Resonance OLE Manager 2017

Версия 4.1.20170208

1. Содержание

2. Общая информация

OLE-automation сервер предназначен для работы с ЭККА Мария-301МТМ Т7-Т11, Мария-М304.
Название COM-сервера(ProgId): **M304Manager.Application**
Также, из соображений обратной совместимости, допускается в качестве ProgId указывать **M301Manager.Application**
(может не поддерживать некоторые функции, относящиеся к новым фискальным аппаратам и присутствующие в M304Manager.Application)
Порядок работы с сервером следующий:

- программно создать COM-объект M304Manager.Application, используя средства языка программирования
- установить соединение с ЭККА, используя функцию Init объекта
- выполнить требуемые операции с ЭККА, вызывая соответствующие функции объекта
- закрыть соединение, используя функция Done

Следует обратить внимание, что все финансовые функции сервера требуют указания сумм в копейках (целых чисел без знака)

Также, начиная с версии 4.1, имеется возможность добавить сборку Resonance.OLEManager.dll в c#-проект Visual Studio и работать с ЭККА без регистрации COM-сервера (в v.4.0 данная возможность официально не поддерживалась, хоть и работала).

3. Функции

1. Установка соединения

int **Init** (int portNumber , string CashierName opt , string EkkaPassword opt , int showStatusDialogs opt)

Устанавливает подключение к ЭККА

Параметры

- **portNumber**: номер ком-порта
- **CashierName**: имя кассира *(необязательный параметр)*
- **EkkaPassword**: пароль доступа к ЭККА *(необязательный параметр)*
- **showStatusDialogs**: отображать состояние длительно выполняющихся операций *(необязательный параметр)*

Возвращаемое значение: 1 в случае успеха, 0 в случае ошибки

Примечание: *Алиас: Connect*

---

int **Init** (string portName , string CashierName opt , string EkkaPassword opt , bool showStatusDialogs opt)

Устанавливает подключение к ЭККА

Параметры

- **portName**: имя порта или IP-адрес ЭККА
- **CashierName**: имя кассира *(необязательный параметр)*
- **EkkaPassword**: пароль доступа к ЭККА *(необязательный параметр)*
- **showStatusDialogs**: отображать состояние длительно выполняющихся операций *(необязательный параметр)*

Возвращаемое значение: 1 в случае успеха, 0 в случае ошибки

Примечание: *В поле portName допускается передача следующих значений:*
- название ком-порта, например "COM1"
- IP-адрес ЭККА, например "192.168.12.34"
- IP-адрес ЭККА и порт, например "192.168.12.34:4545"
- hostname/FQDN аппарата, например "kassa1.company.com.ua" или "kassa1.company.com.ua:4545"
- строка вида tcp://<hostname|ip>[:port], например "tcp://192.168.12.34"

Также следует учитывать, ЭККА одновременно может работать только с одним соединением
Алиас: Connect

---

int **InitAuto** (string username opt , string password opt)

Осуществляет поиск и установку соединения с ЭККА.

Параметры

- **username**: имя кассира *(необязательный параметр)*
- **password**: пароль кассира *(необязательный параметр)*

Возвращаемое значение: 1 в случае успеха, 0 в случае ошибки

Примечание: *Перебирает все доступные в системе COM-порты, пытается установить связь с ЭККА на каждом из этих портов.*
В случае, если к компьютеру подключено более одного ЭККА, рекомендуется использовать функцию Init.
Также следует обратить внимание, что InitAuto не выполняет поиск ЭККА в сети.
Алиас: ConnectAuto

---

int **Done** ()

Закрывает соединение с ЭККА

Возвращаемое значение: 1 в случае успеха, 0 в случае ошибки

Примечание: *Аппаратное ограничение: повторная установка связи через COM-порт возможна не ранее чем через 3 секунды после разъединения (это вызвано особенностью работы ЭККА). В большинстве случаев это не должно создавать проблемы, разве что вызов следующий вызов Init может занимать больше времени*
Алиас: Disconnect

---

2. Печать чеков

---

int **OpenTextDocument** ()

Открывает служебный документ

Возвращаемое значение: 1 в случае успеха, 0 в случае ошибки

---

int **CloseTextDocument** ()

Закрывает и печатает служебный документ

Возвращаемое значение: 1 в случае успеха, 0 в случае ошибки

---

int **AbortCheck** ()

Отменяет чек, открытый командой OpenCheck

Возвращаемое значение: 1 в случае успеха, 0 в случае ошибки

---

int **AddDiscount** (int сумма , string описание opt , int налог1 opt , int налог2 opt)

Скидка на общий оборот по чеку (уменьшение суммы оборота по чеку)

Параметры

- **сумма**: сумма скидки
- **описание**: описание скидки *(необязательный параметр)*
- **налог1**: налоговая ставка или 0 (уменьшает на оборот то указаному налогу в чеке) *(необязательный параметр)*
- **налог2**: налоговая ставка или 0 (уменьшает на оборот то указаному налогу в чеке) *(необязательный параметр)*

Возвращаемое значение: 1 в случае успеха, 0 в случае ошибки

Примечание: *Поддерживается в ЭККА 304Т-2, 304Т1-2, 304Т2-2.*
Функция применяется в чеке реализации/возврата.
Вызов AddDiscount аналогичен вызову AdjustTotals(-сумма, 1, описание, налог1, налог2)

---

int **AddPayment** (int sum , int paymentType opt)

Добавление оплаты в фискальный чек

Параметры

- **sum**: сумма
- **paymentType**: форма оплаты (0 - 'готівка', 1 - 'безготівка.1', 2 - 'безготівка.2' и т.д.) *(необязательный параметр)*

Возвращаемое значение: 1 в случае успеха, 0 в случае ошибки

---

int **CheckCopy** ()

Печатает копию последнего чека. ЭККА допускает печать не более одной копии чека

Возвращаемое значение: 1 в случае успеха, 0 в случае ошибки

---

int **ClearFreeTextLines** ()

Отменяет строки чека, добавленые командой FreeTextLine

Возвращаемое значение: 1 в случае успеха, 0 в случае ошибки

---

int **CloseCheck** ()

Закрывает и печатает фискальный чек

Возвращаемое значение: Сумма оплаты по чеку в копейках с учетом наложенных налогов

---

int **CloseCheckEx** (int Cash opt , int безнал2 opt , int безнал1 opt , int безнал3 opt , string RRN opt)

Закрывает и печатает фискальный чек

Параметры

- **Cash**: сумма оплаты наличными *(необязательный параметр)*
- **безнал2**: сумма оплаты по форме оплаты "Безготівка.2" *(необязательный параметр)*
- **безнал1**: сумма оплаты по форме оплаты "Безготівка.1" *(необязательный параметр)*
- **безнал3**: сумма оплаты по форме оплаты "Безготівка.3" *(необязательный параметр)*
- **RRN**: идентификатор транзакции платёжного терминала (RRN) *(необязательный параметр)*

Возвращаемое значение: Сумма оплаты по чеку в копейках с учетом наложенных налогов или 0 в случае ошибки

Примечание: *(!)Следует обратить внимание на порядок следования параметров безналичной оплаты, такой порядок требуется для совместимости со старыми версиями olemanager (давным-давно кто-то ~~протупил~~ ошибочно поставил неправильный порядок параметров и теперь его невозможно изменить, не сломав обратную совместимость).*
Также, начиная с версии 4.1.20141209, можно закрывать чеки с помощью команд AddPayment(...)+CloseCheck(), этот способ лишён описаного недостатка.
В ЭККА имеется возможность изменить названия форм оплаты "Безготівка.N" на любые другие, это можно сделать с помощью программы "Консоль администратора ЭККА Мария" (см. раздел "Техническая поддержка" на официальном сайте компании)

Печать чеков с использованием платёжного терминала описана в разделе 5.

---

int **CloseGoodsGroup** (string DiscountTotalsName , string UpcountTotalsName)

Закрывает группу фискальных позиций в пределах чека. Группа должна быть предварительно открыта командой OpenGoodsGroup

Параметры

- **DiscountTotalsName**: наименование итога по скидкам в пределах закрываемой группы
- **UpcountTotalsName**: наименование итога по надбавкам в пределах закрываемой группы

Возвращаемое значение: 1 в случае успеха, 0 в случае ошибки

---

int **FiscalLine** (string название , int количество , int цена , int делимость opt, int налог1 opt, int налог2 opt, int код_артикула opt)

Добавляет строку в фискальный чек

Параметры

- **название**: название товара
- **количество**: количество в штуках или граммах (см. параметр GoodsDividual). При желании не печатать явно единичное количество товара, укажите значение параметра 0
- **цена**: цена в копейках
- **делимость**: делимость товара. 0 - неделимый товар, параметр Qty интерпретируется как "штуки", 1 - делимый товар, параметр Qty интерпретируется как "граммы" *(необязательный параметр)*
- **налог1**: номер схемы налогообложения товара (1-8, или 0, если не подлежит налогообложению) *(необязательный параметр)*
- **налог2**: дополнительная схема налогообложения (0-8) *(необязательный параметр)*
- **код_артикула**: код артикула (1..999999999) (при указании значений 0 или null используется автоматическое назначение кода) *(необязательный параметр)*

Возвращаемое значение: Сумму в копейках по данной строке чека в случае успеха (без учёта наложеных налогов), 0 в случае неудачи

