|  |  |
| --- | --- |
|  | **Версия 1.13\_564** |

ШТРИХ-М: JPOS драйвер

Руководство пользователя

«ШТРИХ-М», 2019

Версия: 1.2

Дата: 31.01.2017

Добавлено описание DIO-комманд установки email и телефона при работе с ФН.

Версия: 1.1

Дата: 05.09.2016

# 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

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

{

// The AmountDecimalPlaces property was incorrectly spelled

// AmountDecimalPlace since version 1.3. In version 1.7.2 and later,

// the correct spelling is supported. The old version is left for

// Application and Device Service compatibility. The implementations

// of getAmountDecimalPlaces and getAmountDecimalPlace should be

// identical.

// Properties

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 подключение и инициализацию устройства. Если связаться с устройством с заданными параметрами не удается, драйвер может выполнить поиск устройства и запрограммировать в нем заданные параметры связи (это зависит от параметров searchByPortEnabled и searchByBaudRateEnabled). По умолчанию разрешен поиск на всех скоростях. При инициализации устройства драйвер может записывать данные таб лиц ФР из файла (см. параметры fieldsFileName, fieldsFilesPath).

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

Возвращает "Fiscal Printer Service , SHTRIH-M, 2016"

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

Возвращает 1013273

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

Возвращает свойство FreezeEvents, true – доставка событий запрещена, false – доставка событий разрешена.

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

Разрешает или запрещает доставку событий.

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

Возвращает строку вида "ШТРИХ-МИНИ-ФР-К, 12345678, ПО ФР: A4.12345, 01.01.2012, ПО ФП: 1.35.23456, 02.02.2012"

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

Возвращает строку вида "ШТРИХ-МИНИ-ФР-К, №12345678", то есть название устройства и его серийный номер

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

Текущее состояние устройства. Допустимые значения:

JPOS\_S\_CLOSED Устройство закрыто (не был вызван метод Open)JPOS\_S\_IDLE 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, если устройство имеет датчик крышки корпуса. Значение свойства зависит подключенной модели ФР.

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

Возвращает true, все модели ФР ШТРИХ-М имеют шрифт двойной ширины (шрифт 2 – шрифт удвоенной высоты и ширины, обычно им печатается слово ИТОГ на чеке).

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

Возвращает true, все модели ФР ШТРИХ-М 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 false, 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**;**

Значение зависит от модели ФР.

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

Значение зависит от модели ФР.

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

Значение зависит от модели ФР.

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

Возвращает true, драйвер поддерживает нефискальный режим.

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

Возвращает false, метод PrintRecItemAdjustment должен вызываться после PrintRecItem.

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

Возвращает true, в методе PrintRecItemAdjustment можно задавать значение скидки/надбавки в процентах.

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 ШТРИХ-М не поддерживают печать отчета об отключении питания.

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

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

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

Возвращает true, драйвер поддерживает названия платежей, список которых определен в свойстве PredefinedPaymentLines.

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

Возвращает false, метод [PrintRecNotPaid](#PrintRecNotPaid) нельзя использовать для печати неоплаченной части итога чека.

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

Значение зависит от модели ФР.

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

Значение зависит от модели ФР.

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

Возвращает true, на всех ФР есть чековая станция.

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

Возвращает true, ФР может сообщить количество оставшихся записей в ФП (количество оставшихся смен).

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 false, драйвер не 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**;**

Поддерживает ли ФР датчик наличия подкладного документа. Возвращаемое значение зависит от модели ФР.

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, хотя сам ФР поддерживает.

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

Возвращает false, драйвер не поддерживает полноразмерные подкладные документы.

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

Поддерживает ли ФР датчик конца подкладного документа. Возвращаемое значение зависит от модели ФР.

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

Поддерживает ли ФР печать подкладных документов. Возвращаемое значение зависит от модели ФР.

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**;**

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 идентификатор последнего события.

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

Возвращает тип уведомления о питании принтера. По умолчанию уведомление отключено, powerNotify= PN\_DISABLED.

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

Устанавливает тип уведомления о питании принтера. Возможные значения:

JPOS\_PN\_ENABLED, JPOS\_PN\_DISABLED.

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

Возвращает текущее состояние питания принтера. Возможные значения:

PS\_UNKNOWN Невозможно определить состояние принтера по следующим причинам:

**CapPowerReporting** = PR\_NONE; устройство не поддерживает уведомление о питании.

**PowerNotify** = PN\_DISABLED; уведомления о питании запрещены.

**DeviceEnabled** = false; отслеживание состояния питания не ведется, пока устройство не будет включено.

PS\_ONLINE Устройство включено и готово к работе. Возвращается если   
 **CapPowerReporting** = PR\_STANDARD или PR\_ADVANCED.

PS\_OFF Устройство не включено или отключено от POS терминала. Возвращается если   
 **CapPowerReporting** = PR\_ADVANCED.

PS\_OFFLINE Устройство включено, но не готово или не может отвечать на запросы. Возвращается если  
 **CapPowerReporting** = PR\_ADVANCED.

PS\_OFF\_OFFLINE Устройство выключено или отключено от POS терминала. Возвращается если   
 **CapPowerReporting** = PR\_STANDARD.

Это свойство инициализируется значением PS\_UNKNOWN после вызова метода **open**. Если **PowerNotify** разрешено и **DeviceEnabled** = true, то это свойство обновляется как только сервис определит изменение состояния.

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

Возвращает количество десятичных разрядов, которые ФР использует для вычислений.

Свойство инициализируется, когда устройство включается (DeviceEnabled=true).

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

Если true, то некоторые методы будут выполняться асинхронно, такие методы как **printRecItemAdjustment**, **printRecItem**, **printNormal**, и т.д.

Свойство инициализируется после вызова метода **open**.

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

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**;**

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

{

// The AmountDecimalPlaces property was incorrectly spelled

// AmountDecimalPlace since version 1.3. In version 1.7.2 and later,

// the correct spelling is supported. The old version is left for

// Application and Device Service compatibility. The implementations

// of getAmountDecimalPlaces and getAmountDecimalPlace should be

// identical.

// Properties

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 ControlObject, то есть библиотека jpos. Драйвер получает JposEntry.

1. Тип порта: 0 - последовательный порт, 1 - bluetooth, 2 - socket, 3 - создание класса по названию

<!-- Port type: 0 - serial, 1 - bluetooth, 2 - socket, 3 - from parameter protClass -->

<prop name=**"portType"** type=**"String"** value=**"0"**/>

1. Название класса порта

<!-- portClass -->

<prop name=**"portClass"** type=**"String"** value=**"com.shtrih.fiscalprinter.port.SerialPrinterPort"**/>

1. Тип протокола: 0 - протокол 1.0, 1 - протокол версии 2.0

<!-- ProtocolType, 0 - protocol 1, 1 - protocol 2 -->

<prop name=**"protocolType"** type=**"String"** value=**"0"**/>

1. Имя порта.

<!--Port name-->

<prop name=**"portName"** type=**"String"** value=**"COM1"**/>

1. Скорость связи

<prop name=**"baudRate"** type=**"String"** value=**"115200"**/>

1. Отдел по умолчанию. Можно изменить также через directIO

<!--Default department-->

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

1. Номер шрифта по умолчанию. Можно изменить также через directIO

<!-- Default font number -->

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

1. Текст для закрытия чека

<!-- Close receipt text -->

<prop name=**"closeReceiptText"** type=**"String"** value=**""**/>

1. Текст подитога, печатается в методе printRecSubtotal

<!-- Subtotal text -->

<prop name=**"subtotalText"** type=**"String"** value=**"ПОДИТОГ:"**/>

1. Таймаут приема байта драйвера в миллисекундах.

<!-- Driver byte receive timeout -->

<prop name=**"byteTimeout"** type=**"String"** value=**"3000"**/>

1. Таймаут приема байта устройства в миллисекундах, записывается в ФР при инициализации

<!-- Device byte receive timeout -->

<prop name=**"deviceByteTimeout"** type=**"String"** value=**"3000"**/>

1. Разрешение поиска устройства на всех портах системы. Драйвер начинает поиск устройства, если не удалось подключиться к устройству с заданными параметрами. По умолчанию выключен.

<!-- Device search enabled for all serial ports -->

<prop name=**"searchByPortEnabled"** type=**"String"** value=**"0"**/>

1. Разрешение поиска устройства на всех скоростях. По умолчанию включен.

<!-- Device search enabled for all baud rates -->

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

1. Пароль налогового инспектора

<!-- Tax officer password -->

<prop name=**"taxPassword"** type=**"String"** value=**"0"**/>

1. Пароль оператора

<!-- Operator password -->

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

1. Пароль системного администратора

<!-- System administrator password -->

<prop name=**"sysAdminPassword"** type=**"String"** value=**"30"**/>

1. Разрешение опроса устройства. Опрос устройства нужен для оповещения приложения о

<!-- device state polling enabled -->

<prop name=**"pollEnabled"** type=**"String"** value=**"0"**/>

1. Интервал опроса в миллисекундах

<!-- device state polling interval in milliseconds -->

<prop name=**"pollInterval"** type=**"String"** value=**"100"**/>

1. Коэффициент для сумм

<!-- Amount coefficient -->

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

1. Коэффициент для количества

<!-- Quantity coefficient -->

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

1. Кодировка текстовых строк

<!-- Strings encoding -->

<prop name=**"stringEncoding"** type=**"String"** value=**"Cp866"**/>

1. Имя файла статистики

<!-- Statistics file name -->

<prop name=**"statisticFileName"** type=**"String"** value=**"ShtrihFiscalPrinter.xml"**/>

1. Задержка после печати штрихкода командой "Печать графической линии"

<!-- Barcode print time -->

<prop name=**"graphicsLineDelay"** type=**"String"** value=**"1000"**/>

1. Имя файла для записи таблиц ФР. Драйвер запишет эти данные в ФР при инициализации

<!-- fieldsFileName to initialize printer tables -->

<prop name=**"fieldsFileName"** type=**"String"** value=**"tables.csv"**/>

1. Имя папки для поиска файлов таблиц. Драйвер выбирает файл по имени устройства.

<!-- fieldsFilesPath to initialize printer tables -->

<prop name=**"fieldsFilesPath"** type=**"String"** value=**"I:\Projects\JavaPOS\Bin\tables\"**/>

1. Количество строк заголовка чека

<!-- Number of header lines -->

<prop name=**"numHeaderLines"** type=**"String"** value=**"4"**/>

1. Количество строк рекламного текста

<!-- Number of trailer lines -->

<prop name=**"numTrailerLines"** type=**"String"** value=**"3"**/>

1. Тип отчета, который снимается при вызове printReport: 0 - отчет по ФП, 1 - отчет по ЭКЛЗ

<!-- Device to print report, 0 - fiscal memory (FM), 1 - electronic journal (EJ) -->

<prop name=**"reportDevice"** type=**"String"** value=**"0"**/>

1. Тип отчета, 0 - краткий, 1 - полный

<!-- Report type, 0 - short, 1 - full -->

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

1. Команда для чтения состояния ФР: 0 - краткий запрос состояния 10h, 1 - полный запрос, 2 - автоматический выбор (10h, если поддерживается, иначе 11h)

<!-- Status command: -->

<!-- 0 - command 10h, short status request -->

<!-- 1 - command 11h, long status request -->

<!-- 2 - status command selected by driver -->

<prop name=**"statusCommand"** type=**"String"** value=**"0"**/>

1. Файл для всех сообщений драйвера

<!-- Localization file name -->

<prop name=**"messagesFileName"** type=**"String"** value=**"shtrihjavapos\_en.properties"**/>

1. Разрешение переноса длинных строк

<!-- Wrap text enabled -->

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

1. Задержка после закрытия чека. Может потребоваться для некоторых моделей ФР.

<!-- Sleep time after receipt close -->

<prop name=**"recCloseSleepTime"** type=**"String"** value=**"0"**/>

1. Тип отрезки чека, 0 - полная, 1 - неполная

<!-- Cut type, 0 - full cut, 1 - partial cut -->

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

1. Режим отрезки, 0 - автоматически, 1 - отрезка запрещена

<!-- Cut mode, 0 - auto, 1 - disabled -->

<prop name=**"cutMode"** type=**"String"** value=**"0"**/>

1. Параметр протокола ФР 1.0, максимальное количество запросов ENQ при передаче одной команды

<!-- maxEnqNumber -->

<prop name=**"maxEnqNumber"** type=**"String"** value=**"3"**/>

1. Параметр протокола ФР 1.0, максимальное количество ответов NAK при передаче одной команды, то есть максимальное количество ошибок при передаче команды.

<!-- maxNakCommandNumber -->

<prop name=**"maxNakCommandNumber"** type=**"String"** value=**"3"**/>

1. Параметр протокола ФР 1.0, максимальное количество ответов NAK при приеме ответа.

