Guide of POS_PC library (.NET)

POS_PC Library Info:

File Name : POS_PC.dll
Version : 1.4.9.0
Date : 1391/08/24

Dependency:

To use POS_PC library in your application, you must have Microsoft .NET Framework 2.0 or higher, installed on your Windows system.

What's new?

New features are:

- Adding MultiPayment service with three input parameters (totalAmount , multiPayment request data set list, PrintDetail)

Compatibility:

Current version support following functions and connection type of pos-pc that exists in previous versions.

Functions:

int **Debits_Goods_And_Service** (string Amount, string PayerID, string strMsg)
int **Bill_Payment_Service** (string BillID, string PayID, string Amount, string strMsg)
int **Payment** (string Amount, string PayerID, string AccountID)
int **Refund** (string Amount, string ReferenceNumber)
int **MultiPayment** (string TotalAmount, MultiPaymentReqDataSet[] RequestList)
int **MultiPayment** (string TotalAmount, MultiPaymentReqDataSet[] RequestList, byte PrintDetail)

Connection type :

Serial port connection (RS232 / USB plug)
tcp/ip connection

Class:

Dll Structure Name:

MultiPaymentReqDataSet

For requestList of multiPayment method. The fields of structure are:
- string AccountID
- string Amount
- string PayerID

Dll Class Names:

- **Globals**
  For configure connection type of pos-pc and configure the tcp/ip connection parameters.

- **Transaction**
  For using following functions, first you must create an object from this Class.

Transaction Class Functions:

int **Debits_Goods_And_Service** (string Amount, string PayerID, string strMsg)
int **Bill_Payment_Service** (string BillID, string PayID, string Amount, string strMsg)
int **Payment** (string Amount, string PayerID, string AccountID)
int **Refund** (string Amount, string ReferenceNumber)
int **MultiPayment** (string TotalAmount, MultiPaymentReqDataSet[] RequestList)
int **MultiPayment** (string TotalAmount, MultiPaymentReqDataSet[] RequestList, byte PrintDetail)

Function Details:

❖ int Debits_Goods_And_Service (String Amount , String PayerID , String strMsg)

Function is used to perform a debit (goods and service) transaction.

Parameters [in]:

Name	Format	Length (byte)	Existence	explain
Amount	string ASCII numeric	Variable MIN:1 MAX:12	mandatory	amount of transaction
PayerID	string ASCII numeric	Variable MIN:3 MAX:17	optional	payer id of transaction if necessary
strMsg	Hex string numeric /alphabetic	Variable MAX:160	optional	merchant message if necessary for print on client receipt.

Detailed description:

For performing a debit (Goods and service) transaction this function opens a serial or tcp/ip port (depending on connection type) at first, write a specific data (a message for pos which include the amount, Id Code and merchant custom message of transaction) on the port and then wait for receiving data (a message receive from pos) on it, then close the port and finally return a result for showing that if the transaction has done successfully or not.

Return Values:

Refer to Appendix "A"

- If the transaction is approved successfully, the return value is "**RET_OK"** and the return parameters are accessible via the public properties.

❖ int Bill_Payment_Service (string BillID, string PayID, string Amount, string strMsg)

Function is used to perform a Bill Payment transaction.

Parameters [in]:

Name	Format	Length (byte)	Existence	explain
BillID	string ASCII numeric	Variable MIN:6 MAX:13	mandatory	bill identity of transaction
PayID	string ASCII numeric	Variable MIN:6 MAX:13	mandatory	payment identity of transaction
Amount	string ASCII numeric	Variable MIN:1 MAX:12	mandatory	amount of transaction
strMsg	Hex string numeric /alphabetic	Variable MAX:160	optional	Merchant message if necessary for print on client receipt.

Detailed description:

For performing a bill payment transaction this function opens a serial or tcp/ip port (depending on connection type) at first, write a specific data (a message for pos which include the Bill ID, Payment ID, Amount and merchant custom message of transaction) on the port and then wait for receiving data (a message receive from pos) on it, then close the port and finally return a result for showing that if the transaction has done successfully or not.

