شرکت پرداخت الکترونیک سامان

واحد نرم افزار

Sep.Pc2Pos (C#)

ابزار ارتباط رایانه با کارتخوان (پورت سریال، USB و شبکه)

مستند فنی

نگارش 4.5

سابقه بازنگری			
اقدام کننده	شرح	نگارش	تاریخ
گیلدا خسروی	ایجاد	1.0	94/11/5
گیلدا خسروی	به روز رسانی	2.0	95/02/19
گیلدا خسروی	به روز رسانی	3.0	95/04/14
گیلدا خسروی	به روز رسانی	4.0	95/04/30
سهیل فراهانی	به روز رسانی	4.1	95/09/09
گیلدا خسروی	به روز رسانی	4.2	95/10/04
گیلدا خسروی	به روز رسانی	4.3	96/01/14
سیده سارا حسینی	اضافه کردن قابلیت خرید شناسه دار	4.4	97/07/02
سیده سارا حسینی	اضافه کردن قابلیت خرید شناسه دار چند شناسه ای وجوه دولتی	4.5	98/04/01

نشر			
سمت	مرجع	تاریخ	امضا
تهیه کننده	گیلدا خسروی	94/11/5	
تضمین کیفیت	سهیل فراهانی	95/10/04	

فهرست محتویات

فهرست محتویات 3

1. بسته حاوی نرم افزار 4

2. پیش نیازها 4

3. عملکرد هر بخش 4

3.1 Sep.Pc2Pos.tlb 4

3.2 Sep.Pc2Pos.config 4

3.3 Sep.Pc2Pos.dll 5

4. انواع روشهای ارتباطی رایانه با دستگاه کارتخوان 5

5. کد نویسی 5

5.1 فرآیند کلی برنامه 5

5.2 ساختار اجرایی برنامه نمونه 13

5.3 ایجاد تراکنش Async 18

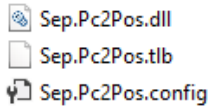
5.4 ایجاد تراکنشSync 22

6. راهنما 25

مقادیر کدهای پاسخ (Response Code) 27

نرم افزار ارتباطی PC-POS

1. بسته حاوی نرم افزار



در یک نگاه این نرم افزار در چند (فایل) ارائه شده است. فایل‌هایی که عدم وجود هریک باعث نقصان در کارکرد نهایی خواهند شد.

- Sep.Pc2Pos.dll
- Sep.Pc2Pos.tlb
- Sep.Pc2Pos.config

2. پیش نیازها

جهت استفاده از این نرم افزار، لازم است:

- حداقل .NET Framework 4.0 روی کامپیوتر مربوطه نصب شده باشد.
- POS در پیکربندی ارتباط با صندوق باشد.
- POS به شکل صحیح با کامپیوتر مرتبط باشد.
- درگاه ارتباطی POS با PC مشخص باشد. (بعنوان مثال COM1 یا ارتباط Ethernet)

3. عملکرد هر بخش

عملکرد هریک از بخش‌های نامبرده، به شرح ذیل می‌باشد:

3.1 Sep.Pc2Pos.tlb

این فایل جهت استفاده نمودن از dll در محیط هایی مانند VB6 مورد استفاده قرار می‌گیرد.

3.2 Sep.Pc2Pos.config

حاوی تنظیمات ابزار است.

- برای لاگ زدن در بخش `appSettings` ، در `value` مربوط به `key="LogPath"` ، می‌توان آدرسی که می‌خواهید لاگ در آنجا زده شود را قرار دهید.
- برای شمارش تراکنش ها به صورت صحیح در بخش `appSettings` ، در `value` مربوط به `key="AuthorizationId"` ، می‌توان شماره شروع تراکنش‌ها را قرار دهید.

- برای مشخص کردن حداقل مقدار ریالی هر تراکنش در بخش `appSettings` ، در `value` مربوط به `key="MinimumAmount"` ، می‌توان حداقل مقدار تراکنش خرید را قرار دهید. این مبلغ نباید کمتر از 1000 ریال باشد.

Sep.Pc2Pos.dll 3.3

این فایل جهت ارتباط PC با POS طراحی و تولید شده است.

با استفاده از این DLL امکان انجام تراکنش خرید به 2 صورت امکان‌پذیر می‌باشد:

- تراکنش خرید به صورت Async: در این تراکنش به ترتیب زیر عمل می‌شود:
  - مقداردهی پارامترهای پیکربندی
  - صدا زدن متد انتظار برای کشیدن کارت روی دستگاه کارتخوان
  - دریافت ترمینال و اطلاعات کارت در Event مربوطه
  - بعد از دریافت ترمینال و اطلاعات کارت، مقدار از رایانه به کارتخوان ارسال می‌شود.
  - بعد از انجام تراکنش پاسخ به کارتخوان در Event مربوطه برگردانده می‌شود.

- تراکنش خرید به صورت Sync: در این تراکنش به ترتیب زیر عمل می‌شود:
  - مقداردهی پارامترهای پیکربندی
  - ارسال مبلغ به دستگاه کارتخوان
  - بعد از انجام تراکنش پاسخ به کارتخوان در خروجی متد برگردانده می‌شود.

4. انواع روش‌های ارتباطی رایانه با دستگاه کارتخوان

- Network
- RS232
- USB

5. کد نویسی

بعد از اضافه کردن Sep.Pc2Pos.dll به برنامه، امکان استفاده از آن فراهم خواهد شد. در این dll متد و Eventهایی وجود دارد که از طریق آنها ارتباط با دستگاه پوز از طریق LAN و یا RS232 به صورت تک حسابی و یا دارای چند حساب تسویه فراهم می‌شود.

5.1 فرآیند کلی برنامه

Sep.Pc2Pos.dll شامل

- `bool Init(AccountType accountType, ClientLanguage language, string comPort)`

این متد جهت تنظیم مقادیر پیکربندی در نوع ارتباط COM مورد استفاده قرار می‌گیرد.

- AccountType: مشخص کننده نوع تسهیم تک حسابی و یا چند حسابی می‌باشد.

```
public enum AccountType
{
    Single = 0,
    Share = 1
}
```

{

- Language: مشخص کننده زبان پاسخ های بازگشتی از سمت پوز می‌باشد.

```
public enum ClientLanguage
{
    Persian = 0,
    English = 1
}
```

- ComPort: نشان دهنده پورت Com و یا USB متصل به رایانه می‌باشد و دارای مقداری مانند "COM4" می‌باشد.

- `bool Init(AccountType accountType, ClientLanguage language, int ListenPort, string IP)`

این متد جهت تنظیم مقادیر پیکربندی در نوع ارتباط Network مورد استفاده قرار می‌گیرد.

- AccountType: مشخص کننده نوع تسهیم تک حسابی و یا چند حسابی می‌باشد.

```
public enum AccountType
{
    Single = 0,
    Share = 1
}
```

{

- Language: مشخص کننده زبان پاسخ های بازگشتی از سمت پوز می‌باشد.

```
public enum ClientLanguage
{
    Persian = 0,
    English = 1
}
```

- ListenPort: پورت TCP که PC روی آن Listen می‌کند و روی دستگاه کارتخوان نیز همین مقدار باید تنظیم شده باشد.
- IP: مشخص کننده IP مربوط به PC می‌باشد و روی دستگاه کارتخوان باید ست شده باشد.

- `bool WaitForPos(int timeout)`

این متد باید در ابتدای هر تراکنش باید فراخوانی شود.

- timeout : مشخص کننده مدت زمان انتظار برای دریافت پاسخ از دستگاه کارتخوان به ثانیه. در صورتی که timeout برابر "0" مقداردهی شود این فیلد به صورت Don't Care در نظر گرفته می‌شود.

