ECRCommX
for connecting to Ingenico POS-terminals using B-POS1 protocol
(ActiveX-component)

ingenico GROUP

Contents

Document revision history

- **February 02, 2021 Version 1.9.4.6**
  - Added missing statuses for get_LastStatMsgCode
  - Changed possible values for property *TxnType*

- **April 17, 2020 Version 1.9.3.5**
  - Fixed problems with migration to crossplatform core

- **July 16, 2018 Version 1.9.0.1**
  - Added method *UseMac*

- **February 09, 2017 Version 1.8.3.0**
  - Added property *SlipPrinted*

- **February 09, 2017 Version 1.8.2.0**
  - Added property *PanHash*

- **September 14, 2016 Version 1.8.1.0**
  - Added method *ReqDataFile*
  - Added property *DataFile*

- **May 20, 2016 Version 1.8.0.0**
  - Added method *Ping*
  - Added method *CheckTerminal*

- **March 16, 2016 Version 1.7.10.0**
  - Added property *ExtraCardData*

- **December 24, 2015 Version 1.7.9.0**
  - Added property *CryptedData*

- **February 12, 2015 Version 1.7.8.0**
  - Added method *CashAdvance*
  - Added method *ReadBankCard*
  - Added property *FlagAcquirer*

- **August 18, 2014 Version 1.23 (library version 1.7.7.0)**
  - Added method *SetExtraXmlData*

- **June 11, 2014 Version 1.22 (library version 1.7.6.1)**
  - Added method *CorrectTransaction*

- **May 22, 2014 Version 1.21 (library version 1.7.5.0)**
  - Added method *SetScreen*

- **April 30, 2014 Version 1.20 (library version 1.7.3.0)**
  - Added method *SendFile*

- **April 28, 2014 Version 1.19 (library version 1.7.2.4)**
  - Added method *useLogging*

- **March 12, 2014 Version 1.18 (library version 1.7.0.6)**
  - Added method *StartScenario*
  - Added method *SetExtraPrintData*
  - Added method *CloseApp*
  - Added property *ScenarioData*

- **February 18, 2014 Version 1.17 (library version 1.7.0.1)**
  - Added method *CommOpenTCP*
  - Added method *SelectApp*
  - Added property *LibraryVersion*
  - Added error description for TCP communication

- **February 11, 2014 Version 1.16**
  - Added description for Confirmation usage with Refund transaction

- **January 17, 2014 Version 1.15 (library version 1.7.0.0)**
  - Added method *PurchaseService*
  - Added method *POSGetInfo*
  - Added method *POSExTransaction*
  - Added property *AddData*
  - Added property *TerminalInfo*
  - Added property *DiscountName*
  - Added property *DiscountAttribute*
  - Added property *ECRDataTM*

  Visual studio C++ 2012 Redistributable required

- **July 19, 2013 Version 1.14**
  - Added property *CurrencyCode*

- **Jun 12, 2013 Version 1.13**
  - Added method *IdenifyCard*
  - Added property *RNK*
  - Added return values of property *TxnType*
  - Added return values of parameter *bAccNumber* in methods *Balance, Deposit, IdenifyCard*

- **May 16, 2013 Version 1.12**
  - Changed order of return values of property *LastStatMsgCode*

- **April 16, 2013 Version 1.11**
  - Added parameter *ulInvoiceNumber* to method *Completion*
  - Added methods *Balance, Deposit, PurchaseService.*
  - Added properties *Track3, TrnStatus, Currency, TrnBatchNum*
  - Added return values of property *LastStatMsgCode*

- **March 19, 2013 Version 1.10**

- Removed *EnterControlMode, ExitControlMode* methods
- Added method *SetControlMode*
- Added method *ReadCard*
- Added parameter *bECRStatus* to methods *ExchangeStatuses*
- Added return value of property *Key, TermStatus*
- Changed return value of property *LastStatMsgCode*

- **January 16, 2013 Version 1.9**
  - Unattended POS support
  - Added method *Completion*
  - Added return value of property *ResponseCode*
  - Added return value of property *TxnType*

- **October 26, 2012 Version 1.8**
  - Property *emvAID* increased to 32 bytes

- **September 12, 2012 Version 1.7**
  - Added method *SetErrorLang*

- **June 22, 2012 Version 1.6**
  - Added parameter *bMerchIdx* to methods *ReqReceiptByInv* and *GetTxnDataByInv*
  - Changed return value type of *EntryMode*

- **June 15, 2012 Version 1.5**
  - Removed *bRcvTimeout* from methods *CommOpen()* and *CommOpenAuto()*
  - Corrected some mistakes in method's descriptions

- **May 31, 2012 Version 1.4**
  - Added method *ReqCurrReceipt***,** *get_Receipt* and property *Receipt* for retrieving of last formatted receipt from terminal
  - Added method *PrintLastSettleCopy*
  - Added method *PrintBatchJournal*
  - Added property *Amount*
  - Added property *AddAmount*
  - Added property *TxnType*
  - Added property *EntryMode*
  - Added property *emvAID*
  - Added property *TotalsCommon*
  - Changed method *GetReceiptByInv* to *ReqReceiptByInv*
  - Example of usage is extended and moved to Appendix A.

- **May 22, 2012 Version 1.3**
  - Changed function *GetTxnDataByInv* (added *MerchIdx* parameter)

- **May 16, 2012 Version 1.2**
  - Fixed some mistakes in function descriptions

- **May 14, 2012 Version 1.1**
  - Added *CheckConnection, GetBatchTotals, GetTxnDataByInv, GetTxnNum,*
  - *GetTxnDataByOrder* methods.

