CSI

197136, Санкт-Петербург, Чкаловский пр., д.50, литера А, пом.5-Н, 2 этаж.
+7 (812) 331-22-55, crystals@crystals.ru
115432 Москва, пр. Андропова, 18, корп. 5, бизнес-парк Nagatino i-Land
+7 (495) 640-63-07, moscow@crystals.ru
8 (800) 222-22-51, www.crystals.ru

Руководство по созданию плагинов для *Set Retail* 10

Дата последних изменений: 18.05.2020

CSI

197136, Санкт-Петербург, Чкаловский пр., д.50, литера А, пом.5-Н, 2 этаж.
+7 (812) 331-22-55, crystals@crystals.ru
115432 Москва, пр. Андропова, 18, корп. 5, бизнес-парк Nagatino i-Land
+7 (495) 640-63-07, moscow@crystals.ru
8 (800) 222-22-51, www.crystals.ru

Содержание

CSI

197136, Санкт-Петербург, Чкаловский пр., д.50, литера А, пом.5-Н, 2 этаж.
+7 (812) 331-22-55, crystals@crystals.ru
115432 Москва, пр. Андропова, 18, корп. 5, бизнес-парк Nagatino i-Land
+7 (495) 640-63-07, moscow@crystals.ru
8 (800) 222-22-51, www.crystals.ru

CSI

197136, Санкт-Петербург, Чкаловский пр., д.50, литера А, пом.5-Н, 2 этаж.
+7 (812) 331-22-55, crystals@crystals.ru
115432 Москва, пр. Андропова, 18, корп. 5, бизнес-парк Nagatino i-Land
+7 (495) 640-63-07, moscow@crystals.ru
8 (800) 222-22-51, www.crystals.ru

1. История изменений

Дата	Описание изменений	Версия кассы, начиная с которой данная версия *API* поддерживается
24.11.2017	Первая версия документа	
04.12.2017	Изменена схема *metainf.xml*. Добавлен раздел про локализованные ресурсы. В пример кода внесены изменения по работе с локализованными ресурсами. Добавлен раздел с историями изменений.	
12.12.2017	Изменена схема *metainf.xml*, *options* имеют отдельные типы. Исправлено комментирование в примерах локализованных файлов в соответствии с *DTD* схемой. В интерфейс чека *Receipt* добавлен метод *getSurchargeSum()*. Изменены интерфейсы некоторых экранных форм. Добавлен *utils/MetainfValidator.jar* для проверки соответствия файла *metainf.xml xsd*.	
15.12.2017	Версия *API* 0.0.2. В *xsd* манифеста добавлен атрибут для указания версии плагина. В пример добавлена реализация метода возврата. В интерфейс *RefundRequest* добавлен метод *getRefundReceipt()* для получения чека возврата.	
20.12.2017	Версия *API* 0.0.3. В *xsd* манифеста добавлен типа параметра *boolean*, в строковой тип (*string*) добавлен атрибут *masked* для маскирования текста при вводе. В интерфейс *PropertiesReader* добавлены методы для получения *boolean* настроек. В пример плагина добавлена реализация аннулирования. В класс *Payment* добавлен метод *getData(String key)*.	
16.01.2018	Добавлена информация о разработке плагинов лояльности.	
17.01.2018	Добавлена информация о разработке плагинов карт.	
25.01.2018	Версия *API* 0.0.6. Добавлена экранная форма сообщения кассиру.	
12.02.2018	Версия *API* 0.0.7. Добавлена информация о списании бонусов как скидки.	
15.02.2018	Версия *API* 0.0.7. Внесены изменения в список библиотек, допустимых для использования.	

197136, Санкт-Петербург, Чкаловский пр., д.50, литера А, пом.5-Н, 2 этаж.
+7 (812) 331-22-55, crystals@crystals.ru
115432 Москва, пр. Андропова, 18, корп. 5, бизнес-парк Nagatino i-Land
+7 (495) 640-63-07, moscow@crystals.ru
8 (800) 222-22-51, www.crystals.ru

28.02.2018	Версия *API* 0.0.8. *PropertiesReader* - добавлено описание возможности сохранения параметров сервиса или плагина в БД.	
16.03.2018	Версия *API* 0.0.9. Возможность добавлять несколько слипов в оплату. Возможность получить информацию о чеке при аннулировании. Возможность получения информации о кассире.	
21.03.2018	Версия *API* 0.0.10. В *UIForms* добавлен метод *showTimingOutForm* для отображения на экране кассы окна ожидания выполнения длительной операции с обратным отсчетом. Из *CardSearchEventSource* удален источник данных для поиска карты “номер мобильного телефона”, добавлен асинхронный метод *searchCardByMobileNumber* для поиска карты по номеру мобильного телефона, в котором позволительно управлять *UI* кассы. Структура *CardInfoResponse* переименована в *CardSearchResponse*, метод *getCardInfo* интерфейса *CardPlugin* переименован в *searchCard*, структура *CardInfoResponseStatus* переименована в *CardSearchResponseStatus*.	
23.03.2018	Версия *API* 0.0.11. Добавлена возможность печатать слипы из плагина лояльности.	
26.03.2018	Версия *Metainf* 0.0.6. В *manifest.xml* добавлено опциональное поле *Description*, которое содержит человекочитаемое описание плагина. Поле ограничено 3000 символами.	
05.04.2018	Версия *Metainf* 0.0.7. В файле *metainf.xml* добавлена возможность задать параметрам значения по умолчанию, чтобы использовать их для настройки плагина на сервере в случае, когда в БД для плагина ещё не было сохранено настроек. Добавлен комментарий об ограничениях в использовании аннотации *ru.crystals.pos.spi.annotation.Inject.*	
09.04.2018	Изменения форматирования в документе, подпись всех рисунков, кодов и таблиц, дополнение главы с описанием файла *manifest.xml* подробным описанием используемых элементов и их атрибутов, добавлено описание особенностей поведения номера чека в плагинах карт и лояльности.	
11.04.2018	Обновлено оглавление. Приведено описание полей в манифесте *jar* плагина.	

197136, Санкт-Петербург, Чкаловский пр., д.50, литера А, пом.5-Н, 2 этаж.
+7 (812) 331-22-55, crystals@crystals.ru
115432 Москва, пр. Андропова, 18, корп. 5, бизнес-парк Nagatino i-Land
+7 (495) 640-63-07, moscow@crystals.ru
8 (800) 222-22-51, www.crystals.ru

12.04.2018	Версия *API* 0.0.12. Добавлен метод *ru.crystals.pos.spi.POSInfo#getShiftNumber()* для получения информации о номере текущей смены. В плагин оплат добавлен метод *PaymentPlugin#createDailyReport* для совершения сверки итогов.В плагин оплат добавлен метод *hasDailyReport*, который позволяет проверить, умеет ли плагин выполнять сверку итогов, не вызывая сам метод сверки.	10.2.46.0
18.04.2018	Дано краткое толкование понятия “Опердень”. Внесено пояснение о написании плагинов в отдельных классах и идентификаторах плагинов.	
25.04.2018	Переработаны примеры плагинов в *SDK*, добавлена поддержка аннотации *javax.annotation.PostConstruct*, помеченный которой метод будет вызываться после загрузки плагина. Эта возможность доступна для кассы версии от 10.2.47.0 и выше.	
08.05.2018	Версия *API* 0.0.13. версия *Metainf* 0.0.8. Добавлен плагин валидации акцизных марок товаров. Соответствующий пример.	10.2.48.0
31.05.2018	Версия *API* 0.0.14. версия *Metainf* 0.0.9. Внесены изменения в методы `eventReceiptFiscalized`, `onRepeatSend` интерфейса плагина валидации акцизных марок. Добавлен плагин типа товаров и соответствующий пример. Добавлена визуальная форма ввода текста по шаблону.	10.2.49.0
05.06.2018	Версия *API* 0.0.15. Экранной форме *showInputScanNumberForm* добавлена возможность прослушивать событие “считана карта с магнитной полосой”.	10.2.49.0
07.06.2018	Версия *API* 0.0.16 Добавлен интерфейс *ReconciliationReportMaker*, реализация которого плагином позволяет последнему участвовать в операции сверки итогов. Плагинам товаров добавлена возможность участвовать в сверке итогов.	10.2.49.0
07.06.2018	Версия *API* 0.0.17 Теперь плагины могут использовать еще один *Injectabl*e: *Printe*r - позволяет плагину печатать слипы на фискальном принтере.	10.2.50.0
20.06.2018	Версия *API* 0.0.18 Плагинам лояльности добавлена возможность добавить к результату расчета скидок список сообщений, которые следует отобразить кассиру.	10.2.50.0
02.07.2018	Версия *API* 0.0.19 Добавлена поддержка дисплея покупателя, обновлён пример плагина товаров для демонстрации работы с дисплеем покупателя.	10.2.51.0

197136, Санкт-Петербург, Чкаловский пр., д.50, литера А, пом.5-Н, 2 этаж.
+7 (812) 331-22-55, crystals@crystals.ru
115432 Москва, пр. Андропова, 18, корп. 5, бизнес-парк Nagatino i-Land
+7 (495) 640-63-07, moscow@crystals.ru
8 (800) 222-22-51, www.crystals.ru

09.07.2018	Версия *API* 0.0.20 Добавлена идентификация типа чека: чек продажи или чек возврата. Чеки возврата теперь могут содержать в себе чек продажи. Добавлено перечисление *ru.crystals.pos.spi.receipt.ReceiptType*, определяющее тип чека. Интерфейсу *ru.crystals.pos.spi.receipt.Receipt* добавлен метод *getType* для получения типа чека, метод *getSaleReceipt* для получения чека продажи для чека возврата.	10.2.52.0
16.07.2018	Версия *API* 0.0.21 Добавлена возможность получения события перед фискализацией чека. Добавлен класс *PreFiscalizationFeedback* использующийся для получения возможных слипов от плагинов лояльности перед фискализацией чека.	10.2.52.0
18.07.2018	Версия *API* 0.0.22 Плагинам лояльности добавлена возможность сохранять в чек произвольные параметры, которые сохраняются на протяжении всей жизни чека и могут быть выгружены в *ERP*.	10.2.53.1
24.07.2018	Версия *API* 0.0.23 Плагинам лояльности добавлена возможность уведомить кассу о начислении на карту бонусных баллов после расчета скидок или перед фискализацией. Транзакция начисления бонусных баллов может отображаться на сервере в Опердне и выгружаться в *ERP*.	10.2.53.1
28.08.2018	Версия *API* 0.0.24 Интерфейс *Receipt* дополнен методом для получения даты создания чека	10.2.55.0
31.08.2018	Версия *API* 0.0.25 Плагинам карт добавлена возможность отображения дополнительной информации по запрашиваемым картам	10.2.56.1
04.09.2018	Версия *API* 0.0.26 Плагинам лояльности добавлен метод, вызываемый при отмене расчета скидок. В класс *ru.crystals.pos.api.ext.loyal.dto.auxiliaries.PreFiscalizationFeedback* добавлен метод *getFeedbacks*, который позволяет плагину создать задание на отложенную отправку при наступлении события предфискализации чека. В интерфейс плагина лояльности *ru.crystals.pos.api.plugin.LoyaltyPlugin* добавлен метод *cancelDiscount*, который вызывается кассой при отмене расчета скидок. При отмене скидок возможно изменить дополнительные атрибуты чека (*ru.crystals.pos.spi.receipt.Receipt#getData*) и создать задание на отложенную отправку.	10.2.56.1

197136, Санкт-Петербург, Чкаловский пр., д.50, литера А, пом.5-Н, 2 этаж.
+7 (812) 331-22-55, crystals@crystals.ru
115432 Москва, пр. Андропова, 18, корп. 5, бизнес-парк Nagatino i-Land
+7 (495) 640-63-07, moscow@crystals.ru
8 (800) 222-22-51, www.crystals.ru

04.09.2018	Версия *API* 0.0.27 У позиций чека (*ru.crystals.pos.spi.receipt.LineItem*) появились признаки: ● *isPayBonusAllowed* - разрешение списания бонусов как скидки; ● *isAccrueBonusAllowed* - разрешение начисления бонусов за приобретение позиции. У скидки (*ru.crystals.pos.api.ext.loyal.dto.Discount*) появился признак: ● *receiptWideDiscount* - скидка на позицию является частью скидки на чек	10.2.56.1
05.09.2018	Версия *API* 0.0.28 Добавлена возможность производить дополнительные действия после добавления позиции в чек (*ru.crystals.pos.spi.plugin.goods.AddForSaleCallback#completed(NewLineItem, Collection)*), метод *completed(NewLineItem)* объявлен как *deprecated.*	10.2.56.1
07.09.2018	Версия *API* 0.0.29 Упорядочены и добавлены к *API*-сущностям методы *toString*()	10.2.56.1
18.09.2018	Версия *API* 0.0.30 Классу *ru.crystals.pos.api.ext.loyal.dto.auxiliaries.LoyProviderFeedback* добавлено поле *savingStrategy*, которое определяет стратегию сохранения фидбеков от плагинов лояльности.	10.2.57.0
03.10.2018	Версия *API* 0.0.31 Интерфейсу *ru.crystals.pos.spi.receipt.Card* и всем реализующим его классам добавлен метод *getInputType*, возвращающий способ, которым карта была введена на кассе.	10.2.58.0
05.10.2018	Версия *API* 0.0.32 Из плагина карт добавлена возможность добавить карте произвольные атрибуты после её поиска. Данные атрибуты карты становятся доступны всем плагинам, которые явно или неявно принимают карты в качестве аргументов.	10.2.58.0
25.10.2018	Версия *API* 0.0.33 В структуру, хранящую фидбеки от плагина лояльности, добавлено поле, определяющее действие с фидбеком в случае, если чек, который образовал эти фидбеки, аннулируется. См. *ru.crystals.pos.api.ext.loyal.dto.auxiliaries.LoyProviderFeedback#getActionOnReceiptCancellation, ActionOnReceiptCancellation*	10.2.60.0
06.11.2018	Версия *API* 0.0.34 В плагин товаров (*ru.crystals.pos.api.plugin.GoodsPlugin*) добавлен метод, вызываемый кассой при совершении произвольного возврата плагинного товара - *addForRefund.*	10.2.60.0

197136, Санкт-Петербург, Чкаловский пр., д.50, литера А, пом.5-Н, 2 этаж.
+7 (812) 331-22-55, crystals@crystals.ru
115432 Москва, пр. Андропова, 18, корп. 5, бизнес-парк Nagatino i-Land
+7 (495) 640-63-07, moscow@crystals.ru
8 (800) 222-22-51, www.crystals.ru

09.11.2018	Версия *API* 0.0.35, *API Metainf* 0.0.10 1. Настройкам плагина и внешней системы, которой он подчинен, добавлен атрибут "*required*", который используется в *GUI* сервера и запрещает или разрешает не заполнять настройку плагина значением. 2. Строковым настройкам (*StringOptionType*) плагина и внешней системы добавлен косметический атрибут *multiline*, выставление в *true* которого приводит к отображению строкового параметра на *GUI* сервера в несколько строк вместо одной. 3. В настройки плагинов и внешних систем добавлен тип параметра, хранящий значения с плавающей точкой - *OptionDecimal.* *4.* Плагину карт добавлен метод списания бонусов, поддерживающий манипулирование *UI* кассы. См. *ru.crystals.pos.api.plugin.CardPlugin#writeOffAsync*.	10.2.61.0
13.11.2018	Версия *API* 0.0.36 Добавлена возможность распечатать *QR*-код в слипе.	10.2.61.0
14.11.2018	Версия *API* 0.0.37 В *ru.crystals.pos.spi.currency.ExtCurrencyHandler* добавлены методы конверсии денежных и количественных величин.	10.2.61.0
19.11.2018	Версия *API* 0.0.38 Плагинам дана возможность отлавливать события открытия и закрытия кассовой смены. Добавлен слушатель смены *ru.crystals.pos.api.events.ShiftEventListener*, маркерный интерфейс *ru.crystals.pos.spi.POSEventListener*, аннотация *ru.crystals.pos.spi.annotation.POSEventListener*	10.2.61.0
26.11.2018	Версия *API* 0.0.39 В плагины лояльности добавлен метод, оповещающий плагин об окончании расчета скидок на кассе. При этом плагину допустимо подкорректировать максимальную величину списания бонусов по картам. См. метод *LoyaltyPlugin#onDiscountCalculationFinished*	10.2.61.0
27.11.2018	Версия *API* 0.0.40 В *ru.crystals.pos.spi.equipment.SetApiPrinter* добавлен метод *getPaperWidth* для получения информации о ширине чековой ленты.	10.2.63.0

197136, Санкт-Петербург, Чкаловский пр., д.50, литера А, пом.5-Н, 2 этаж.
+7 (812) 331-22-55, crystals@crystals.ru
115432 Москва, пр. Андропова, 18, корп. 5, бизнес-парк Nagatino i-Land
+7 (495) 640-63-07, moscow@crystals.ru
8 (800) 222-22-51, www.crystals.ru

15.01.2019	Версия *API* 0.0.41 1. Чековые слипы можно печатать в составе чека, для чего им добавлен атрибут *separated*, определяющий, следует ли слип печатать отдельно или в составе чека. По умолчанию слип печатается отдельно. См. *ru.crystals.pos.api.ext.loyal.dto.Slip#isSeparated.* 2. Для печатных слипов добавлена возможность отключать печатаемый фискальным принтером в заголовке слипа логотип, для чего добавлен флаг *disableLogo*, которое отключает стандартный логотип печатного документа на время печати слипа. См. *ru.crystals.pos.api.ext.loyal.dto.Slip#isDisableLogo.* 3. В чековых слипах добавлена возможность указывать шрифт различного размера и толщины для различных блоков текста. См. *SlipParagraph#paragraphParts и SlipParagraph#FORMATTED_TEXT_PREFIX.* 4. Появилась возможность печати изображений в составе слипа. См *SlipParagraphType#IMAGE*. 5. В настоящий документ добавлена глава “Печать документов”.	10.2.63.0
22.02.2019	Версия *API* 0.0.42 В структуру *BonusWriteOffOperationResponse* добавлены дополнительные атрибуты карты и чека - *cardExtendedAttributesMap* и *receiptExtendedAttributesMap*.	10.2.65.0
25.04.2019	Версия *API* 0.0.43 В *CardSearchEventSource* добавлены источники данных для поиска карты “номер телефона” и “*e-mail*”, добавлен асинхронный метод *searchCardAsync* для поиска карты по *e-mail* или номеру телефона, в котором позволительно управлять *UI* кассы. Метод *searchCardByMobileNumber* объявлен *deprecated*.	
16.05.2019	В настоящий документ добавлена глава “Управление интерфейсом кассы”.	
06.06.2019	Версия *API* 0.0.44 При удалении товара из чека плагин товара должен разрешать удаление до и после подтверждения от кассира. Интерфейс *Receipt* дополнен методом для получения данных о проведённых оплатах. В плагин оплаты добавлена возможность получать уведомления о фискализации и создавать отложенную задачу. Обновлены примеры плагинов оплаты и товара.	
21.06.2019	Версия *API* 0.0.45 Добавлен интерфейс *ru.crystals.pos.spi.ui.MessageQueue*, подключаемый в плагины через аннотацию *@Inject,* который позволяет планировать отображение сообщения на кассе после завершения работы с чеком.	

