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: 24.07.2018

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

1. История изменений

Дата	Описание изменений
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.
28.02.2018	Версия *API* 0.0.8. *PropertiesReader* - добавлено описание возможности сохранения параметров сервиса или плагина в Б.Д.
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.
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 *ru.crystals.pos.api.plugin.PaymentPlugin#createDailyReport* для совершения сверки итогов. В плагин оплат добавлен метод *ru.crystals.pos.api.plugin.PaymentPlugin#hasDailyReport*, который позволяет проверить, умеет ли плагин выполнять сверку итогов, не вызывая сам метод сверки.
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.
31.05.2018	Версия *API* 0.0.14. версия *Metainf* 0.0.9. Внесены изменения в методы `eventReceiptFiscalized`, `onRepeatSend`

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

	интерфейса плагина валидации акцизных марок. Добавлен плагин типа товаров и соответствующий пример. Добавлена визуальная форма ввода текста по шаблону.
05.06.2018	Версия *API* 0.0.15. Экранной форме *showInputScanNumberForm* добавлена возможность прослушивать событие “считана карта с магнитной полосой”.
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.
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.
20.06.2018	Версия *API* 0.0.18 Плагинам лояльности добавлена возможность добавить к результату расчета скидок список сообщений, которые следует отобразить кассиру. Длина каждого сообщения ограничена 141 символами.
02.07.2018	Версия *API* 0.0.19 Добавлена поддержка дисплея покупателя, обновлён пример плагина товаров для демонстрации работы с дисплеем покупателя.
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. *И*нтерфейсу *ru.crystals.pos.spi.receipt.Receipt* добавлен метод *getType* для получения типа чека, метод *getSaleReceipt* для получения чека продажи для чека возврата.
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.
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*.
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*.

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. Перечень терминов

В документе использованы следующие термины и сокращения:

- *CRM* - *Customer relationship management*;
- *ERP* - *Enterprise resource planning*;
- *MSR* - *Magnetic stripe reader*;
- *NFC* - *Near-field communication*;
- *Set API* - прикладной программный интерфейс, посредством которого плагин взаимодействует с программным комплексом *Set Retail 10*;
- *URL* - *Uniform Resource Identifier*;
- *VCS* - Version Control System - система контроля версий;
- Опердень (Операционный день) - рабочее время, в течение которого кассой выполняется обслуживание клиентов;
- РА - рекламная акция;
- ШК - штрих-код.

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

3. 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.