---

int **FiscalLineEx** (string название , int количество , int цена , int делимость , int налог1 , int налог2 , int код_артикула opt, int направление_скидки opt, string название_скидки opt, int сумма_скидки opt)

Добавляет строку в фискальный чек

Параметры

- **название**: название товара
- **количество**: количество в штуках или граммах (см. параметр GoodsDividual). При желании не печатать явно единичное количество товара, укажите значение параметра 0
- **цена**: цена в копейках
- **делимость**: делимость товара. 0 - неделимый товар, параметр Qty интерпретируется как "штуки", 1 - делимый товар, параметр Qty интерпретируется как "граммы"
- **налог1**: номер схемы налогообложения товара (1-8, или 0, если не подлежит налогообложению)
- **налог2**: дополнительная схема налогообложения (0-8)
- **код_артикула**: код артикула (1..999999999) (при указании значений 0 или null используется автоматическое назначение кода) *(необязательный параметр)*
- **направление_скидки**: направление скидки. значения: -1/0/1 – нет_скидки/скидка/надбавка *(необязательный параметр)*
- **название_скидки**: название скидки *(необязательный параметр)*
- **сумма_скидки**: сумма скидки в копейках *(необязательный параметр)*

Возвращаемое значение: Сумму в копейках по данной строке чека в случае успеха, 0 в случае неудачи

---

int **FreeTextLine** (int PlaceBeforeFiscalPart , int PrintOnJournal , int DoubleStrike , string TextMessage)

Добавляет строку служебной информации в фискальный чек или служебный документ

Параметры

- **PlaceBeforeFiscalPart**: размещать данную строку до(1) или после(0) фискальной части чека
- **PrintOnJournal**: печатать(1) или не печатать(0) данную строку на контрольной ленте
- **DoubleStrike**: стиль текста (0/1/2/3 - обычный/двойной ширины/двойной высоты/двойной ширины и высоты)
- **TextMessage**: текст

Возвращаемое значение: 1 в случае успеха, 0 в случае ошибки

---

string **GetCheckResult** ()

Возвращает суммы последнего отпечатаного фискального чека

Возвращаемое значение: Строка вида:
НомерЧека;СумаРеализацииПоЧеку;СумаНаложенныхНалогов;СумаВложенныхНалогов;СумаНаложенныхНалоговСВычитанием;СумаВозвратаПоЧеку;ВозвратНаложенныхНалогов;ВозвратВложенныхНалогов;ВозвратНаложенныхНалоговСВычитанием
все суммы - в копейках

Пример ответа:
0000002338;0000000051;0000000000;0000000009;0000000000;0000000000;0000000000;0000000000;0000000000;

---

string **GetCheckResultXML** ()

Возвращает суммы последнего отпечатаного фискального чека

Возвращаемое значение: XML-строка

Пример ответа:

```
<?xml version="1.0"?>
<m301_check_info>
<check_info check_number="0000000123" total="0000000236" onsale_taxes="0000000000"
included_taxes="0000000000" saleincluded_taxes="0000000000" return="0000000000"
return_onsale_taxes="0000000000" return_included_taxes="0000000000"
return_saleincluded_taxes="0000000000"/>
</m301_check_info>
```

---

int **LongName** (string Name)

Позволяет задать дополнительную строку с названием товара/услуги. Эту функцию следует использовать непосредственно перед FiscalLine/FiscalLineEx

Параметры

- **Name**: дополнительная строка с названием товара/услуги

Возвращаемое значение: 1 в случае успеха, 0 в случае ошибки

---

int **MoveCash** (int MoveDirection , int CashSum)

Выполняет опрерацию внесения/изъятие наличных из кассы

Параметры

- **MoveDirection**: 0/1 - изъятие/внесение
- **CashSum**: сумма в копейках

Возвращаемое значение: 1 в случае успеха, 0 в случае ошибки

---

int **NullCheck** ()

Печатает "нулевой" чек

Возвращаемое значение: 1 в случае успеха, 0 в случае ошибки

---

int **OpenCheck** (string отдел opt)

Открывает фискальный чек

Параметры

- **отдел**: название торгового отдела *(необязательный параметр)*

Возвращаемое значение: 1 в случае успеха, 0 в случае ошибки

---

int **OpenCheckEx** (string department opt , bool use_payment_terminal opt)

Открывает фискальный чек

Параметры

- **department**: торговый отдел *(необязательный параметр)*
- **use_payment_terminal**: разрешает использование платёжного терминала для оплаты (см. раздел 5) *(необязательный параметр)*

Возвращаемое значение: 1 в случае успеха, 0 в случае ошибки

Примечание: *Вызов OpenCheckEx(<отдел>, False) аналогичен вызову OpenCheck(<отдел>)*
Чек, открытый с помощью OpenCheckEx, буферизируется в памяти ole-сервера и передаётся на ЭККА только в момент закрытия чека

---

int **OpenReturnCheck** (string отдел opt)

Открывает фискальный чек возврата

Параметры

- **отдел**: название торгового отдела *(необязательный параметр)*

Возвращаемое значение: 1 в случае успеха, 0 в случае ошибки

---

int **OpenReturnCheckEx** (string department opt , bool use_payment_terminal opt)

Открывает фискальный чек для возврата товаров

Параметры

- **department**: торговый отдел *(необязательный параметр)*
- **use_payment_terminal**: разрешает использование терминала безналичной оплаты (см. раздел 5) *(необязательный параметр)*

Возвращаемое значение: 1 в случае успеха, 0 в случае ошибки

См. также: *OpenCheckEx*

---

int **PrintSlip** (string receipt)

Печатает документ "квитанция платёжного терминала" (Slip)

Параметры

- **receipt**: содержимое документа

Возвращаемое значение: 1 в случае успеха, 0 в случае ошибки

Примечание: *Выполнение команды возможно только при отсутствии открытого (командами OpenChech/OpenReturnCheck/OpenTextDocument) документа*

---

int **SetCheckTotals** (int сумма_реализации , int сумма_возврата)

Устанавливает суммы реализации и возврата, которые ожидаются в текущем чеке

Параметры

- **сумма_реализации**: сумма реализации
- **сумма_возврата**: сумма возврата *(необязательный параметр)*

Возвращаемое значение: 1 в случае успеха, 0 в случае ошибки

Примечание: *Метод SetCheckTotals предназначен для явного указания сумм реализации/возврата, которые ожидаются в фискальном чеке. Он может использоваться для дополнительного контроля при закрытии чека: в случае, если сумма чека, рассчитаная РРО, будет отличаться от сумм, указаных в параметрах "сумма_реализации" и "сумма_возврата", фискальный чек распечатан не будет и вызов CloseCheck вернёт ошибку SOFTCHECK(Недопустимые параметры команды).*
Алиас: SetReceiptTotals

---

int **SetDoubledTaxCalcMode** (int налог1 , int налог2)

Управляет расчётом налогооблагаемого оборота при применении двух налогов

Параметры

- **налог1**: 1-8 - идентификатор (номер) первой применяемой схемы налогообложения
- **налог2**: 1-8 - идентификатор (номер) второй применяемой схемы налогообложения

Возвращаемое значение: 1 в случае успеха, 0 в случае ошибки

Примечание: *Вызывается непосредственно перед каждым вызовом FiscalLine/FiscalLineEx, если требуется указать порядок начисления двух налогов.*
В частности, для корректного начисления 5% акцизного сбора, следует использовать вызов SetDoubledTaxCalcMode(2,1) непосредственно перед вызовом FiscalLine (предполагается, что в РРО предварительно запрограммирован вложеный налог Б=5%)

---

int **SetReturnCheckNumber** (int CheckNumber)

Устанавливает номера чека, по которому производится возврат. Функция должна вызываться перед вызовом OpenReturnCheck

Параметры

- **CheckNumber**: номер чека, по которому производится возврат

Возвращаемое значение: 1 в случае успеха, 0 в случае ошибки

См. также: *SetReturnCheckNumberStr*

---

int **SetReturnCheckNumberStr** (string CheckNumbers)

Устанавливает номера чеков, по которым производится возврат

Параметры

- **CheckNumbers**: номера чеков, по которым производится возврат

Возвращаемое значение: 1 в случае успеха, 0 в случае ошибки

См. также: *SetReturnCheckNumber*

---

int **PrintQR** (string data , int size opt , int error_correction_level opt)

Печатает QR-код на чеке

Параметры

- **data**: текст для кодирования и печати
- **size**: размер QR-кода (10..100) при печати, в процентах от ширины бумаги (по-умолчанию: 67%) *(необязательный параметр)*
- **error_correction_level**: уровень коррекции ошибок (значения: 1(7%),2(15%),3(25%),4(30%) или 7,15,25,30) *(необязательный параметр)*

Возвращаемое значение: 1 в случае успеха, 0 в случае ошибки

Примечание: *Используется так же, как FreeTextLine. Внутренне использует функции UploadImage/PrintImage*

---

int **UploadImage** (Byte[] image , int zoom_x opt , int zoom_y opt)

Загружает штрихкод/2D-код в ЭККА для последующего использования с помощью PrintImage.