14.08.2019	Версия *API* 0.0.46 1. Добавлена структура *ru.crystals.pos.api.comm.CommunicationMessage*, хранящая расширенное сообщение для покупателя или кассира, которое может быть снабжено колонтитулами и изменяемой иконкой. Отображение на кассе такого типа сообщения доступно только для плагинов карт. См. *CardSearchResponse#getCommunicationMessages*. 2. В слипах теперь можно отключать реквизиты магазина, таким образом для печати стало возможно использовать всё пространство слипа. См. поле *Slip#disableRequisites.* 3. Добавлена возможность разрешать или запрещать отрез слипа от чековой ленты. См. *Slip#disableCut*. 4. Слипу можно задать порядок следования за фискальным документом. См. *Slip#index*. 5. При форматировании текста на печатных слипах допустимо использовать абстрактные теги размера “*NORMAL*”, “*SMALL*”, “*BIG*”. 6. При форматировании текста на печатных слипах допустимо использовать абстрактные теги формата шрифта “*NORMAL*”, “*BOLD*”, “*ITALIC*”. 7. В структуру *MerchandiseEntity* добавлено поле, хранящее код товарной группы товара. См. *MerchandiseEntity#groupCode*. 8. В ту же структуру добавлено поле *goodsType*, определяющее тип товара: штучный или весовой. Несмотря на это, плагины товаров всё ещё могут оперировать штучным товаром только. 9. В *POSInfo* добавлен метод *getPOSType*, возвращающий тип кассы, на которой работает плагин: клавиатурная, touch, касса самообслуживания, серверный калькулятор лояльности. 10. В *POSInfo* добавлен метод *getPOSTemplateGuid*, возвращающий код шаблона, который использует касса, на которой развернут плагин. Шаблоном кассы задаётся внешний вид и функционально назначение кассы. 11. В *POSInfo* добавлен метод *getOperationMode*, который возвращает текущий режим работы кассы: продажа, возврат, произвольный возврат. 12. При добавлении карты в чек теперь возможно выкинуть уже добавленную в чек карту и заменить её своей. См. *CardSearchResponse#getCardsToRemove()*. 13. В плагинах лояльности после фискализации чека стало возможным добавить на печать ещё печатных слипов. Для этого метод *LoyaltyPlugin#eventReceiptFiscalized(Receipt, LoyaltyResult)* помечен как устаревший, перестал вызываться кассой и больше не должен быть использован для обработки событий фискализации. Вместо него следует использовать метод *onReceiptFiscalized(Receipt,*	10.2.71.0

	LoyaltyResult). Из соображений обратной совместимости, реализация данного метода по умолчанию вызывает устаревший метод. 14. Задания на отложенную отправку, которые плагин лояльности образовал ранее, теперь приходят в плагин пачками. Для чего метод *LoyaltyPlugin#onSendFeedback(LoyProviderFeedback)* был объявлен как устеревший, более не вызывается кассой и не должен быть использован. Вместо него следует использовать метод *onSendFeedback(Collection<LoyProviderFeedback>)*. 15. В структуру *CardSearchRequest* добавлено поле *receipt*, которое хранит текущий чек (если есть). Посему при поиске карты в плагине карт, последнему доступна возможность изучить чек. 16. В структуру *LineItem* добавлен метод *getAppliedDiscounts*, возвращающий детализированную информацию о скидках и рекламных акциях, применившихся на данную позицию. 17. В структуру *ProcessedPayment*, хранящую примененную к чеку оплату, добавлен метод *getCardNumber*, возвращающий номер карты в случае, если оплата выполнялась с использованием карты (банковской, подарочной или иной). 18. Добавлена возможность получить *QR* ОФД чека после фискализации. Для этого у структуры *Receipt* предусмотрено поле *qrCode*, которое заполняется кассой после фискализации. 19. В *Receipt* добавлен метод *getAppliedAdvertisingActions*, который возвращает список всех рекламных акций, сработавших в чеке, даже если срабатывание акции не привело к скидке. 20. В таблицу истории изменений настоящего документа добавлено поле с версией кассы, начиная с которой описанные изменения в *API* поддерживаются.	
20.08.2019	Версия *API* 0.0.47, версия *API Metainf* 0.0.11 Добавлен новый тип плагина - условие применения рекламной акции *Set Retail 10*. Данный тип плагина позволяет внешней системе разрешать или запрещать рекламной акции *Set Retail 10* участвовать в расчете скидок. Добавлена глава Создание плагина условия рекламной акции. Начиная с кассы версии 10.2.72.0 плагины карт могут ограниченно работать на сервере в таких продуктах, как *Set OMNI*, для чего манифест плагина снабжен атрибутом *workLevel*, определяющим окружение, в котором может работать плагин. Работа на сервере плагинов лояльности в настоящий момент не предусмотрена.	10.2.72.0
27.08.2019	Версия *API* 0.0.48 Добавлен новый метод *LineItem#getSoftReceiptId*, возвращающий идентификатор мягкого чека из которого добавлена это позиция.	10.2.73.0

05.09.2019	Версия *API* 0.0.49 В структуру *Receipt* добавлен метод *getFiscalizationTimestamp*, возвращающий дату и время фискализации чека.	10.2.73.0
20.09.2019	Версия *API* 0.0.50 Расширен дисплей покупателя (*ru.crystals.pos.spi.equipmentCustomerDisplay*), добавлены методы для отображения на нем сообщений с иконкой, заголовком и колонтитулами, таблиц. Способ и возможность отображения таких сообщений зависит от подключенной модели дисплея покупателя. См. методы *CustomerDisplay#append*, *CustomerDisplay#display*.	10.2.73.1
25.10.2019	Версия *API* 0.0.51, версия *API Metainf* 0.0.12 1. В структуре *CardSearchResponse* исправлены названия методов, имена которых ошибочно содержали смесь кириллицы и латиницы. В целях сохранения обратной совместимости методы оставлены и помечены как *deprecated*. Вместо них следует использовать новые методы: *addCashierMessages* *addCashierMessage* 2. Добавлен новый тип иконки *IconType.DISCOUNT_PERCENT* для расширенного сообщения покупателю или кассиру (*CommunicationMessage*). 3. В информацию о рекламных акциях, сработавших в чеке (*AppliedAdvertisingAction*), добавлены флаги запрета начисления и списания бонусов на товары по этой акции. 4. Добавлен новый тип плагина - расширение техпроцесса кассы. Данный тип плагина позволяет прерывать выполнение расчета скидок и дополнять его произвольной логикой. Краткое описание плагина приведено в главе Создание плагина тех. процесса настоящего документа. 5. Добавлен новый элемент графического интерфейса - форма выбора из вариантов. Описание приведено в главе Форма выбора из вариантов настоящего документа.	10.2.75.0
09.01.2020	Версия *API* 0.0.53 В структуру *CardInfo* добавлен метод *getCommunicationMessages,* позволяющий заменять встроенные сообщения пользователю в ситуациях, когда информация по карте недоступна.	10.2.78.0
22.01.2020	В настоящий документ добавлена глава “Возврат наличными из плагина оплаты”.	10.2.78.0
24.01.2020	Версия *API* 0.0.54 В структуру *BonusBalance*, описывающую бонусный баланс карты возвращаемой плагином карт, добавлен метод *getBonusProcessingDisplayName*, позволяющий передавать наименование бонусной программы/процессинга для отображения в *GUI* кассы.	10.2.79.0

	В интерфейс *CardSearchRequest*, где описаны параметры запроса поиска карты, добавлен метод *getLineItem*, который возвращает позицию, к которой применяется позиционный купон. В интерфейс *LineItem*, описывающий позицию в чеке, добавлен метод *getPositionCoupons*, который возвращает позиционные купоны, примененные к этой позиции.	
03.02.2020	Версия *API* 0.0.55 *И*з метода *LoyaltyPlugin#onDiscountCalculationFinished* появилась возможность вернуть дополнительные атрибуты чека и позиций в нём. Для этого в структуру *PostDiscountResult* добавлены методы *getReceiptAttributeMap*, *getPositionAttributeMap* и *addPositionAttribute*. В структуру *BonusBalance*, описывающую бонусный баланс карты возвращаемой плагином карт, добавлены методы *getMultiplier* (коэффициент перевода бонусокопеек в бонусорубли) и *getRate* (курс для конвертации бонусов в деньги)	10.2.79.0
13.02.2020	Версия *API* 0.0.57 В структуру *LoyaltyResult*, описывающую результат расчета скидок, возвращаемый плагином лояльности, добавлено поле *selectedPaymentType*, описывающее тип оплаты, к которому должна перейти касса при переходе к оплатам.	10.2.81.0
28.02.2020	Версия *Metainf* 0.0.14 Для плагинов оплат к фискальным типам добавлен ещё один тип - *BONUS*, который сигнализирует кассе о том, что плагин принимает оплату бонусными баллами. См. таблицу 8 главы Плагин оплат.	10.2.81.0
17.04.2020	Версия *API* 0.0.58 В структуру *Discount* добавлен параметр *quantity*, позволяющий указать количество товара в позиции, на которое предоставлена скидка.	10.2.83.0
28.04.2020	Версия *API* 0.0.59 Добавлен новый тип плагина - *AdvertisingActionProviderPlugin,* который позволяет добавить калькулятору скидок рекламные акции перед началом расчета скидок по чеку. Это может быть полезно для создания “индивидуальных предложений” для держателя карты. В таком случае можно добавлять в расчет рекламные акции в зависимости от номера или состояния карты, притом эти акции будут участвовать в конкуренции (способу выбора акции, которую следует применить в случае, когда она не суммируется всегда и есть аналогичная применимая акция с схожими результатами).	10.2.84.0
06.05.2020	Версия *API* 0.0.60 В структуру *Discount* добавлен параметр *discountType*, позволяющий тип скидки *DiscountType.SUM* (по умолчанию) или *DiscountType.BONUSES*.	10.2.84.0

197136, Санкт-Петербург, Чкаловский пр., д.50, литера А, пом.5-Н, 2 этаж.
+7 (812) 331-22-55, crystals@crystals.ru
115432 Москва, пр. Андропова, 18, корп. 5, бизнес-парк Nagatino i-Land
+7 (495) 640-63-07, moscow@crystals.ru
8 (800) 222-22-51, www.crystals.ru

	В структуру LineItem добавлен метод getAccruedBonuses, возвращающий детализированную информацию о бонусах, начисленных на данную позицию согласно акциям Set10. Добавлена поддержка работы с фишками. Добавлен метод BonusBalance.getBalanceType(), который может быть как для бонусов BalanceType.BONUSES, так и для фишек BalanceType.TOKENS. В CardPlugin добавлен метод списания - writeOff - с параметром бонусного счета для определения, что списываем именно фишки. В структуре LoyaltyResult добавлен метод getWriteOffsLimits, который возвращает потолки списания для балансов по карте. Старый метод getBonusWriteOffsLimits помечен, как устаревший. Структура BonusAccrualResult переработана для указания нескольких начислений. Начисление описывается новым классом BonusAccrual, в котором можно указать баланс, по которому произведено начисление.	

197136, Санкт-Петербург, Чкаловский пр., д.50, литера А, пом.5-Н, 2 этаж.
+7 (812) 331-22-55, crystals@crystals.ru
115432 Москва, пр. Андропова, 18, корп. 5, бизнес-парк Nagatino i-Land
+7 (495) 640-63-07, moscow@crystals.ru
8 (800) 222-22-51, www.crystals.ru

Перечень терминов

В документе использованы следующие термины и сокращения:

- *BMP - BitMap* - формат хранения изображений в виде битовой матрицы;
- *CRM* - *Customer relationship management*;
- *ERP* - *Enterprise resource planning*;
- *GUI* - (*Graphical User Interface*) - графический интерфейс пользователя;
- *MSR* - *Magnetic stripe reader*;
- *NFC* - *Near-field communication*;
- *QR* - *Quick Response Code*;
- *Set API* - прикладной программный интерфейс, посредством которого плагин взаимодействует с программным комплексом *Set Retail 10*;
- *Set Retail 10* - система управления магазином в настоящем руководстве рассматривается написание плагинов для которой;
- *Set OMNI* - модуль расширения *Set Retail 10*, позволяющий подключить к нему внешние каналы продаж (интернет-магазин, мобильное приложение, сервисы самостоятельного сканирования покупателем товаров, *etc.*);
- *UI* (*User Interface*) - интерфейс пользователя;
- *URL* - *Uniform Resource Identifier*;
- *VCS* - *Version Control System* - система контроля версий;
- БД - база данных;
- Мягкий чек - предварительный чек, который может быть создан не на кассе, но в интернет-магазине, например, и затем на кассе преобразован в чек фискальный. Служит для предоставления возможности сформировать список товаров вне кассы, тем самым экономя время;
- Опердень (Операционный день) - рабочее время, в течение которого кассой выполняется обслуживание клиентов;
- ОФД - оператор фискальных данных;
- Произвольный возврат - возврат товара не по чеку продажи;
- ПО - программное обеспечение;
- РА - рекламная акция;
- ФФД - формат фискальных документов - набор требований к документам, которые составляет и передает контрольно-кассовая техника;
- ШК - штрих-код.

197136, Санкт-Петербург, Чкаловский пр., д.50, литера А, пом.5-Н, 2 этаж.
+7 (812) 331-22-55, crystals@crystals.ru
115432 Москва, пр. Андропова, 18, корп. 5, бизнес-парк Nagatino i-Land
+7 (495) 640-63-07, moscow@crystals.ru
8 (800) 222-22-51, www.crystals.ru

2. Введение

Для интеграции с внешними сервисами программный комплекс *Set Retail 10* предоставляет возможность создавать динамически подключаемые плагины. Взаимодействие с плагинами осуществляется при помощи *Java API*.

Основное взаимодействие с плагинами происходит на кассе, где непосредственно осуществляется торговый процесс. Кассы централизованно настраиваются на серверах *Set Retail 10*.

Плагин состоит из двух частей: скомпилированного *Java* кода и описания его настроек в виде файла *metainf.xml*. Для создания плагина необходимо обладать навыками программирования на языке *Java*. Структурная схема взаимодействие внешних систем с кассой посредством плагинов *Set API* представлена на рисунке 1.

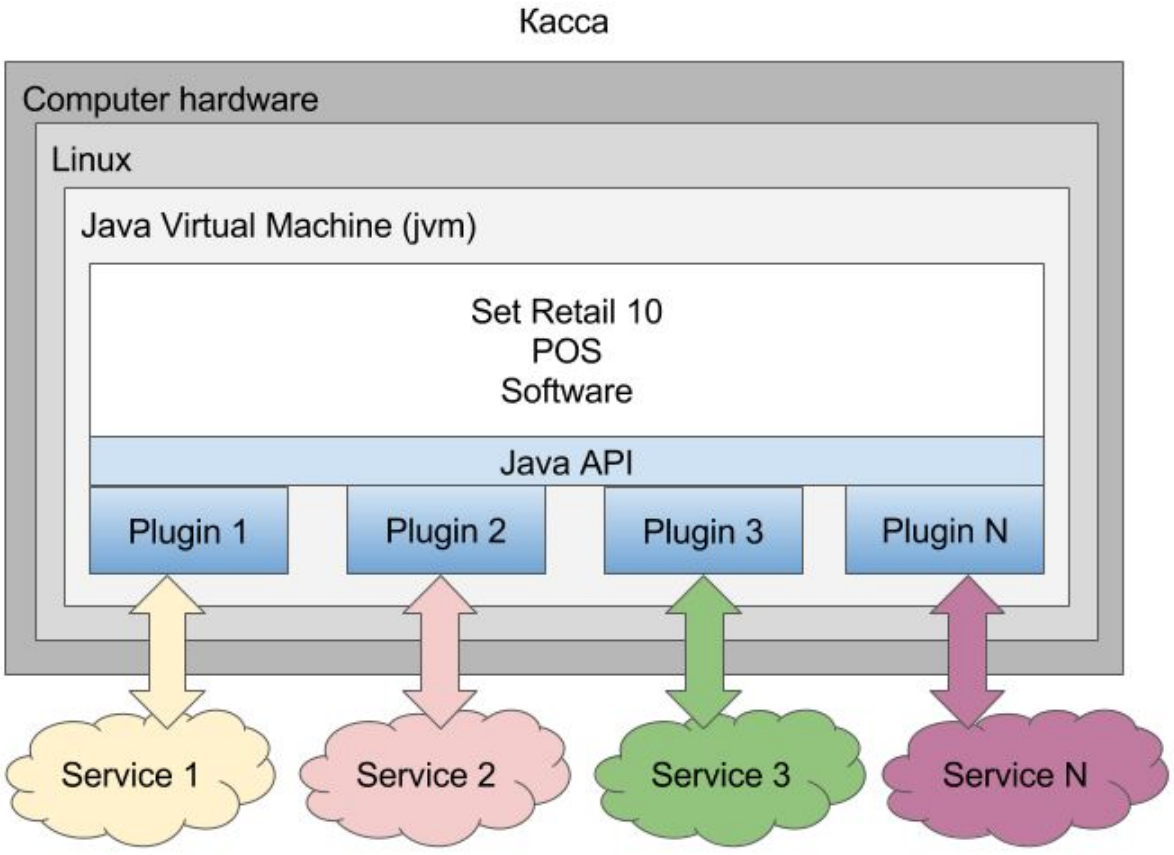

Рисунок 1 - взаимодействие внешних систем с кассой посредством плагинов *Set API*

197136, Санкт-Петербург, Чкаловский пр., д.50, литера А, пом.5-Н, 2 этаж.
+7 (812) 331-22-55, crystals@crystals.ru
115432 Москва, пр. Андропова, 18, корп. 5, бизнес-парк Nagatino i-Land
+7 (495) 640-63-07, moscow@crystals.ru
8 (800) 222-22-51, www.crystals.ru

3. Процесс формирования чека

Блок-схема процесса формирования чека кассой *Set 10* представлена на рисунке 2.

