

# Создание нового модуля для работы с процессингом

Создание нового модуля заключается в создании динамически подгружаемой библиотеки с определенным программным интерфейсом. Все экспортируемые функции модуля можно условно разделить на группы:

- Функции общего назначения
- Функции безналичной оплаты
- Функции приема платежей (оплата услуг)
- Функции работы с подарочными картами
- Функции работы с дисконтными картами

## Функции общего назначения

### Функция GetInfo

Обязательная функция. Вызывается менеджером оплат для получения информации о подключаемом модуле.

```
int GetInfo(Info *info);
```

Параметры:

info ссылка на структуру [Info](#).

Возвращает 1, если выполнена успешно. Если функция вернет 0, менеджер оплат не будет загружать и использовать этот модуль.

### Функция ShowProperties

Не обязательная функция. Вызывается менеджером оплат для отображения окна настроек модуля.

```
int ShowProperties();
```

Возвращает 1, если настройки были изменены. Если настройки не изменены, возвращает 0.

Функция ShowProperties вызывается только для инициализированного модуля. Если инициализация модуля завершилась с ошибкой (функция Init вернула 0), менеджер оплат не будет производить вызов функции ShowProperties.

### Функция Init

Обязательная функция. Вызывается менеджером оплат для инициализации модуля.

```
int Init(char *path, Error *error);
```

Параметры:

path - путь к каталогу, в котором находится загружаемый модуль.

error ссылка на структуру [Error](#).

Возвращает 1, если инициализация модуля прошла успешно. Если инициализация не прошла успешно, то структура error содержит код и описание ошибки, а функция возвращает 0.

## Функция Done

Не обязательная функция. Вызывается менеджером оплат для деинициализации модуля.

```
int Done();
```

Возвращает 1, если деинициализация модуля прошла успешно, в противном случае возвращает 0.

## Функция CancelLastTransaction

Не обязательная функция. Вызывается менеджером оплат для технологической отмены транзакции.

```
int CancelLastTransaction(PayInfo *payinfo);
```

Параметры:

payinfo ссылка на структуру [PayInfo](#).

Возвращает 1, если транзакция успешно отменена, в противном случае возвращает 0.

## Функция CloseDay

Не обязательная функция. Вызывается менеджером оплат завершения смены кассира.

Функциональная нагрузка функции: сверка итогов, фиксация операций в процессинге, печать сменного отчёта и т. п.

```
int CloseDay(PayInfo *payinfo);
```

Параметры:

payinfo ссылка на структуру [PayInfo](#).

Возвращает 1, если закрытие дня выполнено полностью, в противном случае возвращает 0.

## Функция PrintReport

Не обязательная функция. Вызывается менеджером оплат для печати дополнительного отчета.

```
int PrintReport(PayInfo *payinfo);
```

Параметры:

payinfo ссылка на структуру [PayInfo](#).

Возвращает 1, если отчет сформирован успешно и его можно печатать, в противном случае возвращает 0. Текст отчета возвращается в поле StringForPrint структуры payinfo.

## Функция PrintCheckCopy

Не обязательная функция. Вызывается менеджером оплат для повторной печати слипа. Поиск оригинального документа осуществляется по параметрам RRN и AuthCode структуры payinfo.

```
int PrintCheckCopy(PayInfo *payinfo);
```

Параметры:

payinfo ссылка на структуру [PayInfo](#).

Возвращает 1, если слип сформирован успешно и его можно печатать, в противном случае возвращает 0. Текст слипа возвращается в поле StringForPrint структуры payinfo.

## Функция CheckConnection

Не обязательная функция. Вызывается менеджером оплат для проверки связи с узлами модуля. Вызов функции инициирует пользователь.

```
int CheckConnection(PayInfo *payinfo);
```

Параметры:

payinfo ссылка на структуру [PayInfo](#).

Возвращает 1, если связь со всеми узлами установлена, в противном случае возвращает 0. Если функция вернула 0, то в полях структуры должны содержаться код и описание ошибки.