Return Values:

Refer to Appendix "A"

- If the transaction is approved successfully, the return value is "**RET_OK"** and the return parameters are accessible via the public properties.

❖ int Payment (String Amount , String PayerID , String AccountID)

Function is used to perform a payment transaction.

Parameters [in]:

Name	Format	Length (byte)	Existence	explain
Amount	string ASCII numeric	Variable MIN:1 MAX:12	mandatory	amount of transaction
PayerID	string ASCII numeric	Variable MIN:3 MAX:17	optional	payer id of transaction if necessary
AccountID	English string numeric /alphabetic	Variable MAX:24	optional	account number id of payment transaction.

Detailed description:

For performing a payment transaction this function opens a serial or tcp/ip port (depending on connection type) at first, write a specific data (a message for pos which include the amount, payer id and account id of transaction) on the port and then wait for receiving data (a message receive from pos) on it, then close the port and finally return a result for showing that if the transaction has done successfully or not.

Note:

After successful transaction,

- The amount of the payment transaction will be transferred to the merchant account. Merchant account number determines from account id.

Return Values:

Refer to Appendix "A"

- If the transaction is approved successfully, the return value is "**RET_OK"** and the return parameters are accessible via the following public properties.

❖ int Refund (String Amount , String ReferenceNumber)

Function is used to perform a refund purchase transaction.

Parameters [in]:

Name	Format	Length (byte)	Existence	explain
Amount	string ASCII numeric	Variable MIN:1 MAX:12	mandatory	amount of transaction
Reference Number	string ASCII numeric	Variable MIN:1 MAX:18	mandatory	Reference number of debit transaction

Detailed description:

For performing a refund transaction this function opens a serial or tcp/ip port (depending on connection type) port at first, write a specific data (a message for pos which include the amount and reference number of transaction) on the port and then wait for receiving data (a message receive from pos) on it, then close the port and finally return a result for showing that if the transaction has done successfully or not.

Return Values:

Refer to Appendix "A"

- If the transaction is approved successfully, the return value is "**RET_OK"** and the return parameters are accessible via the public properties.

❖ int MultiPayment (String TotalAmount , MultiPaymentReqDataSet[] RequestList)

Function is used to perform a multiPayment transaction.

Parameters [in]:

Name	Format	Length	Existence	explain
Amount	string ASCII numeric	Variable(byte) MIN:1 MAX:12	mandatory	Total amount of multipayment transaction
RequestList	Array of MultiPaymentReqDataSet structure	Variable(index) MIN:1 MAX:10	mandatory	Request list of multipayment data set

MultiPaymentReqDataSet structure fields:

Name	Format	Length (byte)	Existence	explain
AccountID	string ASCII numeric	Variable MIN:1 MAX:10	mandatory	account number of request data set
Amount	string ASCII numeric	Variable MIN:1 MAX:10	mandatory	amount of request data set
PayerID	string ASCII numeric	Variable MIN:3 MAX:17	optional	payer id of request data set

Detailed description:

For performing a multiPayment transaction this function opens a serial or tcp/ip port (depending on connection type) at first, write a specific data (a message for pos which include the total amount and list of request data set (max 10 records) on the port and then wait for receiving data (a message receive from pos) on it, then close the port and finally return a result for showing that if the transaction has done successfully or not.

Return Values:

Refer to Appendix "A"

- If the transaction is approved successfully, the return value is "**RET_OK"** and the return parameters are accessible via the public properties.

❖ int MultiPayment (String TotalAmount , MultiPaymentReqDataSet[] RequestList, byte PrintDetail)

Function is used to perform a multiPayment transaction.