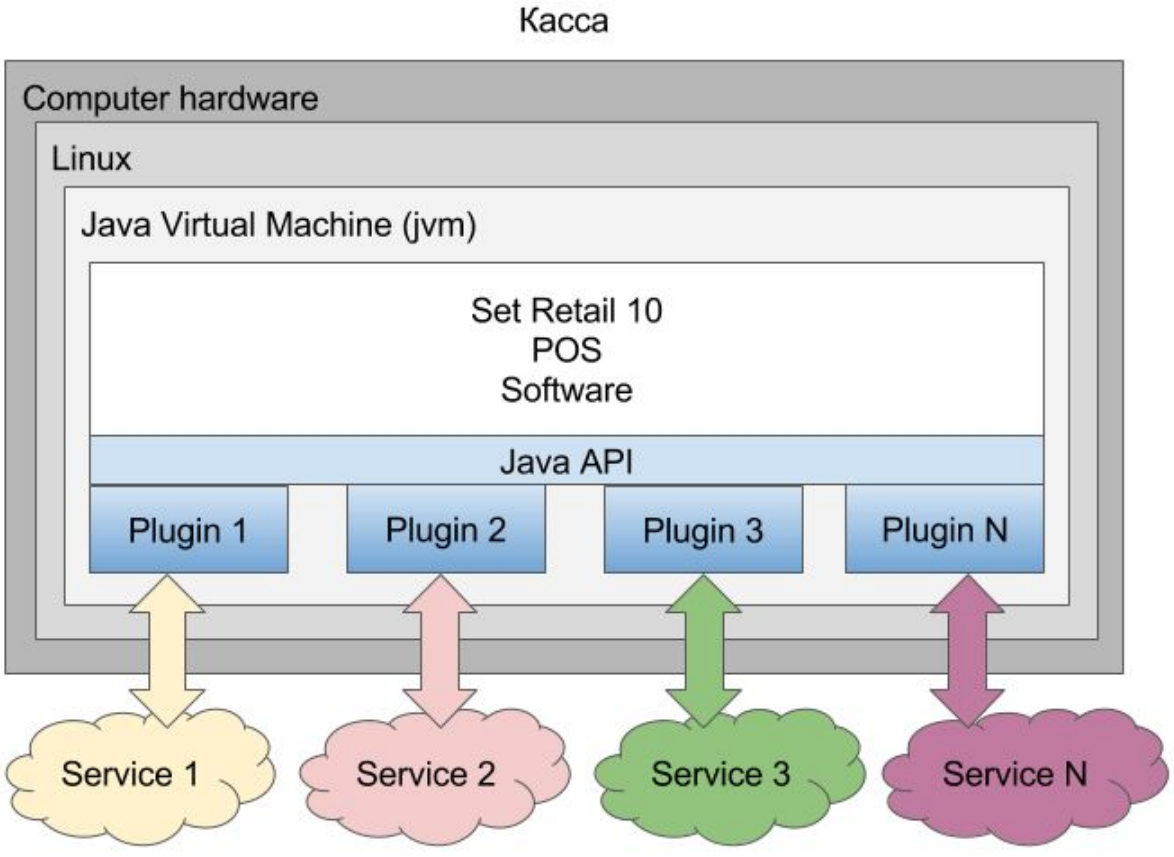

Рисунок 1 - взаимодействие внешних систем с кассой посредством плагинов *Set API*

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

4. Процесс формирования чека

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

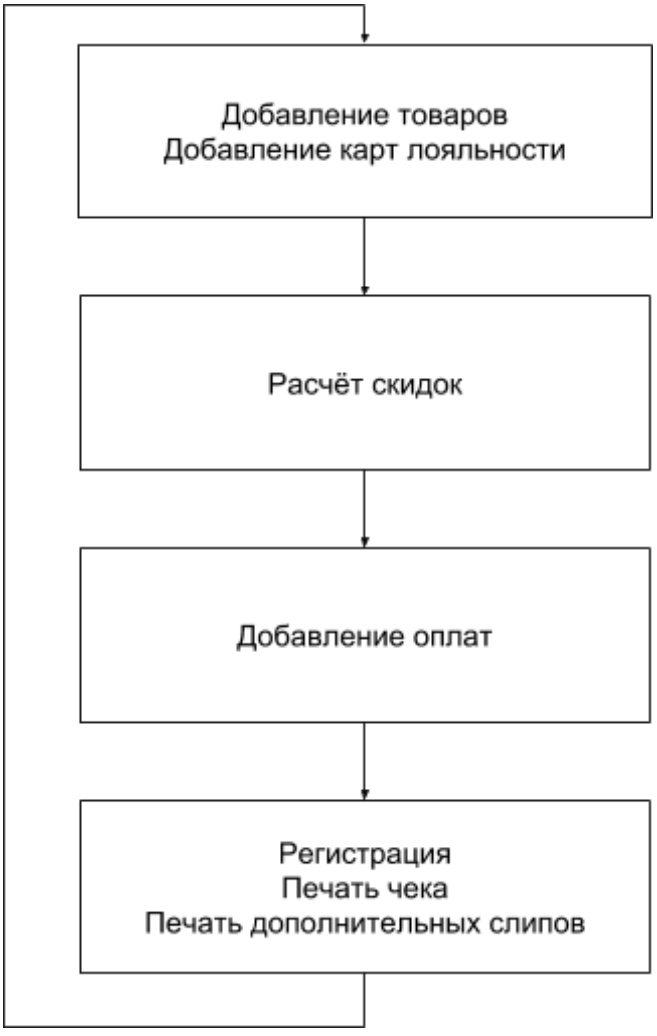

Рисунок 2 - процесс формирования чека

5. 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,

[1] далее в тексте “плагин” и “поставщик услуг лояльности” подразумевают логически одно и то же.

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

добавление заданий на печать купонов/рекламы в процессе фискализации чека) от активированных[2] 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):