- Added *TxtNum, SignVerif* properties.
- Added “Pin entry needed” status message.
- Added mandatory argument **bsRRN** to *Refund* method

- **April 17, 2012 Version 1.0 Document created**

General information

The component realizes a high level application interface between Windows personal computer (PC) or Windows based POS systems for interaction with Ingenico EFT POS terminals using ECR protocol by RS232 or USB connections.

Interface description

The component realizes the interface BPOS1Lib, with the help of which all the actions are being fulfilled.

Running transactions for fulfilling, and also setting the initial parameters, are done with the help of methods of the interface BPOS1Lib. The methods are immediately returning control to the calling side because transactions are performing in a separate thread.

System requirements and installation

Library is x86 based and should work under x86 application environment

Supported OS:
Windows 8 (x86 / x64), Windows XP (x86 / x64), Windows 7 (x86 /x64), Windows 2003, Windows 2008

Register x32 library in Windows x86:
Installation must be done under Administrator user rights.
ECRCommX.dll can be copied to any location on local drive.
To register ActiveX use **regsvr32 ECRCommX.dll**
After successful registration of component, a corresponding message should appear.

Register in x64 library Windows x64:
Installation must be done under Administrator user rights.
ECRCommX.dll can be copied to any location on local drive.
To register ActiveX use **regsvr32 ECRCommX.dll**
After successful registration of component, a corresponding message should appear.

How to register x32 library in Windows x64:
Installation must be done under Administrator user rights.
ECRCommX.dll can be copied to any location on local drive.
To register ActiveX use **C:\windows\syswow64\regsvr32 ECRCommX.dll**
After successful registration of component, a corresponding message should appear.

Additional Requirements:
Visual Studio C++ Redistributable 2012 (from library version 1.7.0.0)
Visual Studio C++ Redistributable 2005 (before library version 1.7.0.0)

When initializing transaction, the *BPOS1Lib* class creates a thread for connecting with terminal and returns the control. To get the results of transaction fulfilling there is used the property *LastResult* (or method *get_LastResult(**pVal**)*. If function returns 0 in *pVal* value, then

the transaction was done successfully, otherwise there was an error, the code of which can be obtained with the property *LastErrorCode* (method*get_ErrorCode*), and the description with Error Description (method *get_ErrorDescription*). See the list of error codes below.

Note. Before getting properties **CardNumber, ResponseCode** it is necessary to ensure that **LastResult** is not equal 2, i.e. the control thread has completed its fulfilling, otherwise values will not be defined.

Error processing

For all functions two possible return values are available – ***S_OK*** in case of success, and ***S_FALSE*** in case of error.

If function returns ***S_FALSE***, application should call *get_LastErrorCode()* function for retrieving detailed information about the error occurred.

Status messages processing

While terminal performs some transaction it can inform you about its current state. For example, about authorization on banking host or card reading process. To check this information use *LastStatMsgCode and LastStatMsgDescription properties* to get the last status of terminal.

Timeouts

Method CommOpen has no timeout, just trying to open port, that you put into this method;

Method CommOpenAuto throughout all active COM-ports on PC, performs ping to everyone with 500ms timeout;

All other methods, like Purchase, Refund etc. has timeout 6 seconds.

Interface methods

CommOpen(BYTE bPort, LONG lBaudRate)

Opens communication port.

[IN] **bPort** – COM port number, 1– “COM1”, 2 – "COM2" etc.

[IN] **lBaudRate** – data transmission speed, e.g. 19200, 38400 etc.

CommOpenTCP(BSTR bsIP, BSTR bsPort)

Opens communication port.

[IN] **bsIP** – IP address, e.g. “192.168.2.33”, “12.25.1.22” etc.

[IN] **bsPort** – IP-port, e.g. 1255, 2050 etc.

CommOpenAuto(LONG lBaudRate)

The same as **CommOpen** except there is performed an automatic search of connected terminal on ports 1…255.

[IN] **lBaudRate** – data transmission speed, e.g. 19200, 38400 etc.

CommClose(void)

Closes communication port opened with **CommOpen.**

No parameters.

CheckConnection(BYTE bMerchIdx)

Terminal will check connection to authorization host.

***[IN]* bMerchIdx** – index of merchant, which will be used for this transaction. Starts from 1.

Purchase(ULONG ulAmount, ULONG ulAddAmount, BYTE bMerchIdx)

Performs "Purchase" transaction.

Purchase is withdrawal from customer's account to pay for goods / services.

***[IN]* ulAmount** – amount of purchase

***[IN]* ulAddAmount** – additional amount (discount). May be used to set purchase discount. In this case ulAmount is original amount and ulAddAmount is discount. Thus, the final amount will be the following: ulAmount – ulAddAmount.

***[IN]* bMerchIdx** – index of merchant, which will be used for purchase transaction. Starts from 1.

Confirmation to POS is required.

Receipt can be received by calling ReqCurrReceipt function in case of successful response. Please see how to retrieve the receipt.

Refund(ULONG ulAmount, ULONG ulAddAmount, BYTE bMerchIdx, BSTR bsRRN)

Performs "Refund" transaction.

Refund to customer's account when he wants to return goods. Refund is allowed to be done at any time, regardless of time when the original purchase transaction was performed.

***[IN]* ulAmount** – amount of purchase

***[IN]* ulAddAmount** – *reserved for future use.*

***[IN]* bMerchIdx** – index of merchant which will be used for refund transaction. Starts from 1.

***[IN]* bsRRN** – Retrieval reference number of original purchase transaction.

Confirmation to POS is required for some financial applications.

Please note, that for the financial application, where Refund can't be cancelled, the confirmation is optional. Please contact application vendor for clarifications.

Receipt can be received by calling ReqCurrReceipt function in case of successful response. Please see how to retrieve the receipt.

Void(ULONG ulInvoiceNum, BYTE bMerchIdx)

Performs “Void” transaction.
Cancel the transaction (purchase or refund). Cancellation of the transaction can be carried out only until the end of settlement day, when the original transaction was performed, because funds that were debited from the customer's account will be credited to the merchant not earlier than the next settlement day.

