



ООО “АСТРОН ЛТД”

**Пакет прикладных программ
PowerPOS Версия 5.0**

**Описание работы с
процессинговым центром**

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 работы с процессинговым центром	4
Операция оплаты	4
Операция возврата	4
Операция отмены	5
Листинг примера работы с процессинговым на основе реализации драйвера банковского терминала	5

Описание работы с процессинговым центром

Приложение, работающее с процессинговым, должно представлять собой сборку для Microsoft .Net 2.0. В сборку необходимо включить ProcessingGlobals.dll и ProcessingServiceProxy.dll.

Для связи с процессинговым центром, необходимо создать экземпляр сервиса счетов, поддерживающего интерфейс IAccountService, для которого необходимо указать адрес процессингового сервера, порт подключения и таймаут ожидания ответа от сервера в секундах.

```
using SoftMarket.MP2003.ProcessingCenter.Service;  
IAccountService accountService =  
    AccountFactory.CreateRemoteService("127.0.0.1:7575", 30);
```

Операция оплаты

Для того, чтобы осуществить операцию оплаты на процессинговом центре, необходимо сформировать документ оплаты PaymentDocument и передать его через созданный сервис счетов accountService.

```
using SoftMarket.MP2003.ProcessingCenter.AccountManagement;  
  
PaymentDocument document = new PaymentDocument(docSource,  
    transactionID, paymentSchemaID, currencyID, cardCode, sum);  
document.SAreaID = sareaID;  
document.SystemID = systemID;  
document.PinCode = PINCode;  
  
accountService.ProcessDocument(document, false);
```

Свойства документа оплаты

Свойство	Описание
transactionID	идентификатор документа
paymentSchemaID	идентификатор схемы оплаты.
currencyID	идентификатор валюты, в которой выполняется оплата
cardCode	код карты, для идентификации клиента
sum	сумма документа
sareaID	идентификатор торговой площадки, сформировавший документ
systemID	идентификатор системы, сформировавший документ
PINCode	ПИН-код карты
docSource	источник, сформировавший документ (см. тип AccountDocSource)

Операция возврата

Для того, чтобы осуществить операцию возврата по документу оплаты, необходимо сформировать документ возврата PaymentRefundDocument и передать его также через сервис счетов accountService.

```
using SoftMarket.MP2003.ProcessingCenter.AccountManagement;  
  
PaymentRefundDocument document = new PaymentRefundDocument(docSource,  
    transactionID, transactionIDToRefund, sum);  
document.SAreaID = sareaID;  
document.SystemID = systemID;
```

```
accountService.ProcessDocument (document, false);
```

Свойства документа возврата

Свойство	Описание
transactionID	идентификатор документа
transactionIDToRefund	идентификатор документа оплаты, по которому выполняется возврат
sum	сумма документа
sareaID	идентификатор торговой площадки, сформировавший документ
systemID	идентификатор системы, сформировавший документ
docSource	источник, сформировавший документ (см. тип AccountDocSource)

Операция отмены

Для того, чтобы провести документ отмены операции (оплаты или возврата), необходимо сформировать документ отмены CancelDocument и передать его также через сервис счетов accountService.

```
using SoftMarket.MP2003.ProcessingCenter.AccountManagement;
```

```
CancelDocument document = new CancelDocument (docSource, transactionID,
    transactionIDToCancel);
document.SAreaID = sareaID;
document.SystemID = systemID;
```

```
accountService.ProcessDocument (document, false);
```

Свойства документа отмены

Свойство	Описание
transactionID	идентификатор документа
transactionIDToCancel	идентификатор документа, который нужно отменить
sareaID	идентификатор торговой площадки, сформировавший документ
systemID	идентификатор системы, сформировавший документ
docSource	источник, сформировавший документ (см. тип AccountDocSource)