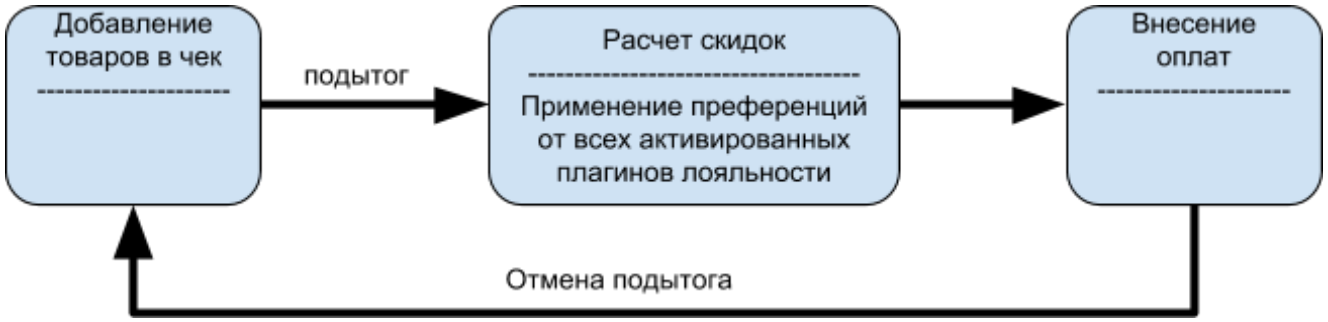

Рисунок 3 - автомат расчета скидок

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 X-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].

[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.

[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.

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

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

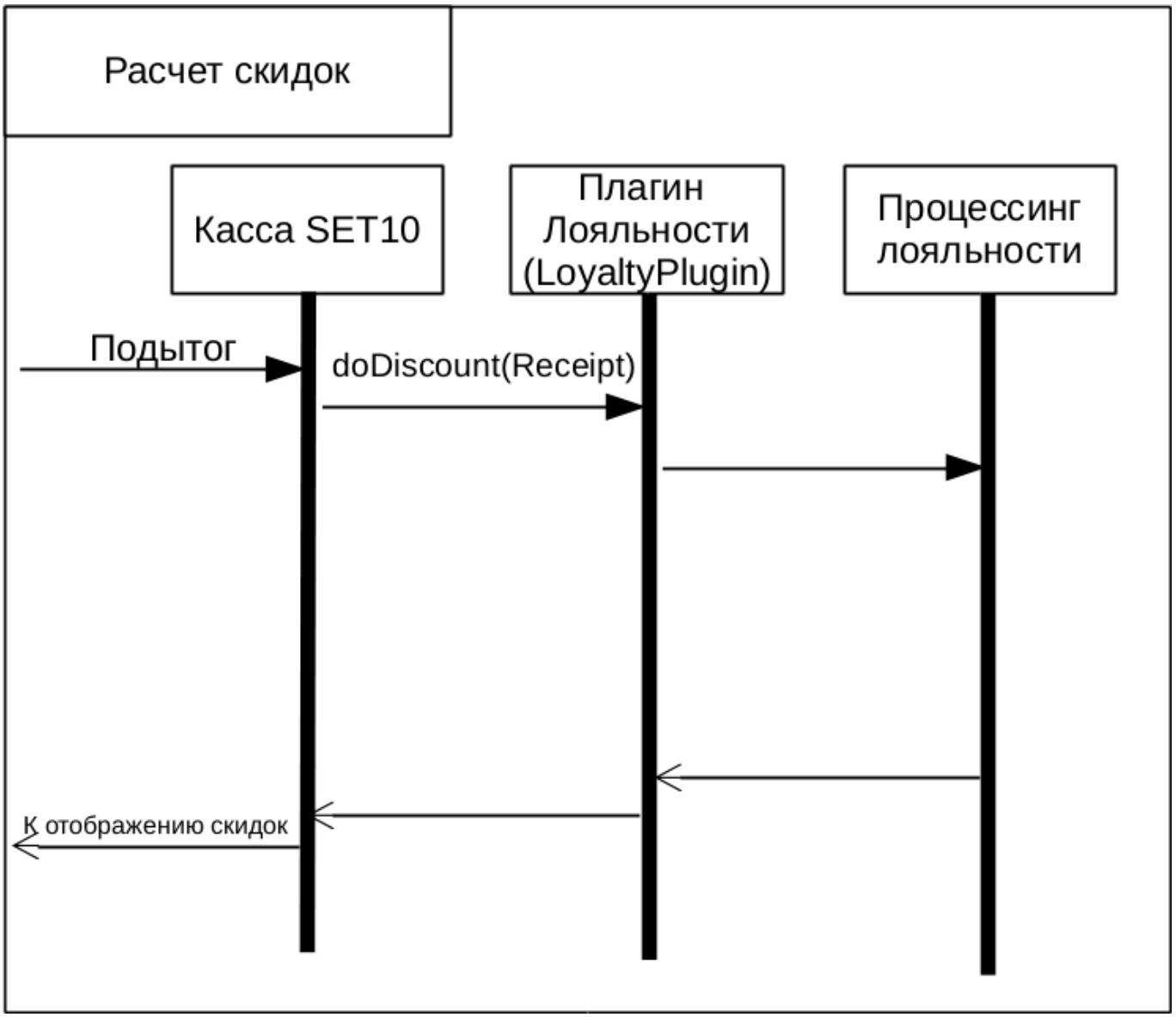

Рисунок 4 - последовательность взаимодействия кассы в плагином лояльности при расчете скидок