Параметры

- **image**: картинка (bmp,png,gif,jpeg)
- **zoom_x**: масштабирование по ширине (1..10) *(необязательный параметр)*
- **zoom_y**: масштабирование по высоте (1..160) *(необязательный параметр)*

Возвращаемое значение: 1 в случае успеха, 0 в случае ошибки

Примечание: *Максимальный размер загружаемой картинки - 160x160, картинка печатается по центру ленты. Для печати произвольных одномерных штрих-кодов рекомендуется загружать картинку высотой в один пиксель и растягивать её по вертикали (zoom_y = ~50). В зависимости от размера картинки и скорости соединения, время выполнения команды может достигать нескольких секунд (максимально: 15с при загрузке 160x160 на минимальной скорости 9600), поэтому рекомендуется минимизировать количество вызовов этой функции и размер загружаемой картинки.*

---

int **LoadImageFrom** (string imagepath , int zoom_x opt , int zoom_y opt)

Загружает штрихкод/2D-код в ЭККА из файла для последующего использования с помощью PrintImage.

Параметры

- **imagepath**: путь к файлу (bmp,png,gif,jpeg)
- **zoom_x**: масштабирование по ширине (1..10) *(необязательный параметр)*
- **zoom_y**: масштабирование по высоте (1..160) *(необязательный параметр)*

Возвращаемое значение: 1 в случае успеха, 0 в случае ошибки

Примечание: *Максимальный размер загружаемой картинки - 160x160, картинка печатается по центру ленты. Для печати произвольных одномерных штрих-кодов рекомендуется загружать картинку высотой в один пиксель и растягивать её по вертикали (zoom_y = ~50). В зависимости от размера картинки и скорости соединения, время выполнения команды может достигать нескольких секунд (максимально: 15с при загрузке 160x160 на минимальной скорости 9600), поэтому рекомендуется минимизировать количество вызовов этой функции и размер загружаемой картинки.*

---

int **PrintImage** (int placeBeforeFiscalPart opt)

Добавляет в чек штрихкод/2D-код, загруженый с помощью UploadImage/LoadImageFrom

Параметры

- **placeBeforeFiscalPart**: печатать картинку до(1) или после(0) фискальной части чека *(необязательный параметр)*

Возвращаемое значение: 1 в случае успеха, 0 в случае ошибки

Примечание: *Используется так же, как FreeTextLine*

---

int **PrintBarcode** (int format, int placement, int printOnJournal, string barcode)

int **PrintBarcode** (string barcode, [Optional] int format, [Optional] int placement, [Optional] int printOnJournal)

int **PrintBarcode** (string barcode)

Добавляет штрих-код в открытый чек

Параметры

- **format**: формат штрихкода (значения: 1 или 13 - EAN13, 2 или 128 - Code128) (необязательный параметр)
- **placement**: 0/1 - печатать перед/после фискальной информации (необязательный параметр)
- **printOnJournal**: 0/1 - печатать на контрольной ленте (необязательный параметр, игнорируется в ЭККА М304)
- **barcode**: штрих-код

Возвращаемое значение: 1 в случае успеха, 0 в случае ошибки

Примечание: *значение format=13 в ЭККА М301 не поддерживается*

---

3. Печать отчётов

---

int **ArticleReport** ()

Печатает отчёт 'Реализация товаров и услуг в разрезе артикулов'

Возвращаемое значение: 1 в случае успеха, 0 в случае ошибки

---

int **CloseBusinessDay** ()

Закрывает фискальную смену и печатает Z-отчёт

Возвращаемое значение: 1 в случае успеха, 0 в случае ошибки

Примечание: *Алиас для ZReport*

См. также: *ZReport*

---

int **CloseBusinessDayAsync** ()

Закрывает фискальную смену и печатает Z-отчёт (операция производится в фоне)

Возвращаемое значение: 1 в случае успеха, 0 в случае ошибки

Примечание: *Алиас для ZReportAsync*

См. также: *ZReportAsync*

---

int **DiscountReport** ()

Печатает отчёт "Примененные скидки и надбавки"

Возвращаемое значение: 1 в случае успеха, 0 в случае ошибки

---

int **PeriodicalFiscalReport** (int FirstZReport , int LastZReport)

Печатает фискальный отчёт за указаный период

Параметры

- **FirstZReport**: номер первого Z-отчёта в периоде
- **LastZReport**: номер последнего Z-отчёта в периоде

Возвращаемое значение: 1 в случае успеха, 0 в случае ошибки

---

int **PeriodicalFiscalReportDate** (DateTime DateFrom , DateTime DateTo)

Печатает фискальный отчёт за указаный период

Параметры

- **DateFrom**: начальная дата
- **DateTo**: конечная дата

Возвращаемое значение: 1 в случае успеха, 0 в случае ошибки

---

int **PeriodicalFiscalReportDateEx** (DateTime DateFrom , DateTime DateTo , bool print_full_report)

Печатает фискальный отчёт за указаный период

Параметры

- **DateFrom**: начальная дата
- **DateTo**: конечная дата
- **print_full_report**: false/true - печать сокращенного/полного отчёта

Возвращаемое значение: 1 в случае успеха, 0 в случае ошибки

---

int **PeriodicalFiscalReportEx** (int FirstZReport , int LastZReport , bool print_full_report)

Печатает фискальный отчёт за указаный период

Параметры

- **FirstZReport**: номер первого Z-отчёта
- **LastZReport**: номер последнего Z-отчёта
- **print_full_report**: false/true - печать сокращенного/полного отчёта

Возвращаемое значение: 1 в случае успеха, 0 в случае ошибки

---

int **XReport** ()

Печатает X-отчёт

Возвращаемое значение: 1 в случае успеха, 0 в случае ошибки

---

int **XReportAsync** ()

Запускает фоновую печать X-отчёта

Возвращаемое значение: 1 в случае успеха, 0 в случае ошибки

---

int **ZReport** ()

Закрывает фискальную смену и печатает Z-отчёт

Возвращаемое значение: 1 в случае успеха, 0 в случае ошибки

---

int **ZReportAsync** ()

Закрывает фискальную смену и печатает Z-отчёт (операция производится в фоне)

Возвращаемое значение: 1 в случае успеха, 0 в случае ошибки

---

4. Настройка параметров

---

int **CutAndBeep** (int CutterOn , int BeepOn , int PartialCutOn)

Управляет работой обрезчика чековой ленты и звуковым сигналом

Параметры

- **CutterOn**: включает(1) или выключает(0) автоматическую обрезку чековой ленты после завершения печати чека
- **BeepOn**: включает(1) или выключает(0) функцию звукового сигнала после завершения печати чека
- **PartialCutOn**: включает(1) или включает(0) функцию неполной обрезки чековой ленты. Имеет значение только при включенной функции автоматической обрезки *(необязательный параметр)*

Возвращаемое значение: 1 в случае успеха, 0 в случае ошибки

---

int **SetArticleMode** (int ArticleMode)

Устанавливает режим артикульной таблицы ЭККА

Параметры
- **ArticleMode**: 0 или 2 - новый режим артикульной таблицы

Возвращаемое значение: 1 в случае успеха, 0 в случае ошибки

Примечание: *Изменение режима артикульной таблицы допускается только при закрытой смене*

---

int **SetCheckBottomLine** (string CheckBottomLine)

Программирование необязательной заключительной строки на чеках

Параметры
- **CheckBottomLine**: текст

Возвращаемое значение: 1 в случае успеха, 0 в случае ошибки

---

int **SetCheckBottomLineEx** (int LineIndex , int PrintOnJournal , int PrintMode , string LineText opt)

Программирование нескольких необязательных заключительных строк на чеке

Параметры
- **LineIndex**: номер строки, 0-9
- **PrintOnJournal**: 0/1 - печать на контрольной ленте
- **PrintMode**: 0/1/2/3 - обычный текст/текст двойной ширины/текст двойной высоты/текст двойной ширины и высоты
- **LineText**: текст *(необязательный параметр)*

Возвращаемое значение: 1 в случае успеха, 0 в случае ошибки

Примечание: *Передача LineIndex=-1 очищает все строки, заданые через SetCheckBottomLineEx*

---

int **SetCheckHeadLine** (string CheckHeadLine)

Программирование заголовочной информационной строки на чеке

Параметры
- **CheckHeadLine**: текст заголовка чеков

Возвращаемое значение: 1 в случае успеха, 0 в случае ошибки

---

int **SetCurrentCheckDirection** (int Direction)

Переключает текущий чек в режим возврата товара

Параметры
- **Direction**: 1

Возвращаемое значение: 1 в случае успеха, 0 в случае ошибки

---

int **SetDeptAlias** (string Alias)

Программирует мнемонику торгового отдела

Параметры
- **Alias**: мнемоника торгового отдела, напр. "Відд." "Окно", "Терм" (макс. 5 символов)

Возвращаемое значение: 1 в случае успеха, 0 в случае ошибки

---

int **SetDiscountPrintMode** (int PrintMode)

Управляет печатью информации о скидках/надбавках в чеке

Параметры
- **PrintMode**: 0 – обычная печать информации о скидке-надбавке после каждой фискальной позиции в чеке, 1 - печать информации об итоговой скидке-надбавке только в итоге чека.

