|  | **Версия 1.16.1** |
| --- | --- |

ПИРИТ: JPOS драйвер

**Руководство пользователя**

Версия: 1.16.1

Дата: 19.08.2025

**Вступление**

Данная документация описывает особенности реализации драйвера и не является руководством по стандарту UPOS. (Со стандартом UPOS можно ознакомиться [здесь](https://nrf.com/resources/retail-technology-standards/unifiedpos)).

**Методы PiritSvc112:**

Основные методы класса PiritSvc112 implements FiscalPrinterService112

*В скобках указан hex номер команды соответствующей из документа «****Инструкция по программированию ККТ Пирит 2Ф\_ОФД****».*

* + 1. **Общие**

**DirectIO** – см. стр. 18

**CheckHealth**

1.1.1 Запрос флагов статуса ККМ (0x00)

**ResetPrinter**

1.1.2 Завершить документ (0x31)

1.1.3 Аннулировать документ (0x32)

**SetTrailerLine**

1.1.3Запись таблицы настроек (0x12)

**GetDate**

1.1.4Чтение даты/времени ККМ (0x13)

**SetDate**

1.1.5Запись даты/времени ККМ (0x14)

**GetData**

1.1.6 Запрос сменных счетчиков и регистров (0x01)

**GetTotalizer**

1.1.7 Запрос сменных счетчиков и регистров (0x01)

1.1.8 Запрос сведений о ККМ (0x02)

1.1.9 Запрос данных по чеку (0x03)

**GetVatEntry**

1.1.10Чтение таблицы настроек (0x11)

**SetVatTable**

1.1.11Запись таблицы настроек (0x12)

**SetVatValue**

1.1.12Подготовка внутренних данных для вызова SetVatTable

**OpenDrawer**

1.1.13 Открыть денежный ящик (0x80)

**DrawerOpened**

1.1.14 Получить статус денежного ящика (0x81)

* + 1. **Регистрация чека**

**BeginFiscalReceipt**

1.2.1 Открыть документ (0x30)

**PrintRecTotal**

1.2.2 Внесение / изъятие оплаты (0x47)

*Выполняется внесение/изъятие одной командой*

**PrintRecCash**

1.2.3 Внесение / изъятие суммы (0x48)

*Выполняется внесение/изъятие после вызова* ***BeginFiscalReceipt*** *и установленном свойстве* ***FiscalReceiptType*** *в значения FPTR\_RT****\_****CASH\_IN или FPTR\_RT\_CASH\_OUT*

**PrintRecItem**

1.2.4 Добавить товарную позицию (0x42)

Параметры:

Description – наименование товара;

Price – цена за позицию (не используется);

Quantity – количество. Значение 0 эквивалентно 1000;

VatInfo – индекс налоговой ставки значения:

0 - Ставка "НДС 20%"

1 - Ставка "НДС 10%"

2 - Ставка "НДС 0%"

3 - Ставка "без НДС"

4 - Ставка "НДС 20/120"

5 - Ставка "НДС 10/110"

6 - Ставка "НДС 5%"

7 - Ставка "НДС 7%"

8 - Ставка "НДС 5/105"

9 - Ставка "НДС 7/107"

UnitPrice – цена единицы товара;

UnitName – единицы измерения.

**PrintRecSubtotal**

1.2.5 Подитог (0x44)

**PrintRecItemAdjustment**

1.2.6 Переданные в метод параметры сохраняются в драйвере и применяются к следующему

вызову PrintRecItem после чего обнуляются.

**PrintRecSubtotalAdjustment**

Сначала вызывается 3.4.12 Подитог (0x44)

1.2.7 Скидка на товар/чек (0x45)

1.2.8 Наценка на товар/чек (0x46)

**PrintRecVoid**

1.2.9 Завершить документ (0x31)

1.2.10 Аннулировать документ (0x32)

**EndFiscalReceipt**

1.2.11 Завершить документ (0x31)

* + 1. **Печать текста**

**BeginNonFiscal**

1.2.12 Открыть документ (0x30)

*Тип - Сервисный документ*

**PrintNormal**

1.2.13 Печать текста (0x40)

*Настройка печати текста/штрихкода см. методы DirectIO*

**EndNonFiscal**

1.2.14 Завершить документ (0x31)

* + 1. **Печать отчетов**

**PrintPeriodicTotalsReport**

1.2.15 Распечатать фискальный отчет по датам (0x62)

*Параметр «Тип отчета» имеет значение 1 (полный)*

**PrintReport**

1.2.16 Распечатать фискальный отчет по сменам (0x61)

*Параметр «Тип отчета» имеет значение 1 (полный)*

*Так же если параметры StartNum=0 и EndNum=0 печатается*

*копия последнего Z-отчета:*

*1.2.17 Системные команды –> Печать копии последнего Z-отчета (0xA1)*

**PrintXReport**

1.2.18 Распечатать отчет без гашения (X-отчет) (0x20)

**PrintZReport**

* + 1. Распечатать отчет с гашением (Z-отчет) (0x21)

**Методы спецификации JPOS:**

public interface BaseService

**{**

// Properties

public String getCheckHealthText**()** **throws** JposException**;**

public boolean getClaimed**()** **throws** JposException**;**

public boolean getDeviceEnabled**()** **throws** JposException**;**

public void setDeviceEnabled**(**boolean deviceEnabled**)**

**throws** JposException**;**

public String getDeviceServiceDescription**()** **throws** JposException**;**

public int getDeviceServiceVersion**()** **throws** JposException**;**

public boolean getFreezeEvents**()** **throws** JposException**;**

public void setFreezeEvents**(**boolean freezeEvents**)** **throws** JposException**;**

public String getPhysicalDeviceDescription**()** **throws** JposException**;**

public String getPhysicalDeviceName**()** **throws** JposException**;**

public int getState**()** **throws** JposException**;**

// Methods

public void claim**(**int timeout**)** **throws** JposException**;**

public void close**()** **throws** JposException**;**

public void checkHealth**(**int level**)** **throws** JposException**;**

public void directIO**(**int command**,** int**[]** data**,** Object object**)**

**throws** JposException**;**

public void open**(**String logicalName**,** EventCallbacks cb**)**

**throws** JposException**;**

public void release**()** **throws** JposException**;**

}

public interface FiscalPrinterService13 **extends** BaseService

**{**

// Capabilities

public boolean getCapAdditionalLines**()** **throws** JposException**;**

public boolean getCapAmountAdjustment**()** **throws** JposException**;**

public boolean getCapAmountNotPaid**()** **throws** JposException**;**

public boolean getCapCheckTotal**()** **throws** JposException**;**

public boolean getCapCoverSensor**()** **throws** JposException**;**

public boolean getCapDoubleWidth**()** **throws** JposException**;**

public boolean getCapDuplicateReceipt**()** **throws** JposException**;**

public void setDuplicateReceipt**(**boolean duplicateReceipt**)** **throws** JposException**;**

public boolean getCapFixedOutput**()** **throws** JposException**;**

public boolean getCapHasVatTable**()** **throws** JposException**;**

public boolean getCapIndependentHeader**()** **throws** JposException**;**

public boolean getCapItemList**()** **throws** JposException**;**

public boolean getCapJrnEmptySensor**()** **throws** JposException**;**

public boolean getCapJrnNearEndSensor**()** **throws** JposException**;**

public boolean getCapJrnPresent**()** **throws** JposException**;**

public boolean getCapNonFiscalMode**()** **throws** JposException**;**

public boolean getCapOrderAdjustmentFirst**()** **throws** JposException**;**

public boolean getCapPercentAdjustment**()** **throws** JposException**;**

public boolean getCapPositiveAdjustment**()** **throws** JposException**;**

public boolean getCapPowerLossReport**()** **throws** JposException**;**

public int getCapPowerReporting**()** **throws** JposException**;**

public boolean getCapPredefinedPaymentLines**()** **throws** JposException**;**

public boolean getCapReceiptNotPaid**()** **throws** JposException**;**

public boolean getCapRecEmptySensor**()** **throws** JposException**;**

public boolean getCapRecNearEndSensor**()** **throws** JposException**;**

public boolean getCapRecPresent**()** **throws** JposException**;**

public boolean getCapRemainingFiscalMemory**()** **throws** JposException**;**

public boolean getCapReservedWord**()** **throws** JposException**;**

public boolean getCapSetHeader**()** **throws** JposException**;**

public boolean getCapSetPOSID**()** **throws** JposException**;**

public boolean getCapSetStoreFiscalID**()** **throws** JposException**;**

public boolean getCapSetTrailer**()** **throws** JposException**;**

public boolean getCapSetVatTable**()** **throws** JposException**;**

public boolean getCapSlpEmptySensor**()** **throws** JposException**;**

public boolean getCapSlpFiscalDocument**()** **throws** JposException**;**

public boolean getCapSlpFullSlip**()** **throws** JposException**;**

public boolean getCapSlpNearEndSensor**()** **throws** JposException**;**

public boolean getCapSlpPresent**()** **throws** JposException**;**

public boolean getCapSlpValidation**()** **throws** JposException**;**

public boolean getCapSubAmountAdjustment**()** **throws** JposException**;**

public boolean getCapSubPercentAdjustment**()** **throws** JposException**;**

public boolean getCapSubtotal**()** **throws** JposException**;**

public boolean getCapTrainingMode**()** **throws** JposException**;**

public boolean getCapValidateJournal**()** **throws** JposException**;**

public boolean getCapXReport**()** **throws** JposException**;**

// Properties

public int getOutputID**()** **throws** JposException**;**

public int getPowerNotify**()** **throws** JposException**;**

public void setPowerNotify**(**int powerNotify**)** **throws** JposException**;**

public int getPowerState**()** **throws** JposException**;**

public int getAmountDecimalPlace**()** **throws** JposException**;**

public boolean getAsyncMode**()** **throws** JposException**;**

public void setAsyncMode**(**boolean asyncMode**)** **throws** JposException**;**

