

ARCUS 2.1

Software for
POS Integration

Administrator Guide

/ Ver.1.10

Content

History of changes	4
Conventions	4
Glossary	5
Abbreviations	6
Introduction	8
1_ ARCUS 2.1 CAP	9
1_1 Software component structure	9
1_1_1 TLS Encryption in "IP via Cash Register" Mode	10
1_2 Cash Register Mode	10
1_2_1 ARCUS.CFG for ISO Processing Systems (OpenWay, SmartVista, Tieto, CTL) ...	11
1_2_2 ARCUS.CFG for Unipay Processing Systems (CompassPlus, SPDH)	13
1_3 Integration	15
1_3_1 Format and contents of output.out file	17
1_4 ARCUS 2.1 test	18
1_4_1 Windows / Linux	18
1_5 Initialization file Cashreg.ini	20
1_6 ARCUS 2 CAP Settings	23
1_6_1 Ops.ini (ISO 8583)	23
1_6_2 ops.ini for Compass+ and SPDH	25
1_6_3 Rc_conv.ini	26
1_6_4 Rc_res.ini	27
1_7 Messages and Menu output on terminal screen (Dialog Direct)	27
1_8 External dialogs	28
1_9 Maintenance	29
1_9_1 Integration of ARCUS2.1 CAP with cash register software	29
1_9_2 Virtual Keyboard	29
1_9_3 Methods and Frequency of Status Monitoring	30
1_9_4 Troubleshooting	30
1_9_5 Log File Description (Fragments)	30
1_9_6 Refund without card	30
2_ Integration via RS232 or USB	31
3_ Low-level integration	32
Annex 1 Keys for set/get methods	33
Annex 2 Example of Library Usage	34
Function Prototypes	34
Example for C++	34

Example for C#.....	37
Example for Object Pascal (Delphi).....	38
Annex 3 List of supported operations.....	42
Annex 4 Prototypes of Arcus.h dialog functions	43
Example	44

History of changes

/ Table 1. History of changes

Version	Date	Author	Description
1.10	18/03/2019	Popova I.L.	Description of 3,9,Terminal Activation command is added [#57557]
1.10	06/03/2019	Popova I.L.	Description of AUTODETECT_OPS parameter in cashreg.ini file is added [#53861]
1.09	28/12/2018	Popova I.L.	Description of transaction_id key is added [#56119] Description of the 2.38 print receipt copy by transaction_id is added [#51140]
1.08	26/10/2018	Popova I.L.	Description of 1.109 Alipay Purchase and 1.110 Alipay Refund is added [#54891]
1.07	19/09/2018	Popova I.L.	Decrtiption of new 3.8 TMS SESSION BY DEFAULT SCRIPT command is added [#53557]
1.06	22/06/2018	Vlasenko I.L.	Parameter terminal_serial_number is added to SET/GET methods, Annex 4 is changed [#50735], [#51424]
1.04	04/05/2018	Vlasenko I.L.	External dialogs [#50735], Messages and Menu output on Terminal screen (Dialog Direct) are added, keys --origin-oper-type=N, --repeat и --dialog-direct are added to commandlinetool
1.02	19/02/2018	Vlasenko I.L.	Command for UID Mifare reading [#46197]
1.02	08/02/2018	Vlasenko I.L.	Annex 4 is added
1.01	18/10/2017	Vlasenko I.L.	Annex 1 : receipt_number parameter is added Annex 2 : new function is added.
1.0	27/06/2017	Vlasenko I.L.	Creation if the document (based on ARCUS2 Admin guide)

Conventions

The following conventions are used in the document:

/ Table 2. Conventions

Symbol	Description
	Note!
	Warning! Be careful with this information

Glossary

/ Table 3. Glossary

Term	Definition
Acquirer, Acquiring bank	A member of a Card Association, for example MasterCard and/or Visa, which maintains merchant relationships and receives all bankcard transactions from the merchant
Cardholder Verification Method	Method of verifying that a purchaser has the right to use the card being offered (e.g. Signature or PIN)
Communication Script	A sequence of commands (mini-program) that defines scripts for connecting POS-terminal to an authorization server or TMS server
EMV	A standard for interoperation of IC cards ("Chip cards") and IC capable POS terminals and ATMs, for authenticating credit and debit card payments (comes from the initial letters of Europay, MasterCard and VISA)
EMV Applications	EMV products designed to support terminals
Host, Communication Server	Payment system database server that sends responds to POS-terminal requests and provides terminals with requires information
Integrated Circuit(s) Card (ICC)	A card into which one or more integrated circuits are inserted to perform processing and memory functions.
Issuer	A legal entity that develops, registers and sells securities for the purpose of financing its operations
Issuer's Script	A command or a string of commands transmitted by the issuer to the terminal for the purpose of being sent serially to the ICC as commands
Magnetic Stripe	The stripe containing magnetically encoded identification information about a card and its holder
Master Key (MK)	ICC Master Key for session key generation
Merchant Category Code (MCC)	A four-digit number assigned to classify the business by the type of goods or services it provides
PIN code	A secret numeric password shared between a user and a system that can be used to authenticate the user to the system
PIN Pad	Arrangement of numeric and command keys to be used for personal identification number (PIN) entry
POS-Terminal, Terminal	The device used in conjunction with the ICC at the point of transaction to perform a financial transaction. The terminal incorporates the interface device and may also include other components and interfaces such as host communications
Profile	One of control objects of the system; it stores groups of customized settings that defines terminal's work algorithm when performing some card or administrative operations
Reference	One of control objects of the system; it stores basic values, templates, terms and algorithms used in POS management.
Service Card Code	Card code that denotes its application areas and type

Term	Definition
Software	A linking element of all parts of payment system. Software is divided into: • POS-Terminal application; • Self-service terminal application; • PIN pad application; • POS-Terminal Servers; • Self-service Terminal Servers; • Cashier Server; • BackOffice Server; • Card application; • Special software
Terminal Action Code	An action performed by terminal's application at a certain case. TAC's value are Default, Denial, Online
Transaction	A unit of work performed within a database management system (or similar system) against a database, and treated in a coherent and reliable way independent of other transactions. An action taken by a terminal at the user's request. For a POS terminal, a transaction might be payment for goods, etc. A transaction selects among one or more applications as part of its processing flow
TMS Update Server	Terminal update server that sends information about updated terminals to the monitoring console

Abbreviations

/ Table 4. Abbreviations

Abbreviation	Stands for
AID	Application Identification Number
ATM	Automated Teller Machine
AVN	Application Version Number
BIN	Bank Identification Number
CVM	Cardholder Verification Method
DDOL	Dynamic Data Authentication Data Object List
ICC	Integrated Circuit(s) Card
ID	Identifier
ISO	International Organization for Standardization
MAC	Message Authentication Code
MCC	Merchant Category Code
MK	Master Key
MS	Magnetic Stripe
TAC	Terminal Action Code(s)
PAN	Primary Account Number
PDOL	Processing options data object list
PIN	Personal Identification Number
POS	Point of Sale
RID	Registered Application Provider Identifier
RDMS	Relational database management system

SHA-1	Secure Hash Algorithm 1
SSL	Secure Sockets Layer
TDOL	Transaction Certificate Data Object List
TMM	Terminal Management Mode
TMS	Terminal Management System

Introduction

- This document is developed by the Ingenico (Russia). It contains complete and detailed description of ARCUS 2.1 integration library for Terminal Payment Application Universal EMV POS and Cash Register



This document is additional for ARCUS 2 Administrator Guide.

All users and administrators working with this software must undergo training and obtain a certificate in Ingenico LTD or other partner company.

If any questions about the software appears, please contact Ingenico support team:
support@ingenico.ru.

There may be some slight discrepancy between this documentation and the software associated with development of software products.

It is not allowed to use texts and images included in this document, without the consent of Ingenico LTD.

1_ARCUS 2.1 CAP

Arcus 2.1 CAP (Cash register Application) system is a library that is used for integration of Cash Register Application with Ingenico terminal equipment.

Cash Register and Terminal Equipment use ARCUS2 protocol for data transfer. to perform payment transactions on Ingenico Terminals with the help of Bank Cards (Magnetic cards, Smart Cards (EMV), Contactless, Manual input).

ARCUS 2.1 CAP functions:

- ARCUS2.1 exchange protocol for POS equipment data exchange;
- Cash Register communications can be used to connect the Bank Host (Cash Reg. IP mode)
- User dialog can be displayed on the Cash Register screen (Cash Reg. Dialog mode).

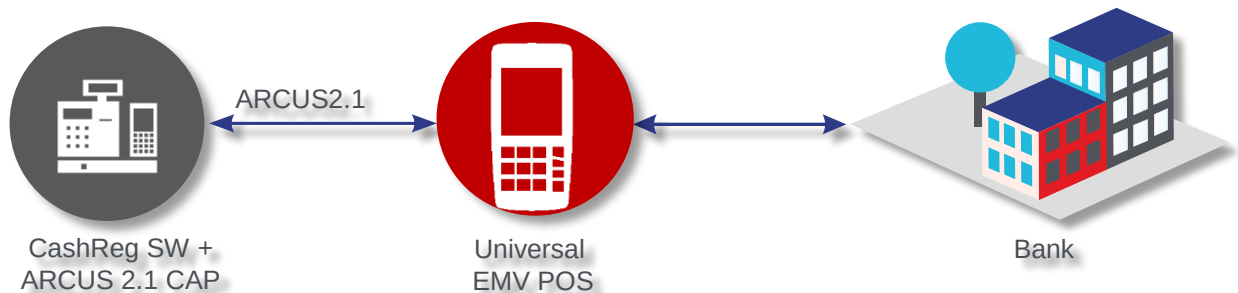


Fig. 1. Arcus 2.1 CAP Interaction scheme

ARCUS 2.1 CAP – is used to integrate Cash Register equipment and Ingenico Telium 1/2, Unicapt32 terminals.

Software works with the following OS:

- Windows (XP SP3 and above).
- Linux (for some OS versions recompilation of ARCUS2.1 CAP module may be required).

1_1 Software component structure

ARCUS 2.1 software application contains the following modules:

EMV application on Ingenico PIN-pad or terminal – (PAP)

Application is a standard POS application with modified scheme of interaction with cash register software. PIN-pad software fully controls operation performance, i.e. it controls interaction with processing, dialog with cashier and client, creation of receipts just like any standard POS-application of a standalone terminal.