Возвращаемое значение: 1 в случае успеха, 0 в случае ошибки

---

int **SetDiscountUpcountTotalsName** (string DiscountTotalsName , string UpcountTotalsName)

Устанавливает наименования итогов по скидкам/надбавкам в пределах закрываемого чека

Параметры
- **DiscountTotalsName**: наименование итога по скидкам
- **UpcountTotalsName**: наименование итога по надбавкам

Возвращаемое значение: 1 в случае успеха, 0 в случае ошибки

---

int **SetInternalTime** (int Hours , int Minutes , int Seconds)

Изменяет текущее время ЭККА

Параметры
- **Hours**: часы
- **Minutes**: минуты
- **Seconds**: секунды

Возвращаемое значение: 1 в случае успеха, 0 в случае ошибки

Примечание: *В соответствии с ограничениями ЭККА, изменение времени допускается только после выполнения Z-отчёта, новое время должно быть в пределах ±90 минут от текущего времени*

---

int **SetLinePrintingOn** ()

Устанавливает режим «построчной» печати чека

Возвращаемое значение: 1 в случае успеха, 0 в случае ошибки

Примечание: *Применяется до открытия чека командой OpenCheck. Действует только в пределах одного чека.*
После каждой команды создания фискальной позиции чека производится полная выгрузка буфера печати
с ожиданием окончания физического процесса печати и контролем исправности принтера (в т.ч. наличия бумаги)

---

int **SetOnSaleTaxes** (int on)

Определяет режим печати информации о наложенных налогах

Параметры

- **on**: 1 - наложенные налоги печатаются после каждой товарной строки в чеке, 0 - сумма наложенных налогов единожды печатается перед строкой "РАЗОМ"

Возвращаемое значение: 1 в случае успеха, 0 в случае ошибки

---

int **SetOnSaleTaxesPrintMode** (int on)

Определяет режима печати информации о наложенных налогах

Параметры

- **on**: 0/1. 1 - наложенные налоги печатаются после каждой товарной строки в чеке, 0 - сумма наложенных налогов единожды печатается перед строкой "РАЗОМ"

Возвращаемое значение: 1 в случае успеха, 0 в случае ошибки

---

int **SetPrintingDoubleWidth** (int DoubleWidth)

Ничего не делает (функция оставлена из соображений обратной совместимости)

Параметры

- **DoubleWidth**:

Возвращаемое значение: 1 в случае успеха, 0 в случае ошибки

---

int **SetTaxesCalcMode** (int SaleTaxesOnTotal)

Переключает режим расчета наложенных налогов (разница в способе округления).

Параметры

- **SaleTaxesOnTotal**: 1 (установка по умолчанию) - наложенные налоги насчисляются со всей сумы чека. 0 - наложенные налоги начисляются на каждый товар в отдельности, а затем сумируются

Возвращаемое значение: 1 в случае успеха, 0 в случае ошибки

---

5. Информация

string **GetArticleInfo** (int Article)

Возвращает регистры учета реализации артикула

Параметры

- **Article**: номер артикула

Возвращаемое значение: Ответ ЭККА на команду ARTD

Ответ на команду содержит информацию: <c1><c2><c3><c4><c5><c6>[<c7>], где
<c1> - 4-х символьный номер запрошенного артикула (номер позиции в таблице артикулов) в формате 0000-F999
<c2> - 24 символа наименование артикула (товара).
<c3> - признак делимости
<c4> - 8 символов состояние налогообложения.
<c5> - 10 символов количество реализованного товара с данным артикулом в формате XXXXXX.XXXX.
<c6> - 10 символов общая сумма реализации товара с данным артикулом в копейках.
<c7> - 9 символов код артикула по бухгалтерской (внутрисистемной) кодировке в диапазоне ["000000001".."999999999"] (<c7> передается только в режиме работы артикульной таблицы «Регистрация новых по бухгалтерским кодам»)

Пример ответа: *0001Пакет 1A00000000000010000000000006*

---

string **GetArticleInfoXML** (int Article)

Возвращает регистр учета реализации артикула

Параметры

- **Article**: номер артикула

Возвращаемое значение: XML-строка

Пример ответа:

```
<?xml version="1.0"?>
<m301_article>
<article index="11" name="Колбаса" dividual="1" taxes="1" soldqty="10000" salesum="100000"/>
</m301_article>
```

---

int **GetBusinessDayState** ()

Позволяет определить состояние фискальной смены ЭККА

Возвращаемое значение: 0 (произошла ошибка), 1 (смена закрыта), 2 (смена открыта)

---

string **GetCashInfo** ()

Возвращает информацию о движении средств по кассе

Возвращаемое значение: Ответ ЭККА на команду CCAS

Формат ответа: <c1><c2><c3><c4><c5><c6><c7><c8>, где <c1>...<c8> - 11-ти разрядные суммы

соответственно: "Початковий Залишок", "Службове Внесення", "Службове Вилучення", "Одержано", "Видано", "Кінцевий залишок","Безготівкова оплата","Безготівкове повернення".

Пример ответа:
0000000000000000000000000000000000000000000600000000000000000000600000000000000000000000000

---

string **GetCashInfoXML** ()

Возвращает информацию о движении средств по кассе

Возвращаемое значение: XML-строка

Пример ответа:

```
<?xml version="1.0"?>
<m301_cash_info>
<cash_info rest="02655501194" income="00000000000" outcome=00000000000"
sales="00000000296" return=""00000000000" total="02655501490" check_income="00000000000"
check_outcome="00000000000"/>
</m301_cash_info>
```

---

string **GetDocumentsInfo** ()

Возвращает информацию о номерах чеков, документов, идентификаторов транзакций

Возвращаемое значение: Ответ ЭККА на команду GLCN протокола ФР (формат ответа отличается у разных моделей ЭККА)

---

string **GetDocumentsInfoXML** ()

Возвращает информацию о номерах чеков, документов, идентификаторов транзакций

Возвращаемое значение: XML-строка

Пример ответа:

```
<?xml version="1.0"?>
<m301_documents_info>
<documents_info last_check_num="50" last_doc_num="12" last_serv_doc_num="238" article_mode="2"
fiscal_report_made="0" fiscal_report_num="10" />
</m301_documents_info>
```

---

string **GetFiscalInfo** ()

Возвращает состояние дневных фискальных регистров ЭККР

Возвращаемое значение: Ответ ЭККА на команду CFIS

Ответ на команду содержит информацию: <c1><c2>...<c20>, где
<c1> - общий оборот реализации (12 цифр)
<c2>...<c8> - суммы оборотов реализации по схемам налогообложения А-3
<c9> - значение 000000000000
<c10> - не облагаемый налогом оборот реализации (12 цифр)
<c11> - общий оборот возврата (12 цифр)
<c12>...<c18> - суммы оборотов возврата по схемам налогообложения А-3
<c19> - значение 000000000000
<c20> - не облагаемый налогом оборот возврата (12 цифр)

Пример ответа:
000000000006000000000006000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000000

---

string **GetFiscalInfoXML** ()

Возвращает состояние дневных фискальных регистров ЭККА

Возвращаемое значение: XML-строка

Пример ответа:

<?xml version="1.0"?>
<m301_fiscal_info>
<fiscal_info total="000000000246" tax1_total="000000000246" tax2_total="000000000000" tax3_total="000000000000" tax4_total="000000000000" tax5_total="000000000000" tax6_total="000000000000" tax7_total="000000000000" tax8_total="000000000000" untaxed_total="000000000000" return="000000000000" tax1_return="000000000000" tax2_return="000000000000" tax3_return="000000000000" tax4_return="000000000000" tax5_return="000000000000" tax6_return="000000000000" tax7_return="000000000000" tax8_return="000000000000" untaxed_return="000000000000"/>
</m301_fiscal_info>

---

int **GetPortNumber** ()

Возвращает номер COM-порта, указаный при вызове Init/InitEx.

Возвращаемое значение: Номер COM-порта. Для TCP/IP-подключений возвращает 0

---

string **GetPrinterConfig** ()

Возвращает общую информации об ЭККА

Возвращаемое значение: Ответ ЭККА на команду CONf

Ответ на команду содержит информацию: <c1><c2>...<c19>, где
<c1> - 10 символов – последние 10 символов заводского номера (задан при изготовлении устройства).
<c2> - 10 символов - регистрационный номер (программируется командой 'INSP').
<c3> - 36 символов - наименование и адрес предприятия
<c4> - 8 символов - текущая дата (по системным часам реального времени) в формате ггггммдд;
<c5> - 6 символов - текущее время (по системным часам реального времени) в формате ччммсс;
<c6> - 1 символ - положение системного ключа
<c7> - 1 символ - признак ожидаемой команды создания документов
<c8> - 1 символ - признак зарегистрированного кассира ('0'/'1' - "не зарегистриран" / "зарегистрирован").
<c9> - 4 символа - идентификатор зарегистрированного кассира (совпадает с 4-мя символами, переданными в качестве <п2> в последней успешно исполненной команде 'UPAS').
<c10> - 1 символ - признак выполнения фискального отчета с обнулением (0/1 - "не выполнен"/"выполнен")
<c11> - 12 символов - номер фискального отчетного чека
<c12> - 12 символов - номер последнего успешно созданного (или открытого по команде OpenCheck) чека.
<c13> - 4 символа - идентификатор последней успешно исполненной команды
<c14> - 4 символа - идентификатор версии ПО ЭККР;
<c15> - 8 символов - дата создания версии ПО ЭККР в формате ггггммдд;

<c16> - 18 символов - текущая информационная строка чека
<c17> - 8 символов - дата программирования валюты ЭККР в формате ггггммдд
<c18> - 1 символ - количество знаков после десятичной точки в изображении сумм
<c19> - 3 символа - сокращенное наименование валюты ЭККР

Пример ответа: *1040022500 20130606110200831adm?
000000000000100000000001CFISUMTM20050208 ЗАВЖДИ РАДІ ВАМ 2013*