***[IN]* ulInvoiceNum** – invoice number of original transaction
***[IN]* bMerchIdx** – index of merchant, which will be used for refund transaction. Starts from 1.

Receipt can be received by calling ReqCurrReceipt function in case of successful response. Please see how to retrieve the receipt.

Balance(BYTE bMerchIdx, BSTR bsCurrCode, BYTE bAccNumber)

Performs “Balance Inquiry” transaction.
Return information about balance account of customer`s card.

***[IN]* bMerchIdx** – index of merchant, which will be used for balance transaction. Starts from 1.
***[IN]* bsCurrCode** – currency code of transaction. Not mandatory.
***[IN]* bAccNumber** – account number. Not mandatory.
bAccNumber has one of following values:

0 – DEFAULT
1 – SAVINGS
2 – CHECKING
3 – CREDIT
4 – UNIVERSAL

Confirmation to POS is required.

Receipt can be received by calling ReqCurrReceipt function in case of successful response. Please see how to retrieve the receipt.

Deposit(BYTE bMerchIdx, ULONG ulAmount, BSTR bsCurrCode, BYTE bAccNumber)

Performs “Deposit” transaction.
Deposit to customer's account.

***[IN]* bMerchIdx** – index of merchant of original transaction. Starts from 1.
***[IN]* ulAmount** – amount of deposit.
***[IN]* bsCurrCode** – currency code of transaction. Not mandatory.
***[IN]* bAccNumber** – account number. Not mandatory.
bAccNumber has one of following values:

0 – DEFAULT
1 – SAVINGS

2 – CHECKING
3 – CREDIT
4 – UNIVERSAL

Confirmation to POS is required.

Receipt can be received by calling ReqCurrReceipt function in case of successful response. Please see how to retrieve the receipt.

CashAdvance(BYTE bMerchIdx, ULONG ulAmount, BSTR bsCurrCode, BYTE bAccNumber)

Performs "Cash Advance" transaction.
Cash withdrawal and debiting of the card account by the specified amount.

***[IN]* bMerchIdx** – index of merchant of original transaction. Starts from 1.
***[IN]* ulAmount** – amount of deposit.
***[IN]* bsCurrCode** – currency code of transaction. Not mandatory.
***[IN]* bAccNumber** – account number. Not mandatory.
bAccNumber has one of following values:

0 – DEFAULT
1 – SAVINGS
2 –CHECKING
3 – CREDIT
4 – UNIVERSAL

Confirmation to POS is required.

Receipt can be received by calling ReqCurrReceipt function in case of successful response. Please see how to retrieve the receipt.

Completion(BYTE bMerchIdx, ULONG ulAmount, BSTR bsRRN, ULONG ulInvoiceNum)

Performs completion of previously performed transaction. Completion *is allowed to be done after the original purchase transaction. Completion* amount can`t be bigger than amount of t*he original purchase transaction.*

***[IN]* bMerchIdx** – index of merchant of original purchase transaction. Starts from 1.
***[IN]* ulAmount** – amount of completion
***[IN]* bsRRN** – Retrieval reference number of original purchase transaction.
***[IN]* ulInvoiceNum** – invoice number of original preauthorization

Confirmation to POS is required.

Receipt can be received by calling ReqCurrReceipt function in case of successful response. Please see how to retrieve the receipt.

ReadCard(void)

This function is used to read only discount card

ReadBankCard(void)

This function is used to read any card

PurchaseService(BYTE bMerchIdx, ULONG ulAmount, BSTR bsServiceParams)

Performs “Service” transaction.
Special operation, which sends additional data with parameters to the host using 63.89 field. Operation is supported only by TITP financial protocol ("Compass+" processing). ***[IN]*** **bMerchIdx** – index of merchant, which will be used for the service transaction. Starts from 1.

> ***[IN]*** **ulAmount** – amount of service operation.
> ***[IN]*** **bsServiceParams** – string with "serv_num/nominal/parameters) format. "/" symbol must separate all the necessary values. For example, "0109//1/5". The first component is 4-digit substring (zero padding from left side must be performed), terminal will perform service operation using 109 service without nominal, but will include parameters "1" and "5".

Confirmation to POS is required.

Receipt can be received by calling ReqCurrReceipt function in case of successful response. Please see how to retrieve the receipt.

IdentifyCard(BYTE bMerchIdx, BSTR bsCurrCode, BYTE bAccNumber)

Performs “Identify card” transaction.
Return customer`s RNK if identify card was successful.

> ***[IN]*** **bMerchIdx** – index of merchant, which will be used for balance transaction. Starts from 1.
> ***[IN]*** **bsCurrCode** – currency code of transaction. Not mandatory.
> ***[IN]*** **bAccNumber** – account number. Not mandatory.
> **bAccNumber** has one of following values:
> 0 – DEFAULT
> 1 – SAVINGS
> 2 – CHECKING
> 3 – CREDIT
> 4 – UNIVERSAL

Confirmation to POS is required.

Receipt can be received by calling ReqCurrReceipt function in case of successful response. Please see how to retrieve the receipt.

POSGetInfo(void)

Returns POS information (software version, profile ID and the list of acquirers will be returned by *TerminalInfo* property).

> *No parameters.*

POSExTransaction (void)

Performs InfoCall operation.
InfoCall is the operation that sends service information to host (operation is supported only by TITP financial protocol and "Compass+" processing).

No parameters.

Settlement(BYTE bMerchIdx)

Performs Settlement transaction.
Settlement is the calculation of totals amount according to transaction data and sending them to authorization host. After Settlement transaction all transaction data will be deleted. Should be performed at the end of the settlement day.
Terminal will print a receipt (if printer is available) with totals of current batch and return these totals to component. To get them use *Totals properties (see description below).*

***[IN]* bMerchIdx** – index of merchant, which will be used for refund transaction. Starts from 1. If equals 0, terminal will perform settlement and return totals for all available merchants.