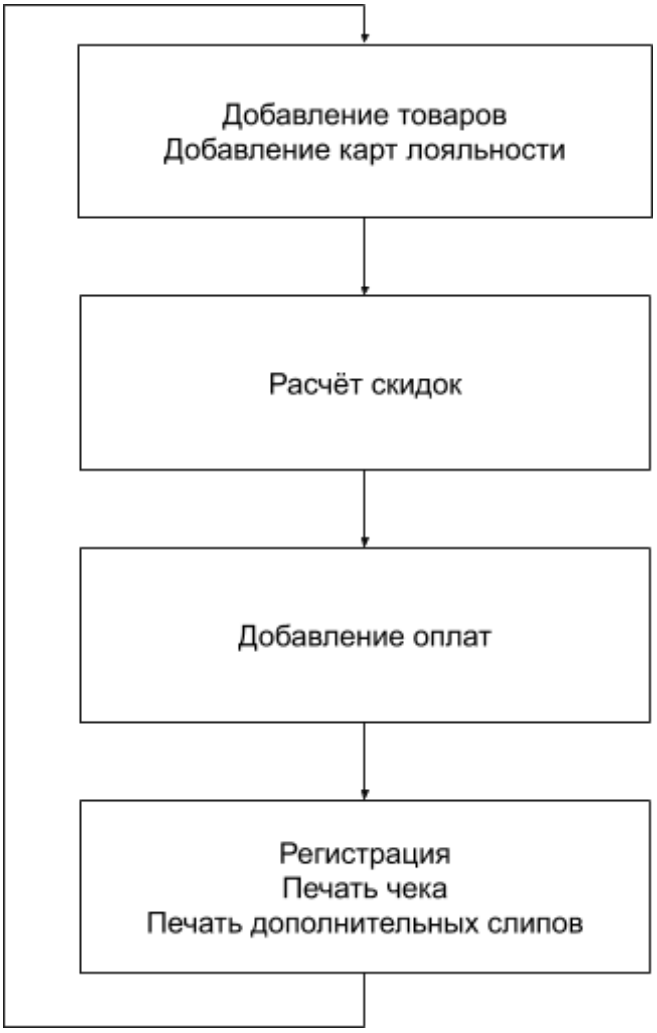

Рисунок 2 - процесс формирования чека

4. Создание плагина лояльности

Плагины[1] лояльности позволяют предоставить различные преференции покупателю при приобретении им товаров на кассе - см. этап “Расчет скидок” схематичного изображения техпроцесса продажи на рисунке 2.

Процесс предоставления преференций запускается при переходе кассы в состояние “Подытога” (переход может быть вызван, например, нажатием клавиши “Подытог” на кассовой клавиатуре). В данном состоянии касса последовательно применяет преференции (скидки, начисления бонусов, добавление заданий на печать купонов/рекламы в процессе фискализации чека) от активированных[2]

[1] далее в тексте “плагин” и “поставщик услуг лояльности” подразумевают логически одно и то же.
[2] под активированным поставщиком услуг лояльности (плагином) понимаются плагин, что был обнаружен кассой и для которого в *SET10* заведена соответствующая РА (Рекламная Акция), дающая право этому плагину на предоставление преференций.

поставщиков “услуг лояльности” (плагинов) к текущему чеку и безусловно переходит в режим внесения оплат. Т.е. подавтомат расчета скидок выглядит примерно следующим образом (рисунок 3):

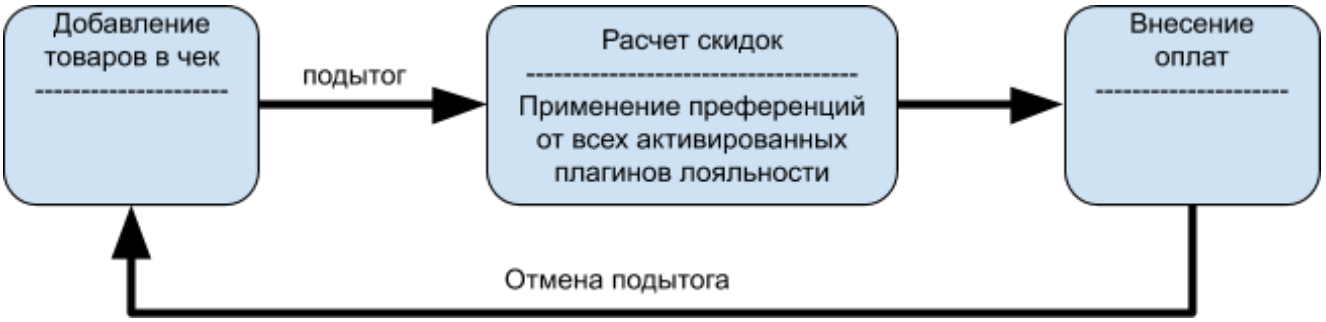

Рисунок 3 - автомат расчета скидок

В свою очередь, взаимодействие (в рамках/контексте одного чека) с каждым из плагинов лояльности выглядит следующим образом:

- на этапе расчета скидок плагин получает текущий чек (с уже примененными преференциями от предыдущих[3] поставщиков услуг лояльности) и применяет к нему свои скидки/преференции;

Внимание! Номер чека, получаемый плагином лояльности, может отличаться от номера, который чек получит при фискализации в случае, если после расчета скидок печатались иные документы, вроде Х-отчета или копии чека, например.

- далее, уже на этапе фискализации чека, плагин получает оповещение о том, какие преференции окончательно в чеке остались (были применены). На этом этапе плагин может приготовить *feedback* для своего процессинга с информацией о выданных преференциях в чеке;
- далее с некоторой периодичностью (по таймаутам) в “теневом” потоке (*background thread*) плагин получает свой ранее заготовленный *feedback* и может отправить его в свой процессинг[4].

Визуально взаимодействие с плагинами можно представить следующими картинками (рисунки 4-6):

[3] под “предыдущими” плагинами понимаются плагины, чья очередность в предоставлении преференций на чек “раньше”/”выше” “текущего” плагина.

[4] плагин должен быть готов к тому, что его *feedback’u* могут быть переданы ему не в той последовательности, в которой они создавались: например, касса *SET10* может выбрать следующую стратегию попыток отправить *feedback’u* (через плагины): сначала отправлять те, что имеют меньшее количество безуспешных попыток отправки.

CSI
197136, Санкт-Петербург, Чкаловский пр., д.50, литера А, пом.5-Н, 2 этаж.
+7 (812) 331-22-55, crystals@crystals.ru
115432 Москва, пр. Андропова, 18, корп. 5, бизнес-парк Nagatino i-Land
+7 (495) 640-63-07, moscow@crystals.ru
8 (800) 222-22-51, www.crystals.ru

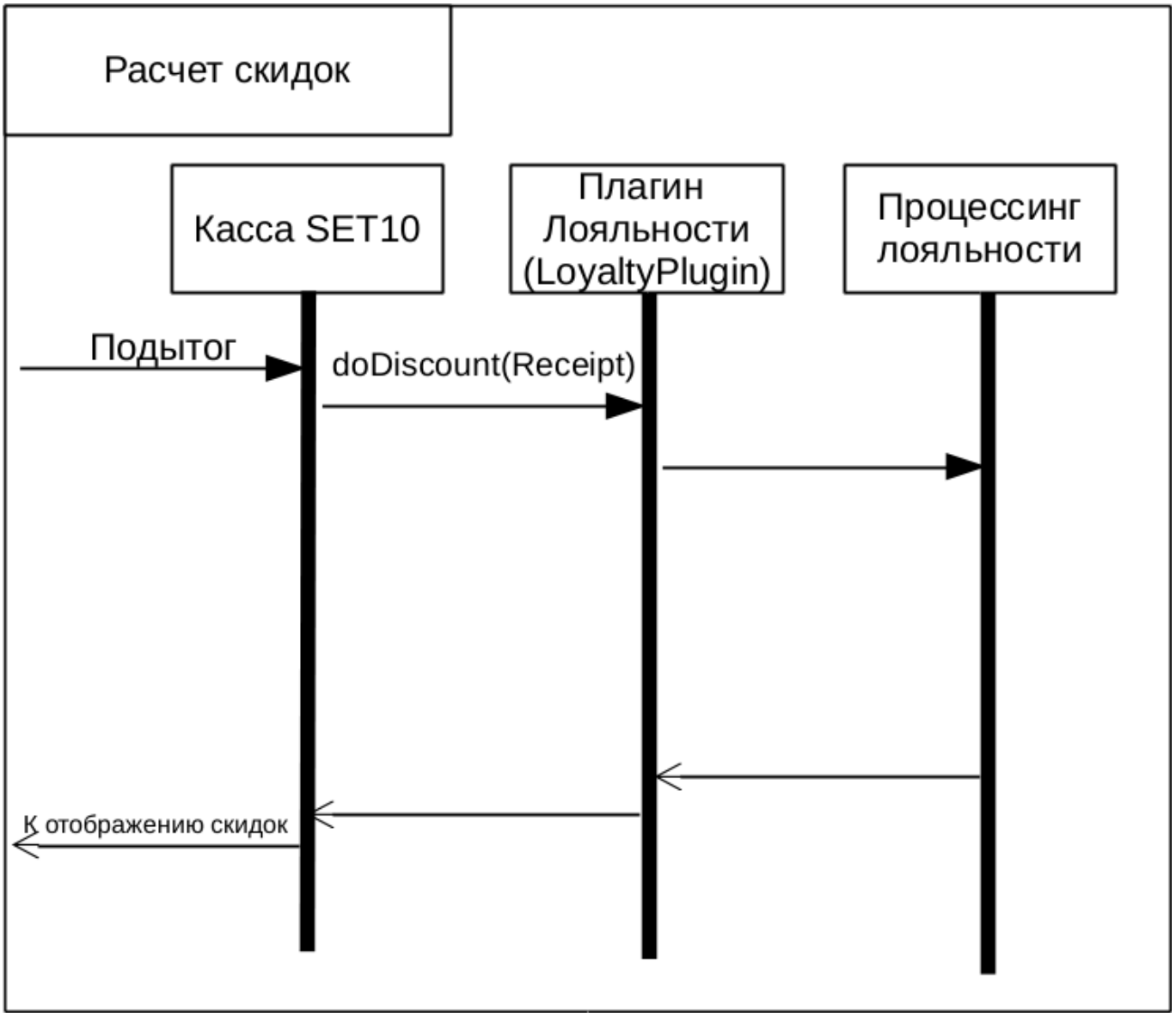

Рисунок 4 - последовательность взаимодействия кассы в плагином лояльности при расчете скидок

CSI
197136, Санкт-Петербург, Чкаловский пр., д.50, литера А, пом.5-Н, 2 этаж.
+7 (812) 331-22-55, crystals@crystals.ru
115432 Москва, пр. Андропова, 18, корп. 5, бизнес-парк Nagatino i-Land
+7 (495) 640-63-07, moscow@crystals.ru
8 (800) 222-22-51, www.crystals.ru

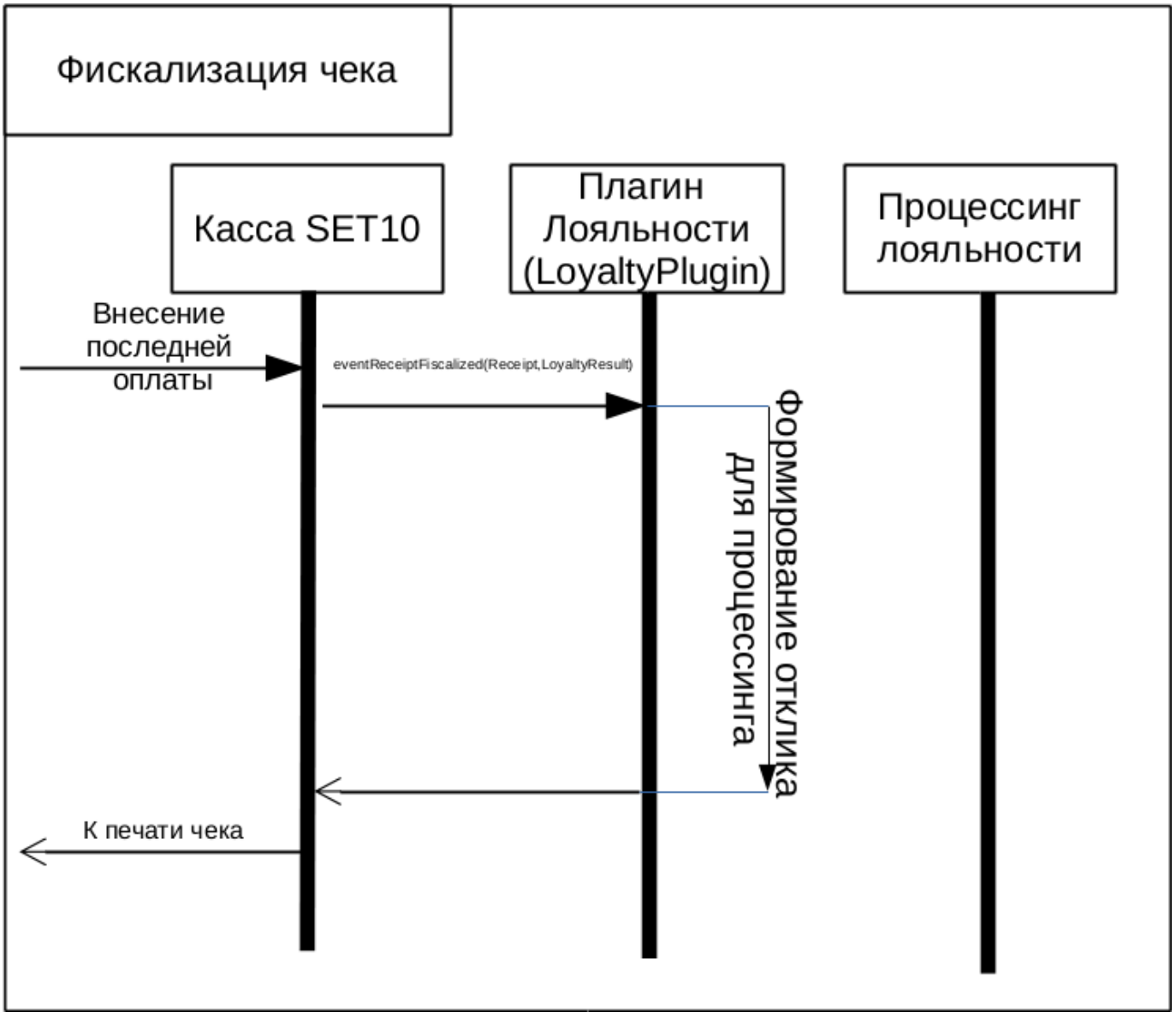

Рисунок 5 - последовательность взаимодействий кассы с плагином лояльности при фискализации чека

197136, Санкт-Петербург, Чкаловский пр., д.50, литера А, пом.5-Н, 2 этаж.
+7 (812) 331-22-55, crystals@crystals.ru
115432 Москва, пр. Андропова, 18, корп. 5, бизнес-парк Nagatino i-Land
+7 (495) 640-63-07, moscow@crystals.ru
8 (800) 222-22-51, www.crystals.ru

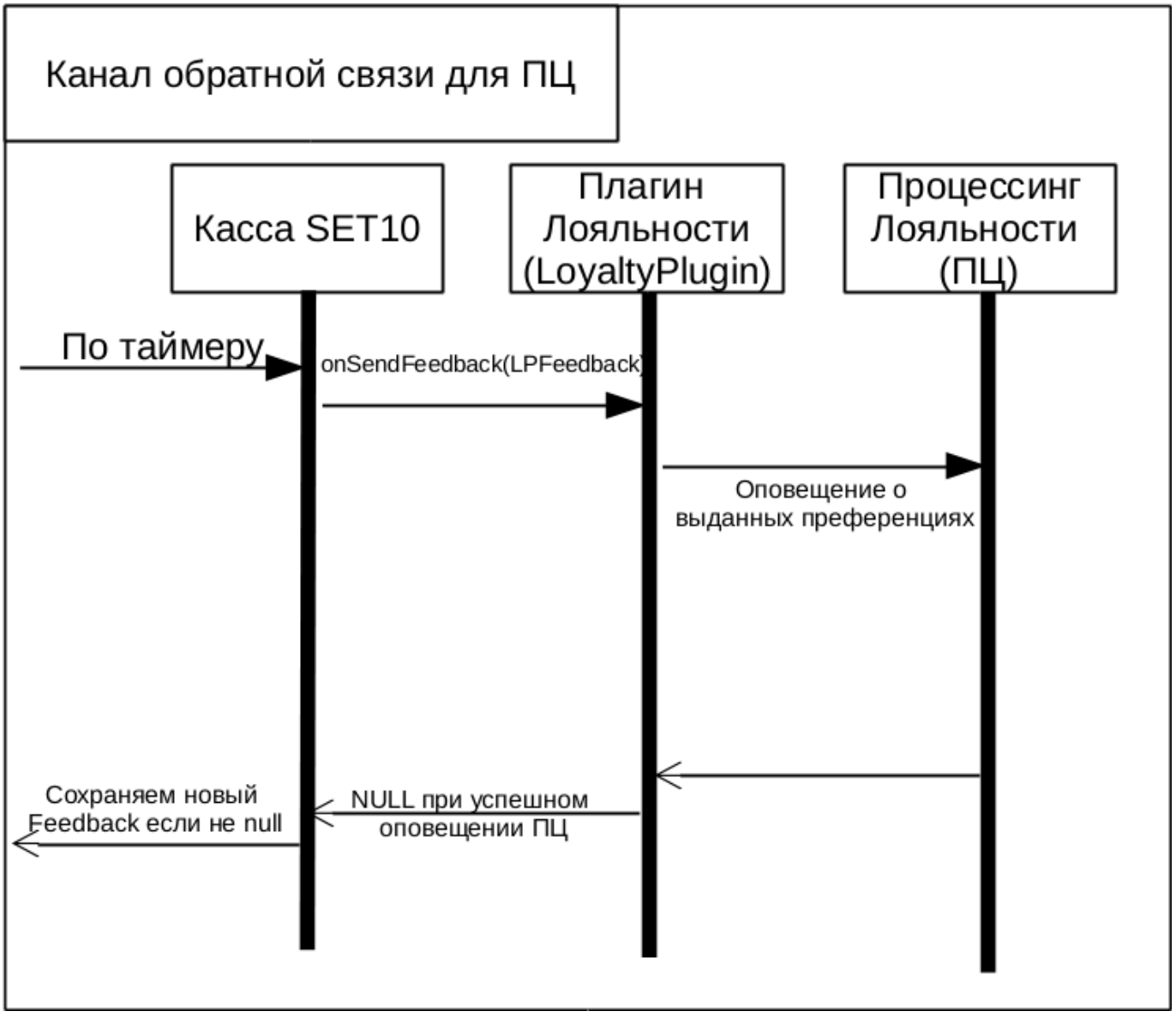

Рисунок 6 - последовательность взаимодействия кассы с плагином лояльности после фискализации чека

С точки зрения разработки ПО, плагин лояльности - это реализация интерфейса *ru.crystals.pos.api.plugin.LoyaltyPlugin*; пример реализации поставляется с *SDK*[5] - в проекте ***LoyaltyPluginExample***.