---

string **GetPrinterConfigXML** (int RefreshFromPrinter)

Возвращает общую информацию об ЭККА

Параметры

- **RefreshFromPrinter**: игнорируется *(необязательный параметр)*

Возвращаемое значение: XML-строка

Пример ответа:

```
<?xml version="1.0"?>
<m301_printer_config>
<common SerialNumber="1234567890" FiscalNumber="1111111111" Owner="<владелец>"
date="12/14/03" time="22:57:00" KeyPosition="R" MaxArticles="9999"/>
<money date="11/18/03" DecPointPosition="2" name="грн"/>
<taxes>
<tax index="0" date="11/19/03" type="1" value="20.00"/>
<tax index="1" date="11/19/03" type="2" value="30.00"/>
<tax index="2" date="11/19/03" type="1" value="10.00"/>
<tax index="3" date="11/19/03" type="1" value="06.00"/>
</taxes>
</m301_printer_config>
```

---

int **GetPrinterKeyPosition** ()

Возвращает положение системного ключа

Возвращаемое значение: Положение ключа: 0/1/2/3/4/5 - O/P/X/Z/П/*

Примечание: *Для ЭККА Мария-М304: сразу после включения возвращает значение 5, означающее, что положение ключа игнорируется.
При необходимости может быть изменено вызовом SetVirtualPrinterKeyPosition*

См. также: *SetVirtualPrinterKeyPosition*

---

string **GetPrinterSerialNumber** ()

Возвращает заводской номер ЭККА

Возвращаемое значение: 10-значный заводской номер

Пример ответа: *1234567890*

---

string **GetPrinterTime** ()

Возвращает текущее время ЭККА

Возвращаемое значение: Ответ ЭККА на команду GETD

Пример ответа: *20130606110200*

---

string **GetTaxesInfo** ()

Возвращает конфигурацию налогообложения ЭККА

Возвращаемое значение: Ответ ЭККА на команду CNAL

Пример ответа:
20130201102000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000

---

string **GetTaxesInfoXML** ()

Возвращает конфигурации налогообложения ЭККА

Возвращаемое значение: XML-строка

Пример ответа:


```
<?xml version="1.0"?>
<m301_taxes>
<taxes>
<tax index="0" date="11/19/03" type="1" value="20.00"/>
<tax index="1" date="11/19/03" type="2" value="30.00"/>
<tax index="2" date="11/19/03" type="1" value="10.00"/>
<tax index="3" date="11/19/03" type="1" value="06.00"/>
</taxes>
</m301_article>
```


---

int **GetTimeToPendingLock** ()

Возвращает время до блокировки ЭККА в секундах (сбрасывается на значение '72 часа' после каждого успешного сеанса связи с сервером ДПА). Отрицательное значение означает, что ЭККА заблокирован. Значение 999999999 означает, что ЭККА не поддерживает блокировку (старые модели)

Возвращаемое значение: Время до блокировки ЭККА в секундах

---

6. Отправка чеков на сервер ДПА

В соответствии с требованиями законодательства, ЭККА передаёт данные о всех торговых операциях на сервер ДПА. В типичной конфигурации процесс передачи данных происходит незаметно и не требует вмешательства со стороны разработчика. Для любых нетипичных конфигураций имеется возможность более тонкого контроля за процессом отправки, которые и описаны в этом разделе. Основные моменты, которые следует иметь ввиду при изменении параметров отправки:

- по умолчанию ЭККА производит попытки отправки данных на сервер каждые 6 часов
- ЭККА блокируется, если определяет, что с момента последнего успешного сеанса связи прошло более 72 часов
- имеется возможность принудительной отправки данных на сервер ДПА (см. метод Settlement)
- текущая реализация алгоритма отправки данных в ЭККА не позволяет одновременно отправлять данные и выполнять другие операции с аппаратом. При больших объёмах передаваемой информации или при низком качестве связи это может проявляться в виде задержек при выполнении операций на ЭККА

int **ConfigureAutoSettlement** (int minutes)

Настроивает процедуру автоматической отправки данных в ДПА

Параметры
- **minutes**: задаёт интервал автоматической отправки данных в ДПА в минутах

Возвращаемое значение: 1 в случае успеха, 0 в случае ошибки

Примечание: *ЭККА будет предпринимать попытку отправки данных в ДПА через указаное количество минут.*
Значение по-умолчанию: 360 (6 часов)
Значение 0 отключает автоматическую отправку данных (в этом случае данные должны передаваться вручную с помощью команды Settlement, иначе через 72 часа ЭККА заблокируется)

---

int **Settlement** ()

Отправляет данные на сервер ДПА

Возвращаемое значение: 1 в случае успеха, 0 в случае ошибки

Примечание: *Операция может занимать от нескольких секунд до нескольких минут, в зависимости от объёма отправляемых данных и качества связи*

См. также: *SettlementAsync*

---

int **SettlementAsync** ()

Инициирует фоновую отправку данных на сервер ДПА

Возвращаемое значение: Всегда возвращает 1

См. также: *Settlement* , *SettlementState* , *AbortSettlementAsync*

---

int **AbortSettlementAsync** (bool wait_idle opt)

Отменяет операцию, инициированую вызовом SettlementAsync()

Параметры
- **wait_idle**: wait_idle=false: прерывает фоновую операцию отправки и немедленно возвращает управление (действие по-умолчанию); wait_idle=true: прерывает фоновую операцию и ждёт, пока фоновая операция будет остановлена (может занять 5-30 секунд) *(необязательный параметр)*

Возвращаемое значение: Всегда возвращает 1

---

7. Авансовые платежи

---

int **OpenAdvanceCheck** (string отдел opt)

Открытие чека “заказа”. Прием авансового платежа.

Параметры
- **отдел**: название отдела *(необязательный параметр)*

Возвращаемое значение: 1 в случае успеха, 0 в случае ошибки

Дригие названия: *ОткрытьЧекАвансовогоПлатежа,ОткрытьЧекЗаказа*

Примечание: *Поддерживается в ЭККА 304T-2, 304T1-2, 304T2-2*

---

int **OpenAdvanceCompletionCheck** (string отдел opt)

Открытие чека “выполнения заказа”. Расчет по авансовому платежу.

Параметры

- **отдел**: название отдела *(необязательный параметр)*

Возвращаемое значение: 1 в случае успеха, 0 в случае ошибки

Дригие названия: *ОткрытьЧекЗавершенияЗаказа,ОткрытьЧекЗавершенияАвансовогоПлатежа*

Примечание: *Поддерживается в ЭККА 304T-2, 304T1-2, 304T2-2*

---

int **OpenAdvanceReturnCheck** (string отдел opt)

Открытие чека “возврата аванса”. Расчет по авансовому платежу

Параметры

- **отдел**: название отдела *(необязательный параметр)*

Возвращаемое значение: 1 в случае успеха, 0 в случае ошибки

Другое название: *ОткрытьЧекВозвратаАванса*

Примечание: *Поддерживается в ЭККА 304T-2, 304T1-2, 304T2-2*

---

int **AdvanceLine** (string наименование , int цена , decimal количество opt)

Добавляет строку товара/услуги в чек авансового платежа

Параметры

- **наименование**: название товара/услуги
- **цена**: цена
- **количество**: количество в штуках *(необязательный параметр)*

Возвращаемое значение: 1 в случае успеха, 0 в случае ошибки

Другое название: *СтрокаАвансовогоПлатежа*

Примечание: *Поддерживается в ЭККА 304T-2, 304T1-2, 304T2-2.*
Используется только в авансовых чеках заказа (см.OpenAdvanceCheck)

---

int **PrintAdvanceTotals** (string дополнительная_информация opt)

Печатает отдельной строкой сумму всех предыдущих позиций авансового чека (см.AdvanceLine)

Параметры

- **дополнительная_информация**: дополнительная текстовая информация. Например, дата исполнения *(необязательный параметр)*

Возвращаемое значение: 1 в случае успеха, 0 в случае ошибки

Другое название: *ПечатьИтоговойСуммыЗаказа*

Примечание: *Поддерживается в ЭККА 304Т-2, 304Т1-2, 304Т2-2.*
Используется только в авансовых чеках заказа (см.OpenAdvanceCheck)

---

int **AdjustTotals** (int сумма , int тип_операции , string описание opt , int налог1 opt , int налог2 opt)

Изменяет сумму оплаты по чеку и печатает соответствующую информацию в чек

Параметры

- **сумма**: сумма (-999999999...+999999999)
- **тип_операции**: 1..6
- **описание**: текстовое описание операции *(необязательный параметр)*
- **налог1**: идентификатор первой схемы налогообложения(1..8) или 0 *(необязательный параметр)*
- **налог2**: идентификатор второй схемы налогообложения(1..8) или 0 *(необязательный параметр)*

Возвращаемое значение: 1 в случае успеха, 0 в случае ошибки

Другое название: *УчётПлатежа*

Примечание: *Поддерживается в ЭККА 304Т-2, 304Т1-2, 304Т2-2.*
После выполнения операции сумма чека увеличится/уменьшится на значение, указаное в параметре "сумма"
В зависимости от значения параметров "типа_операции" и "сумма", в чеке будет напечатано:
если "сумма" > 0:

- тип_операции = 1: (недопустимое значение)
- тип_операции = 2: "передплата на бонусний рахунок"
- тип_операции = 3: "подарунковий сертифікат"
- тип_операции = 4: "передплата на рахунок клубної картки"
- тип_операции = 5: "аванс" (в чеках авансовых платежей), "попередньо сплачений аванс" (в чеках возврата аванса)
- тип_операции = 6: недопустимое значение

если "сумма" < 0:

- тип_операции = 1: "знижка"
- тип_операции = 2: "попередньо сплачені бонуси"
- тип_операции = 3: "подарунковий сертифікат"
- тип_операции = 4: "оплата з рахунку клубної картки"
- тип_операции = 5: "попередньо сплачений аванс" (в чеках выполнения заказа)
- тип_операции = 6: "отримано товарів" (в чеках возврата аванса)

Параметр тип_операции=6 (отримано товарів) допустим только в чеках возврата аванса (см. OpenAdvanceReturnCheck), параметр тип_операции=5 (аванс) - только в чеках авансовых платежей (см. OpenAdvanceCheck, OpenAdvanceReturnCheck, OpenAdvanceCompletionCheck), параметр тип_операции=1 (скидка) может использоваться только с отрицательными суммами.

