30 31 53 31 41 50 4d 4f 4e 30 31 32 30 = '.B201S1APMON0120'
0x0010: 32 32 31 31 36 35 33 32 37 30 30 30 30 30 30 33 = '2211653270000003'
0x0020: 32 41 35 41 35 1c 52 30 30 30 1c 46 53 50 45 43 = '2A5A5.R000.FSPEC'
0x0030: 49 4d 45 4e 1c 69 30 30 31 30 30 33 30 30 31 1c = 'IMEN.i001003001.'
0x0040: 50 34 37 36 31 2a 2a 2a 2a 2a 2a 2a 2a 30 30 32 = 'P4761********002'
0x0050: 38 1c 4a 56 49 53 41 03                         = '8.JVISA.        '
```

Result: **if last transaction wasn't successful**

```
0x0000: 02 42 32 30 31 53 31 41 50 4d 4f 4e 30 31 32 30 = '.B201S1APMON0120'
0x0010: 32 32 31 31 37 30 38 30 37 30 30 30 30 30 30 30 = '2211708070000000'
0x0020: 35 41 35 41 35 1c 52 2d 32 32 03                = '5A5A5.R-22.     '
```

5.3.5. Get Last Batch (83)

This command allows the cash register to retrieve the value of the last close batch. In case of the multi-merchant-terminal, the **FID D** can be used to retrieve the virtual-specific close batch. The close-batch ticket is not 'printer'.

5.3.6. Get Last Batch (83)

This command is used to retrieve the terminal's last Close Batch data. In the case of setting the terminal mode multi-merchant-terminal (terminal is shared by multiple merchants) you can use the data field **FID D** gain relative to transactions closing date, one of the virtual merchants. The Close Batch ticket is not printed when using this command on the terminal.

5.3.7. Close Totals (60)

This command is invoked to execute a terminal Close Batch. The terminal exchanges its totals with the authorisation server and returns the result to the cash register. At the same time, an off-line transaction is sent to the authorisation server. And after executing this command, both terminal and authorisation server totals are reset.

Request:

```
  0x0000: 02 42 31 30 31 20 20 20 20 20 20 20 20 31 37 30 = '.B101        170'
  0x0010: 35 32 39 31 31 33 35 30 36 30 30 30 30 30 30 30 = '5291135060000000'
  0x0020: 34 41 35 41 35 1c 54 36 30 03                   = '4A5A5.T60.    '
```

Progress message:

```
  0x0000: 02 42 30 30 31 54 4a 48 42 30 30 30 33 31 37 30 = '.B001TJHB0003170'
  0x0010: 35 32 39 31 32 33 33 33 31 30 30 30 30 30 30 30 = '5291233310000000'
  0x0020: 30 41 35 41 35 03                               = '0A5A5.         '
```

Response:

```
  0x0000: 02 42 32 30 31 54 4a 48 42 30 30 30 33 31 37 30 = '.B201TJHB0003170'
```

```
0x0010: 35 32 39 31 32 33 33 33 31 30 30 30 32 30 30 33 = '5291233310002003'
0x0020: 44 41 35 41 35 1c 52 30 30 30 1c 54 36 30 1c 6c = 'DA5A5.R000.T60.l'
0x0030: 30 30 31 30 34 37 30 30 30 30 2b 30 30 30 30 30 = '0010470000+00000'
0x0040: 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 = '0000000000000000'
0x0050: 2b 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 = '+000000000000000'
0x0060: 30 30 03                                        = '00.             '
```

5.3.8. Subtotals (65)

This command is invoked to execute the subtotal of the terminal. The terminal exchanges its totals with the authorization server transactions and returns the result to the cashier. Unlike the Close Batch command, the value of the totals is not reset after this command.

Request:

```
0x0000: 02 42 31 30 31 20 20 20 20 20 20 20 20 31 37 30 = '.B101        170'
0x0010: 35 32 39 31 32 30 34 31 36 30 30 30 30 30 30 30 = '5291204160000000'
0x0020: 34 41 35 41 35 1c 54 36 35 03                   = '4A5A5.T65.      '
```

Progress message:

```
 0x0000: 02 42 30 30 31 54 4a 48 42 30 30 30 33 31 37 30 = '.B001TJHB0003170'
 0x0010: 35 32 39 31 33 30 32 34 32 30 30 30 30 30 30 30 = '5291302420000000'
 0x0020: 30 41 35 41 35 03                               = '0A5A5.          '
```

Response:

```
 0x0000: 02 42 32 30 31 54 4a 48 42 30 30 30 33 31 37 30 = '.B201TJHB0003170'
 0x0010: 35 32 39 31 33 30 32 34 32 30 30 30 32 30 30 33 = '5291302420002003'
 0x0020: 44 41 35 41 35 1c 52 30 30 30 1c 54 36 35 1c 6c = 'DA5A5.R000.T65.l'
 0x0030: 30 30 31 30 34 37 30 30 30 30 2b 30 30 30 30 30 = '0010470000+00000'
 0x0040: 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 = '0000000000000000'
 0x0050: 2b 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 = '+000000000000000'
 0x0060: 30 30 03                                        = '00.             '
```

5.3.9. TMS Call (90)

This command is used to call the terminal connection to POSMANAGMENT and the terminal, respectively. MBCA will download current parameters.

Request:

```
 0x0000: 02 42 31 30 31 20 20 20 20 20 20 20 20 31 37 30 = '.B101        170'
 0x0010: 35 32 39 31 31 31 32 34 39 30 30 30 30 30 30 30 = '5291112490000000'
 0x0020: 34 41 35 41 35 1c 54 39 30 03                   = '4A5A5.T90.      '
```

Progress message:

```
 0x0000: 02 42 30 30 31 54 4a 48 42 30 30 30 33 31 37 30 = '.B001TJHB0003170'
 0x0010: 35 32 39 31 32 31 31 31 30 30 30 30 30 30 30 30 = '5291211100000000'
 0x0020: 30 41 35 41 35 03                               = '0A5A5.          '
```

Response: (Connection error)

```
 0x0000: 02 42 32 30 31 54 4a 48 42 30 30 30 33 31 37 30 = '.B201TJHB0003170'
 0x0010: 35 32 39 31 32 31 31 32 32 30 30 30 30 30 30 30 = '5291211220000000'
 0x0020: 39 41 35 41 35 1c 52 2d 30 36 1c 54 39 30 03 = '9A5A5.R-06.T90. '
```

Response: (Connection OK)

```
 0x0000: 02 42 32 30 31 54 4a 48 42 30 30 30 33 31 37 30 = '.B201TJHB0003170'
 0x0010: 35 32 39 31 32 31 31 32 32 30 30 30 30 30 30 30 = '5291211220000000'
 0x0020: 39 41 35 41 35 1c 52 30 30 30 1c 54 39 30 03 = '9A5A5.R000.T90. '
```

5.3.10. Handshake (95)

This command is used to invoke the authentication server connection test. This command verifies the correct setting of the authorisation server address and the terminal communication key.

Request:

```
  0x0000: 02 42 31 30 31 20 20 20 20 20 20 20 20 31 37 30 = '.B101        170'
  0x0010: 35 32 39 31 30 33 34 34 38 30 30 30 30 30 30 30 = '5291034480000000'
  0x0020: 34 41 35 41 35 1c 54 39 35 03                   = '4A5A5.T95.      '
```

Response:

```
  0x0000: 02 42 32 30 31 54 4a 48 42 30 30 30 33 31 37 30 = '.B201TJHB0003170'
  0x0010: 35 32 39 31 31 33 33 31 35 30 30 30 30 30 30 31 = '5291133150000001'
  0x0020: 33 41 35 41 35 1c 52 30 30 30 1c 67 41 50 50 52 = '3A5A5.R000.gAPPR'
  0x0030: 4f 56 45 44 1c 54 39 35 03                      = 'OVED.T95.       '
```

5.4. FIDs in financial transactions

All commands must contain the FID T - transaction type

5.4.1. Normal purchase (00)

This command / transaction is used to make a payment.

Mandatory FIDs for B1: T, B, E1), D2), 9P.
Optional FIDs for B1: S, 9S.

FIDs in response B2: T, P, J3), F, R, g, 9p.

Packet 'Send': len=61

```
0x0000: 02 42 31 30 31 20 20 20 20 20 20 20 20 32 30 30 = '.B101        200'
0x0010: 37 32 37 30 39 35 33 35 35 30 30 30 30 30 30 31 = '7270953550000001'
0x0020: 37 41 35 41 35 1c 54 30 30 1c 42 35 35 30 30 30 = '7A5A5.T00.B55000'
0x0030: 30 30 1c 39 1d 50 31 1c 45 33 34 38 03          = '00.9.P1.E348.   '
```

Packet 'Recv': len=209

```
0x0000: 02 42 32 30 31 48 55 4b 48 30 30 30 31 32 30 30 = '.B201HUKH0001200'
0x0010: 37 32 37 31 31 35 34 31 34 30 30 30 30 30 30 41 = '727115414000000A'
0x0020: 42 41 35 41 35 1c 52 30 31 30 1c 67 70 72 6f 76 = 'BA5A5.R010.gprov'
0x0030: 65 64 65 6e 6f 1c 54 30 30 1c 42 35 35 35 30 39 = 'edeno.T00.B55509'
0x0040: 33 1c 50 32 32 32 32 34 2a 2a 2a 2a 2a 2a 2a 30 = '3.P22224*******0'
0x0050: 31 38 32 1c 4a 4d 41 53 54 45 52 43 41 52 44 1c = '182.JMASTERCARD.'
0x0060: 46 30 30 35 36 33 37 20 20 1c 69 30 30 31 30 31 = 'F005637  .i00101'
0x0070: 37 30 31 38 1c 39 1d 70 41 41 41 42 41 42 45 41 = '7018.9.pAAABABEA'
0x0080: 35 41 66 58 41 6f 49 45 41 77 41 77 4d 44 45 77 = '5AfXAoIEAwAwMDEw'
0x0090: 4d 54 63 77 4d 54 67 41 41 41 41 51 46 50 66 2f = 'MTcwMTgAAAAQFPf/'
0x00a0: 33 54 59 47 42 43 65 57 69 6d 6a 68 36 45 71 71 = '3TYGBCeWimjh6Eqq'
0x00b0: 53 2b 67 48 77 43 6a 71 56 58 67 49 41 41 41 41 = 'S+gHwCjqVXgIAAAA'
0x00c0: 41 41 41 41 41 41 41 41 30 41 6d 59 41 41 3d 3d = 'AAAAAAAA0AmYAA=='
0x00d0: 03                                              = '.               '
```

5.4.2. Preauthorization (01)

Pre-authorization is the same as normal payment.

Mandatory FIDs for requirement B1: T, B, E1), D2), 9P.
Optional FIDs for requirement B1: S, 9S.

FIDs in response B2: T, P, J3), F, R, g, 9p.

Example of communication:

Packet 'Send': len=69