در حالت خرید و خرید شناسه دار تک شناسه ای:

- `bool SendAmount(List<string> amount, string authorizationId, List<PrintItemResult> printList, int timeout, string PurchaseID = null)`

از این متد برای ارسال مبلغ و شناسه ی خرید(در صورت انجام خرید شناسه دار) به دستگاه کارتخوان استفاده می‌شود. این متد بعد از کشیدن کارت روی دستگاه کارتخوان و دریافت اطلاعات کارت و ترمینال فراخوانی می‌شود.

- amount : مشخص کننده مبلغ تراکنش می‌باشد. در مدل تک حسابی این لیست شامل یک مبلغ است و در مدل چندحسابی لیست مبالغ اضافه شده و در انتهای لیست مجموع مبالغ باید اضافه گردند.
- authorizationId: زمانی که کارت کشیده می‌شود، شماره تراکنش (authorizationId) برگردانده می‌شود. این مقدار در این فیلد سمت دستگاه کارتخوان برگردانده می‌شود.
- printList: جهت چاپ داده اضافی روی کارتخوان استفاده می‌شود. . لطفا به راهنما برای توضیحات تکمیلی مراجعه نمایید.
- timeout : مشخص کننده مدت زمان انتظار برای دریافت پاسخ از دستگاه کارتخوان به ثانیه. در صورتی که timeout برابر "0" مقداردهی شود این فیلد به صورت Don't Care در نظر گرفته می‌شود.
- PurchaseID:این پارامتر از نوع رشته ای و با مقدار null به صورت پیش فرض می باشد که در صورت انجام خرید شناسه دار، شناسه ی خرید را ارسال می کند.

نکته: مقدار این پارامتر به صورت پیش فرضnull می باشد که در صورت خالی بودن این پارامتر خرید معمولی و در صورت پر بودن، خرید شناسه دار انجام می شود.لازم به ذکر است این مقدار در حال حاضر 30 رقم در نظر گرفته شده است.

در حالت خرید شناسه دار چند شناسه ای:

- `public bool SendAmountMultiPurchase (string SumAmn, string authorizationId, List<PrintItemResult> printList, int timeout, string PurchaseID = null)`

از این متد برای ارسال مبلغ ، شناسه ی خرید و شماره شبا با فرمتی که در ادامه ذکر خواهد شد به دستگاه کارتخوان استفاده می‌شود. این متد بعد از کشیدن کارت روی دستگاه کارتخوان و دریافت اطلاعات کارت و ترمینال فراخوانی می‌شود.

- SumAmn: مشخص کننده مبلغ تراکنش می‌باشد. این مبلغ به صورت مجموع مبالغ شناسه ها می باشد.
- authorizationId: زمانی که کارت کشیده می‌شود، شماره تراکنش (authorizationId) برگردانده می‌شود. این مقدار در این فیلد سمت دستگاه کارتخوان برگردانده می‌شود.
- printList: جهت چاپ داده اضافی روی کارتخوان استفاده می‌شود. . لطفا به راهنما برای توضیحات تکمیلی مراجعه نمایید.
- timeout : مشخص کننده مدت زمان انتظار برای دریافت پاسخ از دستگاه کارتخوان به ثانیه. در صورتی که timeout برابر "0" مقداردهی شود این فیلد به صورت Don't Care در نظر گرفته می‌شود.
- PurchaseID: این پارامتر از نوع رشته ای و با مقدار null به صورت پیش فرض می باشد که در صورت انجام خرید شناسه دار چند شناسه ای ، شناسه ی خرید، مبلغ و شماره شبا با فرمت زیر را ارسال می کند.

...; شماره شبا:مبلغ:شناسه; شماره شبا:مبلغ:شناسه ;شماره شبا:مبلغ:شناسه

به عنوان نمونه:

367011774140103001712010000001 :1000: IR210100004101047771208981;

نکته:در صورت انجام تراکنش خرید شناسه دار ، در حالت خرید شناسه دار تک شناسه ای، این پارامتر 30 رقم در نظر گرفته می شود و در صورت انجام خرید چند شناسه ای این پارامتر به صورت رشته ی ذکر شده (با مقدار بیشتر از 30 رقم) ارسال می گردد.لازم به ذکر است که حداکثر تعداد شناسه در هر تراکنش 10 عدد می باشد. همچنین dll جلوی تراکنش خرید چند شناسه ای با کمتر از دو شناسه را می گیرد و خطای 12 بر می گرداند.در صورتیکه شناسه نامعتبر باشد نیز خطای "27" (داده اضافی نامعتبر) برگشت داده می شود.

انجام تراکنش با شناسه خرید"00000000000000000000000000000000" نیز امکان پذیر می باشد که در این صورت مبلغ و شماره شبا سمت سوئیچ بررسی می گردد.

- `PosResultEventArgs BlockingTransaction(List<string> amount, List<PrintItemResult> printList, int timeout, string PurchaseID=null)`

این متد به عنوان متد Sync برای تراکنش خرید استفاده می‌شود. بعد از Initialization کافی است این متد را فراخوانی کرده و پاسخ را در خروجی متد دریافت نمایید. با استفاده از این متد نیاز به Assign نمودن Event ها نمی‌باشد.

از این متد برای ارسال مبلغ به دستگاه کارتخوان استفاده می‌شود. این متد بعد از کشیدن کارت روی دستگاه کارتخوان و دریافت اطلاعات کارت و ترمینال فراخوانی می‌شود.

- amount: مشخصکننده مبلغ تراکنش می‌باشد. در مدل تک حسابی این لیست شامل یک مبلغ است و در مدل چندحسابی لیست مبالغ اضافه شده و در انتهای لیست مجموع مبالغ باید اضافه گردند.
- authorizationId: زمانی که کارت کشیده می‌شود، شماره تراکنش (authorizationId) برگردانده می‌شود. این مقدار در این فیلد سمت دستگاه کارتخوان برگردانده می‌شود.
- printList: جهت چاپ داده اضافی روی کارتخوان استفاده می‌شود. لطفا به راهنما برای توضیحات تکمیلی مراجعه نمایید.
- timeout : مشخص کننده مدت زمان انتظار برای دریافت پاسخ از دستگاه کارتخوان به ثانیه. در صورتی که timeout برابر "0" مقداردهی شود این فیلد به صورت Don't Care در نظر گرفته می‌شود.
- PurchaseID:این پارامتر از نوع رشته ای و با مقدار null به صورت پیش فرض می باشد که در صورت انجام خرید شناسه دار، شناسه ی خرید را ارسال می کند.

نکته: مقدار این پارامتر به صورت پیش فرضnull می باشد که در صورت خالی بودن این پارامتر خرید معمولی و در صورت پر بودن، خرید شناسه دار انجام می شود.لازم به ذکر است این مقدار در حال حاضر 30 رقم در نظر گرفته شده است.

```
public PosResultEventArgs BlockingTransactionMultiPurchase (string SumAmn,
List<PrintItemResult> printList, int timeout, string PurchaseID = null)
```

- SumAmn : مشخص کننده مبلغ تراکنش می‌باشد. این مبلغ به صورت مجموع مبالغ شناسه ها می باشد.
- authorizationId: زمانی که کارت کشیده می‌شود، شماره تراکنش (authorizationId) برگردانده می‌شود. این مقدار در این فیلد سمت دستگاه کارتخوان برگردانده می‌شود.
- printList: جهت چاپ داده اضافی روی کارتخوان استفاده می‌شود. . لطفا به راهنما برای توضیحات تکمیلی مراجعه نمایید.
- timeout : مشخص کننده مدت زمان انتظار برای دریافت پاسخ از دستگاه کارتخوان به ثانیه. در صورتی که timeout برابر "0" مقداردهی شود این فیلد به صورت Don't Care در نظر گرفته می‌شود.
- PurchaseID: این پارامتر از نوع رشته ای و با مقدار null به صورت پیش فرض می باشد که در صورت انجام خرید شناسه دار چند شناسه ای ، شناسه ی خرید، مبلغ و شماره شبا با فرمت زیر را ارسال می کند.