Parameters [in]:

Name	Format	Length	Existence	explain
Amount	string ASCII numeric	Variable(byte) MIN:1 MAX:12	mandatory	Total amount of multipayment transaction
RequestList	Array of MultiPaymentReqDataSet structure	Variable(index) MIN:1 MAX:10	mandatory	Request list of multipayment data set
PrintDetail	numeric	Fixed (1byte)	mandatory	Print Details of multipayment on receipt Value = 0 : Print on both receipt Value = 1: Print on customer receipt Value = 2: Print on merchant receipt Value = 3: Not printed on the receipt

MultiPaymentReqDataSet structure fields:

Name	Format	Length (byte)	Existence	explain
AccountID	string ASCII numeric	Variable MIN:1 MAX:10	mandatory	account number of request data set
Amount	string ASCII numeric	Variable MIN:1 MAX:10	mandatory	amount of request data set
PayerID	string ASCII numeric	Variable MIN:3 MAX:17	optional	payer id of request data set

Detailed description:

For performing a multiPayment transaction this function opens a serial or tcp/ip port (depending on connection type) at first, write a specific data (a message for pos which include the total amount and list of request data set (max 10 records) on the port and then wait for receiving data (a message receive from pos) on it, then close the port and finally return a result for showing that if the transaction has done successfully or not.

Return Values:

Refer to Appendix "A"

- If the transaction is approved successfully, the return value is "**RET_OK"** and the return parameters are accessible via the public properties.

❖ Public properties

SerialTransaction:
The local transaction number generated by POS for each transaction request.

TraceNumber:
The unique transaction number which is get from switch for the approved transactions.

PAN:
The permanent account number (the last 4 digit of card number).

BIN:
The bank identity number (the first 6 digit of card number).

TransactionDate:
The date of transaction in shamsi format (yyyy/mm/dd)

TransactionTime:
The time of transaction

AccountNo:
The merchant account number.

TerminalNo:
Terminal identity number.

ReasonCode:

If the transaction failed because of error number 109 (**ERR_POS_FAILED_TRANSACTION OR RET_NOK**), the reason of error will be accessible via this public properties.

See **ReasonCode**.**pdf** for more information.

Appendix A

Return Values:

Response Code	Error Define	Description	Transaction Status
100	RET_*OK*	The process for send and receive data performs successfully	OK
101	ERR_PC_INVALID_REC_SIZE	invalid message size received from pos	Failed
102	ERR_POS_INVALID_DATA	invalid message received from pc (message size, process code)	Failed
103	ERR_PC_INVALID_REC_PROCESS_CODE	invalid process code received from pos in response	Failed
104	ERR_PC_INVALID_AMOUNT	invalid input amount or invalid received amount from pos in response	Failed
105	ERR_PC_INVALID_INPUT_PAYERID	invalid input payer id	Failed
106	ERR_PC_INVALID_INPUT_TIMEOUT	invalid input timeout for waiting to receive data from pos on serial port (the value of it must be : Min = 2000 ms Max = 600000 ms)	Failed
107	ERR_PC_PORT_TIMEOUT_FOR_REC	timeout occurred while waiting for receive data from pos on serial port	Failed
108	ERR_POS_RESPONSE_RECEIVED_TOO_LATE	transaction failed: not receive response from server because of timeout or connection failed	Failed
109	ERR_POS_FAILED_TRANSACTION	transaction failed: credit is not sufficient, server is down or …	Failed
110	ERR_POS_PRINTER	transaction failed because of printer error	Failed

Response Code	Error Define	Description	Transaction Status
111	ERR_POS_COMMUNICATION	transaction failed because of connection error	Failed
112	ERR_POS_TO_SEND_TRANSACTION	transaction failed because of error in send settlement, send reversal or generate transaction	Failed
113	ERR_PC_INVALID_INPUT_PORTNAME	invalid input serial port name	Failed
114	ERR_POS_USER_ABORT	transaction failed because of aborting by user	Failed
115	ERR_PC_INVALID_INPUT_BILLID	invalid input bill id for bill payment transaction	Failed
116	ERR_PC_INVALID_INPUT_PAYID	invalid input payment id for bill payment transaction	Failed
117	ERR_PC_PORT_OPEN_FAILED	error occurred while opening selected serial port	Failed
118	ERR_PC_PORT_ACCESS_FAILED	I/O error or a specific type of security error : the serial port can't write data	Failed
119	ERR_PC_INVALID_PORT_STATE	I/O error occurs: the selected port isn't serial port	Failed
120	ERR_PC_INVALID_PORT_PARAMETERS	argument out of range exception: one or more of the properties for this instance are invalid(on serial port)	Failed
121	ERR_PC_INVALID_PORT_NAME	arguments provided to a method is not valid: the serial port name is invalid	Failed
122	ERR_PC_NULL_STR_TO_WRITE_IN_PORT	Argument Null Exception on serial port	Failed
123	ERR_PC_PORT_TIMEOUT_FOR_SEND	timeout occurred while send data on serial port	Failed

