

# ФИСКАЛЬНОЕ ЯДРО 2.1

## Руководство разработчика

### Отличия от версии 1.xx

#### По структуре документов

- Атрибуты «Место расчетов» и «Адрес расчетов» перенесены в объект Location
- Атрибут «Кассир» перенесен в объект Signature
- Данные о ККМ у фискализированного документа перенесены в объект Signer принадлежащий объекту Signature
- Добавлена единица измерения в класс «Предмет расчета»
- Информация о текущей смене доступна из объекта Shift
- В объекты «Чек» и «Предмет расчета» добавлен класс AgentData содержащий данные об агенте

#### По методам

- В методы добавились шаблоны документов (см. ниже)
- Метод «открыть смену» и «закрыть смену» объединены в один «переключить состояние смены»
- Получение данных о состоянии обмена с ОФД вынесено в отдельный метод
- Добавлен метод печати произвольного документа

### Обновления от 04.09.2019

- Добавлены методы работы с остатками наличности
- Добавлен метод повторной печати/получения в виде изображения произвольного фискального документа по номеру

### Обновления от 16.05.2020

- Операции с деньгами переведены в тип BigDecimal, чтобы избежать ошибок округления. Так как количество товара также связано с деньгами математически, то и оно также переведено в BigDecimal.

## Принцип работы Фискального ядра

Фискальное ядро функционирует как сервис системы, к которой приложения могут подключаться посредством IPC.

Для связи с Фискальным ядром предназначен интент с ACTION **rs.fncore2.FiscalStorage** и с именем пакета **rs.fncore2**.

После подключения к сервису следует создать прокси-объект класса **FiscalStorage** и с его помощью взаимодействовать с Фискальным ядром.

Кроме этого доступны два document provider, для доступа к сменным остаткам и к архиву документов.

Класс FiscalStorage реализует следующие методы:

**int init();** - инициализация фискального накопителя (при необходимости). Должен быть вызван перед началом работы. Возвращает код ошибки-состояния ФН.

**int readKKMInfo(KKMInfo info);** - получить данные о состоянии ФН и параметры регистрации ККМ

**int resetFN();** - произвести сброс фискального накопителя (для МГМ). Так же удаляет все сменные остатки и документы в архиве

**void cancelDocument();** - отменить незавершенный документ

**int doFiscalization(int reason, in OU operator, in KKMInfo info, out KKMInfo signed, String template)** — выполнить регистрацию/изменение параметров ККМ

**int toggleShift(in OU operator, out Shift shift, String template);** - изменить состояние смены

**int updateOFDStatistic(out OFDStatistic statistic);** - получить данные о состоянии обмена с ОФД

**int requestFiscalReport(in OU operator, out FiscalReport report, String template);** - сформировать отчет о состоянии расчетов

**int doSellOrder(in SellOrder order, in OU operator, out SellOrder signed, boolean doPrint, String header, String item, String footer, String footerEx);** - сформировать чек продаж/возврата

**int doArchive(in OU operator, out ArchiveReport report, String template);** - выполнить перевод ФН в постфискальный режим

**int doCorrection(in Correction correction, in OU operator, out Correction signed, String template);** - сформировать чек коррекции

**int doXReport(out XReport report, boolean doPrint, String template);** - сформировать отчет о сменных остатках

**OFDSettings getOFDSettings();** - получить настройки сервера ОФД

**void setOFDSettings(in OFDSettings settings);** - сохранить настройки сервера ОФД

**PrintSettings getPrintSettings();** - получить настройки параметров печати

**void setPrintSettings(in PrintSettings settings);** - установить настройки параметров печати

**void doPrint(String text);** - напечатать произвольный документ

**void pushDocuments();** - немедленно начать отправку документов в ОФД

Методы для прямой работы с последовательным портом

**int openTransaction();** - открыть соединение

**int writeB(int transaction, in byte [] data, int offset, int size);** - отправить байты в последовательный порт

**int readB(int transaction, out byte [] data, int offset, int size);** - прочитать данные из последовательного порта

**void closeTransaction(int transaction);** - закрыть ранее открытое соединение.

## Структура объектов для работы с Фискальным ядром

Для работы с фискальным ядром используются TLV-based объекты, которые являются наследником от класса TLV. Все эти объекты содержат экземпляры класса Tag, который содержит данные полей.