...; شماره شبا:مبلغ:شناسه; شماره شبا:مبلغ:شناسه ;شماره شبا:مبلغ:شناسه

به عنوان نمونه:

36701177414010300171201000000 1 :1000: IR210100004101047771208981;

نکته:در صورت انجام تراکنش خرید شناسه دار ، در حالت خرید شناسه دار تک شناسه ای، این پارامتر 30 رقم در نظر گرفته می شود و در صورت انجام خرید چند شناسه ای این پارامتر به صورت رشته ی ذکر شده (با مقدار بیشتر از 30 رقم) ارسال می گردد.لازم به ذکر است که حداکثر تعداد شناسه در هر تراکنش 10 عدد می باشد. همچنین dll جلوی تراکنش خرید چند شناسه ای با کمتر از دو شناسه را می گیرد و خطای 12 بر می گرداند.

انجام تراکنش با شناسه خرید"00000000000000000000000000000000" نیز امکان پذیر می باشد که در این صورت مبلغ و شماره شبا سمت سوئیچ بررسی می گردد.

به عنوان خروجی متد:

با استفاده از Property های PosResultEventArgs می‌توانید، `IsSuccessful` (تراکنش در صورتی موفق است که این فیلد برابر True و کد بازگشتی برابر 00 باشد)، `ResponseCode` (کد بازگشتی)، `ResponseDescription` (شرح کد بازگشتی) ، `PaidAmount` (مبلغ پرداخت شده) ، `TraceNumber` (شماره پیگیری) ، `SerialId` (شماره سریال) ، `TxnDate` (تاریخ تراکنش)، `TerminalID` (شماره ترمینال) و `ReturnPAN` (شماره کارت) را به دست بیاورید. شماره کارت لطفا به صورت Mask شده نمایش داده شود.

- `void Dispose()`

در اتمام هر تراکنش این متد باید صدا زده شود.

- Event های

- `void CardSwiped(IPCPOS sender, CardSwipedEventArgs e)`

بعد از اینکه دارنده‌کارت، روی دستگاه کارت را کشید، شماره کارت برای رایانه ارسال می‌گردد. . با استفاده از Property های CardSwipedEventArgs می‌توانید، CardNumber(شماره کارت)، TerminalId (شماره ترمینال) و AuthorizationId (شماره تراکنش) را به دست بیاورید. شماره کارت لطفا به صورت Mask شده نمایش داده شود.

- `void ErrorReceived(IPCPOS sender, ErrorReceivedEventArgs e)`

اگر فرایند توسط کاربر بر روی کارتخوان لغو و یا در اطلاعات ورودی و ارتباط پوز با برنامه به خطایی رخ دهد، نتیجه در این رخداد ارسال می‌گردد. با استفاده از Property های ErrorReceivedEventArgs می‌توانید، `ErrorCode` (کد خطا)، `ErrorDescription` (شرح خطا) را به دست بیاورید.

- `void PosResultReceived(IPCPOS sender, PosResultEventArgs e)`

وقتی تراکنش تا انتها انجام شد و جواب از کارتخوان به برنامه برگشت داده شد، با استفاده از Property های PosResultEventArgs می‌توانید، `IsSuccessful` (تراکنش در صورتی موفق است که این فیلد برابر True و کد بازگشتی برابر 00 باشد)، `ResponseCode` (کد بازگشتی)، `ResponseDescription` (شرح کد بازگشتی) ، `PaidAmount` (مبلغ پرداخت شده) ، `TraceNumber` (شماره پیگیری) ، `SerialId` (شماره سریال) ، `TxnDate` (تاریخ تراکنش)، `TerminalID` (شماره ترمینال) و `ReturnPAN` (شماره کارت) را به دست بیاورید. شماره کارت لطفا به صورت Mask شده نمایش داده شود.

مراحل اجرای تراکنش Async به شرح ذیل می‌باشد:

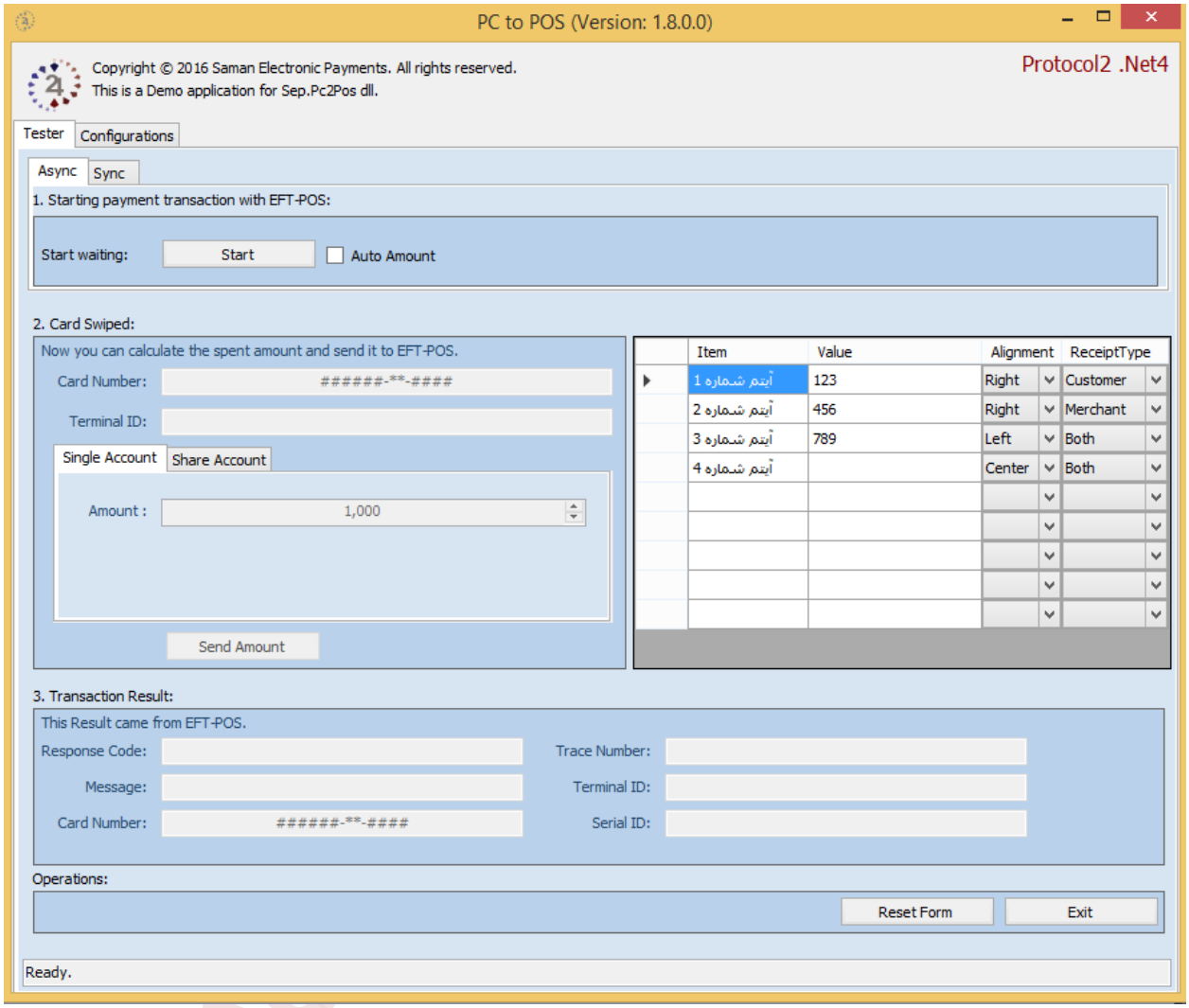
	Application	PosClient (Sep.Pc2Pos.dll)	EFT-POS
1	ایجاد یک نمونه از PosClientBase و نسبت دادن Eventها		
2	Method PosClientBase. Init (مقدار دهی PortName و یا ListenPort و IP) و Language و AccountType)		
4	PosClientBase.WaitForPos	□□□□□□	
5		□□□□□□	کشیدن کارت روی پوز
6		□ Raise Event PosClientBase _CardSwiped	
7	Event Handler PosClientBase _CardSwiped		
8	Method PosClientBase.SendAmount	□□□□□□	
9		□□□□□□	نمایش مبلغ و ورود رمز کارت
10		درصورت بروز اطلاعات نادرست در ورودی و یا لغو تراکنش بر روی کارتخوان توسط کاربر: □ Raise Event PosClientBase _ErrorReceived	
11	Event Handler PosClientBase _ErrorReceived		
12		در صورت دریافت پاسخ پس از ارسال تراکنش □ Raise Event PosClientBase _PosResultReceived	