Cash register application (library) — CAP (Windows/Linux)

Arcus 2 CAP software provides:

- Transparent data transfer between PIN-pad and processing server (directly via TCP/IP using network interface of cash register PC, unless a dedicated connection of PIN-pad to a merchant LAN is provided);
- interaction (dialogue) between PIN-pad software and cashier;
- Integration with cashier's automated workplace.

Cash register application (library) CAP is a dependent on a manufacturer's of cashier's automated workplace part. Nowadays list of cash register software suppliers varies greatly; they differ in platform (Windows, Linux, UNIX, etc.) used by network interface (TCP/IP) as well as in ability to adapt (interact) to software of other manufacturers. Thus, CAP module is a set of implementations of the same task for various manufacturers of cashier's automated workplace. The most commonly used supply variants of cash register software are CAP – Windows or Linux application that interacts with PIN-pad via RS232 and interface, with processing - via TCP/IP.

1_1_1 TLS Encryption in “IP via Cash Register” Mode

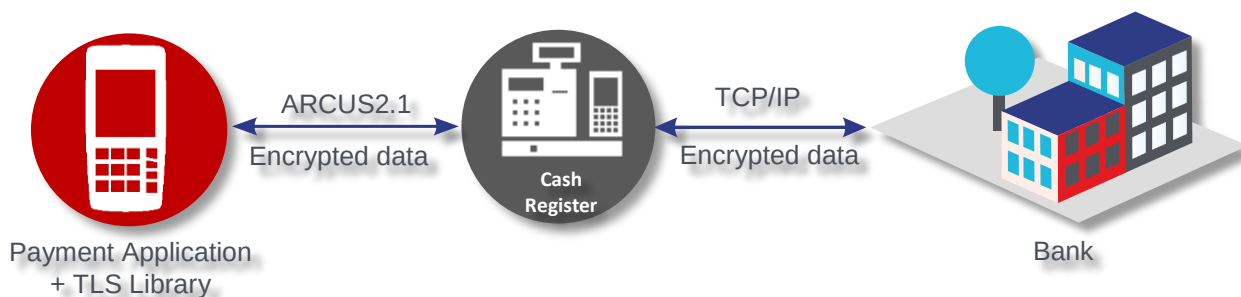


Fig. 1. TLS encryption in “IP via Cash Register” mode

Algorithm:

1. Payment terminal encrypts transaction data using TLS library and sends it via ARCUS2.1 protocol to Cash Register.
2. Cash Register receives data and sends it via TCP/IP to the bank host.
3. Bank host decrypts received data, performs operation, forms reply, encrypts it and sends back to Cash Register.
4. Cash register receives response and sends it to terminal via ARCUS2.1 protocol.
5. Terminal receives response from Cash Register, encrypts it using TLS library and completes the operation.

1_2 Cash Register Mode

Payment terminal can operate in two modes:

- Standalone Mode – terminal is connected to bank host via communication channel (for example Ethernet, GPRS), cashier initiates transaction, performs manual transaction data entry, receives transaction approval and enter information about payment to the cash register.
- Cash Register Mode – Terminal is connected to the cash register (for example via USB, Bluetooth) and waits for the transaction initiation and transaction data form the cash register, performs all required actions and sends result of the transaction to the cash register.

Perform following actions to switch terminal to Cash Register Mode:

1. (1) For Telium terminals wait for the message “Press F to switch Cash Register Mode” after terminal loading and press F.



For specified application settings password may be required.

1. (2) For UNICAPT32 press F3 button (up-right button with Menu or square sign) while terminal loading.
2. Cash Register Mode?

- NO – terminal switches to Standalone Mode;
- YES – Terminal switches to Cash Register Mode Settings menu.



In Universal EMV POS of 4.0.2.435 version and above Cash Register Mode Settings are performed in ARCUS menu.
Initial parameter values can be entered with the help of configuration file ARCUS.CFG.
When terminal is switched to Cash Register Mode all parameters that were entered manually via ARCUS menu are reset to values that are entered in ARCUS.CFG or to default values if file is absent.
For ARCUS menu description see. Universal EMV POS – Admin guide.

ARCUS.CFG is text file in CP1251 encoding, contains settings and connection parameters. It must be loaded to the terminal HOST folder (see Universal EMV POS – Setup Guide).



Settings and Parameters may differ due to the Bank Processing System.

1_2_1 ARCUS.CFG for ISO Processing Systems (OpenWay, SmartVista, Tieto, CTL)

/ Table 5. Settings description in ARCUS.CFG file for ISO

Parameter	Description	Parameter value
Interface parameters – #Interface section		
PORT	Connection port	• COM0 – serial port in the terminal • USB – USB port on the terminal • COM_SL – serial port on MagicBox • COMU – USB port on the base • ETH • BLUETOOTH
PORT-SPEED	Data exchange speed	2400/9600/19200/115200
DATA-BITS	Amount of data bits	7/8
PARTITY	Parity control	NONE/EVEN/ODD
STOP-BITS	Stop bits	1/2
ARCUSSERVER -IP	IP for connection to ARCUS Net Server	Value in xxx.xxx.xxx.xxx format
ARCUSSERVER -PORT	Port for connection to ARCUS Net Server	Value in xxxx format
BLUETOOTH-PIN-LENGTH	PIN length for Bluetooth connection	4-8 (in symbols)
USE-ARCUS-ID	Use ARCUS ID	YES/NO
USE-TMSID	Use TMS ID	YES/NO
ARCUS-ID	ARCUS ID value	
Network parameters - #Network section		
IP-ON-CASHREG	IP via Cash Register	YES/NO
STATIC-IP	Terminal net settings for connection to ARCUS NET Server	YES/NO
TERMINAL-IP		
MASK		
GATEWAY		
	If IP via Cash Register is switched to “NO” and ARCUS NET Server is not used terminal network parameters are entered in the Communication Script! DON’T ENTER THESE PARAMETERS!	
Timeout parameters – #Timeout section		
DIALOG-TIMEOUT	Timeout for dialog display	Value is entered in ms
DATA-TRANSMIT-TIMEOUT	Data transmission parameters	Value is entered in ms
CARD-READ-TIMEOUT	Card reading timeout	Value is entered in ms
Dialog parameters – #Dialogs section		

DIALOGS-ON-CASHREG	Dialogs on Cash register screen	YES/NO
Additional Masking of Card Number for receipt printing – #SpecialMask section		
SPECIAL-MASK	Additional masking activation	YES/NO
PAN-START	Number of displayed symbols in the beginning of card number	0-6, but not more then entered in TMS
PAN-END	Number of displayed symbols in the end of card number	0-4, but not more then entered in TMS
Printing parameters – #PrintSettings section		
PRINTER	Select printer for the receipt printing	CASHREG/INTERNAL/BOTH
PRINTER-ENCODING	Printing encoding	DEFAULT/CP866/CP1251
RECEIPT-WIDTH	Receipt width in symbols	24-48
FORMATTING	Allow to use special symbols	DON'T-USE-ESC/USE-ESC
Additional settings – #AdditionalSettings section		
SEND-PAN-SHA1	Card number SHA1 sending	YES/NO

Example of the file

```
#Interface
PORT=COM0
PORT-SPEED=2400
DATA-BITS=8
PARTITY=NONE
STOP-BITS=1
ARCUSERVER -IP=192.168.0.101
ARCUSERVER -PORT=9401
BLUETOOTH-PIN-LENGTH=4
USE-ARCUS-ID=YES
USE-TMSID=YES
ARCUS-ID=123456790

#Network
IP-ON-CASHREG=YES
STATIC-IP=YES
TERMINAL-IP=192.168.0.10
MASK=255.255.255.0
GATEWAY=192.168.0.1

#Timeouts
DIALOG-TIMEOUT=12000
DATA-TRANSMIT-TIMEOUT=1000
CARD-READ-TIMEOUT=6000

#Dialogs
DIALOGS-ON-CASHREG=YES

#SpecialMask
SPECIAL-MASK=YES
PAN-START=4
PAN-END=4

#PrintSettings
PRINTER=CASHREG
PRINTER-ENCODING=DEFAULT
RECEIPT-WIDTH=24
FORMATTING=DON'T-USE-ESC

#AdditionalSettings
SEND-PAN-SHA1=YES
```

Default values

```
PORT=USB
PORT-SPEED=115200
DATA-BITS=8
PARTITY=NONE
STOP-BITS=1
IP-ON-CASHREG=YES (for iUN and TE_02 - NO)
DIALOGS-ON-CASHREG=NO (for iUN and not SV_05 - YES)
DIALOG-TIMEOUT=12000
DATA-TRANSMIT-TIMEOUT=1000
CARD-READ-TIMEOUT=6000
PRINTER=CASHREG
PRINTER-ENCODING=DEFAULT
FORMATTING=DON'T-USE-ESC
SPECIAL-MASK=NO
SEND-PAN-SHA1=NO
```

1_2_2 ARCUS.CFG for Unipay Processing Systems (CompassPlus, SPDH)

/ Table 6. Settings description in ARCUS.CFG file for UNIPAY

Parameter	Description	Parameter value
Interface parameters – #Interface section		
PORT	Connection port	• COM0 – serial port in the terminal • USB – USB port on the terminal • COM_SL – serial port on MagicBox • COMU – USB port on the base • ETH • BLUETOOTH
PORT-SPEED	Data exchange speed	2400/9600/19200/115200
ARCUSSERVER -IP	IP for connection to ARCUS Net Server	Value in xxx.xxx.xxx.xxx format
ARCUSSERVER – PORT	Port for connection to ARCUS Net Server	Value in xxxx format
BLUETOOTH-PIN-LENGTH	PIN length for Bluetooth connection	4-8
Network parameters - #Network section		
IP-ON-CASHREG	IP via Cash Register	YES/NO
STATIC-IP	Terminal net settings for connection to ARCUS NET Server	YES/NO
TERMINAL-IP		
MASK		
GATEWAY		
	If IP via Cash Register is switched to “NO” and ARCUS NET Server is not used terminal network parameters are entered in the Communication Script! DON’T ENTER THESE PARAMETERS!	
Dialog parameters – #Dialogs section		
DIALOGS-ON-CASHREG	Dialogs on Cash register screen	YES/NO
Additional Masking of Card Number for receipt printing – #SpecialMask section		
SPECIAL-MASK	Additional masking activation	YES/NO
PAN-START	Number of displayed symbols in the beginning of card number	0-6, but not more then entered in TMS