public boolean getCheckTotal**()** **throws** JposException**;**

public void setCheckTotal**(**boolean checkTotal**)** **throws** JposException**;**

public int getCountryCode**()** **throws** JposException**;**

public boolean getCoverOpen**()** **throws** JposException**;**

public boolean getDayOpened**()** **throws** JposException**;**

public int getDescriptionLength**()** **throws** JposException**;**

public boolean getDuplicateReceipt**()** **throws** JposException**;**

public int getErrorLevel**()** **throws** JposException**;**

public int getErrorOutID**()** **throws** JposException**;**

public int getErrorState**()** **throws** JposException**;**

public int getErrorStation**()** **throws** JposException**;**

public String getErrorString**()** **throws** JposException**;**

public boolean getFlagWhenIdle**()** **throws** JposException**;**

public void setFlagWhenIdle**(**boolean flagWhenIdle**)** **throws** JposException**;**

public boolean getJrnEmpty**()** **throws** JposException**;**

public boolean getJrnNearEnd**()** **throws** JposException**;**

public int getMessageLength**()** **throws** JposException**;**

public int getNumHeaderLines**()** **throws** JposException**;**

public int getNumTrailerLines**()** **throws** JposException**;**

public int getNumVatRates**()** **throws** JposException**;**

public String getPredefinedPaymentLines**()** **throws** JposException**;**

public int getPrinterState**()** **throws** JposException**;**

public int getQuantityDecimalPlaces**()** **throws** JposException**;**

public int getQuantityLength**()** **throws** JposException**;**

public boolean getRecEmpty**()** **throws** JposException**;**

public boolean getRecNearEnd**()** **throws** JposException**;**

public int getRemainingFiscalMemory**()** **throws** JposException**;**

public String getReservedWord**()** **throws** JposException**;**

public boolean getSlpEmpty**()** **throws** JposException**;**

public boolean getSlpNearEnd**()** **throws** JposException**;**

public int getSlipSelection**()** **throws** JposException**;**

public void setSlipSelection**(**int slipSelection**)** **throws** JposException**;**

public boolean getTrainingModeActive**()** **throws** JposException**;**

// Methods

public void beginFiscalDocument**(**int documentAmount**)**

**throws** JposException**;**

public void beginFiscalReceipt**(**boolean printHeader**)**

**throws** JposException**;**

public void beginFixedOutput**(**int station**,** int documentType**)**

**throws** JposException**;**

public void beginInsertion**(**int timeout**)** **throws** JposException**;**

public void beginItemList**(**int vatID**)** **throws** JposException**;**

public void beginNonFiscal**()** **throws** JposException**;**

public void beginRemoval**(**int timeout**)** **throws** JposException**;**

public void beginTraining**()** **throws** JposException**;**

public void clearError**()** **throws** JposException**;**

public void clearOutput**()** **throws** JposException**;**

public void endFiscalDocument**()** **throws** JposException**;**

public void endFiscalReceipt**(**boolean printHeader**)** **throws** JposException**;**

public void endFixedOutput**()** **throws** JposException**;**

public void endInsertion**()** **throws** JposException**;**

public void endItemList**()** **throws** JposException**;**

public void endNonFiscal**()** **throws** JposException**;**

public void endRemoval**()** **throws** JposException**;**

public void endTraining**()** **throws** JposException**;**

public void getData**(**int dataItem**,** int**[]** optArgs**,** String**[]** data**)**

**throws** JposException**;**

public void getDate**(**String**[]** Date**)** **throws** JposException**;**

public void getTotalizer**(**int vatID**,** int optArgs**,** String**[]** data**)**

**throws** JposException**;**

public void getVatEntry**(**int vatID**,** int optArgs**,** int**[]** vatRate**)**

**throws** JposException**;**

public void printDuplicateReceipt**()** **throws** JposException**;**

public void printFiscalDocumentLine**(**String documentLine**)**

**throws** JposException**;**

public void printFixedOutput**(**int documentType**,** int lineNumber**,**

String data**)** **throws** JposException**;**

public void printNormal**(**int station**,** String data**)** **throws** JposException**;**

public void printPeriodicTotalsReport**(**String date1**,** String date2**)**

**throws** JposException**;**

public void printPowerLossReport**()** **throws** JposException**;**

public void printRecItem**(**String description**,** long price**,** int quantity**,**

int vatInfo**,** long unitPrice**,** String unitName**)**

**throws** JposException**;**

public void printRecItemAdjustment**(**int adjustmentType**,**

String description**,** long amount**,** int vatInfo**)**

**throws** JposException**;**

public void printRecMessage**(**String message**)** **throws** JposException**;**

public void printRecNotPaid**(**String description**,** long amount**)**

**throws** JposException**;**

public void printRecRefund**(**String description**,** long amount**,** int vatInfo**)**

**throws** JposException**;**

public void printRecSubtotal**(**long amount**)** **throws** JposException**;**

public void printRecSubtotalAdjustment**(**int adjustmentType**,**

String description**,** long amount**)** **throws** JposException**;**

public void printRecTotal**(**long total**,** long payment**,** String description**)**

**throws** JposException**;**

public void printRecVoid**(**String description**)** **throws** JposException**;**

public void printRecVoidItem**(**String description**,** long amount**,**

int quantity**,** int adjustmentType**,** long adjustment**,**

int vatInfo**)** **throws** JposException**;**

public void printReport**(**int reportType**,** String startNum**,** String endNum**)**

**throws** JposException**;**

public void printXReport**()** **throws** JposException**;**

public void printZReport**()** **throws** JposException**;**

public void resetPrinter**()** **throws** JposException**;**

public void setDate**(**String date**)** **throws** JposException**;**

public void setHeaderLine**(**int lineNumber**,** String text**,**

boolean doubleWidth**)** **throws** JposException**;**

public void setPOSID**(**String POSID**,** String cashierID**)**

**throws** JposException**;**

public void setStoreFiscalID**(**String ID**)** **throws** JposException**;**

public void setTrailerLine**(**int lineNumber**,** String text**,**

boolean doubleWidth**)** **throws** JposException**;**

public void setVatTable**()** **throws** JposException**;**

public void setVatValue**(**int vatID**,** String vatValue**)**

**throws** JposException**;**

public void verifyItem**(**String itemName**,** int vatID**)** **throws** JposException**;**

**}**

public interface FiscalPrinterService16

**extends** FiscalPrinterService15

**{**

// Capabilities

public boolean getCapAdditionalHeader**()** **throws** JposException**;**

public boolean getCapAdditionalTrailer**()** **throws** JposException**;**

public boolean getCapChangeDue**()** **throws** JposException**;**

public boolean getCapEmptyReceiptIsVoidable**()** **throws** JposException**;**

public boolean getCapFiscalReceiptStation**()** **throws** JposException**;**

public boolean getCapFiscalReceiptType**()** **throws** JposException**;**

public boolean getCapMultiContractor**()** **throws** JposException**;**

public boolean getCapOnlyVoidLastItem**()** **throws** JposException**;**

public boolean getCapPackageAdjustment**()** **throws** JposException**;**

public boolean getCapPostPreLine**()** **throws** JposException**;**

public boolean getCapSetCurrency**()** **throws** JposException**;**

public boolean getCapTotalizerType**()** **throws** JposException**;**

// Properties

public int getActualCurrency**()** **throws** JposException**;**

public String getAdditionalHeader**()** **throws** JposException**;**

public void setAdditionalHeader**(**String additionalHeader**)**

**throws** JposException**;**

public String getAdditionalTrailer**()** **throws** JposException**;**

public void setAdditionalTrailer**(**String additionalTrailer**)**

**throws** JposException**;**

public String getChangeDue**()** **throws** JposException**;**

public void setChangeDue**(**String changeDue**)** **throws** JposException**;**

public int getContractorId**()** **throws** JposException**;**

public void setContractorId**(**int contractorId**)** **throws** JposException**;**

public int getDateType**()** **throws** JposException**;**

public void setDateType**(**int dateType**)** **throws** JposException**;**

public int getFiscalReceiptStation**()** **throws** JposException**;**

public void setFiscalReceiptStation**(**int fiscalReceiptStation**)**

**throws** JposException**;**

public int getFiscalReceiptType**()** **throws** JposException**;**

public void setFiscalReceiptType**(**int fiscalReceiptType**)**

**throws** JposException**;**

public int getMessageType**()** **throws** JposException**;**

public void setMessageType**(**int messageType**)** **throws** JposException**;**

public String getPostLine**()** **throws** JposException**;**

public void setPostLine**(**String postLine**)** **throws** JposException**;**

public String getPreLine**()** **throws** JposException**;**

public void setPreLine**(**String preLine**)** **throws** JposException**;**

public int getTotalizerType**()** **throws** JposException**;**

public void setTotalizerType**(**int totalizerType**)** **throws** JposException**;**

// Methods

public void setCurrency**(**int newCurrency**)** **throws** JposException**;**

public void printRecCash**(**long amount**)** **throws** JposException**;**

public void printRecItemFuel**(**String description**,** long price**,**

int quantity**,** int vatInfo**,** long unitPrice**,**

String unitName**,** long specialTax**,** String specialTaxName**)**

**throws** JposException**;**

public void printRecItemFuelVoid**(**String description**,** long price**,**

int vatInfo**,** long specialTax**)** **throws** JposException**;**

public void printRecPackageAdjustment**(**int adjustmentType**,**

String description**,** String vatAdjustment**)**

**throws** JposException**;**

public void printRecPackageAdjustVoid**(**int adjustmentType**,**

String vatAdjustment**)** **throws** JposException**;**

public void printRecRefundVoid**(**String description**,** long amount**,**

int vatInfo**)** **throws** JposException**;**

public void printRecSubtotalAdjustVoid**(**int adjustmentType**,** long amount**)**