## Функция Commit

Экспорт этой функции обязателен, если проводятся операции, требующие подтверждения. Если операция требует подтверждения, то после успешного ее выполнения в поле MustCommit структуры одного из параметров функции возвращается 1. Это необходимое условие для того, чтобы менеджер оплат вызвал функцию Commit.

Функция Commit вызывается не для всех операций, требующих подтверждения. Решение

подтвердить или нет операцию, требующую подтверждения, принимается менеджером оплат. Для идентификации подтверждаемой операции менеджер оплат передает ее RRN и AuthCode в параметрах структуры `payinfo`.

```
int Commit(PayInfo *payinfo);
```

Параметры:

`payinfo` ссылка на структуру [PayInfo](#).

Возвращает 0, только если операцию подтвердить невозможно. Если операция не подтверждена из-за отсутствия связи с процессингом, то следует возвращать 1 и выполнять операцию в фоне или по закрытию дня.

## Функции безналичной оплаты

Если модуль поддерживает обработку безналичной оплаты, то функция `Pay` является обязательной к экспорту. Остальные функции этой группы не экспортируются, если не экспортируется функция `Pay`.

### Функция `Pay`

Обязательная функция (если модуль поддерживает обработку безналичной оплаты). Вызывается менеджером оплат для выполнения операции с безналичным счетом клиента.

```
int Pay(PayInfo *payinfo);
```

Параметры:

`payinfo` ссылка на структуру [PayInfo](#).

Возвращает 1, если операция прошла успешно, в противном случае возвращает 0. Если функция выполнялась некорректно, то поля структуры `payinfo` содержат код и описание ошибки.

### Функция `VoicePay`

Не обязательная функция. Вызывается менеджером оплат для выполнения операции с безналичным счетом клиента, используя голосовую авторизацию.

```
int VoicePay(PayInfo *payinfo);
```

Параметры:

`payinfo` ссылка на структуру [PayInfo](#).

Возвращает 1, если операция прошла успешно, в противном случае возвращает 0. Если функция выполнялась некорректно, то поля структуры `payinfo` содержат код и описание ошибки.

## Функции оплаты услуг

### Функция ServicePay

Не обязательная функция. Вызывается менеджером оплат для выполнения операции по приему платежей (оплаты услуг).

```
int ServicePay(int iterID, ServicePayInfo *payinfo);
```

Параметры:

iterID. Принимает значение 0 или 1. Если iterID равен 0, то происходит проверка возможности провести платеж. Если 1, то происходит фиксация платежа в процессинге.

payinfo указатель на структуру [ServicePayInfo](#).

Возвращает 1, если операция прошла успешно, в противном случае возвращает 0. Если функция выполнялась некорректно, то поля структуры payinfo содержат код и описание ошибки.

Примечание. Если iterID равен 1, то функция может вернуть 0 только в случае фатальной ошибки. К фатальным ошибкам относится, например, неверные идентификатор сессии или идентификатор транзакции. Отсутствие связи с процессингом не относится к фатальным ошибкам. При отсутствии связи с процессингом или других ошибках, подразумевающих повторение операции, рекомендуется возвращать 1 и повторять выполнение операции в фоновом режиме или при сверке итогов.

## Функции работы с подарочными картами

### Функция GetGiftCardNominal

Не обязательная функция. Вызывается менеджером оплат для получения номинала подарочной карты.

```
int GetGiftCardNominal(PayInfo *payinfo);
```

Параметры:

payinfo ссылка на структуру [PayInfo](#).

Возвращает 1, если операция прошла успешно, в противном случае возвращает 0. Если функция выполнялась корректно, то номинал карты возвращается в поле Amount структуры.

### Функция ActivateGiftCard

Не обязательная функция. Вызывается менеджером оплат для активации подарочной карты в процессинге.

```
int ActivateGiftCard(PayInfo *payinfo);
```

Параметры:

payinfo ссылка на структуру [PayInfo](#).

Возвращает 1, если карта активирована, в противном случае возвращает 0.

## Функция **GetGiftCardBalance**

Не обязательная функция. Вызывается менеджером оплат для получения остатка на подарочной карте.

```
int GetGiftCardBalance(GiftCardInfo *info);
```

Параметры:

info ссылка на структуру [GiftCardInfo](#).