Листинг примера работы с процессинговым на основе реализации драйвера банковского терминала

```
using SoftMarket.Devices;
using SoftMarket.Globals.Units;
using SoftMarket.Devices.IO;
using SoftMarket.Globals;
using SoftMarket.MP2003.ProcessingCenter.Service;
using SoftMarket.MP2003.ProcessingCenter;
using SoftMarket.MP2003.Globals;

/// <summary>
/// Класс драйвера банковского терминала
/// </summary>
public class BankTerminal : IBankTerminal
{
    private string virtualPortMessage = String.Empty;
```

```

private bool isStopped = false;

private int sareaID;
private int systemID;
private int currencyID;

private IAccountService accountService = null;

private delegate void RunPatmentTransaction(TransactionInfo
    transactionInfo);

/// <summary>
/// Структура данных о выполняемой транзакции
/// </summary>
private struct TransactionInfo
{
    public Guid TransactionID;
    public Money Sum;
    public string DCardCode;
    public string PINCode;
    public int PaymentSchemaID;
    public Guid TransactionIDToCancel;
    public Guid TransactionIDToRefund;
}

/// <summary>
/// Инициализация начальных параметров
/// </summary>
/// <param name="sareaID">Идентификатора торговой площадки</param>
/// <param name="systemID">Идентификатор кассы</param>
/// <param name="currencyId">Идентификатор валюты</param>
/// <param name="deviceId">Идентификатор устройства</param>
public void Init(int sareaID, int systemID, int currencyID,
    int deviceId)
{
    this.sareaID = sareaID;
    this.systemID = systemID;
    this.currencyID = currencyID;
}

/// <summary>
/// Устанавливает соединение с устройством
/// </summary>
/// <param name="port">Порт, по которому устанавливается
соединение</param>
public void Open(Port port)
{
    try
    {
        this.accountService =
            AccountFactory.CreateRemoteService("127.0.0.1:7575",
30);
    }
    catch (Exception e)
    {
        throw new DeviceException(e.Message, e);
    }
}

/// <summary>
/// Прерывает выполнение текущей транзакции
/// </summary>
public void Stop()
{

```

```

        this.isStopped = true;
    }

    /// <summary>
    /// Выполняет транзакцию указанного типа
    /// </summary>
    /// <param name="summa">Сумма</param>
    /// <param name="paymentParameters">
    /// Информация о предыдущей транзакции для операций отмены и возврата
    /// </param>
    /// <param name="transactionType">Тип транзакции</param>
    public PaymentParameters ExecuteTransaction(Money summa,
        PaymentParameters paymentParameters,
        TransactionType transactionType,
        TransactionInfoParameter[] extParameters)
    {
        this.isStopped = false;
        this.virtualPortMessage = "";

        try
        {
            string cardCode = String.Empty;

            // Определяем, есть ли в доп. параметрах код карты
            bool isCardCodeParamSpecified = false;
            foreach (TransactionInfoParameter parameter in
extParameters)
            {
                if (parameter.Id ==
(int)BankTerminalExtParam.CardCode)
                {
                    cardCode = parameter.Value;
                    isCardCodeParamSpecified = true;

                    break;
                }
            }

            // Если тип оплаты «оплата» или «оплата бонусами» или
            // установлен флаг «проверять код карты» и
            // код карты не определен на предыдущем шаге, тогда
            // считываем код карты с порта устройства
            if ((transactionType == TransactionType.Payment
                || transactionType == TransactionType.BonusPayment
                || this.checkCardCode) && !isCardCodeParamSpecified)
            {
                while (String.IsNullOrEmpty(this.virtualPortMessage))
                {
                    if (this.isStopped)
                        throw new DeviceException
                            ("Операция отменена пользователем");

                    Thread.Sleep(50);
                }

                cardCode = this.virtualPortMessage;
            }

            // Если установлен флаг «проверять код карты», тогда
            // для возврата и отмены проверяем соответствие кода карты
            if (this.checkCardCode)
            {
                switch (transactionType)
                {