<!-- maxNakAnswerNumber -->

<prop name=**"maxNakAnswerNumber"** type=**"String"** value=**"3"**/>

1. Максимальное количество повторов команды

<!-- maxRepeatCount -->

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

1. Названия типов оплаты приложения

<!-- Payment types -->

<prop name=**"payType0"** type=**"String"** value=**"0"**/>

<prop name=**"payType39"** type=**"String"** value=**"3"**/>

1. Названия типов оплаты ФР

<!-- Payment names -->

<prop name=**"paymentName1"** type=**"String"** value=**"CASH"**/>

<prop name=**"paymentName2"** type=**"String"** value=**"CREDIT"**/>

<prop name=**"paymentName3"** type=**"String"** value=**"КАРТА"**/>

<prop name=**"paymentName4"** type=**"String"** value=**"СКИДКА"**/>

1. Названия налоговых групп

<!-- Tax names -->

<prop name=**"taxName0"** type=**"String"** value=**"НДС 10%"**/>

<prop name=**"taxName1"** type=**"String"** value=**"НДС 18%"**/>

<prop name=**"taxName2"** type=**"String"** value=**"C"**/>

<prop name=**"taxName3"** type=**"String"** value=**"D"**/>

1. Разрешение получения Z- отчета в виде XML файла. Драйвер сохраняет данные всех денежных и операционных регистров XML файле.

<!-- create Z-report in XML format -->

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

1. Добавлять номер смены к имени файла отчета.

<!-- Add day number to Z report filename - ZReport\_0001.xml -->

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

1. Имя XML файла Z отчета

<!-- XML Z-report file name -->

<prop name=**"XmlZReportFileName"** type=**"String"** value=**"ZReport.xml"**/>

1. Разрешение получения Z- отчета в виде CSV файла. Драйвер сохраняет данные всех денежных и операционных регистров CSV файле.

<!-- create Z-report in CSV format -->

<prop name=**"CsvZReportEnabled"** type=**"String"** value=**"0"**/>

1. Имя CSV файла Z отчета

<!-- CSV Z-report file name -->

<prop name=**"CsvZReportFileName"** type=**"String"** value=**"ZReport.csv"**/>

1. Разрешение обработки ESC команд в тексте. Это нужно для поддержки некоторых устаревших приложений

<!-- ESC commands enabled -->

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

1. Режим записи таблиц, 0 - автоматически, 1 - запрещена запись в таблицы

<!-- Table mode, 0 - auto, 1 - disabled -->

<prop name=**"tableMode"** type=**"String"** value=**"0"**/>

1. Режим логотипа перед заголовком чека, 0 - промотка чека на величину клише, 1 - печатать логотип в 2 этапа

<!-- Logo mode, 0 - feed paper, 1 - split image -->

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

1. Режим поиска ФР. 0 - нет поиска, 1 - поиск ФР при ошибках

<!-- SearchMode, 0 - none, 1 - search on error -->

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

1. Задержка после отрезки чека в миллисекундах

<!-- Paper cut delay in milliseconds -->

<prop name=**"cutPaperDelay"** type=**"String"** value=**"0"**/>

1. Тип чека продажи, 0 - обычный, 1 - текстовый чек

<!-- Sales receipt type, 0 - normal, 1 - GLOBUS -->

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

1. Длина поля "цена" для формата чека

<!-- Amount field length -->

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

1. Длина поля "количество" для формата чека

<!-- Quantity field length -->

<prop name=**"RFQuantityLength"** type=**"String"** value=**"10"**/>

1. Порт для системы мониторинга. Система мониторинга нужна для удаленного запроса состояния ФР и ЭКЛЗ

<!-- Monitoring server port -->

<prop name=**"MonitoringPort"** type=**"String"** value=**"50000"**/>

1. Разрешение работы системы мониторинга

<!-- Monitoring enabled -->

<prop name=**"MonitoringEnabled"** type=**"String"** value=**"0"**/>

1. Разрешение сохранения чека в текстовом виде

<!-- Receipt report enabled -->

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

1. Название файла чека

<!-- Receipt report file name -->

<prop name=**"receiptReportFileName"** type=**"String"** value=**"ZCheckReport.xml"**/>

1. Тип открытия чека, 0 - открыть чек при печати позиции, 1 - открыть чека в методе beginFiscalReceipt

<!-- openReceiptType, 0 - open receipt on item print, 1 - open receipt in beginFiscalReceipt -->

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

1. Режим заголовка, 0 - заголовок в драйвере, 1 - в принтере

<!-- headerMode, 0 - header in driver, 1 - header in printer -->

<prop name=**"headerMode"** type=**"String"** value=**"0"**/>

1. Позиция логотипа в заголовке чека

<!-- headerImagePosition, -->

<!-- SMFPTR\_LOGO\_AFTER\_HEADER = 0 -->

<!-- SMFPTR\_LOGO\_BEFORE\_TRAILER = 1 -->

<!-- SMFPTR\_LOGO\_AFTER\_TRAILER = 2 -->

<!-- SMFPTR\_LOGO\_AFTER\_ADDTRAILER = 3 -->

<!-- SMFPTR\_LOGO\_BEFORE\_HEADER = 4 -->

<prop name=**"headerImagePosition"** type=**"String"** value=**"0"**/>

1. Позиция логотипа в рекламном тексте

<!-- trailerImagePosition, -->

<!-- SMFPTR\_LOGO\_AFTER\_HEADER = 0 -->

<!-- SMFPTR\_LOGO\_BEFORE\_TRAILER = 1 -->

<!-- SMFPTR\_LOGO\_AFTER\_TRAILER = 2 -->

<!-- SMFPTR\_LOGO\_AFTER\_ADDTRAILER = 3 -->

<!-- SMFPTR\_LOGO\_BEFORE\_HEADER = 4 -->

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

1. Центрировать строки заголовка чека

<!-- Center header and trailer text automatically -->

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

1. Разрешение лога

<!-- Log file enabled -->

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

1. Передавать ENQ перед каждой командой

<!-- Send ENQ before every command or not -->

<prop name=**"sendENQ"** type=**"String"** value=**"0"**/>

1. Печатать буквы налоговых групп на текстовом чеке

<!-- Enable tax letters for GLOBUS receipt -->

<prop name=**"taxLettersEnabled"** type=**"String"** value=**"0"**/>

1. Префикс штрихкода для печати штрихкода как текста

<!-- Barcode prefix -->

<prop name=**"barcodePrefix"** type=**"String"** value=**"#\*~\*#"**/>

1. Тип штрихкода:   
   UPCA=0, UPCE=1, EAN13=2, EAN8=3, CODE39=4, ITF=5, CODABAR=6, CODE93=7, CODE128=8, PDF417=10, GS1\_OMNI=11, GS1\_TRUNC=12, GS1\_LIMIT=13, GS1\_EXP=14, GS1\_STK=15, GS1\_STK\_OMNI=16, GS1\_EXP\_STK=17, AZTEC=18, DATA\_MATRIX=19, MAXICODE=20, QR\_CODE=21, RSS\_14=22, RSS\_EXPANDED=23, UPC\_EAN\_EXTENSION=24

<!-- Barcode type -->

<!-- UPCA=0, UPCE=1, EAN13=2, EAN8=3, CODE39=4, ITF=5, CODABAR=6, CODE93=7, CODE128=8, PDF417=10, GS1\_OMNI=11, -->

<!-- GS1\_TRUNC=12, GS1\_LIMIT=13, GS1\_EXP=14, GS1\_STK=15, GS1\_STK\_OMNI=16, GS1\_EXP\_STK=17, AZTEC=18, DATA\_MATRIX=19, -->

<!-- MAXICODE=20, QR\_CODE=21, RSS\_14=22, RSS\_EXPANDED=23, UPC\_EAN\_EXTENSION=24 -->

<prop name=**"barcodeType"** type=**"String"** value=**"21"**/>

1. Ширина штриха в точках. Обычно 2-3 для одномерного штрихкода, 3-4 для двумерного

<!-- Barcode bar/module width -->

<prop name=**"barcodeBarWidth"** type=**"String"** value=**"4"**/>

1. Высота штрихкода. Имеет значение для одномерных штрихкодов.

<!-- Barcode height -->

<prop name=**"barcodeHeight"** type=**"String"** value=**"100"**/>

1. Позиция текста относительно штрихкода: 0 - не печатать, 1 - сверху, 2 - снизу, 3 - сверху и снизу

<!-- Barcode text position NOTPRINTED=0, ABOVE=1, BELOW=2, BOTH=3 -->

<prop name=**"barcodeTextPosition"** type=**"String"** value=**"2"**/>

1. Штрифт текста штрихкода, 1..7

<!-- Barcode text font, 1..7, default 1 -->

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

1. Соотношение ширины и высоты для штрихкода

<!-- Barcode aspect ratio -->

<prop name=**"barcodeAspectRatio"** type=**"String"** value=**"3"**/>

1. Совместимость со старыми версиями драйвера: 0 - нет, 1 - полная

<!-- Compatibility level, 0 - NONE, 1 - FULL -->

<prop name="compatibilityLevel" type="String" value="0"/>

1. Задержка после печати штрихкода в миллисекундах

<!-- Delay after barcode printed -->

<prop name=**"barcodeDelay"** type=**"String"** value=**"0"**/>

public FontNumber subtotalFont;

public FontNumber discountFont;

public String receiptVoidText = "ЧЕК ОТМЕНЕН";

public int taxPassword = 0;

public int usrPassword = 1;

public int sysPassword = 30;

public String statisticFileName = "ShtrihFiscalPrinter.xml";

public int pollInterval = 500;

public boolean pollEnabled = false;

public double amountFactor = 1;

public double quantityFactor = 1;

public boolean centerImage = true; // center BMP image

public String fieldsFileName = "";

public String fieldsFilesPath = "";

public int numHeaderLines;

public int numTrailerLines;

public String[] taxNames = {"А", "Б", "В", "Г"};

public final Vector paymentNames = new Vector();

public int reportDevice = SmFptrConst.SMFPTR\_REPORT\_DEVICE\_EJ;

public int reportType = PrinterConst.SMFP\_REPORT\_TYPE\_FULL;

public String messagesFileName = "";

public boolean wrapText = true;

public int cutType = PrinterConst.SMFP\_CUT\_PARTIAL;

public int maxEnqNumber = 10;

public int maxNakCommandNumber = 3;

public int maxNakAnswerNumber = 3;

public int maxAckNumber = 3;

public int maxRepeatCount = 3;

public boolean xmlZReportEnabled = false;

public boolean csvZReportEnabled = false;

public String xmlZReportFileName = "ZReport.xml";

public String csvZReportFileName = "ZReport.csv";

public static final boolean defaultAutoOpenDrawer = false;

public boolean autoOpenDrawer = defaultAutoOpenDrawer;

public int tableMode = SmFptrConst.SMFPTR\_TABLE\_MODE\_AUTO;

public int cutMode = SmFptrConst.SMFPTR\_CUT\_MODE\_AUTO;

public int logoMode = SmFptrConst.SMFPTR\_LOGO\_MODE\_SPLIT\_IMAGE;

public int searchMode = SmFptrConst.SMFPTR\_SEARCH\_NONE;

public String preLine = "";

public String postLine = "";

public boolean zeroPriceFilterEnabled = false;

public String zeroPriceFilterTime1 = "21:00";

public String zeroPriceFilterTime2 = "11:00";

public String zeroPriceFilterErrorText = "";

public boolean discountFilterEnabled = false;

public int RFAmountLength = 8;

public int RFQuantityLength = 10;

public int RFSeparatorLine = SmFptrConst.SMFPTR\_SEPARATOR\_LINE\_DASHES;

private final PayTypes payTypes = new PayTypes();

public int salesReceiptType = SmFptrConst.SMFPTR\_RECEIPT\_NORMAL;

public int cutPaperDelay = 0;

private int monitoringPort = 50000;

public boolean monitoringEnabled = false;

public boolean receiptReportEnabled = false;

public String receiptReportFileName = "ZCheckReport.xml";

public int protocolType = SmFptrConst.SMFPTR\_PROTOCOL\_1;

public int openReceiptType = SmFptrConst.SMFPTR\_OPEN\_RECEIPT\_ITEM;

public boolean ZReportDayNumber = false;

public int headerMode = SmFptrConst.SMFPTR\_HEADER\_MODE\_DRIVER;

private static CompositeLogger logger = CompositeLogger.getLogger(FptrParameters.class);

public int headerImagePosition = SmFptrConst.SMFPTR\_LOGO\_AFTER\_HEADER;

public int trailerImagePosition = SmFptrConst.SMFPTR\_LOGO\_AFTER\_TRAILER;