[5] т.е. в артефакте ***set10pos-api-x.x.x.jar***

197136, Санкт-Петербург, Чкаловский пр., д.50, литера А, пом.5-Н, 2 этаж.
+7 (812) 331-22-55, crystals@crystals.ru
115432 Москва, пр. Андропова, 18, корп. 5, бизнес-парк Nagatino i-Land
+7 (495) 640-63-07, moscow@crystals.ru
8 (800) 222-22-51, www.crystals.ru

5. Создание плагина карт

Плагины карт предназначены для применения на кассе карт лояльности или купонов, информация о которых хранится в внешней по отношению к *Set Retail 10* системе. Карта лояльности, в общем случае, это физическая или виртуальная карта, которая выдаётся покупателю и применение которой на кассе обеспечивают ему какие-либо бенефиты, в виде скидки, например. Карта также может содержать бонусный баланс, который допустимо тратить на кассе, образуя скидку. По техпроцессу кассы добавленные в чек карты могут быть использованы:

- плагинами лояльности как триггер срабатывания расчета скидок в нем или предоставления возможности списывать бонусы;
- для идентификации покупателя. После совершения покупки информация о чеке, включая примененную карту, может быть выгружена в внешнюю *ERP*-систему и в дальнейшем использована для различных *CRM*-инициатив, например для предоставления дополнительных преференций при дальнейших покупках.

Процесс поиска и добавления карты в чек может быть запущен одним из следующих способов:

- ввод номера карты на кассе вручную;
- прокатыванием магнитной полосы карты через считывающее устройство (*MSR*);
- сканирование штрих-кода, нанесенного на карту;
- считыванием *NFC*-метки карты;
- определение номера карты по номеру телефона, принадлежащего держателю карты;
- определение номера карты по адресу электронной почты держателя (доступно не на всех типах касс).

Считанная информация последовательно передаётся плагинам карт до тех пор, пока один из них не признает свою карту и не ответит информацией по ней, либо о том, что карты не существует. С учетом того, что плагинов карт может быть несколько, а поиск карты во внешней системе занимать продолжительное время, рекомендуется иметь возможность не отсылая запрос в процессинг определить, принадлежит ли карта плагину. Например, используя комбинацию префикса карты и длины её номера, и только в случае, если префикс и длина удовлетворяют требованиям, приступать к поиску карты в внешней системе. Также рекомендуется немедленно возвращать управление, не признавая карту, в случае, если плагинные карты не могут быть добавлены на кассу переданным способом ввода. Например, плагинные карты могут быть добавлены только сканированием магнитной полосы, а в плагин приходит карта, введенная руками или считанная при помощи сканера баркодов. Таким образом, для идентификации карты плагину необходимо знать её номер и способ ввода или номер телефона держателя, за которым закреплена карта.

Упрощенный автомат добавления карт лояльности в чек представлен на рисунке 7. Пример реализации плагина карт поставляется с *SDK* - в проекте ***CardPluginExample***.

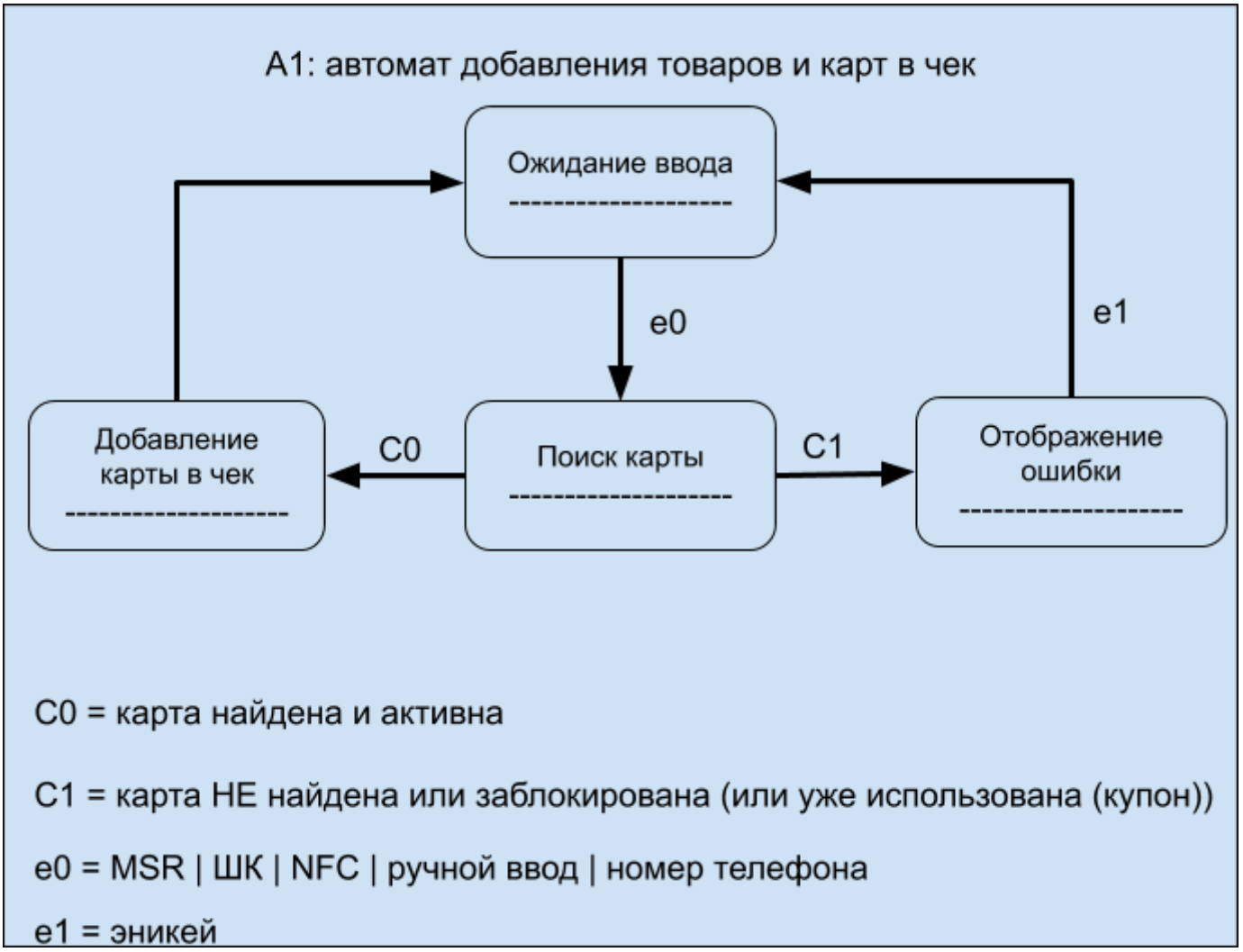

Рисунок 7 - упрощенная схема автомата добавления карт лояльности в чек

5.1. Взаимодействие плагина карт и плагина лояльности: списание бонусов как скидки

Плагин лояльности при расчете скидок может вернуть информацию о потолке списания бонусов как скидки с бонусной карты в текущем чеке. Далее данную информацию можно использовать при списании бонусов.

Упрощенно процесс списания бонусов состоит из следующих шагов:

1. кассир добавляет товары в чек;
2. добавляет карту - идет запрос в плагин карт;
3. добавляет остальные товары в чек;
4. запускает процесс расчета скидок (например, нажав клавишу “Подытог”) - идет запрос в плагин лояльности;
5. плагин лояльности кроме предоставления скидок возвращает также информацию о потолке списания бонусов с карты в текущем чеке;
6. после расчета скидок касса переходит в состояние ожидания внесения оплат;
7. до внесения первой оплаты [и при наличии в чеке бонусной карты, с которой можно списать бонусы] кассир может запустить сценарий списания бонусов (например, выбрав

197136, Санкт-Петербург, Чкаловский пр., д.50, литера А, пом.5-Н, 2 этаж.
+7 (812) 331-22-55, crystals@crystals.ru
115432 Москва, пр. Андропова, 18, корп. 5, бизнес-парк Nagatino i-Land
+7 (495) 640-63-07, moscow@crystals.ru
8 (800) 222-22-51, www.crystals.ru

соответствующий пункт меню и введя в диалоговом окне количество бонусов для списания) - идет запрос в плагин карт;

a. в случае успешного списания бонусов автоматически запускается процесс пересчета скидок - плагин лояльности может предоставить дополнительные скидки на основании данных о списании бонусов (т.е., “превратить” списание бонусов в скидку);
    i. если плагин лояльности не “использует” всю сумму списанных бонусов для предоставления скидки, автоматически запустится процесс отмены списания бонусов[6] и пересчета скидок; кассиру при этом будет показано сообщение об этой ошибке;
b. в случае критичной[7] ошибки при попытке списания кассиру отображается сообщение об ошибке, при закрытии которого касса автоматически переходит в состояние внесения оплат; процесс пересчета скидок не запускается;
c. в случае не критичной ошибки кассиру отображается диалог с возможностью выбора: отказаться от списания или повторить попытку; при отказе от списания касса автоматически переходит в состояние внесения оплат (процесс пересчета скидок не запускается);

8. процесс списания бонусов завершен: далее следуют этапы кассового техпроцесса внесения оплат и фискализации чека.

Внимание! С одной карты допускается только одно списание бонусов в чеке (1 чек - максимум одна [не отмененная] транзакция списания бонусов).

Внимание! Если кассир переведет кассу из состояния внесения оплат в состояние добавления товаров в чек (например, нажав клавишу “Отмена” [подытога]), то все транзакции списания бонусов будут отменены: процесс списания придется повторить.

Внимание! Номер чека, получаемый плагином карт, может отличаться от номера, который чек получит при фискализации в случае, если после расчета скидок печатались иные документы, вроде *X*-отчета или копии чека, например.

Упрощенно процесс списания бонусов можно представить рисунком 8.

[6] запустится процесс в “теневом” потоке (*background thread*), что будет периодически отправлять в плагин карт запросы на отмену списания до тех пор, пока запрос не будет успешно обработан (или до тех пор, пока плагин карт не ответит на запрос ошибкой, указывающей на то, что повторять запрос более не следует).

[7] здесь под критичной ошибкой понимаем ошибку, что не допускает попыток повторения запроса: все последующие запросы тоже будут безуспешными; под не критичной ошибкой понимаем ошибку, что может не повториться при следующей попытке повторения запроса.

197136, Санкт-Петербург, Чкаловский пр., д.50, литера А, пом.5-Н, 2 этаж.
+7 (812) 331-22-55, crystals@crystals.ru
115432 Москва, пр. Андропова, 18, корп. 5, бизнес-парк Nagatino i-Land
+7 (495) 640-63-07, moscow@crystals.ru
8 (800) 222-22-51, www.crystals.ru

Рисунок 8 - схема автомата списания бонусов

6. Создание плагина оплат

Процесс оплаты чека подразумевает взаимодействие кассира с покупателем и кассой - необходимо выбрать способ оплаты и сумму. Один чек может быть оплачен несколькими типами платежей, например наличными и банковской картой. Каждый тип оплаты имеет свой уникальный процесс. Например: считывание банковской карты, обращение в банковский процессинг, печать слипов после печати чека. Этот процесс определяется внутри плагина.

Кассовое ПО после выбора способа оплаты передает управление плагину и предоставляет плагину возможность взаимодействовать с пользовательским интерфейсом и оборудованием кассы на высоком уровне абстракции до завершения процесса.

Процесс оплаты может быть завершен успешно или неуспешно. При успешном завершении касса сохраняет в базу данных информацию об оплате переданную плагином. Эта информация может быть использована для:

- отмены оплаты при аннулировании чека или возврате приобретенных товаров;
- поиска и просмотра чека;
- экспорта данных чека в *ERP* системы магазина.

197136, Санкт-Петербург, Чкаловский пр., д.50, литера А, пом.5-Н, 2 этаж.
+7 (812) 331-22-55, crystals@crystals.ru
115432 Москва, пр. Андропова, 18, корп. 5, бизнес-парк Nagatino i-Land
+7 (495) 640-63-07, moscow@crystals.ru
8 (800) 222-22-51, www.crystals.ru

Далее рассматривается пример создания плагина оплат. Исходный код примера находится в папке ***PaymentPluginExample***.

Для начала работы необходимо создать новый проект в *IDE*. Подключить *gradle* зависимости, как в примере, и создать новый класс имплементирующий интерфейс ***PaymentPlugin***. Класс обязательно должен иметь конструктор по умолчанию, без аргументов. Экземпляр класса создается один раз при старте кассы, т.е. он является синглтоном. Каркас плагина оплат изображен в коде 1.

```
package foo.payment.plugin;

import ru.crystals.pos.api.plugin.PaymentPlugin;
import ru.crystals.pos.spi.plugin.payment.CancelRequest;
import ru.crystals.pos.spi.plugin.payment.PaymentRequest;
import ru.crystals.pos.spi.plugin.payment.RefundRequest;

public class FooPaymentPlugin implements PaymentPlugin {

    @Override
    public void doPayment(PaymentRequest paymentRequest) {
    }

    @Override
    public void doPaymentCancel(CancelRequest cancelRequest) {
    }

    @Override
    public void doRefund(RefundRequest refundRequest) {
    }

    @Override
    public boolean isAvailable() {
        return false;
    }
}
```

Код 1 - каркас плагина оплат

Классу необходимо добавить аннотацию (код 2):

```
@POSPlugin(id="foo.service.payment")
public class FooPaymentPlugin implements PaymentPlugin
```

Код 2 - аннотация *POSPlugin* должна быть добавлена классу чтобы он распознавался как плагин

Аннотация эта используется для динамического подключения плагина и его идентификации. Значение аргумента *id* должно соответствовать ***id*** указанному в описании ***metainf.xml***.

Важно! Всякий плагин, будь то плагин карт, лояльности или оплат, должен иметь свой уникальный идентификатор и быть помеченным аннотацией *POSPlugin*! Из этого также следует, что всякий плагин должен быть реализован в виде отдельного класса. Идентификатор в атрибуте *id* аннотации *POSPlugin* должен совпадать с идентификатором описания плагина в *metainf.xml.*

197136, Санкт-Петербург, Чкаловский пр., д.50, литера А, пом.5-Н, 2 этаж.
+7 (812) 331-22-55, crystals@crystals.ru
115432 Москва, пр. Андропова, 18, корп. 5, бизнес-парк Nagatino i-Land
+7 (495) 640-63-07, moscow@crystals.ru
8 (800) 222-22-51, www.crystals.ru

Для подробностей об устройстве файла *metainf.xml* плагина, смотрите главу Файл манифеста плагина - metainf.xml настоящего руководства.

Метод ***isAvailable()*** нужен для того чтобы определить возможен ли выбор данного типа оплаты. В методе можно, например, выполнить проверку заполненности необходимых для работы настроек (код 3):

```
@Override
public boolean isAvailable() {
    return properties.getServiceProperties().get("foo.service.url") != null;
}
```

Код 3 - метод проверки доступности плагинов. Изображен пример доступности плагина на основании наличия настроек

NOTE: *настоятельно рекомендуется при возвращении негативного ответа методом* ***isAvailable()*** *залогировать (с уровнем не ниже INFO) причину, по которой данный тип оплаты невозможно использовать - это позволит быстрее выяснить и устранить корень проблемы.*

Для выполнения оплаты потребуется получение настроек, взаимодействие с пользовательским интерфейсом кассы, чтение локализованных строк. Добавим всё необходимое для этого (код 4):

```
@POSPlugin(id="foo.service.payment")
public class FooPaymentPlugin implements PaymentPlugin {
    @Inject
    private POSInfo pos;
    @Inject
    private IntegrationProperties properties;

    @Inject
    private UIForms ui;

    @Inject
    private ResBundle res;
}
```

Код 4 - инъекции вспомогательных элементов *Set API* в плагин

Реализуем метод оплаты ***doPayment***. В аргументе ***paymentRequest*** содержится информация о текущем чеке, сумме к оплате и *callback* интерфейс для возврата результата.