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

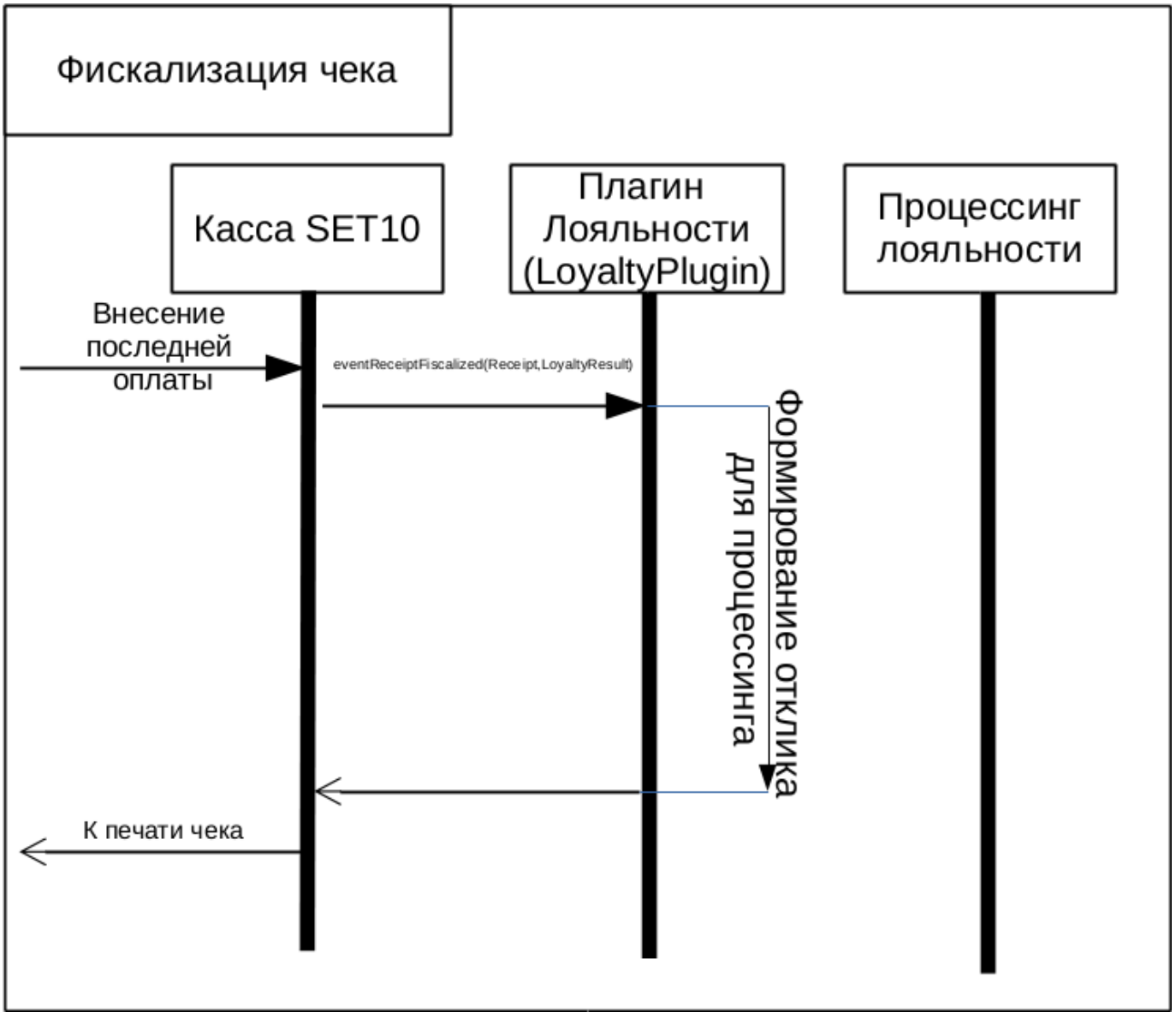

Рисунок 5 - последовательность взаимодействий кассы с плагином лояльности при фискализации чека