Возвращает 1, если остаток получен, в противном случае возвращает 0. Если функция выполнена успешно, то поле Amount содержит остаток средств на карте.

## Функция **ActivateGiftCardArbitraryNominal**

Не обязательная функция. Вызывается менеджером оплат для активации подарочной карты произвольного номинала.

```
int ActivateGiftCardArbitraryNominal(PayInfo *info);
```

Параметры:

info ссылка на структуру [PayInfo](#). В поле Amount менеджер оплат передаёт значение номинала активируемой карты.

Возвращает 1, если карта успешно активирована в процессинге, в противном случае возвращает 0.

## Функция **GiftCardPreActivationSlip**

Не обязательная функция. Вызывается менеджером оплат для получения преактивационного слипа.

```
int GiftCardPreActivationSlip(PayInfo *info);
```

Параметры:

info ссылка на структуру [PayInfo](#).

Возвращает 1, если карта может быть активирована, в противном случае возвращает 0. Текст слипа должен передаться менеджеру оплат в поле StringForPrint структуры info.

## Функции работы с дисконтными картами

Работа с дисконтными картами реализуется одним из двух режимов:

- Процессинг является ведомым
- Процессинг является ведущим

Первый режим предоставляет возможность получить номер карты из процессинга и отправить в процессинг результат обработки карты в чеке (предоставленную скидку, начисленные бонусы и т. п.). Для реализации этого режима необходимо реализовать экспорт функция `ExternalDiscount_GetID` и `ExternalDiscount_UpdateDC`.

Второй режим подразумевает полную обработку скидок процессингом и только оповещает менеджер оплат о необходимости произвести начисление скидки на чек, сообщает менеджеру оплат текст с дополнительной информацией для печати на чеке. Реализация этого режима производится экспортом функций `CorrectDiscountCard` и `GetDiscountCardInfo`.

### Функция `ExternalDiscount_GetID`

Не обязательная функция. Вызывается менеджером оплат для получения номера дисконтной карты.

```
int ExternalDiscount_GetID(DiscountCardInfo *info);
```

Параметры:

info ссылка на структуру [DiscountCardInfo](#).

Возвращает 1, если номер карты получен из процессинга, в противном случае возвращает 0. Номер карты возвращается в поле `CardNumber` структуры `info`.

### Функция `ExternalDiscount_UpdateDC`

Не обязательная функция. Вызывается менеджером оплат для информирования процессинга о скидке по карте, начисленных бонусах.

```
int ExternalDiscount_UpdateDC(PayInfo *info);
```

Параметры:

info ссылка на структуру [PayInfo](#).

Возвращает 1, если функция выполнена успешно, в противном случае возвращает 0.

### Функция `GetDiscountCardInfo`

Не обязательная функция. Вызывается менеджером оплат для получения информации о карте.

```
int GetDiscountCardInfo(DiscountCardInfo *info);
```

Параметры:

info ссылка на структуру [DiscountCardInfo](#).

Возвращает 1, если функция выполнена успешно, в противном случае возвращает 0.

## Функция CorrectDiscountCard

Не обязательная функция. Вызывается менеджером оплат для информирования процессинга о составе чека для корректирования информации по карте.

```
int CorrectDiscountCard(PayInfo *info);
```

Параметры:

info ссылка на структуру [PayInfo](#).

Возвращает 1, если функция выполнена успешно, в противном случае возвращает 0.

## Функция GetMaxPayAmountOnCheck

Не обязательная функция. Вызывается менеджером оплат для получения от процессинга информации о максимально возможной сумме списания баллов с карты и о сумме скидки, которую необходимо применить к чеку. Предполагается, что после вызова этой функции состав чека меняться не будет.

```
int GetMaxPayAmountOnCheck(PayInfo *info);
```

Параметры:

info ссылка на структуру [PayInfo](#).

Возвращает 1, если функция выполнена успешно, в противном случае возвращает 0.

Функция модифицирует поля структуры info.

Amount – Максимально возможная сумма списания баллов в текущем чеке.

DiscountAmount – Скидка, которую кассовому ПО следует применить к чеку.