В любом методе обычно используются два экземпляра одного класса — первый содержит данные, заполняемые при вызове метода и передается в Фискальное ядро, второй экземпляр возвращается Фискальным ядром и содержит документ с заполненными полями фискальных данных и/или информацией, возвращаемой ядром.

Таким образом, дополнительные теги могут быть добавлены разработчиком напрямую в документ. Однако, некоторые теги могут быть перекрыты полями документа, о чем будет написано отдельно.

## Регистрация/изменение параметров ККМ

Для регистрации или изменения параметров регистрации ККМ используется метод **doFiscalization(int reason, in OU operator, in KKMInfo info, out KKMInfo signed, String template)**. Метод принимает следующие параметры

**reason** — причина изменения данных. Константа REASON\_xxx из класса KKMInfo (описание констант смотри в javadoc).

**info** — экземпляр класса KKMInfo с заполненными данными. В этом классе следующие поля перекрывают теги, которые может определить пользователь:

getOwner().setName — заменяет тег 1048 если значение не пустое

ofd().setName(), ofd().setINN() - заменяют теги 1046 и 1017 соответственно

setFFDProtocolVersion() - заменяет тег 1209

**operator** — данные оператора (кассира). Заменяет теги 1021, 1203

**signed** — возвращаемый Фискальным ядром документ, содержащий Фискальный Номер и Фискальную Подпись документа, а так же информацию о ККМ выполнившей операцию.

**template** — шаблон для печати документа, или null если используется внутренний шаблон Фискального Ядра. Правила построения шаблонов описаны в соответствующем разделе.

Пример регистрации ККМ.

```
KKMInfo nfo = new KKMInfo();
nfo.setKKMNumber("0000000011032845");
nfo.getTaxModes().add(TaxMode.Common); // Общая система
налогообложения
nfo.setOfflineMode(false);
nfo.getOwner().setINN("7718776723");
nfo.getOwner().setName("000 «РАЙТСКАН»");
nfo.getLocation().setAddress("Москва, ул. Малая Семеновская 11/2
стр. 4");
nfo.getLocation().setPlace("Офис, 2ой этаж");
nfo.ofd().setINN("7704211201");
nfo.ofd().setName("Такском ОФД");
int r = _fs.doFiscalization(KKMInfo.REASON_REPLACE_FN, signer,
nfo,info,null);
OU signer = new OU(«Администратор»);
Toast.makeText(this,String.format("Result %02X", r),
Toast.LENGTH_SHORT).show();
```

Данную процедуру рекомендуется выполнять в потоке, хотя она и не занимает много времени.

## Получение/установка параметров сервера ОФД.

Данная операция является нефискальной и может быть выполнена в любое время. Она реализуется с помощью методов **getOFDSettings()** и **setOFDSettings()**. Использование этих методов имеет смысл только если не выбран автономный режим работы ФН. Этот метод может быть выполнен в UI потоке. Пример

```
OFDSettings s = _fs.getOFDSettings();
s.setServerAddress("fltest.taxcom.ru");
s.setServerPort(7778);
s.setServerTimeout(180);
s.setImmediatelyMode(true);
_fs.setOFDSettings(s);
```

## Настройка параметров печати

Вы можете управлять размером и гарнитурой шрифта по умолчанию и отступами от края слева и справа. Стоит помнить, что размер чека шириной 58 мм в точках составляет 384. Исходя из этой цифры стоит устанавливать размер полей и шрифта. Для управления печатью используются методы **getPrintSettings()/setPrintSettings()**. Их параметр описан в javadoc. Эти методы могут быть выполнены в UI потоке.

## Операции со сменой

Для операций со сменой предназначен метод **toggleShift()**. Этот метод работает со следующими параметрами

**operator** — описание оператора (кассира) выполняющего операцию

**shift** — возвращаемый Фискальным ядром экземпляр класса Shift содержащий описание открытой(закрытой) смены и данные фискализации

**template** — шаблон документа открытия/закрытия смены или null

Этот метод меняет состояние смены. Т.е. если смена была закрыта — открывает ее и наоборот. Метод рекомендуется выполнять в отдельном потоке. Так же, при закрытии смены, если в сетевых настройках ОФД стоит **setImmediatelyMode(false)** то начинается отправка данных в ОФД, если ККМ не находится в автономном режиме.

При открытии смены сменные остатки сбрасываются.

Метод рекомендуется выполнять в отдельном потоке.