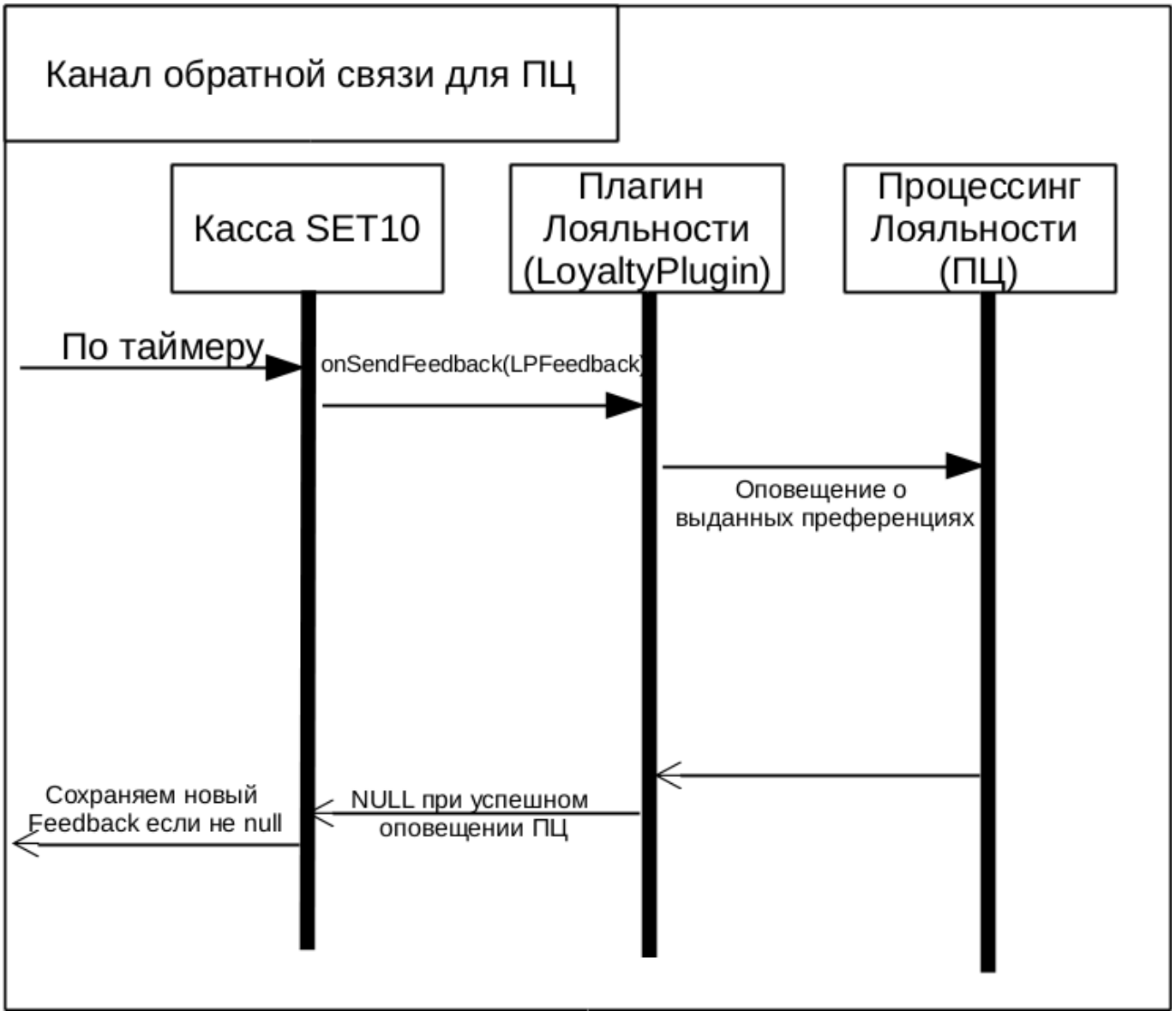

Рисунок 6 - последовательность взаимодействия кассы с плагином лояльности после фискализации чека

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

6. Создание плагина карт

Плагины карт предназначены для обеспечения возможности применения/использования сторонних[6] карт лояльности[7] в чеке (см. этап “добавление карт лояльности” в Процессе формирования чека). Далее по техпроцессу добавленные в чек карты могут быть использованы:

- плагинами лояльности (см. Создание плагина лояльности) для предоставления преференций в текущем чеке;
- [в случае дисконтной карты] для идентификации покупателя, совершившего эту покупку: данная информация может быть выгружена во внешнюю *ERP*-систему и там использована для различных *CRM*[8]-инициатив (для предоставления преференций в следующих покупках).