## Функция ConvertDiscountCard

Не обязательная функция. Вызывается менеджером оплат для конвертации дисконтной карты или для связи двух карт.

```
int ConvertDiscountCard(DiscountCardInfo *info);
```

Параметры:

info ссылка на структуру [DiscountCardInfo](#).

Возвращает 1, если функция выполнена успешно, в противном случае возвращает 0.

## Структуры в параметрах функций

### Структура Info

```
struct Info
{
    int version;
    char *name;
    char *terminal_id;
    char *number_of_point;
    AddInfo *addinfo;
};
```

Описание полей:

version – содержит текущее значение версии протокола.

name – Наименование процессинга для менеджера оплат. Если NULL, используется имя из конфигурационного файла менеджера оплат.

terminal\_id – не используется.

number\_of point – не используется.

addinfo – Указатель на структуру [AddInfo](#). Экземпляр структуры модуль создает в памяти и хранит до завершения своей работы.

### Структура Error

```
struct Error
{
    int code;
    char *message;
};
```

Структура соержжит описание ошибки. Предназначена для передачи информации об ошибке в менеджер оплат.

Описание полей:

code – внутренний код ошибки модуля.

message – текстовое описание ошибки.

### Структура AddInfo

```
struct AddInfo
{
    int ReadCardOnCash;
    int MultiAuth;
    int ReadPhoneNumberOnCash;
```

```
int ReadServiceProviderOnCash;  
int MultiCloseDay;  
};
```

Структура предназначена для передачи дополнительной информации о модуле менеджеру оплат.

Поля:

ReadCardOnCash – Принимает значение 0, если считывание карты производится периферией модуля. 1, если менеджер оплат обязан произвести чтение карты своими средствами.

MultiAuth – Принимает значение 1, если модуль поддерживает работу с несколькими юридическими лицами. В противном случае 0.

ReadPhoneNumberOnCash – Если номер телефона (при оплате услуг) поставляется менеджером оплат, то значение поля должно быть равно 1. В противном случае 0.

ReadServiceProviderOnCash – Если выбор провайдера услуг (при оплате услуг) производится модулем, то значение поля должно быть равно 0. В противном случае (1) менеджер оплат передаст в модуль условный код провайдера.

MultiCloseDay – Поле может быть равно 1 только при MultiAuth равном 1. Если поле равно 1, то менеджер оплат будет вызывать функцию CloseDay отдельно для каждого юридического лица. В противном случае функция CloseDay будет вызвана только один раз (даже при работе с несколькими юридическими лицами).

## Структура PayInfo

```
struct PayInfo  
{  
    int Amount;  
    char *CardNumber;  
    int KKMNumber;  
    int CheckNumber;  
    char *AuthCode;  
    char *RRN;  
    int Error;  
    char *ErrorMsg;  
    char *StringForPrint;  
    char *DisplayMessage;  
    int DiscountAmount;  
    int MustCommit;  
    char *StringForBuy;  
    int ComissionAmount;  
    int BonusAmount;  
    int GoodsCount;  
    void *Goods;  
    int PaysCount;  
    void *Pays;  
    int CardInputType;  
    int CashierCode;  
    char *CashierName;  
    char *CardType;
```

```
};
```

Поля:

Amount – Сумма операции в сотых долях валюты.

CardNumber – Номер карты (второй трэк).

KKMNumber – Логический номер кассы или номер юридического лица, если модуль работает в режиме нескольких юридических лиц.

CheckNumber – номер чека, получается кассовым ПО.

AuthCode – Код авторизации.

RRN – RRN.

Error – Код ошибки.

ErrorMsg – Текстовое представление ошибки.

StringForPrint – Текст слипа для печати на чековой ленте.

DisplayMessage – Сообщение, которое необходимо отобразить в диалоге, если таковой выводится при операции.

DiscountAmount – Сумма скидки.

MustCommit – Выходной параметр. Если значение равно 1, то операция должна быть подтверждена функцией Commit. Если 0, то операция не нуждается в подтверждении.

StringForBuy – Текст, который кассовое ПО напечатает в кассовом чеке.