## Перевод ФН в постфискальный режим

Для перевода ФН в постфискальный режим рабочая смена должна быть закрыта. Операция выполняется методом **doArchive()**. Метод использует следующие параметры

**operator** — описание оператора (кассира) выполняющего операцию

**report** — отчет о закрытии фискального накопителя содержащий данные фискализации. Возвращается Фискальным ядром

**template** — шаблон печати отчета или null

Метод рекомендуется выполнять в потоке. Сразу после успешного выполнения метода начинается передача данных в ОФД если ККМ не находится в автономном режиме.

## Формирование отчета о состоянии расчетов

Для формирования отчета о состоянии расчетов достаточно вызвать метод

**requestFiscalReport**. Этот метод использует следующие параметры

**operator** — описание оператора (кассира) выполняющего операцию

**report** — отчет о состоянии расчетов, содержащий данные фискализации. Возвращается Фискальным ядром

**template** — шаблон печати отчета или null

Метод рекомендуется вызывать в отдельном потоке.

## Отчет о сменных остатках.

Для формирования отчета о сменных остатках предназначен метод **doXReport()**. Этот метод не создает фискальный документ. Метод использует следующие параметры

**report** — документ, содержащий значение счетчиков остатков, заполняется Фискальным ядром.

**doPrint** — признак печати отчета. Если false то предыдущее поле заполняется, но отчет не выводится на принтер

**template** — шаблон отчета или null

Метод может быть вызван из UI-потока.

## Формирование чека продаж/возврата

Для формирования корректного чека продаж/возврата сначала требуется создать и заполнить экземпляр класса `SellOrder`. При создании требуется указать тип чека и используемую СНО.

```
SellOrder order = new SellOrder(OrderType.Income, TaxMode.Common);
```

После чего требуется добавить предметы расчета, используя метод `addItem()`. Следует учесть что вы можете добавить только один предмет расчета с типом оплаты «Оплата кредита» и он должен быть единственным. Для создания предмета расчета используется класс `SellItem`.

Обратите внимание, что следующие поля в `SellItem` заменяют добавленные разработчиком теги при формировании чека

`PaymentType` заданный в конструкторе заменяет тег 1214

`ItemType` заданный в конструкторе заменяет тег 1212

`Name` заменяет тег 1030

`MeasureName` заменяет тег 1197

`Price` заменяет тег 1079

`Qtty` заменяет тег 1023

Значение ставки НДС переданное в конструкторе заменяет тег 1199

При значениях ставки НДС отличных от 0 и «не облагается» значение ставки заменяет тег 1200

Сумма (перемножение `Price` на `Qtty`) заменяет тег 1043

Если установлены агентские данные (`setType()` у метода `getAgentData()` вызван с ненулевым параметром) то заменяются теги 1224 и 1223 на данные полученные из `AgentData`

Пример создания предметов расчета приведен ниже.

```
order.addItem(new SellItem("Какой-то товар", new BigDecimal(1.0),  
new BigDecimal(555.23), VAT.vat_20)); // Добавление "простого"  
предмета расчета
```

```
order.addItem(new  
SellItem(SellItemType.Service,ItemPaymentType.Full,"Какая-то  
услуга",new BigDecimal(1.0),"шт.",new  
BigDecimal(220.56),VAT.vat_10)); // Услуга
```

```
order.addItem(new  
SellItem(SellItemType.Reserved2,ItemPaymentType.Full,"10",new  
BigDecimal(1.00),"рубль",new BigDecimal(8000),VAT.vat_20)); //  
Тип, ПФ которого определяется ФФД
```

```
SellItem item = new  
SellItem(SellItemType.Good,ItemPaymentType.Full,"Комиссионный  
товар",new BigDecimal(1),"шт.",new  
BigDecimal(300),VAT.vat_none); // Товар с агентскими данными  
item.getAgentData().setType(AgentType.Commissionaire); // Заполнение  
данных об агенте
```

```
item.getAgentData().setProviderName("000 Рога и Копыта");  
item.getAgentData().setProviderPhone("8-800-000-00-00");  
order.addItem(item);
```

Так же вы можете указать агентские данные для всего чека, используя `getAgentData()` у экземпляра `SellOrder`.