		Event Handler PosClientBase _PosResultReceived	13

پرداخت الکترونیک سامان

5.2 ساختار اجرایی برنامه نمونه

ساختار اجرایی برنامه به صورت زیر می‌باشد:



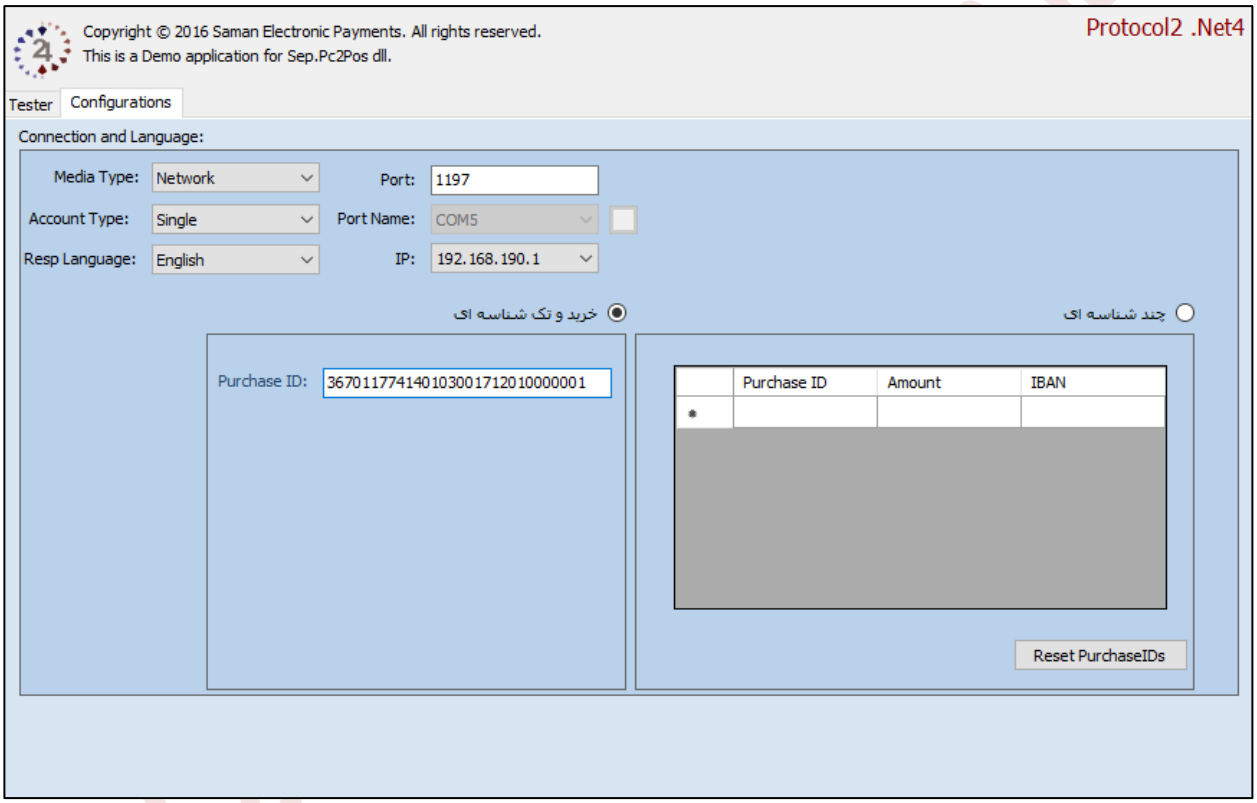
برنامه شامل 2 قسمت Tester و Configuration می‌باشد. در بخش Tester با انتخاب Tab مربوط به Sync و یا Async نوع تراکنش مشخص می‌شود:

در بخش Async:

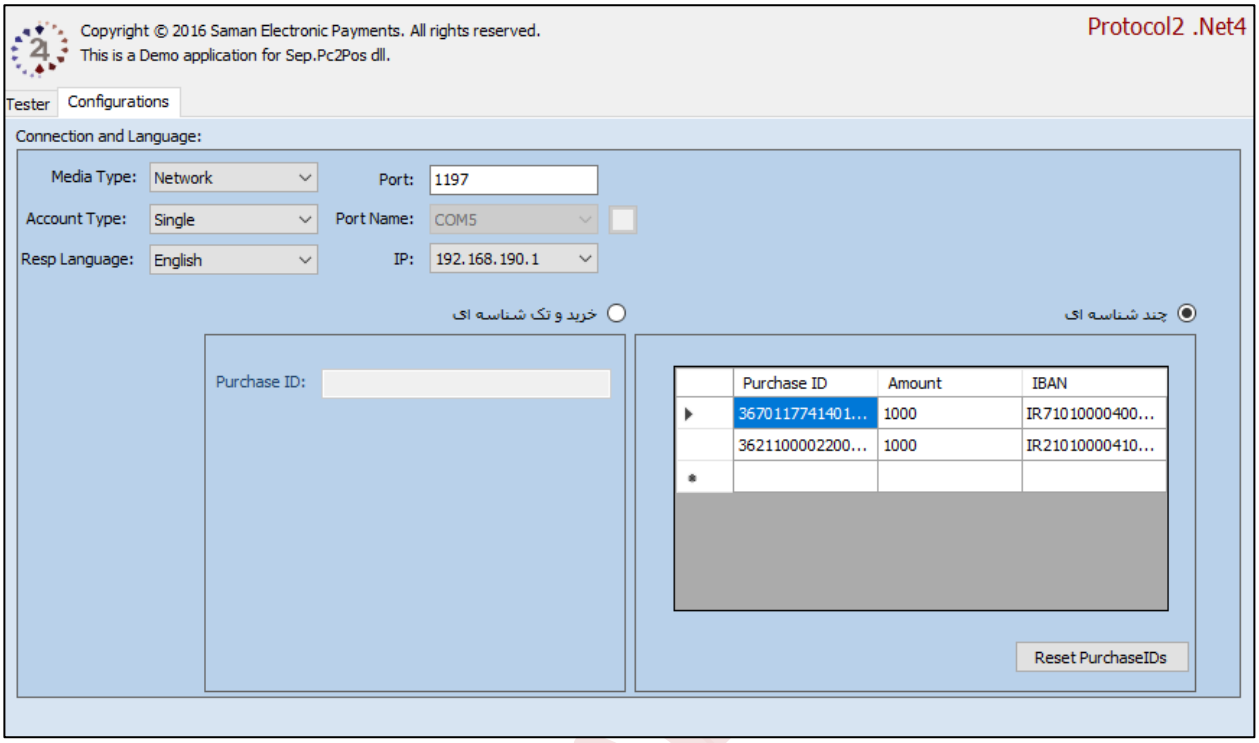
با زدن کلید Start ، برنامه در انتظار کشیدن کارت روی پوز می‌ماند. بعد از کشیدن کارت، شماره کارت کشیده شده و ترمینال دستگاه کارتخوان بر روی برنامه ظاهر می‌شود. با توجه به پیکربندی کارتخوان شماره ترمینال می‌تواند ارسال نشود. پس از محاسبه ی مبلغ تراکنش برای این کارت، کلید SendAmount فعال میشود. در حالت تک حسابی ، بعد از وارد کردن مقدار Amount که در