```

```

        case TransactionType.BonusRefund:
        case TransactionType.Refund:
        case TransactionType.BonusCancel:
        case TransactionType.Cancel:
        {
            if (String.Compare(cardCode,
                                paymentParameters.CardNumber, true)
                != 0)
                throw new DeviceException(«Неверный код
                карты»);

            break;
        }
    }

    if (this.isStopped)
        throw new DeviceException(«Операция отменена
пользователем»);

    // Определяем PIN-код карты, если есть
    string pinCode = String.Empty;
    foreach (TransactionInfoParameter parameter in
extParameters)
    {
        if (parameter.Id ==
(int)SettingsParameter.PINCodeEnter)
        {
            pinCode = parameter.Value;
            break;
        }
    }

    Guid transactionID = Guid.NewGuid();
    RunPatmentTransaction runTransaction = null;

    TransactionInfo transactionInfo = new TransactionInfo();
    transactionInfo.TransactionID = transactionID;

    switch (transactionType)
    {
        case TransactionType.BonusRefund:
        case TransactionType.Refund:
            transactionInfo.Sum = sum;
            transactionInfo.DCardCode = cardCode;
            transactionInfo.TransactionIDToRefund =
                new
Guid(paymentParameters.TransactionInfo);

            runTransaction = new
RunPatmentTransaction(RefundSum);

            break;

        case TransactionType.BonusPayment:
        case TransactionType.Payment:
            transactionInfo.Sum = sum;
            transactionInfo.DCardCode = cardCode;
            transactionInfo.PINCode = pinCode;
            transactionInfo.PaymentSchemaID =
this.paymentSchemaID;

            runTransaction = new
RunPatmentTransaction(PaymentSum);

```



```

        break;

        case TransactionType.BonusCancel:
        case TransactionType.Cancel:
            transactionInfo.TransactionIDToCancel =
                new
Guid(paymentParameters.TransactionInfo);

            runTransaction= new
RunPatmentTransaction(PaymentCancel);

            break;

        default:
            throw new DeviceException («Операция не
поддерживается»);
    }

    IAsyncResult asyncResult =
        runTransaction.BeginInvoke(transactionInfo, null,
null);

    bool isSeccessful = false;
    if (asyncResult.AsyncWaitHandle.WaitOne(30 * 1000, false))
    {
        if (asyncResult.IsCompleted)
            runTransaction.EndInvoke(asyncResult);

        isSeccessful = true;

        break;
    }

    PaymentParameters paymentParameter = new
PaymentParameters(sum,
        String.Empty, String.Empty, cardCode, this.bankMFO,
        transactionType);
    paymentParameter.TransactionInfo = transactionID.ToString();

    return paymentParameter;
}

catch (Exception e)
{
    throw new DeviceException(e.Message, e);
}

}

/// <summary>
/// Выполняет операцию оплаты
/// </summary>
/// <param name="transactionInfo">Информация о транзакции</param>
private void PaymentSum(TransactionInfo transactionInfo)
{
    PaymentDocument document = new PaymentDocument(this.DocSource,
        transactionInfo.TransactionID,
transactionInfo.PaymentSchemaID,
        this.currencyID, transactionInfo.DCardCode,
        transactionInfo.Sum.DecimalAmount);
    document.SAreaID = this.sareaID;
    document.SystemID = this.systemID;
}

```

```

        document.PinCode = transactionInfo.PINCode;

        this.accountService.ProcessDocument(document, false);
    }

    /// <summary>
    /// Выполняет операцию возврата по документу оплаты
    /// </summary>
    /// <param name="transactionInfo">Информация о транзакции</param>
    private void RefundSum(TransactionInfo transactionInfo)
    {
        PaymentRefundDocument document = new
        PaymentRefundDocument(this.DocSource,
            transactionInfo.TransactionID,
            transactionInfo.TransactionIDToRefund,
            transactionInfo.Sum.DecimalAmount);
        document.SAreaID = this.sareaID;
        document.SystemID = this.systemID;

        this.accountService.ProcessDocument(document, false);
    }

    /// <summary>
    /// Выполняет операцию отмены документа
    /// </summary>
    /// <param name="transactionInfo">Информация о транзакции</param>
    private void PaymentCancel(TransactionInfo transactionInfo)
    {
        CancelDocument document = new CancelDocument(this.DocSource,
            transactionInfo.TransactionID,
            transactionInfo.TransactionIDToCancel);
        document.SAreaID = this.sareaID;
        document.SystemID = this.systemID;

        this.accountService.ProcessDocument(document, false);
    }
}

```