**throws** JposException**;**

public void printRecTaxID**(**String taxID**)** **throws** JposException**;**

**}**

public interface FiscalPrinterService17

**extends** FiscalPrinterService16

**{**

public int getAmountDecimalPlaces**()** **throws** JposException**;**

}

public interface FiscalPrinterService18

**extends** FiscalPrinterService17

**{**

// Capabilities

public boolean getCapStatisticsReporting**()** **throws** JposException**;**

public boolean getCapUpdateStatistics**()** **throws** JposException**;**

// Methods

public void resetStatistics**(**String statisticsBuffer**)**

**throws** JposException**;**

public void retrieveStatistics**(**String**[]** statisticsBuffer**)**

**throws** JposException**;**

public void updateStatistics**(**String statisticsBuffer**)**

**throws** JposException**;**

}

public interface FiscalPrinterService19

**extends** FiscalPrinterService18

**{**

// Capabilities

public boolean getCapCompareFirmwareVersion**()** **throws** JposException**;**

public boolean getCapUpdateFirmware**()** **throws** JposException**;**

// Methods

public void compareFirmwareVersion**(**String firmwareFileName**,** int**[]** result**)**

**throws** JposException**;**

public void updateFirmware**(**String firmwareFileName**)**

**throws** JposException**;**

**}**

public interface FiscalPrinterService111 **extends** FiscalPrinterService110

**{**

// Capabilities

public boolean getCapPositiveSubtotalAdjustment**()** **throws** JposException**;**

// Methods

public void printRecItemVoid**(**String description**,**

long price**,**

int quantity**,**

int vatInfo**,**

long unitPrice**,**

String unitName**)**

**throws** JposException**;**

public void printRecItemAdjustmentVoid**(**int adjustmentType**,**

String description**,**

long amount**,**

int vatInfo**)**

**throws** JposException**;**

**}**

public interface FiscalPrinterService112 **extends** FiscalPrinterService111

**{**

// Methods

public void printRecItemRefund**(**String description**,**

long amount**,**

int quantity**,**

int vatInfo**,**

long unitAmount**,**

String unitName**)**

**throws** JposException**;**

public void printRecItemRefundVoid**(**String description**,**

long amount**,**

int quantity**,**

int vatInfo**,**

long unitAmount**,**

String unitName**)**

**throws** JposException**;**

**}**

**Свойства :**

public String getCheckHealthText**()** **throws** JposException**;**

Свойство содержит результат вызова метода checkHealth() в текстовом виде.

public boolean getClaimed**()** **throws** JposException**;**

Свойство имеет значение True после успешного вызова метода Claim, False после вызова Release.

public boolean getDeviceEnabled**()** **throws** JposException**;**

Свойство имеет значение True после успешного вызова метода setDeviceEnabled.

public void setDeviceEnabled**(**boolean deviceEnabled**)** **throws** JposException**;**

Метод выполняет инициализацию драйвера после успешного выполнения open() и claim().

public String getDeviceServiceDescription**()** **throws** JposException**;**

Возвращает "PIRIT 2016"

public int getDeviceServiceVersion**()** **throws** JposException**;**

Возвращает 1013015

public boolean getFreezeEvents**()** **throws** JposException**;**

Не поддерживается данной версией драйвера.

public void setFreezeEvents**(**boolean freezeEvents**)** **throws** JposException**;**

Не поддерживается данной версией драйвера.

public String getPhysicalDeviceDescription**()** **throws** JposException**;**

Возвращает строку вида "Fiscal Printer PIRIT"

public String Fiscal Printer PIRIT**()** **throws** JposException**;**

Возвращает строку вида "Fiscal Printer PIRIT"

public int getState**()** **throws** JposException**;**

Текущее состояние устройства. Допустимые значения:

1 Устройство закрыто (не был вызван метод Open)

2 Устройство находится в рабочем состоянии и не занято выполнением операций.

public boolean getCapAdditionalLines**()** **throws** JposException**;**

Возвращает true, метод PrintRecMessage может печатать произвольный текст после печати итога.

public boolean getCapAmountAdjustment**()** **throws** JposException**;**

Возвращает true, метод PrintRecItemAdjustment поддерживает скидку в денежных единицах.

public boolean getCapAmountNotPaid**()** **throws** JposException**;**

Возвращает false.

public boolean getCapCheckTotal**()** **throws** JposException**;**

Возвращает true, проверка итога в методе PrintRecTotal может производиться.

public boolean getCapCoverSensor**()** **throws** JposException**;**

Возвращает true. Все модели ФР ПИРИТ имеют датчик открытия крышки.

public boolean getCapDoubleWidth**()** **throws** JposException**;**

Возвращает true. Все модели ФР ПИРИТ имеют шрифт двойной ширины.

public boolean getCapDuplicateReceipt**()** **throws** JposException**;**

Возвращает true, все модели ФР ПИРИТ могут распечатать копию последнего чека.

public void setDuplicateReceipt**(**boolean duplicateReceipt**)** **throws** JposException**;**

Устанавливает свойство DuplicateReceipt. Ничего не делает в драйвере, так как копия данных чека есть в ФР.

public boolean getCapFixedOutput**()** **throws** JposException**;**

Возвращает false, драйвер не поддерживает печать методами BeginFixedOutput, PrintFixedOutput и EndFixedOutput.

public boolean getCapHasVatTable**()** **throws** JposException**;**

Возвращает true, все модели ФР поддерживают таблицу налоговых ставок.

public boolean getCapIndependentHeader**()** **throws** JposException**;**

Возвращает true, драйвер автоматически печатает клише следующего чека после предыдущего.

public boolean getCapItemList**()** **throws** JposException**;**

Возвращает false, ФР не печатает список товаров в отчете по налогам.

public boolean getCapJrnEmptySensor**()** **throws** JposException**;**

Возвращает true.

public boolean getCapJrnNearEndSensor**()** **throws** JposException**;**

Возвращает false.

public boolean getCapJrnPresent**()** **throws** JposException**;**

Возвращает false.

public boolean getCapNonFiscalMode**()** **throws** JposException**;**

Возвращает true.

public boolean getCapOrderAdjustmentFirst**()** **throws** JposException**;**

Возвращает false.

public boolean getCapPercentAdjustment**()** **throws** JposException**;**

Возвращает false.

public boolean getCapPositiveAdjustment**()** **throws** JposException**;**

Возвращает true, драйвер позволяет делать надбавки на позицию с помощью метода PrintRecItemAdjustment.

public boolean getCapPowerLossReport**()** **throws** JposException**;**

Возвращает false, фискальные регистраторы ПИРИТ не поддерживают печать отчета об отключении питания.

public int getCapPowerReporting**()** **throws** JposException**;**

Возвращает 1, драйвер может определять два состояния ФР, OFF\_OFFLINE (отключен или выключен) и ONLINE (подключен).

public boolean getCapPredefinedPaymentLines**()** **throws** JposException**;**

Возвращает false.

public boolean getCapReceiptNotPaid**()** **throws** JposException**;**

Возвращает false, метод [PrintRecNotPaid](#bookmark=id.1t3h5sf) нельзя использовать для печати неоплаченной части итога чека.

public boolean getCapRecEmptySensor**()** **throws** JposException**;**

Возвращает true.

public boolean getCapRecNearEndSensor**()** **throws** JposException**;**

Возвращает true.

public boolean getCapRecPresent**()** **throws** JposException**;**

Возвращает true, на всех ФР есть чековая станция.

public boolean getCapRemainingFiscalMemory**()** **throws** JposException**;**

Возвращает false.

public boolean getCapReservedWord**()** **throws** JposException**;**

Возвращает false, в ФР и драйвере нет зарезервированных слов.

public boolean getCapSetHeader**()** **throws** JposException**;**

Возвращает true, драйвер позволяет устанавливать заголовок чека.

public boolean getCapSetPOSID**()** **throws** JposException**;**

Возвращает true, драйвер позволяет устанавливать номер ФР в магазине .

public boolean getCapSetStoreFiscalID**()** **throws** JposException**;**

Возвращает true, драйвер позволяет устанавливать ИНН ФР.

public boolean getCapSetTrailer**()** **throws** JposException**;**

Возвращает true, драйвер позволяет устанавливать подвал чека.

public boolean getCapSetVatTable**()** **throws** JposException**;**

Возвращает true, драйвер позволяет устанавливать налоговые ставки.

public boolean getCapSlpEmptySensor**()** **throws** JposException**;**

Возвращает false.

public boolean getCapSlpFiscalDocument**()** **throws** JposException**;**

Поддеживает ли ФР фискальную печать на подкладном документе.

Возвращает false, так как драйвер не поддерживает фискальную печать на подкладном документе

public boolean getCapSlpFullSlip**()** **throws** JposException**;**

Возвращает false, драйвер не поддерживает полноразмерные подкладные документы.

public boolean getCapSlpNearEndSensor**()** **throws** JposException**;**

Возвращает false, драйвер не поддерживает датчик конца подкладного документа.

public boolean getCapSlpPresent**()** **throws** JposException**;**

Возвращает false, драйвер не поддерживает печать подкладных документов.

public boolean getCapSlpValidation**()** **throws** JposException**;**

Возвращает false, драйвер не поддерживает печать проверочной информации на подкладном документе.

public boolean getCapSubAmountAdjustment**()** **throws** JposException**;**

Возвращает true, фискальный регистратор поддерживает скидки суммой.

public boolean getCapSubPercentAdjustment**()** **throws** JposException**;**

Возвращает true, фискальный регистратор поддерживает процентные скидки.

public boolean getCapSubtotal**()** **throws** JposException**;**

Возвращает true, драйвер поддерживает метод printRecSubtotal.

public boolean getCapTrainingMode**()** **throws** JposException**;**

Возвращает false. Режим тренировки не поддерживается в драйвере.

public boolean getCapValidateJournal**()** **throws** JposException**;**

Возвращает false.

public boolean getCapXReport**()** **throws** JposException**;**

Возвращает true, ФР поддерживает печать X отчета.

public int getOutputID**()** **throws** JposException**;**

Возвращает -1 драйвер поддерживает не последнего события.

public int getPowerNotify**()** **throws** JposException**;**

Возвращает тип уведомления о питании принтера. По умолчанию уведомление отключено, powerNotify= 0.

public void setPowerNotify**(**int powerNotify**)** **throws** JposException**;**

Драйвер не поддерживает уведомления о питании принтера.

public int getPowerState**()** **throws** JposException**;**

Не поддерживается данной версией драйвера.

public int getAmountDecimalPlace**()** **throws** JposException**;**

Возвращает - 2 - количество десятичных разрядов, которые ФР использует для вычислений.

public boolean getAsyncMode**()** **throws** JposException**;**

Возвращает false.

public void setAsyncMode**(**boolean asyncMode**)** **throws** JposException**;**

Не поддерживается данной версией драйвера.

public boolean getCheckTotal**()** **throws** JposException**;**

Если true, то драйвер автоматически сравнивает сумму чека фискального регистратора и сумму чека приложения. Если возвращается false, то автоматическое сравнение итогов не поддерживается и свойство **CheckTotal** доступно только для чтения.

public void setCheckTotal**(**boolean checkTotal**)** **throws** JposException**;**

Установка свойства **CheckTotal.**

public int getCountryCode**()** **throws** JposException**;**

Возвращает значение, определяющее какие страны поддерживает ФР. Возможные значения:

FPTR\_CC\_BRAZIL ФР поддерживае налоговые правила Бразилии.