اینجا برابر 1000 ریال به صورت پیش فرض در نظر گرفته شده است و شناسه خرید(درصورت درخواست تراکنش خرید شناسه دار)، کلید SendAmount باید زده شود.(یک نمونه از نحوه ی مقدار دهی شناسه خرید در حالت تکی و چند شناسه ای در ادامه ذکر شده است.)در این صورت مبلغ بر روی کارتخوان ظاهر شده و در انتظار وارد نمودن رمز می‌ماند. با وارد کردن رمز روی کارتخوان و تایید، تراکنش ارسال شده و پاسخ بازگشت داده می‌شود. در حالت چند حسابی که Tab، Share Account فعال می‌شود: در این صورت، مقادیر حساب های متفاوت درج شده و با کلیک کلید AddAmount به ListBox Amounts اضافه شده و مجموع مقادیر در Sum نمایش داده می‌شود.

نمونه ای از نحوه ی مقدار دهی شناسه خرید در حالت تک شناسه ای:



نمونه ای از نحوه ی مقدار دهی شناسه خرید در حالت چند شناسه ای:

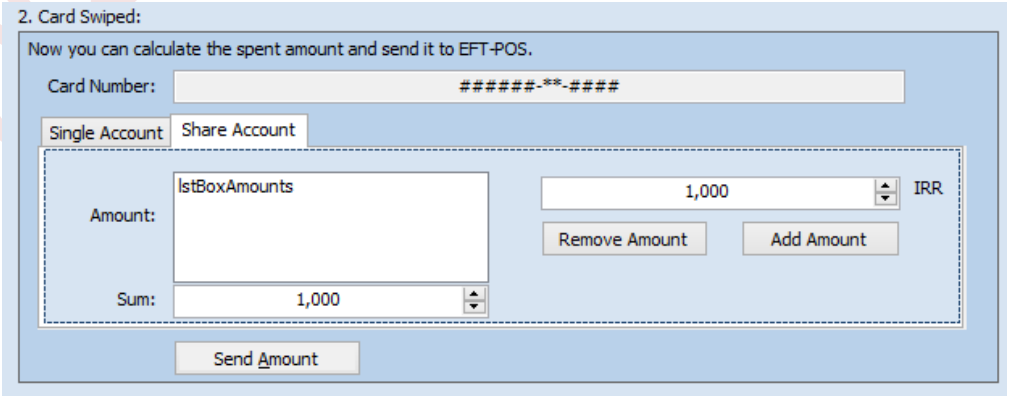


در این حالت با زدن کلید SendAmount، مبلغ 2000 ریال و رشته زیر به عنوان PurchaseID ارسال می گردد.

3670117741401030017120100000001:1000: IR710100004001047704001199; 3621100002200001221000000000000:1000: IR210100004101047771208981;

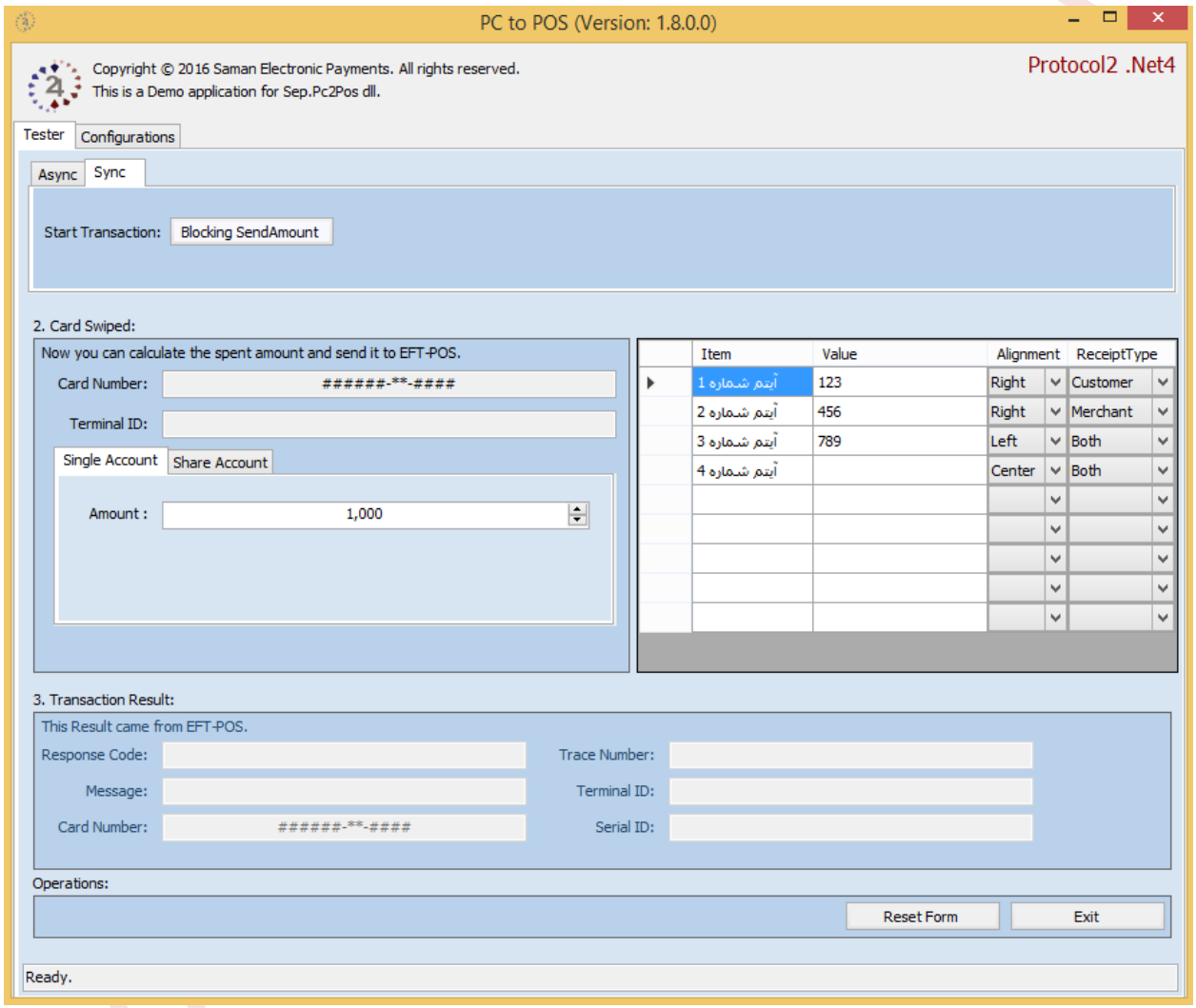
- تراکنش چند شناسه ای در این مرحله از سمت Pc تک حسابی بوده و تسهیم به صورت آفلاین سمت سوئیچ صورت می گیرد.