Отобразим форму ввода суммы к оплате. Событие ***eventCancelled*** возникнет, если на данной форме кассир нажимает кнопку “Отмена”. В таком случае продолжение процесса невозможно, сообщим кассе, что оплата закончилась неудачно. Касса, получив уведомление о том, что процесс завершился неудачно, возвращается к выбору способов оплаты чека. См. код 5 для иллюстрации.

197136, Санкт-Петербург, Чкаловский пр., д.50, литера А, пом.5-Н, 2 этаж.
+7 (812) 331-22-55, crystals@crystals.ru
115432 Москва, пр. Андропова, 18, корп. 5, бизнес-парк Nagatino i-Land
+7 (495) 640-63-07, moscow@crystals.ru
8 (800) 222-22-51, www.crystals.ru

```
    private void inputSum(PaymentRequest paymentRequest, BigDecimal defaultSum) {
        // Показываем форму ввода суммы к оплате
        SumToPayFormParameters parameters = new SumToPayFormParameters(
res.getString("payment.name"), paymentRequest.getReceipt());
        parameters.setInputHint(res.getString("enter.sum.to.pay"));
        parameters.setDefaultSum(defaultSum);
        ui.getPaymentForms().showSumToPayForm(parameters, new SumToPayFormListener() {
            @Override
            public void eventCanceled() {
                // была нажата <Отмена> завершим процесс
                paymentRequest.getPaymentCallback().paymentNotCompleted();
            }
            @Override
            public void eventSumEntered(BigDecimal sumToPay) {
                sumEntered(paymentRequest, sumToPay);
            }
        });
    }
```

Код 5 - отображение приглашающего ввести сумму к оплате окна

В данном примере оплата происходит с некоего электронного кошелька, у которого есть номер, который можно ввести вручную или просканировать *QR* код. После ввода суммы отобразим форму ввода или сканирования и подпишемся на события этой формы. При получении событии о нажатии кнопки “Отмена”, покажем снова форму ввода суммы к оплате (код 6)

```
private void sumEntered(PaymentRequest paymentRequest, BigDecimal sumToPay) {
        ui.getInputForms().showInputScanNumberForm(res.getString("payment.name"),
res.getString("scanQR"), res.getString("wallet.number"), 16, new InputScanNumberFormListener() {
            @Override
            public void eventBarcodeScanned(String barcode) {
                remoteRequest(paymentRequest, barcode, sumToPay);
            }
            @Override
            public void eventNumberEntered(String number) {
                remoteRequest(paymentRequest, number, sumToPay);
            }
            @Override
            public void eventCanceled() {
                // если нажали отмену, возвращаемся в форме ввода суммы к оплате
                inputSum(paymentRequest, sumToPay);
            }
        });
    }
```

Код 6 - обработка ввода пользователем суммы к оплате

Когда введен номер кошелька или отсканирован его *QR* код, можно начинать процесс оплаты. Добавим метод ***remoteRequest*** для выполнения удаленного запроса на сервер. Поскольку этот процесс может быть длительным по времени, отобразим сразу форму спиннера. Когда отображается форма спиннера на экране кассы - весь ввод становится заблокированным, кассир никак не может прервать процесс (код 7).

197136, Санкт-Петербург, Чкаловский пр., д.50, литера А, пом.5-Н, 2 этаж.
+7 (812) 331-22-55, crystals@crystals.ru
115432 Москва, пр. Андропова, 18, корп. 5, бизнес-парк Nagatino i-Land
+7 (495) 640-63-07, moscow@crystals.ru
8 (800) 222-22-51, www.crystals.ru

```
private void remoteRequest(PaymentRequest paymentRequest, String walletNumber, BigDecimal
sumToPay) {
        ui.showSpinnerForm(res.getString("connecting"));
        // читаем необходимые настройки:
        String url = properties.getServiceProperties().get("foo.service.url");
        int timeout = properties.getServiceProperties().getInt("timeout", 10000);
        // читаем необходимые настройки кассы:
        int posNumber = pos.getPOSNumber();
        int shopNumber = pos.getShopNumber();
        // Получить информацию о кассире
        User currentCashier = pos.getUser();
        try {
            // выполнение удаленного вызова, получение ответа формирование данных для
callback
            Payment payment = new Payment();
            payment.setSum(sumToPay);
            payment.getData().put("authorization.code", "some code");
            payment.getData().put("client.id", "some client ID");
            // формируем слип для печати
            StringBuilder sb = new StringBuilder();
            sb.append(res.getString("slip.header"));
            sb.append("\n").append(sumToPay.toString());
            sb.append(res.getString("slip.rub")).append("\n");
            sb.append(res.getString("slip.transaction.number"));
            sb.append("01234567890");
            payment.getSlips().add(sb.toString());
            // Можно сохранять какие либо параметры необходимые для работы сервиса
            int persistCounter = properties.getServiceProperties().getInt("PersCounter", 0);
            persistCounter++;
            properties.getServiceProperties().setInt("PersCounter", persistCounter);
            // процесс оплаты успешно завершен
            paymentRequest.getPaymentCallback().paymentCompleted(payment);
            // Касса продолжает процесс оплаты и регистрации чека.
            // С этого момента отображение форм уже недопустимо.
            // Попытка отобразить форму после отправка callback приведет к исключению
            // IncorrectStateException
        } catch (Exception e) {
            ui.showErrorForm(res.getString("connection.error"), new ConfirmListener() {
                @Override
                public void eventConfirmed() {
                    paymentRequest.getPaymentCallback().paymentNotCompleted();
                }
            });
        }
    }
```

Код 7 - создание оплаты и распечатка чековых документов

Подобным образом реализуются методы отмены оплаты и возврата. В аргументах методов ***doPaymentCancel*** и ***doRefund*** содержится информация о ранее выполненных оплатах. Отмена оплаты и возврат для кассового процесса разные операции, хотя со стороны внешних процессингов это может быть одна операция. Отмена оплаты выполняется при аннулировании текущего чека, возврат выполняется когда покупатель возвращает приобретенный ранее товар.

Если есть необходимость сохранять какие-либо параметры, необходимые для работы плагина/сервиса, в *properties.getPluginProperties()* и *properties.getServiceProperties()* есть набор методов, позволяющих сохранять параметры в БД.

Пример реализации возврата смотрите в примере *Examples/PaymentPluginExample,* поставляемого *SDK*.

Возврат наличными из плагина оплаты

Плагину оплат допустимо не выполнять возвраты тем же способом оплаты, что и продажа. В этом случае, начиная с кассы версии 10.2.78.0, предусмотрена возможность производить возврат плагинной оплаты наличными. Для этого каждому плагину оплаты в настройки автоматически добавляется опция “выполнять возврат наличными” (рисунок 9).

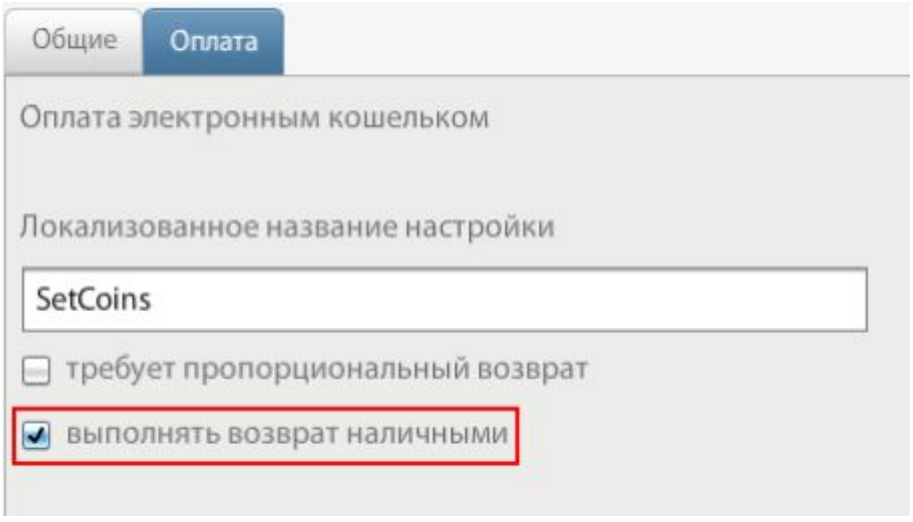

Рисунок 9 - опция плагина оплат “выполнять возврат наличным”

При включении данной опции, в случае возврата, взаимодействия с плагином не происходит и возврат оплаты выполняется наличными. Плагин может получить значение этой опции из своих параметров по ключу “*payment.config.refund.payment.as.cash*”.

7. Создание плагина валидации акцизных марок товаров

Касса *Set Retail 10* может быть настроена таким образом, что при продаже и возврате алкогольного товара, перед добавлением товара в чек, будет выполняться проверка отсканированной акцизной марки. Интеграцию с внешними сервисами валидации акцизных марок выполняет плагин валидации акцизных марок. Пример реализации смотрите в примере *Examples/ExciseValidationPluginExample*.

Плагин реализует следующие методы:

- *validateExciseForSale* (проверка возможности продажи);
- *validateExciseForRefund* (проверка возможности возврата);
- *eventReceiptFiscalized* (событие о фискализации чека);
- *onRepeatSend* (повторная фоновая отправка данных).

Подробное описание методов и их аргументов см. в *API*.

Поскольку взаимодействие данного типа плагина с визуальными формами на данный момент не предусмотрено, вызов *UIForms, UIInputForms, UIPaymentForms* не будет иметь эффекта.

197136, Санкт-Петербург, Чкаловский пр., д.50, литера А, пом.5-Н, 2 этаж.
+7 (812) 331-22-55, crystals@crystals.ru
115432 Москва, пр. Андропова, 18, корп. 5, бизнес-парк Nagatino i-Land
+7 (495) 640-63-07, moscow@crystals.ru
8 (800) 222-22-51, www.crystals.ru

8. Создание плагина типа товара

Продажа товаров и услуг на кассе *Set Retail 10* это процесс преобразования сущности товара в сущность позиции чека. Добавление в чек позиций различных типов товаров и услуг позволяет плагин типа товара `GoodsPlugin`. Способ взаимодействия кассы *Set10* с плагином типа товара:

1. Когда касса находится в режиме добавления товаров в чек продажи, при сканировании или ручном вводе штрихкода или артикула товара и, если товар не найден в базе данных кассы, вызывается метод плагина `MerchandiseEntity findByBarcode(String barcode)`. Плагин по неким правилам (префиксам / суффиксам) должен определить, принадлежит ли искомая строка поиска к товарам, продажу которых данный плагин осуществляет. Если нет, то плагин возвращает кассе `null`. Иначе возвращает новый экземпляр объекта найденного товара `MerchandiseEntity`, далее этот объект будет передан плагину для осуществления продажи.
2. Передача управления плагину для добавления позиции в чек выполняется вызовом метода `void addForSale(AddForSaleRequest addForSaleRequest)`. `AddForSaleRequest` содержит текущий чек, добавляемый товар (из п.1) и `AddForSaleCallback` - интерфейс для возврата управления кассе с двумя вариантами: `notCompleted()` и `completed(NewLineItem newLineItem)`.
3. При удалении уже добавленной позиции в чек вызывается метод плагина `void removeFromSale(RemoveFromSaleRequest request);`

Дополнительно можно реализовать методы `eventReceiptFiscalized` и `onRepeatSend` для получения событий о фискализации и аннулирования чека, с целью гарантированного уведомления удаленных процессингов.

Пример реализации смотрите в примере *Examples/GoodsPluginExample*.

9. Создание плагина условия рекламной акции

Плагины условия рекламной акции позволяют внешней системе в некоторой степени управлять рекламными механиками *Set Retail 10*, разрешая или запрещая какой-либо акции участвовать в расчете скидок. Для этого такая рекламная акция в условиях своего участия должна содержать условие “Определяет внешняя система” (вкладка “Общие правила”). Создание и настройка акции выполняется на сервере. Реализацию этого условия и представляет рассматриваемый плагин. При проверке условия участия рекламной акции, плагину на вход поступает информация о заведенной в *Set Retail 10* рекламной акции, которой управляет плагин и чек, по которому в настоящий момент считаются скидки. На основе этих данных плагину предлагается вынести бинарное суждение о допускать рекламную акцию к участию в расчете скидок или нет.

Плагины условия применения рекламной акции должны реализовывать интерфейс *ru.crystals.pos.api.plugin.AdvertisingActionConditionPlugin* и процесс создания их аналогичен процессу создания иных плагинов *Set API*.

10. Создание плагина тех. процесса

Плагины тех. процесса позволяют внешней системе расширять поведение кассы, и в первую очередь интерактивное взаимодействие с кассиром. Касса умеет прерывать некоторое подмножество

197136, Санкт-Петербург, Чкаловский пр., д.50, литера А, пом.5-Н, 2 этаж.
+7 (812) 331-22-55, crystals@crystals.ru
115432 Москва, пр. Андропова, 18, корп. 5, бизнес-парк Nagatino i-Land
+7 (495) 640-63-07, moscow@crystals.ru
8 (800) 222-22-51, www.crystals.ru

фаз тех. процесса и передавать управление плагину. Также как и другие плагины, данный плагин может иметь произвольные настройки.

Плагины условия применения рекламной акции должны реализовывать интерфейс *ru.crystals.pos.api.plugin.TechProcessPlugin* и процесс создания их аналогичен процессу создания иных плагинов *Set API*. Пример реализации смотрите в примере *Examples/TechProcessPluginExample.*

11. Слежение за кассовыми событиями

Плагины *Set API* могут следить за кассовыми событиями, такими как открытие или закрытие смены. Для этого предусмотрен механизм слушателей. Чтобы создать слушатель определенного события, необходимо создать класс, реализующий интерфейс этого слушателя (интерфейсы предоставляет *Set API*) и сделать его полем класса плагина, пометив аннотацией *ru.crystals.pos.spi.annotation.POSEventListener*. В настоящий момент доступен только один слушатель - слушатель событий кассовой смены (*ru.crystals.pos.api.events.ShiftEventListener*). Он позволяет отлавливать событие “Открыта кассовая смена”, “Закрыта кассовая смена”. Список событий, которые получает данный слушатель, приведены в таблице 1.

12. Создание плагина поставщика рекламных акций

Плагины-поставщики рекламных акций позволяют передать кассе набор рекламных акций в формате действующего импорта акций из внешней системы на сервере, которые затем будут добавлены к уже имеющимся на кассе акциям и будут участвовать в расчете лояльности для текущего чека. Большинство ограничений серверного импорта, касающихся условий и результатов РА, также актуальны и для акций плагинов поставщиков. Заполнение большинства основных полей акции (даты действия, зона охвата, приоритет и т.д.) является необязательным. Обязательным является заполнение лишь поля external-code. В случае совпадения external-code либо guid акции с соответствующим значением имеющейся на кассе акции, акция плагина заменяет соответствующую акцию на время расчета данного чека.

Плагины условия применения рекламной акции должны реализовывать интерфейс *ru.crystals.pos.api.plugin.AdvertisingActionProviderPlugin* и процесс создания их аналогичен процессу создания иных плагинов *Set API*.

197136, Санкт-Петербург, Чкаловский пр., д.50, литера А, пом.5-Н, 2 этаж.
+7 (812) 331-22-55, crystals@crystals.ru
115432 Москва, пр. Андропова, 18, корп. 5, бизнес-парк Nagatino i-Land
+7 (495) 640-63-07, moscow@crystals.ru
8 (800) 222-22-51, www.crystals.ru

Таблица 1 - события слушателя *ShiftEventListener*

Событие	**Метод**	**Когда срабатывает**
Открыта новая кассовая смена	*onShiftOpened*	При открытии новой кассовой смены, в аргументах передаётся новый номер смены. Этому событию не обязательно предшествует событие закрытия смены, потому что касса могла быть перезагружена, а значит плагин был тоже.
Кассовая смена закрыта	*onShiftClosed*	При закрытии кассовой смены. Данный метод не принимает аргументов. Этому событию не обязательно предшествует событие открытия смены по тем же причинам, что и для события открытия.

Таким образом, для создания слушателя событий кассовой смены, необходимо:

1. Создать в плагине класс, реализующий интерфейс *ShiftEventListener.*
2. Сделать экземпляр этого класса полем плагина (того класса, который помечен аннотацией *ru.crystals.pos.spi.annotation.POSPlugin*).
3. Пометить полученное поле аннотацией *ru.crystals.pos.spi.annotation.POSEventListener*.

Пример подключения в плагин товаров слушателя событий кассовой смены приведен в коде 8. Упоминаемый в коде *ShiftEventListenerImpl* предполагается реализовать разработчику плагина самостоятельно.

197136, Санкт-Петербург, Чкаловский пр., д.50, литера А, пом.5-Н, 2 этаж.
+7 (812) 331-22-55, crystals@crystals.ru
115432 Москва, пр. Андропова, 18, корп. 5, бизнес-парк Nagatino i-Land
+7 (495) 640-63-07, moscow@crystals.ru
8 (800) 222-22-51, www.crystals.ru

```
@POSPlugin(id="my.awesome.goods.plugin")
public class GoodsPluginExample implements GoodsPlugin {

	@POSEventListener
	private ShiftEventListener shiftEventListener;
	/**
	* Инициализация плагина.<br>
	* Метод вызывается один раз, после создания экземпляра класса, заполнения всех необходимых
инъекций, перед первым обращением к плагину.<br>
	*/
	@PostConstruct
	public void init() {
	// В отличии от полей, помеченных аннотацией Inject, за инициализацию и логику работы
	// слушателей кассовых событий отвечает плагин. Посему в нем и реализован интерфейс и
инициализация.
	shiftEventListener = new ShiftEventListenerImpl();
	}

	@Override
	public void addForSale(AddForSaleRequest request) {
	request.getCallback().notCompleted();
	}

	@Override
	public void removeFromSale(RemoveFromSaleRequest request) {
	request.getCallback().notCompleted();
	}

	@Override
	public MerchandiseEntity findByBarcode(String barcode) {
	return null;
	}
}
```

Код 8 - пример подключения слушателя событий кассовой смены в плагин товаров.