FPTR\_CC\_GREECE ФР поддерживае налоговые правила Греции.

FPTR\_CC\_HUNGARY ФР поддерживае налоговые правила Венгрии.

FPTR\_CC\_ITALY ФР поддерживае налоговые правила Италии.

FPTR\_CC\_POLAND ФР поддерживае налоговые правила Польши.

FPTR\_CC\_TURKEY ФР поддерживае налоговые правила Турции.

FPTR\_CC\_RUSSIA ФР поддерживае налоговые правила России.

FPTR\_CC\_BULGARIA ФР поддерживае налоговые правила Болгарии.

FPTR\_CC\_ROMANIA ФР поддерживае налоговые правила Румынии.

FPTR\_CC\_CZECH\_REPUBLIC ФР поддерживае налоговые правила Чехии.

FPTR\_CC\_UKRAINE ФР поддерживае налоговые правила Украины.

FPTR\_CC\_SWEDEN ФР поддерживае налоговые правила Швеции.

FPTR\_CC\_OTHER Неизвестная или новая страна. .

Свойство инициализируется после первого включения устройства.

public boolean getCoverOpen**()** **throws** JposException**;**

Если true, значит крышка ФР открыта.

Если **CapCoverSensor** = false, значит у ФР нет датчика крышки и это свойство всегда false.

Свойство обновляется, если

public boolean getDayOpened**()** **throws** JposException**;**

public int getDescriptionLength**()** **throws** JposException**;**

public boolean getDuplicateReceipt**()** **throws** JposException**;**

public int getErrorLevel**()** **throws** JposException**;**

public int getErrorOutID**()** **throws** JposException**;**

public int getErrorState**()** **throws** JposException**;**

public int getErrorStation**()** **throws** JposException**;**

public String getErrorString**()** **throws** JposException**;**

public boolean getFlagWhenIdle**()** **throws** JposException**;**

public void setFlagWhenIdle**(**boolean flagWhenIdle**)** **throws** JposException**;**

public boolean getJrnEmpty**()** **throws** JposException**;**

public boolean getJrnNearEnd**()** **throws** JposException**;**

public int getMessageLength**()** **throws** JposException**;**

public int getNumHeaderLines**()** **throws** JposException**;**

public int getNumTrailerLines**()** **throws** JposException**;**

public int getNumVatRates**()** **throws** JposException**;**

public String getPredefinedPaymentLines**()** **throws** JposException**;**

public int getPrinterState**()** **throws** JposException**;**

public int getQuantityDecimalPlaces**()** **throws** JposException**;**

public int getQuantityLength**()** **throws** JposException**;**

public boolean getRecEmpty**()** **throws** JposException**;**

public boolean getRecNearEnd**()** **throws** JposException**;**

public int getRemainingFiscalMemory**()** **throws** JposException**;**

public String getReservedWord**()** **throws** JposException**;**

public boolean getSlpEmpty**()** **throws** JposException**;**

public boolean getSlpNearEnd**()** **throws** JposException**;**

public int getSlipSelection**()** **throws** JposException**;**

public void setSlipSelection**(**int slipSelection**)** **throws** JposException**;**

public boolean getTrainingModeActive**()** **throws** JposException**;**

public void beginFiscalDocument**(**int documentAmount**)**

**throws** JposException**;**

public void beginFiscalReceipt**(**boolean printHeader**)**

**throws** JposException**;**

public void beginFixedOutput**(**int station**,** int documentType**)**

**throws** JposException**;**

public void beginInsertion**(**int timeout**)** **throws** JposException**;**

public void beginItemList**(**int vatID**)** **throws** JposException**;**

public void beginNonFiscal**()** **throws** JposException**;**

public void beginRemoval**(**int timeout**)** **throws** JposException**;**

public void beginTraining**()** **throws** JposException**;**

public void clearError**()** **throws** JposException**;**

public void clearOutput**()** **throws** JposException**;**

public void endFiscalDocument**()** **throws** JposException**;**

public void endFiscalReceipt**(**boolean printHeader**)** **throws** JposException**;**

public void endFixedOutput**()** **throws** JposException**;**

public void endInsertion**()** **throws** JposException**;**

public void endItemList**()** **throws** JposException**;**

public void endNonFiscal**()** **throws** JposException**;**

public void endRemoval**()** **throws** JposException**;**

public void endTraining**()** **throws** JposException**;**

public void getData**(**int dataItem**,** int**[]** optArgs**,** String**[]** data**)**

**throws** JposException**;**

public void getDate**(**String**[]** Date**)** **throws** JposException**;**

public void getTotalizer**(**int vatID**,** int optArgs**,** String**[]** data**)**

**throws** JposException**;**

public void getVatEntry**(**int vatID**,** int optArgs**,** int**[]** vatRate**)**

**throws** JposException**;**

public void printDuplicateReceipt**()** **throws** JposException**;**

public void printFiscalDocumentLine**(**String documentLine**)**

**throws** JposException**;**

public void printFixedOutput**(**int documentType**,** int lineNumber**,**

String data**)** **throws** JposException**;**

public void printNormal**(**int station**,** String data**)** **throws** JposException**;**

public void printPeriodicTotalsReport**(**String date1**,** String date2**)**

**throws** JposException**;**

public void printPowerLossReport**()** **throws** JposException**;**

public void printRecItem**(**String description**,** long price**,** int quantity**,**

int vatInfo**,** long unitPrice**,** String unitName**)**

**throws** JposException**;**

public void printRecItemAdjustment**(**int adjustmentType**,**

String description**,** long amount**,** int vatInfo**)**

**throws** JposException**;**

public void printRecMessage**(**String message**)** **throws** JposException**;**

public void printRecNotPaid**(**String description**,** long amount**)**

**throws** JposException**;**

public void printRecRefund**(**String description**,** long amount**,** int vatInfo**)**

**throws** JposException**;**

public void printRecSubtotal**(**long amount**)** **throws** JposException**;**

public void printRecSubtotalAdjustment**(**int adjustmentType**,**

String description**,** long amount**)** **throws** JposException**;**

public void printRecTotal**(**long total**,** long payment**,** String description**)**

**throws** JposException**;**

public void printRecVoid**(**String description**)** **throws** JposException**;**

public void printRecVoidItem**(**String description**,** long amount**,**

int quantity**,** int adjustmentType**,** long adjustment**,**

int vatInfo**)** **throws** JposException**;**

public void printReport**(**int reportType**,** String startNum**,** String endNum**)**

**throws** JposException**;**

public void printXReport**()** **throws** JposException**;**

public void printZReport**()** **throws** JposException**;**

public void resetPrinter**()** **throws** JposException**;**

public void setDate**(**String date**)** **throws** JposException**;**

public void setHeaderLine**(**int lineNumber**,** String text**,**

boolean doubleWidth**)** **throws** JposException**;**

public void setPOSID**(**String POSID**,** String cashierID**)**

**throws** JposException**;**

public void setStoreFiscalID**(**String ID**)** **throws** JposException**;**

public void setTrailerLine**(**int lineNumber**,** String text**,**

boolean doubleWidth**)** **throws** JposException**;**

public void setVatTable**()** **throws** JposException**;**

public void setVatValue**(**int vatID**,** String vatValue**)**

**throws** JposException**;**

public void verifyItem**(**String itemName**,** int vatID**)** **throws** JposException**;**

**}**

public interface FiscalPrinterService16

**extends** FiscalPrinterService15

**{**

// Capabilities

public boolean getCapAdditionalHeader**()** **throws** JposException**;**

public boolean getCapAdditionalTrailer**()** **throws** JposException**;**

public boolean getCapChangeDue**()** **throws** JposException**;**

public boolean getCapEmptyReceiptIsVoidable**()** **throws** JposException**;**

public boolean getCapFiscalReceiptStation**()** **throws** JposException**;**

public boolean getCapFiscalReceiptType**()** **throws** JposException**;**

public boolean getCapMultiContractor**()** **throws** JposException**;**

public boolean getCapOnlyVoidLastItem**()** **throws** JposException**;**

public boolean getCapPackageAdjustment**()** **throws** JposException**;**

public boolean getCapPostPreLine**()** **throws** JposException**;**

public boolean getCapSetCurrency**()** **throws** JposException**;**

public boolean getCapTotalizerType**()** **throws** JposException**;**

// Properties

public int getActualCurrency**()** **throws** JposException**;**

public String getAdditionalHeader**()** **throws** JposException**;**

public void setAdditionalHeader**(**String additionalHeader**)**

**throws** JposException**;**

public String getAdditionalTrailer**()** **throws** JposException**;**

public void setAdditionalTrailer**(**String additionalTrailer**)**

**throws** JposException**;**

public String getChangeDue**()** **throws** JposException**;**

public void setChangeDue**(**String changeDue**)** **throws** JposException**;**

public int getContractorId**()** **throws** JposException**;**

public void setContractorId**(**int contractorId**)** **throws** JposException**;**

public int getDateType**()** **throws** JposException**;**

public void setDateType**(**int dateType**)** **throws** JposException**;**

public int getFiscalReceiptStation**()** **throws** JposException**;**

public void setFiscalReceiptStation**(**int fiscalReceiptStation**)**

**throws** JposException**;**

public int getFiscalReceiptType**()** **throws** JposException**;**

public void setFiscalReceiptType**(**int fiscalReceiptType**)**

**throws** JposException**;**

public int getMessageType**()** **throws** JposException**;**

public void setMessageType**(**int messageType**)** **throws** JposException**;**

public String getPostLine**()** **throws** JposException**;**

public void setPostLine**(**String postLine**)** **throws** JposException**;**

public String getPreLine**()** **throws** JposException**;**

public void setPreLine**(**String preLine**)** **throws** JposException**;**

public int getTotalizerType**()** **throws** JposException**;**

public void setTotalizerType**(**int totalizerType**)** **throws** JposException**;**