در مدل چندحسابی:



در این صورت، مقادیر حساب های متفاوت درج شده و با کلیک کلید AddAmount به ListBox Amounts اضافه شده و مجموع مقادیر در Sum نمایش داده می‌شود.

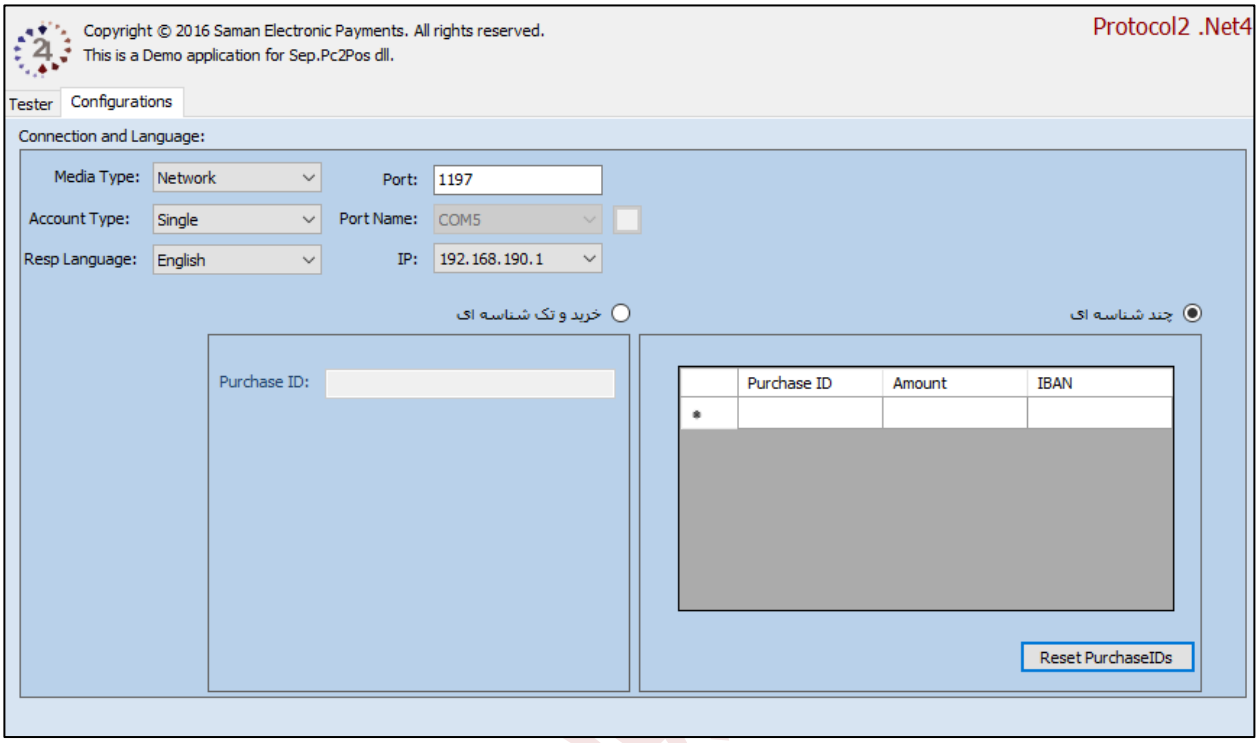
در بخش Sync :



کافی است مبلغ و اطلاعات داده اضافی وارد شود و سپس کلید Blocking SendAmount فشرده شود.

در صورت درخواست تراکنش شناسه دار تمامی مراحل ورود شناسه همانند تراکنش Async می باشد.

در بخش Configuration:



موارد زیر قبل از انجام عملیات باید انتخاب شود:

Media Type: ارتباط از طریق Network و یا Com (RS232 و یا Usb).

Resp Language: زبان پاسخ های دریافتی از پوز که می‌تواند فارسی و یا انگلیسی باشد.

Account Type: وضعیت تسهیم به صورت تک حسابی و یا چند حسابی.

Port Name: در صورتی که Media Type برابر Com انتخاب شود فعال می‌شود و پورت مورد نظر باید انتخاب گردد.

Port: در صورتی که Media Type برابر با Network انتخاب شود فعال می‌شود و نشان دهنده پورت TCP است که به صورت پیش فرض برابر با 1197 در نظر گرفته شده است. حداقل باید دارای 4 رقم باشد.

IP: نشان دهنده IP دستگاه ارسال کننده‌ی تراکنش یا صندوق می‌باشد.

Purchase ID:این فیلد به صورت پیش فرض خالی می باشد که در صورت خرید معمولی خالی بوده و درصورت خرید شناسه دار تکی با 30 رقم شناسه و در حالت خرید شناسه دار چند شناسه ای با مقدار بیشتر از 30 رقم به صورت رشته ی ذکر شده در بالا ، پر می شود.

5.3 ایجاد تراکنش Async

جهت استفاده کافی است مراحل ذیل را اجرا نمایید: (مثال ها با استفاده از زبان C# نوشته شده است.)

ایجاد یک نمونه از کلاس PosClientBase

```
if (_PosClient == null)
{
   _PosClient = new PosClientBase();
 }
```

- Assign نمودن Event ها

```
_PosClient.CardSwiped += _PosClient_CardSwiped;
_PosClient.PosResultReceived += _PosClient_PosResultReceived;
_PosClient.ErrorReceived += _PosClient_ErrorReceived;
```

- جهت ست کردن PortName و AccountType و ListenPort و IP و Language متد Init استفاده میشود که متناسب با نوع Media انتخاب شده یکی از overload های متد قابل استفاده می‌باشد.
  - یرای کار با Network از متد زیر استفاده می‌شود:

```
if (_MediaType == MediaType.LAN)
     PosClient.Init(_AccountType, lang, port, cmbIP.SelectedItem.ToString());
```

  - برای کار با Com از متد زیر استفاده می‌شود:

```
else if (_MediaType == MediaType.RS232)
      _PosClient.Init(_AccountType, lang, portName);
```

- در ابتدا برنامه باید آماده دریافت اطلاعات از سمت درستگاه کارتخوان بشود. بدین جهت بعد از ست کردن Property های ذکر شده متد WaitForPos باید صدا زده شود.

```
_PosClient.WaitForPos();
```

- مشخص نمودن Event های مربوط به PosClient (3 Event تعریف شده است)
  - **PosClient_CardSwiped**: بعد از اینکه کارت دارنده‌کارت کشیده شد شماره کارت را برای رایانه ارسال می گردد. . با استفاده از Property های CardSwipedEventArgs می‌توانید، CardNumber(شماره کارت)،

TerminalId (شماره ترمینال) و AuthorizationId (شماره تراکنش) را به دست بیاورید. شماره کارت لطفا به صورت Mask شده نمایش داده شود.

- **PosClient_ErrorReceived**: اگر فرایند توسط کاربر بر روی کارتخوان لغو و یا در اطلاعات ورودی و ارتباط پوز با برنامه به خطایی رخ دهد، نتیجه در این رخداد ارسال می‌گردد. با استفاده از Property های ErrorReceivedEventArgs می‌توانید، ErrorCode (کد خطا)، ErrorDescription (شرح خطا) را به دست بیاورید.

- **PosClient_PosResultReceived**: وقتی تراکنش تا انتها انجام شد و جواب از کارتخوان به برنامه برگشت داده شد، با استفاده از Property های PosResultEventArgs می‌توانید، IsSuccessful (تراکنش در صورتی موفق است که این فیلد برابر True و کد بازگشتی برابر 00 باشد)، ResponseCode (کد بازگشتی)، ResponseDescription (شرح کد بازگشتی) ، PaidAmount (مبلغ پرداخت شده) ، TraceNumber (شماره پیگیری) ، SerialId (شماره سریال) ، TxnDate (تاریخ تراکنش)، TerminalID (شماره ترمینال) و ReturnPAN (شماره کارت) را به دست بیاورید. شماره کارت لطفا به صورت Mask شده نمایش داده شود.