public boolean centerHeader = false;

public boolean cancelIO = false;

public boolean logEnabled = false;

public boolean stripExceptionDetails = true;

public boolean sendENQ = true;

private boolean taxLettersEnabled = true;

public boolean zeroReceiptEnabled = true;

public String barcodePrefix = "BARCODE:";

public int barcodeType = SmFptrConst.SMFPTR\_BARCODE\_QR\_CODE;

public int barcodeBarWidth = 2;

public int barcodeHeight = 100;

public int barcodeTextPosition = SmFptrConst.SMFPTR\_TEXTPOS\_BELOW;

public int barcodeTextFont = 1;

public int barcodeAspectRatio = 3;

public boolean FSServiceEnabled = false;

public String FSHost = "k-server.test-naofd.ru";

public int FSPort = 7779;

public int FSConnectTimeout = 3000; // 3000 ms

public int FSPollPeriod = 3000; // 3000 ms - poll period

public int compatibilityLevel = SmFptrConst.SMFPTR\_COMPAT\_LEVEL\_NONE;

public boolean checkTotal = false;

public boolean checkTotalEnabled = false;

public int receiptNumberRequest = SmFptrConst.SMFPTR\_RN\_FP\_DOCUMENT\_NUMBER;

public boolean FSDiscountEnabled = true;

public boolean FSCombineItemAdjustments = true;

public boolean textReportEnabled = false;

public boolean readDiscountMode = true;

public String textReportFileName = "documents.txt";

public boolean FSPrintTags = false;

public int FSTagsPlacement = 0;

public boolean textReportEmptyLinesEnabled = true;

public boolean ReceiptTemplateEnabled = false;

public String ItemTableHeader = null;

public String ItemTableTrailer = null;

public String ItemRowFormat = null;

public String discountFormat = null;

public String chargeFormat = null;

public boolean capJrnPresent = true;

public boolean nonFiscalHeaderEnabled = false;

public boolean fsHeaderEnabled = false;

public int nonFiscalDocNumber = 1;

public String fieldSeparator = "\n";

public int printerIDMode = PrinterConst.PRINTER\_ID\_SERIAL;

public int swapGraphicsLine = PrinterConst.SWAP\_LINE\_AUTO;

public boolean subtotalTextEnabled = true;

public String weightUnitName = "г.";

public String firmwarePath = "firmware";

public boolean graphicsLineEnabled = true;

public String preLinePrefix = "";

public String postLinePrefix = "";

public boolean combineReceiptItems = false;

public boolean printRecVoidItemAmount = false;

public boolean FSReceiptItemDiscountEnabled = false;

public boolean quantityCorrectionEnabled = false;

public boolean paymentSumCorrectionEnabled = true;

public long[] taxAmount = new long[6];

public int taxSystem = 0;

public Long itemTotalAmount = null;

public Long itemTaxAmount = null;

public int paymentType = 4;

public int subjectType = 1;

public boolean calcReportEnabled = false;

public boolean openReceiptOnBegin = false;

public boolean printVoidedReceipt = false;

public boolean capUpdateFirmware = true;

public boolean capScocUpdateFirmware = false;

public int printStringDelayInMs = 0;

public boolean fastConnect = false;

public boolean statisticEnabled = true;

public boolean autoOpenShift = true;

public boolean forceOpenShiftOnZReport = true;

public boolean footerFlagEnabled = true;

public boolean postLineAsItemTextEnabled = false;

public boolean canDisableNonFiscalEnding = true;

public boolean checkItemCodeEnabled = false;

public int newItemStatus = FSCheckBarcode.FS\_ITEM\_STATUS\_RETAIL;

public int itemCheckMode = FSCheckBarcode.FS\_CHECK\_MODE\_FULL;

public int itemMarkType = FptrParameters.MARK\_TYPE\_TOBACCO;

public int userExtendedTagPrintMode = USER\_EXTENDED\_TAG\_PRINT\_MODE\_DRIVER;

public boolean jsonUpdateEnabled = false;

public int jsonUpdatePeriodInMinutes = 5;

public String jsonUpdateServerURL = "http://127.0.0.1:8888/check\_firmware";

public int writeTagMode = FptrParameters.WRITE\_TAG\_MODE\_IN\_PLACE;

public int commandDelayInMs = 0;

# Метод DirectIO

## **Выполнить произвольную команду.**

/\*\* Execute command object \*\*/

public static final int SMFPTR\_DIO\_COMMAND **=** 0x00**;**

Драйвер выполняет отдельную команду ФР. В качестве команды передается объект PrinterCommand. Например:

public byte**[]** executeCommand**(**byte**[]** tx**,** int timeout**)** **throws** JposException **{**

int**[]** data **=** **new** int**[**1**];**

Object**[]** object **=** **new** Object**[**2**];**

data**[**0**]** **=** timeout**;**

object**[**0**]** **=** tx**;**

directIO**(**SmFptrConst**.**SMFPTR\_DIO\_COMMAND**,** data**,** object**);**

byte**[]** rx **=** **(**byte**[])** object**[**1**];**

**return** rx**;**

**}**

## **Печать штрихкода.**

Параметры передаются через объект PrinterBarcode.

/\*\* Print barcode object \*\*/

public static final int SMFPTR\_DIO\_PRINT\_BARCODE\_OBJECT **=** 0x01**;**

public void printBarcode**(**PrinterBarcode barcode**)** **throws** JposException **{**

printer**.**directIO**(**SmFptrConst**.**SMFPTR\_DIO\_PRINT\_BARCODE\_OBJECT**,**

**null,** barcode**);**

**}**

## **Установка номера отдела.**

/\*\* Set department \*\*/

public static final int SMFPTR\_DIO\_SET\_DEPARTMENT **=** 0x02**;**

public void setDepartment**(**int department**)** **throws** JposException **{**

int**[]** data **=** **new** int**[**1**];**

int**[]** value **=** **new** int**[**1**];**

value**[**0**]** **=** department**;**

directIO**(**SmFptrConst**.**SMFPTR\_DIO\_SET\_DEPARTMENT**,** data**,** value**);**

**}**

## **Чтение номера отдела.**

/\*\* Get department \*\*/

public static final int SMFPTR\_DIO\_GET\_DEPARTMENT **=** 0x03**;**

public int getDepartment**()** **throws** JposException **{**

int**[]** data **=** **new** int**[**1**];**

int**[]** value **=** **new** int**[**1**];**

directIO**(**SmFptrConst**.**SMFPTR\_DIO\_GET\_DEPARTMENT**,** data**,** value**);**

**return** value**[**0**];**

**}**

## **Выполнить команду ФР.**

/\*\* Execute string command \*\*/

public static final int SMFPTR\_DIO\_STRCOMMAND **=** 0x04**;**

public String executeCommand**(**int code**,** int timeout**,** String inParams**)**

**throws** JposException **{**

int**[]** data **=** **new** int**[**1**];**

data**[**0**]** **=** code**;**

String**[]** lines **=** **new** String**[**3**];**

lines**[**0**]** **=** String**.**valueOf**(**timeout**);**

lines**[**1**]** **=** inParams**;**

directIO**(**SmFptrConst**.**SMFPTR\_DIO\_STRCOMMAND**,** data**,** lines**);**

String outParams **=** lines**[**2**];**

**return** outParams**;**

**}**

## **Чтение таблицы ФР**

/\*\* Read table command \*\*/

public static final int SMFPTR\_DIO\_READTABLE **=** 0x05**;**

public String readTable**(**int tableNumber**,** int rowNumber**,** int fieldNumber**)**

**throws** JposException **{**

String**[]** params **=** **new** String**[**4**];**

params**[**0**]** **=** String**.**valueOf**(**tableNumber**);**

params**[**1**]** **=** String**.**valueOf**(**rowNumber**);**

params**[**2**]** **=** String**.**valueOf**(**fieldNumber**);**

directIO**(**SmFptrConst**.**SMFPTR\_DIO\_READTABLE**,** **null,** params**);**

**return** params**[**3**];**

**}**

Параметр data - не используется. Параметр object - массив строк длины 4.

Строка с индексом 0 - номер таблицы

Строка с индексом 1 - номер ряда

Строка с индексом 2 - номер поля

Строка с индексом 3 - возвращаемое значение

## **Запись таблицы ФР**

/\*\* Write table command \*\*/

public static final int SMFPTR\_DIO\_WRITETABLE **=** 0x06**;**

public void writeTable**(**int tableNumber**,** int rowNumber**,** int fieldNumber**,**

String fieldValue**)** **throws** JposException **{**

String**[]** params **=** **new** String**[**4**];**

params**[**0**]** **=** String**.**valueOf**(**tableNumber**);**

params**[**1**]** **=** String**.**valueOf**(**rowNumber**);**

params**[**2**]** **=** String**.**valueOf**(**fieldNumber**);**

params**[**3**]** **=** fieldValue**;**

directIO**(**SmFptrConst**.**SMFPTR\_DIO\_WRITETABLE**,** **null,** params**);**

**}**

Параметр data - не используется. Параметр object - массив строк длины 4.

Строка с индексом 0 - номер таблицы

Строка с индексом 1 - номер ряда

Строка с индексом 2 - номер поля

Строка с индексом 3 - возвращаемое значение

## **Чтение наименования типа оплаты**

/\*\* Read payment type name \*\*/

public static final int SMFPTR\_DIO\_READ\_PAYMENT\_NAME **=** 0x07**;**

public String readPaymentName**(**int number**)** **throws** JposException **{**

int**[]** data **=** **new** int**[**1**];**

data**[**0**]** **=** number**;**

String**[]** fieldValue **=** **new** String**[**1**];**

directIO**(**SmFptrConst**.**SMFPTR\_DIO\_READ\_PAYMENT\_NAME**,** data**,** fieldValue**);**

**return** fieldValue**[**0**];**

**}**

В параметре data[0] передается номер типа оплаты от 1 до 4.

Параметр object является массивом строк длины 1. В первой строке массива возвращается название типа оплаты.

## **Запись наименования типа оплаты**

В параметре data[0] передается номер типа оплаты от 1 до 4.

Параметр object является массивом строк длины 1. В первой строке массива передается название типа оплаты.

/\*\* write payment type name \*\*/

public static final int SMFPTR\_DIO\_WRITE\_PAYMENT\_NAME **=** 0x08**;**

public void writePaymentName**(**int number**,** String value**)** **throws** JposException **{**

int**[]** data **=** **new** int**[**1**];**

data**[**0**]** **=** number**;**

String**[]** fieldValue **=** **new** String**[**1**];**

fieldValue**[**0**]** **=** value**;**

directIO**(**SmFptrConst**.**SMFPTR\_DIO\_WRITE\_PAYMENT\_NAME**,** data**,** fieldValue**);**

**}**

## **Чтение флага окончания смены (когда 24 часа смены истекли)**

/\*\* Read end of day flag \*\*/

public static final int SMFPTR\_DIO\_READ\_DAY\_END **=** 0x09**;**

public boolean readDayEnd**()** **throws** JposException **{**

int**[]** data **=** **new** int**[**1**];**

directIO**(**SmFptrConst**.**SMFPTR\_DIO\_READ\_DAY\_END**,** data**,** **null);**

**return** data**[**0**]** **!=** 0**;**

**}**

В параметре data передается массив int длиной 1. В элементе с индексом 0 передается флаг окончания смены

## **Печать штрихкода**

/\*\* Print barcode command \*\*/

public static final int SMFPTR\_DIO\_PRINT\_BARCODE **=** 0x0A**;**

public void printBarcode**(**String barcode**,** String label**,** int barcodeType**,**

int barcodeHeight**,** int printType**,** int barWidth**,** int textPosition**,**

int textFont**,** int aspectRatio**)** **throws** JposException **{**

Object**[]** params **=** **new** Object**[**9**];**

params**[**0**]** **=** barcode**;** // barcode data

params**[**1**]** **=** label**;** // barcode label

params**[**2**]** **=** **new** Integer**(**barcodeType**);** // barcode type

params**[**3**]** **=** **new** Integer**(**barcodeHeight**);** // barcode height in pixels

params**[**4**]** **=** **new** Integer**(**printType**);** // print type

params**[**5**]** **=** **new** Integer**(**barWidth**);** // barcode bar width in pixels

params**[**6**]** **=** **new** Integer**(**textPosition**);** // text position

params**[**7**]** **=** **new** Integer**(**textFont**);** // text font

params**[**8**]** **=** **new** Integer**(**aspectRatio**);** // narrow to width ratio, 3 by default

directIO**(**SmFptrConst**.**SMFPTR\_DIO\_PRINT\_BARCODE**,** **null,** params**);**

**}**

Параметр data не используется.

