



ООО “АСТРОН ЛТД”

**Пакет прикладных программ
PowerPOS Версия 5.0**

**Создание драйверов
банковских терминалов**

Минск – 2015

ООО "Астрон ЛТД"
220113, г. Минск, ул. Мележа 5, корп. 2 оф. 1201
тел. +375 (17) 392-56-00, 01, 02, 03, 04, 05
факс +375 (17) 392-56-06
<http://www.astron.by>

© Copyright Astron Ltd., 2014. All right reserved.

Данная публикация или ее часть не могут быть воспроизведены в любой форме без предварительного письменного разрешения фирмы Астрон ЛТД.

Содержание:

Описание драйвера банковского терминала	4
Интерфейс IBankTerminal	4
Свойства	4
Методы.....	5
События	6
Делегаты	6
Класс TransactionInfoParameter.....	6
Конструкторы.....	6
Свойства	7
Перечисление TransactionType	7
Значения	7
Интерфейс IDevice.....	7
Свойства	7
Методы.....	8
Класс Money	8
Конструкторы.....	8
Свойства	8
Класс MessageEventArgs	9
Конструкторы.....	9
Свойства	9
Класс PaymentParameters.....	9
Конструкторы.....	9
Свойства	9
Класс Port.....	9
Свойства	10
Перечисление PortType	10
Значения	10
Класс Parameter	10
Конструкторы.....	10
Свойства	11
Пример реализации.....	11

Описание драйвера банковского терминала

Драйвер банковского терминала должен представлять собой сборку для Microsoft .Net 2.0.

Класс драйвера должен реализовать интерфейс

SoftMarket.Devices.BankTerminals.IBankTerminal, находящийся в сборке DriverGlobals.dll.

Сборку, реализующую драйвер, необходимо положить в каталог приложения (по умолчанию C:\Program Files\Astron\PowerPOS\v5).

Все исключения, генерируемые драйвером должны быть SoftMarket.Devices.DeviceException (сборка DriverGlobals.dll), или быть унаследованы от него.

В проект драйвера нужно включить сборки DriverGlobals.dll и Globals.dll.

Интерфейс IBankTerminal

Сборка: DriverGlobals.dll.

Полное имя: SoftMarket.Devices.BankTerminals.IBankTerminal.

Описание: Интерфейс банковского терминала.

```
public interface IBankTerminal : IDevice
{
    string BankName {get;}
    bool IsOfflineMode{get;}

    bool CanExecuteTransaction(TransactionType transactionType);
    void CheckExtendedParameters(TransactionInfoParameter[] extParameters,
TransactionType transactionType);
    PaymentParameters ExecuteTransaction(Money summa, PaymentParameters
paymentParameters, TransactionType transactionType, TransactionInfoParameter[]
extParameters, decimal currencySum, int currencyId, byte decimalPlace);
    TransactionInfoParameter[] GetExtendedParameters(TransactionType
transactionType);
    void Init(int sareaId, int systemId, int currencyId, int deviceId);
    void OfflineAuthorization(string authorizationCode);
    void Stop();
    void VirtualPort(string message);
    void OpenWorkDay();
    void CloseWorkDay();
    object ExecuteUserCommand(int commandId, object commandData);

    event StatusMessageEventHandler StatusMessage;
    event PrintMessageEventHandler PrintMessage;
}

public delegate void StatusMessageEventHandler(object sender, MessageEventArgs
e);
public delegate void PrintMessageEventHandler(object sender, MessageEventArgs
e);
```

Свойства

Свойство	Описание
<code>string</code> BankName	Название банка. Используется, если к кассе подключено более одного терминала. Желательно возвращать значение, заданное через IDevice.Parameters.
<code>bool</code> IsOfflineMode	Признак того, что терминал работает в автономном режиме, то есть, не подключен к кассе.