Receipt can be received by calling ReqCurrReceipt function in case of successful response. Please see how to retrieve the receipt.

PrintBatchTotals(BYTE bMerchIdx)

Terminal will print receipt (if it's possible) with totals of current batch* and return these totals to component. To get them use *Totals properties (see description below).*
Does not perform Settlement transaction unlike Settlement() method.
**Batch – is the transaction data within current settlement day.*
It prints the same receipt as after settlement.

***[IN]* bMerchIdx** – index of merchant which will be used for current transaction. Starts from 1. If equals 0, terminal will print receipts and return totals for all available merchants.

Receipt can be received by calling ReqCurrReceipt function in case of successful response. Please see how to retrieve the receipt.

PrintLastSettleCopy(BYTE bMerchIdx)

Terminal will print receipt of previous settlement transaction.

***[IN]* bMerchIdx** – index of merchant which will be used for current transaction. Starts from 1. If equals 0, terminal will print receipts and return totals for all available merchants.

Receipt can be received by calling ReqCurrReceipt function in case of successful response. Please see how to retrieve the receipt.

PrintBatchJournal(BYTE bMerchIdx)

Terminal will print receipt with info about all transactions of current batch.

[IN] **bMerchIdx** – index of merchant which will be used for current transaction. Starts from 1. If equals 0, terminal will print receipts and return totals for all available merchants.

Receipt can be received by calling ReqCurrReceipt function in case of successful response. Please see how to retrieve the receipt.

GetBatchTotals(BYTE bMerchIdx)

Terminal will return totals of current batch* to component. To get them use *Totals properties (see description below).*
Method performs same as PrintBatchTotals but terminal does not print receipt.
**Batch – is the transaction data within current settlement day.*

[IN] **bMerchIdx** – index of merchant which will be used for current transaction. Starts from 1. If equals 0, terminal will return totals for all available merchants.

GetTxnDataByInv(ULONG ulInvoiceNum, BYTE bMerchIdx)

Terminal will return transaction data (Response code, PAN... etc.) of transaction with given Invoice Number.

[IN] **ulInvoiceNum** – invoice number of required transaction

[IN] **bMerchIdx** – index of merchant which will be used for current transaction. Starts from 1.

GetTxnNum(void)

Terminal will return a number of transactions into transaction journal of terminal.
To get number of transactions use TxnNum property.

GetTxnDataByOrder(ULONG ulOrderNum)

Terminal will return transaction data (Response code, PAN... etc.) of transaction with given Order Number.

[IN] **ulOrderNum** – order number of transaction in transaction journal. Starts from 1 and should not be greater than number of transactions (see GetTxnNum() method).

ReqCurrReceipt(void)

This function is used to send request to POS for the content of current transaction receipt. (Approved / Decline)
In case of successful command receipt will be received by ***get_Receipt***

ReqReceiptByInv(ULONG ulInvoiceNum, BYTE bMerchIdx)

This function is used to get receipt of transaction with current invoice number.
In case of successful command receipt will be received by ***get_Receipt***

[IN] **ulInvoiceNum** – invoice number of required transaction.

[IN] **bMerchIdx** – index of merchant which will be used for current transaction. Starts from 1.

Receipt can be received by calling ReqCurrReceipt function in case of successful response. Please see how to retrieve the receipt.

Confirm(void)

This function is used to confirm the transaction for terminal. It should be used after terminal has finished the financial transaction/scenario (*Purchase, Refund, except Void*) and LastResult is eq. 0. **Otherwise terminal will cancel it.**

SelectApp(BSTR bsAppName, ULONG ulAppIdx)

This function is used to select application on POS to proceed operations in the chosen app. Usage: after communication open, and before operation is needed to perform.

[IN] **bsAppName** – the name of application on POS, e.g. “TE7E”, “TE75” etc. Max size is 15 bytes.

[IN] **ulAppIdx** – index of application on POS, e.g. 12555, 3458 etc.

CloseApp(void)

This function is used to close application on POS to finish all operation with app.

StartScenario(ULONG ulScenarioID, BSTR bsScenarioData)

This function is used to run scenario on POS.

[IN] **ulScenarioID** – index of scenario on POS, e.g. 1, 17 etc.

[IN] **bsScenarioData** – the xml-buffer with scenario specification. Max size is 1200 bytes.

SetExtraPrintData(BSTR bsExtraPrintData)

This function is used to set additional data before the financial transaction (*Purchase, Refund, etc., except Void*). Data will be cleaned after each transaction.

[IN] **bsExtraPrintData** – the additional data which can be printed on receipt. Max size is 1024 bytes.

SetExtraXmlData(BSTR bsExtraXmlData)

This function is used to set additional data before the financial transaction (*Purchase, Refund, etc., except Void*). Data will be cleaned after each transaction.

[IN] **bsExtraXmlData** – the additional data which is presented XML. Max size is 512 bytes.

SetErrorLang(BYTE bErrLanguage)

This function is used to set language of property (*messages that library returns, not POS*)

get_LastStatMsgDescription.

***[IN]* bErrLanguage** – type of language:

0 – ENG
1 – UKR
2 – RUS

SendFile(BSTR bsFullPath, BYTE bECRDataType, BYTE bECRCommand)

This function is used to send file to POS terminal.

***[IN]* bsFullPath** – location path of a file subject to printing:

***[IN]* bECRDataType** – type of file:

1 – receipt
2 – bmp
3 – data

***[IN]* bECRCommand** – type of operation determining terminal's actions towards received file:

0 – save
1 – print
2 – print and delete

CorrectTransaction(ULONG ulAmount, ULONG ulAddAmount)

This function is used to send updated information when operation is in progress (LastStatMsgCode = 11).