نتیجه‌ی آن در این event ارسال می‌گردد. در صورتی تراکنش موفق می‌باشد که 3 شرط زیر برقرار باشند:

- ☐ e.IsSuccessful مقدار true باشد.
- ☐ e.ResponseCode = "00" باشد.
- ☐ e.PaidAmount == txnAmount (مبلغ اولیه با پرداخت شده برابر باشد).

```
        private void _PosClient_CardSwiped(PosClient sender, CardSwipedEventArgs e)
                {
            string maskedPAN = GetMasked(e.CardNumber);
            _AuthorizationId = e.AuthorizationId;
            Debug.WriteLine("ThreadID4: " + Thread.CurrentThread.ManagedThreadId);
            if (txtPAN.InvokeRequired)
                this.Invoke(new MethodInvoker(() =>
                {
                    GetReady();
                    txtPAN.Text = maskedPAN;
                    txtTerminalID1.Text = e.TerminalId;
                    progressBar.Visible = false;
                    txtStatus.Text = string.Format("Card swiped with \"{0}\" card number,
and Authorization ID is: \"{1}\"", GetMasked(e.CardNumber), _AuthorizationId);
                    pnlSwipeCard.Enabled = true;
                    btnSendAmount.Focus();
                }));
            else
            {

                GetReady();
                txtPAN.Text = maskedPAN;
```

```
                txtTerminalID1.Text = e.TerminalId;
                progressBar.Visible = false;
                txtStatus.Text = string.Format("Card swiped with \"{0}\" card number, and
Authorization ID is: \"{1}\"", GetMasked(e.CardNumber), _AuthorizationId);
                pnlSwipeCard.Enabled = true;
                btnSendAmount.Focus();
                    }
            }

private void _PosClient_PosResultReceived(IPCPOS sender, PosResultEventArgs e)
        {
            if (e.IsSuccessful && e.ResponseCode == "00" &&
(string.Equals(e.PaidAmount.Trim(), numSumAmount.Value.ToString()) && _AccountType ==
AccountType.Share) || (string.Equals(e.PaidAmount.Trim(), numAmount.Value.ToString()) &&
_AccountType == AccountType.Single))
            {
                picResult.Image = Resource.Successful;
                if (this.InvokeRequired)
                    this.Invoke(new MethodInvoker(() =>
                    {
                        txtStatus.Text = string.Format("Successful Transaction. Trace#:
\"{0}\", Authorization ID: \"{1}\"", e.TraceNumber, e.AuthorizationId);
                    }));
                else
                {
                    txtStatus.Text = string.Format("Successful Transaction. Trace#:
\"{0}\", Authorization ID: \"{1}\"", e.TraceNumber, e.AuthorizationId);
                }

            }
            else
            {
                picResult.Image = Resource.Error;
                if (this.InvokeRequired)
                    this.Invoke(new MethodInvoker(() =>
                    {
                        txtStatus.Text = string.Format("Transaction Error. Serial ID:
\"{0}\"", e.SerialId);
                    }));
                else
                {
                    txtStatus.Text = string.Format("Transaction Error. Serial ID:
\"{0}\"", e.SerialId);
                }

            }

            if (this.InvokeRequired)
                this.Invoke(new MethodInvoker(() =>
                {
                    txtResponseCode.Text = e.ResponseCode;
                    txtMessage.Text = e.ResponseDescription;
                    txtPAN2.Text = GetMasked(e.ReturnPAN);
                    txtTraceNumber.Text = e.TraceNumber;
                    txtTerminalID.Text = e.TerminalID;
                    txtSersialId.Text = e.SerialId;
```

```
                pnlResult.Enabled = true;
                progressBar.Visible = false;
                pnlStart.Enabled = true;
                btnStart.Focus();
            }));
        else
        {
            txtResponseCode.Text = e.ResponseCode;
            txtMessage.Text = e.ResponseDescription;
            txtPAN2.Text = GetMasked(e.ReturnPAN);
            txtTraceNumber.Text = e.TraceNumber;
            txtTerminalID.Text = e.TerminalID;
            txtSersialId.Text = e.SerialId;
            pnlResult.Enabled = true;
            progressBar.Visible = false;
            pnlStart.Enabled = true;
            btnStart.Focus();
        }
    }


    private void _PosClient_ErrorReceived(PosClient sender, ErrorReceivedEventArgs e)
        {
            if (this.InvokeRequired)
                this.Invoke(new MethodInvoker(() =>
                {
                    pnlResult.Enabled = true;
                    txtPAN2.Text = "######-**-####";
                    txtTraceNumber.Clear();
                    txtTerminalID.Clear();
                    txtSersialId.Clear();
                    txtStatus.Text = string.Format("Transaction failed. Error
Code: \"{0}\", Error Message: \"{1}\"", e.ErrorCode, e.ErrorDescription);
                    txtResponseCode.Text = e.ErrorCode;
                    txtMessage.Text = e.ErrorDescription;
                    picResult.Image = Resource.Failed;
                    pnlStart.Enabled = true;
                    pnlSwipeCard.Enabled = false;
                    progressBar.Visible = false;
                    btnStart.Focus();
                }));
            else
            {
                pnlResult.Enabled = true;
                txtPAN2.Text = "######-**-####";
                txtTraceNumber.Clear();
                txtTerminalID.Clear();
                txtSersialId.Clear();
                txtStatus.Text = string.Format("Transaction failed. Error Code:
\"{0}\", Error Message: \"{1}\"", e.ErrorCode, e.ErrorDescription);
                txtResponseCode.Text = e.ErrorCode;
                txtMessage.Text = e.ErrorDescription;
                picResult.Image = Resource.Failed;
                pnlStart.Enabled = true;
                progressBar.Visible = false;
                pnlSwipeCard.Enabled = false;
                btnStart.Focus();
```

```
            }
        }
```

- صدا کردن متد PosClient.SendAmount (بعد از اینکه کارت کشیده شد و مبلغ خرید وارد شد باید کلید SendAmount زده شود)

دقت شود در انتهای لیست ارسالی در مدل تسهیم چند حسابی مجموع مبالغ باید اضافه گردد.

```
    private void btnSendAmount_Click(object sender, EventArgs e)
        {

             List<string> lstAmnts = new List<string>();
            if (_AccountType == AccountType.Single)
            {
                lstAmnts.Add(numAmount.Value.ToString());
                _PosClient.SendAmount(lstAmnts, _AuthorizationId, prntItemResults, 0,
PurchaseID);
            }

            else
            {
                lstAmnts.AddRange(lstBoxAmounts.Items.Cast<string>().ToList());
                lstAmnts.Add(numSumAmount.Value.ToString());
                _PosClient.SendAmount(lstAmnts, _AuthorizationId, prntItemResults, 0,
PurchaseID);
            }


        }

_posClient.SendAmountMultiPurchase (SumAmn, _AuthorizationId, prntItemResults, 0,
PurchaseID);
```

SumAmn: همان طور که قبلا اشاره شد این فیلد برابر مجموع مبالغ شناسه ها می باشد.

PurchaseID: همان طور که قبلا اشاره شد این فیلد برابر رشته ی ذکر شده با فرمتی که در بخش فرآیند کلی برنامه آمده است می باشد.