После добавления предметов расчета нужно добавить способы оплаты с помощью метода `addPayment()`. Можно добавить по одной оплате каждого типа. При попытке добавить уже определенный тип оплаты метод вернет `false`. Сумма оплат должна быть равна сумме всех предметов расчета. Превышение суммы оплат возможно только для типа оплат «Наличные», в этом случае будет заполнено поле «Сдача».

Пример формирования оплаты.

```
order.addPayment(new Payment(PaymentType.Card, new  
BigDecimal(1075.79))); // Добавление безналичного платежа  
order.addPayment(new Payment(PaymentType.Cash, new  
BigDecimal(8000))); // Добавление наличного платежа
```

При формировании чека следующие поля заменят теги, определенные ранее разработчиком:

Значения по соответствующим ставкам НДС заменят соответствующие теги 1102-1107,

Значения оплат заменят соответствующие теги 1215,1216,1217,1031,1081.

Значения из поля `AgentData` заменят теги 1025,1223,1224

Значение поля `setRecipientAddress()` заменит тег 1008.

После формирования документа чека можно вызвать метод **`doSellOrder()`** которому передаются следующие параметры

`order` — сформированный документ

`operator` — описание оператора (кассира)

`signed` — документ, сформированный на основании входящего чека и дополненный данными фискализации и информации о сдаче (при наличии),

`doPrint` — признак, осуществлять ли печатать чека

`header` — шаблон заголовка чека (до списка предметов расчета)

`item` — шаблон строки предмета расчета

`footer` — шаблон подвала чека (способы оплаты, общие суммы и т. д.)

`footerEx` — дополнительный подавал, который может быть напечатан после основного (информация о скидках, акциях и т. д.)

Этот метод рекомендуется выполнять в потоке.

## Формирование чека коррекции

Механизм формирования чека коррекции похож на предыдущий механизм, т. е. сначала формируется экземпляр класса `Correction`, потом вызывается соответствующий метод Фискального ядра, который вернет объект дополненный данными фискализации.

Пример создания коррекции приведен ниже.

```
Correction cor = new Correction(CorrectionType.byArbitrariness,  
OrderType.Outcome, new BigDecimal(4000), VAT.vat_20,  
TaxMode.Common);
```

```
cor.setBaseDocumentDate(System.currentTimeMillis());  
cor.setBaseDocumentNumber("Возврат чека 11");  
cor.addPayment(new Payment(new BigDecimal(4000)));
```

Следующие теги в экземпляре Correction заменяют указанные пользователем

Используемая ставка НДС задаваемая в конструкторе заменяет тег 1055

Тип коррекции задаваемый в конструкторе заменяет тег 1173

Данные документа-основания заменяют тег 1174

Значение ставки НДС заменяют соответствующие теги 1102-1107

Данные о платежах заменяют теги 1215,1216,1217,1031,1081

Этот метод рекомендуется вызывать в потоке.

## Доступ к архиву проведенных документов

Для того, что бы обратиться к данным которых были фискализированы накопителем надо сделать выборку из content-provider **content://rs.fncore2.data/documents** Эти данные доступны только для чтения. Запрос вернет следующие поля

FNSN — строка, серийный номер ФН

DOCNO — число, фискальный номер документа

DOCDATE — число, дата документа в мс с 01.01.1970

DOCTYPE — строка, имя класса документа

CASIER — строка, имя оператора

BODY — двоичные данные, сериализованный документ

OFD — двоичные данные, ответ оператора ОФД если документ успешно отправлен.

Для восстановления документа нужно воспользоваться методом **deserializeDocument** из класса **Utils**.

## Доступ к сменным остаткам.

Для доступа к сменным остаткам предназначен content-provider

**content://rs.fncore2.data/registers**. Он предоставляет данные только для чтения и содержит в себе суммовые значения остатков за смену по типам платежа. Он предоставляет следующие поля

PType — тип платежа, соответствует значению ordinal() элемента перечисления PaymentType

PDIR- «направление» платежа (0 — приход, 1 - расход)

PVALUE — сумма платежей.

Таким образом вы можете получить значения по суммам прихода и расхода за смену для каждого типа платежей.

## Доступ и корректировка остатков наличности

Для быстрого получения остатков наличности можно использовать метод **double**

**getCashRest()** Этот метод возвращает сумму остатков (приход-расход) по типу платежей «Наличные». Для корректировки этого значения используется метод **int**

**putOrWithdrawCash(double v, in OU operator, String template)** который так же печатает нефискальный документ внесения/изъятия денежных средств. В этот метод передается сумма



(положительная — внесение, отрицательная — изъятие), реквизиты оператора (кассира) проводящего операцию и шаблон печати.