PAN-END	Number of displayed symbols in the end of card number	0-4, but not more then entered in TMS
Printing parameters – #PrintSettings section		
PRINTER	Select printer for the receipt printing	CASHREG/INTERNAL
RECEIPT-WIDTH	Receipt width in symbols	24-48
FORMATTING	Allow to use special symbpls	DON'T-USE-ESC/USE-ESC
Additional settings –#AdditionalSettings section		
AUTH-CODE-FROM-CASHREGISTER	Get authorization code from Cash Register	YES/NO
PRINT-JOURNAL	Settings for operation log printing	YES/NO/MISMATCH

Example of the file

```
#Interface
PORT=COM0
PORT-SPEED=19200
ARCUSSEVER -IP=192.168.0.101
ARCUSSEVER -PORT=9401
BLUETOOTH-PIN-LENGTH=4

#Network
IP-ON-CASHREG=YES
STATIC-IP=NO
TERMINAL-IP=192.168.0.10
MASK=255.255.255.0
GATEWAY=192.168.0.1

#Dialogs
DIALOGS-ON-CASHREG=YES

#SpecialMask
SPECIAL-MASK=YES
PAN-START=6
PAN-END=4

#PrintSettings
PRINTER=INTERNAL
RECEIPT-WIDTH=24
FORMATTING= USE-ESC

#AdditionalSettings
AUTH-CODE-FROM-CASHREGISTER=YES
PRINT-JOURNAL= YES
```

Default values

```
PORT=USB (If COM0 is installed on the terminal default value is COM0 (for UC32 - COM1))
PORT-SPEED=19200
RECEIPT-WIDTH=24
IP-ON-CASHREG=YES
DIALOGS-ON-CASHREG=YES
PRINTER=CASHREG
AUTH-CODE-FROM-CASHREGISTER=NO
SPECIAL-MASK=NO
PRINT-JOURNAL= NO
```

1_3 Integration

2. When the payment transaction is performed, Cash Register software transfers operation parameters:

- Mandatory parameters:

- operation code;
- currency code;
- operation amount in a minimum currency units;



In SW versions released after November 2014 r., transaction will be finished if terminal don't get operation amount.

- Optional parameters:

- authorization code [preauthorization confirmation];
- RRN [preauthorization confirmation].

If some information is missing, there will be request to input these data on the terminal.

3. Function call:

- a) Create – function returns object for further action.
- b) Set – set required value: amount, currency code etc.
- c) Run – start operation on POS terminal.
- d) Get – get data when operation on the POS terminal is over.

4. During operation process via bank card PIN_pad SW may require additional data entry:

- CVC2/CVV2 code
- PIN-code
- Authorization code
- RRN

5. Other dialogs for entry of various parameters can be displayed.



Dialogs can be displayed both on PIN-pad or Cash Register's screen (depending on settings).

All text data is sent in encoding that depends on the operating system: WIN1251 for Windows, DOS866 for Dos and Linux.

Interaction between cashier or cardholder can be represented by the following dialogs:

- Yes/No questions;
- Selection dialogs;
- Request for text or numeric data entry;
- Status and error messages.

6. The returned value of Run() function is a response code. If operation is successfully performed, the returned value is «000» all other values mean that operation failed.



Response Codes depends on Processing.
ARCUS2 also uses internal response codes:

- 992=OPERATION IS INTERRUPTED BY THE CLIENT
 - Client or Cashier refused to perform operation when dialogs were displayed on the terminal;
 - Chip card «refused» to perform financial operation (ex. AAC response code);
 - In ARCUS parameters unavailable parameters was received from the cash register (ex. Unsupported Currency Code)
- 998=CONNECTION ERROR, CONTACT YOUR BANK – IP connection attempt:
 - error;
 - timeout;
 - connection break etc.
- 999=NO PIN PAD CONNECTION – enable to send data via RS232/USB between Cash Register and ARCUS2:
 - Broken wires or slots
 - PIN Pad power supply
 - USB PIN Pad switched to energy saving mode (depends on power supply parameters in computer BIOS, USB cable between PIN Pad and Cash Registers)

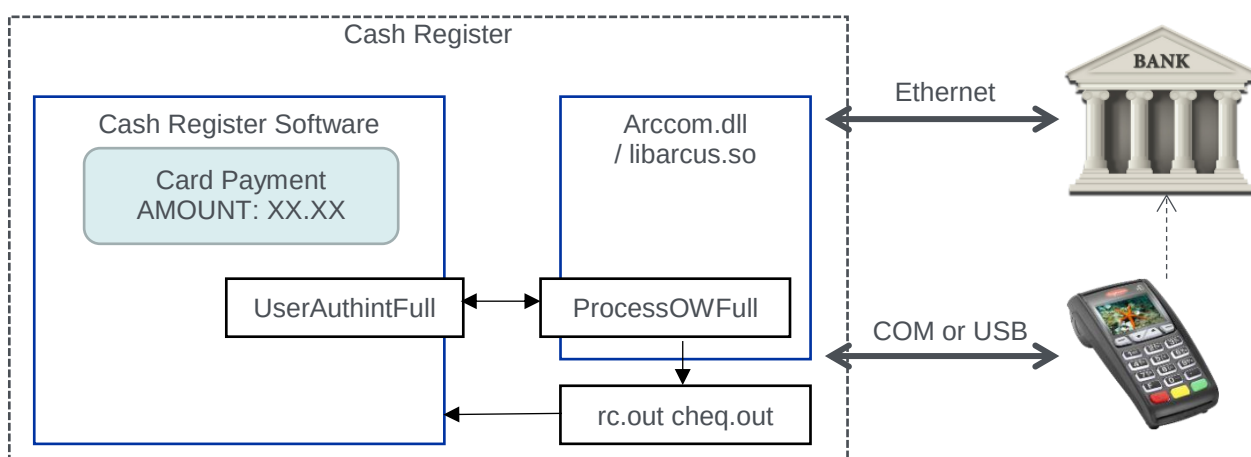


Fig. 2. Interaction of Cash Register Application and Arcus2.1 via exported ProcessOWFull function

3. As a result ARCUS 2.1 module creates 2 or 3 text files:

- Response code file (for example RESULT_FILE of rc.out) contains 3-digit value of bank response code. If the operation is successfully performed, the returned value is 000. All other values mean that operation is failed.
- Receipt file (for example cheq.out) contains operation data in specified for OS encoding for printing. Receipt format and content for bank transaction is specified by bank specialists during parameterization of pin-pad's software. Receipt file contains print symbols only; it does not have control symbols or ESC-sequences.
- Data file [option] (file name for Windows is output.dat, for Linux output.out) with data lines for operation in specified for OS encoding of code page (text file).

Example:

After successful operation receipt will be saved to the file.

```

=====Receipt= example=====
OJS "VTB Bank"
Pr. M. Govorova 52
VTB Bank
Terminal      R0000009
16/07/10 13:29 Cashier:0

PRODUCT PURCHASE
0.02 RUR
Response code:      1
Authorization code 404365 B
Card 4*****6016
VISA      01/11
CARD TEST
Receipt #114      1.1.4
I confirm operation:

_____  

(Client's signature)
=====end of example=====

```

Fig. 3. Bank receipt

In case of hardware or software failure error code is added to the response code file, and service receipt with error code and description is added to the receipt-file.

```

=====example=====
ERROR
Error code:999 (999)
No response code
===== end of example =====

=====example=====
999
===== end of example =====

```

Fig. 4. Error receipt

1_3_1 Format and contents of output.out file

000	- response code (NNN 000 - 999)
447520*****7014	- PAN
R0000009	- Terminal ID (Terminal ID in payment system)
738593	- Authorization code (for offline operations - last 6 symbols of RRN)
VISA	- Card type
APPROVED	- Text message
500	- Transaction amount (if specified in command line) 5.00 RUR
12	- Receipt number in the terminal
4816D0C8FF699AE1B7638672E880D7943DC95117598971170C1C9D921F9AF09C	- card data cryptogram.

As a result of transaction Terminal creates receipt that can be printed on Terminal internal printer or on Cash Register printer.

Receipt format that is printed on Terminal printer depends on Terminal SW configuration, defined by Bank specialist.

If receipt must be printed on Cash register, data for printing can be sent by one of the following ways:

- Print data from cheq.out file and display data from response file rc.out to the screen;
- Cash Register SW forms receipt for printing based on data from Get function.

Mandatory data printed on a receipt for card transaction are:

- Authorization code;
- Operation currency;
- Place for Client's signature;
- Name, country and city (address) of the organization;
- Response code and encoding of rejection reason;
- Card type and expiration date;
- Acquiring bank name;
- EMV data (if transaction is performed via EMV card).

If cash register application uses file-receipt, then Bank specialists must keep in mind requirements of international payment system regarding receipt contents for card transactions during PIN-pad parameterization.

If cash register application uses data from ARCUS 2.1 module or data file, fulfillment of international payment system requirements regarding receipt contents for card transactions is regulated by cash register application.

Receipt printing is not required for response codes 99x (x = 1...9) — these are the codes of technical failures or rejections sent from ARCUS 2.1 that are not response codes received from bank processing.

1_4 ARCUS 2.1 test

1_4_1 Windows / Linux

Standard integration with Cash Register is performed via Arccom.dll / libarcus.so library.

To test work of ARCUS 2.1 module use cash register emulator CommandLineTool.

Parameters for CommandLineTool:

- **/o [Cash register operation code]** – operation code (Number before the sign = line in file ops.ini).
- **/c [Currency code]** – Operation currency code.
- **/a [Amount]** – Operation amount in minimal currency units without separator.
- **/i [Terminal ID]** – Terminal ID up to 9 symbols (defined by TMS).