Вместо числовых значений в параметре "тип_операции" можно использовать значения, возвращаемые следующими свойствами COM-объекта:

1. Discount, Скидка, Знижка
2. BonusAccount, Bonuses, БонусныйСчёт, БонуснийРахунок, Бонусы, Бонуси
3. GiftCertificate, GiftCard, ПодарочныйСертификат, ПодарунковийСертифікат
4. ClubCard, КлубнаяКарта, КлубнаКартка
5. Advance, Аванс
6. GoodsSold, ПолученоТоваров, ОтриманоТоварів

Примеры:
m304.AdjustTotals(-100, m304.BonusAccount, "№12345", 1);
m304.УчётПлатежа(-100, m304.КлубнаяКарта, "№23456", 1)

---

8. Разное

---

int **ClearLogo** ()

Отменяет печать логотипа на чеках

Возвращаемое значение: 1 в случае успеха, 0 в случае ошибки

---

int **CloseTextDocument** ()

Закрывает и печатает служебный документ

Возвращаемое значение: 1 в случае успеха, 0 в случае ошибки

---

int **Feed** (int LinesToFeed)

Осуществляет прогон чековой ленты

Параметры

- **LinesToFeed**: число от 0 до 65535 – количество шагов двигателя протяжки чековой ленты (шаг 0,125 мм)

Возвращаемое значение: 1 в случае успеха, 0 в случае ошибки

---

int **OpenBusinessDay** ()

Принудительно открывает новую фискальную смену.

Возвращаемое значение: 1 в случае успеха, 0 в случае ошибки

Примечание: *Обычно фискальная смена открывается автоматически при выполнении фискальной операции, потому использовать данную функцию для открытия смены необязательно.*
Данную функцию следует использовать, когда необходимо явно открыть смену

---

int **OpenCashBox** ()

Открывает денежный ящик

Возвращаемое значение: 1 в случае успеха, 0 в случае ошибки

---

int **OpenGoodsGroup** (string название)

Открывает группу товарных позиций

Параметры

- **название**: название группы

Возвращаемое значение: 1 в случае успеха, 0 в случае ошибки

String[] **Passthrough** (string cmd)

Передаёт команду низкоуровневого протокола для обработки непосредственно в ЭККА

Параметры

- **cmd**: команда протокола ЭККА

Возвращаемое значение: Ответ ЭККА на команду, в соответствии с протоколом ЭККА (массив строк)

Пример ответа: *["WAIT", "SYNC123", "DONE", "READY"]*

---

int **PutSumToDisplay** (int sum , int kind opt)

Выводит сумму на встроеный дисплей ЭККА

Параметры

- **sum**: сумма
- **kind**: тип выводимого числа: 1/2/3 - цена/сумма/сдача *(необязательный параметр)*

Возвращаемое значение: 1 в случае успеха, 0 в случае ошибки

---

int **PutToDisplay** (string Line1 , string Line2 opt)

Выводит сумму на встроеный дисплей ЭККА

Параметры

- **Line1**: одно из следующих значений: "ВАРТІСТЬ"/"СУМА"/"ЗДАЧА"
- **Line2**: сумма для отображения *(необязательный параметр)*

Возвращаемое значение: 1 в случае успеха, 0 в случае ошибки

Примечание: *Функцию также можно вызывать с одним параметром вместо двух, тогда первый параметр считается разным "СУМА"*

---

int **PutToExternalDisplay** (string LineText)

Выводит текст на выносной дисплей ЭККА

Параметры

- **LineText**: текст

Возвращаемое значение: 1 в случае успеха, 0 в случае ошибки

---

int **SetLogo** (Byte[] image , bool invert opt)

Загружает логотип в ЭККА из памяти

Параметры

- **image**: логотип
- **invert**: инвертировать цвета *(необязательный параметр)*

Возвращаемое значение: 1 в случае успеха, 0 в случае ошибки

Примечание: *Поддерживаемые форматы: bmp, gif, jpeg, png*
Если размер изображения превышает 432x192, оно будет уменьшено

---

int **SetLogoFromFile** (string path , bool invert opt)

Загружает логотип в ЭККА из файла

Параметры
- **path**: путь к файлу логотипа
- **invert**: инвертировать цвета *(необязательный параметр)*

Возвращаемое значение: 1 в случае успеха, 0 в случае ошибки

Примечание: *Поддерживаемые форматы: bmp, gif, jpeg, png*
Если размер изображения превышает 432x192, оно будет уменьшено

---

int **SetVirtualPrinterKeyPosition** (int p)

Устанавливает положение виртульного ключа ЭККА М304

Параметры
- **p**: новое положение ключа: 0/1/2/3/4 - О/Р/Х/Z/П

Возвращаемое значение: 1 в случае успеха, 0 в случае ошибки

Примечание: *Работает только с ЭККА Мария М304*

---

4. Свойства

int **ArticleMode**

Режим артикульной таблицы ЭККА

---

int **CheckOnSaleTaxes**

Сумма наложенных налогов по текущему (открытому с помощью OpenCheck) фискальному чеку реализации

Примечание: *Для ЭККА Мария М304Т это значение будет всегда нулевым*

---

int **CheckReturnOnSaleTaxes**

Сумма наложеных налогов по текущему фискальному чеку возврата

Примечание: *Для ЭККА Мария М304Т это значение будет всегда нулевым*

---

int **CheckReturnSum**

Сумма возврата по текущему фискальному чеку возврата без наложенных налогов

---

int **CheckSum**

Сумма реализации по текущему фискальному чеку, без наложеных налогов

---

int **FiscalLines**

Возвращает количество фискальных позиций в текущем открытом чеке

---

int **Hardware**

Возвращает тип ЭККА (значения: 301 для M301, 304 для M304)

---

int **LastCheckNumber**

Номер последнего распечатаного фискального чека

---

int **LastDocumentNumber**

Номер последнего распечатаного служебного документа.

Примечание: *Может соответствовать документу, распечатаному ЭККА по собственной инициативе (например, документ включения питания)*

---

string **LastErrorCode**

Мнемоника сообщение об ошибке - соответствует мнемоникам ошибок, возвращаемым ЭККА.
Значение обнуляется после прочтения. В случае отсутствия ошибок возвращается null.
Добавлены также следующие мнемоники ошибок:
TOOBIGQTY - Слишком большое количество товара в строке чека
INVALIDTAX - недопустимая налоговая ставка

---

string **LastErrorMessage**

Возвращает строку с сообщением о последней произошедшей ошибке. Это же сообщение выводится на экран, если ShowErrorMessages=1.
Значение обнуляется после прочтения.

---

int **LastLineTaxes**

Сумма наложеных налогов в последней строке чека

---

int **LastTextDocumentNumber**

Номер последнего документа, распечатаного с помощью OpenTextDocument()...CloseTextDocument()

---

string **LogFile**

Задаёт имя файла для ведения логов (имеет более высокий приоритет, чем аналогичная настройка, заданая через штатную утилиту настройки)

---

int **NextCheckNumber**

Номер следующего фискального чека (LastCheckNumber+1)

---

int **NextDocumentNumber**

Номер следующего служебного документа

---

int **NextZNumber**

Номер следующего z-отчёта (номер текущей фискальной смены)

---

string **PaymentTerminal**

Строка соединения с платёжным терминалом. Свойство предназначено для переопределения глобальных параметров связи с терминалом

Строка соединения имеет вид:
1. device=<тип>;port=<порт>
2. device=<тип>
3. <тип>

Параметр <тип> может принимать значения: ingenico, verifone, emulator
Параметр <порт> обязателен для терминалов verifone

Пример: *Device=ingenico*
device=verifone;port=COM5
ingenico
emulator

---

int **SerialPortRate**

Задаёт скорость подключения к ЭККА через последовательный порт

Примечание: *Установленое значение будет использоваться при подключении к ЭККА через COM-порт при следующем вызове Init.*
Допустимые значения: 4800,9600,19200,38400,57600,115200 (по-умолчанию: 115200)

---

int **SettlementState**

Возвращает текущее состояние операции фоновой отправки данных на сервер ДПА.
Значения: 0 (операция SettlementAsync не запускалась), 1 (операция в процессе выполнения), 2 (операция завершена успешно), 3 (операция завершена с ошибкой), 4 (операция отменена вызовом AbortSettlementAsync)

---

int **ShowErrorMessages**

Включает/выключает вывод на экран сообщений об ошибках

---

int **ShowStatusDialogs**

Отображение состояния длительно выполняющихся операций

---

bool **ThrowExceptionsOnError**

Управляет поведением при возникновении ошибок.
При установке значения ThrowExceptionsOnError в false (значение по-умолчанию), ошибки при выполнении методов приводят к возврату значения '0'

При установке значения ThrowExceptionsOnError в true, ошибки при выполнении методов приводят к генерации исключений

---

bool **VerboseErrorMessages**

Включает/выключает расширеную информацию об ошибках

---

string **Version**

Возвращает версию ole-сервера

Пример: *4.0.20130125*

---

5. # Печать чеков с использованием терминала безналичной оплаты

Существует два способа печати чеков с использованием терминала безналичной оплаты, один простой, второй ~~сложный~~ универсальный.