***[IN]* ulAmount** – amount of purchase.

***[IN]* ulAddAmount** – additional amount (discount). May be used to set purchase discount. In this case ulAmount is original amount and ulAddAmount is discount. Thus, the final amount will be the following: ulAmount – ulAddAmount.

useLogging(BYTE bLoggingLevel, BSTR bsFilePath)

This function is used to set log level and full path to log file

***[IN]* bLoggingLevel** – indicate level of log detailing:

0 – no logs
1 – simple log level
2 – detailed log level with get-function

***[IN]* bsFilePath** – full path to log file.

UseMac(BYTE macType, BSTR bsKey)

This function is used to set mac algorithm and key for it. If method is not called, BPOS will try to work without MAC verification

***[IN]* macType** – indicate mac algorithm:

0 – AES

***[IN]* bsKey** – key in hex format.

get_LastResult(BYTE* pVal), property *LastResult*

Gets the result of fulfilling of the last transaction

***[OUT]* pVal**- Returned values:

0 – successfully fulfilled

1 – error (to get the error code, use the property LastErrorCode or the method get_LastErrorCode)

2 – in progress

get_LastErrorCode (BYTE* pVal), property *LastErrorCode*

***[OUT]* pVal** - returned values:

1 – error opening COM port

2 – need to open COM port

3 – error connecting with terminal

4 – terminal returned an error. For additional analysis Response Code is used (see below).

get_LastErrorDescription(BSTR* pVal), property *LastErrorDescription*

***[OUT]* pVal** – returns error description – null-terminated string for error description. Max size is 100 bytes.

Gets text description of error either from the library or from the terminal (if *LastErrorCode* is 4)

Cancel(void)

Cancels fulfilling the operation in case ***LastResult*** returns 2.

Ping(void)

This function is used to check terminal connection status except in case ***LastResult*** returns 2.

CheckTerminal(void)

Check terminal connection status in the operation in case ***LastResult*** returns 2 and **LastStatMsgCode** doesn't change a long time.

get_LastStatMsgCode(BYTE* pVal), property *LastStatMsgCode*

***[OUT]* pVal -** returns one following status codes:

0 – status code is not available.
1 – card was read
2 – used a chip card
3 – authorization in progress
4 – waiting for cashier action
5 – printing receipt
6 – pin entry is needed
7 – card was removed
8 – EMV multi aid’s
9 – waiting for card
10 – in progress
11 – correct transaction
12 – Pin input wait key
13 – Pin input backspace pressed
14 – Pin input key pressed
15 – Account Selection
16 – Purchase only
17 – Confirmation Purchase only
18 – Waiting finger match verification

get_LastStatMsgDescription(BSTR* pVal), property *LastStatMsgDescription*

***[OUT]* pVal** – returns error description – null-terminated string for status description. Max size is 100 bytes.

Gets text description of status from the terminal (if LastStatMsgCode is not 0)

get_ResponseCode(ULONG* pVal), property *ResponseCode*

***[OUT]* pVal** – returns response code.
Is used in case of approved / not approved authorization.
Codes below 1000 are received from host.
1000 – General error (should be used in exceptional case)
1001 – Transaction canceled by user
1002 – EMV Decline
1003 – Transaction log is full. Need close batch
1004 – No connection with host
1005 – No paper in printer
1006 – Error Crypto keys
1007 – Card reader is not connected
1008 – Transaction is already complete

get_Receipt (BSTR* pVal), property *Receipt*

***[OUT]* pVal** – returns null-terminated string for current receipt slip of transaction.

Length is variable, maximum length is **32648** bytes *(reserved for 140-150 financial transactions in batch totals transaction log printing)*
If the length eaqual to 0, it means no receipt slip. Should be requested in case LastResult = 0 or (LastResult = 1 and LastErrorCode = 4) TEXT Encoding WIN1251.

get_LibraryVersion(BSTR* pVal), property *LibraryVersion*

[OUT] pVal – current library version.
Gets the library version which is used on PC currently.

ReqDataFile (void)

This function is used to send request to POS for the content of data file. In case of successful command data file will be received by get_DataFile

get_DataFile (VARIANT* pVal), property *DataFile*

***[OUT]* pVal** – returns SAFEARRAY of bytes for data file.
Maximum size is 204800 bytes

Functions (properties) described below are available only for financial transactions (purchase, refund, void) and for GetTxnDataByInv and GetTxnDataByOrder. Otherwise they will return empty results.

get_PAN(BSTR* pVal), property *PAN*

***[OUT]* pVal** – returns PAN of card. Length is variable.
Gets the number of client's account in case transaction successfully completed.

get_PanHash(BSTR* pVal), property *PanHash*

***[OUT]* pVal** – returns PAN hash of card. Lengh is variable.
Gets the number of client's account in case transaction successfully completed.

get_SlipPrinted(BYTE* pVal), property *SlipPrinted*

***[OUT]* pVal** – returns result of print operation on POS-terminal.
0 – slip has not been printed
1 – slip has been printed

get_DateTime(BSTR* pVal), property *DateTime*

***[OUT]* pVal** – returns null-terminated string for transaction date time
Gets transaction date and time in format YYMMDDhhmmss in case transaction successfully completed.

get_TerminalID(BSTR* pVal), property *TerminalID*

***[OUT]* pVal** – returns null-terminated string for TerminalID. Length is 8 bytes.
Gets Terminal ID.

get_MerchantID(BSTR* pVal), property *MerchantID*

***[OUT]* pVal** – returns null-terminated string for MerchantID. Length is variable.
Gets Merchant ID.

get_AuthCode(BSTR* pVal), property *AuthCode*

***[OUT]* pVal** – returns null-terminated string for Authorization code of transaction. Length is 6 bytes.
Gets Authorization Code.