If this key contains data with size of 1 byte, "/i" is number of department for UNIPAY SW

- **/v [Authorization code]** – Authorization code , 6-9 symbols.
- **/r [Reference]** — RRN, up to 12 symbols.
- **/e [Card data]** – Encoded card data, 64 symbols
- **/t [Track2]** – The second card's track.
- **/b[TraceID]** – preauthorization ID (available since January 2015 for TE_02 product only)
- **/h[Original Amount]** – Total authorized amount (available since January 2015 for TE_02 product only)
- **/f[Original Date/Time]** – date and time of the original Preauthorization (available since January 2015 for TE_02 product only)
- **/mcc** – Merchant Category Code (available since July 2017);
- **--origin-oper-type=N** – type of original operation for Cancel out of Batch (OW, Fuel+);
- **--repeat** – number of calls of executable function (without library loading);
- **--dialog-direct** – switch on Dialog Direct (must be configured in TMS).



Mandatory parameters are: Operation Code, Currency and Amount.
Any other parameters can be skipped. If supplied information is not enough for operation, pin-pad requests to enter parameters one more time (a corresponding dialog will be displayed on the screen).



Working of unattended devices (iUN250, i9500 и др.) is performed without client dialogs and manual input, so in this case all required parameters must be transmitted (for example for PAYMENT operation currency code and amount are obligatory).

Example:

```
CommandLineTool.exe /o1 /c643 /a12345
```

- Purchase for 123,45 rub. Currency code 643. Card is entered via PIN Pad.



Card number and expiration date can be entered manually from PIN-pad keyboard. To use this function press YELLOW during display of initial screen.

In case when the Track is received from the Cash Register software the following format is used:

```
CommandLineTool.exe /o1 /c643 /a12345 /t;1000023000000034=06062011021300000641?
```

- Purchase for 123,45 rub (currency code 643) by the card that was presented to the Cash Register Software, track 2;1000023000000034=06062011021300000641?.

Track2 data is opened by ';' symbol and closed with '?' symbol.

In case of manual input track2 is sent in the following format:

```
M[Card Number]=[Expiration date in YYYY format]?  
CommandLineTool.exe /o1 /c643 /a12345 /tm1000023000000034=1106?
```

- Purchase for 123,45 rub (643 currency code) by the card with 1000023000000034 number, that was entered manually. Expiration date June 2011.

To transmit authorization code, use parameter /v with 6 symbols following the parameter.

For example:

```
CommandLineTool.exe /o4 /v123456
```



Arcus2.1 is excluded from the certification scope of Payment Application Data Security Standards (PA DSS), as it doesn't stores and transmits unencrypted data of payment cards when it is configured correctly. If special CAP SW versions are used and track data is entered via Cash Register it is mandatory to certificate Cash Register SW.



For HRS mode use /e key to encrypt card data:

CommandLineTool.exe /o1 /a2000

/e3AB5E25D6091053EAB71AFAE303B5327F3D713043819649F3D20D11F78A73811

To transmit card data from cash register application, special ARCUS II setting is required

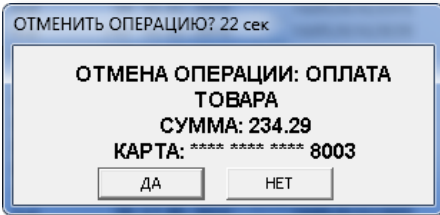
– Add. settings – use enc. param. (use encryption parameters).

Data is encrypted by the 3DES CBC encryption. Encryption key must be entered in open view in Administrator menu – PIN Pad – ARCUS Key.

1_5 Initialization file Cashreg.ini

Variable	Description
PORT parameters	
PORT	Serial port name (e.g., system name of serial port in Windows).  It is recommended to use 1-9 COM ports for ARCUS 2 CAP. If it is necessary to use port 10-99 use PORT=\\./COM[Port_number] or in Windows device manager: Podt parameters - additional - COM port number.
PORT_IP	IP for connection to ArcusIPService /ArcusNETServer /IngenicoNETServer
ARCUS_ID	Identifier for ArcusNetServer or IngenicoNETServer, for authType="arcusid" mode
PORT_IP	IP порт для подключения к ArcusIPService /ArcusNETServer /IngenicoNETServer
SPEED	Data exchange speed. Available values 2400, 4800, 9600, 19200, 38400, 57600, 115200.  Recommended values for RS232 – 9600, for USB – 115200.
BYTE	Amount of data bits.
PARITY	Parity (N- no parity control, E – evenness control, # O – oddness control).
STOP	Stop bits.
TIMEOUT	timeout for getting one message from pin-pad (in ms). Parameter can be used in Cash Register mode – Cash Register Dialogs – YES.
PPAD_IDLE_TO	Inactivity timeout for situations: Card is left in Chip reader Session with chip card is over Answers to dialogs.  If dialog on the Cash register was initialized by the terminal "Dialog timeout" parameter sets maximum time value for dialog to be displayed (Cash register mode). So this parameter must be less or equal to Dialog timeout.
File parameters	
OPERATION_INI_FILE	Operation code conversations file name. Default value - ops.ini.
RC_RESOLVE_FILE	Name of the file for response code decoding. Default value - rc_res.ini.  For MultiHost solution: RC_RESOLVE_FILE_LT - name of the file for response code decoding for loyalty cards RC_RESOLVE_FILE_PT - name of the file for response code decoding for fuel cards.
DEFAULT_RC_STRING	File for response code default decoding if the response code couldn't be found in RC_RESOLVE_FILE.
USEORIGINALRC	Activates functionality of using response code before the conversation. If it is turned off, converted value is used (RC_CONVERT_FILE and rc_conv.ini).

Variable	Description
RC_CONVERT_FILE	Configuration file for response code converting. Default value - rc_conv.ini.
CHEQ_FILE	File for receipt output. Default value - cheq.out.
RESULT_FILE	File for response code output. Default value - rc.out.
CANCEL_CH_FILE	File for receipt output in case of automatic reversal. Default value - auto_can.out.
CANCEL_RC_FILE	File for response code output in case of automatic reversal. Default value - can_rc.out.
DATETIME_FORMAT	Формат вывода даты/время в файл лога, может принимать значение: • 1 - YYYYMMDD HHMMSS.SSS, • 2 - только секунды (как в 2.0. было), • по умолчанию - YYYY/MM/DD HH:MM:SS
Log file parameters (available in versions released after 07.08.2015)	
NDAYS	Period for log file storing (in days). Default value – 130.
NFILES	maximum number of log files in Arcus2/logs folder. Default value - 0 (not limited).
NSIZE	maximum size of all log files in Arcus2/logs folder, in MB. Default value - 0 (not limited).
LOG_FOLDER_NAME	Name of folder with log files. Default value – Logs.  Log files are stored in Arcus2/logs up to the moment when the value of at least one parameter is less then apesified values. Parameter values are checked whrn ARCUS library is called. Log-files folder must be entered in «working directory» settings and may differ from default folder «C:\Arcus2».
LOG_FOLDER_MODE (L)	битовая маска установки мод. доступа к папке в восьмиричной системе, по умолчанию "770". (только для Arcus2.1 Linux)
DISABLE_NAK	отключение вывода в лог файл команд NAK
DEBUG	в лог файл будут выводиться все "распознанные" теги, в виде <название поля>: <строка данных> [<бинарное представление>]
Security parameters	
NO_PA_DSS	If the variable is defined ARCUS2 process track2 of the card that is sent via Cash Register application, if it is not defined track 2 is ignored, the card must be read via PIN Pad.
NOTIMESYNC	Activates time synchronization with cash register function.
USEAPPLOG	Activates record of the trace information into the log files.
USEPPADTRACE	Activates record of the trace information into the log files.

Variable	Description
USEJL	variable enables usage of few legal entities (virtual terminals) if it is available of terminal application.  Algorithm: Create jl.txt file in the same folder where “ArcCom.dll”. Specify virtual terminal TID (for ISO projects) or virtual terminal sequence number (for Compass+/SPDH projects) legal entity number Follow one of the following logic: Create folders with ArcCom.dll modules and jl.txt files. External Cash Register SW will interact with various modules. External Cash Register SW rewrites jl.txt before calling ArcCom.dll.
Last operation Cancel confirmation (in Cash Reg mode only)	
DEF_OPDET_RESPONSE	Functionality of Last operation Cancel confirmation - If flag is not set Confirmation request will be displayed on Cash Register screen with YES/NO answers; If variable is set to «YES» or «NO», dialog will not be displayed, parameter value will be used as answer. 
Additional parameters	
INFO_DIALOG	switches on payoffINFO dialogs (for InfoKiosks only). For proper work in TMS Additional Settings «Info-Diologs» flag must be switched on
USE_DLG_DIRECT	Activates Dialog Direct functionality (available for versions released after November 2015) – switches display of the message on the terminal screen from the cash register with possibility to receive replies (pressed key value). Corresponding parameters must be entered in TMS.
OPCHARSET	Кодировка сообщений на экране кассы (CP1251 для Windows, UTF-8 для Linux).
OUTPUT_FILE	С июля 2016 года доступна версия ARCUS2 CAP с поддержкой передачи специфичных данных, характерных для топливных карт и карт лояльности. Для активации функционала в файле cashreg.ini задайте параметр OUTPUT_FORMAT_V=1. В результате при отправке данных о платеже будет сформирован результирующий файл, указанный в параметре OUTPUT_FILE в cashreg.ini, в следующем формате: [параметр]=[значение]
NO_HRS	Отключает использование HRS (для интеграции со структурой без данного поля)
USE_UNIPAY	Receipt Number tag index for Universal EMV POS (Unipay) SW
AUTH_CODE_FULL	Sign for code authorization type definition (TPTP_09): • 1 - (default value) authorization code will be appended by Space sign and Letter (if \x1F\x2D tag contains it); • 0 – authorization code is not changed.
NODIALOGS	Swotch on/off cashregistre dialogs.

Variable	Description
AUTODETECT_OPS	(NW_TPTP_15) Activation of operation selection by specialcriteria.  Set values for OPS_CONV_INI_FILE (correspondence of TID and ops.ini) and ID_CONV_INI_FILE (correspondance of additional and real TID) parameters.

1_6 ARCUS 2 CAP Settings

1_6_1 Ops.ini (ISO 8583)

File with operation code that is sent from the cash register and PIN-pad response codes.

Op code	Op class	Op id	Op Name
1=	1,	1,	PURCHASE

Op code – Cash Register Operation Code.

Op class – Operation Class (1 – financial, 2 - technical, 3 - administrative).

Op id – operation ID.