// Methods

public void setCurrency**(**int newCurrency**)** **throws** JposException**;**

public void printRecCash**(**long amount**)** **throws** JposException**;**

public void printRecItemFuel**(**String description**,** long price**,**

int quantity**,** int vatInfo**,** long unitPrice**,**

String unitName**,** long specialTax**,** String specialTaxName**)**

**throws** JposException**;**

public void printRecItemFuelVoid**(**String description**,** long price**,**

int vatInfo**,** long specialTax**)** **throws** JposException**;**

public void printRecPackageAdjustment**(**int adjustmentType**,**

String description**,** String vatAdjustment**)**

**throws** JposException**;**

public void printRecPackageAdjustVoid**(**int adjustmentType**,**

String vatAdjustment**)** **throws** JposException**;**

public void printRecRefundVoid**(**String description**,** long amount**,**

int vatInfo**)** **throws** JposException**;**

public void printRecSubtotalAdjustVoid**(**int adjustmentType**,** long amount**)**

**throws** JposException**;**

public void printRecTaxID**(**String taxID**)** **throws** JposException**;**

**}**

public interface FiscalPrinterService17

**extends** FiscalPrinterService16

**{**

public int getAmountDecimalPlaces**()** **throws** JposException**;**

}

public interface FiscalPrinterService18

**extends** FiscalPrinterService17

**{**

// Capabilities

public boolean getCapStatisticsReporting**()** **throws** JposException**;**

public boolean getCapUpdateStatistics**()** **throws** JposException**;**

// Methods

public void resetStatistics**(**String statisticsBuffer**)**

**throws** JposException**;**

public void retrieveStatistics**(**String**[]** statisticsBuffer**)**

**throws** JposException**;**

public void updateStatistics**(**String statisticsBuffer**)**

**throws** JposException**;**

}

public interface FiscalPrinterService19

**extends** FiscalPrinterService18

**{**

// Capabilities

public boolean getCapCompareFirmwareVersion**()** **throws** JposException**;**

public boolean getCapUpdateFirmware**()** **throws** JposException**;**

// Methods

public void compareFirmwareVersion**(**String firmwareFileName**,** int**[]** result**)**

**throws** JposException**;**

public void updateFirmware**(**String firmwareFileName**)**

**throws** JposException**;**

**}**

public interface FiscalPrinterService111 **extends** FiscalPrinterService110

**{**

// Capabilities

public boolean getCapPositiveSubtotalAdjustment**()** **throws** JposException**;**

// Methods

public void printRecItemVoid**(**String description**,**

long price**,**

int quantity**,**

int vatInfo**,**

long unitPrice**,**

String unitName**)**

**throws** JposException**;**

public void printRecItemAdjustmentVoid**(**int adjustmentType**,**

String description**,**

long amount**,**

int vatInfo**)**

**throws** JposException**;**

**}**

public interface FiscalPrinterService112 **extends** FiscalPrinterService111

**{**

// Methods

public void printRecItemRefund**(**String description**,**

long amount**,**

int quantity**,**

int vatInfo**,**

long unitAmount**,**

String unitName**)**

**throws** JposException**;**

public void printRecItemRefundVoid**(**String description**,**

long amount**,**

int quantity**,**

int vatInfo**,**

long unitAmount**,**

String unitName**)**

**throws** JposException**;**

**}**

**Параметры драйвера**

Параметры драйвера хранятся в файле jpos.xml. Загрузкой параметров занимается библиотека jpos. Драйвер получает JposEntry.

1. Имя порта к которому подключается драйвер.

<!--Port name-->

<prop name=**"portName"** type=**"String"** value=**"COM3"**/>

1. Скорость связи

<!-- Serial port baudRate -->

<prop name=**"baudRate"** type=**"String"** value=**"57600"**/>

1. Число битов данных в байте

<!-- Serial port dataBits -->

<prop name=**"dataBits"** type=**"String"** value=**"8"**/>

1. Число стоповых битов в байте.

<!-- Serial port stopBits -->

<prop name=**"reportType"** type=**"String"** value=**"1"**/>

1. Значение serviceDescription

<!-- serviceDescription -->

<prop name=**"serviceDescription"** type=**"String"** value=**"PIRIT, 2017"**/>

1. Значение productDescription

<!-- productDescription -->

<prop name=**"productDescription"** type=**"String"** value=**"Fiscal Printer PIRIT"**/>

1. Значение productName

<!-- productName -->

<prop name=**"productName"** type=**"String"** value=**"Fiscal Printer PIRIT"**/>

1. Значение statusTimer - таймер опроса статуса ФР.

<!-- statusTimer -->

<prop name="**statusTimer**" type="String" value="2000"/>

**Использование метода DirectIO в JPOS драйвере ФР «ПИРИТ»**

**Команды DirectIO :**

void directIO(int arg0, int[] arg1, Object arg2)

**Список команд** arg0**:**

1 – установка номера чека

2 – установка режима печати штрихкода

3 – установка атрибутов текста

4 – установка позиции штрихкода при печати

5 – установка ширины штрихкода

6 – установка высоты штрихкода

7 – установка типа штрихкода

8 – открыть денежный ящик

9 – получить статус денежного ящика

10 – установка названия ставки налога

11 – получить название ставки налога

12 - Включить/выключить счетчик товаров

13 - Включить/выключить режим пакетной печати

17 - Установить атрибут текста при печати свободной строкой

18 - Команда отрезки бумаги

19 - Команда печати разделительной линии

20 - Команда печати QR кода с текстом

52 – печать реквизита для ОФД

26 - Получить данные о подписанных документах в ФН

27 – получить данные регистров CASH\_REG

28 – получить дату окончания ресурса ФН

29 – получить регистрационный номер ФН

30 – получить дату и время регистрации

31 – получение кол-ва неотправленных документов в ОФД

32 – получение Даты/времени первого док-та для

передачи в ОФД

33 – установка адреса покупателя

34 – получения состояния печатающего устройства

35 - Запрос РНМ (Регистрационного номера ККТ)

36 - Запрос номера последнего ФД

37 - Запрос даты и времени последнего ФД в виде (DDMMYYYYhhmm)

38 - Получение квитанций от заданного номера до последнего включительно

39 - Запрос номера смены

40 - Запрос кол-во чеков за смену

41 - Запрос номера документа последней перерегистрации

42 - Получение квитанций из ФН по указанному диапазону номеров ФД.

55 - Печать копии Z-отчета

56 - Запрос информации с ФР

57 - Запрос данных последнего записанного ФД в ФН

58 - Запрос информации с отчета об открытии смены

59 - Запрос информации с отчета об закрытии смены

61 - Установка флага ожидания статуса связи при окончании бумаги  
74 - Передача КМ в ФН для проверки достоверности в ФН и ОИСМ

75 - Передача КМ для включения в кассовый чек

76 - Количество маркированного товара

**1**

arg0**: 1**

Установка номера чека.

В метод передается параметр int[] arg1. Параметр Object arg2 не используется.

Номер чека установится по значению arg1[0].

**2**

arg0**: 2**

Установка режима печати штрихкода.

В метод передается параметр int[] arg1. Параметр Object arg2 не используется.

Режим печати установится по значению arg1[0]. Если arg1[0] !=0 включен режим

печати штрихкода. Иначе включен режим печати текста.

**3**

arg0**: 3**

Установка атрибутов текста.

В метод передается параметр int[] arg1. Параметр Object arg2 не используется.

Атрибут текста установится по значению arg1[0].

**4**

arg0**: 4**

Установка позиции штрихкода при печати.

В метод передается параметр int[] arg1. Параметр Object arg2 не используется.

Позиция штрихкода установится по значению arg1[0].

**5**

arg0**: 5**

Установка ширины штрихкода.

В метод передается параметр int[] arg1. Параметр Object arg2 не используется.

Ширина штрихкода установится по значению arg1[0].

**6**

arg0**: 6**

Установка высоты штрихкода.

В метод передается параметр int[] arg1. Параметр Object arg2 не используется.

Высота штрихкода установится по значению arg1[0].

**7**

arg0**: 7**

Установка типа штрихкода

В метод передается параметр int[] arg1. Параметр Object arg2 не используется.

Тип штрихкода установится по значению arg1[0].

**8**

arg0**: 8**

Открыть денежный ящик

Параметр int[] arg1 и параметр Object arg2 не используется.

**9**

arg0**: 9**

Получить статус денежного ящика

В метод передается параметр Object arg2. Параметр int[] arg1 не используется.

Object arg2 присвоит результат выполнения команды.

**10**

arg0**: 10**

Установка названия ставки налога.

В метод передается параметр int[] arg1 и параметр Object arg2.

Id ставки налога устанавливается по значению arg1[0].