13. Сверка итогов в плагинах

Сверка итогов это процесс получения из внешней системы информации о совершенных за день продажах и печать его для ручной сверки. В плагинах *SetAPI* предусмотрена возможность печати сверки итогов по кнопке или при закрытии смены. Сверку итогов поддерживают плагины оплат, товаров, а также любые плагины, реализующие интерфейс *ru.crystals.pos.spi.plugin.ReconciliationReportMaker.*

197136, Санкт-Петербург, Чкаловский пр., д.50, литера А, пом.5-Н, 2 этаж.
+7 (812) 331-22-55, crystals@crystals.ru
115432 Москва, пр. Андропова, 18, корп. 5, бизнес-парк Nagatino i-Land
+7 (495) 640-63-07, moscow@crystals.ru
8 (800) 222-22-51, www.crystals.ru

14. Печать документов

Плагинам доступно уведомлять кассу о желании ими напечатать дополнительный документ, прилагаемый к чеку (слип). Делается это путём передачи в качестве результатов работы плагина структуры *ru.crystals.pos.api.ext.loyal.dto.Slip* там где это возможно. Данная структура представляет собой описание текстового документа и помимо текста может содержать баркоды, *QR*, изображения и форматированный текст.

Слип содержит в себе список параграфов. Каждый параграф описывает печатаемую на документе строку. Если окажется, что строка содержит переносы или не помещается в одну строку чековой ленты, фискальный принтер может разделить такой параграф на несколько строк. Данные, которые параграф содержит, описываются его типом. Параграф может быть текстовым, содержать штрих-код, *QR* или изображение. Комбинирование типов в рамках одного параграфа не допускается, т.е если параграф является типом “изображение”, его текстовое содержимое игнорируется. См. структуру *ru.crystals.pos.api.ext.loyal.dto.SlipParagraphType*, описывающую типы параграфа.

В текстовых параграфах допустима печать текста различными шрифтами, для этого в начале текста параграфа необходимо указать ключевой символ *SlipParagraph#FORMATTED_TEXT_PREFIX*, по которому касса определяет, что текст является форматированным и выполняет разбор. Касса не оперирует понятиями атрибутов шрифта (жирный, курсив, etc.), но оперирует понятием атласа шрифта, т.е таблицы ключ-значения, где ключом служит код шрифта, а значением - шрифт, которым будет печататься текст. Для смены шрифта в тексте следует использовать управляющую последовательность “[*x*]”, где *x* - код шрифта в атласе. Для печати символов “[“ и “]” в таком форматированном тексте необходимо экранирование символом “\\”. Пример форматирования текста в параграфе представлен в коде 9. Доступные шрифты перечислены в таблице 2.

```
\\f[18]Hello [2]world!
```

Код 9 - пример форматирования текста

197136, Санкт-Петербург, Чкаловский пр., д.50, литера А, пом.5-Н, 2 этаж.
+7 (812) 331-22-55, crystals@crystals.ru
115432 Москва, пр. Андропова, 18, корп. 5, бизнес-парк Nagatino i-Land
+7 (495) 640-63-07, moscow@crystals.ru
8 (800) 222-22-51, www.crystals.ru

Таблица 2 - атлас шрифтов Пирит 2Ф

Код	Шрифт
0	13x24
1	10x20
2	13x24 жирный
3	10x20 жирный
4	8x14
5	24x45
6	24x45 жирный

К перечисленным кодам можно добавить атрибуты двойной высоты или ширины, установив соответственно бит 4 или 5. Для этого предлагается пользоваться константами из *ru.crystals.pos.api.ext.loyal.dto.SlipTextOptions*, выполняя побитовое ИЛИ над ними и кодом шрифта.
Для печати элементов текста различными шрифтами также можно воспользоваться классом *ru.crystals.pos.api.ext.loyal.dto.SlipText*, в котором можно задать элемент текста и код шрифта для него. Список таких объектов необходимо задать в *SlipParagraph#paragraphParts*.

Некоторые клиенты настраивают свои фискальные регистраторы печатать логотип магазина в заголовке каждого документа. Это поведение отключаемо. Если плагину требуется распечатать слип без обязательного логотипа, он может отключить его печать, используя параметр *Slip#disableLogo*.

Слипы допустимо печатать в составе чека. Для этого у структуры *Slip* предусмотрен параметр *separated*, который включает или отключает печать слипа в составе чека. При печати копии чека слипы повторно не печатаются, даже если они были включены в его состав изначально. Для включения слипа в состав чека в шаблоны печатных документов кассы должны быть внесены дополнительные секции “*includedSlips*” в местах, в которых будут печататься слипы, однако настройка шаблонов фискальных документов кассы не является частью *API* и выходит за рамки данного туториала. Это действие относится к этапу настройки кассы и не является этапом написания плагина.

На печатаемые в слипах изображения накладываются ограничения. Для чековой ленты шириной 80 мм размер изображения не должен превышать 512x512 точек, для чековой ленты шириной 57 мм - 336x400 точек. Длина и высота изображения должна быть кратна 8. Размер растра изображения, если принять, что пиксель описывается одним битом, не должна превышать 22 КБ для фискальных принтеров Пирит 2Ф, 11 Кб для Пирит 1Ф. Требования допустимо не соблюдать, отсылая, например полноцветное изображение с некратной восьми шириной и высотой, тогда касса предпримет попытку преобразовать изображение к допустимому формату, однако не гарантируется успешность такой конверсии и она отрицательно скажется на качестве изображения. Посему лучшим будет передавать на печать монохромные *bmp* с глубиной цвета 1 бит, удовлетворяющие представленным требованиям.

15. Управление интерфейсом кассы

Для манипулирования *UI* кассы в *Set API* предусмотрен класс *ru.crystals.pos.spi.ui.UIForms*, который позволяет плагину отображать визуальные формы, которые он может конфигурировать под

197136, Санкт-Петербург, Чкаловский пр., д.50, литера А, пом.5-Н, 2 этаж.
+7 (812) 331-22-55, crystals@crystals.ru
115432 Москва, пр. Андропова, 18, корп. 5, бизнес-парк Nagatino i-Land
+7 (495) 640-63-07, moscow@crystals.ru
8 (800) 222-22-51, www.crystals.ru

свои нужды. Не всем плагинам и не в каждом методе допустимо отображать кассовые формы, поэтому следует смотреть каждый метод индивидуально.
Условно *UIForms* делит кассовые формы на три типа: оплаты, ввода и прочие. Для доступа к формам оплаты служит метод *getPaymentForms*, для доступа к формам ввода – метод *getInputForms*. *UIForms* не должен создаваться вручную, а должен подключаться в плагин через аннотацию *@Inject*. См. примеры плагинов в *SDK*. Для переноса строк в тексте форм следует использовать тег *
*, для курсива - теги **.
Доступные плагинам кассовые формы будут рассмотрены ниже.

Форма ввода номера

Предназначена для ввода произвольных цифр при помощи клавиатуры или сканера штрихкодов. Установка на кассе этой формы осуществляется вызовом метода *UIForms#getInputForms#showInputNumberForm*. Допустимо скорректировать заголовок и поясняющий текст формы. Пример изображен на рисунке 10.

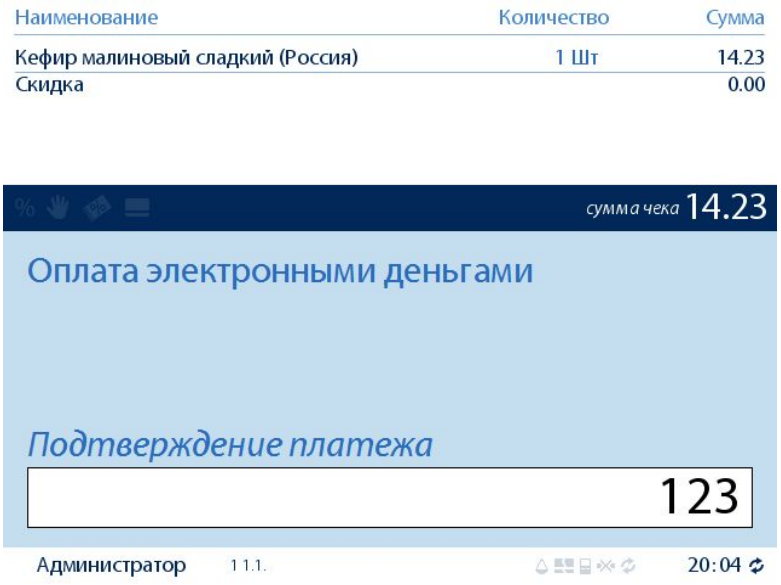

Рисунок 10 - пример формы ввода номера

Форма сканирования

Предназначено для сканирования *QR*, штрих-кодов или магнитных карт в случаях, когда ввод их руками недопустим. Вызывается как *UIForms#getInputForms#showInputScanNumberForm*. Внешний вид формы представлен на рисунке 11. При вызове допустимо менять заголовок и поясняющий текст формы.

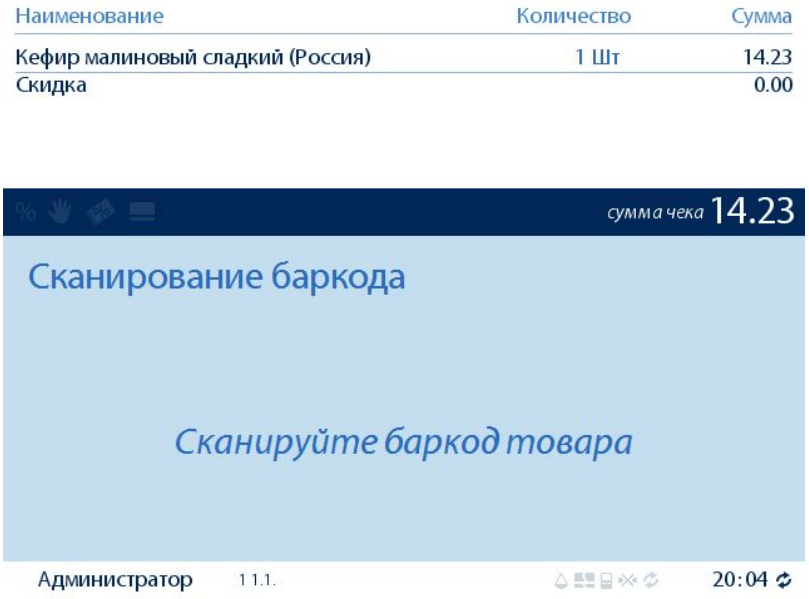

Рисунок 11 - форма сканирования

Форма ввода текста по шаблону

Основной сценарий использования такой формы - ввод номера мобильного телефона в заданном формате, однако технически фора предназначена для ввода любого типа текста в заданном формате. Описание задание формата приведено в прилагаемой к методу документации. Форма вызывается как *UIForms#getInputForms#showPatternInputForm*, внешний вид её представлен на рисунке 12.

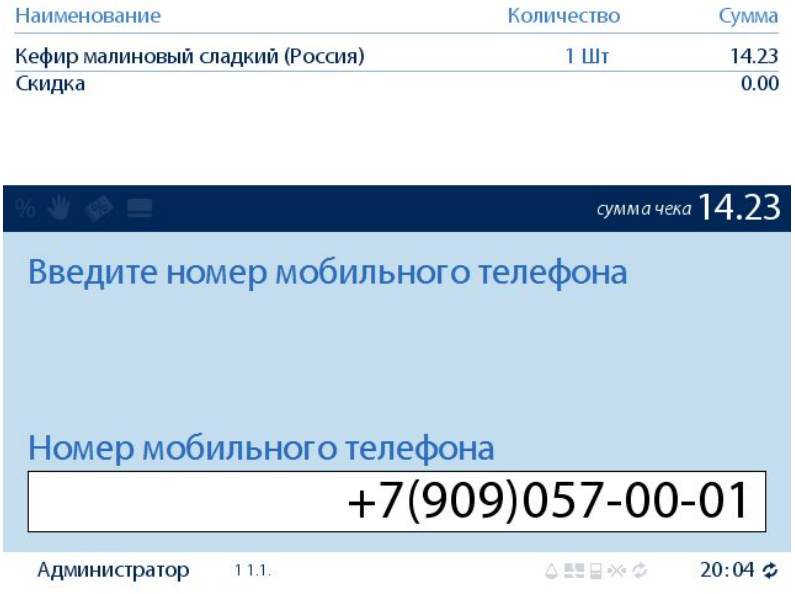

Рисунок 12 - форма ввода текста по шаблону

Форма оплаты

Используется преимущественно плагинами оплат, необходима для отображения и ввода суммы к оплате в принимаемых плагином платежных величинах. Касса сама отслеживает сколько к оплате было внесено, сколько следует доплатить. Вызов этой формы осуществляется как *UIForms#getPaymentForms#showSumToPayForm*. Пример отображения формы показан на рисунке 13.

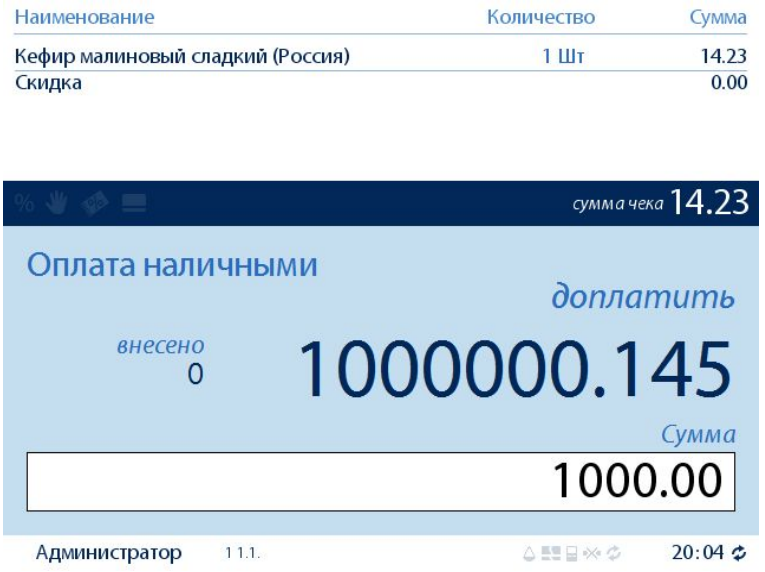

Рисунок 13 - форма оплаты

Форма диалога

Используется для предоставления кассиру тернарного выбора: согласиться с левой кнопкой, согласиться с правой кнопкой, отказаться от принятия решения совсем, нажав кнопку “Отмена” на клавиатуре. Вызывается форма методом *UIForms#showDialogForm*. Пример формы показан на рисунке 14.

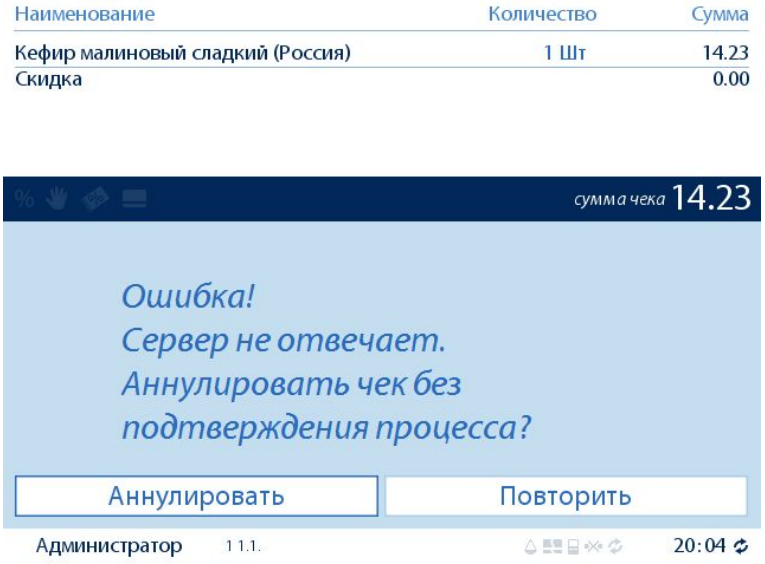

Рисунок 14 - форма диалога

197136, Санкт-Петербург, Чкаловский пр., д.50, литера А, пом.5-Н, 2 этаж.
+7 (812) 331-22-55, crystals@crystals.ru
115432 Москва, пр. Андропова, 18, корп. 5, бизнес-парк Nagatino i-Land
+7 (495) 640-63-07, moscow@crystals.ru
8 (800) 222-22-51, www.crystals.ru

Форма сообщения

Отображает информационное сообщение. От окна отображения ошибки отличается отсутствием слева иконки с восклицательным знаком. Вызывается как *UIForms#showMessageForm*. Внешний вид формы представлен на рисунке 15.

Рисунок 15 - форма сообщения

Форма сообщения об ошибке

Используется для вывода сообщений об ошибках. От окна сообщения это окно отличает наличие слева иконки с восклицательным знаком. Вызов формы осуществляется как *UIForms#showErrorForm*.

Пример формы приведен на рисунке 16, на котором также продемонстрирована максимальный объём текста, способный целиком уместиться на форме.

Рисунок 16 - пример формы сообщения об ошибке

Форма ожидания

Используется чтобы проинформировать кассира о выполнении длительной операции и занять интерфейс кассы, чтобы блокировать выполнение иных действий, пока текущее не завершится. Существует в отменяемом и неотменяемом исполнении: *UIForms#showSpinnerFormWithCancel* и *UIForms#showSpinnerForm* соответственно. Внешне же эти формы не отличаются. Пример представлен на рисунке 17.