Параметр object является массивом объектов длиной 9.

params[0] - данные штрихкода, строка.

params[1] - текст штрихкода, строка.

params[2] - тип штрихкода, Integer

params[3] - высота штрихкода в точках, Integer

params[4] - метод печати штрихкода, Integer

params[5] - ширина элемента в точках, Integer

params[6] - положение текста, Integer

params[7] - шрифт текста, Integer

params[8] - соотношение ширины и высоты штрихкода, Integer

## **Загрузка изображения из файла**

/\*\* Load image from file \*\*/

public static final int SMFPTR\_DIO\_LOAD\_IMAGE **=** 0x0B**;**

public int loadImage**(**String fileName**)** **throws** JposException **{**

int**[]** data **=** **new** int**[**1**];**

String**[]** command **=** **new** String**[**1**];**

command**[**0**]** **=** fileName**;**

directIO**(**SmFptrConst**.**SMFPTR\_DIO\_LOAD\_IMAGE**,** data**,** command**);**

**return** data**[**0**];**

**}**

в параметре data[0] возвращается индекс загруженного изображения.

Параметр object является массивом строк длины 1. В первом элементе передается имя файла изображения.

## **Печать изображения**

/\*\* Print image \*\*/

public static final int SMFPTR\_DIO\_PRINT\_IMAGE **=** 0x0C**;**

public void printImage**(**int imageIndex**)** **throws** JposException **{**

int**[]** data **=** **new** int**[**1**];**

data**[**0**]** **=** imageIndex**;**

directIO**(**SmFptrConst**.**SMFPTR\_DIO\_PRINT\_IMAGE**,** data**,** **null);**

**}**

В параметре data[0] передается индекс загруженного изображения.

1. Удаление всех изображений

/\*\* Clear all images \*\*/

public static final int SMFPTR\_DIO\_CLEAR\_IMAGES **=** 0x0D**;**

public void clearImages**()** **throws** JposException **{**

directIO**(**SmFptrConst**.**SMFPTR\_DIO\_CLEAR\_IMAGES**,** **null,** **null);**

**}**

Параметры не используются

1. Добавление логотипа

/\*\* Set logo \*\*/

public static final int SMFPTR\_DIO\_ADD\_LOGO **=** 0x0E**;**

public void addLogo**(**int imageIndex**,** int logoPosition**)** **throws** JposException **{**

int**[]** data **=** **new** int**[**2**];**

data**[**0**]** **=** imageIndex**;**

data**[**1**]** **=** logoPosition**;**

directIO**(**SmFptrConst**.**SMFPTR\_DIO\_ADD\_LOGO**,** **null,** data**);**

**}**

Параметр data не используется, параметр object представляет собой массив int длиной 2.

Индекс 0 массива - индекс изображения

Индекс 1 массива - позиция логотипа

## **Удаление логотипов**

/\*\* Clear logo \*\*/

public static final int SMFPTR\_DIO\_CLEAR\_LOGO **=** 0x0F**;**

public void clearLogo**()** **throws** JposException **{**

directIO**(**SmFptrConst**.**SMFPTR\_DIO\_CLEAR\_LOGO**,** **null,** **null);**

**}**

Параметры не используются

## **Печать линии**

/\*\* Print black line \*\*/

public static final int SMFPTR\_DIO\_PRINT\_LINE **=** 0x10**;**

public void printLine**(**int lineType**,** int lineHeight**)** **throws** JposException **{**

int**[]** data **=** **new** int**[**2**];**

data**[**0**]** **=** lineHeight**;**

data**[**1**]** **=** lineType**;**

directIO**(**SmFptrConst**.**SMFPTR\_DIO\_PRINT\_LINE**,** **null,** data**);**

**}**

Параметр data не используется, параметр object представляет собой массив int длиной 2. Первый элемент массива - высота линии в точках, второй элемент - тип линии (0 - черная линия, 1 - белая).

public static final int SMFPTR\_LINE\_TYPE\_BLACK = 0;

public static final int SMFPTR\_LINE\_TYPE\_WHITE = 1;

## **Чтение параметра драйвера**

/\*\* Get driver parameter \*\*/

public static final int SMFPTR\_DIO\_GET\_DRIVER\_PARAMETER **=** 0x11**;**

public String getParameter(int paramType) throws JposException {

int data[] = new int[1];

String object[] = new String[1];

data[0] = paramType;

directIO(SmFptrConst.SMFPTR\_DIO\_GET\_DRIVER\_PARAMETER, data, object);

return object[0];

}

Номер параметра передается в параметре data. Параметр object представляет собой массив строк длиной 1. В первом элементе возвращается значение параметра в виде строки.

Номера параметров:

/\*\* Report device for printReport, printPeriodicTotalsReport \*\*/

public static final int SMFPTR\_DIO\_PARAM\_REPORT\_DEVICE **=** 0**;**

/\*\* Report type for printReport, printPeriodicTotalsReport \*\*/

public static final int SMFPTR\_DIO\_PARAM\_REPORT\_TYPE **=** 1**;**

/\*\* Number of header lines \*\*/

public static final int SMFPTR\_DIO\_PARAM\_NUMHEADERLINES **=** 2**;**

/\*\* Number of trailer lines \*\*/

public static final int SMFPTR\_DIO\_PARAM\_NUMTRAILERLINES **=** 3**;**

/\*\* Polling enabled \*\*/

public static final int SMFPTR\_DIO\_PARAM\_POLL\_ENABLED **=** 4**;**

/\*\* Cut mode \*\*/

public static final int SMFPTR\_DIO\_PARAM\_CUT\_MODE **=** 5**;**

/\*\* Font number \*\*/

public static final int SMFPTR\_DIO\_PARAM\_FONT\_NUMBER **=** 6**;**

## **Запись параметра драйвера**

/\*\* Set driver parameter \*\*/

public static final int SMFPTR\_DIO\_SET\_DRIVER\_PARAMETER **=** 0x12**;**

public void setParameter**(**int paramType**,** int paramValue**)**

**throws** JposException **{**

int data**[]** **=** **new** int**[**1**];**

String object**[]** **=** **new** String**[**1**];**

data**[**0**]** **=** paramType**;**

object**[**0**]** **=** String**.**valueOf**(**paramValue**);**

directIO**(**SmFptrConst**.**SMFPTR\_DIO\_SET\_DRIVER\_PARAMETER**,** data**,** object**);**

**}**

Номер параметра передается в параметре data. Параметр object представляет собой массив строк длиной 1. В первом элементе передается значение параметра в виде строки.

Номера параметров приведены выше.

## **Печать текста**

/\*\* Print text \*\*/

public static final int SMFPTR\_DIO\_PRINT\_TEXT **=** 0x13**;**

public void printText**(**String text**)** **throws** JposException **{**

int data**[]** **=** **new** int**[**1**];**

directIO**(**SmFptrConst**.**SMFPTR\_DIO\_PRINT\_TEXT**,** data**,** text**);**

**}**

Параметр data не используется, параметр object представляет текст для печати.

## **Запись таблиц ФР**

/\*\* Write table values from file \*\*/

public static final int SMFPTR\_DIO\_WRITE\_TABLES **=** 0x14**;**

public void writeTables**(**String fileName**)** **throws** JposException **{**

int**[]** data **=** **new** int**[**1**];**

directIO**(**SmFptrConst**.**SMFPTR\_DIO\_WRITE\_TABLES**,** data**,** fileName**);**

**}**

Параметр data не используется, параметр object представляет имя файла таблиц ФР.

## **Чтение таблиц ФР**

/\*\* Read table values to file \*\*/

public static final int SMFPTR\_DIO\_READ\_TABLES **=** 0x15**;**

public void readTables**(**String fileName**)** **throws** JposException **{**

int**[]** data **=** **new** int**[**1**];**

directIO**(**SmFptrConst**.**SMFPTR\_DIO\_READ\_TABLES**,** data**,** fileName**);**

**}**

Параметр data не используется, параметр object представляет имя файла таблиц ФР.

## **Чтение серийного номера ФР**

/\*\* Read device serial number \*\*/

public static final int SMFPTR\_DIO\_READ\_SERIAL **=** 0x16**;**

public String readSerial**()** **throws** JposException **{**

String**[]** serial **=** **new** String**[**1**];**

directIO**(**SmFptrConst**.**SMFPTR\_DIO\_READ\_SERIAL**,** **null,** serial**);**

**return** serial**[**0**];**

**}**

Параметр data не используется, параметр object представляет массив строк длины 1.

## **Чтение серийного номера ЭКЛЗ**

/\*\* Read EJ serial number \*\*/

public static final int SMFPTR\_DIO\_READ\_EJ\_SERIAL **=** 0x17**;**

public String readEJSerial**()** **throws** JposException **{**

String**[]** serial **=** **new** String**[**1**];**

directIO**(**SmFptrConst**.**SMFPTR\_DIO\_READ\_EJ\_SERIAL**,** **null,** serial**);**

**return** serial**[**0**];**

**}**

Параметр data не используется, параметр object представляет массив строк длины 1.

## **Открыть денежный ящик**

/\*\* Open cash drawer \*\*/

public static final int SMFPTR\_DIO\_OPEN\_DRAWER **=** 0x18**;**

public void openCashDrawer**(**int drawerNumber**)** **throws** JposException **{**

int**[]** data **=** **new** int**[**1**];**

data**[**0**]** **=** drawerNumber**;**

directIO**(**SmFptrConst**.**SMFPTR\_DIO\_OPEN\_DRAWER**,** data**,** **null);**

**}**

Параметр object не используется, в параметре data передается номер денежного ящика. Обычно денежный ящик один с номером 0

## **Прочитать состояние денежного ящика**

/\*\* Read cash drawer state \*\*/

public static final int SMFPTR\_DIO\_READ\_DRAWER\_STATE **=** 0x19**;**

public boolean readDrawerState**()** **throws** JposException **{**

int**[]** data **=** **new** int**[**1**];**

data**[**0**]** **=** 0**;**

directIO**(**SmFptrConst**.**SMFPTR\_DIO\_READ\_DRAWER\_STATE**,** data**,** **null);**

**return** data**[**0**]** **!=** 0**;**

**}**

Параметр object не используется, в параметре data возвращается состояние ящика. 0 - ящик закрыт, 1 - открыт.

## **Прочитать состояние принтера**

/\*\* Read printer status \*\*/

public static final int SMFPTR\_DIO\_READ\_PRINTER\_STATUS **=** 0x1A**;**

public PrinterStatus readPrinterStatus**()** **throws** JposException **{**

int**[]** data **=** **new** int**[**1**];**

Object**[]** object **=** **new** Object**[**1**];**

directIO**(**SmFptrConst**.**SMFPTR\_DIO\_READ\_PRINTER\_STATUS**,** data**,** object**);**

**return** **(**PrinterStatus**)**object**[**0**];**

**}**

Параметр data не используется, в параметре object возвращается объект состояния, PrinterStatus.

## **Прочитать денежный регистр**

/\*\* Read cash register \*\*/

public static final int SMFPTR\_DIO\_READ\_CASH\_REG **=** 0x1B**;**

public CashRegister readCashRegister**(**int number**)** **throws** JposException **{**

int**[]** data **=** **new** int**[**1**];**

Object**[]** object **=** **new** Object**[**1**];**

data**[**0**]** **=** number**;**

directIO**(**SmFptrConst**.**SMFPTR\_DIO\_READ\_CASH\_REG**,** data**,** object**);**

**return** **(**CashRegister**)**object**[**0**];**

**}**

В параметре data передается номер регистра, параметр object - массив объектов длины 1. В нем возвращается класс денежного регистра.

## **Прочитать операционный регистр**

/\*\* Read operation register \*\*/

public static final int SMFPTR\_DIO\_READ\_OPER\_REG **=** 0x1C**;**

public OperationRegister readOperRegister**(**int number**)** **throws** JposException **{**

int**[]** data **=** **new** int**[**1**];**

Object**[]** object **=** **new** Object**[**1**];**

data**[**0**]** **=** number**;**

directIO**(**SmFptrConst**.**SMFPTR\_DIO\_READ\_OPER\_REG**,** data**,** object**);**

**return** **(**OperationRegister**)**object**[**0**];**

**}**

В параметре data передается номер регистра, параметр object - массив объектов длины 1. В нем возвращается класс операционного регистра.

## **Выполнить команду**

/\* Execute command \*/

public static final int SMFPTR\_DIO\_COMMAND\_OBJECT **=** 0x1D**;**

public void executeCommand(PrinterCommand command) throws JposException {

int[] data = new int[1];

directIO(SmFptrConst.SMFPTR\_DIO\_COMMAND\_OBJECT, data, command);

}

Параметр data не используется, параметр object - массив объектов длины 1. В нем передается объект команды (класс PrinterCommand).