5.4 ایجاد تراکنشSync

با استفاده از متد `PosResultEventArgs BlockingTransaction(List<string> amount, List<PrintItemResult> printList, int timeout, string PurchaseID=null);` می توان تراکنش را به صورت Sync تعریف نمود:

در حالت چند شناسه ای :

```
PosResultEventArgs          BlockingTransactionMultiPurchase(string          SumAmn,
          List<PrintItemResult> printList, int timeout, string PurchaseID = null)
```

ایجاد یک نمونه از کلاس PosClientBase

```
    if (_PosClient == null)
    {
          _PosClient = new PosClientBase();
    }
```

جهت ست کردن PortName و AccountType و ListenPort و IP و Language متد Init استفاده میشود که متناسب با نوع Media انتخاب شده یکی از overload های متد قابل استفاده می‌باشد.

- برای کار با Network از متد زیر استفاده می‌شود:

```
   if (_MediaType == MediaType.LAN)
        PosClient.Init(_AccountType, lang, port, cmbIP.SelectedItem.ToString());
```

برای کار با Com از متد زیر استفاده می‌شود:

```
   else if (_MediaType == MediaType.RS232)
         _PosClient.Init(_AccountType, lang, portName);
```

صدا زدن متد تراکنش Sync:

```
private void btnBlockingSendAmount_Click(object sender, EventArgs e)
        {
            List<string> lstAmnts = new List<string>();
            List<PrintItemResult> prntItemResults = new List<PrintItemResult>();

            progressBar.Visible = true;
            foreach (DataGridViewRow dr in grdViewPrintItem.Rows)
            {
                var item = dr.Cells["Item"].Value == null ? "" :
dr.Cells["Item"].Value.ToString();
                var value = dr.Cells["Value"].Value == null ? "" :
dr.Cells["Value"].Value.ToString();
                var alignment = dr.Cells["Alignment"].Value == null ? "" :
dr.Cells["Alignment"].Value.ToString();
                var receiptType = dr.Cells["ReceiptType"].Value == null ? "" :
dr.Cells["ReceiptType"].Value.ToString();
                if (alignment == "")
                    alignment = "Center";
                if (receiptType == "")
                    receiptType = "Both";
                if (item != "")
                {
                    prntItemResults.Add(new PrintItemResult(item, value,
                        (Alignment)Enum.Parse(typeof(Alignment), alignment, true),
                        (ReceiptType)Enum.Parse(typeof(ReceiptType), receiptType,
true)));
                }
                else
                    break;
```

```
            }
            tabMain.Enabled = false;
            PosResultEventArgs posResult = null;
            StartTransaction();
            if (_AccountType == AccountType.Single)
            {
                lstAmnts.Add(numAmount.Value.ToString(CultureInfo.InvariantCulture));
                posResult = _PosClient.BlockingTransaction(lstAmnts, prntItemResults, 60,
PurchaseID);
            }

            else
            {
                foreach (var item in lstBoxAmounts.Items)
                {
                    lstAmnts.Add(item.ToString());
                }
                lstAmnts.Add(numSumAmount.Value.ToString(CultureInfo.InvariantCulture));
                posResult = _PosClient.BlockingTransaction((lstAmnts, prntItemResults, 0,
PurchaseID);
            }
            tabMain.Enabled = true;

            pnlSwipeCard.Enabled = false;
            progressBar.Visible = false;

            ResultReceived(posResult);

        }        }
```

در حالت چند شناسه ای :

```
posResult = _posClient.BlockingTransactionMultiPurchase (SumAmn, prntItemResults,
                                                         60, PurchaseID);
```

پرداخت الکترونیک سامان

6. راهنما

- پاسخ زبان نتیجه تراکنش

```
public enum ResponseLanguage
 {
     Persian = 0,
     English = 1
 }
```

- نوع ارتباط با رایانه

```
 public enum MediaType
 {
     Com = 1,
     Network = 2,
 }
```

- نوع تسهیم حساب

```
 public enum AccountType
 {
     Single = 0,
     Share = 1
 }
```

- نوع Alignment متن چاپ شده روی پرینت کارتخوان

```
 public enum Alignment
 {
     Right = 0,
     Left = 1,
     Center = 2
 }
```

- نوع رسید متن چاپ شده روی پرینت کارتخوان

```
 public enum ReceiptType
 {
     Customer = 0,
     Merchant = 1,
     Both = 2
 }
```

- جهت چاپ داده اضافی روی کارتخوان استفاده می‌شود.

```
 public class PrintItemResult
 {
         public string Item { get; set; }
         public string Value { get; set; }
         public Alignment Alignment { get; set; }
   public ReceiptType ReceiptType { get; set; }
  }
```

در این کلاس:

- ☐ Item نشان دهنده عبارتی مانند شماره قبض می‌باشد.
- ☐ Value نشان دهنده عبارتی مانند 5144241806235 می باشد.

پرداخت الکترونیک شاهان

☐ Alignment نشان دهنده قرارگیری متن در هنگام چاپ رسید می باشد.

```
public enum Alignment
     {
         Right = 0,
         Left = 1,
         Center = 2
     }
```

☐ ReceiptType نشان دهنده نوع رسیدی است که اطلاعات روی آن چاپ می‌شود:

```
public enum ReceiptType
     {
         Customer = 0,
         Merchant = 1,
         Both = 2
     }
```

وقتی Item مقدار داشته باشد و Value خالی باشد می توانید از نوع Center استفاده نمایید. در این صورت مقدار وارد شده در Item در مرکز رسید چاپ می‌شود.

در صورتی که نیاز به چاپ داده اضافی روی پوز ندارید، مقدار این ورودی در این متد باید NULL داده شود.

- نمایش شماره کارت : شماره کارت به صورت کامل نباید نشان داده شود و بدین منظور از تابع زیر استفاده شده است

```
private string GetMasked(string pan)
        {
            if (pan.Length == 16)
            {
                return pan.Substring(0, 6) + "-**-" + pan.Substring(pan.Length - 4, 4);
            }
            else if (pan.Length == 19)
            {
                if (pan.Substring(16, 3) != "000")
                    return pan.Substring(0, 6) + "-**-" + pan.Substring(pan.Length - 4, 4);
                else
                    return pan.Substring(0, 6) + "-**-" + pan.Substring(pan.Length - 7, 4);
            }
            else
            {
                return pan;
            }
        }
```

مقادیر کدهای پاسخ (Response Code)

کد	پیام
تراکنش با موفقیت انجام پذیرفت	00
کارت کشیده شد	1
مبلغ تراکنش نمی تواند از "حداقل مبلغ" کوچکتر باشد	2
عدم ارتباط با دستگاه	3
اطلاعات نامعتبر	4
صفر ریال بدهی	5
خطا در دریافت اطلاعات	6
عدم دسترسی به این عملیات	7
تراکنش یافت نشد	8
ترمینال نامعتبر	9
تراکنش نامعتبر	12
مبلغ نامعتبر	13 / 79
خطا در مقداردهی	14
پاسخ نامعتبر	20
خطا در تراکنش	
خطا در قالب اطلاعات	30
تاریخ انقضای کارت سپری شده است	33
تعداد دفعات ورود رمز غلط بیش از حدمجاز است	38

موجودی کافی نمی باشد	51
رمز کارت نا معتبر است	55
انجام تراکنش مربوطه توسط دارنده ی کارت مجاز نمی باشد	57
انجام تراکنش مربوطه توسط پایانه ی انجام دهنده مجاز نمی باشد	58
مبلغ تراکنش بیش از حد مجاز می باشد	61
عدم دریافت پاسخ در زمان مناسب	68
تعداد دفعات ورود رمزغلط بیش از حدمجاز است	69 / 75
کارت غیرفعال میباشد	78
عدم پاسخ از سوی صادرکننده ی کارت	80 / 84 / 91
مبالغ متفاوت	92
خطای نامشخص	**96**
عدم ارتباط با مرکز	97
لغو عملیات توسط کاربر	**98**
عدم دریافت پاسخ در زمان مناسب در کارتخوان	92