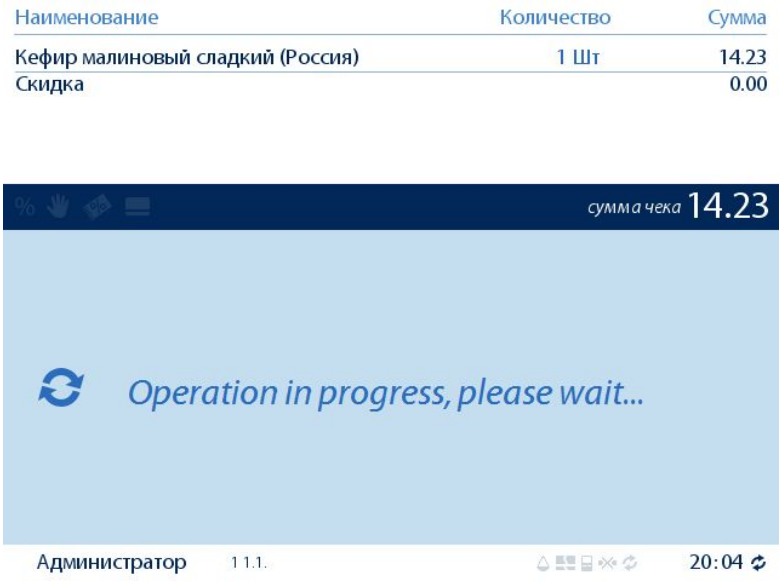

Рисунок 17 - форма ожидания

Форма с таймером

Назначение схоже с формой ожидания, однако позволяет отображать время выполнения какой-либо операции. На этой форме располагается счетчик обратного отсчета, значение которого декрементируется каждую секунду. Как только оно станет равно нулю, плагин получит уведомление. Сценарий использования данной формы - дать кассиру знать, как долго (в худшем случае) следует ожидать выполнения длительной операции. Сделано это потому что кассиры впадают в панику при сколько-нибудь длительном ожидании от кассы действий и воспринимают такое поведение как зависание кассы. Форма вызывается как *UIForms#showTimingOutForm*, внешний вид её представлен на рисунке 18.

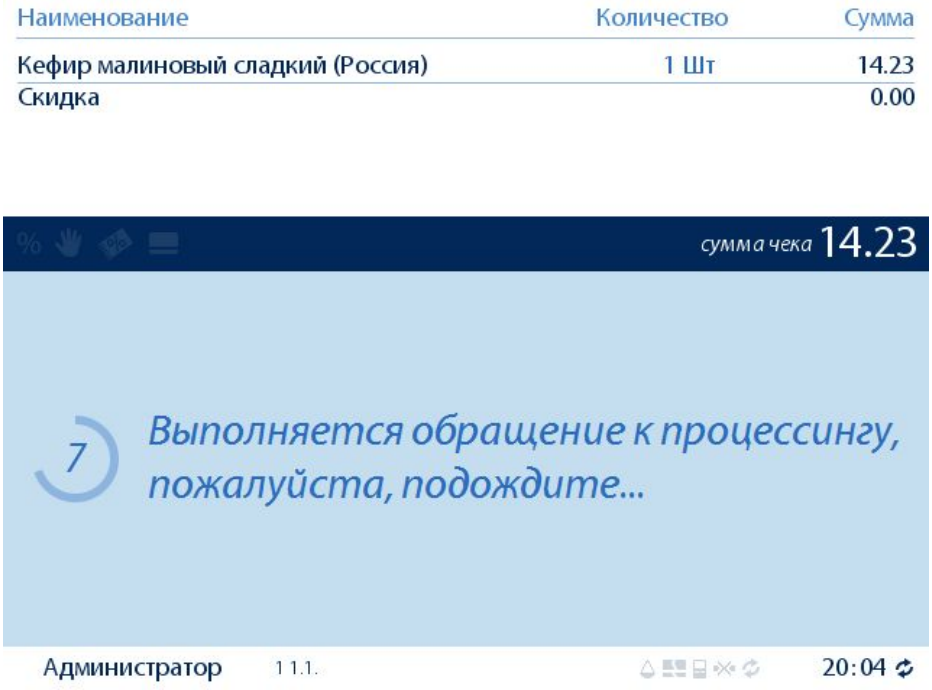

Рисунок 18 - форма с обратным отсчетом

Форма выбора из вариантов

Назначение формы - интерактивное взаимодействие с пользователем. Позволяет сделать выбор из предложенных вариантов. Поле выбора представляет собой таблицу. Каждая строка выбора - это текстовый список. Количество столбцов в таблице равняется размеру самого длинного списка. Форма вызывается как *UIInputForms#showSelectionForm*, внешний вид её представлен на рисунке 19.

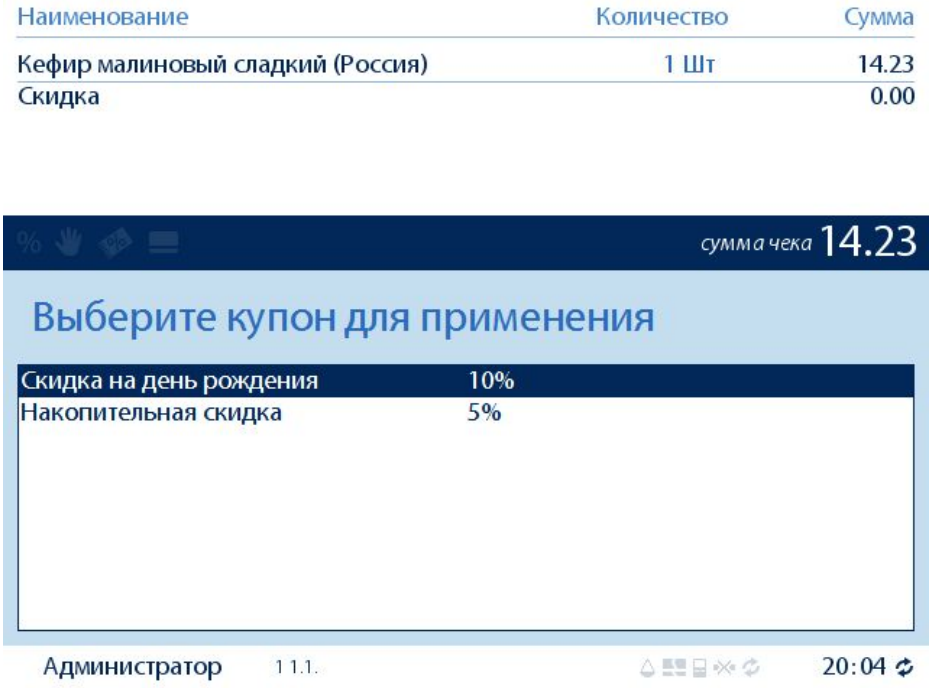

Рисунок 19 - форма выбора из вариантов

16. Файл манифеста плагина - *metainf.xml*

Файл манифеста необходим для регистрации плагина на сервере магазина и на кассе, для выполнения его настройки, для визуализации сохраненных данных и для экспорта этих данных в *ERP* системы магазина. Настройка плагинов всегда выполняется централизованно: на сервере центрального офиса сети магазинов или на каждом сервере магазина. Такие данные как: номер кассы, номер магазина плагин может узнать непосредственно при работе обращением к интерфейсу ***POSInfo***.

XSD схема файла манифеста ***metainf.xsd*** с описанием всех атрибутов находится в составе *SDK*.

Файл ***metainf.xml*** должен находится внутри *jar* файла плагина.

Интеграция с внешним сервисом может включать в себя несколько плагинов разных типов, плагины могут иметь в себе какие-то дополнительные настройки.

С точки зрения настройки *Set Retail 10*, настройки внешних систем и настройки плагинов оплат разделены логически и в визуализации сервера находятся в разных разделах.

Настройки внешнего взаимодействия, такие как: адреса сервера, таймаут, сертификат и пр. рекомендуется размещать в настройках внешнего сервиса *<ExternalService>*. Пример содержимого файла *metainf.xml,* в котором описан плагин оплат, изображен в коде 10. Данный пример файла находится в папке *PaymentPluginExample/src/main/resources*.

```
<?xml version="1.0" encoding="UTF-8"?>
<SetIntegration set10-api-version="0.0.6" version="1.0.0"
xmlns="http://crystals.ru/set10/api/metainf"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://crystals.ru/set10/api/metainf metainf.xsd ">
    <ExternalService serviceType="PAYMENT" localeKey="service.name" id="foo.service">
        <Options>
            <URL key="foo.service.url" localeKey="url.option" />
            <Integer key="timeout" localeKey="timeout.option" minValue="1000" maxValue="60000"/>
        </Options>
        <PaymentPlugin paymentType="ELECTRONIC" localeKey="payment.name"
id="foo.service.payment">
            <Description>Пример плагина оплат</Description>
            <Options>
                <String key="some.payment.plugin.property" localeKey="some.property.name"/>
            </Options>
            <PersistedField exportable="true" key="authorize.code" visible="true"
localeKey="autorise.field.name" />
        </PaymentPlugin>
    </ExternalService>
</SetIntegration>
```

Код 10 - пример файла *manifest.xml*, в котором описан плагин оплат

Для облегчения настройки плагина на сервере, параметры могут быть снабжены параметрами по умолчанию. Эти значения будут подставлены в интерфейс настройки плагина на сервере в случае, если это первая настройка и конфигурация плагина в БД сервера ещё сохранена не была. Значения параметров по умолчанию доступны для типов параметров *URL*, *String*, *Integer*, *Boolean* и недоступны для параметров типа *BinaryFile*. Следует отметить, это больше косметический функционал и плагин

197136, Санкт-Петербург, Чкаловский пр., д.50, литера А, пом.5-Н, 2 этаж.
+7 (812) 331-22-55, crystals@crystals.ru
115432 Москва, пр. Андропова, 18, корп. 5, бизнес-парк Nagatino i-Land
+7 (495) 640-63-07, moscow@crystals.ru
8 (800) 222-22-51, www.crystals.ru

никак больше не использует эти значения по умолчанию. Требования к содержимому у этих значений те же, что и к значению параметра. Пример задания параметров по умолчанию приведен в коде 11.

197136, Санкт-Петербург, Чкаловский пр., д.50, литера А, пом.5-Н, 2 этаж.
+7 (812) 331-22-55, crystals@crystals.ru
115432 Москва, пр. Андропова, 18, корп. 5, бизнес-парк Nagatino i-Land
+7 (495) 640-63-07, moscow@crystals.ru
8 (800) 222-22-51, www.crystals.ru

```
<?xml version="1.0" encoding="UTF-8"?>
<SetIntegration set10-api-version="0.0.5" version="1.0.0"
	xmlns="http://crystals.ru/set10/api/metainf"
	xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://crystals.ru/set10/api/metainf metainf.xsd ">
	<ExternalService serviceType="LOYALTY" localeKey="card.processing.name"
id="card.processing.example">
	<Options>
		<URL key="primary.get.info.url" localeKey="primary.get.info.url.description"
default-value="http://prima.ru"/>
		<URL key="secondary.get.info.url" localeKey="secondary.get.info.url.description" />
		<Integer key="connection.timeout" localeKey="connection.timeout.description"
minValue="1000" maxValue="60000" default-value="5000"/>
		<Integer key="read.timeout" localeKey="read.timeout.description" minValue="1000"
maxValue="60000" default-value="5000"/>
	</Options>
	<CardPlugin localeKey="service.name" id="example.card.provider">
		<Description>Тестовый плагин карт</Description>
		<Options>
		<String key="card.number.prefixes" localeKey="card.number.prefixes.description"
default-value="78"/>
		<Integer key="card.number.length" localeKey="card.number.length.description"
default-value="12"/>
		<String key="coupon.number.prefixes" localeKey="coupon.number.prefixes.description"
default-value="79"/>
		<Integer key="coupon.number.length" localeKey="coupon.number.length.description"
default-value="12"/>
		</Options>
	</CardPlugin>
	</ExternalService>
</SetIntegration>
```

Код 11 - пример описания карточного плагина, некоторые параметры которого снабжены значением по умолчанию

Атрибуты `default-value` в *xml* выше - значения по умолчанию для параметров. Эти атрибуты опциональны и могут быть пропущены.
Корневым элементом файла *metainf.xml* служит структура *SetIntegration*, которая является контейнером всех описаний плагинов. Атрибуты у этой структуры приведены в таблице 3.

Таблица 3 - описание атрибутов элемента *SetIntegration*

Атрибут	Обязательный	Описание	Пример
set10-api-version	да	Содержит минимальную версию *Set API*, с которой способны работать задекларированные в *metainf.xml* плагины. Его следует задавать той версией API, которая использовалась при создании плагина.	0.0.3
version	да	Содержит версию декларируемых в данном *metainf.xml* плагинов.	1.0.0

Ниже приводится описание дочерних элементов структуры *SetIntegration*, приведенное в порядке их декларирования.

ExternalService

ExternalService является структурой, которая содержит в себе декларацию плагинов. Допустимо в документе иметь одну или более таких структур. Помимо содержания плагинов эта структура несёт в себе информацию о категории плагина (электронные платежи, лояльность, бонусные системы). Эта информация используется для каталогизации плагинов на *UI* сервера. Атрибуты элемента приведены в таблице 4.

Таблица 4 - атрибуты элемента *ExternalService*

Атрибут	Обязательный	Описание	Пример
serviceType	да	Тип сервиса, который представляют хранящиеся в нем плагины. Определяет, в какую категорию внешних систем они будут отнесены при отображении на *UI* сервера. Возможные значения: *LOYALTY*, *PAYMENT*.	*LOYALTY*
localeKey	да	Ключ в файле локализации, которым задаётся человекочитаемое название внешней системы	*card.processing.name*
id	да	Уникальный идентификатор внешней системы.	*card.processing.id*

Внутри элемента *ExternalService* может находиться элемент *Description*, который несёт в себе человекочитаемое описание внешней системы. В настоящий момент оно используется исключительно как средство справочной информации и ограничено 3000 символами. Это поле позволительно оставить

197136, Санкт-Петербург, Чкаловский пр., д.50, литера А, пом.5-Н, 2 этаж.
+7 (812) 331-22-55, crystals@crystals.ru
115432 Москва, пр. Андропова, 18, корп. 5, бизнес-парк Nagatino i-Land
+7 (495) 640-63-07, moscow@crystals.ru
8 (800) 222-22-51, www.crystals.ru

пустым или не декларировать вообще. Далее следует описание опций внешней системы. Всякий плагин, относящийся к этой системе, может получить её настройки. Настройки находятся в элементе *Option*, который может содержать ноль или более элементов, приведенных в таблице 5. Если внешняя система не имеет настроек, секция с ними не декларируется.

Таблица 5 - типы элементов для задания параметров конфигурации внешней системы или плагина

Элемент	**Тип параметра**	**Описание**
Boolean	*Boolean*	Позволяет задать булевский параметр, возможные значение - *true*/*false*
BinaryFile	*byte[]*	Позволяет использовать произвольный файл в качестве параметра конфигурации плагина
Integer	*Integer*	Задает целочисленный параметр.
Decimal	*BigDecimal*	Задает численный параметр с плавающей точкой.
String	*String*	Задает строковый параметр.
URL	*String*, должен удовлетворять валидному *URL*	Задает параметр, отвечающий *URL*-схеме. Необходим для валидации ввода *URL* на странице конфигурации плагина. Рекомендуется использовать именно его, а не *String*, если параметр должен представлять собой ссылку на ресурс.

Все элементы имеют два обязательных и несколько опциональных атрибутов, представленных в таблице 6. Также элементы могут иметь специфичные только для себя значения, которые представлены в таблице 7. Не все типы параметров имеют специфичные атрибуты. Такие типы параметров в таблице 6 не перечислены.

197136, Санкт-Петербург, Чкаловский пр., д.50, литера А, пом.5-Н, 2 этаж.
+7 (812) 331-22-55, crystals@crystals.ru
115432 Москва, пр. Андропова, 18, корп. 5, бизнес-парк Nagatino i-Land
+7 (495) 640-63-07, moscow@crystals.ru
8 (800) 222-22-51, www.crystals.ru

Таблица 6 - атрибуты конфигурационного параметра внешней системы или плагина

Атрибут	Тип	Обязательный	Описание	Пример
key	*String*	да	Уникальный ключ, который идентифицирует параметр. По нему значение параметра может быть получено плагином.	*my.param.id*
localeKey	*String*	да	Ключ в файле локализации, указывающий на текст, которые следует отображать на UI при настройке.	*string.payment.success*
default-value	*String*	нет, не содержится в элементе типа *BinaryFile*	Значение по умолчанию параметра в случае, когда он впервые настраивается на сервере.	42
required	*Boolean*	нет	При настройке плагина через *UI* сервера определяет, разрешено ли оставлять поле пустым. Выставление этого параметра в *true* означает, настройку плагина нельзя завершить на *UI* сервера, не заполнив данный параметр. Значение по умолчанию - *false*, что означает, параметр допустимо оставлять пустым.	*true*

197136, Санкт-Петербург, Чкаловский пр., д.50, литера А, пом.5-Н, 2 этаж.
+7 (812) 331-22-55, crystals@crystals.ru
115432 Москва, пр. Андропова, 18, корп. 5, бизнес-парк Nagatino i-Land
+7 (495) 640-63-07, moscow@crystals.ru
8 (800) 222-22-51, www.crystals.ru

Таблица 7 - специфичные атрибуты определенных типов параметров

Тип параметра	**Атрибут**	**Тип**	**Обязательный**	**Описание**
Integer	*maxValue*	*Integer*	нет	Максимальное значение, которое может принимать этот параметр.
	minValue	*Integer*	нет	Минимальное значение, которое может принимать этот параметр. Должно быть меньше максимального значения.
String	*masked*	*Boolean*	нет	Булевский параметр, определяющий, следует ли скрывать от пользователя вводимые символы, заменяя их на «*» на визуализации.
	multiline	*Boolean*	нет	Определяет, следует ли отображать на *UI* сервера данный параметр как мультистрочное поле. *true* если следует, *false*, если достаточно отображать его одной строкой. Значение по умолчанию - *false*.

С точки зрения серверного *UI,* параметры внешней системы отображаются как на рисунке 20.