	In ARCUS2 released after July 2014 command class “8” is added. This class is developed to work with Dynamic Menu.
---	--

/ Table 7. List of operations for Universal EMV POS (NewWay)

Class	Code	Name
BASIC FINANCIAL OPERATIONS		
1	1	PURCHASE
1	2	PURCHASE DUAL
1	3	PRE-AUTHORIZATION
1	4	PRE-AUTHORIZATION COMPLETION
1	5	CANCEL
1	6	CANCEL ADVICE (TIETO)
1	8	MINI STATEMENT
1	10	UNIVERSAL CANCEL ADVICE
1	11	REFUND
1	15	CREDIT VOUCHER (OW)
BANK, CASH ADVANCE OFFICE		
1	7	BALANCE REQUEST (including Deposit mashines functionality SV_05)
1	12	CASH WITHDRAWALS (including Deposit mashines functionality SV_05)
1	13	CASH WITHDRAWALS DUAL
1	14	CARD ACCOUNT CREDIT
1	47	BALANCE ACC
PAYMENTS / TRANSFER		

Class	Code	Name
1	16	UTILITY PAYMENT
1	17	BILLING PAYMENT SMS
1	18	BILLING PAYMENT DUAL
1	40	CARD2CARD TRANSFER
1	41	CASH2CARD TRANSFER
1	109	Alipay Purchase (TPTP_NW)
1	110	Alipay Refund (TPTP_NW)
TECHNICAL OPERATIONS		
2	3	REVERSAL (TECHNICAL)
2	4	MAGNETIC CARD READING
2	6	HOST CONNECTION TEST – check possibility to connect to the host, without host authorization, sending commands or receipt printing
2	18	CONNECTION TEST EXCHANGE (REQUEST - RESPONSE) test connection with host (authorization, sending echo request), with receipt printing
2	21	TERMINAL INFO
REPORTS / RESULTS		
2	0	FULL TRANSACTION LOG
2	1	RECONCILIATION
2	10	SHORT REPORT
2	19	DELAYED REVERSAL SENDING
RECEIPT COPY PRINTING		
2	27	PRINT COPY OF LAST RECEIPT
2	38	PRINT COPY OF RECEIPT BY TRANSACTION_ID (NW_TPTP)
MENU		
2	2	CASHIER'S MENU
2	9	SENIOR CASHIER PASSWORD
COMMENTS FOR NEWWAY sw WITH SPDH / COMPASS+ PROCESSING SYSTEMS		
2	0	REPORT (PACKET)
2	1	RECONCILIATION (PACKET)
2	22	RECONCILIATION (DAY)
2	23	REPORT (SHIFT)
2	24	REPORT (DAY)
2	25	RECONCILIATION (SHIFT)
ADMINISTRATIVE OPERATIONS		
3	1	TIME SYNCHRONIZATION
3	2	TIME SYNCHRONIZATION
3	4	TMS SESSION
3	5	ADMINISTRATOR'S MENU
3	8	TMS SESSION BY DEFAULT SCRIPT

Class	Code	Name
3	9	TERMINAL ACTIVATION
CASH OPERATIONS (“Cash” card type must be entered in TMS)		
4	89	CASH PAYMENT
4	90	CASH REFUND
OPERATIONS WITH MIFARE CARDS		
9	40	MIFARE – read UID and send it to the Cash Register



Dynamic menu functionality is available in Universal EMV POS – use “8” command class in Cash register mode. Value for Op id – value of TMS – Terminal groups – Dynamic Menu – Cash Register (from 1 to 20).

1_6_2 ops.ini for Compass+ and SPDH

/ Table 8. List of operations for Universal EMV POS (Unipay)

Class	Code	Name
BASIC FINANCIAL OPERATIONS		
0	128	PURCHASE
0	130	REFUND
0	133	PRE-AUTHORIZATION
0	134	PRE-AUTHORIZATION COMPLETION
0	224	CANCEL LAST (TECHNICAL REVERSAL)
0	225	CANCEL BY RECEIPT (UNIVERSAL)
RECONCILIATION, REPORTS		
0	160	CLOSE BATCH
0	161	CLOSE SHIFT
0	162	CLOSE DAY
0	179	LOCAL RESULTS
0	180	SHORT LOG
0	181	LOG PRINTING
0	186	TRANSACTION LOG PRINTING
0	187	FULL LOG
BANK, CASH ADVANCE OFFICE		
0	129	CASHLESS TRANSFER
0	131	ACCEPT CASH
0	132	CASH WITHDRAWALS
0	136	BALANCE
0	154	CHECK CARD
0	231	PIN CHANGE

Class	Code	Name
CORRECTIONS		
0	144	PURCHASE CORRECTION
0	147	CASH ACCEPTANCE CORRECTION
0	148	CASH WITHDRAWAL CORRECTION
PAYMENTS		
0	138	PAYMENTS
0	142	CARD TO CARD TRANSFER
0	143	CARD CHARGE BY CASH
SERVICE		
0	176	COPY OF THE LAST RECEIPT
0	177	COPY OF THE RECEIPT BY NUMBER
0	182	EMV DETAILS
SPECIAL SW VERSIONS		
0	220	CANCEL BY AMOUNT (SPDH_03) amount – obligatory data
		 This operation is available for Cash Register Mode, Parameter “Card for cancel” must be switched on.
ADMINISTRATIVE OPERATIONS		
0	50	CASHIER MENU
0	53	ADMINISTRATIVE MENU
0	166	CHECK CONNECTION WITH HOST
0	167	LOAD PARAMETERS FROM THE HOST
0	204	TMS SESSION

1_6_3 Rc_conv.ini

In Rc_conv.ini file the converting table of the PIN Pad reply codes into the codes that are passed to the Cash register.

For example if it is required to process response code 001 as 005 and 123 as 843 just add these two lines in the following format:

[Pin Pad response code]=[Cash Register response code]

Example:

001=005
123=843

If PIN Pad sends 001 then response code 005 is used. If response code is 123 then 843.

If there is no response code in file rc_conv.ini then unmodified value from PIN Pad is used.



Incorrect values in rc_conv.ini can cause problems with response codes processing by Cash register and subsequent financial issues. Developers and administrators of Cash register solution must apply measures for protection rc_conv.ini file against unauthorized change

1_6_4 Rc_res.ini

The Rc_res.ini file processing reply codes are entered.

The table contains data about the host reply codes and corresponding message in the screen. Data must be in the following format:

[response code]=[message]

000=	Approved
001=	Approved
002=	Approved
003=	Approved
004=	Approved
005=	Approved
006=	Approved
007=	Approved
008=	Approved
009=	Approved
050=	Rejection
051=	Card is not valid
052=	PIN limit is exceeded
053=	Wrong Administrative card
054=	No secret block
055=	Invalid operation



Detailed information about the contents of initialization file is available in the development or project implementation specification.



If you work with the Linux OS use USEORIGINALRC flag in cashreg.ini file (p. 1.3.3 Cashreg.ini file).

1_7 Messages and Menu output on terminal screen (Dialog Direct)

For integration with Cash Register SW for Windows OS via ArcCom.dll it is possible to display messages and Menu on Terminal screen. To activate functionality USE_DLG_DIRECT parameter must be set in cashreg.ini file.

Return values:

- 0 : OK
- 1: NOK
- 3: ERROR_NO_ANSWER
- 9: ERROR_NO_SESSION

Start DialogDirect session

int dlgSessionStart(void);

Stop Dialog Direct session

int dlgSessionEnd(void);

Display string

int dlgWriteLine(int dest, int rowNum, int cellNum, char *line);

- dest – command for terminal (1), command for PIN-pad (0)
- rowNum – row number

- cellNum – column number
- line – line pointer

Read string

int dlgReadLine(int dest, int rowNum, int cellNum, int isMask, int timeout, int length, char *buffer);

- dest – command for terminal (1), command for PIN-pad (0)
- rowNum – row number
- cellNum – column number
- line – line pointer
- isMask – activate/deactivate input string masking
- timeout – timeout
- length - maximum input string length
- buffer – pointer to memory sell where result will be saved

Read key

int dlgReadKey(int dest, int timeout, char *keyCode);

- dest – command for terminal (1), command for PIN-pad (0)
- timeout – timeout
- buffer – pointer to memory sell where result will be saved

Clear screen

int dlgClrScreen(int dest);

Display menu

int dlgDisplayMenu(int dest, const char *caption, const char *const items[], size_t items_size, int *selection);

- dest – command for terminal (1)
- caption – pointer to string with menu caption
- items – array of pointers with pointers to menu items captions
- items_size – amount of pointers in items argument
- selection – pointer to memory sell where number of selected menu item will be saved

1_8 External dialogs

Functionality of dialogs on the Cash Register is available since July 2016. Dialog can be configured by arcus.h file:

```
typedef struct
{
    size_t size;
    void * userdata;
    cbMenu cbfMenu;
    cbYesNoDialog cbfYesNoDialog;
    cbWarningDialog cbfWarningDialog;
    cbInfo cbfInfo;
    cbStatus cbfStatus;
    cbStringEdit cbfStringEdit;
}
DialogsCBs;
typedef struct
{
    size_t size;
    cbI344Window cbfI344OpenWindow;
    cbI344Window cbfI344ClearWindow;
    cbI344DisplayLine cbfI344DisplayLine;
    cbI344ClearLine cbfI344ClearLine;
    cbI344KeyReq cbfI344KeyReq;
} DialogsI344CBs;
```



userdata is not used, always NULLPTR.

How to set pointer to the function:

DialogsCBS callbacks;

```
...
    if (use_console_dialogs != 0)
    {
        callbacks.userdata = nullptr;
        callbacks.size = sizeof(callbacks);
        callbacks.cbMenu = callback_menu;
        callbacks.cbYesNoDialog = callback_yesno;
        callbacks.cbWarningDialog = callback_warning;
        callbacks.cbInfo = callback_info;
        callbacks.cbStatus = callback_status;
        callbacks.cbStringEdit = callback_stringedit;
        ITPosSet(itpos, "DialogsCBS", std::to_string((long) &callbacks).c_str(), -1);
    }
...
```

Test for Windows version:

To check work use in ArcCom.dll key /console - Use callback dialog functions, for example:
CommandLineTool /o1 /a101 /console

Test for Linux version:

./CommandLineTool /o1 /a101 /c643 /console

1_9 Maintenance

1_9_1 Integration of ARCUS2.1 CAP with cash register software

As a result of re-mastering cash register software, Arcus 2.1 CAP becomes an additional module of cash register software that expands functioning possibilities of cash register software at performing transaction with cards in merchant organizations and bank accounts in bank branches.