Методы

Метод	Описание
<code>PaymentParameters</code> <code>ExecuteTransaction (Money summa, PaymentParameters paymentParameters, TransactionType transactionType, TransactionInfoParameter[] extParameters, decimal currencySum, int currencyId, byte decimalPlace)</code>	<p>Производит указанный тип операции: оплата, возврат или отмена. Результат операции возвращается в <code>PaymentParameters</code>. Если операция не удалась, или возникли ошибки протокола, то необходимо сгенерировать исключение <code>SoftMarket.Devices.DeviceException</code> с описанием возникшей проблемы. Оплата производится в разных валютах, поэтому передается сумма в базовой валюте – <code>summa</code> и сумма в валюте операции – <code>currencySum</code>. Также передается идентификатор валюты и количество знаков после запятой для валюты в которой нужно провести операцию.</p>
<code>bool CanExecuteTransaction (TransactionType transactionType)</code>	<p>Проверяет возможность выполнения терминалом определенного типа операции. Возвращает <code>true</code>, если терминал поддерживает выполнение операции указанного типа, иначе <code>false</code>.</p>
<code>void CheckExtendedParameters (TransactionInfoParameter[] extParameters, TransactionType transactionType)</code>	<p>Проверяет указанные параметры на корректность. Если какой-либо из параметров некорректен, то необходимо сгенерировать исключение <code>SoftMarket.Devices.DeviceException</code> с соответствующим описанием.</p>
<code>TransactionInfoParameter[] GetExtendedParameters (TransactionType transactionType)</code>	<p>Возвращает список дополнительных параметров, необходимых для выполнения указанной операции.</p>
<code>void Stop()</code>	<p>Прерывает выполнение транзакции. Вызывается в потоке отличном от потока метода <code>ExecuteTransaction</code>. При вызове этого метода, если драйвер находился в процессе выполнения транзакции, метод <code>ExecuteTransaction</code> должен сгенерировать исключение <code>SoftMarket.Devices.DeviceException</code> («Операция прервана пользователем»).</p>
<code>void OfflineAuthorization (string authorizationCode)</code>	<p>Подтверждает транзакцию в автономном режиме. Используется только, если терминал работает в автономном режиме. Вызывается в потоке отличном от потока метода <code>ExecuteTransaction</code>. При вызове этого метода, если драйвер находился в процессе выполнения транзакции, метод <code>ExecuteTransaction</code> должен корректно завершить работу и передать результат через <code>PaymentParameters</code>.</p>
<code>void VirtualPort (string message)</code>	<p>Вводит данные, поступившие в виртуальный порт.</p>
<code>void Init (int sareaId, int</code>	<p>Инициализирует терминал начальными</p>

<code>systemId, int currencyId, int deviceId)</code>	параметрами: идентификатор торговой площадки, идентификатор системы, идентификатор валюты, идентификатор устройства.
<code>void OpenWorkDay()</code>	Открывает рабочий день по терминалу.
<code>void CloseWorkDay()</code>	Закрывает рабочий день по терминалу.
<code>object ExecuteUserCommand(int commandId, object commandData)</code>	Выполняет пользовательскую команду. Передается код команды и данные для ее выполнения. Возвращает результат выполнения команды.

События

Событие	Описание
<code>StatusMessageEventHandler</code> <code>StatusMessage</code>	Генерируется драйвером, если необходимо отобразить информацию на мониторе кассы.
<code>PrintMessageEventHandler</code> <code>PrintMessage</code>	Генерируется драйвером, если необходимо напечатать информацию на принтере кассы.

Делегаты

Делегат	Описание
<code>void</code> <code>StatusMessageEventHandler(object sender, EventArgs e)</code>	Делегат для события <code>StatusMessage</code> .
<code>void</code> <code>PrintMessageEventHandler(object sender, EventArgs e)</code>	Делегат для события <code>PrintMessage</code> .

Класс TransactionInfoParameter

Сборка: DriverGlobals.dll.

Полное имя: SoftMarket.Devices.BankTerminals.TransactionInfoParameter.

Описание: Класс для хранения дополнительных параметров транзакции.

Конструкторы

Конструктор	Описание
<code>TransactionInfoParameter(int id, string description, bool needFill)</code>	<code>id</code> – идентификатор параметра. <code>description</code> – описание параметра. <code>needFill</code> – признак обязательного заполнения.
<code>TransactionInfoParameter(int id, string description, bool needFill, bool isPassword)</code>	<code>id</code> – идентификатор параметра. <code>description</code> – описание параметра. <code>needFill</code> – признак обязательного заполнения. <code>isPassword</code> – признак того, что параметр является паролем.

Свойства

Свойство	Описание
<code>string Description{get;}</code>	Описание параметра.
<code>int Id{get;}</code>	Идентификатор параметра.
<code>bool IsPassword{get;}</code>	Признак того, что параметр является паролем.
<code>bool NeedFill{get;}</code>	Признак обязательного заполнения.
<code>string Value{get;set;}</code>	Значение параметра.