ComissionAmount – Сумма комиссии в сотых долях валюты.

BonusAmount – Начисленный бонус в сотых долях валюты.

GoodsCount – Количество элементов массива Goods.

Goods – Массив указателей на структуры [GoodItem](#).

PaysCount – Количество элементов массива Pays.

Pays – Массив указателей на структуры [PayItem](#).

CardInputType – Способ ввода карты. Принимает значения:

0 – Ридер магнитных карт

1 – Сканер штрихкода

2 – Ручной ввод

CashierCode – Код кассира.

CashierName – ФИО Кассира.

CardType - Выходной. Тип карты, которой совершалась операция.

## Структура ServicePayInfo

```
struct ServicePayInfo
{
    int Amount;
    char *PhoneNumber;
    char *TransID;
    char *OperCode;
    char *OperName;
    char *SessionID;
    int ComissionAmount;
    int ErrorCode;
    char *ErrorMessage;
    char *StringForPrint;
    char *StringForBuy;
};
```

## Структура GiftCardInfo

```
struct GiftCardInfo
{
    int Amount;
    char *CardNumber;
    int ErrorCode;
    char *ErrorMessage;
    char *StringForPrint;
    char *DisplayMessage;
};
```

## Структура DiscountCardInfo

```
struct DiscountCardInfo
{
    int Balance;
    int CheckCount;
    int CheckSum;
    char *CardNumber;
    int ErrorCode;
    char *ErrorMessage;
    char *StringForPrint;
    char *DisplayMessage;
    int Discount;
    int CardInputType;
    char *AltCardNumber;
    int AltCardInputType;
};
```

```
char *ClientName;  
SYSTEMTIME ClientBirthday;  
};
```

ClientName – ФИО клиента. Входной/Выходной в зависимости от контекста выполнения.  
ClientBirthday – День рождения клиента. Входной/Выходной в зависимости от контекста выполнения. См. также [SYSTEMTIME](#).

## Структура GoodItem

```
struct GoodItem  
{  
    char *articul;  
    char *name;  
    int quant;  
    int price;  
    int amount;  
    int discount;  
    int maxdiscount;  
    char *barcode;  
    int CategoriesCount;  
    char** Categories;  
};
```

Поля:

articul – Артикул товара.

name – Наименование товара.

quant – Количество товара в тысячных долях единицы.

price – цена за единицу товара в сотых долях валюты.

amount – стоимость товара с учётом скидки в сотых долях валюты.

discount – сумма скидки на товар в сотых долях валюты.

maxdiscount – максимально возможная сумма скидки на товар в сотых долях валюты.

barcode – Штрихкод товара.

CategoriesCount – количество элементов в массиве Categories;

Categories – массив доп. категорий товаров;

## Структура PayItem

```
struct PayItem  
{  
    int articul;  
    int ecr_articul;  
    char *name;  
    int amount;  
    int CategoriesCount;  
    char** Categories;  
};
```

Поля:

articul – Артикул вида оплаты.

esc\_articul – Номер вида оплаты в ККМ.

name – Наименование вида оплаты.

amount – Сумма в сотых долях валюты по виду оплаты в чеке.

CategoriesCount – количество элементов в массиве Categories;

Categories – массив доп. категорий оплат;

From: <https://wiki.ilexx.ru/> - Штрих-М: Документация

Permanent link: [https://wiki.ilexx.ru/doku.php?id=%D0%BC%D0%B5%D0%BD%D0%B4%D0%B6%D0%B5%D1%80\\_%D0%BE%D0%BF%D0%BB%D0%B0%D1%82:%D1%80%D0%B0%D1%81%D1%88%D0%B8%D1%80%D0%BD%D0%B8%D0%B5&rev=1563821788](https://wiki.ilexx.ru/doku.php?id=%D0%BC%D0%B5%D0%BD%D0%B4%D0%B6%D0%B5%D1%80_%D0%BE%D0%BF%D0%BB%D0%B0%D1%82:%D1%80%D0%B0%D1%81%D1%88%D0%B8%D1%80%D0%BD%D0%B8%D0%B5&rev=1563821788)

Last update: 2019/07/22 21:56