1_9_2 Virtual Keyboard

For Cash Registers with OS Windows but without keyboard, for WAY4 Card Management System, CTL, SmartVista (BPC), Tieto, Cortex, TITP processing virtual keyboard is available since 2014 July.



Fig. 5. Virtual Keyboard

The following data can be entered:

- Numeric input and editing;
- IP- address input and editing;
- HEX input and editing.

1_9_3 Methods and Frequency of Status Monitoring

Monitoring status of Arcus 2.1 CAP is performed in the scope of complex technical support for acquiring bank in merchant organization by technical experts of bank cards acquiring services in processing part and by company that developed Arcus 2.1 CAP according to the statistic failure data received from bank processing and IT-experts from merchant organization.

In case cash register software manufacturer informs about update release of the existing version or completely new version software release, test control of Arcus 2.1 CAP configuration compatibility is initiated and a range of test series is performed on a stand of cash register manufacturer or in merchant company.

In case existing version of Arcus 2.1 CAP is updated or new version of Arcus 2.1 CAP is released, test control of Arcus 2.1 CAP module configuration and interface compatibility is initiated and a range of test series is run on a stand of cash register manufacturer or in merchant company.

1_9_4 Troubleshooting

In case of some crucial situations (overwriting or deleting critical files from Arcus 2.1 CAP package, OS or software failure), merchant technical experts and acquiring technical support take recovery measures, configure Arcus 2.1 CAP module, and perform test running session in merchant organization.

In case of failures (financial or administrative operation failure), merchant technical experts and acquiring technical support analyze the reason of failure. According to data from log files in Arcus 2.1 CAP complete report on the detected problem and its reason is formed and sent to software technical support and development department of Arcus 2.1 CAP software developer to fix detected bugs in Arcus 2.1 CAP and release update. Then debugging, testing and start-up testing in testing services of Arcus 2 CAP software developer and merchant organization are performed.

1_9_5 Log File Description (Fragments)

- ArcCom.dll build date

Version: 2.1.0.1
Build date:
Tue Dec 23 17:56:01 2014

1_9_6 Refund without card

Refund without card is available for product SPDH_03 only. It implements Refund, Cancel and Preauthorization Completion without card presenting by authorization code in cash register mode.

For detailed information, please contact Ingenico Support Team.

2_Integration via RS232 or USB

Terminal can be connected to the Cash Register via RS232 or USB.

If the connection to the Bank Processing must be performed via Cash Register equipment Network adapter must be plugged into it.

If the connection to the processing is performed via terminal equipment, it must be connected to the specified channel.

Cash Register OS must be Windows 98, 2000 or XP. ARCUS2 software must be installed.

If the terminal is connected to the Cash Register via USB virtual COM Port driver must be installed.

Terminal software must correspond to the bank processing type.

Interaction scheme

When the transaction is starting the following data is sent to the terminal:

- Operation Code;
- Currency Code;
- Operation amount with smallest currency unites.

These data are stored in SET method. Cash Register SW calls Run function with operation number as an input parameter. Function Run blocks Cash Register Application and Terminal application controls all dialogs.

During operation process CVC2/CVV2, and PIN cod input can be required. All text information has WIN1251 encoding.

3_Low-level integration

Cash Register Application can interact with PIN Pad without ARCUS2. In this case in Cash Register Application should implement full support of the ARCUS2 Protocol.

PIN Pad must be configured to work with Cash Register in accordance with the document "ARCUS2 for integration with POS solutions installation Guide, p. 3.2. Cash Register Mode». When the PIN Pad is switched on and software is working it is waiting for the command to start operation via configured communication channel (RS232, USB, TCP ...).

After receiving command PIN Pad switches into active mode and begins processing the requested operation. During execution, the device sends to Cash Register command to open resources, to read or to write, to print and to display dialogs. Received command is processed on the Cash Register and then response command must be sent to PIN Pad.

At the end of each operation PIN Pad sends a command to complete the operation, Cash register application sends reply and close all open resources. The operation is considered as completed.

The result of the implementation is determined by the value of the response code received from the PIN Pad. If the Cash Register application didn't received any response code the result of the operation is considered to be negative (operation failed with unknown error). During the implementation process several response codes can be sent, in this case last received response code is the required value.

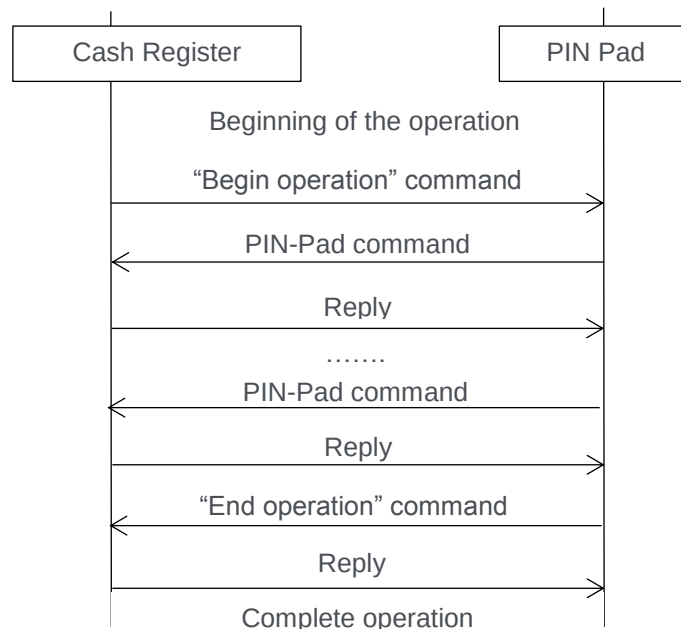


Fig. 6. Interaction scheme

To get description of the low-level communication Protocol request to the technical support service Ingenico.

ANNEX 1

KEYS FOR SET/GET METHODS

Interaction via Set/Get(IN/OUT) methods with the following keys:



Keys are case sensitive.

Key	Description	Direction	MAX data length (ISO8583 recommended)
currency	Currency code ISO	IN	3
amount	Operation amount without separators (in minimal currency unites)	IN	12
result_amount	Amount after operation (in minimal currency unites)	OUT	12
original_amount	Original operation amount without separators (in minimal currency unites)	IN	12
rfu	Track2 ISO	IN	60
pan	Masked PAN	OUT	20
expiry	Expiry Date (YYMM)	OUT	4
aid	EMV AID	OUT	80
card_type	Card type	OUT	80
application_label	EMV ApplicationLabel	OUT	80
date	Date (Processing) (DDMM)	OUT	4
time	Time (Processing) (hhmmss)	OUT	6
terminal_id	Terminal ID TID (Processing)	IN, OUT	8
terminal_serial_number	Terminal serial number	OUT	24
ARCUS_ID	Arcus2 ID (Cashregister)	IN	12
response_code	Response code – operation result	OUT	3
auth_code	Authorization code	IN, OUT	8
rrn	RRN	IN, OUT	12
tvr	EMV TVR	OUT	80
received_text_message	Text message	OUT	80
text_message	Response code description	OUT	80
cardholder_name	Cardholder name	OUT	27
enc_data \ result_enc_data	Card encrypted data for HRS functionality	IN / OUT	64
trace_id	Preauthorization ID	IN, OUT	60
original_date_time	Date-time of original operation (YYMMDDhhmmss)	IN, OUT	12
receipt_number	Receipt nubmer	IN, OUT	4
transaction_id	Transaction ID	IN / OUT	18

ANNEX 2

EXAMPLE OF LIBRARY USAGE

Function Prototypes

```
extern "C" ARCCOM_API void* __cdecl CreateITPos();
extern "C" ARCCOM_API void __cdecl DeleteITPos(void *obj);
extern "C" ARCCOM_API int __cdecl ITPosSet(void *obj, const char *key, const char *value, int size = -1);
extern "C" ARCCOM_API int __cdecl ITPosGet(void *obj, const char *key, char *value, int size = -1);
extern "C" ARCCOM_API int __cdecl ITPosRun(void *obj, int cmd);
extern "C" ARCCOM_API int __cdecl ITPosRunCmd(void *obj, const char *cmd_str, const char *value, int size = -1);
extern "C" ARCCOM_API void __cdecl ITPosClear(void *obj);
extern "C" ARCCOM_API void __cdecl Stop(void *obj);
```

- obj – object of interaction with POS terminal;
- key – parameter name transferred as ASCII string (with terminal symbol);
- value – parameter value (IN/OUT), transferred as symbolic array;
- size – data size, that are transferred in parameter, if -1, length is calculated same way as for ASCII string (only for NULL terminated strings);
- cmd – operation code from ops.ini;
- cmd_str. – call special operations.