Перечисление *TransactionType*

Значения

Значение	Описание
Payment	Операция оплаты.
Refund	Возврат суммы.
Cancel	Отмена существующей транзакции.
BonusPayment	Оплата бонусами.
BonusRefund	Возврат бонусов.
BonusCancel	Отмена существующей бонусной транзакции.

Интерфейс IDevice

Сборка: DriverGlobals.dll.

Полное имя: SoftMarket.Devices.IDevice.

Описание: Общий интерфейс устройств.

```
public interface IDevice
{
    String DeviceName{get;}
    String DeviceFriendlyName{get;}
    Port DefaultPort{get;}
    PortType[] PortTypes{get;}
    Parameter[] Parameters{get;set;}
    bool IsOpened{get;}

    void Open(Port port);
    void Close();
    Parameter[] GetDefaultParameters(PortType portType);
    Parameter[] GetParameters(PortType portType);
    bool IsPortSupported(PortType portType);
}
```

Свойства

Свойство	Описание
<code>String DeviceName</code>	Уникальный идентификатор устройства может совпадать с DeviceFriendlyName.

<code>String</code> <code>DeviceFriendlyName</code>	Название устройства, отображаемое при настройке кассы. Обычно название протокола и его версия.
<code>Port</code> <code>DefaultPort</code>	Порт устройства по умолчанию.
<code>PortType[]</code> <code>PortTypes</code>	Типы портов, поддерживаемые драйвером.
<code>Parameter[]</code> <code>Parameters</code>	Дополнительные параметры устройства, которые редактируются пользователем в процессе настройки драйвера. Например: название банка, скорость СОМ порта и т. д.
<code>bool</code> <code>IsOpened</code>	Признак того, что соединение с устройством установлено.

Методы

Метод	Описание
<code>void</code> <code>Open</code> (<code>Port</code> <code>port</code>)	Устанавливает соединение с устройством. При ошибке соединения с устройством необходимо сгенерировать исключение <code>SoftMarket.Devices.DeviceException</code> с соответствующим описанием.
<code>void</code> <code>Close</code> ()	Разрывает соединение с устройством.
<code>bool</code> <code>IsPortSupported</code> (<code>PortType</code> <code>portType</code>)	Проверяет поддержку терминалом указанного типа порта.
<code>Parameter[]</code> <code>GetParameters</code> (<code>PortType</code> <code>portType</code>)	Возвращает дополнительные параметры устройства для указанного порта.
<code>Parameter[]</code> <code>GetDefaultParameters</code> (<code>PortType</code> <code>portType</code>)	Возвращает дополнительные параметры устройства, заданные по умолчанию.

Класс Money

Сборка: `Globals.dll`.

Полное имя: `SoftMarket.Globals.Units.Money`.

Описание: Класс реализующий работу с деньгами.

Конструкторы

Конструктор	Описание
<code>Money</code> (<code>int</code> <code>amount</code>)	<code>Amount</code> – сумма в копейках.
<code>Money</code> (<code>decimal</code> <code>amount</code>)	<code>Amount</code> – сумма в копейках.

Свойства

Свойство	Описание
<code>int</code> <code>Amount</code> { <code>get</code> ;}	Сумма в копейках.
<code>decimal</code> <code>DecimalAmount</code> { <code>get</code> ;}	Сумма в копейках.

Класс MessageEventArgs

Сборка: DriverGlobals.dll.

Полное имя: SoftMarket.Devices.BankTerminals.MessageEventArgs.

Описание: Класс для событий драйверов банковских терминалов.

Конструкторы

Конструктор	Описание
MessageEventArgs(<code>string</code> message)	message – текстовая информация.

Свойства

Свойство	Описание
<code>string</code> Message{get;}	Текстовая информация.

Класс PaymentParameters

Сборка: DriverGlobals.dll.

Полное имя: SoftMarket.Devices.BankTerminals.PaymentParameters.

Описание: Класс для хранения результатов транзакции банковского терминала.

Конструкторы

Конструктор	Описание
PaymentParameters(<code>Money</code> summa, <code>string</code> acode, <code>string</code> rcode, <code>string</code> cardNumber, <code>string</code> mfo, <code>TransactionType</code> transactionType)	summa – сумма транзакции acode – код авторизации rcode – код ответа от хоста cardNumber – номер карточки mfo – МФО банка transactionType – тип транзакции

Свойства

Свойство	Описание
<code>string</code> ACode{get;}	Код авторизации.
<code>string</code> RCode{get;}	Код ответа от хоста.
<code>string</code> CardNumber{get;}	Номер карточки.
<code>string</code> TransactionInfo{get; set;}	Дополнительная информация о транзакции.
<code>string</code> MFO{get;}	МФО банка.
<code>Money</code> Summa{get;}	Сумма транзакции.
<code>string</code> TransactionId{get; set;}	Идентификатор транзакции.
<code>TransactionType</code> transactionType{get;}	Тип транзакции.