## **Сохранить данные Z отчета в XML файле**

/\* Save XML Z report \*/

public static final int SMFPTR\_DIO\_XML\_ZREPORT **=** 0x1E**;**

public void saveXmlZReport**(**String fileName**)** **throws** JposException **{**

int**[]** data **=** **new** int**[**1**];**

directIO**(**SmFptrConst**.**SMFPTR\_DIO\_XML\_ZREPORT**,** data**,** fileName**);**

**}**

Параметр data не используется, параметр object - имя XML файла.

## **Сохранить данные Z отчета в CSV файле**

/\* Save CSV Z report \*/

public static final int SMFPTR\_DIO\_CSV\_ZREPORT **=** 0x1F**;**

public void saveCsvZReport**(**String fileName**)** **throws** JposException **{**

int**[]** data **=** **new** int**[**1**];**

directIO**(**SmFptrConst**.**SMFPTR\_DIO\_CSV\_ZREPORT**,** data**,** fileName**);**

**}**

Параметр data не используется, параметр object - имя XML файла.

## **Запись параметра устройства**

/\* Write parameter \*/

public static final int SMFPTR\_DIO\_WRITE\_DEVICE\_PARAMETER **=** 0x20**;**

## **Чтение параметра устройства**

/\* Read parameter \*/

public static final int SMFPTR\_DIO\_READ\_DEVICE\_PARAMETER **=** 0x21**;**

## **Загрузка логотипа**

/\*\* Load logo \*\*/

public static final int SMFPTR\_DIO\_LOAD\_LOGO **=** 0x22**;**

public int loadLogo**(**int logoPosition**,** String fileName**)** **throws** JposException **{**

int**[]** data **=** **new** int**[**1**];**

data**[**0**]** **=** logoPosition**;**

directIO**(**SmFptrConst**.**SMFPTR\_DIO\_ADD\_LOGO**,** data**,** fileName**);**

**return** data**[**0**];**

**}**

В параметре data передается позиция логотипа, параметр object - имя файла логотипа. В параметре data возвращается индекс изображения.

## **Чтение состояния смены (открыта, закрыта, истекли 24 часа).**

/\*\* Read fiscal day status \*\*/

public static final int SMFPTR\_DIO\_READ\_DAY\_STATUS **=** 0x23**;**

public int readDayStatus**()** **throws** JposException **{**

int**[]** data **=** **new** int**[**1**];**

directIO**(**SmFptrConst**.**SMFPTR\_DIO\_READ\_DAY\_STATUS**,** data**,** **null);**

**return** data**[**0**];**

**}**

В параметре data возвращается состояние смены в ФР. Возможные значения:

public static final int SMFPTR\_DAY\_STATUS\_CLOSED **=** 1**;**

public static final int SMFPTR\_DAY\_STATUS\_OPENED **=** 2**;**

public static final int SMFPTR\_DAY\_STATUS\_EXPIRED **=** 3**;**

public static final int SMFPTR\_DAY\_STATUS\_UNKNOWN **=** 4**;**

## **Чтение номера лицензии.**

/\*\* Read device license number \*\*/

public static final int SMFPTR\_DIO\_READ\_LICENSE **=** 0x24**;**

public String readLicense**()** **throws** JposException **{**

int**[]** data **=** **new** int**[**1**];**

String**[]** object **=** **new** String**[**1**];**

directIO**(**SmFptrConst**.**SMFPTR\_DIO\_READ\_LICENSE**,** data**,** object**);**

**return** object**[**0**];**

**}**

Параметр data не используется, параметр object представляет собой массив строк длиной 1. В первом элементе возвращается номер лицензии в виде строки.

## **Чтение готовности к печати фискальных документов.**

/\*\* Is printer ready for fiscal documents \*\*/

public static final int SMFPTR\_DIO\_IS\_READY\_FISCAL **=** 0x25**;**

public boolean isReadyFiscal**(**String**[]** text**)** **throws** JposException **{**

int**[]** data **=** **new** int**[**1**];**

directIO**(**SmFptrConst**.**SMFPTR\_DIO\_IS\_READY\_FISCAL**,** data**,** text**);**

**return** data**[**0**]** **!=** 0**;**

**}**

В параметре data возвращается состояние, 0 или 1. 1 возвращается, если в ФР смена закрыта или открыта. Если 24 часа истекли, или ФР в другом состоянии, то возвращается 0.

## **Чтение готовности к печати фискальных документов.**

/\*\* Is printer ready for non fiscal documents \*\*/

public static final int SMFPTR\_DIO\_IS\_READY\_NONFISCAL **=** 0x26**;**

public boolean isReadyNonfiscal**(**String**[]** text**)** **throws** JposException **{**

int**[]** data **=** **new** int**[**1**];**

directIO**(**SmFptrConst**.**SMFPTR\_DIO\_IS\_READY\_NONFISCAL**,** data**,** text**);**

**return** data**[**0**]** **!=** 0**;**

**}**

В параметре data возвращается состояние, 0 или 1. 1 возвращается, если в ФР смена закрыта или открыта или 24 часа истекли. Если ФР в другом состоянии, то возвращается 0.

## **Определение максимального размера графики**

/\*\* Read maximum graphics size \*\*/

public static final int SMFPTR\_DIO\_READ\_MAX\_GRAPHICS **=** 0x27**;**

public int readMaxGraphics**()** **throws** JposException **{**

int**[]** data **=** **new** int**[**1**];**

directIO**(**SmFptrConst**.**SMFPTR\_DIO\_READ\_MAX\_GRAPHICS**,** data**,** **null);**

**return** data**[**0**];**

**}**

Драйвер выполняет определение при помощи команды 0xC3, Расширенная печать графики. Драйвер действует методом половинного деления. При определении принтер будет подавать сигналы об ошибке.

## **Чтение строки заголовка чека**

/\*\* Read header line \*\*/

public static final int public String getHeaderLine**(**int lineNumber**)** **throws** JposException **{**

int**[]** data **=** **new** int**[**1**];**

String**[]** lines **=** **new** String**[**1**];**

data**[**0**]** **=** lineNumber**;**

directIO**(**SmFptrConst**.**SMFPTR\_DIO\_GET\_HEADER\_LINE**,** data**,** lines**);**

**return** lines**[**0**];**

**}**

В параметре data передается номер строки. Параметр object представляет собой массив строк длины 1. В первом элементе передается строка заголовка.

1. Чтение строки рекламного текста

/\*\* Read trailer line \*\*/

public static final int SMFPTR\_DIO\_GET\_TRAILER\_LINE **=** 0x29**;**

public String getTrailerLine**(**int lineNumber**)** **throws** JposException **{**

int**[]** data **=** **new** int**[**1**];**

String**[]** lines **=** **new** String**[**1**];**

data**[**0**]** **=** lineNumber**;**

directIO**(**SmFptrConst**.**SMFPTR\_DIO\_GET\_TRAILER\_LINE**,** data**,** lines**);**

**return** lines**[**0**];**

**}**

В параметре data передается номер строки. Параметр object представляет собой массив строк длины 1. В первом элементе возвращается строка заголовка.

1. Получение длины строки

/\*\* Read text length \*\*/

public static final int SMFPTR\_DIO\_GET\_TEXT\_LENGTH **=** 0x2A**;**

public int getTextLength**(**int fontNumber**)** **throws** JposException **{**

int**[]** data **=** **new** int**[**1**];**

data**[**0**]** **=** fontNumber**;**

directIO**(**SmFptrConst**.**SMFPTR\_DIO\_GET\_TEXT\_LENGTH**,** data**,** **null);**

**return** data**[**0**];**

**}**

В параметре data передается номер шрифта. Параметр object не используется. В параметре data возвращается длина строки.

1. Чтение имени кассира

/\*\* Read cashier name \*\*/

public static final int SMFPTR\_DIO\_READ\_CASHIER\_NAME **=** 0x2B**;**

public String readCashierName**()** **throws** JposException **{**

String**[]** lines **=** **new** String**[**1**];**

directIO**(**SmFptrConst**.**SMFPTR\_DIO\_READ\_CASHIER\_NAME**,** **null,** lines**);**

**return** lines**[**0**];**

**}**

Параметр data не используется. Параметр object представляет собой массив строк длины 1. В первом элементе возвращается имя кассира.

1. Запись имени кассира

/\*\* Write cashier name \*\*/

public static final int SMFPTR\_DIO\_WRITE\_CASHIER\_NAME **=** 0x2C**;**

public void writeCashierName**(**String value**)** **throws** JposException **{**

String**[]** lines **=** **new** String**[**1**];**

lines**[**0**]** **=** value**;**

directIO**(**SmFptrConst**.**SMFPTR\_DIO\_WRITE\_CASHIER\_NAME**,** **null,** lines**);**

**}**

Параметр data не используется. Параметр object представляет собой массив строк длины 1. В первом элементе передается имя кассира.

1. Отрезка чека

/\*\* Cut paper \*\*/

public static final int SMFPTR\_DIO\_CUT\_PAPER **=** 0x2D**;**

// cutMode: 0 - full cut, 1 - partial cut

public void cutPaper**(**int cutMode**)** **throws** JposException **{**

int**[]** data **=** **new** int**[**1**];**

data**[**0**]** **=** cutMode**;**

directIO**(**SmFptrConst**.**SMFPTR\_DIO\_CUT\_PAPER**,** data**,** **null);**

**}**

Параметр object не используется. В параметре data передается тип отрезки: 0 – полная, 1 - неполная.

1. Ожидание завершения печати

/\*\* Wait for printing \*\*/

public static final int SMFPTR\_DIO\_WAIT\_PRINT **=** 0x2E**;**

public void waitForPrinting**()** **throws** JposException **{**

directIO**(**SmFptrConst**.**SMFPTR\_DIO\_WAIT\_PRINT**,** **null,** **null);**

**}**

Параметры data и object не используются. Метод опрашивает состояние принтера и ждет завершения печати.

1. Краткий запрос состояния

/\*\* Short status \*\*/

public static final int SMFPTR\_DIO\_READ\_SHORT\_STATUS **=** 0x30**;**

public ShortPrinterStatus readShortPrinterStatus**()** **throws** JposException **{**

Object**[]** object **=** **new** Object**[**1**];**

directIO**(**SmFptrConst**.**SMFPTR\_DIO\_READ\_SHORT\_STATUS**,** **null,** object**);**

**return** **(**ShortPrinterStatus**)**object**[**0**];**

**}**

Параметр data не используется. Параметр object представляет собой массив объектов. В первом элементе возвращается краткий запрос состояния, класс ShortPrinterStatus.

1. Полный запрос состояния

/\*\* Long status \*\*/

public static final int SMFPTR\_DIO\_READ\_LONG\_STATUS **=** 0x31**;**

public LongPrinterStatus readLongPrinterStatus**()** **throws** JposException **{**

Object**[]** object **=** **new** Object**[**1**];**

directIO**(**SmFptrConst**.**SMFPTR\_DIO\_READ\_LONG\_STATUS**,** **null,** object**);**

**return** **(**LongPrinterStatus**)**object**[**0**];**

**}**

Параметр data не используется. Параметр object представляет собой массив объектов. . В первом элементе возвращается полный запрос состояния, класс LongPrinterStatus.

1. Запись строки тега

Для записи в ФР строкового тега можно использовать метод fsWriteTag:

public void fsWriteTag**(**int tagId**,** String tagValue**)** **throws** Exception

**{**

int**[]** data **=** **new** int**[**1**];**

data**[**0**]** **=** tagId**;**

directIO**(**SmFptrConst**.**SMFPTR\_DIO\_FS\_WRITE\_TAG**,** data**,** tagValue**);**

**}**

tagId - идентификатор тега, идентификаторы приведены в документе " Форматы фискальных документов". tagId = 1008 - адрес покупателя. tagValue - строка значения тега.

1. Запись в чек email покупателя при работе с ФН.

public static final int SMFPTR\_DIO\_FS\_WRITE\_CUSTOMER\_EMAIL **=** 0x39**;**

public void fsWriteCustomerEmail**()** **throws** JposException **{**

String email **=** ”foo@example.com”**;**

directIO**(**SMFPTR\_DIO\_FS\_WRITE\_CUSTOMER\_EMAIL**,** **null,** email**);**

**}**

В параметре object передается строка содержащая email покупателя.

1. Запись в чек телефона покупателя при работе с ФН.

public static final int SMFPTR\_DIO\_FS\_WRITE\_CUSTOMER\_PHONE **=** 0x3A**;**

public void fsWriteCustomerPhone**()** **throws** JposException **{**

String phoneNumber **=** ”89261112233”**;**

directIO**(**SMFPTR\_DIO\_FS\_WRITE\_CUSTOMER\_PHONE**,** **null,** phoneNumber**);**

**}**

В параметре object передается строка содержащая телефон покупателя.