Response Code	Error Define	Description	Transaction Status
124	ERR_POS_CARD_SWIPE_FAILED	the card has not been swiped on pos	Failed
125	ERR_PC_INVALID_INPUT_ACCOUNT_ID	invalid input account id for payment transaction	Failed
126	ERR_POS_INVALID_INPUT_ACCOUNT_ID	invalid received account id from pc	Failed
127	ERR_POS_INVALID_INPUT_PAYERID	invalid received payer id from pc	Failed
128	ERR_POS_INVALID_INPUT_AMOUNT	invalid received amount from pc	Failed
129	ERR_POS_INVALID_INPUT_REFERENCE_NUMBER	invalid received reference number from pc	Failed
130	ERR_POS_INVALID_INPUT_BILL_ID	invalid received bill id from pc	Failed
131	ERR_POS_INVALID_INPUT_PAYMENT_ID	invalid received payment id from pc	Failed
132	ERR_POS_INVALID_INPUT_ADDITIONALDATA	invalid received additional data from pc	Failed
133	ERR_POS_INVALID_INPUT_MULTI_PAYMENT_AMOUNT	invalid received multi payment total amount from pc	Failed
134	ERR_POS_UNCONFIRM_REC_DATA	unconfirmed received data from pc by user	Failed
161	ERR_PC_INVALID_INPUT_MULTI_PAYMENT_REQUEST_LIST	Invalid input multi payment request data set list (max 10 records)	Failed
162	ERR_PC_INVALID_INPUT_MULTI_PAYMENT_AMOUNT	invalid input multi payment amount	Failed
163	ERR_PC_INVALID_INPUT_REFERENCE_NUMBER	invalid input reference number	Failed
164	ERR_POS_PC_CRCERROR_INVALID_DATA	Invalid received data on pos or pc (crc error)	Failed

Response Code	Error Define	Description	Transaction Status
165	ERR_PC_INVALID_POSPC_COMMUNICATION_TYPE	invalid input connection type of pos-pc (the value of it must be : Serial or tcp/ip)	Failed
166	ERR_PC_INVALID_INPUT_TCP_SOCKET_PORT	invalid input tcp/ip connection port number (the value of it must be : MIN : 1024 MAX: 65535)	Failed
167	ERR_PC_INVALID_INPUT_TCP_SOCKET_RECEIVED_TIMEOUT	invalid input timeout for waiting to receive data from pos on tcp/ip connection (the value of it must be : Min = 2000 ms Max = 600000 ms)	Failed
168	ERR_PC_TCP_SOCKET_FAILED	error occurred while creating socket on pc in tcp/ip communication	Failed
169	ERR_PC_TCP_SOCKET_SEND_MSG_FAILED	error occurred while send data to pos in tcp/ip communication	Failed
170	ERR_PC_TCP_SOCKET_RECEIVED_MSG_FAILED	error occurred while receiving data from pos on tcp/ip communication because of timeout or …	Failed
171	ERR_PC_INVALID_INPUT_MERCHANT_MESSAGE	invalid format of input merchant message	Failed
172	ERR_PC_PREPARE_TLV_MSG_FAILED	error occurred while prepare TLV message for send	Failed
173	ERR_PC_PORT_EXCEPTION_FOR_REC	Serial port exception for receiving data from pos	Failed
174	ERR_PC_NULL_STR_IN_READ_PORT	Serial port exception for receiving data from pos	Failed
175	ERR_PC_CALCULATE_CRC_ERROR	error occurred while calculate crc of message	Failed
200	ERR_POS_PC_OTHER	Other exception	Failed