Класс Port

Сборка: DriverGlobals.dll.

Полное имя: SoftMarket.Devices.Port.

Описание: Класс для хранения данных порта устройства (LPT, COM, ETHERNET).

Свойства

Свойство	Описание
<code>int</code> Number{get;}	Номер порта.
<code>PortType</code> Type{get;}	Тип порта.
<code>string</code> IPAddress{get;}	IP адрес.

Перечисление PortType

Значения

Значение	Описание
COM	COM порта.
LPT	LPT порта.
KEYBOARD	Клавиатура
ETHERNET	Сеть Ethernet
VIRTUAL	Виртуального порта.
USB	USB порта.

Класс Parameter

Сборка: Globals.dll.

Полное имя: SoftMarket.Globals.Parameter.

Описание: Класс для работы с дополнительными настройками устройств.

Конструкторы

Конструктор	Описание
<code>Parameter(int key, int intValue, int minValue, int maxValue, string description)</code>	key - уникальный идентификатор параметра intValue - значение параметра minValue - минимальное значение параметра maxValue - максимальное значение параметра description - описание параметра
<code>Parameter(int key, int intValue, int minValue, int maxValue, string description, bool isSecure)</code>	key - уникальный идентификатор параметра intValue - значение параметра minValue - минимальное значение параметра maxValue - максимальное значение параметра description - описание параметра isSecure - отображать значение звездочками
<code>Parameter(int key, string stringValue, int minValue, int maxValue, string description)</code>	key - уникальный идентификатор параметра stringValue - значение параметра minValue - минимальная длина параметра maxValue - максимальная длина параметра description - описание параметра
<code>Parameter(int key, string stringValue, int minValue, int maxValue, string description, bool isSecure)</code>	key - уникальный идентификатор параметра stringValue - значение параметра minValue - минимальная длина параметра maxValue - максимальная длина параметра description - описание параметра isSecure - отображать значение звездочками
<code>Parameter(int key, int intValue, int[] intEnumValues,</code>	key - уникальный идентификатор параметра intValue - значение параметра

<code>string</code> description)	intEnumValues - возможные значения параметра description - описание параметра
Parameter(int key, string stringValue, string[] stringEnumValues, string description)	key - уникальный идентификатор параметра stringValue - значение параметра stringEnumValues - возможные значения параметра description - описание параметра
Parameter(int key, bool boolValue)	key - уникальный идентификатор параметра boolValue - значение параметра

Свойства

Свойство	Описание
<code>int</code> Key{get;}	Идентификатор параметра.
<code>int</code> IntValue{get;set;}	Целочисленное значение параметра.
<code>string</code> StringValue{get;set;}	Строковое значение параметра.
<code>bool</code> BoolVal{get;set;}	Булево значение параметра.
<code>string</code> Description{get;}	Описание параметра.
<code>int</code> MaxValue{get;}	Максимальное значение/длина параметра.
<code>int</code> MinValue{get;}	Минимальное значения/длина параметра.
<code>bool</code> IsSecure{get;}	Признак отображения значения звездочками.
<code>int[]</code> IntEnumValues{get;}	Список возможных целочисленных значений параметра.
<code>string[]</code> StringEnumValues{get;}	Список возможных строковых значений параметра.