Название ставки налога передается параметром Object arg2. (String)

**11**

arg0**: 11**

Получить название ставки налога

В метод передается параметр int[] arg1 и параметр Object arg2.

Id запрашиваемой ставки налога устанавливается по значению arg1[0].

Параметр Object arg2 присвоит название ставки.

**12**

arg0**: 12**

Включить/выключить счетчик товаров.

В метод передается параметр int[] arg1. Параметр Object arg2 не используется.

Счетчик включен при значении arg1[0]!=0.

**13**

arg0**: 13**

Включить/выключить режим пакетной печати.

В метод передается параметр int[] arg1. Параметр Object arg2 не используется.

Режим включен при значении arg1[0]!=0.

**14**

arg0**: 14**

Включить/выключить режим печати на чековой ленте.

В метод передается параметр int[] arg1. Параметр Object arg2 не используется.

Печать включена при значении arg1[0]!=0.

**15**

arg0**: 15**

Включить/выключить отрезка чека.

В метод передается параметр int[] arg1. Параметр Object arg2 не используется.

Отрезка включена при значении arg1[0]!=0.

**16**

arg0**: 16**

Включить/выключить блока расчета.

В метод передается параметр int[] arg1. Параметр Object arg2 не используется.

Блока расчета не печатается при значении arg1[0]!=0.

**17**

arg0**: 17**

Установить атрибут текста при печати свободной строкой.

В метод передается параметр int[] arg1. Параметр Object arg2 не используется.

Атрибут устанавливается по значению arg1[0].

**18**

arg0**: 18**

Команда отрезки бумаги.

Параметры не передаются.

**19**

arg0**: 19**

Команда печати разделительной линии.

В метод передается параметр int[] arg1. Параметр Object arg2 не используется.

Позиция начала линии задается arg1[0].

Позиция конца линии задается arg1[1].

Толщина линии задается arg1[1].

**20**

arg0**: 20**

Команда печати QR кода с текстом.

В метод передается параметр int[] arg1. и Object arg2.

Атрибут текста задается arg1[0].

Ширина QR-кода задается arg1[1].

Начало текста задается arg1[2].

Начало QR-кода задается arg1[3].

Строка1 задается arg2[0].

Строка2 задается arg2[1].

Содержимое QR-кода arg2[2].

**21**

arg0**: 21**

Команда установки атрибута признак способа расчета (Тег 1214).

Атрибут задается arg1[0].

**22**

arg0**: 22**

Команда установки атрибута признак предмета расчета (Тег 1212).

Атрибут задается arg1[0].

**24**

arg0**: 24**

Команда открыть смену.

arg1[0] (Число) номер часовой зоны (1..11) (тег 1011)

| **Временная зона** | **Описание** | **Значение тега 1011** |
| --- | --- | --- |
| MSK−1 (UTC+2) | Калининградская область | 1 |
| MSK (UTC+3) | Основная часть европейской России, включая Москву, СПб и др. | 2 |
| MSK+1 (UTC+4) | Самарская, Ульяновская, Астраханская области и др. | 3 |
| MSK+2 (UTC+5) | Екатеринбургское время — Пермь, Тюмень, Челябинск и др. | 4 |
| MSK+3 (UTC+6) | Омская область | 5 |
| MSK+4 (UTC+7) | Новосибирск, Красноярск, Тыва и др. | 6 |
| MSK+5 (UTC+8) | Иркутская область, Бурятия | 7 |
| MSK+6 (UTC+9) | Амурская область, запад Якутии | 8 |
| MSK+7 (UTC+10) | Приморский и Хабаровский края, центр Якутии | 9 |
| MSK+8 (UTC+11) | Магаданская область, восточная Якутия, север Сахалина | 10 |
| MSK+9 (UTC+12) | Камчатка, Чукотка, Курильские острова | 11 |

**52**

arg0**: 52**

Печать реквизита для ОФД.

В метод передается параметр int[] arg1 и параметр Object arg2.

Код реквизита устанавливается по значению arg1[0].

Значение реквизита передается параметром Object arg2.

Значение атрибутов текста устанавливается по умолчанию «0 - Шрифт 13х24»

**53**

arg0**: 53**

Запрос квитанции из ФН по номеру.

В метод передается параметр int[] arg1 и параметр Object arg2[].

Документ запрашивается по номеру переданному в значении arg1[0].

Значение квитанции передается параметром Object arg2.

Например: в качестве приемника результата передается String[], результат присвоит параметр String[0]).

**54**

arg0**: 54**

Установить итог чека.

В метод передается параметр int[] arg1, параметр Object arg2[] не используется.

Итог устанавливается по значению переданному в значении arg1[0].

**55**

arg0**: 55**

Печать копии z-отчета.

Печать копии z-отчета. Параметр int[] arg1 и параметр Object arg2[] не используются.

**56**

arg0**: 56**

Запрос информации с ФР.

В метод передается параметр String arg2[], параметр int[] arg1 не используется.

Результат запроса будет записан как массив в String arg2[] в следующем виде:

· РН ККТ

· ЗН ККТ

· Модель ФР

· Наименование ОФД

· Адрес сервера ОФД

· № порта

· Таймаут ожидание ответа от ОФД

**57**

arg0**: 57**

Запрос данных последнего записанного ФД в ФН.

В метод передается параметр String arg2[], параметр int[] arg1 не используется.

Результат запроса будет записан как массив в String arg2[] в следующем виде:

· Тип документа

· Флаг получения квитанции ОФД

· Дата и время

· ФД №

· ФПД №

· Тип операции

· Сумма

**58**

arg0**: 58**

Запрос информации с отчета об открытии смены.

В метод передается параметр String arg2[], параметр int[] arg1 не используется.

Результат запроса будет записан как массив в String arg2[] в следующем виде:

· Номер смены

· Тип документа (отчет об открытии)

· Дата и время

· РН ККТ

· ЗН ККТ

· ФН №

· ФД №

· ФПД №

· Наименование ОФД

**59**

arg0**: 59**

Запрос информации с отчета об закрытии смены.

В метод передается параметр String arg2[], параметр int[] arg1 не используется.

Результат запроса будет записан как массив в String arg2[] в следующем виде:

· Номер смены

· Тип документа (отчет об открытии)

· Дата и время

· РН ККТ

· ЗН ККТ

· ФН №

· ФД №

· ФПД №

· Наименование ОФД

**25**

arg0**: 25**

Проверить статус документа на “Документ закрыт в ФН”

В метод передается параметр Object arg2. Параметр int[] arg1 не используется.

Object arg2 присвоит результат выполнения команды.

**26**

arg0**: 26**

Получить данные о подписанных документах в ФН.

В метод передается параметр int[] arg1 и параметр Object arg2.

Массив с номерами запрашиваемых документов передается в значении arg1[].

Результат будет передан параметру Object arg2 в виде массива в котором: под индексом 0 будет коллекция типа Vector<PiritFNTicket>, где в классe PiritFNTicket метод getdocNumber() вернет HEX массив с квитанцией ОФД.

Если документ не был подписан на ОФД, в коллекции Vector<PiritFNTicket> его не будет.

Формат HEX квитанции:

| **Поле** | **Тип** | **Длина** |
| --- | --- | --- |
| Дата и время | DATE\_TIME | 5 |
| Фискальный признак ОФД | DATA | 18 |
| Номер ФД | Uint32, LE | 4 |

**27**

arg0**: 27**

Получить данные регистров CASH\_REG.

В метод передается параметр int[] arg1 и параметр Object arg2.

Номер регистра устанавливается по значению arg1[0]:

112: UNNULL\_TOTAL\_COMING\_START\_SHIFT

113: UNNULL\_TOTAL\_REFOUND\_START\_SHIFT

114: UNNULL\_TOTAL\_EXPEND\_START\_SHIFT

115: UNNULL\_TOTAL\_EXPEND\_REFOUND\_START\_SHIFT

212: UNNULL\_TOTAL\_COMING\_START\_REG\_FR

213: UNNULL\_TOTAL\_REFOUND\_START\_REG\_FR

214: UNNULL\_TOTAL\_EXPEND\_START\_REG\_FR

215: UNNULL\_TOTAL\_\_EXPEND\_REFOUND\_START\_REG\_FR

121: DAY\_TOTAL\_SALE\_ALL

123: DAILY\_TOTAL\_REFUND\_ALL

185: DAILY\_TOTAL\_DISCOUNT\_SALE\_PAYMENT1

187: DAILY\_TOTAL\_DISCOUNT\_REFOUND\_PAYMENT3

193: DAILY\_TOTAL\_SALE\_PAYMENT0

197: DAILY\_TOTAL\_SALE\_PAYMENT1

198: DAILY\_TOTAL\_SALE\_PAYMENT2

202: DAILY\_TOTAL\_SALE\_PAYMENT3

206: DAILY\_TOTAL\_SALE\_PAYMENT4

232: DAILY\_TOTAL\_SALE\_PAYMENT5

233: DAILY\_TOTAL\_SALE\_PAYMENT6

234: DAILY\_TOTAL\_SALE\_PAYMENT7

235: DAILY\_TOTAL\_SALE\_PAYMENT8

236: DAILY\_TOTAL\_SALE\_PAYMENT9

237: DAILY\_TOTAL\_SALE\_PAYMENT10

238: DAILY\_TOTAL\_SALE\_PAYMENT11

239: DAILY\_TOTAL\_SALE\_PAYMENT12

240: DAILY\_TOTAL\_SALE\_PAYMENT13

201: DAILY\_TOTAL\_SALE\_PAYMENT14

205: DAILY\_TOTAL\_SALE\_PAYMENT15

195: DAILY\_TOTAL\_REFUND\_PAYMENT0

199: DAILY\_TOTAL\_REFUND\_PAYMENT1

203: DAILY\_TOTAL\_REFUND\_PAYMENT2

220: DAILY\_TOTAL\_REFUND\_PAYMENT3

221: DAILY\_TOTAL\_REFUND\_PAYMENT4

222: DAILY\_TOTAL\_REFUND\_PAYMENT5

223: DAILY\_TOTAL\_REFUND\_PAYMENT6

224: DAILY\_TOTAL\_REFUND\_PAYMENT7

225: DAILY\_TOTAL\_REFUND\_PAYMENT8

226: DAILY\_TOTAL\_REFUND\_PAYMENT9

227: DAILY\_TOTAL\_REFUND\_PAYMENT10

228: DAILY\_TOTAL\_REFUND\_PAYMENT11

229: DAILY\_TOTAL\_REFUND\_PAYMENT12

230: DAILY\_TOTAL\_REFUND\_PAYMENT13

231: DAILY\_TOTAL\_REFUND\_PAYMENT14

207: DAILY\_TOTAL\_REFUND\_PAYMENT15

241: TOTAL\_CASH

242: DAILY\_TOTAL\_CASH\_IN

243: DAILY\_TOTAL\_CASH\_OUT

Значение регистра присвоится параметру Object arg2. (Например: в качестве

приемника результата передается String[], результат присвоит параметр String[0]).