```
0x0000: 02 42 31 30 31 20 20 20 20 20 20 20 20 32 30 30 = '.B101        200'
0x0010: 34 31 35 31 30 33 33 35 33 30 30 30 30 30 30 31 = '4151033530000001'
0x0020: 46 41 35 41 35 1c 54 30 31 1c 42 35 35 35 35 1c = 'FA5A5.T01.B5555.'
0x0030: 39 1d 53 4e 41 5a 44 41 52 31 32 33 34 35 36 1c = '9.SNAZDAR123456.'
0x0040: 45 32 30 33 03                                  = 'E203.           '
```

Packet 'Recv': len=223

```
  0x0000: 02 42 32 30 31 54 31 53 54 30 32 34 36 32 30 30 = '.B201T1ST0246200'
  0x0010: 34 31 35 31 32 33 34 30 33 30 30 30 32 30 30 42 = '415123403000200B'
  0x0020: 39 41 35 41 35 1c 52 30 30 30 1c 67 70 72 6f 76 = '9A5A5.R000.gprov'
  0x0030: 65 64 65 6e 6f 1c 54 30 31 1c 42 35 35 35 35 1c = 'edeno.T01.B5555.'
  0x0040: 39 1d 53 4e 41 5a 44 41 52 31 32 33 34 35 36 1c = '9.SNAZDAR123456.'
  0x0050: 50 35 34 30 30 39 2a 2a 2a 2a 2a 2a 2a 36 39 36 = 'P54009*******696'
  0x0060: 33 1c 4a 4d 41 53 54 45 52 43 41 52 44 1c 46 53 = '3.JMASTERCARD.FS'
  0x0070: 50 43 4d 4e 30 39 34 1c 69 30 30 31 30 32 34 30 = 'PCMN094.i0010240'
  0x0080: 30 38 1c 39 1d 70 49 51 41 42 41 42 67 41 35 41 = '08.9.pIQABABgA5A'
  0x0090: 65 66 41 64 49 45 41 77 41 77 4d 44 45 77 4d 6a = 'efAdIEAwAwMDEwMj'
  0x00a0: 51 77 4d 44 67 41 41 41 41 51 46 4c 33 6f 33 4c = 'QwMDgAAAAQFL3o3L'
  0x00b0: 37 4d 2f 55 56 6f 6b 69 31 64 37 77 71 30 78 75 = '7M/UVoki1d7wq0xu'
  0x00c0: 71 2f 77 53 6a 71 73 78 55 41 41 41 41 41 41 41 = 'q/wSjqsxUAAAAAAA'
  0x00d0: 41 41 41 41 41 41 67 41 57 41 41 41 3d 3d 03    = 'AAAAAAgAWAAA==. '
```

5.4.3. Completion of preauthorization (02)

When the pre-authorization is completed, it is necessary to send an additional "pre-authorization data" to the terminal Auth code, SEQ ID ..

When pre-authorization is completed, preauth data (FID 9p) must be sent to the terminal.

Operation T02 requires the use of a card, or manual entry directly into the terminal, provided that the bank allows manual entry. It is possible to initiate from the cash register only under the conditions of reading the card (CL or chip) or manually entering the card number into the terminal. **It is not possible to perform from the cashier without this previous activity and it is not possible to complete it without the presence of the card.** The PAN card is on the receipt from the pre-authorization.

Mandatory FIDs for requirement B1: T, B, E$^{1)}$, D$^{2)}$, 9p, i.
Optional FIDs for requirement B1: S, 9S, 9P.

FIDs in response B2: T, P, J$^{3)}$, F, R, g, 9p.

Example of communication:

1) **Termination using the card:**

Packet 'Send': len=198

```
  0x0000: 02 42 31 30 31 20 20 20 20 20 20 20 20 32 30 30 = '.B101        200'
  0x0010: 34 31 35 31 30 33 39 32 34 30 30 30 30 30 30 41 = '415103924000000A'
  0x0020: 30 41 35 41 35 1c 54 30 32 1c 42 34 34 34 34 1c = '0A5A5.T02.B4444.'
  0x0030: 39 1d 53 4e 41 5a 44 41 52 31 32 33 34 35 36 1d = '9.SNAZDAR123456.'
  0x0040: 70 49 51 41 42 41 42 67 41 35 41 65 66 41 64 49 = 'pIQABABgA5AefAdI'
  0x0050: 45 41 77 41 77 4d 44 45 77 4d 6a 51 77 4d 44 67 = 'EAwAwMDEwMjQwMDg'
  0x0060: 41 41 41 41 51 46 4c 33 6f 33 4c 37 4d 2f 55 56 = 'AAAAQFL3o3L7M/UV'
  0x0070: 6f 6b 69 31 64 37 77 71 30 78 75 71 2f 77 53 6a = 'oki1d7wq0xuq/wSj'
  0x0080: 71 73 78 55 41 41 41 41 41 41 41 41 41 41 41 41 = 'qsxUAAAAAAAAAAAA'
  0x0090: 41 67 41 57 41 41 41 3d 3d 1c 69 30 30 31 30 32 = 'AgAWAAA==.i00102'
  0x00a0: 34 30 30 38 1c 46 53 50 43 4d 4e 30 39 34 1c 45 = '4008.FSPCMN094.E'
  0x00b0: 32 30 33 1c 50 35 34 30 30 39 2a 2a 2a 2a 2a 2a = '203.P54009******'
  0x00c0: 2a 36 39 36 33 03                               = '*6963.          '
```

Packet 'Recv': len=223

```
  0x0000: 02 42 32 30 31 54 31 53 54 30 32 34 36 32 30 30 = '.B201T1ST0246200'
  0x0010: 34 31 35 31 32 33 39 34 35 30 30 30 32 30 30 42 = '415123945000200B'
  0x0020: 39 41 35 41 35 1c 52 30 30 30 1c 67 70 72 6f 76 = '9A5A5.R000.gprov'
  0x0030: 65 64 65 6e 6f 1c 54 30 32 1c 42 34 34 34 34 1c = 'edeno.T02.B4444.'
  0x0040: 39 1d 53 4e 41 5a 44 41 52 31 32 33 34 35 36 1c = '9.SNAZDAR123456.'
  0x0050: 50 35 34 30 30 39 2a 2a 2a 2a 2a 2a 2a 36 39 36 = 'P54009*******696'
  0x0060: 33 1c 4a 4d 41 53 54 45 52 43 41 52 44 1c 46 53 = '3.JMASTERCARD.FS'
  0x0070: 50 43 4d 4e 30 39 34 1c 69 30 30 31 30 32 34 30 = 'PCMN094.i0010240'
  0x0080: 30 38 1c 39 1d 70 49 67 41 42 41 42 67 41 35 41 = '08.9.pIgABABgA5A'
  0x0090: 65 66 41 64 63 45 4c 51 41 77 4d 44 45 77 4d 6a = 'efAdcELQAwMDEwMj'
  0x00a0: 51 77 4d 44 67 41 41 41 41 51 46 4c 33 6f 33 4c = 'QwMDgAAAAQFL3o3L'
  0x00b0: 35 45 6b 33 78 67 6b 69 31 64 37 77 71 30 78 75 = '5Ek3xgki1d7wq0xu'
  0x00c0: 71 2f 77 53 6a 71 58 42 45 41 41 41 41 41 41 41 = 'q/wSjqXBEAAAAAAA'
  0x00d0: 41 41 41 41 41 41 67 41 57 55 41 41 3d 3d 03    = 'AAAAAAgAWUAA==. '
```

2) **Termination without the presence of a card, PAN is inserted directly into the terminal:**

Packet 'Send': len=183

```
  0x0000: 02 42 31 30 31 20 20 20 20 20 20 20 20 32 30 30 = '.B101        200'
  0x0010: 34 31 35 31 30 34 32 33 39 30 30 30 30 30 30 39 = '4151042390000009'
  0x0020: 31 41 35 41 35 1c 54 30 32 1c 42 34 34 34 34 1c = '1A5A5.T02.B4444.'
  0x0030: 39 1d 53 4e 41 5a 44 41 52 31 32 33 34 35 36 1d = '9.SNAZDAR123456.'
  0x0040: 70 49 51 41 42 41 42 67 41 35 41 65 66 41 64 49 = 'pIQABABgA5AefAdI'
  0x0050: 45 41 77 41 77 4d 44 45 77 4d 6a 51 77 4d 44 67 = 'EAwAwMDEwMjQwMDg'
  0x0060: 41 41 41 41 51 46 4c 33 6f 33 4c 37 4d 2f 55 56 = 'AAAAQFL3o3L7M/UV'
  0x0070: 6f 6b 69 31 64 37 77 71 30 78 75 71 2f 77 53 6a = 'oki1d7wq0xuq/wSj'
  0x0080: 71 73 78 55 41 41 41 41 41 41 41 41 41 41 41 41 = 'qsxUAAAAAAAAAAAA'
  0x0090: 41 67 41 57 41 41 41 3d 3d 1c 69 30 30 31 30 32 = 'AgAWAAA==.i00102'
  0x00a0: 34 30 30 38 1c 46 53 50 43 4d 4e 30 39 34 1c 45 = '4008.FSPCMN094.E'
  0x00b0: 32 30 33 1c 50 2d 03                            = '203.P-.         '
```

Packet 'Recv': len=223

```
  0x0000: 02 42 32 30 31 54 31 53 54 30 32 34 36 32 30 30 = '.B201T1ST0246200'
  0x0010: 34 31 35 31 32 34 32 35 39 30 30 30 32 30 30 42 = '415124259000200B'
  0x0020: 39 41 35 41 35 1c 52 30 30 30 1c 67 70 72 6f 76 = '9A5A5.R000.gprov'
  0x0030: 65 64 65 6e 6f 1c 54 30 32 1c 42 34 34 34 34 1c = 'edeno.T02.B4444.'
  0x0040: 39 1d 53 4e 41 5a 44 41 52 31 32 33 34 35 36 1c = '9.SNAZDAR123456.'
  0x0050: 50 35 34 30 30 39 2a 2a 2a 2a 2a 2a 2a 36 39 36 = 'P54009*******696'
  0x0060: 33 1c 4a 4d 41 53 54 45 52 43 41 52 44 1c 46 53 = '3.JMASTERCARD.FS'
  0x0070: 50 43 4d 4e 30 39 34 1c 69 30 30 31 30 32 34 30 = 'PCMN094.i0010240'
  0x0080: 30 38 1c 39 1d 70 41 67 41 42 41 42 67 41 35 41 = '08.9.pAgABABgA5A'
  0x0090: 65 66 41 64 6f 45 4f 77 41 77 4d 44 45 77 4d 6a = 'efAdoEOwAwMDEwMj'
  0x00a0: 51 77 4d 44 67 41 41 41 41 51 41 4c 33 6f 33 4c = 'QwMDgAAAAQAL3o3L'
  0x00b0: 34 41 41 41 41 41 6b 69 31 64 37 77 71 30 78 75 = '4AAAAAki1d7wq0xu'
  0x00c0: 71 2f 77 53 6a 71 58 42 45 41 41 41 41 41 41 41 = 'q/wSjqXBEAAAAAAA'
  0x00d0: 41 41 41 41 41 41 2f 2f 2f 31 41 41 3d 3d 03    = 'AAAAAA///1AA==. '
```

5.4.4. Merchandise return (04)

This command / transaction is used to return funds to the customer's account.

Mandatory FIDs for B1: T, B, E1), D2)
Optional FIDs for B1: S, 9S, P, 9P.

FIDs in response B2: T, P, J3), F, R, g, 9p.

5.4.5. Cash advance (05)

This command / transaction is used to select cash. Typical uses are hotel reception or currency exchange.

Mandatory FIDs for B1: T, B, E1), D2).
Optional FIDs for B1: S, 9S, 9P.

FIDs in response B2: T, P, J3), F, R, g, 9p.

5.4.6. Card verify (06)

This operation is used to verify the card (or cardholder).

Mandatory FIDs for B1: T, E (*for multicurrency terminal), D (*if multiple merchant support is enabled)
Optional FIDs for B1: 9

FIDs in response B2: T, R, g, P, F, J, 9p, 1

Example of communication:

```
Packet 'Send': len=42
  0x0000: 02 42 31 30 31 20 20 20 20 20 20 20 20 32 33 31 = '.B101        231'
  0x0010: 32 31 38 31 31 30 31 32 34 30 30 30 30 30 30 30 = '2181101240000000'
  0x0020: 34 41 35 41 35 1c 54 30 36 03                   = '4A5A5.T06.      '

Packet 'Recv': len=285
  0x0000: 02 42 32 30 31 54 54 4f 4f 50 39 39 36 32 33 31 = '.B201TTOOP996231'
  0x0010: 32 31 38 31 32 30 31 32 39 30 30 30 32 30 30 46 = '218120129000200F'
  0x0020: 37 41 35 41 35 1c 52 30 30 30 1c 67 70 72 6f 76 = '7A5A5.R000.gprov'
  0x0030: 65 64 65 6e 6f 1c 54 30 36 1c 50 35 33 35 35 36 = 'edeno.T06.P53556'
  0x0040: 31 2a 2a 2a 2a 2a 2a 36 39 34 39 1c 4a 4d 41 53 = '1******6949.JMAS'
  0x0050: 54 45 52 43 41 52 44 1c 46 36 38 32 39 33 32 20 = 'TERCARD.F682932 '
  0x0060: 20 1c 69 30 30 31 30 32 36 30 30 36 1c 31 31 30 = ' .i001026006.110'
  0x0070: 32 30 34 33 34 43 32 46 44 42 33 44 32 36 37 36 = '20434C2FDB3D2676'
  0x0080: 37 34 44 36 36 41 34 31 38 35 44 34 42 36 42 30 = '74D66A4185D4B6B0'
  0x0090: 36 44 42 31 41 36 36 36 36 43 37 44 46 44 32 36 = '6DB1A6666C7DFD26'
  0x00a0: 32 33 42 34 45 45 32 31 45 46 33 33 37 43 44 30 = '23B4EE21EF337CD0'
  0x00b0: 1c 61 41 30 30 30 30 30 30 30 30 34 31 30 31 30 = '.aA0000000041010'
  0x00c0: 1c 39 1d 70 4b 41 41 42 41 42 6f 41 35 77 66 43 = '.9.pKAABABoA5wfC'
  0x00d0: 42 4c 45 45 48 51 41 77 4d 44 45 77 4d 6a 59 77 = 'BLEEHQAwMDEwMjYw'
```

```
0x00e0: 4d 44 59 41 53 41 41 51 46 47 68 57 48 64 39 64 = 'MDYASAAQFGhWHd9d'
0x00f0: 39 75 58 65 47 6a 70 44 37 37 4c 38 46 2b 67 2f = '9uXeGjpD77L8F+g/'
0x0100: 77 53 6a 71 2f 2f 2f 2f 2f 77 41 41 41 41 41 41 = 'wSjq/////wAAAAAA'
0x0110: 41 41 41 41 4d 67 71 58 41 41 3d 3d 03          = 'AAAAMgqXAA==.   '
```

5.4.7. Reversal (10)

This command / transaction is used to cancel the last signing transaction made at the payment terminal, in case of disagreement. It can only be executed immediately after a payment transaction and no closing date can be made between the payment transaction and its reverse. Optionaly, FID 9p can be used, which allows transaction reversal based on its pre-auth data. These transactions can be reversed up to the time the closing was made.

Mandatory FIDs for B1: T, F. (Optional 9p)

Mandatory FIDs in response B2: T, R, g.

5.4.8. Balance Inquiry (07)

This command / transaction is used to get the balance on your card account. If the cashier sends FID P, a query is made according to the inserted PAN, otherwise the customer terminal prompts the cardholder to insert the card and loads the card number itself.

Mandatory FIDs for B1: T (Optional P)

FIDs in response B2: T, L, R, g.

5.4.9. Purchase with cashback (08)

This command / transaction is used to make a cash withdrawal payment.

Mandatory FIDs for B1: T, B, C, E[1)], D[2)].
Optional FIDs for B1: S, 9S, 9P

FIDs in response B2: T, P, J[3)], F, R, g, 9p.
Note:
[1)] Mandatory for the multicurrency terminal.
[2)] Required for financial transactions on terminals with multiple merchant support turned on.
[3)] This field contains the balance on the card, it returns if this functionality is supported by the appropriate authorization server.

6. TRANSACTION FLOW

6.1. Financial transaction

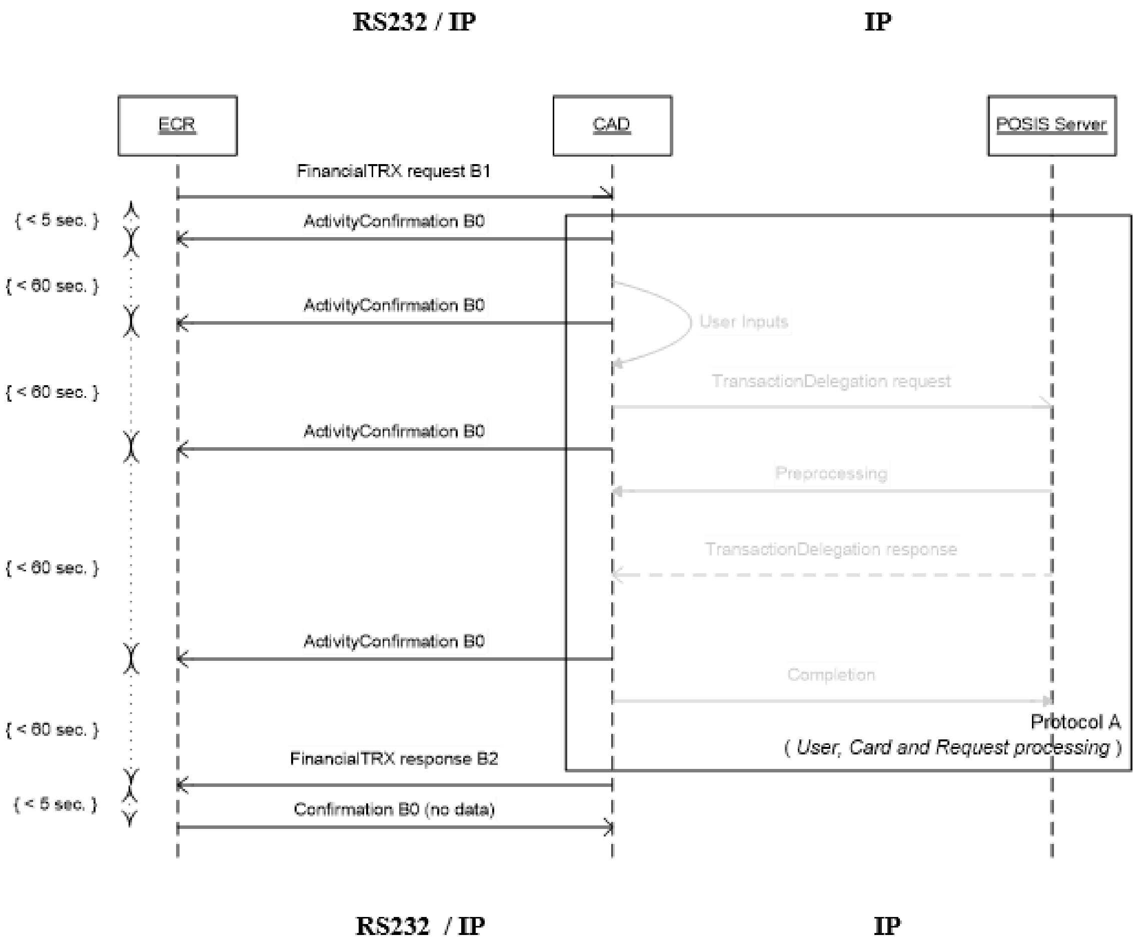


Diagram 3 *Financial transaction flow*

Description

Steps	Processing
1	The ECR sends request message B1 for a financial transaction to CAD and sets timeout 5s for ActivityConfirmation message B0 receiving
2	CAD sends ActivityConfirmation message to ECR immediately after B1 came
3	ECR gets B0 and waits B0 or B2 message CAD reads card data, takes amount from B1 and start communication with POSIS (over IP) (authorization protocol A) and during this time period repeatedly sends B0 (ECR B0 timer = 60s)
4	Authorisation is finished, CAD sends B2 end message to ECR
5	ECR sends confirmation message B0 into the CAD (timeout upto 5sec).

Table 7 *Description of financial transaction flow*

If the cashier reverses the last transactions, the FID field F (authorization code) should be set according to the result of the last confirmed transaction that should be reversed.

6.2. EXPLICIT CONFIRMATION

In a case where explicit confirmation from the cash register is needed and the terminal should make a reversal, if this confirmation is not received, the cash register can use the FLAGS bit 15.

When this flag is set on the B1 request (payment only), it's echoed on the B2 response. If cash register receives FLAGS bit 15 set in B2 financial response, the terminal waits at most 5 seconds for the confirmation (B0 message). If no such confirmation is received, the terminal makes the reversal of the current transaction. Example of communication:

Cash request:

```
0x0000: 02 42 31 30 31 20 20 20 20 20 20 20 20 31 37 31 = '.B101        171'
0x0010: 30 32 34 31 33 35 36 32 38 38 30 30 30 30 30 31 = '0241356288000001'
0x0020: 39 41 35 41 35 1c 54 30 30 1c 42 31 30 30 1c 39 = '9A5A5.T00.B100.9'
0x0030: 1d 53 41 42 43 44 31 32 33 34 45 46 47 48 03 = '.SABCD1234EFGH. '
```

8000 flag on header = bit 15 is set – requested explicit confirmation

Confirmation of cash request

```
0x0000: 02 42 30 30 31 54 31 53 54 30 32 33 30 31 37 31 = '.B001T1ST0230171'
0x0010: 30 32 34 31 35 35 36 32 37 30 30 30 30 30 30 30 = '0241556270000000'
0x0020: 30 41 35 41 35 03                               = '0A5A5.         '
```

Progress messages from terminal (0..n)