Пример реализации

```
using System;
using System.Collections.Generic;
using System.Text;
using SoftMarket.Globals.Units;
using SoftMarket.Globals;
using System.Threading;
using SoftMarket.Devices.Printers;

namespace SoftMarket.Devices.BankTerminals
{
    public class BankTerminalEmu : IBankTerminal
    {
        private string bankName = "";
        protected bool isStop = false;
        protected string mfo = string.Empty;
        protected BankTerminalTransactionMode transactionMode =
BankTerminalTransactionMode.Sales;
        protected string yes = BankTerminalStrings.GetString((int)Message.Yes);
        protected string no = BankTerminalStrings.GetString((int)Message.No);
        private bool isOpen;

        #region IBankTerminal Members

        public object ExecuteUserCommand(int commandId, object commandData)
        {
            return null;
        }

        public string BankName
        {
            get
            {
                return bankName;
            }
        }
    }
}
```

```

    }
}

public SoftMarket.Globals.Parameter[] Parameters
{
    get
    {
        List<Parameter> parameters = new List<Parameter>();

        parameters.Add(new Parameter((int)SettingsParameter.BankName, bankName, 0,
50, BankTerminalStrings.GetString((int)Message.BankNameDescription)));
        parameters.Add(new Parameter((int)SettingsParameter.MFO, mfo, 0, 50,
BankTerminalStrings.GetString((int)Message.MFODescription)));
        parameters.Add(new Parameter((int)SettingsParameter.TransactionMode,
BankTerminalHelper.BankTerminalTransactionModeToString(transactionMode),
BankTerminalHelper.GetBankTerminalTransactionModeDescriptions(),
BankTerminalStrings.GetString((int)Message.TransactionModeDescription)));

        return parameters.ToArray();
    }
    set
    {
        foreach(Parameter parameter in value)
        {
            switch(parameter.Key)
            {
                case (int)SettingsParameter.BankName:
                    bankName = parameter.StringValue;
                    break;
                case (int)SettingsParameter.MFO:
                    mfo = parameter.StringValue;
                    break;
                case (int)SettingsParameter.TransactionMode:
                    transactionMode =
BankTerminalHelper.StringToBankTerminalTransactionMode(parameter.StringValue);
                    break;
            }
        }
    }
}

public Port DefaultPort
{
    get
    {
        return new Port( PortType.KEYBOARD);
    }
}

public PortType[] PortTypes
{
    get
    {
        return new PortType[] { PortType.KEYBOARD };
    }
}

public bool CanExecuteTransaction(TransactionType transactionType)
{
    switch(transactionType)
    {
        case TransactionType.Payment:
        case TransactionType.Refund:
        case TransactionType.Cancel:
            return transactionMode == BankTerminalTransactionMode.Sales;
        case TransactionType.BonusPayment:
        case TransactionType.BonusRefund:
        case TransactionType.BonusCancel:
            return transactionMode == BankTerminalTransactionMode.Bonus;
        case TransactionType.ClosedLoopPayment:
        case TransactionType.ClosedLoopRefund:
    }
}

```

```

        case TransactionType.ClosedLoopCancel:
            return transactionMode == BankTerminalTransactionMode.ClosedLoop;
        case TransactionType.Ext1Payment:
        case TransactionType.Ext1Refund:
        case TransactionType.Ext1Cancel:
            return transactionMode == BankTerminalTransactionMode.Ext1;
        case TransactionType.Ext2Payment:
        case TransactionType.Ext2Refund:
        case TransactionType.Ext2Cancel:
            return transactionMode == BankTerminalTransactionMode.Ext2;
        case TransactionType.Ext3Payment:
        case TransactionType.Ext3Refund:
        case TransactionType.Ext3Cancel:
            return transactionMode == BankTerminalTransactionMode.Ext3;
        default:
            return false;
    }
}

public PaymentParameters ExecuteTransaction(Money summa, PaymentParameters
paymentParameters, TransactionType transactionType, TransactionInfoParameter[] extParameters,
decimal currencySum, int currencyCode, byte decimalPlace)
{
    if (!CanExecuteTransaction(transactionType))
        throw new
DeviceException(BankTerminalStrings.GetString((int)Message.TransactionTypeNotSupport));

    switch (transactionType)
    {
        case TransactionType.Payment:
            return RunTransaction(summa, false, transactionType);
        case TransactionType.Refund:
            return RunTransaction(summa, true, transactionType);

        case TransactionType.BonusPayment:
            return RunTransaction(summa, false, transactionType);
        case TransactionType.BonusRefund:
            return RunTransaction(summa, true, transactionType);

        case TransactionType.ClosedLoopPayment:
            return RunTransaction(summa, false, transactionType);
        case TransactionType.ClosedLoopRefund:
            return RunTransaction(summa, true, transactionType);

        case TransactionType.Ext1Payment:
            return RunTransaction(summa, false, transactionType);
        case TransactionType.Ext1Refund:
            return RunTransaction(summa, true, transactionType);

        case TransactionType.Ext2Payment:
            return RunTransaction(summa, false, transactionType);
        case TransactionType.Ext2Refund:
            return RunTransaction(summa, true, transactionType);

        case TransactionType.Ext3Payment:
            return RunTransaction(summa, false, transactionType);
        case TransactionType.Ext3Refund:
            return RunTransaction(summa, true, transactionType);

        case TransactionType.Ext1Cancel:
            if (paymentParameters.TransactionType == TransactionType.Ext1Payment)
                return RunTransaction(paymentParameters.Summa, true,
TransactionType.Ext1Refund);
            else
                throw new
DeviceException(BankTerminalStrings.GetString((int)Message.TransactionTypeNotSupport));
        case TransactionType.Ext2Cancel:
            if (paymentParameters.TransactionType == TransactionType.Ext2Payment)
                return RunTransaction(paymentParameters.Summa, true,
TransactionType.Ext2Refund);
    }
}