get_Amount(ULONG *pVal), property *Amount*

***[OUT]* pVal** – gets final amount. It can differ from initial amount that was sent from the ECR

get_AddAmount(ULONG *pVal), property *AddAmount*

***[OUT]* pVal** – gets final additional amount. Additional amount can consist discount or bonus amount if it was used by terminal.

get_TxnType(BYTE* pVal), property *TxnType*

[OUT] **pVal** – gets transaction type of current transaction. Intended for usage for GetTxnDataByInv and GetTxnDataByOrder methods.
TxnType has one of following values:

0 – undefined
1 – Purchase
2 – Refund
3 – Void
4 – Completion
5 – IdentifyCard
6 – BalanceInquiry
7 – Deposit
8 – CardVoucher
9 – Cashback // deprecated
10 – Installment
20 – CashAdvance
21 – AccountFunding
22 – AccountConfirmation
23 – PurchaseWithCashback

get_EntryMode(BYTE* pVal), property *EntryMode*

[OUT] **pVal** – Get type of the card that was used to perform transaction. Length - three bytes.
EntryMode has one of following values:

0 – undefined
1 – Magnetic stripe card
2 – EMV chip card
3 – Contactless chip card
4 – Contactless stripe card
5 – Fallback (magnetic stripe was used by card that has EMV chip)
6 – Manual (card number was entered manually)

get_emvAID(BSTR* pVal), property *emvAID*

[OUT] **pVal** – returns EMV AID, as string. Max length is 32 bytes. If not EMV card was used *emvAID* will be empty.

get_ExpDate(BSTR* pVal), property *ExpDate*

[OUT] **pVal** – returns null-terminated string for card's expiration date.
Format of returned value is YYMM.

get_CardHolder(BSTR* pVal), property *CardHolder*

[OUT] **pVal** – returns null-terminated string for Authorisation code of transaction. . Length is variable. Gets Cardholder name.

get_IssuerName(BSTR* pVal), property *IssuerName*

[OUT] **pVal** – returns null-terminated string for issuer's name of card. Length is variable.

get_InvoiceNum(ULONG* pVal), property *InvoiceNum*

***[OUT]* pVal** – returns invoice number of transaction.
This number is used to void the original transaction.

get_CompletionInvoiceNum(ULONG* pVal), property *CompletionInvoiceNum*

***[OUT]* pVal** – returns invoice number of second transaction in two-pass scheme. It will be presented only after original purchase was complete (*Completion* was executed).
This number is used in case when transaction has been completed to void completion receipt.

get_RRN(BSTR* pVal), property *RRN*

***[OUT]* pVal** – returns null-terminated string for RRN of transaction. Length is 6 bytes.
RRN (Retrieval reference number) is unique for each transaction.

get_SignVerif (BYTE* pVal), property *SignVerif*

***[OUT]* pVal** – returns:
1 – if signature verification is needed for current transaction.
0 – if it is not needed.

Signature verification – is one of the cardholder verification methods. If verification was failed, transaction should be voided.

get_Track3(BSTR* pVal), property *Track3*

***[OUT]* pVal** – returns Track3 of card. Length is variable.

Gets the specific magnetic stripe information of card in case transaction successfully completed.

get_AddData(BSTR* pVal), property *AddData*

***[OUT]* pVal** – returns 63.89 field that contains host service data.

Can return the specific information from financial packet after service operation performing.

get_CryptedData(BSTR* pVal), property *CryptedData*

***[OUT]* pVal** – returns encrypted data.

Can return the specific information, which will be encrypted by POS.

get_ExtraCardData(BSTR* pVal), property *ExtraCardData*

***[OUT]* pVal** – returns extra information about card in TLV format.

Can return the extra information about card (length up to 150 symbols), which will be in TLV format send by POS.

get_TerminalInfo(BSTR* pVal), property *TerminalInfo*

***[OUT]* pVal** – returns POS information.

Gets the string of terminal software version + terminal profile ID + POS S/N + the list of acquirers after *POSGetInfo* operation. For example: TE7E118 0000005100CT20221154/AQ1/AQ2/AQ3.

get_DiscountName(BSTR* pVal), property *DiscountName*

***[OUT]* pVal** – returns information about the card.

Gets the information about the card, if it is also presented in Discount cards table.

get_DiscountAttribute(BSTR* pVal), property *DiscountAttribute*

***[OUT]* pVal** – returns information about the card.

Gets the information about the card attribute, if it is also presented in Discount cards table.

get_ECRDataTM(BSTR* pVal), property *ECRDataTM*

***[OUT]* pVal** – returns specific information from POS.

Gets the specific string (length up to 50 symbols) from POS.

get_TrnStatus(BYTE * pVal), property *TrnStatus*

***[OUT]* pVal** – Get transaction status

TrnStatus has one of following values:

0 – undefined
1 – approved
2 – declined
3 – reversed
4 – canceled

get_Currency(BSTR * pVal), property *Currency*

***[OUT]* pVal** – return currency of transaction.

get_TrnBatchNum(ULONG* pVal), property *TrnBatchNum*

***[OUT]* pVal** – returns batch number of transaction.

get_RNK(BSTR* pVal), property *RNK*

***[OUT]* pVal** – returns customer`s RNK. Max size is 20 bytes.

Available only for IdentifyCard method.

get_CurrencyCode(BSTR* pVal), property *CurrencyCode*

***[OUT]* pVal** – returns currency code. Max size is 3 bytes.

Functions (properties) below are available only for financial transactions and ReadBankCard. Otherwise they will return empty results.

get_FlagAcquirer(ULONG* pVal), property *FlagAcquirer*

***[OUT]* pVal** – determine native card.
Gets the statement is this native card or no.
FlagAcquirer has one of following values:
0 – false
1 – true