Function description:

Function	Description
CreateITPos	Creation of new communication object with POS terminal
DeleteITPos	Delete POS terminal communication object
ITPosSet	Set operation parameters
ITPosGet	Get parameters after operation execution
ITPosRun	Run operation
ITPosRunCmd	Run specific operation (for example, TEST_PING)
ITPosClear	Clear parameters before performance of new operation
Stop	Cancel card expectation (Accynchron method)

Example for C++



__LINUX__ define must be added for Linux OS

```
//-----
#include "stdafx.h"
#include <string>
#include <conio.h>

#ifdef __LINUX__
    #include <stdexcept>
    #include <dlfcn.h>
#else
    #include <windows.h>
#endif
//-----
#ifdef __LINUX__
```

```

#ifndef __cdecl
#define __cdecl __attribute__((cdecl))
#endif

#define HINSTANCE void*
#define LoadLibrary(x) dlopen((x), RTLD_LAZY)
#define FreeLibrary(x) dlclose((x))
#define GetProcAddress(x, y) dlsym((x), (y))
#define GetLastError std::to_string(errno).c_str
#endif

typedef void *(__cdecl *ARCUS_CREATE)();
typedef void (__cdecl *ARCUS_DELETE)(void *);
typedef int (__cdecl *ARCUS_SET)(void *, const char *, const char *, int);
typedef int (__cdecl *ARCUS_GET)(void *, const char *, char *, int);
typedef int (__cdecl *ARCUS_RUN)(void *, int);
typedef int (__cdecl *ARCUS_RUN_CMD)(void *, const char *, const char *, int);
typedef int (__cdecl *ARCUS_CLEAR)(void *);

#ifdef __LINUX__
const char dll_name[] = "./libarccom.so";
#else
const char dll_name[] = "Arccom.dll";
#endif
//-----
int main(int argc, char *argv[])
{
    void *pos_obj = NULL;
    ARCUS_DELETE ArcusDelete = NULL;

    try
    {
        /// load extern DLL with require functions
        HINSTANCE dll = LoadLibrary(dll_name);

        if (dll == NULL)
        {
            /// exit if DLL file not loaded
            throw std::runtime_error("DLL not Load");
        }

        /// create all DLL functions
        ArcusDelete = (ARCUS_DELETE) GetProcAddress(dll, "DeleteITPos");

        ARCUS_CREATE ArcusCreate = (ARCUS_CREATE) GetProcAddress(dll, "CreateITPos");
        ARCUS_SET ArcusSet = (ARCUS_SET) GetProcAddress(dll, "ITPosSet");
        ARCUS_GET ArcusGet = (ARCUS_GET) GetProcAddress(dll, "ITPosGet");
        ARCUS_RUN ArcusRun = (ARCUS_RUN) GetProcAddress(dll, "ITPosRun");
        ARCUS_RUN_CMD ArcusRunCmd = (ARCUS_RUN_CMD) GetProcAddress(dll, "ITPosRunCmd");
        ARCUS_CLEAR ArcusClear = (ARCUS_CLEAR) GetProcAddress(dll, "ITPosClear");

        /// check DLL functions
        if (ArcusCreate == NULL) throw std::runtime_error("function ArcusCreate not loaded");
        if (ArcusDelete == NULL) throw std::runtime_error("function ArcusDelete not loaded");
        if (ArcusSet == NULL) throw std::runtime_error("function ArcusSet not loaded");
        if (ArcusGet == NULL) throw std::runtime_error("function ArcusGet not loaded");
        if (ArcusRun == NULL) throw std::runtime_error("function ArcusRun not loaded");
        if (ArcusRunCmd == NULL) throw std::runtime_error("function ArcusRunCmd not loaded");
        if (ArcusClear == NULL) throw std::runtime_error("function ArcusClear not loaded");

        /// initialize object for POS
        pos_obj = ArcusCreate();

        if (pos_obj == NULL)
        {
            /// exit if object not created
            throw std::runtime_error("create object fail");
        }

        /// repeat operation
        while (1)
        {
            if ((argc > 1) && (std::string(argv[1]) == "ping"))
            {
                printf("PING\n");
            }
        }
    }
}

```

```

        /// run PING POS operation without create log files
        ArcusRunCmd(pos_obj, "TEST_PING", "1", -1);
    }
    else
    {
        /// set amount and currency for purchase operation
        if (ArcusSet(pos_obj, "amount", "2000", -1) != 0)            throw
std::runtime_error("set amount fail");
        if (ArcusSet(pos_obj, "currency", "643", -1) != 0)            throw
std::runtime_error("set currency fail");

        - ERROR

        /// start purchase (1), result is response code of this operation: 0 - OK, else

        int result = ArcusRun(pos_obj, 1);
        printf("Command result: %d\n", result);

        if (result != 0)
        {
            /// exit if operation return not 0
            throw std::runtime_error("operation fail");
        }

        /// if send NULL that function return length of this parameter
        int size = ArcusGet(pos_obj, "rrn", NULL, -1);

        if (size <= 0)
        {
            /// exit if object can't return size of RRN buffer
            throw std::runtime_error("get RRN fail");
        }

        /// create buffer for parameter more than 1 byte for terminate character
        char *rrn = new char[size + 1];

        /// for example: try to get RRN of operation
        if (ArcusGet(pos_obj, "rrn", rrn, size + 1) < 0)
        {
            /// exit if object can't return RRN
            delete[] rrn;
            throw std::runtime_error("get RRN fail");
        }

        printf("RRN: %s\n", rrn);

        /// free buffer
        delete[] rrn;

        /// clear all transaction data before start new operation
        ArcusClear(pos_obj);
    }

    printf("Press any key for repeat (ESC - exit) ...\n");
    int c = getch();

    if (c == 0x1B)
    {
        /// exit after press ESC
        break;
    }
}
}
catch (std::exception &e)
{
    printf("ERROR: %s\n", e.what());
}
catch (...)
{
    printf("ERROR: unhandled exception\n");
}

if ((ArcusDelete != NULL) && (pos_obj != NULL))
{
    ArcusDelete(pos_obj);
}

```

```

        return 0;
    }
    //-----

```

Example for C#

```

using System;
using System.Text;
using System.Runtime.InteropServices;
using System.IO;

namespace TestArcus
{
    class ITPos
    {
        [DllImport("Arccom.dll")]
        static public extern IntPtr CreateITPos();

        [DllImport("Arccom.dll")]
        static public extern IntPtr DeleteITPos(IntPtr pClassNameObject);

        [DllImport("Arccom.dll")]
        static public extern int ITPosSet(IntPtr pClassNameObject, byte[] key, byte[] value, int len);

        [DllImport("Arccom.dll")]
        static public extern int ITPosGet(IntPtr pClassNameObject, byte[] key, byte[] value, int len);

        [DllImport("Arccom.dll")]
        static public extern int ITPosRun(IntPtr pClassNameObject, int cmd);

        // in
        public int OperationCode = -1;
        public string Amount = "";
        public string CurrencyCode = "";
        public string ArcusID = "";

        // out
        public string ResponseCodeHost = "";
        public string ReferenceNumber = "";
        public string Slip = "";
        public int Run()
        {
            // create object
            IntPtr itpos = CreateITPos();

            if (itpos == null)
            {
                // if object not created
                return -1;
            }

            byte[] amt = Encoding.ASCII.GetBytes(Amount);
            byte[] cur = Encoding.ASCII.GetBytes(CurrencyCode);
            byte[] id = Encoding.ASCII.GetBytes(ArcusID);

            // set require parameters
            ITPosSet(itpos, Encoding.ASCII.GetBytes("amount"), amt, amt.Length);
            ITPosSet(itpos, Encoding.ASCII.GetBytes("currency"), cur, cur.Length);
            ITPosSet(itpos, Encoding.ASCII.GetBytes("ARCUS_ID"), id, id.Length);

            if (ReferenceNumber.Length > 0)
            {
                byte[] rrn_in = Encoding.ASCII.GetBytes(ReferenceNumber);
                ITPosSet(itpos, Encoding.ASCII.GetBytes("rrn"), rrn_in, rrn_in.Length);
            }

            byte[] empty = { 0 };
            ITPosSet(itpos, Encoding.ASCII.GetBytes("NO_CHECK_LAST_OP"), empty, 0);

            if (!File.Exists(@"debug"))
            {

```

```

        // disable all temporary files
        ITPosSet(itpos, Encoding.ASCII.GetBytes("TRACE_FILE"), empty, 0);
        ITPosSet(itpos, Encoding.ASCII.GetBytes("OUTPUT_FILE"), empty, 0);
        ITPosSet(itpos, Encoding.ASCII.GetBytes("RESULT_FILE"), empty, 0);
        ITPosSet(itpos, Encoding.ASCII.GetBytes("CHEQ_FILE"), empty, 0);
    }

    // start operation
    int res = ITPosRun(itpos, OperationCode);

    byte[] rrn = new byte[20];
    byte[] rc = new byte[20];
    byte[] slip = new byte[4096];

    // get result fields
    ITPosGet(itpos, Encoding.ASCII.GetBytes("rrn"), rrn, rrn.Length);
    ITPosGet(itpos, Encoding.ASCII.GetBytes("response_code"), rc, rc.Length);
    ITPosGet(itpos, Encoding.ASCII.GetBytes("slip"), slip, slip.Length);

    ResponseCodeHost = Encoding.ASCII.GetString(rc);
    ReferenceNumber = Encoding.ASCII.GetString(rrn);
    Slip = Encoding.ASCII.GetString(slip);

    // delete object
    DeleteITPos(itpos);

    return res;
}
}
}

```

Example for Object Pascal (Delphi)

```

unit test;

{$mode objfpc}{$H+}

interface

uses
    Classes, SysUtils, FileUtil, Forms, Controls, Graphics, Dialogs, StdCtrls, dynlibs;

type
    { TArcusForm }

    TArcusForm = class(TForm)
    AmountEdit: TEdit;
    AdminButton: TButton;
    MessageEdit: TEdit;
    MessageLabel: TLabel;
    BalanceButton: TButton;
    ResultEdit: TEdit;
    ResultLabel: TLabel;
    PurchaseButton: TButton;
    CurrencyEdit: TEdit;
    AmountLabel: TLabel;
    CurrenyLabel: TLabel;
    procedure AdminButtonClick(Sender: TObject);
    procedure BalanceButtonClick(Sender: TObject);
    procedure FormCreate(Sender: TObject);
    procedure PurchaseButtonClick(Sender: TObject);
    private
        { private declarations }
        procedure StartArcus(cmd : integer);
    public
        { public declarations }
    end;

var
    ArcusForm: TArcusForm;

implementation