```

```

        else
            throw new
DeviceException(BankTerminalStrings.GetString((int)Message.TransactionTypeNotSupport));
        case TransactionType.Ext3Cancel:
            if (paymentParameters.TransactionType == TransactionType.Ext3Payment)
                return RunTransaction(paymentParameters.Summa, true,
TransactionType.Ext3Refund);
            else
                throw new
DeviceException(BankTerminalStrings.GetString((int)Message.TransactionTypeNotSupport));
        case TransactionType.Cancel:
            if (paymentParameters.TransactionType == TransactionType.Payment)
                return RunTransaction(paymentParameters.Summa, true,
TransactionType.Refund);
            else
                throw new
DeviceException(BankTerminalStrings.GetString((int)Message.TransactionTypeNotSupport));

        case TransactionType.BonusCancel:
            if (paymentParameters.TransactionType == TransactionType.BonusPayment)
                return RunTransaction(paymentParameters.Summa, true,
TransactionType.BonusRefund);
            else
                throw new
DeviceException(BankTerminalStrings.GetString((int)Message.TransactionTypeNotSupport));

        case TransactionType.ClosedLoopCancel:
            if (paymentParameters.TransactionType ==
TransactionType.ClosedLoopPayment)
                return RunTransaction(paymentParameters.Summa, true,
TransactionType.ClosedLoopRefund);
            else
                throw new
DeviceException(BankTerminalStrings.GetString((int)Message.TransactionTypeNotSupport));

        default:
            throw new
DeviceException(BankTerminalStrings.GetString((int)Message.TransactionTypeNotSupport));
    }
}

public TransactionInfoParameter[] GetExtendedParameters(TransactionType
transactionType)
{
    return new TransactionInfoParameter[0];
}

public void CheckExtendedParameters(TransactionInfoParameter[] extParameters,
TransactionType transactionType)
{
}

public void Init(int sareaId, int systemId, int currencyId, int deviceId)
{
}

public void Stop()
{
    isStop = true;
}

public void Open(Port port)
{
    isStop = false;
    isOpen = true;
}

public void Close()
{
    isOpen = false;
}

```

```

public string DeviceName
{
    get
    {
        return "D06DCC04-D06B-4fbf-ABB6-4C1823EDE993";
    }
}

public String DeviceFriendlyName
{
    get
    {
        return
BankTerminalStrings.GetString((int)Message.BankTerminalEmuFriendlyName);
    }
}

public bool IsOfflineMode
{
    get
    {
        return false;
    }
}

public void OfflineAuthorization(string authorizationCode)
{
}

public void VirtualPort(string message)
{
}

public event StatusMessageEventHandler StatusMessage;

public event PrintMessageEventHandler PrintMessage;
protected void OnStatusMessage(string message)
{
    MessageEventArgs args = new MessageEventArgs(message);

    if(StatusMessage != null)
        StatusMessage(this, args);
}

public AccountInfo[] GetAccountsInfo(string cardCode)
{
    return null;
}

public void OpenWorkDay()
{
}

public void CloseWorkDay()
{
}

#endregion

#region IDevice Members

public Parameter[] GetParameters(PortType portType)
{
    return Parameters;
}

public Parameter[] GetDefaultParameters(PortType portType)
{
    return GetParameters(portType);
}

```

```

public bool IsPortSupported(PortType portType)
{
    foreach(PortType supPort in PortTypes)
    {
        if(supPort == portType)
            return true;
    }

    return false;
}

public bool IsOpened
{
    get
    {
        return isOpen;
    }
}

    #endregion

private PaymentParameters RunTransaction(Money summa, bool refund, TransactionType
transactionType)
{
    return new PaymentParameters(summa, "", "", "", mfo, transactionType);
}
protected enum SettingsParameter
{
    BankName = 1,
    MFO = 2,
    TransactionMode = 4,
}
}

```