1. Печать отчета о состоянии расчетов ФН.

public static final int SMFPTR\_DIO\_FS\_PRINT\_CALC\_REPORT **=** 0x3B**;**

public void fsPrintCalcReport() throws JposException {

directIO(SmFptrConst.SMFPTR\_DIO\_FS\_PRINT\_CALC\_REPORT, null, null);

}

1. Печато контрольной ленты.

public static final int SMFPTR\_DIO\_PRINT\_JOURNAL **=** 0x3C**;**

/\*\*

\* Печать СКЛ за текущую смену

\*/

public void printJournalCurrentDay**()** **throws** JposException **{**

int**[]** params **=** **new** int**[**1**];**

params**[**0**]** **=** SmFptrConst**.**SMFPTR\_JRN\_REPORT\_CURRENT\_DAY**;**

directIO**(**SmFptrConst**.**SMFPTR\_DIO\_PRINT\_JOURNAL**,** **null,** params**);**

**}**

/\*\*

\* Печать СКЛ за указанную смену

\*/

public void printJournalDayNumber**(**int dayNumber**)** **throws** JposException **{**

int**[]** params **=** **new** int**[**2**];**

params**[**0**]** **=** SmFptrConst**.**SMFPTR\_JRN\_REPORT\_DAY\_NUMBER**;**

params**[**1**]** **=** dayNumber**;**

directIO**(**SmFptrConst**.**SMFPTR\_DIO\_PRINT\_JOURNAL**,** **null,** params**);**

**}**

/\*\*

\* Печать документа из СКЛ с указанным номером

\*/

public void printJournalDocNumber**(**int docNumber**)** **throws** JposException **{**

int**[]** params **=** **new** int**[**2**];**

params**[**0**]** **=** SmFptrConst**.**SMFPTR\_JRN\_REPORT\_DOC\_NUMBER**;**

params**[**1**]** **=** docNumber**;**

directIO**(**SmFptrConst**.**SMFPTR\_DIO\_PRINT\_JOURNAL**,** **null,** params**);**

**}**

/\*\*

\* Печать документа из СКЛ с указанным номером

\*/

public void printJournalDocRange**(**int docNumber1**,** int docNumber2**)** **throws** JposException **{**

int**[]** params **=** **new** int**[**3**];**

params**[**0**]** **=** SmFptrConst**.**SMFPTR\_JRN\_REPORT\_DOC\_RANGE**;**

params**[**1**]** **=** docNumber1**;**

params**[**2**]** **=** docNumber2**;**

directIO**(**SmFptrConst**.**SMFPTR\_DIO\_PRINT\_JOURNAL**,** **null,** params**);**

**}**

1. Задать сумму скидки на чек от 1 до 99 копеек.

public static final int SMFPTR\_DIO\_SET\_DISCOUNT\_AMOUNT **=** 0x3D**;**

public void setDiscountAmount**(**int amount**)** **throws** Exception **{**

directIO**(**SmFptrConst**.**SMFPTR\_DIO\_SET\_DISCOUNT\_AMOUNT**,** **null,**

**new** Integer**(**amount**));**

**}**

1. Прочитать параметры ФН.

public static final int SMFPTR\_DIO\_READ\_FS\_PARAMS **=** 0x3E**;**

В параметре object должен быть передан список строк (List<String>), в который будут добавлены значения. Порядок значений в списке:

0 - серийный номер ФН

1 - РНМ

2 - кол-во неотправленных ФД

3 - дата и время самого раннего неотправленного ФД (DDMMYYYYhhmm)

4 - номер документа последней перерегистрации

5 - дата и время последней перерегистрации (DDMMYYYYhhmm)

6 - дата окончания срока действия (DDMMYYYY)

7 - кол-во чеков за смену

8 - номер смены

9 - номер последнего ФД

10 - дата и время последнего ФД (DDMMYYYYhhmm)

11 - фискальный признак последнего ФД

12 - ЗН ККТ

13 - Модель ФР

14 - Наименование ОФД

15 - Адрес сервера ОФД

16 - № порта

17 - Таймаут ожидание ответа от ОФД

1. Чтение квитанций ФН.

public static final int SMFPTR\_DIO\_READ\_FS\_TICKETS **=** 0x3F**;**

public String**[]** fsReadTickets**(**int**[]** numbers**)** **throws** Exception **{**

Object**[]** object **=** **new** Object**[**1**];**

directIO**(**SmFptrConst**.**SMFPTR\_DIO\_READ\_FS\_TICKETS**,** numbers**,** object**);**

**return** **(**String**[])** object**[**0**];**

**}**

Позволяет читать квитанции по номерам документов ФН. Возвращает список строк, где каждая строка соответствует тексту квитанции. В тексте квитанции передается через точку с запятой дата, время, ФПД, номер ФД.

1. Чтение квитанций ФН.

public static final int SMFPTR\_DIO\_READ\_FS\_TICKETS2 **=** 0x40**;**

public Vector**<**FSTicket**>** fsReadTickets2**(**int**[]** numbers**)** **throws** Exception **{**

Object**[]** object **=** **new** Object**[**1**];**

object**[**0**]** **=** **null;**

directIO**(**SmFptrConst**.**SMFPTR\_DIO\_READ\_FS\_TICKETS2**,** numbers**,** object**);**

**return** **(**Vector**<**FSTicket**>)** object**[**0**];**

**}**

Позволяет читать квитанции по номерам документов ФН.

Возвращает массив объектов FSTicket.

1. Чтение квитанций ФН начиная с определенного номера документа.

public static final int SMFPTR\_DIO\_READ\_FS\_TICKETS3 **=** 0x41**;**

public String**[]** fsReadTickets3**(**int number1**)** **throws** Exception **{**

Object**[]** object **=** **new** Object**[**1**];**

int**[]** data **=** **new** int**[**1**];**

data**[**0**]** **=** number1**;**

directIO**(**SmFptrConst**.**SMFPTR\_DIO\_READ\_FS\_TICKETS3**,** data**,** object**);**

**return** **(**String**[])** object**[**0**];**

**}**

public String**[]** fsReadTickets3**(**int number1**,** int count**)** **throws** Exception **{**

Object**[]** object **=** **new** Object**[**1**];**

int**[]** data **=** **new** int**[**2**];**

data**[**0**]** **=** number1**;**

data**[**1**]** **=** count**;**

directIO**(**SmFptrConst**.**SMFPTR\_DIO\_READ\_FS\_TICKETS3**,** data**,** object**);**

**return** **(**String**[])** object**[**0**];**

**}**

Позволяет читать квитанции от номера документа number1 до последней квитанции или заданное количество квитанций. Возвращает список строк, где каждая строка соответствует тексту квитанции. В тексте квитанции передается через точку с запятой дата, время, ФПД, номер ФД.

1. Чтение квитанций ФН начиная с определенного номера документа.

public static final int SMFPTR\_DIO\_READ\_FS\_TICKETS4 **=** 0x42**;**

public Vector**<**FSTicket**>** fsReadTickets4**(**int number1**)** **throws** Exception **{**

int**[]** data **=** **new** int**[**1**];**

data**[**0**]** **=** number1**;**

Object**[]** object **=** **new** Object**[**1**];**

object**[**0**]** **=** **null;**

directIO**(**SmFptrConst**.**SMFPTR\_DIO\_READ\_FS\_TICKETS4**,** data**,** object**);**

**return** **(**Vector**<**FSTicket**>)** object**[**0**];**

**}**

public Vector**<**FSTicket**>** fsReadTickets4**(**int number1**,** int count**)** **throws** Exception **{**

int**[]** data **=** **new** int**[**2**];**

data**[**0**]** **=** number1**;**

data**[**1**]** **=** count**;**

Object**[]** object **=** **new** Object**[**1**];**

object**[**0**]** **=** **null;**

directIO**(**SmFptrConst**.**SMFPTR\_DIO\_READ\_FS\_TICKETS4**,** data**,** object**);**

**return** **(**Vector**<**FSTicket**>)** object**[**0**];**

**}**

Позволяет читать квитанции от номера документа number1 до последней квитанции или заданное количество квитанций.  
Возвращает массив объектов FSTicket.

1. Печать чека коррекции.

public static final int SMFPTR\_DIO\_PRINT\_CORRECTION2 **=** 0x43**;**

public void fsPrintCorrectionReceipt3**(**

int correctionType**,**

int paymentType**,**

long total**,**

long payments0**,**

long payments1**,**

long payments2**,**

long payments3**,**

long payments4**,**

long taxTotals0**,**

long taxTotals1**,**

long taxTotals2**,**

long taxTotals3**,**

long taxTotals4**,**

long taxTotals5**,**

int taxSystem**,**

Object**[]** outParams**)** **throws** JposException **{**

Object**[]** params **=** **new** Object**[**18**];**

params**[**0**]** **=** correctionType**;**

params**[**1**]** **=** paymentType**;**

params**[**2**]** **=** total**;**

params**[**3**]** **=** payments0**;**

params**[**4**]** **=** payments1**;**

params**[**5**]** **=** payments2**;**

params**[**6**]** **=** payments3**;**

params**[**7**]** **=** payments4**;**

params**[**8**]** **=** taxTotals0**;**

params**[**9**]** **=** taxTotals1**;**

params**[**10**]** **=** taxTotals2**;**

params**[**11**]** **=** taxTotals3**;**

params**[**12**]** **=** taxTotals4**;**

params**[**13**]** **=** taxTotals5**;**

params**[**14**]** **=** taxSystem**;**

directIO**(**SmFptrConst**.**SMFPTR\_DIO\_PRINT\_CORRECTION2**,** **null,** params**);**

int receiptNumber **=** **(**Integer**)** params**[**15**];**

int documentNumber **=** **(**Integer**)** params**[**16**];**

long documentDigest **=** **(**Long**)** params**[**17**];**

outParams**[**0**]** **=** receiptNumber**;**

outParams**[**1**]** **=** documentNumber**;**

outParams**[**2**]** **=** documentDigest**;**

**}**

Выполняет команду «Сформировать чек коррекции V2 FF4AH».

1. Чтение необнуляемых счетчиков.

public static final int SMFPTR\_DIO\_READ\_TOTALS **=** 0x44**;**

public void readTotals**(**long**[]** totals**)** **throws** Exception **{**

int**[]** data **=** **new** int**[**1**];**

data**[**0**]** **=** SmFptrConst**.**FMTOTALS\_ALL\_FISCALIZATIONS**;**

Long**[]** object **=** **new** Long**[**4**];**

directIO**(**SmFptrConst**.**SMFPTR\_DIO\_READ\_TOTALS**,** data**,** object**);**

totals**[**0**]** **=** **(**Long**)** object**[**0**];**

totals**[**1**]** **=** **(**Long**)** object**[**1**];**

totals**[**2**]** **=** **(**Long**)** object**[**2**];**

totals**[**3**]** **=** **(**Long**)** object**[**3**];**

**}**

Если ФР поддерживает ФН, то счетчики читаются при помощи сервисной команды. Если ФР с ФП, то читаются счетчики ФП.