Totals properties

Functions (properties) below are available only for Settlement, PrintBatchTotals, GetBatchTotals. Otherwise they will return empty results.

get_TotalsDebitAmt(ULONG* pVal), property *TotalsDebitAmt*

***[OUT]* pVal** – total amount of debit transactions. Includes purchases.
Gets the total amount of debit transactions within current batch.

get_TotalsDebitNum(ULONG* pVal), property *TotalsDebitNum*

***[OUT]* pVal** – number of debit transactions. Includes purchases.
Gets the total number of debit transactions within current batch.

get_TotalsCreditAmt(ULONG* pVal), property *TotalsCreditAmt*

***[OUT]* pVal** – total amount of credit transactions. Includes refunds.
Gets the total amount of credit transactions within current batch.

get_TotalsCreditNum(ULONG* pVal), property *TotalsCreditNum*

***[OUT]* pVal** – number of credit transactions. Includes refunds.
Gets the total number of credit transactions within current batch.

get_TotalsCancelledAmt(ULONG* pVal), property *TotalsCancelledAmt*

***[OUT]* pVal** – total amount of cancelled transactions. Includes voids.
Gets the total amount of cancelled transactions within current batch.

get_TotalsCancelledNum(ULONG* pVal), property *TotalsCancelledNum*

***[OUT]* pVal** – number of cancelled transactions. Includes voids.
Gets the total number of cancelled transactions within current batch.

get_TxnNum(ULONG* pVal), property *TxnNum*

***[OUT]* pVal** – number of transactions.
Gets the number of transactions stored in transaction journal of terminal
Available only for GetTxnNum method.

Functions (properties) below are available only for StartScenario. Otherwise they will return empty results.

get_ScenarioData(BSTR* pVal), property *ScenarioData*

***[OUT]* pVal** – returns any information in xml about scenario.
Gets the xml with information about scenario, transaction, etc.

UNATTENDED SECTION

get_Key(BYTE* pVal), property *Key*

[OUT] pVal – returns pressed key

! USED FOR UN-ATTENDED POS'es ONLY

get_TermStatus(BYTE* pVal), property *TermStatus*

***[OUT]* pVal** – returns status of POS Terminal
Terminal status has one of following values:

11 – POS. UNCKNOWN STATUS
12 – POS. WORK IN NORMAL MOD
13 – POS. SYSTEM OPERATION
14 – POS. CLOSE BATCH NEED
15 – POS. READERS FAILED.

! USED FOR UN-ATTENDED POS'es ONLY

CONTROL MODE FUNCTIONS

EnterControlMode(void)

Enter into display and keyboard control mode from ECR.
Can't be used during financial transaction

! USED FOR UN-ATTENDED POS'es ONLY

ExitControlMode(void)

! USED FOR UN-ATTENDED POS'es ONLY

SetControlMode(VARIANT_BOOL isCtrlMode)

Set Control Mode:
Enter into display and keyboard control mode from ECR(ON), or
Exit from control mode (OFF).
Can't be used during financial transaction

***[IN]* isCtrlMode –** parameter for ON(1) or OFF(0) CONTROL Mode

! USED FOR UN-ATTENDED POS'es ONLY

ReadKey(BYTE bTimeOut)

Returns code of pressed key. *(Used in Control mode only)*

[IN] **bTimeOut –** parameter for key waiting (in)

! USED FOR UN-ATTENDED POS'es ONLY

DisplayText (BYTE bBeep)

Display text of prepared line by SetLine function. *(Used in Control mode only)*

[IN] **bBeep –** parameter for sound behavior on terminal (in)

Time OUT to "ECR not connected" – 15 sec

! USED FOR UN-ATTENDED POS'es ONLY

SetLine(BYTE bRow, BYTE bCol, BSTR bsText, BYTE bInvert)

Set line Text (*Used in Control mode only)*

[IN] **bRow –** line display offset Y (number of line)

[IN] **bCol –** line display offset X (character offset)

[IN] **bInvert –** if 0, no inversion for string, else inverted

Time OUT to "ECR not connected" – 15 sec

! USED FOR UN-ATTENDED POS'es ONLY

SetScreen(ULONG ulScreenNumber)

Set screen number of special external display with reserved screens (*Used in Control mode only)*

[IN] **ulScreenNumber –** number of screen

! USED FOR UN-ATTENDED POS'es ONLY

0 – no Screen

ExchangeStatuses(BYTE bECRStatus)

[IN] **bECRStatus –** status of ECR.

Send status of ECR and receive status of POS terminal. *(Used in Control mode only)*

ECR status has one of following values:

01 – ECR. NOT SUPPORTED
02 – ECR. WORK IN NORMAL MOD.
03 – ECR. CUSTOMER IN PROGRESS
04 – ECR. WORK IN MAINTANANEC MOD.
05 – ECR. NOT CONNECT.

! USED FOR UN-ATTENDED POS'es ONLY

Appendix A. Interface in use

Below there is an example (for VBA):