Сам процесс поиска и добавления карты лояльности в чек может быть запущен одним из следующих способов:

- прокатыванием магнитной полосы карты через считывающее устройство (*MSR*);
- сканированием ШК, нанесенным на карту;
- считыванием *NFC*-метки с карты;
- вводом номера карты вручную;
- вводом номера мобильного телефона покупателя (что забыл карту дома).

Далее эта информация последовательно передается зарегистрированным плагинам карт до тех пор, пока одна из них не найдет эту карту в своем процессинге либо не сообщит о том, что такой карты не существует.

Важно! С учетом того, что плагинов карт может быть несколько, в реализациях плагинов рекомендуется иметь возможность определения того, что искомая карта заведомо не “принадлежит” их процессингу (т.е., без обращения в сам процессинг (долго!)). Например, если процесс поиска запущен после считывания *NFC*-метки и процессинг не поддерживает обработку карт с *NFC*, то имеет смысл сразу же ответить, что данная карта не может быть найдена в процессинге - и после этого (быстро!) управление будет передано следующему плагину карт.

[6] под “сторонними” картами понимаются карты, что не зарегистрированы в *SET10*: с точки зрения *SET10* эти карты не существуют (не могут быть найдены в базах *SET10* ни по номерам, ни по диапазонам номеров).
[7] здесь и далее в тексте под “картой лояльности” понимается “дисконтная карта” либо “купон”.
[8] можно, например, на основе истории покупок конкретного клиента сделать ему персональные предложения (как это делает *SetMachine*) или просто начислить бонусы.

Упрощенный автомат добавления карт лояльности в чек можно представить следующей картинкой (рисунок 7):