1. Печать графики.

public static final int SMFPTR\_DIO\_PRINT\_RAW\_GRAPHICS **=** 0x46**;**

public void printRawGraphics**(**byte**[][]** data**)** **throws** Exception **{**

directIO**(**SmFptrConst**.**SMFPTR\_DIO\_PRINT\_RAW\_GRAPHICS**,** **new** int**[**0**],** data**);**

**}**

1. Чтение ширины печати принтера в точках.

public static final int SMFPTR\_DIO\_READ\_MAX\_GRAPHICS\_WIDTH **=** 0x47**;**

public int readMaxGraphicsWidth**()** **throws** JposException **{**

int**[]** data **=** **new** int**[**1**];**

directIO**(**SmFptrConst**.**SMFPTR\_DIO\_READ\_MAX\_GRAPHICS\_WIDTH**,** data**,** **null);**

**return** data**[**0**];**

**}**

1. Печать чека коррекции.

public static final int SMFPTR\_DIO\_PRINT\_CORRECTION2 **=** 0x43**;**

public void fsPrintCorrectionReceipt3**(**

int correctionType**,**

int paymentType**,**

long total**,**

long payments0**,**

long payments1**,**

long payments2**,**

long payments3**,**

long payments4**,**

long taxTotals0**,**

long taxTotals1**,**

long taxTotals2**,**

long taxTotals3**,**

long taxTotals4**,**

long taxTotals5**,**

int taxSystem**,**

Object**[]** outParams**)** **throws** JposException **{**

Object**[]** params **=** **new** Object**[**18**];**

params**[**0**]** **=** correctionType**;**

params**[**1**]** **=** paymentType**;**

params**[**2**]** **=** total**;**

params**[**3**]** **=** payments0**;**

params**[**4**]** **=** payments1**;**

params**[**5**]** **=** payments2**;**

params**[**6**]** **=** payments3**;**

params**[**7**]** **=** payments4**;**

params**[**8**]** **=** taxTotals0**;**

params**[**9**]** **=** taxTotals1**;**

params**[**10**]** **=** taxTotals2**;**

params**[**11**]** **=** taxTotals3**;**

params**[**12**]** **=** taxTotals4**;**

params**[**13**]** **=** taxTotals5**;**

params**[**14**]** **=** taxSystem**;**

directIO**(**SmFptrConst**.**SMFPTR\_DIO\_PRINT\_CORRECTION2**,** **null,** params**);**

int receiptNumber **=** **(**Integer**)** params**[**15**];**

int documentNumber **=** **(**Integer**)** params**[**16**];**

long documentDigest **=** **(**Long**)** params**[**17**];**

outParams**[**0**]** **=** receiptNumber**;**

outParams**[**1**]** **=** documentNumber**;**

outParams**[**2**]** **=** documentDigest**;**

**}**

public void fsPrintCorrectionReceipt3**(**

int correctionType**,**

int paymentType**,**

long total**,**

long payments0**,**

long payments1**,**

long payments2**,**

long payments3**,**

long payments4**,**

long taxTotals0**,**

long taxTotals1**,**

long taxTotals2**,**

long taxTotals3**,**

long taxTotals4**,**

long taxTotals5**,**

int taxSystem**)** **throws** JposException **{**

Object**[]** params **=** **new** Object**[**15**];**

params**[**0**]** **=** correctionType**;**

params**[**1**]** **=** paymentType**;**

params**[**2**]** **=** total**;**

params**[**3**]** **=** payments0**;**

params**[**4**]** **=** payments1**;**

params**[**5**]** **=** payments2**;**

params**[**6**]** **=** payments3**;**

params**[**7**]** **=** payments4**;**

params**[**8**]** **=** taxTotals0**;**

params**[**9**]** **=** taxTotals1**;**

params**[**10**]** **=** taxTotals2**;**

params**[**11**]** **=** taxTotals3**;**

params**[**12**]** **=** taxTotals4**;**

params**[**13**]** **=** taxTotals5**;**

params**[**14**]** **=** taxSystem**;**

directIO**(**SmFptrConst**.**SMFPTR\_DIO\_PRINT\_CORRECTION2**,** **null,** params**);**

**}**

1. Запрос необнуляемых сумм из ФП или ФН.

public static final int SMFPTR\_DIO\_READ\_TOTALS **=** 0x44**;**

public void readTotals**(**long**[]** totals**)** **throws** Exception **{**

int**[]** data **=** **new** int**[**1**];**

data**[**0**]** **=** SmFptrConst**.**FMTOTALS\_ALL\_FISCALIZATIONS**;**

Long**[]** object **=** **new** Long**[**4**];**

directIO**(**SmFptrConst**.**SMFPTR\_DIO\_READ\_TOTALS**,** data**,** object**);**

totals**[**0**]** **=** **(**Long**)** object**[**0**];**

totals**[**1**]** **=** **(**Long**)** object**[**1**];**

totals**[**2**]** **=** **(**Long**)** object**[**2**];**

totals**[**3**]** **=** **(**Long**)** object**[**3**];**

**}**

Для ФН возвращается 4 числа - приход, расход, возврат прихода, возврат расхода.  
Для ФП возвращается 4 числа - сумма продаж, сумма покупок, сумма возвратов продаж, сумма возвратов покупок.

1. Печать графики из массива.

public static final int SMFPTR\_DIO\_PRINT\_RAW\_GRAPHICS **=** 0x46**;**

public void printRawGraphics**(**byte**[][]** data**)** **throws** Exception **{**

directIO**(**SmFptrConst**.**SMFPTR\_DIO\_PRINT\_RAW\_GRAPHICS**,** **new** int**[**0**],** data**);**

**}**

1. Чтение максимальной ширины печати.

public static final int SMFPTR\_DIO\_READ\_MAX\_GRAPHICS\_WIDTH **=** 0x47**;**

public int readMaxGraphicsWidth**()** **throws** JposException **{**

int**[]** data **=** **new** int**[**1**];**

directIO**(**SmFptrConst**.**SMFPTR\_DIO\_READ\_MAX\_GRAPHICS\_WIDTH**,** data**,** **null);**

**return** data**[**0**];**

**}**

1. Печать чека коррекции.

public static final int SMFPTR\_DIO\_PRINT\_CORRECTION **=** 0x48**;**

public void fsPrintCorrectionReceipt**(**int operation**,** long amount**)** **throws** JposException **{**

Object**[]** params **=** **new** Object**[**2**];**

params**[**0**]** **=** **new** Integer**(**operation**);**

params**[**1**]** **=** **new** Long**(**amount**);**

directIO**(**SmFptrConst**.**SMFPTR\_DIO\_PRINT\_CORRECTION**,** **null,** params**);**

**}**

1. Промотка бумаги на заданное количество строк.

public static final int SMFPTR\_DIO\_FEED\_PAPER **=** 0x49**;**

public void feedPaper**(**int linesNumber**)** **throws** Exception **{**

directIO**(**SmFptrConst**.**SMFPTR\_DIO\_FEED\_PAPER**,** **new** int**[]{**linesNumber**},** **null);**

**}**

1. Проверка состояния потока отправки данных из ФН в ОФД.

Поток нужен для отправки данных в ОФД для ФР ШТРИХ-МОБАЙЛ.

public static final int SMFPTR\_DIO\_GET\_FS\_SERVICE\_STATE **=** 0x4A**;**

public boolean isFSServiceStarted**()** **throws** Exception **{**

Boolean**[]** object **=** **new** Boolean**[**1**];**

directIO**(**SmFptrConst**.**SMFPTR\_DIO\_GET\_FS\_SERVICE\_STATE**,** **null,** object**);**

**return** object**[**0**];**

**}**

1. Установка состояния потока отправки данных из ФН в ОФД, запустить или остановить.

public static final int SMFPTR\_DIO\_SET\_FS\_SERVICE\_STATE **=** 0x4B**;**

public void startFSService**()** **throws** Exception **{**

int**[]** data **=** **new** int**[]{**1**};**

directIO**(**SmFptrConst**.**SMFPTR\_DIO\_SET\_FS\_SERVICE\_STATE**,** data**,** **null);**

**}**

public void stopFSService**()** **throws** Exception **{**

int**[]** data **=** **new** int**[]{**0**};**

directIO**(**SmFptrConst**.**SMFPTR\_DIO\_SET\_FS\_SERVICE\_STATE**,** data**,** **null);**

**}**

1. Передача команды продолжения печати.

public static final int SMFPTR\_DIO\_CONTINUE\_PRINT **=** 0x4C**;**

public void continuePrint**()** **throws** Exception **{**

directIO**(**SmFptrConst**.**SMFPTR\_DIO\_CONTINUE\_PRINT**,** **new** int**[**0**],** **null);**

**}**

1. Чтение документа ФН из ФР.

public static final int SMFPTR\_DIO\_FS\_READ\_DOCUMENT\_TLV **=** 0x4E**;**

public DocumentTLV fsReadDocumentTLV**(**int documentNumber**)** **throws** Exception **{**

Object**[]** outParams **=** **new** Object**[**2**];**

directIO**(**SmFptrConst**.**SMFPTR\_DIO\_FS\_READ\_DOCUMENT\_TLV**,** **new** int**[]{**documentNumber**},** outParams**);**

int documentType **=** **(**Integer**)** outParams**[**0**];**

byte**[]** tlv **=** **(**byte**[])** outParams**[**1**];**

**return** **new** DocumentTLV**(**documentNumber**,** documentType**,** tlv**);**

**}**

public Vector**<**String**>** fsReadDocumentTLVText**(**int documentNumber**)** **throws** Exception **{**

Object**[]** outParams **=** **new** Object**[**1**];**

directIO**(**SmFptrConst**.**SMFPTR\_DIO\_FS\_READ\_DOCUMENT\_TLV\_TEXT**,** **new** int**[]{**documentNumber**},** outParams**);**

Vector**<**String**>** document **=** **(**Vector**<**String**>)** outParams**[**0**];**

**return** document**;**

**}**

1. Чтение документа об открытии смены.

public static final int SMFPTR\_DIO\_FS\_READ\_DAY\_OPEN **=** 0x4F**;**

public Vector**<**String**>** fsReadDayOpen**()** **throws** Exception **{**

Vector**<**String**>** lines **=** **new** Vector**<**String**>();**

directIO**(**SmFptrConst**.**SMFPTR\_DIO\_FS\_READ\_DAY\_OPEN**,** **null,** lines**);**

**return** lines**;**

**}**

1. Чтение документа о закрытии смены.

public static final int SMFPTR\_DIO\_FS\_READ\_DAY\_CLOSE **=** 0x50**;**

public Vector**<**String**>** fsReadDayClose**()** **throws** Exception **{**

Vector**<**String**>** lines **=** **new** Vector**<**String**>();**

directIO**(**SmFptrConst**.**SMFPTR\_DIO\_FS\_READ\_DAY\_CLOSE**,** **null,** lines**);**

**return** lines**;**

**}**

1. Чтение последнего чека.

public static final int SMFPTR\_DIO\_FS\_READ\_RECEIPT **=** 0x51**;**

public Vector**<**String**>** fsReadReceipt**()** **throws** Exception **{**

Vector**<**String**>** lines **=** **new** Vector**<**String**>();**

directIO**(**SmFptrConst**.**SMFPTR\_DIO\_FS\_READ\_RECEIPT**,** **null,** lines**);**

**return** lines**;**

**}**

1. Чтение состояния ФН.

public static final int SMFPTR\_DIO\_FS\_READ\_STATUS **=** 0x52**;**

Возвращается массив строк:

- Состояние фазы жизни

- Текущий документ

- Данные документа получены

- Смена в ФН открыта

- Флаги предупреждения

- Дата

- Время

- Номер ФН

- Номер последнего ФД

1. Запрос документа ФН.

\*/

public static final int SMFPTR\_DIO\_FS\_FIND\_DOCUMENT **=** 0x53**;**

Возвращается массив строк:

- Тип документа

- Получена квитанция

- Дата и время

- Номер документа

- Подпись документа

1. Запрет печати завершения документа.

Это нужно для печати нескольких документов без отрезки.

public static final int SMFPTR\_DIO\_FS\_DISABLE\_DOCEND **=** 0x54**;**

1. Запрос параметра открытия ФН.

public static final int SMFPTR\_DIO\_FS\_READ\_FISCALIZATION\_TAG **=** 0x55**;**

public byte**[]** readFiscalizationTag**(**int fiscalizationNumber**,** int tagNumber**)** **throws** Exception **{**

Object**[]** outParams **=** **new** Object**[**1**];**

directIO**(**SmFptrConst**.**SMFPTR\_DIO\_FS\_READ\_FISCALIZATION\_TAG**,** **new** int**[]{**fiscalizationNumber**,** tagNumber**},** outParams**);**

**return** **(**byte**[])** outParams**[**0**];**

**}**

1. Запрос документа открытия ФН.

public static final int SMFPTR\_DIO\_FS\_READ\_FISCALIZATION\_TLV **=** 0x56**;**

public byte**[]** readFiscalizationTLV**(**int fiscalizationNumber**)** **throws** Exception **{**

Object**[]** outParams **=** **new** Object**[**1**];**

directIO**(**SmFptrConst**.**SMFPTR\_DIO\_FS\_READ\_FISCALIZATION\_TLV**,** **new** int**[]{**fiscalizationNumber**},** outParams**);**

**return** **(**byte**[])** outParams**[**0**];**

**}**

1. Запрос параметров устройтва.

public static final int SMFPTR\_DIO\_READ\_DEVICE\_METRICS **=** 0x57**;**

public DeviceMetrics readDeviceMetrics**()** **throws** JposException **{**

Object**[]** object **=** **new** Object**[**1**];**

directIO**(**SmFptrConst**.**SMFPTR\_DIO\_READ\_DEVICE\_METRICS**,** **null,** object**);**

**return** **(**DeviceMetrics**)** object**[**0**];**

**}**

1. Чтение счетчиков по типу чека.

Возможные значения для типа чека: 1..4.

Метод передает в ФР команды:

FE F4 01 00 00 00 - возвращает 16 8-ми байтовых счетчиков (по типам оплаты 1-16) для приходов.

FE F4 02 00 00 00 - возвращает 16 8-ми байтовых счетчиков (по типам оплаты 1-16) для возвратов приходов.