1. **Простой способ**
   Этот способ позволяет печатать чеки практически так же, как обычные чеки безналичной оплаты, при закрытии чека olemanager автоматически проведёт операцию оплаты на терминале.
   Поддерживаемые терминалы: ingenico ict220 (Банкомсвязь) и verifone vx510 (Inpas)

   Последовательность действий:
   1. открыть чек с помощью функции OpenCheckEx("название отдела", True)
   2. добавить в чек фискальные позиции с помощью функций FiscalLine и/или FiscalLineEx
   3. закрыть чек с помощью CheckCheckEx . При выполнении этой функции будет произведена оплата через платёжный терминал, на ЭККА будет распечатан документ платёжного терминала и распечатан фискальный чек. Процесс оплаты будет отображаться в открывшемся окне
      - Чек, должен закрываться с помощью метода CheckCheckEx , один из параметров безналичной оплаты должен быть ненулевым.
      - Параметр RRN метода CloseCheckEx следует указывать только для чека возврата

   Для терминала ingenico ict220 (Банкомсвязь):
   - В системе должен быть установлен COM-сервер ECRCommX.BPOS1Lib
   - тип терминала должен быть выбран в конфигурации olemanager (пуск/программы/resonance jsc/olemanager/resonance olemanager 2015/платёжный терминал)

   Для терминала verifone vx510 (Inpas):
   - тип терминала должен быть выбран в конфигурации olemanager (пуск/программы/resonance jsc/olemanager/resonance olemanager 2015/платёжный терминал)

   Также имеется возможность указать тип терминала программно (см. свойство PaymentTerminal ole-сервера)

   Преимущества: простота реализации (минимальные отличия от штатной процедуры печати чеков)
   Недостатки: работает только с двумя типами терминалами, отсутствует возможность изменения последовательности печати чеков или окна состояния

2. **Универсальный способ**
   Работа с терминалом безналичной оплаты происходит без участия olemanager, olemanager используется только для печати чека и документа "квитанция платёжного терминала"

   Последовательность действий:

1. выполнить транзакцию на банковском терминале, после выполнения оплаты запомнить RRN и, при необходимости, текст чека (Slip)
2. распечатать квитанцию платёжного терминала через olemanager, используя функцию PrintSlip(receipt)
3. распечатать фискальный чек, используя последовательность команд OpenCheck-FiscalLine-...-CloseCheckEx(sum1, sum2, sum3, sum4, RRN), параметр RRN должен иметь значение, возвращённое платёжным терминалом
4. Распечатать квитанцию платёжного терминала ещё раз, использую PrintSlip(receipt)

Преимущества: полный контроль над печатью чека, возможность работы с любым платёжным терминалом
Недостатки: требуется ручная работа с платёжным терминалом

6. Чтение электронного журнала чеков (КЛЭФ/КСЕФ)

КЛЭФ (контрольная лента в электронной форме, укр. КСЕФ - контрольна стрічка в електронній формі) - подсистема ЭККА, обеспечивающая хранение фискальных чеков и прочих документов в электронной форме. ЭККА "Мария М304" хранит все распечатаные фискальные чеки, а также z-отчёты и служебные документы внесения/изъятия наличных на внутреннем носителе информации и предоставляет средства для чтения этих данных в формате XML. OLEManager позволяет читать данные КЛЭФ в формате XML, а также предоставляет средства разбора полученых XML-документов.
Для работы с КЛЭФ в **M304Manager.Application** имеется свойство **Journal** (реализует COM-интерфейс IDispatch), у которого определены следующие методы:

- **FindFirstXML(date)** - ищет в КЛЭФ первый документ за указаную дату (дата задаётся объектом типа VT_DATE или строкой в формате региональных настроек операционной системы). Возвращает XML-строку или NULL, если за указаную дату документов не найдено.
- **FindNextXML()** - ищет следующий документ в КЛЭФ. Возвращает XML-строку или NULL, если достигнут конец КЛЭФ
- **FindFirst(date)** - аналогично FindFirstXML, но возвращает объект OLE-automation.
- **FindNext()** - аналогично FindNextXML, но возвращает объект OLE-automation.

Например, для поиска документов за 1 декабря 2014г. можно использовать конструкции вида:

```
xml = m304.Journal.FindFirstXML("01.12.2014")
doc = m304.Journal.FindFirst("01.12.2014")
```

См. также пример "Чтение чеков КЛЭФ на php"
Также у объекта **M304Manager.Application** есть свойства **КЛЭФ** и **КСЕФ**, которые являются синонимами для **Journal**.

Функции **FindFirstXML**/**FindNextXML** возвращают XML-строку с содержимым документа, функции **FindFirst**/**FindNext** возвращают объект OLE-automation со следующими свойствами:

- **Id** - внутренний идентицификатор документа в КЛЭФ
- **DocumentType** - тип документа (значения: 1 - фискальный чек реализации, 2 - фискальный чек возврата, 3 - Z-отчёт, 4 - документ служебного внесения денег в кассу, 5 - документ служебного изъятия денег из кассы)
- **Timestamp** - дата/время документа
- **RawXML** - содержимое документа в XML-формате (также возвращается командами **FindFirstXML**/**FindNextXML**)

В зависимости от значения поля **DocumentType**, в OLE-документе также присутствуют дополнительные свойства:

1. Фискальный документ реализации (**DocumentType**=1) или возврата (**DocumentType**=2)
   - **Number** - номер фискального чека
   - **Count** - количество фискальных позиций
   - **Sum** - Сумма оплаты по чеку
   - **Cash** - Сумма оплаты по чеку наличными
   - **Payments** - Сумма оплаты по чеку различными формами оплаты (**Payments[0]** - оплата наличными, **Payments[1]** - оплата по форме оплаты "безнал1" и т.д.)
   - **ReferencedCheckNumbers** - номер чека, по которому был возврат

Также можно получить доступ к отдельным фискальным строкам документа через индексатор (напр. doc[1] - первая фискальная строка чека, doc[doc.Count] - последняя)
Фискальные позиции чека имеют следующие свойства:

- **Id** - номер позиции
- **Name** - наименование товара
- **Code** - код артикула
- **Quantity** - количество товара
- **Price** - цена
- **Sum** - сумма (с учётом скидки)
- **SumWithoutDiscount** - сумма по позиции без учёта скидки, Price*Quantity
- **Discount** - сумма скидки (в случае надбавки значение будет отрицательным)
- **SumWithDiscount** - сумма (с учётом скидки)

2. Z-отчёт (**DocumentType**=3)
   - **Number** - номер Z-отчёта
   - **FiscalCheckCount** - количество фискальных чеков за смену
   - **SaleTotals** - общий оборот реализации за смену по формам оплаты
     Например, **SaleTotals[0]** - общий оборот реализации за наличный расчёт, SaleTotals[1] - общий оборот реализации по форме оплаты "безнал1" и т.д.
   - **ReturnTotals** - общий оборот возврата за смену по формам оплаты
     Например, **ReturnTotals[0]** - общий оборот возврата за наличный расчёт, **ReturnTotals[1]** - общий оборот возврата по форме оплаты "безнал1" и т.д.
3. Служебный документ внесения/изъятия наличных (**DocumentType**=4 и **DocumentType**=5)
   - **Sum** - сумма операции

Все описаные в этом разделе OLE-объекты также имеют метод **ToString()**, возвращающий сжатое текстовое описание содержимого объекта.
См. также пример "Чтение чеков КЛЭФ на php"

7. Примеры

Печать чека из 1С

```
o = Новый COMОбъект("M304Manager.Application");
o.ThrowExceptionsOnError = True;
Попытка
        o.InitAuto();
        o.OpenCheck();
        o.FiscalLine("тест", 1, 100, 0, 1, 0, 1111);
        o.CloseCheck();
        Сообщить("Чек распечатан");
Исключение
        Сообщить("Ошибка печати чека: " + o.LastErrorMessage);
КонецПопытки;
o.Done();
```

Печать служебного документа на C#

```
using System;

namespace hello
{
    class world
    {
        static void Main()
        {
            using (var m304 = new Resonance.M304ManagerApplication { ThrowExceptionsOnError = true })
            {
                try
                {
                    m304.InitAuto();
                    m304.OpenTextDocument();
                    m304.FreeTextLine("hello, world", doubleWidth: true, doubleHeight: true);
                    m304.CloseTextDocument();
                    Console.WriteLine("Документ распечатан");
                }
```

```
                catch
                {
                    Console.WriteLine("Ошибка печати документа");
                }
            }
        }
    }
}
```