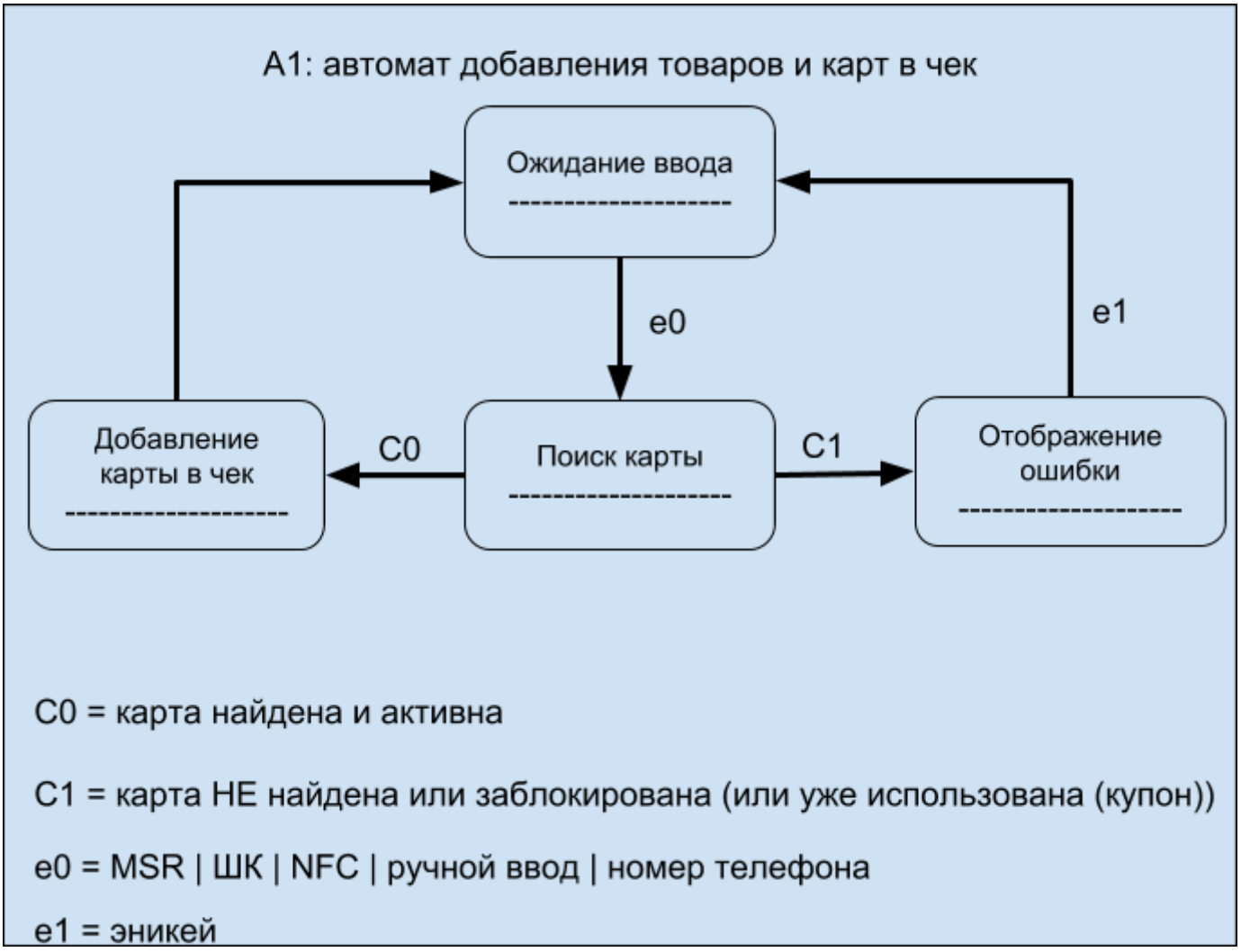

Рисунок 7 - упрощенная схема автомата добавления карт лояльности в чек

Пример реализации плагина карт поставляется с *SDK* - в проекте ***CardPluginExample***.

6.1. 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;

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

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 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[9] и пересчета скидок; кассиру при этом будет показано сообщение об этой ошибке;
    b. в случае критичной[10] 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.

[9] 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).
[10] 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

Упрощенно процесс списания бонусов можно представить следующей картинкой (рисунок 8):

Рисунок 8 - схема автомата списания бонусов

7. 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 взаимодействовать с пользовательским интерфейсом и оборудованием кассы на высоком уровне абстракции до завершения процесса.

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

Процесс оплаты может быть завершен успешно и 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.

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

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

Аннотация 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.* 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 - внедрение в плагин доступных вспомогательных классов

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

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 ***eventCanceled*** 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.

```
    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

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

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).

```
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 Б.Д..

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

8. 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, внедрение `UIForms, UIInputForms, UIPaymentForms` кассой не выполняется.

9. Создание плагина типа товара

Продажа товаров и услуг на кассе *Set10* 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);`

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

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.

10. 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 8. 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 8 - 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