```

```

{$R *.lfm}

{ TArcusForm }

type
  CharArray = array[0..1024] of char;
  // function creates object for interaction with terminal
  // result: pointer to object for interaction with POS terminal, if "nil" – error on object creation
  TCreateITPos = function () : Pointer; cdecl;
  // function deletes object (after this function it is not possible to use pointer, CreateITPos must be used)
  // input data
  // Ptr – object pointer
  TDeleteITPos = procedure (var Ptr : Pointer); cdecl;
  // parameter installation
  // input data
  // Ptr - object pointer
  // Key – parameter name, transferred as ASCII string with ending character
  // Value – parameter value, data buffer of specified Size, if Size=-1, length is calculated by terminal character (for ASCII string)
  // Size – size of transferred data, -1 – size is calculated automatically
  // result: if 0 – parameter is successfully recorded, not 0 - error
  TITPosSet = function (Ptr : Pointer; Key : PChar; Value : PChar; Size : Integer) : Integer; cdecl;
  // get parameter
  // input data
  // Ptr - object pointer
  // Key – parameter name, transferred as ASCII string with ending terminal character
  // output data
  // Value - parameter value, data buffer
  // Size – size of reserved buffer
  // result: if 0 - parameter is successfully recorded, not 0 - error (for example buffer size is too small)
  TITPosGet = function (Ptr : Pointer; Key : PChar; var Value : CharArray; Size : Integer) : Integer; cdecl;
  // run operation on terminal
  // input data
  // Ptr - object pointer
  // Cmd – command number from "ops.ini"
  TITPosRun = function (Ptr : Pointer; Cmd : Integer) : Integer; cdecl;

var
  // global object of loaded library
  IHandle : TLibHandle;
  // function creates object for interaction with terminal
  CreateITPos : TCreateITPos;
  // function deletes object (after this function it is not possible to use pointer, CreateITPos must be used)
  DeleteITPos : TDeleteITPos;
  // set parameter
  ITPosSet : TITPosSet;
  // get parameter
  ITPosGet : TITPosGet;
  // run operation on terminal
  ITPosRun : TITPosRun;

// transfer buffer to string – for Russian letters (win1251) – string from "?"
function ArrayToString(const a: array of char): string;
begin
  if Length(a) > 0 then
    SetString(Result, PChar(@a[0]), Length(a))
  else
    Result := '';
end;

// start work, clear pointers
procedure TArcusForm.FormCreate(Sender: TObject);
begin
  IHandle := NILHandle;
  CreateITPos := nil;
  DeleteITPos := nil;
  ITPosSet := nil;
  ITPosGet := nil;
  ITPosRun := nil;
end;

// procedure of interaction with terminal
procedure TArcusForm.StartArcus(cmd : integer);
var
  Res : Integer;
  Buffer : CharArray;

```

```

PosPtr : Pointer;
begin
    // library initialization incase of first run
    if IHandle = NILHandle then
    begin
        IHandle := LoadLibrary('Arccom.dll');

        if IHandle <> NILHandle then
        begin
            // initialization of interaction function
            Pointer(CreateITPos) := GetProcAddress(IHandle, 'CreateITPos');
            Pointer>DeleteITPos := GetProcAddress(IHandle, 'DeleteITPos');
            Pointer(ITPosSet) := GetProcAddress(IHandle, 'ITPosSet');
            Pointer(ITPosGet) := GetProcAddress(IHandle, 'ITPosGet');
            Pointer(ITPosRun) := GetProcAddress(IHandle, 'ITPosRun');
        end;
    end;

    // if function is not initialized – exit (with error code)
    if @CreateITPos = nil then exit;
    if @DeleteITPos = nil then exit;
    if @ITPosSet = nil then exit;
    if @ITPosGet = nil then exit;
    if @ITPosRun = nil then exit;

    // creates object for interaction with POS terminal, use this object for all other functions
    PosPtr := CreateITPos();

    // if object is not created – exit (with error code)
    if PosPtr = nil then exit;

    // use try, library does not generate exceptions
    try
        // set operation amount
        ITPosSet(PosPtr, 'amount', PChar(AmountEdit.Text), -1);
        // set currency
        ITPosSet(PosPtr, 'currency', PChar(CurrencyEdit.Text), -1);
        // run operation with "cmd" number
        Res := ITPosRun(PosPtr, cmd);

        // in Res variable - result of POS terminal work: 0 - success, all other values - error
        ResultEdit.Text := IntToStr(Res);

        // clear array to get additional information about operation
        FillChar(Buffer[0], sizeof(Buffer), #0);

        // operation RRN request
        ITPosGet(PosPtr, 'rrn', Buffer, sizeof(Buffer));

        MessageEdit.Text := ArrayToString(Buffer);
    except
        showmessage('unexpected error');
    end;

    // delete work object
    DeleteITPos(PosPtr);
end;

// run PURCHASE operation
procedure TArcusForm.PurchaseButtonClick(Sender: TObject);
begin
    StartArcus(1);
end;

// run BALANCE operation (operation number depends on ops.ini)
procedure TArcusForm.BalanceButtonClick(Sender: TObject);
begin
    StartArcus(62);
end;

// run Admin menu
procedure TArcusForm.AdminButtonClick(Sender: TObject);
begin
    StartArcus(99);
end;

```

end.

ANNEX 3

LIST OF SUPPORTED OPERATIONS

Cash register SW user interface must support ARCUS 2 module to perform the following operations:

- For trade terminals – (for Bank)
 - 1 PURCHASE (CASH ADVANCE)
 - 2 REFUND (CREDIT, ACCOUNT REPLENISHMENT)
 - 3 CANCEL LAST (CANCEL ADVANCE / CREDIT)
 - 4 CLOSE DAY (BATCH, DAY, PACKET RECONCILIATION — Z report)
 - 5 PRINT OPERATION LOG
 - 6 CASHIER MENU (ADMINISTRATOR MENU)
- Hotels, gas stations, restorans
 - 7 PREAUTHORIZATION (amount reservation) [optional]
 - 8 PREAUTHORIZATION COMPLETION [optional]
- If available
 - 9 CANCEL BY RECEIPT NUMBER [optional]

For most cases the following operations are defined:

- REFUND – returns funds to the client account. In most cases it can be performed if it is necessary to cancel operation that was performed in another batch or another terminal. Client card must be presented to perform this operation. Other data can differ (receipt number, RRN, amount) and can be configured. REFUND is a credit operation that transfers funds so it takes long time to complete it (from 3 to 40 days).
- CANCEL LAST – technical cancellation without requiry of any transaction data (receipt number, RRN, PAN etc.). This command can be performed when it is necessary to cancel last successful payment operation (cancellation of other operations can also be performed, depending on processing type). If it is impossible to cancel operation corresponding message will be displayed on the screen.
- UNIVERSAL REVERSAL – to cancel specified operation by transaction data and with client card presentation. Required data can be configured. In most cases it is possible to perform cancellation of the operation that was performed in current shift (RRN or receipt number is required). Funds returns back to the client account rather fast.



It is possible to get RRN, so it is better to use it for UNIVERSAL REVERSAL and FERUND operations.



Possibility to use operations must be discussed with the Bank

ANNEX 4

PROTOTYPES OF ARCUS.H DIALOG FUNCTIONS

```
/**
 * \brief Menu dialog      - menu description;
 * \param caption [in]    - menu title;
 * \param items [in]      - pointer to menu item;
 * \param number [in]     - number of menu item;
 * \param timeout [in]    - selection timeout (milliseconds);
 * \param userdata [in]   - pointer to user data (see DialogsCBS);
 * \function returns number of selected menu item or «-1» in case of decline
 */
typedef int (__cdecl *cbMenu)(const char *caption, const char **items, int number,
uint32_t timeout, void *userdata);
/**
 * \brief Yes/No dialog    - yes/no dialog;
 * \param caption [in]    - dialog title;
 * \param message [in]    - message;
 * \param timeout [in]    - selection timeout (milliseconds);
 * \param userdata [in]   - pointer to user data (see DialogsCBS);
 * \function returns 1 - yes, 0 - no.
 */
typedef int (__cdecl *cbYesNoDialog)(const char *caption, const char *message, uint32_t
timeout, void *userdata);
/**
 * \brief Warning dialog   - important information;
 * \param message [in]    - message;
 * \param timeout [in]    - selection timeout (milliseconds);
 * \param userdata [in]   - pointer to user data (see DialogsCBS);
 * \returns 0 or «-1» after specified timeout.
 */
typedef int (__cdecl *cbWarningDialog)(const char *message, uint32_t timeout, void
*userdata);
/**
 * \brief Information message to the client - short information for the client;
 * \param message [in]    - message;
 * \param userdata [in]   - pointer to user data (see DialogsCBS);
 * \function returns 0 - success, «-1» error.
 */
typedef int (__cdecl *cbInfo)(const char *message);
/**
 * \brief Terminal status message - terminal status;
 * \param message [in]    - message;
 * \param userdata [in]   - pointer to user data (see DialogsCBS);
 * \function returns 0 - success, «-1» error.
 */
typedef int (__cdecl *cbStatus)(const char *message, void *userdata);
/**
 * \brief Edit string dialog - dialog to change entered data;
 * \param caption [in]    - dialog title;
 * \param data [in]       - data for change;
 * \param buffer [out]    - outgoing buffer;
 * \param buffersize [in] - size of outgoing buffer;
 * \param mask [in]       - mask of entered data;
 * \param mode [in]       - entry mode;
 * \param timeout [in]    - selection timeout (milliseconds);
 * \param userdata [in]   - pointer to user data (see DialogsCBS);
 * \function returns 0 - success, «-1» error.
 */
typedef int (__cdecl *cbStringEdit)(const char *caption, const char *data, char *buffer,
size_t buffersize, const char *mask, const char *mode, uint32_t timeout, void *userdata);
typedef int (__cdecl *cbI344Window)(void);
typedef int (__cdecl *cbI344DisplayLine)(const char *text, unsigned int line, unsigned int
column, unsigned int inv_type);
typedef int (__cdecl *cbI344ClearLine)(unsigned int line);
typedef int (__cdecl *cbI344KeyReq)(unsigned int to);
```

Example

```
DialogsCBS callbacks;
DialogsI344CBS i344callbacks;

if (use_console_dialogs != 0)
{
    callbacks.userdata = nullptr;
    callbacks.size = sizeof(callbacks);
    callbacks.cbMenuItem = callback_menu;
    callbacks.cbYesNoDialog = callback_yesno;
    callbacks.cbWarningDialog = callback_warning;
    callbacks.cbInfo = callback_info;
    callbacks.cbStatus = callback_status;
    callbacks.cbStringEdit = callback_stringedit;

    i344callbacks.size = sizeof(i344callbacks);
    i344callbacks.cbI344OpenWindow = callback_i344openwindow;
    i344callbacks.cbI344ClearWindow = callback_i344clearwindow;
    i344callbacks.cbI344DisplayLine = callback_i344displayline;
    i344callbacks.cbI344ClearLine = callback_i344clearline;
    i344callbacks.cbI344KeyReq = callback_i344keyrequest;

    ITPosSet(itpos, "DialogsCBS", std::to_string((long)&callbacks).c_str(), -1);
    ITPosSet(itpos, "DialogsI344CBS", std::to_string((long)&i344callbacks).c_str(), -1);
}
```