## Повторная печать документа

Для повторной печати (получения изображения чека) используется метод **int printExistingDocument(int number,in ParcelableStrings template,boolean doPrint, out ParcelableBytes result)**; В него передается номер документа, список шаблонов используемых для печати или пустой ParcelableStrings если используются шаблоны по умолчанию,признак осуществлять ли печать на принтере и сереализуемый массив байт. При doPrint == false в него будет помещено PNG изображение документа, к которому можно получить доступ через метод getRawBytes().

## Шаблоны документов

Для печати документов используется свой текстовый язык шаблонов. Для форматирования используется блок, синтаксис которого выглядит следующим образом  
{\тег параметры\данные блока}

Теги могут быть вложенными. Параметры описываются следующим образом:

**имя:значение;**

Общие параметры для всех блоков:

fontSize — размер шрифта. Может задаваться как в абсолютных величинах (точках) так и в процентах от размера шрифта по умолчанию. Пример:

**fontSize:20; fontSize:80%;**

style — тип шрифта. Допускает значения normal(обычный), bold (жирный) italic (наклонный) stikeout (зачеркнутый) underline (подчеркнутый). Типы могут комбинироваться через запятую. Пример:

**style:bold; style:bold,italic; style:italic,underline;**

if — условие, при котором будет печататься блок (не применим к блоку s). Синтаксис **if:lvalue cond rvalue**; где lvalue и rvalue сравниваемые значения, cond — оператор сравнения (= != > < ). Операторы «больше» и «меньше» применимы только для числовых выражений, если аргументы не являются числом то всегда возвращается истина. Аргументы стоит указывать в двойных кавычках Пример:

**if:"\$substitute\$"!=""; if:"\$numValue\$">"10";**

nobreak — запрет переноса в блоке. Значения могут быть true/false. Если установлен nobreak:true то текст внутри блока не переносится а обрезается по границе блока. Пример:

**nobreak:true;**

padding — отступы от границ блока (не применим к блоку s). Параметрами являются 4 числа через запятую задающие левую, верхнюю, правую и нижнюю границы в точках.

«Хвостовые» значения можно опускать. Пример:

**padding:20,10,5,5; padding:20,0,5; padding:10;**

**border** — ширина рамки вокруг блока в точках (не применим к блоку s). Правила такие же как и для параметра padding. Пример:

**border:2,2; border:0,0,0,2;**

**height** — высота блока (не применим к блоку s). Может быть указана как в пикселях, так и в процентах от размера предыдущего. Может быть использован для одного дочернего блока

символ \* - остаток высоты. Пример

```
{\block height:100;\n  {\block height:10%;\Высота блока 10 точек }\n  {\block height:*\;\ Высота блока 90 точек }\n}
```

**align** — выравнивание текста по блоку. Параметры right, left, center.

**valign** — вертикальное выравнивание в блоке. Параметры top,center,bottom.

**width** — ширина блока (не применим к блоку s). Параметры соответствуют блоку height.

Пример

```
{\block width:100%;\n  {\block width:40%;\ Ширина блока 40%}\n  {\block width:*\;\ Ширина блока 60%}\n}
```

Значение параметров style и fontSize наследуются для вложенных блоков. Т.е.

```
{\block style:bold\Это жирный шрифт\n  {\block\ Это тоже будет жирный шрифт}\n}\n{\block\ Это будет обычный шрифт}
```

### Используемые типы блоков

**s** — определение стиля. Этот блок не отображается на печати переносом, он просто задает стиль. Пример

Это {\s style:bold\жирный} а это {\s style:italic\наклонный текст}

**p** — параграф. Этот блок формирует перенос строки перед собой.

Тут находится текст{\r\не смотря ни на что, это будет на новой строке}

**table** — начало табличной разметки. В таблице может быть произвольное число строк, каждая из которых может содержать произвольное (не обязательно одинаковое во всех строках) число ячеек

**tr** — строка табличной разметки

**td** — ячейка табличной разметки

Ниже приведен пример простой таблицы

```
{\table width:300;\n  {\tr\n    {\td width:40%;\ Ячейка 1}\n    {\td width:*\;\ Ячейка 2}\n  }\n  {\tr\n    {\td width:100%;align:center;\Ячейка 3 }\n  }\n}
```