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 параметров типа *BinaryFile*. Следует отметить, это больше косметический функционал и плагин 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 (код 9):

```
<?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 9 - 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 1.

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

Таблица 1 - описание атрибутов элемента *SetIntegration*

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 2.

Атрибут	Обязательный	Описание	Пример
serviceType	да	Тип сервиса, реализуемый плагинами. 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*

Таблица 2 - атрибуты элемента *ExternalService*

Внутри элемента *ExternalService* может находиться элемент *Description*, который несёт в себе человекочитаемое описание внешней системы. В настоящий момент оно используется исключительно

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

как средство справочной информации и ограничено 3000 символами. Это поле позволительно оставить 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 3. Если внешняя система не имеет настроек, секция с ними не декларируется.

Элемент	Тип параметра	Описание
Boolean	булевский	Позволяет задать булевский параметр, возможные значение - *true*/*false*
BinaryFile	массив байт	Позволяет использовать произвольный файл в качестве параметра конфигурации плагина
Integer	целочисленный	Задаёт целочисленный параметр.
String	строковый	Задаёт строковый параметр.
URL	строковый, должен удовлетворять валидному *URL*	Задаёт 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.

Таблица 3 - типы элементов для задания параметров конфигурации внешней системы или плагина

Все элементы имеют два обязательных атрибута и один опциональный, представленные в таблице 4, кроме элемента типа *BinaryFile*, который не имеет атрибута *default-value.* Также элементы могут иметь специфичные только для себя значения, которые представлены в таблице 5. Не все типы параметров имеют специфичные атрибуты. Такие типы параметров в таблице 5 не перечислены.

Атрибут	Обязательный	Описание	Пример

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

key	да	Уникальный ключ, который идентифицирует параметр. По нему значение параметра может быть получено плагином.	*my.param.id*
localeKey	да	Ключ в файле локализации, указывающий на человекочитаемое название параметра.	*string.payment.success*
default-value	нет, не содержится в элементе типа *BinaryFile*	Значение по умолчанию параметра в случае, когда он впервые настраивается на сервере.	42

Таблица 4 - атрибуты конфигурационного параметра внешней системы или плагина

Тип параметра	**Атрибут**	**Обязательный**	**Описание**
Integer	*maxValue*	нет	Максимальное значение, которое может принимать этот параметр.
	minValue	нет	Минимальное значение, которое может принимать этот параметр. Должно быть меньше максимального значения.
String	*masked*	нет	Булевский параметр, определяющий, следует ли скрывать от пользователя вводимые символы, заменяя их на «*» на визуализации.

Таблица 5 - специфичные атрибуты определенных типов параметров

С точки зрения серверного *UI,* параметры внешней системы отображаются как на рисунке 9.

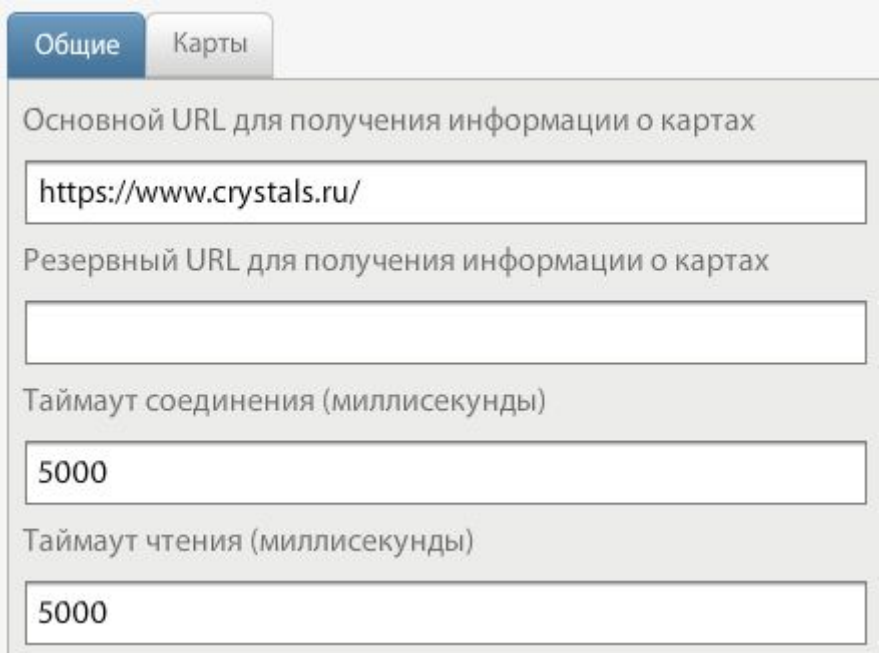

Рисунок 9 - визуализация настроек внешней системы на сервере

Рисунок 9 соответствует фрагменту кода из *manifest.xml*, изображенному в коде 10.

```
        <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 10 - фрагмент декларации параметров внешней системы, визуализация которого проиллюстрирована на рисунке 9

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 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

```
<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 11 - 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 12. Возможные значения атрибута *paymentType* перечислены в таблице 6.

```
       <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="authorization.code" visible="true"
localeKey="autorise.field.name" />
              <PersistedField exportable="true" key="transaction.number" visible="false"
```

```
localeKey="slip.transaction.number"/>
        </PaymentPlugin>
```

Код 12 - пример декларации плагина оплат в *metainf.xml*

Значение	Описание
CASH	Плагин принимает платежи наличными
ELECTRONIC	Плагин принимает платежи любым электронным суррогатом
PREPAY	Плагин принимает предоплату
POSTPAY	Плагин принимает постоплату
OTHER	Плагин принимает вид платежа, не вошедший в вышеперечисленное

Таблица 6 - допустимые значения атрибута *paymentType* элемента *PaymentPlugin*

Пример выгрузки чека с оплатами в ERP изображен в коде 13. 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="Оплата
электронным кошельком">
                <plugin-property key="authorization.code" value="385281233141592"/>
                <plugin-property key="transaction.number" value="01234567890"/>
                </payment></payments></purchase></purchases>
```

Код 13 - 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.

11. 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*** (код 14):

```
<?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 14 - пример файла локализации

12. 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*.

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

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 15):

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

Код 15 - пример внедрения логера

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',
'joda-time:joda-time:2.5',
'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 7.

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.	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

Таблица 7 - атрибуты манифеста *jar* с плагином, обязательными к заполнению

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 *METAINF.MF* представлен в коде 16.

```
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 16 - пример заполненного файла манифеста плагина

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