PURCHASE (simplified example)

```
Dim Terminal As Object
Const BaudRate As Double = 115200
Const timeout As Double = 200
Const Amount As Double = 1000
Const AddAmount As Double = 0
Const MerchantIdx As Double = 1


' ****************************************Wait Response FUNCTION
Public Function SetCellValueInt(ByVal sCellName As String, ByVal sCellValue As Integer, ByVal i As Integer) As Boolean
   If (sCellValue = 0) Then
      Worksheets("Sheet1").Range(sCellName).OffSet(0, i).Value = "empty"
   Else
      Worksheets("Sheet1").Range(sCellName).OffSet(0, i).Value = sCellValue
   End If
End Function

Rem  ***********************************************
Rem * Author: , Desc:  TestUnit *
Rem ***********************************************

Public Function SetCellValue(ByVal sCellName As String, ByVal sCellValue As String, ByVal i As Integer) As Boolean

   If (sCellValue = "") Then
      Worksheets("Sheet1").Range(sCellName).OffSet(0, i).Value = "empty"
   Else
      Worksheets("Sheet1").Range(sCellName).OffSet(0, i).Value = sCellValue
   End If

End Function

Public Function WaitResponse(ByRef obj)
   Dim Res As Boolean
   Dim PAN As String
   Dim EntryMode As String
   Dim LastStMsCode As Integer
   LastStMsCode = 0
```

```
Do While obj.LastResult = 2
      DoEvents
      If Terminal.LastStatMsgCode <> 0 And Terminal.LastStatMsgCode <> LastStMsCode
Then
'HAS POS status progress message, we can display it on the ECR
'POS can sent multiple status messages as (CardRead , PIN Required, Host Auth
LastStMsCode = Terminal.LastStatMsgCode
      End If
     If LastStMsCode == 11 Then
               PAN = Terminal.PAN
               EntryMode = Terminal.EntryMode
               Terminal.CorrectTransaction(Amount,  AddAmount)
          End If
     End If
   Loop
End Function
Sub Pause(Wait)
   Dim Current As Long

   Current = Timer
   Do Until Timer - Current >= Wait
     DoEvents
   Loop
End Sub

' ****************************************Return values FUNCTION
Sub ReturnValues(ByRef Terminal As Object)

  mErr = WaitResponse(Terminal)

  If Terminal.LastResult = 1 And Val(Terminal.ResponseCode) <> 20 Then
     'Error occured
     Status = "ERROR"
     LastResult = Terminal.LastErrorCode
     ErrorDescr = Terminal.LastErrorDescription

     If Terminal.LastErrorCode = 4 Then
     'transaction has been declined (host or terminal declined), we need response code
       Res = SetCellValueInt("B13", Terminal.ResponseCode, 0)
       Res = SetCellValue("B14", Terminal.TerminalID, 0)
       Res = SetCellValue("B15", Terminal.MerchantID, 0)
      'And other params
      End If
   Else

      Res = SetCellValue("B10", "Successful", 0)
```

```
        Res = SetCellValue("B13", Terminal.ResponseCode, 0)
        Res = SetCellValue("B14", Terminal.TerminalID, 0)
        Res = SetCellValue("B15", Terminal.MerchantID, 0)
        Res = SetCellValue("B16", Terminal.PAN, 0)
        Res = SetCellValue("B17", Terminal.RRN, 0)
        Res = SetCellValue("B18", Terminal.AuthCode, 0)
        Res = SetCellValue("B19", Terminal.DateTime, 0)
        Res = SetCellValueInt("B20", Terminal.InvoiceNum, 0)
        Res = SetCellValueInt("B21", Terminal.ExpDate, 0)
        Res = SetCellValue("B22", Terminal.CardHolder, 0)
        Res = SetCellValue("B23", Terminal.IssuerName, 0)
        Res = SetCellValue("B24", Terminal.Amount, 0)
        Res = SetCellValue("B25", Terminal.SignVerif, 0)
        Res = SetCellValueInt("B26", Terminal.TxnNum, 0)
        Res = SetCellValueInt("B27", Terminal.TxnType, 0)

        'This part is required for Transaction totals only
        Res = SetCellValue("B37", Terminal.TotalsDebitNum, 0)
        Res = SetCellValue("B38", Terminal.TotalsDebitAmt, 0)
        Res = SetCellValue("B39", Terminal.TotalsCreditNum, 0)
        Res = SetCellValue("B40", Terminal.TotalsCreditAmt, 0)
        Res = SetCellValue("B41", Terminal.TotalsCancelledNum, 0)
        Res = SetCellValue("B42", Terminal.TotalsCancelledAmt, 0)
    End If

End Sub

' ****************************************************************
' ************************s**************MAIN FUNCTION **************
Sub ECRMAIN()

Dim Res As Boolean

If (Terminal Is Nothing) Then
Rem Initialize library


Set Terminal = CreateObject("ECRCommX.BPOS1Lib")
Terminal.CommClose

Terminal.SetErrorLang(1) '***** Set library messages to Ukrainian

Res = Terminal.CommOpenAuto(BaudRate)

Res = Terminal.Purchase(Amount, AddAmount, MerchantIdx)

Call ReturnValues(Terminal)
```

```
If Terminal.LastResult = 0 Then
            Terminal.Confirm
          'Check if confirmation is received
            mErr = WaitResponse(Terminal) 'please see implementation above

          'Get formatted receipt into a one buffer
          If Terminal.LastResult = 0 Then

                    mErr=Terminal.ReqCurrReceipt
                  'Check  if  ReqCurrReceipt  is  sent
                    mErr = WaitResponse(Terminal)

                  If Terminal.LastResult = 0 Then
                            ReceiptSlip = Terminal.ReceiptSlip
                  End If
          End If

Else   If   Terminal.LastErrorCode   =   4   Then
                    mErr=Terminal.ReqCurrReceipt
                  'Check  if  ReqCurrReceipt  is  sent
                    mErr = WaitResponse(Terminal)

                  If Terminal.LastResult = 0 Then
                            ReceiptSlip = Terminal.ReceiptSlip
                  End If
End If



    'CLOSE Communication port
    Terminal.CommClose
   Set Terminal = Nothing
End If
End Sub
```

Importer in Ukraine: Ingenico Ukraine LLC, 6 Oleny Telihy str., 04112, Kyiv, Ukraine, Forum West Side Centre, building 7, 4th floor
Імпортер в Україні: ТОВ Інженіко Україна, вул. Олени Теліги 6, корп. 7, 4-ий поверх, 04112, Київ, Україна