**28**

arg0**: 28**

Получить дату окончания ресурса ФН.

Параметр int[] arg1 не используется.

Результат запроса присвоится параметру Object arg2. (Например: в качестве

приемника результата передается String[], результат присвоит параметр String[0]).

**29**

arg0**: 29**

Получить регистрационный номер ФН.

Параметр int[] arg1 не используется.

Результат запроса присвоится параметру Object arg2. (Например: в качестве

приемника результата передается String[], результат присвоит параметр String[0]).

**30**

arg0**: 30**

Получить дату и время регистрации.

Параметр int[] arg1 не используется.

Результат запроса присвоится параметру Object arg2. (Например: в качестве

приемника результата передается String[], результат присвоит параметр String[0]).

**31**

arg0**: 31**

Получение кол-ва неотправленных документов в ОФД.

Параметр int[] arg1 не используется.

Результат запроса присвоится параметру Object arg2. (Например: в качестве

приемника результата передается String[], результат присвоит параметр String[0]).

**32**

arg0**: 32**

Получение Даты/времени первого док-та для передачи в ОФД.

Параметр int[] arg1 не используется.

Результат запроса присвоится параметру Object arg2. (Например: в качестве

приемника результата передается String[], результат присвоит параметр String[0]).

**33**

arg0**: 33**

Установка адреса покупателя.

Параметр int[] arg1 не используется.

Адрес покупателя передается параметром Object arg2.

**34**

arg0**: 34**

Получение состояния печатающего устройства

Id запрашиваемого статуса задается параметром int[] arg1.

Результат запроса присвоится параметру Object arg2. (Например: в качестве

приемника результата передается String[], результат присвоит параметр String[0]).

**35**

arg0**: 35**

Запрос РНМ (Регистрационного номера ККТ).

Параметр int[] arg1 не используется.

Результат передается параметром Object arg2.

**36**

arg0**: 36**

Запрос номера последнего ФД.

Параметр int[] arg1 не используется.

Результат передается параметром Object arg2.

**37**

arg0**: 37**

Запрос даты и времени последнего ФД в виде (DDMMYYYYhhmm)

Параметр int[] arg1 не используется.

Результат передается параметром Object arg2.

**38**

arg0**: 38**

Получение квитанций от заданного номера до последнего включительно.

В метод передается параметр int[] arg1 и параметр Object arg2.

Параметр arg1[0] содрежит номер от которого начнется выборка документов.

Результат будет передан параметру Object arg2 в виде массива в котором под индексом 0 будет коллекция типа Vector<PiritFNTicket>

Где в классe PiritFNTicket метод getdocNumber() вернет HEX массив с квитанцией ОФД. getDocNumber() вернет int номер документа.

isSigned() вернет boolean статус - подписан ли документ в ОФД.

**39**

arg0**: 39**

Запрос номера смены.

Параметр int[] arg1 не используется.

Результат передается параметром Object arg2.

**40**

arg0**: 40**

Запрос кол-во чеков за смену

Параметр Object arg2 не используется.

Результат передается параметром int[] arg1[0].

**41**

arg0**: 41**

Запрос номера документа последней перерегистрации.

Параметр Object arg2 не используется.

Результат передается параметром int[] arg1[0].

**42**

arg0**: 42**

Получение квитанций из ФН по указанному диапазону номеров ФД.

В метод передается параметр int[] arg1 и параметр Object arg2.

Параметр arg1[0] содержит номер первого документа, arg1[1] содержит номер последнего документа диапазона в рамках которого будет произведена выборка документов.

Результат будет передан параметру Object arg2 в виде массива в котором под индексом 0 будет коллекция типа Vector<PiritFNTicket>

Где в классe PiritFNTicket метод getdocNumber() вернет HEX массив с квитанцией ОФД. getDocNumber() вернет int номер документа.

isSigned() вернет boolean статус - подписан ли документ в ОФД.

**61**

arg0**: 61**

Команда установки флага ожидания.

Атрибут задается arg1[0].

**62**

arg0**: 62**

Атрибут задается arg1[0].

int arg1[0] = 1; - печатать блок

arg0**: 73**

Атрибут задается arg2[0] в виде массива строковых параметров.

* (Строка[1..32]) Код товарной номенклатуры
* (Строка[1..64]) Дополнительный реквизит предмета расчёта
* (Строка) Зарезервировано
* (Целое число) Признак агента по предмету расчёта
* (Строка)[0..12] ИНН поставщика
* (Строка)[0..19] Телефон поставщика
* (Строка)[0..256] Наименование поставщика
* (Строка)[0..256] Адрес оператора перевода (для банк.пл.агента/банк.пл.субагента, иначе пустой)
* (Строка)[0..12] ИНН оператора перевода (для банк.пл.агента/банк.пл.субагента, иначе пустой)
* (Строка)[0..64] Наименование оператора перевода (для банк.пл.агента/банк.пл.субагента, иначе пустой)
* (Строка)[0..19] Телефон оператора перевода (для банк.пл.агента/банк.пл.субагента, иначе пустой)
* (Строка)[0..24] Операция платежного агента (для банк.пл.агента/банк.пл.субагента, иначе пустой)
* (Строка)[0..19] Телефон платежного агента (для пл.агента/пл.субагента, иначе пустой)
* (Строка)[0..19] Телефон оператора по приему платежей (для пл.агента/пл.субагента, иначе пустой
* (Число)[0..3] Идентификатор ФОИВ (тег 1262). Значение определяется ФНС РФ. Параметр используется только при регистрации ККТ в режиме ФФД 1.2.
* (Дата) Дата документа основания (тег 1263). Допускается дата после 1999 года. Должен содержать сведения об НПА отраслевого регулирования. Параметр используется только при регистрации ККТ в режиме ФФД 1.2.  
  Формат даты: Строка длиной 6 символов вида «ДДММГГ»
* (Строка)[0..32] Номер документа основания (тег 1264). Должен содержать сведения об НПА отраслевого регулирования. Параметр используется только при регистрации ККТ в режиме ФФД 1.2.
* (Строка)[0..256] Значение отраслевого реквизита (тег 1265). Значение определяется отраслевым НПА. Параметр используется только при регистрации ККТ в режиме ФФД 1.2.

int arg1[0] не используется

piritSvc112.directIO(73,null,arg2);

**74**

arg0**: 74**

Передача КМ в ФН для проверки достоверности в ФН и ОИСМ.

Код маркировки задается в arg2[0] в виде строки.

Атрибуты проверки КМ задаются в виде int массива в arg1:

* (Целое число) Планируемый статус товара (тег 2003)
* (Целое число) Мера количества (тег 2108)
* (Целое число)[0..1] Режим работы - Если не передавать, режим работы = 1: “все равно проверять КМ в ИСМ, даже если ФН проверил код с отрицательным результатом”

Результат проверки записывается в элементы массива параметра arg1, для получения полного результата размер arg1 должен быть 5 или более.

Значение arg1 после проверки КМ:

* (Целое число) Результат проверки КМ в ФН (тег 2106)
* (Целое число) Причина того, что КМ не проверен в ФН
* (Целое число) Результаты обработки запроса (тег 2005)
* (Целое число) Код обработки запроса (тег 2105)
* (Целое число) Сведения о статусе товара (тег 2109)

Значения тегов в ответе расшифровываются согласно таблицам документации ФФД.

Причина того, что КМ не проверен в ФН, расшифровывается как:

0 КМ проверен в ФН

1 КМ данного типа не подлежит проверке в ФН

2 ФН не содержит ключ проверки кода проверки этого КМ

3 Проверка невозможна, так как отсутствуют теги 91 и/или 92

4 Внутренняя ошибка в ФН при проверке этого КМ

**75**

arg0**: 75**

Передача КМ для включения в кассовый чек.

Код маркировки задается в arg2[0] в виде строки.

Атрибуты добавляемого КМ задаются в виде int массива в arg1:

* (Целое число) Присвоенный статус товара (тег 2110)
* (Целое число) Мера количества (тег 2108)
* (Целое число) Режим работы, значение(по умолчанию 1):

0 Исключение этого КМ из документа

1 Принятие КМ для включения в документ

**Значения реквизита "присвоенный статус товара" (тег 2110)**

| **Значение реквизита** | **Присвоенный статус товара** |
| --- | --- |
| 1 | Штучный товар, подлежащий обязательной маркировке средством идентификации, реализован |
| 2 | Мерный товар, подлежащий обязательной маркировке средством идентификации, в стадии реализации |
| 3 | Штучный товар, подлежащий обязательной маркировке средством идентификации, возвращен |
| 4 | Часть товара, подлежащего обязательной маркировке средством идентификации, возвращена |
| 5 | Штучный товар, подлежащий обязательной маркировке средством идентификации, в стадии реализации |
| 6 | Мерный товар, подлежащий обязательной маркировке средством идентификации, реализован |
| 255 | Статус товара, подлежащего обязательной маркировке средством идентификации, не изменился |

**76**

arg0**: 76**

Количество маркированного товара. Тег ОФД 1023.