```
0x0000: 02 42 30 30 31 54 31 53 54 30 32 33 30 31 37 31 = '.B001T1ST0230171'
0x0010: 30 32 34 31 35 35 36 32 37 30 30 30 30 30 30 30 = '0241556270000000'
0x0020: 30 41 35 41 35 03                               = '0A5A5.         '
```

Result of payment transaction by terminal

```
0x0000: 02 42 32 30 31 54 31 53 54 30 32 33 30 31 37 31 = '.B201T1ST0230171'
0x0010: 30 32 34 31 35 35 36 34 32 38 30 30 32 30 30 46 = '024155642800200F'
0x0020: 43 41 35 41 35 1c 52 30 30 35 1c 67 70 72 6f 76 = 'CA5A5.R005.gprov'
0x0030: 65 64 65 6e 6f 1c 54 30 30 1c 42 31 30 30 1c 39 = 'edeno.T00.B100.9'
0x0040: 1d 53 41 42 43 44 31 32 33 34 45 46 47 48 1c 50 = '.SABCD1234EFGH.P'
0x0050: 35 31 36 38 33 2a 2a 2a 2a 2a 2a 2a 35 30 34 34 = '51683*******5044'
0x0060: 1c 4a 4d 41 53 54 45 52 43 41 52 44 1c 46 36 31 = '.JMASTERCARD.F61'
0x0070: 31 30 34 38 20 20 1c 69 30 30 31 30 36 31 30 30 = '1048  .i00106100'
0x0080: 34 1c 31 31 31 34 43 37 46 38 45 39 35 35 30 34 = '4.1114C7F8E95504'
0x0090: 30 33 37 42 44 38 34 41 44 42 43 30 37 39 31 33 = '037BD84ADBC07913'
0x00a0: 37 46 31 31 44 36 42 36 42 46 33 45 33 31 43 43 = '7F11D6B6BF3E31CC'
0x00b0: 35 46 36 36 36 36 35 43 37 41 39 34 41 35 39 46 = '5F66665C7A94A59F'
0x00c0: 34 30 39 36 33 1c 39 1d 70 49 41 41 42 41 44 30 = '40963.9.pIAABAD0'
0x00d0: 41 34 51 63 41 42 42 51 47 4b 67 41 77 4d 44 45 = 'A4QcABBQGKgAwMDE'
0x00e0: 77 4e 6a 45 77 4d 44 51 41 41 41 41 51 46 48 35 = 'wNjEwMDQAAAAQFH5'
0x00f0: 37 2f 4c 54 4a 78 47 47 76 59 69 42 75 37 31 71 = '7/LTJxGGvYiBu71q'
```

```
  0x0100: 37 6d 75 6a 58 77 69 6a 71 5a 41 41 41 41 41 41 = '7mujXwijqZAAAAAA'
  0x0110: 41 41 41 41 41 41 41 41 41 31 77 64 4e 41 41 3d = 'AAAAAAAAA1wdNAA='
  0x0120: 3d 03                                            = '=.              '
```

8002 flag on header of answer:
  bit 15 is set – requests explicit confirmation
  bit 1 is set – chash print tickets

Confirmation message to result message from terminal

```
 0x0000: 02 42 30 30 31 54 31 53 54 30 32 33 30 31 37 31 = '.B001T1ST0230171'
 0x0010: 30 32 34 31 35 35 36 32 37 30 30 30 30 30 30 30 = '0241556270000000'
 0x0020: 30 41 35 41 35 03                                = '0A5A5.          '
```

This B0 packet must be sent to the terminal us explicit confirmation of the terminal payment result.

6.3. Terminal in Busy state (RC -30)

The Busy RC -30 response occurs when the terminal is not ready to perform the requested operation. For example: the terminal is currently making a payment, batch close, TMS call, etc., or the terminal is not in the basic state, eg if the user moves in the menu.

Example:

Payment request:

```
Packet 'Send': len=55
  0x0000: 02 42 31 30 31 20 20 20 20 20 20 20 20 32 32 30 = '.B101        220'
  0x0010: 33 30 33 31 34 30 33 31 33 30 30 30 30 30 30 31 = '3031403130000001'
  0x0020: 31 41 35 41 35 1c 54 30 30 1c 42 31 30 30 1c 45 = '1A5A5.T00.B100.E'
  0x0030: 32 30 33 1c 44 31 03                             = '203.D1.
```

Response (terminal in busy state):

```
Packet 'Recv': len=61
  0x0000: 02 42 32 30 31 54 45 54 52 41 4d 30 31 32 32 30 = '.B201TETRAM01220'
  0x0010: 33 30 33 31 35 30 33 31 32 30 30 30 30 30 30 31 = '3031503120000001'
  0x0020: 37 41 35 41 35 1c 54 30 30 1c 52 2d 33 30 1c 67 = '7A5A5.T00.R-30.g'
  0x0030: 42 75 73 79 1c 42 31 30 30 1c 44 31 03          = 'Busy.B100.D1.   '
```

7. COMMUNICATION LINK

7.1. UDP / IP / Ethernet

The terminal must know the IP address of the cash register (configurable via Manager Menu). The cash register must know the IP address of the terminal.
Communication is carried over IP on the UDP port 33333.
Each complete B protocol message (including the <STX> and <ETX>) is encapsulated into one UDP datagram.

7.2. TCP / IP / Ethernet - terminal as a client

The terminal must know the IP address and the port of the cash register (configurable via Manager Menu). The cash register is a server, which listens on a configured port.

Terminal is a slave, which periodically (~15 secs) tries to connect to the cash register. When connected, it just waits for B-Protocol commands. **Because the terminal is a slave, he can't initiate any sort of keepalive activity!!** So it's the duty of either

a) cash register server to use TCP keep-alive packets, or
b) the configuration of all network elements between the cash register and terminal

to allow the terminal-cashregister connection to be persistent.

If the cash register crashes, in case of re-establishing a TCP connection by the cash register or terminals (depending on the type of client/server connection) after reconnection, it is necessary to determine the last transaction status using the T82 Get Last Transaction command.

The B protocol message over TCP is the same as on the serial port - it's just a stream of data.

7.2.1. Multiple ports cashregister server

Cash register listens on a bunch of specific TCP ports, one per terminal. Each terminal is configured to connect to its own port. It's the duty of the server to keep the translation table PORT <--> TERMINAL.

7.2.2. Single port cashregister server

Cash register listens on only one TCP port. Each terminal is configured to connect to this one port. After TCP connect, the cash register can issue either:

- the new Get App Info (80) request
- special 'malformed' B-protocol request

to retrieve the terminal identification from the B-protocol response header.

One example of such 'malformed' request, response pair could be:

Response:

```
  0x0000: 02 42 30 30 31 20 20 20 20 20 20 20 20 31 30 30 = '.B101        100'
  0x0010: 35 31 33 31 33 32 36 31 31 30 30 30 30 30 30 30 = '5131326110000000'
  0x0020: 32 41 35 41 35 1c 54 03                        = '2A5A5.T.        '
```

Response:

```
  0x0000: 02 42 32 30 31 50 56 54 4d 4f 4e 32 30 31 30 30 = '.B201PVTMON20100'
  0x0010: 35 31 33 31 35 32 36 33 30 30 30 30 30 30 30 30 = '5131526300000000'
  0x0020: 35 41 35 41 35 1c 52 2D 32 32 03              = '5A5A5.R-22.     '
```

7.3. TCP / IP / Ethernet - terminal as server

In the case of communication of the cash register via TCP in the TCP server terminal mode, the IP address of the cash register should be set in the terminal (settings in the Manager application menu). Alternatively in the MCASH application. With this method of communication, the terminal is a server that listens on the set port.

If desired, just fill in the port, without specifying the IP (no IP address of the cash register needs to be filled in).

Cash register then acts as a client that tries periodically (recommended ~ 15 secs) to connect to the terminal. After establishing the connection terminal waits to receive the B-protocol commands.

The terminal allows to configure multiple (max 4) cash registers (IP addresses) that can connect. With an active connection from the cash register under IP1, which has established a TCP connection, the terminal will communicate on this IP until another cash register under IP2 connects to the terminal and establishes a new TCP connection.
If this happens during authorization, it may happen that a request sent from IP1 + the state when IP2 connects, the response will be sent to the cash register under IP2.

An integrator provides the synchronization of cash registers, one of the cash registers can't throw down another cash register. The terminal communicates only to the cash register that is currently connected, even within one transaction.

To cover the loss of communication or restart the terminal, eg. Due to a power failure or a problem with the cables, it is necessary to support the cash register applications keep-alive mechanisms.

Because TCP implementation in OS terminal does not know the line-keep-alive, is supported by keep-alive applications as well as in client mode. If eg. power failure terminal, checkout this mechanism detects a problem, it establishes a new connection and is able to correctly complete the transaction. A detailed description of the mechanism is given in Chapter 10.

8. PRINTER-LESS TERMINAL

8.1. Message definition

For printer-less terminals, the ticket has to be printed on ECR. Moreover, the ECR has to compare the sign of the cardholder (if indicated by a terminal). In the event of an unmatched sign, it's the responsibility of the ECR to process the REVERSAL of the transaction (transaction type FID_T='10' with Authorisation-Code FID_F).

Because the terminal has to notify the ECR, that the ticket MUST BE PRINTED and that (if needed) the SIGN have to be VALIDATED, the FLAGS field of the protocol header is used. The information is carried in the final B2 authorisation response:

- Bit 0 : 0 = NO SIGNATURE required, 1 = SIGNATURE required
- Bit 1 : 0 = TICKET on CAD, 1 = ECR must print TICKET
- Bit 2-15 : not used
- Bit 13: 1 = ECR supports the Keep-alive mechanism on the TCP connection
- Bit 14: 1 = Purchase will check whitelist
- Bit 15: 1 = Receipt confirmation is requested from the cash register

For the purpose of the ticket, 2 new messages are defined:

- B3 - Ticket Request (cashregister -> terminal)
- B4 - Ticket Response (terminal -> cashregister)

Steps	Processing
0	The ECR gets the B2 result with FLG:Bit0 to 1
1	The ECR sends Request message B3 for a ticket portion
2	CAD sends ActivityConfirmation message to ECR immediately after B3 came
3	ECR gets B0 and waits B0 or B4 message
4	CAD sends B4 message with ticket portion to ECR
5	ECR sends confirmation message B0 into the CAD (timeout upto 5sec).

Table 8 *Description of transaction flow*

8.1.1. FID 9 - Terminal customer data

FID 9 is so-called subFID. That means, it doesn't carry information itself, but it is the 'envelope' for other customer FIDs FIDs. Inside this subFID, the FIDs are separated by Group Separator GS (1Dh).

8.1.2. FID 9t – Terminal ticket information

Ticket information. Fixed, 1 byte. Because of buffers limitation, one ticket can be split into more B3/B4 message pairs.

Request: Fixed, 1 byte. ECR specifies, which ticket and what portion it wants:
'M' - merchant, first part
'C' - customer, first part
' ' - continue with next part (space, ASCII 32, 0x20)

Response: Fixed, 1 byte. CAD indicates, if there's another portion of ticket:
'0' - this is a final part
'1' - more data to print, ECR should call B3 again

8.1.3. FID 9T – Terminal ticket line

Multiple tag, represents one logical line of printed text. Should not contain any ASCII control characters. Can contain ISO-8859-2 accented characters. The lines/tags are printed in the order received.

Structure of this tag:

- first character – FONT SELECTOR, select then font this line should be printed with
- next are the data for ticket line

Currently, there are 3 fonts used by the terminal:

Format	Font used
0	No font change, use the previous selected one
1	12 characters per line
2	24 characters per line
3	42 characters per line

Table 9 *Formats of used font*

Request: Not used.

Response: Variable, up to 43 characters.

8.2. Example of communication

ECR: `[STX]B3...[FS]9[GS]tM[ETX]`
CAD: `[STX]B0..[ETX]`
CAD: `[STX]B4..[FS]9[GS]t1[GS]T1text1[GS]T2text2[ETX]`
ECR: `[STX]B0..[ETX]`
ECR: `[STX]B3...[FS]9[GS]t[SPACE][ETX]`
CAD: `[STX]B0..[ETX]`
CAD: `[STX]B4..[FS]9[GS]t0[GS]T3text3[GS]T1end[ETX]`
ECR: `[STX]B0..[ETX]`

9. PAYMENT CARD SUPPORT WITH DIRECT DISCOUNTS

Payment cards with direct discounts are those credit cards, on which the merchant gives a discount. These cards are defined by their prefix. To use these cards allow terminal upload a list of these card prefixes. Items in this list contain the type of discount [1B] and prefix PAN card [1-7B].

To use this functionality are defining new fields for messages B protocol.

FID	Picture	Length (B)	Field Name	Rqst	Resp
9I	PIC H(8)	8	Whitelist ID/Version	✓	✓
9C	PIC 9	1-4	Whitelist items count	✓	✓
9D	PIC 9(1)	1-2	Whitelist loading Seq. no	✓	✓
9W	PIC X(8)	2-8	Whitelist item	✓	
9V	PIC X(1)	1	Whitelist info		✓
9E	PIC X(1)	1	POS Entry mode		✓
9Z	PIC H(24)	24	Card token		✓

Table 10 *Table of possible fids*

9.1. Optional data fields

Descriptions of FIDs are included on this and the following pages in the format listed below. Explanations of the headings are included in the format.

9.1.1. FID 9I - Whitelist ID/Version

This field is used to check the current version of the whitelist, is entered when creating a whitelist is returned when reading whitelist info. Its content is 8 hexadecimal digits [8B], these numbers have no significance for the terminal - it is the identifier for the cash register

Request: Mandatory for whitelist load start.

Response: Is returned by the whitelist info if it is defined whitelist.

9.1.2. FID 9C - Whitelist items count

The number of items in the whitelist, range 1-9999 [1-4B].

Request: Mandatory for whitelist load start.

Response: Is returned by the whitelist info if it is defined whitelist and it is a part of response to command whitelist load end.

9.1.3. FID 9N - Whitelist loading Seq. no

The number of the current sequence recording 1-99 [1-2B].

Request: Mandatory for whitelist load next.

Response: It is a part of response to command whitelist load next.

9.1.4. FID 9W - Whitelist item

This field contains one entry whitelist, which is formed and the tag 1B rebate card for which the prefix is determined by the discount this prefix is in the range of 1-7 digits PAN card. Length field 2-8B.

Request: Mandatory for whitelist load next.

Response: -

9.1.5. FID 9V - Whitelist info

This field is returned in case of split payment and indicates that the payment was made to set the type of card discounts, this type of discount [1B] It is 1.B of items whitelist.

Request: -

Response: As a result, split payment.

9.1.6. FID 9E - POS Entry mode

Field determines how the implementation of payment [1B].

- K Manually entered card
- C Payment by magnetic card
- S Chip card
- L Contactless card

Request: -

Response: As a result, split payment.

9.1.7. FID 9Z - Card token

Token generated by PAN card using a hash function 24 hexa numbers [24B], is used to check the card number during the subsequent partition of the payment see. below.

Request: -

Response: As a result, split payment.

9.2. New messages to support Whitelist

For uploading whitelist and check your current version serves the following messages:

T40 - whitelist info
T41 - whitelist load start
T42 - whitelist load next
T43 - whitelist load end

9.3. Whitelist info

This message is used to determine the current status of the terminal whitelist.

<- B1: T40
-> B2: T40 R [9I 9C]

9.4. Whitelist Loading

Uploading whitelist is done using several messages, begins with a message *whitelist loading start*.

<- B1: T41 9I 9C
-> B2: T41 R

Then follow your own items using whitelist messages *whitelist load next.*

<- B1: T42 9N 9W 9W ... 9W (max 15 items)
-> B2: T42 R 9N

A whole the uploading is terminated by *whitelist load end*.

<- B1: T43 9I 9C
-> B2: T43 R 9I 9C

If the uploading was properly the old whitelist is replaced.

9.5. Payment using Whitelist

Applying whitelist for payment is made by setting HEADER-FLAGS. If it is set to 0x4000 mask will make payments with respect to the whitelist, otherwise the whitelist when paying ignored.

Payment with immediate application of discounts:

<- B1: T00 (Flags | 0x4000 - split sale support) Bxxxxxxx
-> B2: T00 (Flags | 0x4000 - split sale transaction) R-99 [9V] 9E 9Z

When using split payment (payment using whitelist) payment after loading the card cash register and discontinued response is returned B2, which is the same as in B1 requirement is set the Flags | 0x4000 and the response code is -99, if prefix card fits into any of the prefixes whitelists is returned to the discount type in a field FID 9V, each response contains the FID field 9E - method of entering the card and FID 9Z - token generated based on the PAN card.

Completion of the split payment excluding contactless cards

Cash register could at this point to do one of the following operations:

- perform passivation = cancel the payment
- make re-payment (without using whitelistu) either the original or amount with apply the discount.

For these payments, magnetic card is not swiped again and chip card remaining in the reader - no need to check the card number.

Note: When a payment is canceled / passivated, the „transaction aborted“ message appears on the terminal. To suppress this statement, you need to set the "flag" entry to the value of || = 0x4000 (logical sum of 0x4000) when passing a message (T81) in the B-protocol header.

Completion of the split payment by contactless cards

For contactless card payment using a discount for the customer is necessary to read the payment card twice. At first the reading card number is readed for comparison to the whitelist and returns the appropriate field. Cash registers are able to interrupt payment on this step by performing passivation or complete payment for the same amount - will be re-payment without FLAGS set to 0x4000.
For the discount is necessary after reading the card on which you want to apply a discount to perform passivation and redo the split payment (FLAGS | 0x4000), this time with a reduced amount (next reading of customer's card), after loading the card again gets a cash register answer with the token in the field FID 9Z a cash register must check that is the same as the card at first reading. If the same token 9Z completes payment already reduced amount as the payments without a set FLAGS.

10. KEEP-ALIVE MECHANISM TO THE COMMUNICATION LINE

Keep-alive will only work on terminals with B-protocol in TCP variant. Serves to maintain an open TCP channel between cash register and the terminal. The terminal sends a keep-alive request to cash register And cash register answer them in this case. Activation mechanism requires support on both sides and is described in the next chapter

Keep-alive is implemented on the link layer and consists with challenge <ENQ> terminal and Answers <ACK> cash register If the cash register does not respond to the <ENQ> in 5 seconds, the line is considered to be interrupted and the terminal will attempt to reestablish a connection with the cash register.

The terminal will <ENQ> send, if any communication was on the line in the last 15 seconds. Any received (<STX> <ACK> <ENQ>) or sent (<STX>) data this timeout restarts.

Note: For the Tetra application, a keep-alive (ENQ) default timeout of 15 seconds (interval between ENQ from the terminal) is set, which can be changed by a parameter (eg from TMS or in the mcash bprotocol application menu).

Be careful that the <ENQ> and <ACK> on line can be interleaved with application B-protocol packet <STX> .... <ETX> or <ENQ> from the other communication side. This condition must be treated because it can lead to race-condition (both sides begin to independently broadcast):
- Terminal <ENQ> comes from the cashier response <STX> … <ETX>, takes it as a confirmation of keep-alive.
- To follow an unexpected <ACK> terminal does not respond, it ignores.

Note: Keep-alive primarily sends and cash register terminal reply, however implement solution is possible in the opposite direction. (No practical use in case of the current method of communication).

10.1. Keep-alive flag

To ensure compatibility with existing solutions in the field of B-flag 0x2000 protocol used in the header of **all messages**, as follows:

a) cash register setting this flag (in each B-protocol message) says it supports keep-alive
b) if it comes from the terminal (on request with the set flag) response also set Flagg 0x2000, it means that even the terminal supports keep-alive and turned it on.

11. SUPPORT OF 'OFFLINE TRANSACTIONS' IN B-PROTOCOL

MCBA supports the new 'backup' offline transactions at 'cash register' for special solutions now. It is the responsibility of the cash register to deliver these transactions (binary blob data) to the bank for processing (SFTP?).

Cash register indicates support of this feature by setting a bit in the flags 0x0800 B1 in the report for the start of the transaction.
If the terminal supports this feature and the transaction was offline, it indicates this fact by setting a bit in the flags 0x0800 B2 in response to the transaction.

Cash register readout data of transactions by using B5 / B6 messages.

B5 - request
Supported FID:

- 9X - format H4; offset for the data in the first request 0000, in another echo of answers

B6 - response:
Supported FIDs:

- 9Y - X32 transaction name format, composed of PosID + date and time
- 9X - format H4; offset for the next request, if this is already the last, is not present
- 9x - X400 format; Base64 encoding of the binary data transaction blob

12. N PROTOCOL

In addition to the MBCA payment application, which communicates via the B-protocol, there are more on the terminal Functionality / application. E.g. the Maintenance application, which communicates with the N-protocol cashier.

The N protocol differs from the B protocol in particular by the 'N' in the header of the protocol as opposed to the original 'B'. This change was forced by the coexistence of multiple applications on the terminal connected to the cash register via the RS232 interface.
When communicating over the UDP protocol, there is also a difference in the port used, for N protocol 33332 is reserved.

Because this protocol is used to control an application that administers applications in the terminal, this protocol contains only two commands:

- GET APP INFO (80)
- TMS CALL (90)

errorrespo
These N protocol commands have a similar function to the B protocol. GetAppInfo is used to get the application version and TMSCall to make a call to the POSMANAGEMENT server. During this call, Maintenance can download new versions of applications to the terminal and perform terminal keys update.

Note: The terminal can also include other applications, a description of these protocols are engaged in other documents.

Report usage examples:

12.1. Get App Info

Request:

```
  0x0000: 02 4e 31 30 31 20 20 20 20 20 20 20 20 31 37 30 = '.N101        170'
  0x0010: 35 32 39 31 32 31 38 34 34 30 30 30 30 30 30 30 = '5291218440000000'
  0x0020: 34 41 35 41 35 1c 54 38 30 03                   = '4A5A5.T80.      '
```

Response:

```
  0x0000: 02 4e 32 30 31 20 20 20 20 20 20 20 20 31 37 30 = '.N201        170'
  0x0010: 35 32 39 31 33 31 37 30 36 30 30 30 30 30 30 31 = '5291317060000001'
  0x0020: 33 41 35 41 35 1c 54 38 30 1c 52 30 30 30 1c 67 = '3A5A5.T80.R000.g'
  0x0030: 56 3a 31 2e 32 2e 34 31 03                      = 'V:1.2.41.       '
```

12.2. TMS Call

Request:

```
  0x0000: 02 4e 31 30 31 20 20 20 20 20 20 20 20 31 37 30 = '.N101        170'
  0x0010: 35 32 39 31 32 32 34 32 35 30 30 30 30 30 30 30 = '5291224250000000'
  0x0020: 34 41 35 41 35 1c 54 39 30 03                   = '4A5A5.T90.      '
```

Progress message:

```
  0x0000: 02 4e 30 30 31 20 20 20 20 20 20 20 20 31 37 30 = '.N001        170'
  0x0010: 35 32 39 31 33 32 33 34 36 30 30 30 30 30 30 30 = '5291323460000000'
  0x0020: 30 41 35 41 35 03                               = '0A5A5.          '
```

Response: (Connection error)

```
  0x0000: 02 4e 32 30 31 20 20 20 20 20 20 20 20 31 37 30 = '.N201        170'
  0x0010: 35 32 39 31 33 32 33 34 36 30 30 30 30 30 30 30 = '5291323460000000'
  0x0020: 39 41 35 41 35 1c 54 39 30 1c 52 2d 31 39 03 = '9A5A5.T90.R-19. '
```

Response: (Connection OK)

```
  0x0000: 02 4e 32 30 31 20 20 20 20 20 20 20 20 31 37 30 = '.N201        170'
  0x0010: 35 32 39 31 33 32 33 34 36 30 30 30 30 30 30 30 = '5291323460000000'
  0x0020: 39 41 35 41 35 1c 54 39 30 1c 52 30 30 30 03 = '9A5A5.T90.R000. '
```

13. PAYMENT APPLICATION REQUIREMENTS

Payment applications for payment terminals should support at least a minimum set of terminal commands to ensure successful payment execution.

You need to support these commands:

- N-Protocol: TMS Call
- B-Protocol: TMS Call
- B-Protocol: Handshake
- B-Protocol: Normal purchase
- B-Protocol: Reversal
- B-Protocol: Close Totals

This listing is valid for terminals where the payment application card is printed on the terminal printer. In case of printing tickets to the cash register printer, it is also necessary to support commands to read the ticket data.

The first two TMS Call commands are used to call up the terminal server with the messaging server and to check the up-to-date version of the applications and their parameters, Handshake is a diagnostic command to check the connection to the authorization server, and also to check the correct configuration of the terminal's communication keys.

The Normal Purchase command is a custom payment, when the terminal waits for a payment card and authorizes the requested payment when the parameters of this command are met correctly.

The Reversal command is used to cancel a previous payment, primarily for payments where cardholder signature verification is required, but can generally cancel payments until the terminal closes.

The Close Totals command is used to execute the shutdown of the terminal and to send offline transactions from the terminal to the authorization server.

The description of the individual commands and their Fid and SubFid parameters is discussed in the previous chapters.

APPENDIX 1 : Server Response Codes

Approved Codes

000 = Approved online
001 - 009 = Approved
010 = Partial approvement

Declined Codes

050 = General
051 = Server not connected
052 = Number of PIN tries exceeded
053 = Server connected but request error
055 = Transaction denied by EMV chip card after 1.decision
056 = Transaction denied by EMV chip card after online authorisation (2.decision)
060 = Merchant cancelation
061 = Customer cancelation
062 = Customer declines amount
063 = Timeout cancellation
070 = Reversal unsuccessful
100 = Not allowed transaction
100 = Unsupported transaction
101 = Wrong card
102 = Expired card
103 = Message Data Format Error
104 = CVC error

+ many more, see ACI SPDH manual

APPENDIX 2 : Terminal Response Codes

```
-01 = MERCH_ERR_USER     1 //  normalni ukonceni (uzivatel zrusil castku, PIN, ....)
-02 = MERCH_ERR_CANTDOIT  2 //  nemohu provest
-03 = MERCH_ERR_NOPARAM   3 //  nejsou definovany parametry karty
-04 = MERCH_ERR_KOPARAM   4 //  chyba v parametrech karty
-05 = MERCH_ERR_TABLE     5 //  obecna chyba tabulky
-06 = MERCH_ERR_DIAL      6 // nemohu se dovolat
-07 = MERCH_ERR_UNKNCARD  7 //  neznamy typ karty
-08 = MERCH_ERR_NOTPRIVILEGED 8 //  nemam opravneni
-09 = MERCH_ERR_NOTVALID  9 //  chyba karty (spatny PAN len, luhn, service)
-10 = MERCH_ERR_EXPIRED   10  // prosla
-11 = MERCH_ERR_TOOYOUNG  11 // jeste neplati
-12 = MERCH_ERR_TOOLITTLE  12 // prilis maly obnos -> pouzijte cash
-13 = MERCH_ERR_TOOBIG    13 // prilis velky obnos, pouzijte ?!?!?!?
-14 = MERCH_ERR_SAMEPAN   14 // dve operace se stejnou kartou po sobe
-15 = MERCH_ERR_SERVICE   15 // chyba servisniho kodu
-16 = MERCH_ERR_PINPAD    16 // chyba pinpadu
-17 = MERCH_ERR_TRANS_KEY  17 // chyba transportniho klice
-18 = MERCH_ERR_TIMEOUT   18  // run out of time ..........

-19 = MERCH_ERR_AUTH      19  // chyba autorizace
-20 = MERCH_ERR_FULL      20  // no more place to make transaction ...
-21 = MERCH_ERR_INTERN    21 // internal, unspecified error
-22 = MERCH_ERR_MASTER       22      //  error  in  datas  suplier  by  master,  cash register,  ...
-23 = MERCH_ERR_SIGN      23  // signs don't match

-24 = MERCH_ERR_EMV                      24      // EMV card & not supported callback
-25 = MERCH_ERR_ARPC  25  // EMV card rejected, after online on 2nd GENERATE AC command
-29 =MERCH_ERR_BLOCK  29 // EMV card blocked !!!!

-99= Special, split-sale intermediate result  99
```

APPENDIX 3 : Transaction trace example

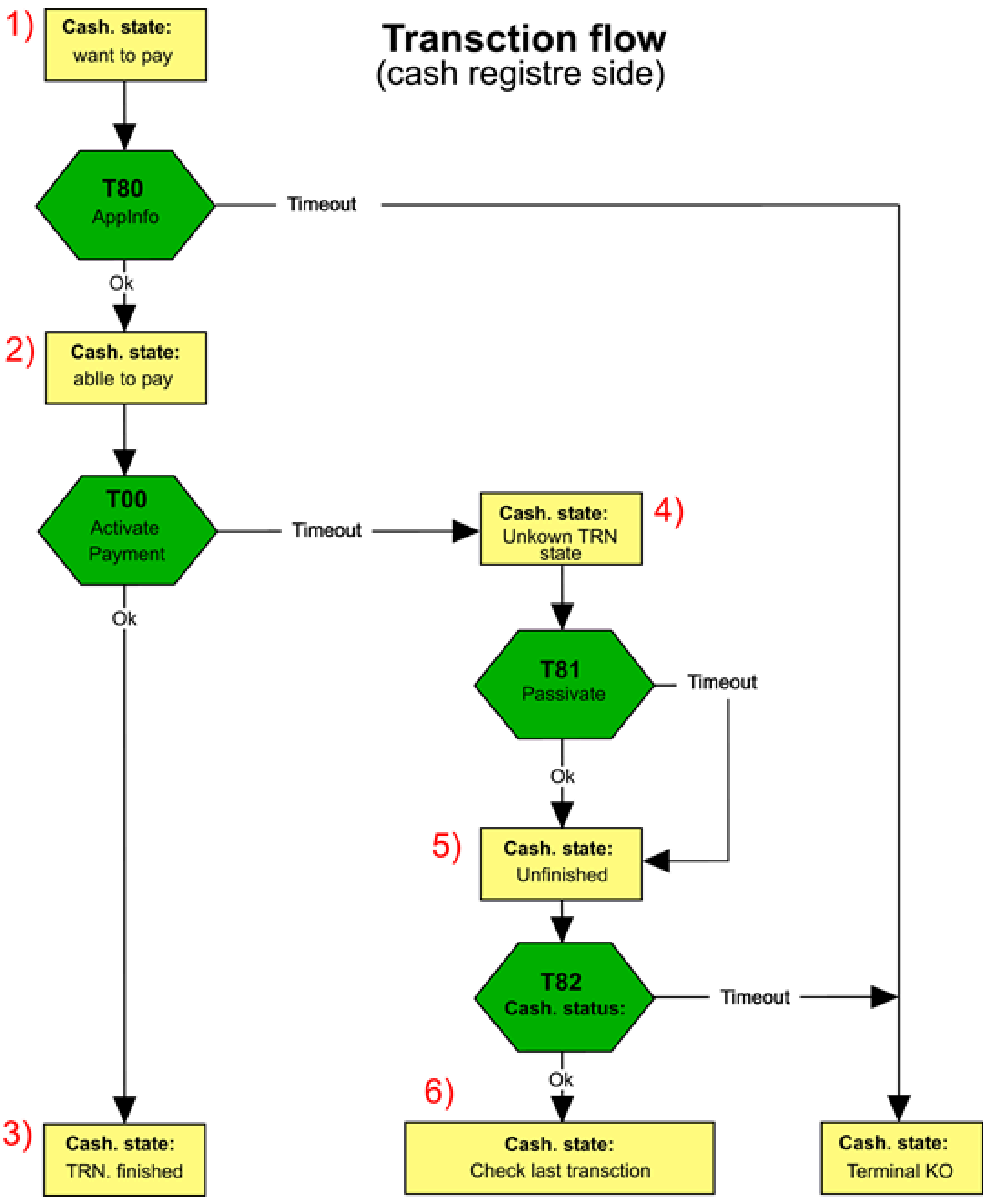
X - control characters
X - alternate fields in the header
X - FID identifiers

```
Packet 'Send': len=55
 0x0000: 02 42 31 30 31 20 20 20 20 20 20 20 20 31 30 30 = '.B101        100'
 0x0010: 35 31 33 31 33 32 36 31 31 30 30 30 30 30 30 31 = '5131326110000001'
 0x0020: 31 41 35 41 35 1c 54 30 30 1c 42 31 30 30 1c 45 = '1A5A5.T00.B100.E'
 0x0030: 32 30 33 1c 44 31 03                            = '203.D1.         '
BxSend(): send() -> 55