Печать X-отчёта на JScript

```
var o = new ActiveXObject("M304Manager.Application");
o.ShowErrorMessages = 1;
if (o.InitAuto() == 1)
{
        var r = o.XReport();
        o.Done();
        WScript.echo("M304Manager.Application.XReport() = " + r);
}
```

Определение заводского номера ЭККА на Delphi

```
program Project2;

{$APPTYPE CONSOLE}

uses
  SysUtils, ComObj, ActiveX;
var m304 : Variant;
begin
    CoInitialize(nil);
    m304 := CreateOleObject('M304Manager.Application');
    m304.InitAuto;
    Writeln('S/N: ', m304.GetPrinterSerialNumber);
    m304.Done;
end.
```

Чтение чеков КЛЭФ на Python

```
import win32com.client

m=win32com.client.Dispatch("M304Manager.Application")
m.Init(6)
sn = m.GetPrinterSerialNumber
print("S/N: %s" % sn)
doc = m.Journal.FindFirst("03.12.2014")
while (doc != None):
    print(doc.ToString)
    doc = m.Journal.FindNext

m.Close
```

Чтение чеков КЛЭФ на PHP

```
<?php
$m = new COM("M304Manager.Application");
$m->LogFile = "logfile.log";
$m->ShowErrorMessages = 1;
$m->InitAuto();
$doc = $m->Journal->FindFirst("03.12.2014");
while($doc != NULL)
{
  printf("Document #%s: %s\r\n", $doc->Id, $doc->ToString());
  switch ($doc->DocumentType)
  {
    case 1:
    case 2:
          printf("\t#%s, sum:%s (%s lines)\r\n", $doc->Number,  $doc->Sum, $doc->Count);
          for ($i=1; $i <= $doc->Count; $i++)
          {
            $line = $doc[$i];
            printf("\t%s\r\n", $line->ToString());
          }
      break;
      case 3:
```

```
        printf("\tZ#%s: %s checks\r\n", $doc->Number, $doc->FiscalCheckCount);
        printf("\t  готівка:  +%s  -%s\r\n", $doc->SaleTotals[0], $doc->ReturnTotals[0]);
        printf("\t  безготівка.1: +%s  -%s\r\n", $doc->SaleTotals[1], $doc->ReturnTotals[1]);
        printf("\t  безготівка.2: +%s  -%s\r\n", $doc->SaleTotals[2], $doc->ReturnTotals[2]);
        printf("\t  безготівка.3: +%s  -%s\r\n", $doc->SaleTotals[3], $doc->ReturnTotals[3]);
        break;
      case 4:
        printf("\t внесение: %s\r\n", $doc->Sum);
        break;
      case 5:
        printf("\t изъятие: %s\r\n", $doc->Sum);
        break;
  }

  $doc = $m->Journal->FindNext();
}

$m->Close();
?>
```

Авансовые платежи из 1С

```
m = new COMObject("M304Manager.Application");
m.Init("COM2");

//оплата клубной картой, бонусами + скидка на чек
m.OpenCheck("№1");
m.FiscalLine("пирожок", 1, 1000, 0, 1);
m.FiscalLine("булочка", 1, 1000, 0, 1);
m.AdjustTotals(-500, m.ClubCard, "№4567", 1);
m.AdjustTotals(-234, m.Bonuses, "№12345", 1);
m.AddDiscount(100, "акция", 1);
m.AddPayment(1200);
m.CloseCheck();

//авансовый платёж (используются русскоязычные синонимы для имён функций)
m.ОткрытьЧекАвансовогоПлатежа("№2");//или m.OpenAdvanceCheck("№2")
m.СтрокаАвансовогоПлатежа("товар1", 1234);
m.СтрокаАвансовогоПлатежа("товар2", 2345, 1.234);//или m.AdvanceLine("товар2", 2345, 1.234)
m.ПечатьИтоговойСуммыЗаказа();
m.УчётПлатежа(+1200, m.Аванс, null, 1);//или m.AdjustTotals(+1200, m.Advance, null, 1);
m.AddPayment(1000, 3);
m.CloseCheck();

//завешение авансового платежа
m.OpenAdvanceCompletionCheck("№2");
m.FiscalLine("товар1", 1, 1234, 0, 1);
m.FiscalLine("товар2", 1, 2345, 0, 1);
m.AdjustTotals(-1200, m.Advance, "за чеком №12345", 1);
m.AdjustTotals(-1000, m.GiftCard, "№123456", 1);
m.AddPayment(1500);
m.CloseCheck();

//возврат аванса
m.ОткрытьЧекВозвратаАванса("№3");
m.AdjustTotals(+10000, m.Аванс, "за чеком №7890", 1);
m.AdjustTotals(-5678, m.ПолученоТоваров, "за чеком №8901", 1);
m.CloseCheck();

//покупка подарочного сертификата + акциз(ставка №4(Г))
m.OpenCheck("№4");
m.SetDoubledTaxCalcMode(4, 1);
m.AdjustTotals(+10000, m.GiftCertificate, "№222333", 4, 1);
m.SetDoubledTaxCalcMode(4, 1);
m.AdjustTotals(+10000, m.ClubCard, "№333444", 4, 1);
m.SetDoubledTaxCalcMode(4, 1);
m.AdjustTotals(-5000, m.Bonuses, "№444555", 4, 1);
m.AddPayment(10000, 3);
m.AddPayment(10000, 0);
m.CloseCheck();
m.Done();
```

8. Список изменений

v4.1.20140325

- понижены требования к версии .net framework: текущие требования - .Net Framework v3.5-4.5
- оптимизация работы в терминальной сессии (rdp)
- возможность использовать библиотеку olemanager2013.dll как обычную .Net-сборку, без необходимости регистрации COM-сервера
- Добавлены функции: OpenBusinessDay , CloseBusinessDay , CloseBusinessDayAsync , GetBusinessDayState , PrintBarcode , Hardware , GetTimeToPendingLock , ConfigureAutoSettlement ,

Settlement , SettlementAsync , SettlementState , AbortSettlementAsync

v4.1.20141209

- добавлена возможность чтения чеков с электронной контрольной ленты (для аппаратов с КЛЭФ)
- добавлена возможность программного управления логированием
- добавлены свойства/методы: LastCheckNumber , NextCheckNumber , LastDocumentNumber , NextDocumentNumber , LastTextDocumentNumber , NextZNumber , ArticleMode , LogFile , GetBusinessDayState , AddPayment .
- добавлены примеры

v4.1.20150115

- Исправлена печать фискальных позиций с двумя схемами налогообложения (метод SetDoubledTaxCalcMode возвращал ошибку, если чек открыт с помощью OpenCheckEx)

v4.1.20150206

- Добавлена возможность отображения длительно выполняющихся операция (свойство ShowStatusDialogs и параметр в методе Init)

v4.1.20150522

- Добавлена возможность работы с РРО по TCP/IP-протоколу (Мария 304Т-2, 304Т1-2, 304Т2-2) (см. изменения в методе Init)

v4.1.20150602

- Добавлены функции для управления логотипом в чеках (SetLogoFromFile , SetLogo , ClearLogo)

v4.1.20150612

- Доработана поддержка работы с РРО по TCP/IP, обновлен пример обработка для 1С (см. файл Samples\Resonance_Maria304_v1.2.epf в папке с установленой программой)

v4.1.20150616

- Добавлена возможность автоназначения кодов для артикулов, см. параметр Article методов FiscalLine и FiscalLineEx (требуется для работы с 1С 8.3 и новых конфигураций 1С 8.2)
- Обновлена обработка 1С для работы с акцизными налогами (см. файл samples\Resonance_Maria304_v4.10.epf в папке с установленой программой)

v4.1.20150622

- Исправлена функция InitAuto и кнопка "Тест" в служебной утилите (ошибка появилась в v.4.1.20150522)

v4.1.20150624

- Добавлена возможность указывать скорость COM-порта при подключении к ЭККА (см. SerialPortRate)

v4.1.20150706

- Добавлена возможность задания скидок на весь чек (см. AddDiscount)
- Добавлены функции работы с авансовыми платежами(см. OpenAdvanceCheck , OpenAdvanceCompletionCheck , OpenAdvanceReturnCheck , AdvanceLine , PrintAdvanceTotals , а также пример использования)
- Добавлена возможность оплаты бонусами, клубными картами, подарочными сертификатами (см. AdjustTotals и пример использования)

Для работы этих функций требуется ЭККА 304Т-2, 304Т1-2, 304Т2-2

v4.1.20150727

- Добавлены фунция для печати произвольных одномерных/двумерных кодов (см. UploadImage , LoadImageFrom , PrintImage)

- Добавлена фунция для печати QR-кодов (см. PrintQR) как пример использования функций UploadImage/PrintImage

Для работы этих функций требуется ЭККА 304T-2, 304T1-2, 304T2-2

v4.1.20160719

- Добавлена функция SetCheckTotals для дополнительного контроля над операциями печати фискальных чеков

v4.1.20170123

- Исправлено: метод Done при определённых условиях вызов мог проводить к зависанию
- Исправлено: в конфигурационной утилите не работала кнопка "Тест", если использовался проброс ком-портов в терминальную сессию windows 2008/2012/2016
- Добавлены свойства CheckSum и FiscalLines для получения информации о текущем открытом чеке

v4.1.20170208

- Исправлена функция FiscalLineEx (ошибка появилась в версии 4.1.20170123)