Атрибут задается arg2[0] в виде строки.

Применяется при выполнении команд DirectIO 74 и printRecItem. После выполнения printRecItem сохраненное значение очищается.

**79**

arg0**: 79**

Дополнительный параметр товарной позиции, после выполнения printRecItem сохраненное значение очищается.

Параметр int[] arg1 не используется.

Параметры задаются в виде String[] массива в arg2:

* Мера количества предмета расчета (тег 2108)

При работе в режиме ФФД 1.05 в поле указывается строка с названием единицы, которая передается, как тег 1197.

При работе в режиме ФФД 1.2 в поле указывается число с мерой количества предмета расчета, которая передается, как тег 2108.  
Если параметр не передан, применяется значение в зависимости от quantity в printRecItem, для кратных 0 - шт, иначе 11 - кг.

**80**

arg0**: 80**

Получить статус обработки КМ по данным в ФН

Результат запроса статуса присвоится параметру arg1 в виде массива:

* (Целое число) Состояние по проверке КМ
* (Целое число) Состояние по формированию уведомления
* (Целое число) Флаги разрешения команд работы с КМ
* (Целое число) Количество сохранённых результатов проверки КМ
* (Целое число) Количество КМ, включенных в уведомление о реализации
* (Целое число) Предупреждение о заполнении области хранения уведомлений о реализации маркированного товара

**81**

arg0**: 81**

Удаление сохраненных КМ в ФН.

Команда предназначена для удаления сохраненных КМ в ФН, подлежащих включению в “Уведомление о реализации маркированного товара”. При закрытии фискального документа (кассового чека), ФН удалит сохраненные КМ автоматически.

Не использовать на открытом фискальном документе.

**82**

arg0**: 82**

Дополнительный параметр документа продажи, после выполнения endFiscalReceipt сохраненное значение очищается.

Параметр int[] arg1 не используется.

Параметры задаются в виде String[] массива в arg2:

* (Число)[0..3] Гражданство (тег 1244). Параметр передается в случаях, установленных законодательством РФ и только при регистрации ККТ в режиме ФФД 1.2.
* (Число)[0..2] Код вида документа, удостоверяющего личность (тег 1245). Параметр передается в случаях, установленных законодательством РФ и только при регистрации ККТ в режиме ФФД 1.2.
* (Строка)[0..64] Данные документа, удостоверяющего личность (Тег 1246). Параметр передается в случаях, установленных законодательством РФ и только при регистрации ККТ в режиме ФФД 1.2.
* (Строка)[0..256] Адрес покупателя (клиента), географический адрес, не email (Тег 1254). Параметр передается в случаях, установленных законодательством РФ и только при регистрации ККТ в режиме ФФД 1.2.

**83**

arg0**: 83**

Установка таймаута проверки КМ в ОИСМ. При таймауте = 0 проверка КМ в ОИСМ не выполняется.

Атрибут задается arg1[0] в секундах.

**84**

arg0**: 84**

Принятие/Исключение КМ для последующего включения в кассовый чек.

Параметр String[] arg2 не используется.

Тип операции задается в виде int массива в arg1:

* (Целое число) Режим работы, значение(по умолчанию 1):

0 Исключение этого КМ из документа

1 Принятие КМ для включения в документ  
Результат проверки при принятии КМ записывается в первый элемент массива параметра arg1, для получения результата размер arg1 должен быть 1 или более.

Значение arg1 после проверки КМ:

* (Целое число) Результат проверки КМ в ФН (тег 2106)

Результат проверки (тег 2106) необходимо сохранить для передачи в параметр directIO(85) при добавлении соответствующей марки в чек.

**85**

arg0**: 85**

Передача КМ для включения в кассовый чек.

Код маркировки задается в arg2[0] в виде строки.

Атрибуты добавляемого КМ задаются в виде int массива в arg1:

* (Целое число) Присвоенный статус товара (тег 2110)
* (Целое число) Мера количества (тег 2108)
* (Целое число) Результат проверки КМ в ФН (тег 2106) из результата проверки соответствующей КМ(arg2[0]) в directIO(74) или directIO(84)

**86**

arg0**: 86**

Включить/выключить флаг признака расчета в "Интернет" (тег 1125)

По умолчанию признак выключен. Для его использования должны быть заполнены поля "Место расчётов" (Тег 1187) и "Адрес покупателя"(Тег 1008) через методы

DIO 64 и 33.

После команды закрытия документа, переданные в DIO 33, 64, 86 данные отчищаются.

Флаг передается в виде int в arg1[0]:

* (Целое число) 0 - выключить
* (Целое число) 1 - включить

**87**

arg0**: 87**

Указание дополнительных данных об оплате безналичными (тег 1234) (0x47)

Данные тегов 1236, 1237, 1238 передаются в массиве arg2:

* (Целое число(Строкой)) Признак способа оплаты безналичными (тег 1236)
* (Строка[0..256]) Идентификатор безналичной оплаты (тег 1237)
* (Строка[0..256]) Дополнительные сведения о безналичной оплате (тег 1238)

**Реализован интерфейс EventCallBacks:**

При открытии крышки ФР генерируется событие FPTR\_SUE\_COVER\_OPEN

При закрытии крышки ФР FPTR\_SUE\_COVER\_OK

При окончании бумаги ФР генерируется событие FPTR\_SUE\_REC\_EMPTY

При поступлении бумаги ФР генерируется событие FPTR\_SUE\_REC\_PAPEROK

При потери связи с ФР генерируется событие JPOS\_SUE\_POWER\_OFFLINE

При установки связи с ФР генерируется событие JPOS\_PS\_ONLINE

Пример мониторинга статуса ФР через EventCallBacks:

PiritSvc112 piritSvc112 = new PiritSvc112();

EventCallbacks callBacks = piritSvc112.createPiritCallBacks();

StatusUpdateListener listener = new StatusUpdateListener(){

@Override

public void statusUpdateOccurred(StatusUpdateEvent statusUpdateEvent) {

if(statusUpdateEvent.getStatus()== JposConst.JPOS\_PS\_ONLINE) {

System.out.println("FR online");

}

}

piritSvc112.addStatusUpdateListener(listener);

piritSvc112.open("PIRIT-2F",callBacks);

**Руководство по использованию :**

**Пример инициализации драйвера и последующей работы с ФР.**

После установки в jpos.xml всех необходимых параметров.

**Начинаем работу с ФР.**

piritSvc112.open("PIRIT-2F", null);

// где "PIRIT-2F" должно соответствовать параметру logicalName в jpos.xml.

piritSvc112.claim(1000);

// осуществляется соединение с ФР, в устанавливается таймаут ожидания пакетов от // драйвера 1000 мс.

piritSvc112.setDeviceEnabled(true);

int[] arg1 = {1};

String[] result1 = new String[1];

piritSvc112.directIO(34, arg1, result1);

//Выполняем метод directIO с номером 34 - Получение состояния печатающего //устройства и выводим результат.

System.out.println(result1[0]);

//Печатаем чек

//Открываем документ

piritSvc112.beginFiscalReceipt(true);

//Добавляем товарную позицию

piritSvc112.printRecItem("\*11196 KAMIS Зелень петрушки в пак. 8г \*", 20000, 1000, 1, 20021, "tax");

//Вводим оплату

piritSvc112.printRecTotal(0, 40000, "00");

//Закрываем документ

piritSvc112.endFiscalReceipt(true);

**Краткое руководство по использованию методов DirectIO.**

**Проверка состояния печатающего устройства.**

directIO(command, args, dataObject);

где:

command = 34

args = int[] arg1 - номер бита которому соответствует требуемый параметр:

0 - Принтер не готов

1 - В принтере нет бумаги

2 - Открыта крышка принтера

3 - Ошибка резчика принтера

7 - Нет связи с принтером

String[] dataObject = new String[1]; dataObject присвоит результат.

Например для получения статуса состояния крышки принтера необходимо вызвать:

String[] dataObject = new String[1];

int[] arg1 = {2};

directIO(34, arg1 , dataObject);

в результате в *dataObject* будет строка с флагом состояния крышки.

**Получить регистрационный номер ФН.**

directIO(command, args, dataObject);

где:

command = 29

args = null:

String[] dataObject = new String[1]; dataObject присвоит результат.

**Для включения/выключения отрезки чека необходим вызов метода directIO перед вызовом метода beginFiscalReceipt**

directIO(command, args, dataObject);

где:

command = 15

args[0] = 1 - не выполнять отрезку чека, 0 - выполнять

dataObject - не используется.

x и z отчет отрезаются всегда.

**Метод DirectIO для получения квитанций ОФД начиная с заданного номер чека и количество чеков.**

используемый метод - DirectIO(command,args,dataObject);

где:

command = 42

int[] args={2}

args[0] содержит номер от которого начнется выборка документов.

args[1] содержит количество запрашиваемых чеков.

Object[] dataObject = new Object[1]; - Массив Object в котором под индексом 0 будет коллекция типа Vector<PiritFNTicket> где каждый экземпляр PiritFNTicket соответствует квитанции полученной из ФН.

Для извлечения информации из квитанции в классe PiritFNTicket предусмотрены следующие методы :

getdocNumber() - вернет HEX массив с квитанцией ОФД.

getDocNumber() - вернет int номер чека.

isSigned() - вернет boolean статус - подписан ли документ в ОФД.

getDateOfDoc() - вернет String с датой и временем квитанции в формате - ddMMyyyyHHmm.

getTicketNumber() - вернет String с номером квитанции.

getValueFPD() - вернет String значение ФПД.

**Для включения/выключения печати блока расчета драйвером используется directIO**

directIO(command, args, dataObject);

где:

command = 16

args[0] = 1 - печатать блок, 0 - не печатать

dataObject - не используется.

**Для настройки атрибутов текста при печати свободной строкой драйвером используется directIO**

directIO(command, args, dataObject);

где:

command = 17

args[0] = номер атрибута.

dataObject - не используется.