Packet 'Recv': len=38
 0x0000: 02 42 30 30 31 50 56 54 4d 4f 4e 32 00 31 30 30 = '.B001PVTMON2.100'
 0x0010: 35 31 33 31 35 32 36 31 30 30 30 30 30 30 30 30 = '5131526100000000'
 0x0020: 30 41 35 41 35 03                               = '0A5A5.          '
Got packet: B0
    Activity/Confirmation

Packet 'Recv': len=38
 0x0000: 02 42 30 30 31 50 56 54 4d 4f 4e 32 00 31 30 30 = '.B001PVTMON2.100'
 0x0010: 35 31 33 31 35 32 36 31 30 30 30 30 30 30 30 30 = '5131526100000000'
 0x0020: 30 41 35 41 35 03                             = '0A5A5.          '
Got packet: B0
    Activity/Confirmation

Packet 'Recv': len=38
 0x0000: 02 42 30 30 31 50 56 54 4d 4f 4e 32 00 31 30 30 = '.B001PVTMON2.100'
 0x0010: 35 31 33 31 35 32 36 31 31 30 30 30 30 30 30 30 = '5131526110000000'
 0x0020: 30 41 35 41 35 03                             = '0A5A5.          '
Got packet: B0
    Activity/Confirmation

Packet 'Recv': len=38
 0x0000: 02 42 30 30 31 50 56 54 4d 4f 4e 32 00 31 30 30 = '.B001PVTMON2.100'
 0x0010: 35 31 33 31 35 32 36 31 36 30 30 30 30 30 30 30 = '5131526160000000'
 0x0020: 30 41 35 41 35 03                             = '0A5A5.          '
Got packet: B0
    Activity/Confirmation

Packet 'Recv': len=38
 0x0000: 02 42 30 30 31 50 56 54 4d 4f 4e 32 00 31 30 30 = '.B001PVTMON2.100'
 0x0010: 35 31 33 31 35 32 36 31 37 30 30 30 30 30 30 30 = '5131526170000000'
 0x0020: 30 41 35 41 35 03                             = '0A5A5.          '
Got packet: B0
    Activity/Confirmation

Packet 'Recv': len=38
 0x0000: 02 42 30 30 31 50 56 54 4d 4f 4e 32 00 31 30 30 = '.B001PVTMON2.100'
 0x0010: 35 31 33 31 35 32 36 31 38 30 30 30 30 30 30 30 = '5131526180000000'
```

```
  0x0020: 30 41 35 41 35 03                           = '0A5A5.       '
Got packet: B0
        Activity/Confirmation

Packet 'Recv': len=38
  0x0000: 02 42 30 30 31 50 56 54 4d 4f 4e 32 00 31 30 30 = '.B001PVTMON2.100'
  0x0010: 35 31 33 31 35 32 36 31 39 30 30 30 30 30 30 30 = '5131526190000000'
  0x0020: 30 41 35 41 35 03                          = '0A5A5.            '
Got packet: B0
        Activity/Confirmation

Packet 'Recv': len=38
  0x0000: 02 42 30 30 31 50 56 54 4d 4f 4e 32 00 31 30 30 = '.B001PVTMON2.100'
  0x0010: 35 31 33 31 35 32 36 32 30 30 30 30 30 30 30 30 = '5131526200000000'
  0x0020: 30 41 35 41 35 03                           = '0A5A5.           '
Got packet: B0
        Activity/Confirmation

Packet 'Recv': len=38
  0x0000: 02 42 30 30 31 50 56 54 4d 4f 4e 32 00 31 30 30 = '.B001PVTMON2.100'
  0x0010: 35 31 33 31 35 32 36 32 31 30 30 30 30 30 30 30 = '5131526210000000'
  0x0020: 30 41 35 41 35 03                           = '0A5A5.           '
Got packet: B0
        Activity/Confirmation

Packet 'Recv': len=38
  0x0000: 02 42 30 30 31 50 56 54 4d 4f 4e 32 00 31 30 30 = '.B001PVTMON2.100'
  0x0010: 35 31 33 31 35 32 36 32 31 30 30 30 30 30 30 30 = '5131526210000000'
  0x0020: 30 41 35 41 35 03                           = '0A5A5.           '
Got packet: B0
        Activity/Confirmation

Packet 'Recv': len=38
  0x0000: 02 42 30 30 31 50 56 54 4d 4f 4e 32 00 31 30 30 = '.B001PVTMON2.100'
  0x0010: 35 31 33 31 35 32 36 32 31 30 30 30 30 30 30 30 = '5131526210000000'
  0x0020: 30 41 35 41 35 03                           = '0A5A5.           '
Got packet: B0
        Activity/Confirmation

Packet 'Recv': len=38
  0x0000: 02 42 30 30 31 50 56 54 4d 4f 4e 32 00 31 30 30 = '.B001PVTMON2.100'
  0x0010: 35 31 33 31 35 32 36 32 31 30 30 30 30 30 30 30 = '5131526210000000'
  0x0020: 30 41 35 41 35 03                           = '0A5A5.           '
Got packet: B0
        Activity/Confirmation

Packet 'Recv': len=38
  0x0000: 02 42 30 30 31 50 56 54 4d 4f 4e 32 00 31 30 30 = '.B001PVTMON2.100'
  0x0010: 35 31 33 31 35 32 36 32 34 30 30 30 30 30 30 30 = '5131526240000000'
  0x0020: 30 41 35 41 35 03                             = '0A5A5.         '
Got packet: B0
        Activity/Confirmation
```

```
Packet 'Recv': len=38
  0x0000: 02 42 30 30 31 50 56 54 4d 4f 4e 32 00 31 30 30 = '.B001PVTMON2.100'
  0x0010: 35 31 33 31 35 32 36 32 36 30 30 30 30 30 30 30 = '5131526260000000'
  0x0020: 30 41 35 41 35 03                               = '0A5A5.          '
Got packet: B0
        Activity/Confirmation

Packet 'Recv': len=38
  0x0000: 02 42 30 30 31 50 56 54 4d 4f 4e 32 00 31 30 30 = '.B001PVTMON2.100'
  0x0010: 35 31 33 31 35 32 36 32 36 30 30 30 30 30 30 30 = '5131526260000000'
  0x0020: 30 41 35 41 35 03                               = '0A5A5.          '
Got packet: B0
        Activity/Confirmation

Packet 'Recv': len=38
  0x0000: 02 42 30 30 31 50 56 54 4d 4f 4e 32 00 31 30 30 = '.B001PVTMON2.100'
  0x0010: 35 31 33 31 35 32 36 32 37 30 30 30 30 30 30 30 = '5131526270000000'
  0x0020: 30 41 35 41 35 03                               = '0A5A5.          '
Got packet: B0
        Activity/Confirmation

Packet 'Recv': len=38
  0x0000: 02 42 30 30 31 50 56 54 4d 4f 4e 32 00 31 30 30 = '.B001PVTMON2.100'
  0x0010: 35 31 33 31 35 32 36 32 37 30 30 30 30 30 30 30 = '5131526270000000'
  0x0020: 30 41 35 41 35 03                               = '0A5A5.          '
Got packet: B0
        Activity/Confirmation

Packet 'Recv': len=61
  0x0000: 02 42 32 30 31 50 56 54 4d 4f 4e 32 00 31 30 30 = '.B201PVTMON2.100'
  0x0010: 35 31 33 31 35 32 36 33 30 30 30 30 30 30 30 31 = '5131526300000001'
  0x0020: 37 41 35 41 35 1c 52 31 35 30 1c 50 2a 2a 2a 2a = '7A5A5.R150.P****'
  0x0030: 2a 2a 2a 2a 2a 2a 2a 2a 30 35 33 34 03          = '********0534.   '
Got packet: B2
        Response
Payment: RC='150', Auth='', PAN='************0534', SeqID=''