Sample Code :

```
/* pos-pc via Serial port */

Globals.POSPC_CommunicationType = “serial”;

Transaction TXN = new Transaction();
POS_PC.Transaction.return_codes retCode = Transaction.return_codes.ERR_POS_PC_OTHER;

TXN.PC_PORT_Name = "COM1";
TXN.PC_PORT_BaudRate = 115200;
TXN.PC_PORT_ReadTimeout = 180000;

        /* Debit */
retCode = TXN.Debits_Goods_And_Service(strAmount, strPayerId, strMerchantMsg);
OR      /* BillPayment */
retCode = TXN.Bill_Payment_Service(strBillId, strPayCode, strAmount,   strMerchantMsg);
OR      /* Payment */
retCode = TXN.Payment(strAmount, strPayerId, strAcountId);
OR      /* Refund */
retCode = TXN.Refund (strAmount, strReferenceNumber);
OR       /*MultiPayment */
Transaction. MultiPaymentReqDataSet[] RequesttList = new Transaction. MultiPaymentReqDataSet[10];

RequestList[0].AccountID = strAccountId;
RequestList[0].Amount = strAmount;
 RequestList[0].PayerID = strPayerId;
. . . . ;
/* with 2 input parameters */
retCode = TXN.MultiPayment(strTotalAmount , RequesttList);

/* with 3 input parameters */
byte PrintDetail = 0;
retCode = TXN.MultiPayment(strTotalAmount , RequesttList, PrintDetail);

if (retCode == Transaction.return_codes.RET_OK)
{
        /* the transaction has been done */
}
else
{
        MessageBox.Show(retCode.ToString());
         /* If the transaction failed because of error number 109
        (ERR_POS_FAILED_TRANSACTION),   the reason of error will be accessible via TXN.ReasonCode */
}
```

Sample Code:

```
/* pos-pc via tcp/ip */

Globals.POSPC_CommunicationType = “tcp/ip”;
Globals.POSPC_TCPCOMMU_SocketPortNumebr= 1024;
Globals.POSPC_TCPCOMMU_SocketRecTimeout = 180000;

Transaction TXN = new Transaction();
POS_PC.Transaction.return_codes retCode = Transaction.return_codes.ERR_POS_PC_OTHER;

do
{       /* Debit */
        retCode = TXN.Debits_Goods_And_Service(strAmount, strPayerId, strMerchantMsg);
        OR      /* BillPayment */
        retCode = TXN.Bill_Payment_Service(strBillId, strPayCode, strAmount,   strMerchantMsg);
        OR      /* Payment */
        retCode = TXN.Payment(strAmount, strPayerId, strAcountId);
        OR      /* Refund */
        retCode = TXN.Refund (strAmount, strReferenceNumber);
        OR       /*MultiPayment */
        Transaction. MultiPaymentReqDataSet[] RequesttList = new Transaction.
        MultiPaymentReqDataSet[10];

        RequestList[0].AccountID = strAccountId;
        RequestList[0].Amount = strAmount;
         RequestList[0].PayerID = strPayerId;
        . . . . ;
        /* with 2 input parameters */
        retCode = TXN.MultiPayment(strTotalAmount , RequesttList);

        /* with 3 input parameters */
        byte PrintDetail = 0;
        retCode = TXN.MultiPayment(strTotalAmount , RequesttList, PrintDetail);

        if (retCode == Transaction.return_codes.RET_OK)
        {
                /* the transaction has been done */
        }
        else
        {
                MessageBox.Show(retCode.ToString());
                 /* If the transaction failed because of error number 109
                (ERR_POS_FAILED_TRANSACTION),   the reason of error will be accessible via
                TXN.ReasonCode */
        }

} while (
            (Globals.POSPC_CommunicationType.ToUpper() == "TCP/IP") &&
            (TXN.IS_FATAL_ERROR(retCode) == false)     );
```

- you can use following method to check the type of error:

  bool **IS_FATAL_ERROR**(Transaction.return_codes retCode)
  if the error of transaction is not a fatal error, the tcp/ip socket connection will not be closed.