FE F4 03 00 00 00 - возвращает 16 8-ми байтовых счетчиков (по типам оплаты 1-16) для расходов.

FE F4 04 00 00 00 - возвращает 16 8-ми байтовых счетчиков (по типам оплаты 1-16) для возвратов расходов.

Чтобы получить общую сумму для новых счетчиков надо просуммировать все 16 чисел.

public static final int SMFPTR\_DIO\_READ\_TOTALIZER **=** 0x58**;**

public long**[]** readTotalizers**(**int recType**)** **throws** JposException **{**

int**[]** data **=** **new** int**[**1**];**

Object**[]** object **=** **new** Object**[**1**];**

data**[**0**]** **=** recType**;**

directIO**(**SmFptrConst**.**SMFPTR\_DIO\_READ\_TOTALIZER**,** data**,** object**);**

**return** **(**long**[])** object**[**0**];**

**}**

1. Чтение счетчиков по типу чека.

Чтение документа из ФН по номеру в виде текста.

public static final int SMFPTR\_DIO\_FS\_READ\_DOCUMENT\_TLV\_TEXT **=** 0x59**;**

public Vector**<**String**>** fsReadDocumentTLVText**(**int documentNumber**)** **throws** Exception **{**

Object**[]** outParams **=** **new** Object**[**1**];**

directIO**(**SmFptrConst**.**SMFPTR\_DIO\_FS\_READ\_DOCUMENT\_TLV\_TEXT**,** **new** int**[]{**documentNumber**},** outParams**);**

Vector**<**String**>** document **=** **(**Vector**<**String**>)** outParams**[**0**];**

**return** document**;**

**}**

1. Запись TLV данных, привязанных к операции.

public static final int SMFPTR\_DIO\_FS\_WRITE\_OPERATION\_TLV **=** 0x5A**;**

public void fsWriteOperationTLV**(**byte**[]** data**)** **throws** JposException **{**

directIO**(**SmFptrConst**.**SMFPTR\_DIO\_FS\_WRITE\_OPERATION\_TLV**,** **null,** data**);**

**}**

1. Чтение документа ЭКЛЗ по номеру.

public static final int SMFPTR\_DIO\_READ\_EJ\_DOCUMENT **=** 0x5B**;**

public String**[]** readEJDocument**(**int documentNumber**)** **throws** JposException **{**

int**[]** data **=** **new** int**[**1**];**

Object**[]** object **=** **new** Object**[**1**];**

data**[**0**]** **=** documentNumber**;**

directIO**(**SmFptrConst**.**SMFPTR\_DIO\_READ\_EJ\_DOCUMENT**,** data**,** object**);**

**return** **(**String**[])** object**[**0**];**

**}**

1. Начать закрытие смены.

public static final int SMFPTR\_DIO\_START\_DAY\_CLOSE **=** 0x5C**;**

1. Начать открытие смены.

public static final int SMFPTR\_DIO\_START\_DAY\_OPEN **=** 0x5D**;**

1. Начать фискализацию.

public static final int SMFPTR\_DIO\_START\_FISCALIZATION **=** 0x5E**;**

1. Начать отчет о состоянии расчетов.

public static final int SMFPTR\_DIO\_START\_CALC\_REPORT **=** 0x5F**;**

1. Начать закрытие фискального режима.

public static final int SMFPTR\_DIO\_START\_FISCAL\_CLOSE **=** 0x60**;**

1. Передача кода товара для привязки к позиции чека.

public static final int SMFPTR\_DIO\_SEND\_ITEM\_CODE **=** 0x61**;**

public void sendItemCode**(**String barcode**)** **throws** JposException **{**

directIO**(**SmFptrConst**.**SMFPTR\_DIO\_SEND\_ITEM\_CODE**,** **null,** **new** String**[]{**barcode**});**

**}**

1. Проверка кода товара средствами ФН. Для проверки кода товара должна быть специальная версия ФН.

public static final int SMFPTR\_DIO\_CHECK\_ITEM\_CODE **=** 0x62**;**

public void checkItemCode**(**String barcode**)** **throws** JposException **{**

directIO**(**SmFptrConst**.**SMFPTR\_DIO\_CHECK\_ITEM\_CODE**,** **null,** **new** String**[]{**barcode**});**

**}**

1. Принять или отвергнуть код товара.

public static final int SMFPTR\_DIO\_ACCEPT\_ITEM\_CODE **=** 0x63**;**

public void acceptItemCode**(**boolean accept**)** **throws** JposException **{**

String action **=** "0"**;**

**if** **(**accept**)** action **=** "1"**;**

directIO**(**SmFptrConst**.**SMFPTR\_DIO\_ACCEPT\_ITEM\_CODE**,** **null,** **new** String**[]{**action**});**

**}**

1. Привязать код товара к позиции чека.

public static final int SMFPTR\_DIO\_BIND\_ITEM\_CODE **=** 0x64**;**

public void bindItemCode**(**String barcode**)** **throws** JposException **{**

String**[]** params **=** **new** String**[**6**];**

params**[**0**]** **=** barcode**;**

directIO**(**SmFptrConst**.**SMFPTR\_DIO\_BIND\_ITEM\_CODE**,** **null,** params**);**

**}**

1. Получить статус информационного обмена с АС «Серверы СКЗКМ».

public static final int SMFPTR\_DIO\_READ\_KM\_SERVER\_STATUS **=** 0x65**;**

Метод передает команду «Получить статус информационного обмена с АС «Серверы СКЗКМ» FF68H»

**Код команды FF68h**. Длина сообщения: 6 байт.

Пароль оператора: 4 байта

**Ответ: FF68h** Длина сообщения: 18 байт.

Код ошибки: 1 байт

Состояние транспортного соединения : 1 байт

Статус чтения сообщения для ИСМ : 1 байт

Количество сообщений «Отчет об изменении статуса» в очереди : 2 байта

Номер сообщения для ИСМ : 4 байта

Дата-время документа для ИСМ : 5 байт

Размер свободной области для хранения «Отчет об изменении статуса» в килобайтах : 4 байта

Возвращается массив строк:

- Состояние транспортного соединения

- Количество сообщений «Отчет об изменении статуса» в очереди

- Номер сообщения для ИСМ

- Дата документа для ИСМ

- Время документа для ИСМ

- Размер свободной памяти в килобайтах

1. Чтение версии ФФД ФН.

public static final int SMFPTR\_DIO\_READ\_FFD\_VERSION **=** 0x66**;**

public int readFFDVersion**()** **throws** JposException **{**

int**[]** data **=** **new** int**[**1**];**

directIO**(**SmFptrConst**.**SMFPTR\_DIO\_READ\_FFD\_VERSION**,** data**,** **null);**

**return** data**[**0**];**

**}**

1. Запись версии ФФД ФН.

public static final int SMFPTR\_DIO\_WRITE\_FFD\_VERSION **=** 0x67**;**

public void writeFFDVersion**(**int version**)** **throws** JposException **{**

int**[]** data **=** **new** int**[]** **{** version **};**

directIO**(**SmFptrConst**.**SMFPTR\_DIO\_WRITE\_FFD\_VERSION**,** data**,** **null);**

**}**

1. Установка кода товара для передачи тега маркировки 1162.

public static final int SMFPTR\_DIO\_SET\_ITEM\_CODE **=** 0x78**;**

public void setItemCode**(**String GTIN**,** String serial**)** **throws** JposException **{**

directIO**(**SmFptrConst**.**SMFPTR\_DIO\_SET\_ITEM\_CODE**,** **null,** **new** String**[]{**GTIN**,** serial**});**

**}**

public void setItemCode**(**String barcode**)** **throws** JposException **{**

setItemCode**(**barcode**,** ""**);**

**}**

## Работа с фискальным накопителем

Для работы с фискальным накопителем можно использовать следующие методы:

### Запрос статуса ФН FF01H

public void fsReadStatus**(**FSReadStatus command**)** **throws** JposException **{**

executeCommand**(**command**);**

**}**

### Запрос номера ФН FF02H

public void fsReadSerial**(**FSReadSerial command**)** **throws** JposException **{**

executeCommand**(**command**);**

**}**

### Чтение срока действия ФН:

public void fsReadExpDate**(**FSReadExpDate command**)** **throws** JposException **{**

executeCommand**(**command**);**

**}**

### Запрос версии ФН FF04H

public void fsReadVersion**(**FSReadVersion command**)** **throws** JposException **{**

executeCommand**(**command**);**

**}**

### Сформировать отчёт о регистрации ККТ FF06H

public void fsFiscalization**(**FSFiscalization command**)** **throws** JposException **{**

executeCommand**(**command**);**

**}**

### Сброс состояния ФН FF07H

public void fsResetState**(**FSResetState command**)** **throws** JposException **{**

executeCommand**(**command**);**

**}**

### Отменить документ в ФН FF08H

public void fsCancelDoc**(**FSCancelDoc command**)** **throws** JposException **{**

executeCommand**(**command**);**

**}**

### Запрос итогов фискализации FF09H:

public void fsReadFiscalization**(**FSReadFiscalization command**)** **throws** JposException **{**

executeCommand**(**command**);**

**}**

### Найти фискальный документ по номеру FF0AH

public void fsFindDocument**(**FSFindDocument command**)** **throws** JposException **{**

executeCommand**(**command**);**

**}**

### Открыть смену в ФН FF0BH

public void fsOpenDay**(**FSOpenDay command**)** **throws** JposException **{**

executeCommand**(**command**);**

**}**

### Передать произвольную TLV структуру FF0CH

public void fsWriteTLV**(**FSWriteTLV command**)** **throws** JposException **{**

executeCommand**(**command**);**

**}**

### Операция со скидками и надбавками FF0DH

public void fsSale**(**FSSale command**)** **throws** JposException **{**

executeCommand**(**command**);**

**}**

### Сформировать отчёт о перерегистрации ККТ FF34H

public void fsRegistrationReport**(**FSRegistrationReport command**)** **throws** JposException **{**

executeCommand**(**command**);**

**}**

### Начать формирование чека коррекции FF35H

public void fsStartCorrectionReceipt**(**FSStartCorrectionReceipt command**)** **throws** JposException **{**

executeCommand**(**command**);**

**}**

### Сформировать чек коррекции FF36H

public void fsPrintCorrectionReceipt**(**FSPrintCorrectionReceipt command**)** **throws** JposException **{**

executeCommand**(**command**);**

**}**

### Начать формирование отчёта о состоянии расчётов FF37H

public void fsStartCalcReport**(**FSStartCalcReport command**)** **throws** JposException **{**

executeCommand**(**command**);**

**}**

### Сформировать отчёт о состоянии расчётов FF38H

public void fsPrintCalcReport**(**FSPrintCalcReport command**)** **throws** JposException **{**

executeCommand**(**command**);**

**}**

### Получить статус информационного обмена FF39H

public void fsReadCommStatus**(**FSReadCommStatus command**)** **throws** JposException **{**

executeCommand**(**command**);**

**}**

### Запрос квитанции о получении данных в ОФД по номеру документа FF3CH

public void fsReadDocTicket**(**FSReadDocTicket command**)** **throws** JposException **{**

executeCommand**(**command**);**

**}**

### Начать закрытие фискального режима FF3DH

public void fsStartFiscalClose**(**FSStartFiscalClose command**)** **throws** JposException **{**

executeCommand**(**command**);**

**}**

### Закрыть фискальный режим FF3EH

public void fsPrintFiscalClose**(**FSPrintFiscalClose command**)** **throws** JposException **{**

executeCommand**(**command**);**

**}**

### Запрос количества ФД на которые нет квитанции FF3FH

public void fsReadDocCount**(**FSReadDocCount command**)** **throws** JposException **{**

executeCommand**(**command**);**

**}**

### Запрос параметров текущей смены FF40H

public void fsReadDayParameters**(**FSReadDayParameters command**)** **throws** JposException **{**

executeCommand**(**command**);**

**}**

### Начать открытие смены FF41H

public void fsStartDayOpen**(**FSStartDayOpen command**)** **throws** JposException **{**

executeCommand**(**command**);**

**}**

### Начать закрытие смены FF42H

public void fsStartDayClose**(**FSStartDayClose command**)** **throws** JposException **{**

executeCommand**(**command**);**

**}**

### Закрыть смену в ФН FF43H

public void fsDayClose**(**FSDayClose command**)** **throws** JposException **{**

executeCommand**(**command**);**

**}**

### Операция со скидками, надбавками и налогом FF44H

public void fsSale2**(**FSSale2 command**)** **throws** JposException **{**

executeCommand**(**command**);**

**}**

### Закрытие чека расширенное вариант №2 FF45H

public void fsCloseReceipt**(**FSCloseReceipt command**)** **throws** JposException **{**

executeCommand**(**command**);**

**}**