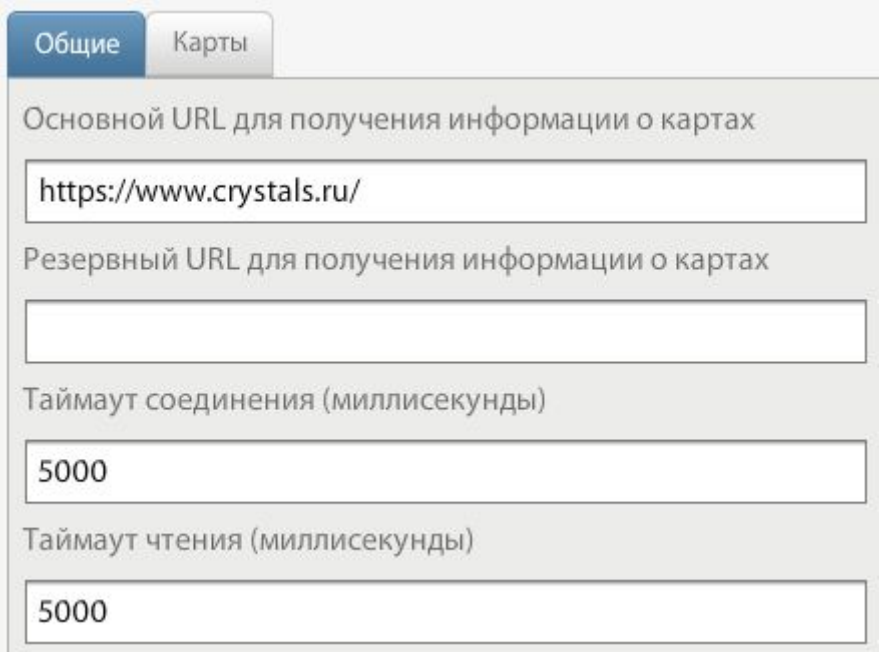

Рисунок 20 - визуализация настроек внешней системы на сервере

Рисунок 20 соответствует фрагменту кода из *manifest.xml*, изображенному в коде 12.

```
        <Options>
                <URL key="primary.get.info.url" localeKey="primary.get.info.url.description"
default-value="http://prima.ru"/>
                <URL key="secondary.get.info.url" localeKey="secondary.get.info.url.description" />
                <Integer key="connection.timeout" localeKey="connection.timeout.description" minValue="1000"
maxValue="60000" default-value="5000"/>
                <Integer key="read.timeout" localeKey="read.timeout.description" minValue="1000"
maxValue="60000" default-value="5000"/>
        </Options>
```

Код 12 - фрагмент декларации параметров внешней системы, визуализация которого проиллюстрирована на рисунке 20

Вслед за декларацией параметров внешней системы следует декларация плагинов, которая описывает свойства и настройки плагинов, подчиненных внешней системе. Допустимо декларировать произвольное количество плагинов любого типа. Всего типов плагина три: карточный процессинг, плагин лояльности и плагин оплат, каждый из которых будет рассмотрен ниже.

Плагин лояльности

Плагин лояльности задаётся элементом *LoyaltyPlugin,* фрагмент кода, описывающий декларацию плагина лояльности, представлен в коде 13.

```
<LoyaltyPlugin localeKey="service.name" id="ext.loy.processing.example">
              <Description>Тестовый плагин лояльности.</Description>
               <Options>
               <String key="alipay.service.ip" localeKey="ip.option" />
               <Integer key="timeout" localeKey="timeout.option" minValue="100" maxValue="60000"/>

               <Integer key="retrytimeout" localeKey="retrytimeout.option" minValue="1000"
maxValue="60000"/>
               <Integer key="retrycount" localeKey="retrycount.option" minValue="1" maxValue="10"/>
               <Integer key="port" localeKey="port.option" minValue="0" maxValue="99999"/>
               <String key="acceptor" localeKey="acceptor.option" />
               <String key="merchantid" localeKey="merchantid.option" />
               </Options>
       </LoyaltyPlugin>
```

Код 13 - пример декларации плагина лояльности

Декларация плагина лояльности практически ничем не отличается от декларации внешней системы, за исключением у первого отсутствует атрибут *serviceType*.

Плагин карт

Декларация плагина карт ничем не отличается от декларации плагина лояльности, за исключением названия элемента, который называется *CardPlugin* в случае плагина карт.

Плагин оплат

Плагины оплат имеют тот же способ задания настроек, что плагины карт и лояльности, однако, помимо настроек также могут декларировать заполняемые плагином в ходе работы параметры, которые затем могут выгружаться в *ERP* или отображаться в опердне. Плагин оплат определяется элементом *PaymentPlugin*, который, помимо атрибутов, присущих всем плагинам, имеет свой уникальный атрибут *paymentType*, который определяет тип оплат, с которыми работает плагин. Пример декларации плагина оплат изображен в коде 14. Возможные значения атрибута *paymentType* перечислены в таблице 8.

```
       <PaymentPlugin paymentType="ELECTRONIC" localeKey="payment.name" id="foo.service.payment">
              <Description>Плагин оплат</Description>
              <Options>
              <String key="some.payment.plugin.property" localeKey="some.property.name"/>
              </Options>
       </PaymentPlugin>
```

Код 14 - пример декларации плагина оплат в *metainf.xml*

Таблица 8 - допустимые значения атрибута *paymentType* элемента *PaymentPlugin*

Значение	Описание
CASH	Плагин принимает платежи наличными
ELECTRONIC	Плагин принимает платежи любым электронным суррогатом
PREPAY	Плагин принимает предоплату
POSTPAY	Плагин принимает постоплату
BONUS	Плагин принимает оплату бонусными баллами (в ФФД засчитывается как “электронный способ оплаты”)
OTHER	Плагин принимает вид платежа, не вошедший в вышеперечисленное

Пример выгрузки чека с оплатами в *ERP* изображен в коде 15. Поля, которые появились в выгрузке вследствие декларирования соответствующих полей в плагине оплат как выгружаемые, выделены красным.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<purchases count="1">
	<purchase tabNumber="1" userName="1 1 1" operationType="true" operDay="2018-04-09+03:00"
shop="6502" cash="1" shift="3" number="3" saletime="2018-04-09T12:46:34.875+03:00"
begintime="2018-04-09T12:45:08.751+03:00" amount="14.23" discountAmount="0.0" inn="123456987654">
	<positions>
		<position order="1" departNumber="1" goodsCode="00001" barCode="4600001000007" count="1.0"
cost="14.23" nds="18.0" ndsSum="2.17" discountValue="0.0" costWithDiscount="14.23" amount="14.23"
dateCommit="2018-04-09T12:45:08.761+03:00"/>
	</positions>
	<payments>
		<payment typeClass="foo.service.payment" amount="14.23" description="О п л а т а
э л е к т р о н н ы м  к о ш е л ь к о м">
		<plugin-property key="authorization.code" value="385281233141592"/>
		<plugin-property key="transaction.number" value="01234567890"/>
		</payment></payments></purchase></purchases>
```

Код 15 - пример выгрузки чека с оплатами в *ERP*. Красным отмечены поля, которые были отмечены как выгружаемые в *ERP* в файле *metainf.xml* плагина

Атрибуту *typeClass* в элементе *payment*, описывающим оплаты, соответствует идентификатор плагина, при работе которого эта оплата образовалась.

197136, Санкт-Петербург, Чкаловский пр., д.50, литера А, пом.5-Н, 2 этаж.
+7 (812) 331-22-55, crystals@crystals.ru
115432 Москва, пр. Андропова, 18, корп. 5, бизнес-парк Nagatino i-Land
+7 (495) 640-63-07, moscow@crystals.ru
8 (800) 222-22-51, www.crystals.ru

Плагин тех. процесса кассы

Декларация плагина тех. процесса ничем не отличается от декларации плагина лояльности, за исключением названия элемента, который называется *TechProcessPlugin* в случае плагина тех. процесса.

Персистентные поля плагинов

Плагины товаров и оплат обладают также так называемыми персистентными полями. Это поля подобны расширенным атрибутам, которые прикрепляются к картам, или чеку, только отличие их состоит в том, что поля эти могут быть прикреплены к товару или оплате и значение их может отображаться при просмотре на сервере информации по созданному чеку, а также выгружаться в *ERP*. Поля такие декларируются не в разделе *Options*, а отдельно. С точки зрения разработчика нет никакой разницы между персистентным атрибутом и атрибутом обычным. Для него техники работы с ними не различаются. В *metainf.xml* персистентный атрибут задаётся элементом *PersistedField*. Имейте в виду, такие элементы поддерживаются только в плагинах товаров и оплат. Пример декларации персистентного атрибута в *metainf.xml* плагина товаров представлен в коде 16.

```
<PaymentPlugin paymentType="ELECTRONIC" localeKey="payment.name" id="foo.service.payment">
	<Description>Плагин оплат</Description>
	<Options>
	<String key="some.payment.plugin.property" localeKey="some.property.name"/>
	</Options>
	<PersistedField exportable="true" key="my.payment.plugin.authorization.code"
visible="true" localeKey="autorise.field.name" />
	<PersistedField exportable="true" key="my.payment.plugin.transaction.number"
visible="false" localeKey="slip.transaction.number"/>
</PaymentPlugin>
```

Код 16 - пример декларации персистентных полей в плагине оплат

Атрибуты элемента *PersistedField* приведены в таблице 9. При составлении ключа атрибута следует выбирать возможно более уникальные значения, чтобы не пересекаться с другими плагинами, которые тоже могут использовать такое имя ключа. Лучше всего для этого указывать в имени ключа идентификатор плагина. Пример негодного именования ключа персистентного атрибута - *authorization.code*, пример хорошего - *my.payment.plugin.attr.authorization.code*.

197136, Санкт-Петербург, Чкаловский пр., д.50, литера А, пом.5-Н, 2 этаж.
+7 (812) 331-22-55, crystals@crystals.ru
115432 Москва, пр. Андропова, 18, корп. 5, бизнес-парк Nagatino i-Land
+7 (495) 640-63-07, moscow@crystals.ru
8 (800) 222-22-51, www.crystals.ru

Таблица 9 - атрибуты элемента *PersistedField*

Атрибут	Назначение	Тип	Пример значения
exportable	Флаг, определяющий, следует ли выгружать данное поле в *ERP*. true если следует и false в противном случае.	*boolean*	*true*
key	Ключ данного атрибута. Аналогичен ключу расширенного атрибута, служит для получения значения. С этим же значением ключа поле выгружается в *ERP*.	*String*	*my.payment.plugin.authorization.code*
visible	Флаг, определяющий, следует ли отображать значение данного поля при просмотре чека на сервере в сеансе Опердня. В качестве поясняющей надписи значения используется значение атрибута, описываемого атрибутом *localeKey*.	*boolean*	*true*
localeKey	Ключ локализации, указывает на человекочитаемую строку в файле *strings.xml* плагина, откуда следует брать строку для иллюстрации значения атрибута при отображении его на сервере в сеансе Опердня.	*String*	*attribute.authorization.code*

17. Файлы с локализованными ресурсами (строками)

Интерфейс пользователя *Set Retail 10* поддерживает работу с двумя языками: русским и английским. Локализованные ресурсы хранятся в отдельных файлах для каждого языка. В зависимости от выбранного языка система выбирает из них значения строк по ключу. В плагине должны существовать два файла с локализованными строками: ***strings_ru.xml*** и ***strings_en.xml***. Значения из этих файлов используются как при работе плагина, при помощи интерфейса ***ResBundle***, так и при его настройке на сервере - атрибуты ***localeKey*** из файла ***metainf.xml***.
Пример файла ***PaymentPluginExample/src/main/resources/strings_ru.xml*** приведен в коде 17.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE properties SYSTEM "http://java.sun.com/dtd/properties.dtd">
<properties>
    <!-- For metainf -->
    <entry key="service.name">Электронный кошелёк (провайдер Foo)</entry>
    <entry key="url.option">Адрес процессинга</entry>
    <entry key="timeout.option">Таймаут в миллисекундах</entry>
    <entry key="cert.file.option">Файл сертификата</entry>
    <entry key="payment.name">Оплата электронным кошельком</entry>
    <entry key="some.property.name">Локализованное название настройки</entry>
    <entry key="autorise.field.name">Код авторизации</entry>

    <!-- For metainf -->
    <entry key="enter.sum.to.pay">Введите сумму к оплате</entry>
    <entry key="scanQR">Сканируйте QR код электронного кошелька</entry>
    <entry key="wallet.number">Номер кошелька</entry>
    <entry key="connecting">Выполняется запрос к процессингу</entry>
    <entry key="connection.error">Процессинг недоступен</entry>

    <!-- For slip -->
    <entry key="slip.header">С вашего кошелька успешно списано</entry>
    <entry key="slip.rub">руб.</entry>
    <entry key="slip.transaction.number">Номер транзакции:</entry>
</properties>
```

Код 17 - пример файла локализации

18. Требования и рекомендации

Кодировка исходного java кода должна быть *UTF-8*.

Все классы, которые можно использовать для внедрения в поля плагина (*dependency injection*) являются наследниками интерфейса ***Injectable***, также можно внедрить все классы, которые предоставляет интерфейс ***Facade***.

Важно! Инъекции осуществляются только в класс, помеченный аннотацией *@ru.crystals.pos.spi.annotation.POSPlugin*.

Асинхронное взаимодействие с графическим интерфейсом кассы выполняется через интерфейс *`UIForms`*.

Рекомендуется логировать различные действия плагина, касса будет записывать их в отдельный лог файл. Реализацию логера можно добавить следующим образом (код 18).

```
    @Inject
    private Logger log;
```

Код 18 - пример внедрения логера

CSI

197136, Санкт-Петербург, Чкаловский пр., д.50, литера А, пом.5-Н, 2 этаж.
+7 (812) 331-22-55, crystals@crystals.ru
115432 Москва, пр. Андропова, 18, корп. 5, бизнес-парк Nagatino i-Land
+7 (495) 640-63-07, moscow@crystals.ru
8 (800) 222-22-51, www.crystals.ru

Список библиотек, которые можно использовать:

```
'commons-lang:commons-lang:2.6',
'commons-codec:commons-codec:1.7',
'commons-collections:commons-collections:3.2.1',
'org.apache.commons:commons-compress:1.8.1',
'commons-io:commons-io:2.4',
'com.google.guava:guava:18.0',
'org.apache.httpcomponents:httpclient:4.5.2',
'org.apache.httpcomponents:httpmime:4.5.2',
'org.postgresql:postgresql:9.4-1201-jdbc41',
'junit:junit:4.12',
'org.mockito:mockito-core:1.10.19',
'com.thoughtworks.xstream:xstream:1.4.7',
'com.fasterxml.jackson.core:jackson-annotations:2.9.3',
'com.fasterxml.jackson.core:jackson-core:2.9.3',
'com.fasterxml.jackson.core:jackson-databind:2.9.3',
'com.fasterxml.jackson.module:jackson-module-jaxb-annotations:2.9.3'
```

Также этот список указан в файле *build.gradle* проектов с примерами плагинов.

197136, Санкт-Петербург, Чкаловский пр., д.50, литера А, пом.5-Н, 2 этаж.
+7 (812) 331-22-55, crystals@crystals.ru
115432 Москва, пр. Андропова, 18, корп. 5, бизнес-парк Nagatino i-Land
+7 (495) 640-63-07, moscow@crystals.ru
8 (800) 222-22-51, www.crystals.ru

Манифест *jar*, содержащей плагины

К *jar*-файлу, содержащему плагины, предъявляются определенные требования: он должен содержать файл манифеста (*/META-INF/MANIFEST.MF*), в котором должны находиться атрибуты, представленные в таблице 10.

Таблица 10 - атрибуты манифеста *jar* с плагином, обязательными к заполнению

Атрибут	**Обязательный**	**Назначение**	**Пример**
Build-Date	да	Дата сборки плагина. Должно быть датой в формате *dd.MM.yyyy HH:mm:ss*	11.04.2018 14:05:32
Implementation-Version	да	Версия плагина. Начиная с кассы версии 10.2.61.0 именно отсюда будет браться версия плагина.	1.0.0
Project	да	Название проекта, из которого собирается плагин.	*CardPluginExample*
Vendor-URL	нет	Адрес разработчика плагина в Интернете.	*http://crystals.ru*
Build-Machine	да	Имя машины, на которой был собран плагин.	*set-builder*
Vendor-Email	да, если не заполнено поле *Vendor-Phone*	Адрес электронной почты разработчика плагина.	*vendor@example.org*
Vendor-Phone	да, если не заполнено поле *Vendor-Email*	Контактный телефон разработчика плагина.	+7 (812) 331-22-55
Branch	да	Имя ветки в *VCS,* из которой производилась сборка плагина.	*plugin-release-v1.0*
Implementation-Vendor	да	Человекочитаемое название разработчика плагина.	*CSI*
Revision	да	Номер ревизии исходных кодов, из которых собран плагин.	5b65984

197136, Санкт-Петербург, Чкаловский пр., д.50, литера А, пом.5-Н, 2 этаж.
+7 (812) 331-22-55, crystals@crystals.ru
115432 Москва, пр. Андропова, 18, корп. 5, бизнес-парк Nagatino i-Land
+7 (495) 640-63-07, moscow@crystals.ru
8 (800) 222-22-51, www.crystals.ru

Пример содержимого файла *MANIFEST.MF* представлен в коде 19.

```
Manifest-Version: 1.0
Build-Date: 11.04.2018 13:42:10
Implementation-Version: 1.0
Project: CardPluginExample
Vendor-URL: https://www.crystals.ru/
Build-Machine: build-container4
Vendor-Email: crystals@crystals.ru
Vendor-Phone: +7 (812) 331-22-55
Branch: CORE-242
Implementation-Vendor: CSI
Revision: 5b65984
```

Код 19 - пример заполненного файла манифеста плагина

Пример получения необходимой для формирования манифеста информации приведен в исходных кодах примеров плагинов (*build.gradle)*, находящихся в директории *Examples SDK.*