Packet 'Send': len=38
  0x0000: 02 42 30 30 31 20 20 20 20 20 20 20 20 31 30 30 = '.B001        100'
  0x0010: 35 31 33 31 33 32 36 31 31 30 30 30 30 30 30 31 = '5131326290000000'
  0x0020: 30 41 35 41 35 03                               = '0A5A5.          '
BxSend(): send() -> 38
```

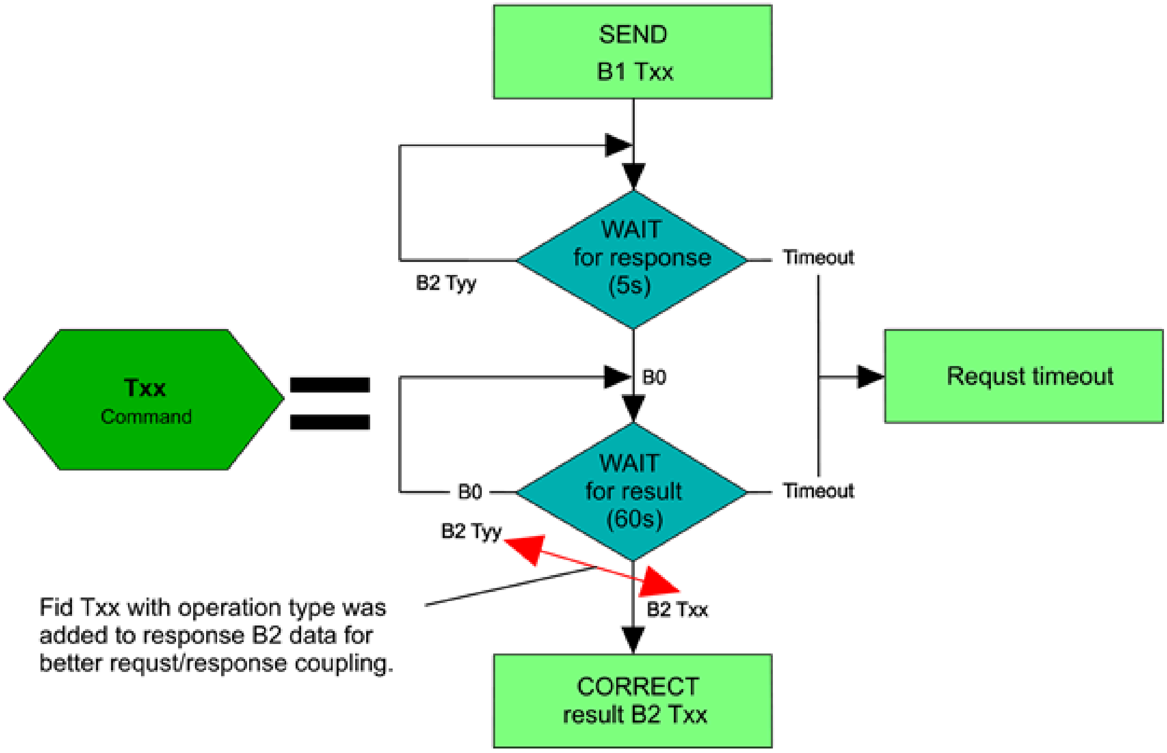
APPENDIX 4 : Transaction flowchart



Description of picture

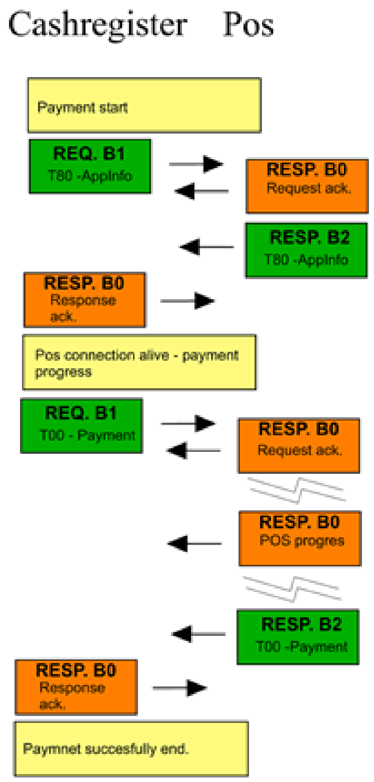
1) Terminal is in an idle state and registered cash is ready to pay. Cash tests the state of the terminal in the first step. The function AppInfo (command T80, viz. description of message protocol) are used for terminal status checking. When this function fails it is necessary to check the connection of the terminal and it is unable to start payment.
2) Registered cash receives terminal response and prepares a sale operation that sends to the terminal (command T00, data of command description of message protocol). If cash registers do not obtain response, it will try to find out operation status. There are two different options of fault:
   - Command did not come to the terminal.
   - Cash registered didn't get response from terminal after successful payment
3) Correct answer arrived from terminal so cash registered parse Response code and match Variable symbol and amount in this answer with previous request value.
4) Cash registered have to send Passivate command to Terminal in case of incorrect response for sale operation (command T81, description of message protocol) to ensure terminal stop working.
5) Cash registered send command LastTrn (command T82, description of message protocol) to get the last transaction provided by terminal and result of this last transaction.
6) Cash registered got last TRN of terminal. Now, cash registered have compare data of their last transaction with data from terminal. And again there are two possible results:
   - Variable symbol and amount of Terminal data are the same as Cash registered transaction data. Result of this payment depends on the value of ResponseCode.
   - Variable symbol and amount of Terminal data are not the same as Cash registered transaction data. It means that the request of the payment didn't arrive at the Terminal. So payment wasn't done and cash registered can repeat to send payment request.

One step of payment transction
(detail of communication)



Fid Txx with operation type was added to response B2 data for better requst/response coupling.

Normal payment flow



Cashregister Pos

Payment start

REQ. B1
T80 -AppInfo

RESP. B0
Request ack.

RESP. B2
T80 -AppInfo

RESP. B0
Response ack.

Pos connection alive - payment progress

REQ. B1
T00 - Payment

RESP. B0
Request ack.

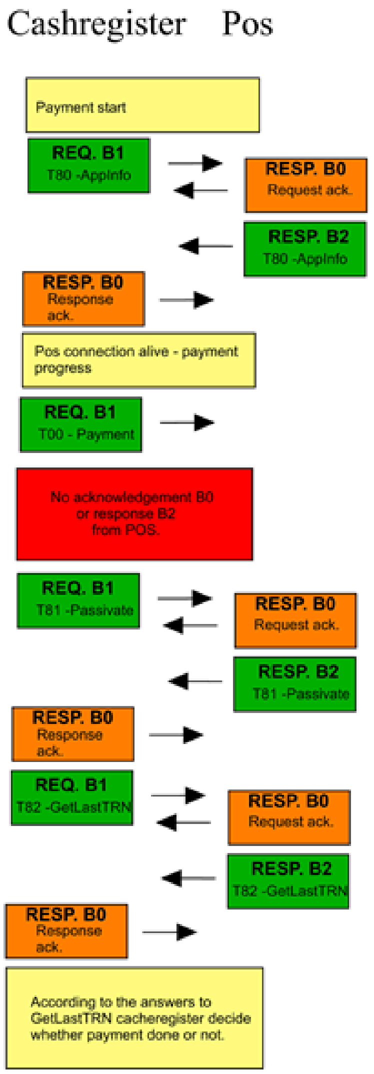
RESP. B0
POS progres

RESP. B2
T00 -Payment

RESP. B0
Response ack.

Paymnet succesfully end.

Payment with comunication errors



Cashregister Pos

Payment start

REQ. B1
T80 -AppInfo

RESP. B0
Request ack.

RESP. B2
T80 -AppInfo

RESP. B0
Response ack.

Pos connection alive - payment progress

REQ. B1
T00 - Payment

No acknowledgement B0 or response B2 from POS.

REQ. B1
T81 -Passivate

RESP. B0
Request ack.

RESP. B2
T81 -Passivate

RESP. B0
Response ack.

REQ. B1
T82 -GetLastTRN

RESP. B0
Request ack.

RESP. B2
T82 -GetLastTRN

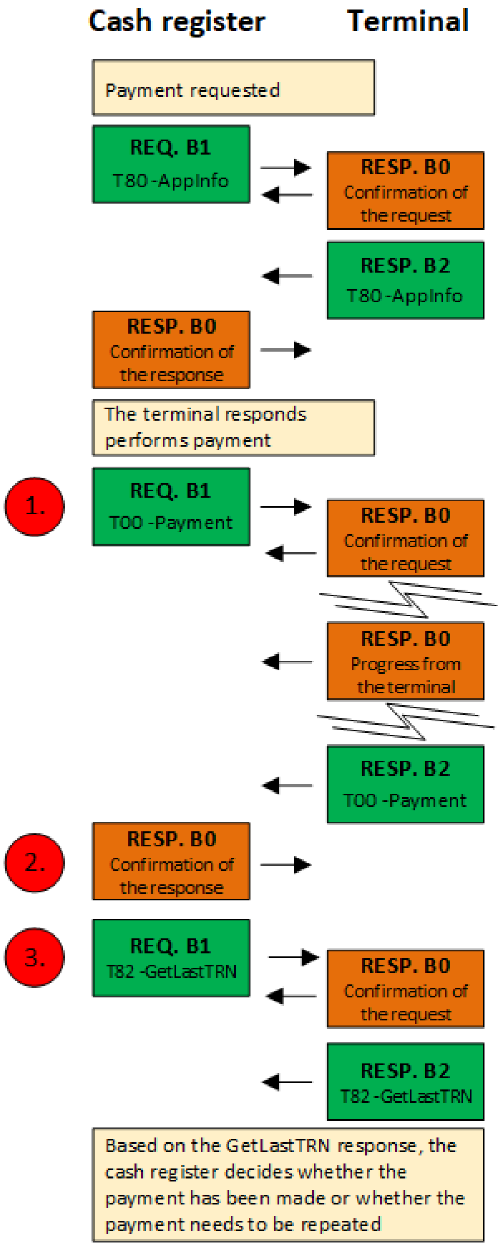
RESP. B0
Response ack.

According to the answers to GetLastTRN cacheregister decide whether payment done or not.

APPENDIX 5: Payment transactions with explicit confirmation

The explicit payment confirmation mechanism ensures automatic cancellation of the payment (Reversal) in case the loss of connectivity between the payment terminal and the cash register. At the same time, however, it can cause automatic cancellation even if the cash register fails to deliver the confirmation within 5 seconds of sending the payment result.

When using explicit confirmation, we recommend that after sending a confirmation of the payment result by the cash register, execute the T82 Get Last transaction command and compare whether the response to T82 is the same as the previous payment result to verify that the terminal received the payment confirmation on time and the payment was not reversed.


Cash register
Terminal
Payment requested
REQ. B1
T80 -AppInfo
RESP. B0
Confirmation of the request
RESP. B2
T80 -AppInfo
RESP. B0
Confirmation of the response
The terminal responds performs payment
1.
REQ. B1
T00 -Payment
RESP. B0
Confirmation of the request
RESP. B0
Progress from the terminal
RESP. B2
T00 -Payment
2.
RESP. B0
Confirmation of the response
3.
REQ. B1
T82 -GetLastTRN
RESP. B0
Confirmation of the request
RESP. B2
T82 -GetLastTRN
Based on the GetLastTRN response, the cash register decides whether the payment has been made or whether the payment needs to be repeated

Note:

1. Payment with explicit confirmation request
2. Confirmation of receipt of the result of payment by the cash register
3. Verify delivery of the confirmation, by detecting the last transaction performed

APPENDIX 6: FAQ

Questions about B protocol:

1. *How does the Keep Alive mechanism work in case of setting the TCI/IP/Ethernet communication line - terminal SERVER mode, it is not clear from the text what is "B-protocol in TCP terminal" see description in chapter Keep-alive mechanism?*

Mode setting

- TCP / IP / Ethernet - terminal CLIENT mode
- TCP / IP / Ethernet - terminal SERVER mode

It affects the TCP connection of the terminal and the cash register, which initiates a TCP connection from these devices.

As for the Keep-alive mechanism, it is the same in both modes, after its start, for example with the GET APP INFO command with the flag set to 0x2000, the terminal starts sending the <ENQ> character, to which the <ACK> character expects a response from the cash register.

See Chapter 10. Keep-alive.

2. *Chapter 6 Transaction progress is a sequence diagram describing the execution of a payment transaction. Are the communications different for CLIENT mode and SERVER mode?*

In all ways of connecting the cash register and the terminal (serial, UDP, TCP-Server, TCP-Client), there is a master cash register in the B-protocol and the slave terminal, ie the cash register sends requests and the terminal responds.

3. *Is it possible to cancel a payment with the T81 PASSIVATE command before confirming it with a B2 message, or can the terminal reject the command (eg it has already been completed at the bank)?*

It is always possible to send, but it is not possible to pass the established communication with the authorization server. That is, if you send a PASSIVATE while waiting for a card or PIN, the transaction will be interrupted if the terminal receives a PASSIVATE only during communication with the authorization server - the payment transaction will not be interrupted and you will receive the result of the payment transaction.

4. *Does transaction T10 REVERSAL apply only to the last executed transaction? If not until when (under what conditions) can REVERSAL be performed?*

Yes, REVERSAL can only be performed on the last payment transaction until the closing date. This order is intended for the cancellation of a payment transaction in case of disagreement of the signature or non-printing of tickets for the payment transaction, so it is rather assumed that it will be used immediately after the payment transaction.

5. *What is a T05 CASH ADVANCE transaction?*

Cash withdrawal - hotels, exchange offices.

6. *What* is a T95 HANDSHAKE transaction for?

To verify the connection of the terminal to the authorization server.

7. *What does T65 SUBTOTALS return?*

It exchanges totals with the authorization server, as at the closing batch (T60 CLOSE TOTALS), but does not reset them.

8. *Does a T60 CLOSE TOTALS transaction return values from or initiate the last closing batch initiated manually on the terminal?*

T60 CLOSE TOTALS - performs the closing batch.

9. *We assume that the cash register must be connected to the terminal for the entire duration of the ongoing payment. Or just ask for a task and then ask for the result?*

Yes, the connection between the cash register and the terminal should not be interrupted. If it does not receive a response to the authorization, there is a mechanism for detecting the last response (GetLastTransaction), but it is more intended to deal with emergency situations.