**image** - печать картинки. Содержимым тега должна быть картинка в формате base64

Пример

```
{\image width:40;height:40;\n  iVBORw0KGgoAAAANSUhEUgAAAGAAAIAACAMAAADDPiTIAAAAAA3NCSVQICAjb4U/
```

gAAAACXBIWXMAAEbeAABG3gGOJjJbAAAAGXRFWHRTb2Z0d2FyZQB3d3cuaW5rc2NhcGUub3Jnm+48GgAAAv1QTFR...}

**barcode** — печать штрихкода. Значением штрихкода является текст внутри блока. Содержит дополнительный параметр type определяющий тип баркода. Может принимать значения code128, ean8, ean13, code39, code93, qr и dm. Если не указан используется code128. Пример `{\barcode type:qr;width:150;height:150;\rightscan.ru}`

## Переменные используемые в шаблонах

Для каждого типа документов используется свой набор переменных. Переменная шаблона это значение ограниченное символами \$. Т.е. \$order.Number\$ и \$T\_1022.1045\$ являются переменными шаблона.

Для всех типов шаблонов кроме шаблона предмета расчета существуют общие переменные

signature.Date — фискальная дата документа

signature.Number — фискальный номер документа

signature.sign — фискальная подпись документа

device.Number — заводской номер ККМ

device.regNo — регистрационный номер ККМ

device.FN — серийный номер ФН

device.Version — версия ПО на устройстве

operator.Name — наименование оператора (кассира)

operator.INN — ИНН оператора (кассира)

Address — адрес расчетов

Location — место расчетов

owner.Name — наименование владельца ККМ

owner.INN — ИНН владельца ККМ

T\_XXXX[.YYYY] — получение значения тега по номеру. Если сам тег является TLV то через точку можно указать значение «вложенного» тега. Это поле так же применимо для строки предмета расчета

Для документа **Регистрация/изменение информации о ККТ** доступны следующие переменные

reason.Type — тип причины перерегистрации (регистрация, замена ФН, изменение данных о ККМ)

reason.Name — наименование причины перерегистрации

TaxModes — список СНО через запятую

encryption — режим «Шифрование» (Да/Нет)

isInternetMode — режим «Продажа в сети Интернет»

isServiceMode — режим «Оказание услуг»

isExcisesMode — режим «Продажа подакцизного товара»

isCasinoMode — режим «Проведение азартных игр»

isLotteryMode — режим «Проведение лотереи»

fns\_url — адрес сайта ФНС

sender\_email — почтовый адрес отправителя

ofd.INN — ИНН ОФД

ofd.Name — наименование ОФД

Для документа **Отчет о закрытии/открытии смены** доступны следующие переменные

shift.NumDocuments — номер последнего документа за смену

shift.NumChecks — количество чеков за смену

ofd.NumUnsent — количество не отправленных в ОФД документов

ofd.DateUnsent — дата первого неотправленного документа в формате ДД/ММ/ГГГГ, ЧЧ:мм

ofd.FirstUnsentNo — номер первого неотправленного документа

shift.Number — номер смены

shift.IsOpen — признак «Смена открыта»

Для документа **Чек продаж/возврата** доступны следующие переменные

order.Type — тип чека (приход, расход, возврат прихода, возврат расхода)

order.Number — номер чека

order.Sum.Total — общая сумма по чеку

order.Payment.xxxx — сумма платежа по типу, где xxxx — имя из перечисления PaymentType

order.Refund — сумма сдачи наличными по чеку

order.Barcode — значение qr кода чека

order.AgentType — тип агентской услуги (если задан)

Для строки **предмета расчета**

item.name — наименование предмета расчета

item.qty — количество с точностью 3 знака после запятой

item.measure — наименование единицы измерения

item.price — цена

item.sum — сумма

item.Vat.Name — наименование ставки НДС

item.Vat.Value — значение ставки НДС

item.PaymentType — наименование типа оплаты

item.ItemType — наименование типа предмета расчета

item.AgentType — наличие признака агента

Для документа **внесения/изъятия ДС**

owner.INN — ИНН владельца ККМ

owner.Name — наименование владельца ККМ

Date — дата документа

isIncome — признак внесения

sum — сумма операции

Шаблоны для документов по умолчанию находятся в составе СДК.